Neural Implicit Reduced Fluid Simulation

YUANYUAN TAO, McGill University, Canada and Huawei Technologies Canada, Canada
IVAN PUHACHOV, Université de Montréal, Canada
DEREK NOWROUZEZAHRAI, McGill University, Canada
PAUL KRY, McGill University, Canada

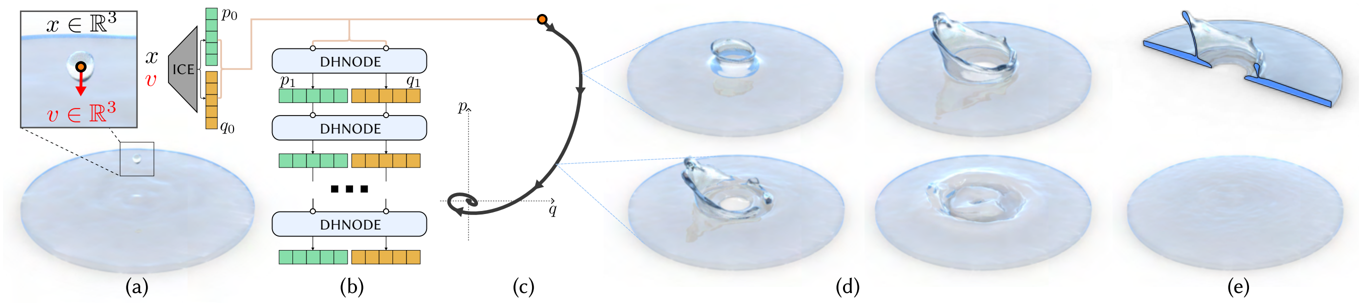


Fig. 1. Neural implicit reduced fluid simulation (NIRFS) performs high-fidelity fluid simulation orders of magnitude faster than conventional full-order methods. An initial condition encoder (ICE) maps a novel physical initial condition, e.g., droplet position $\boldsymbol{x}$ and velocity $\boldsymbol{v}$, to a latent initial condition $(\boldsymbol{q}_0, \boldsymbol{p}_0)$ (a). In a physically structured latent space, a damped Hamiltonian neural ODE (DHNODE) is integrated (b), generating a latent fluid trajectory (c). From the latent geometry vectors $\boldsymbol{q}$ in the trajectory, an implicit neural representation (INR) decoder constructs fluid geometries with fine details (a and d) and high-frequency structures (e top). The correct steady state is stably reached in the end (e bottom).

High-fidelity simulation of fluid dynamics is challenging because of the high dimensional state data needed to capture fine details and the large computational cost associated with advancing the system in time. We present neural implicit reduced fluid simulation (NIRFS), a reduced fluid simulation technique that combines an implicit neural representation of fluid shapes and a neural ordinary differential equation to model the dynamics of fluid in the reduced latent space. The latent trajectories are computed at very little cost in comparison to simulations for training, while preserving fine physical details. We show that this approach can work well, capturing the shapes and dynamics involved in a variety of scenarios with constrained initial conditions, e.g., droplet-droplet collisions, crown splashes, and fluid slosh in a container. In each scenario, we learn the latent implicit representation of fluid shapes with a deep-network signed distance function, as well as the energy function and parameters of a damped Hamiltonian system, which helps guarantee desirable properties of the latent dynamics. To ensure that latent shape representations form smooth and physically meaningful trajectories, we simultaneously learn the latent representation and dynamics. We evaluate novel simulations for conservation of volume and momentum conservation, discuss design decisions, and demonstrate an application of our method to fluid control.

Authors' Contact Information: Yuanyuan Tao, McGill University, Canada and Huawei Technologies Canada, Canada, yuanyuan.tao@mail.mcgill.ca; Ivan Puhachov, Université de Montréal, Canada, ivan.puhachov@umontreal.ca; Derek Nowrouzezahrai, McGill University, Canada, derek@cim.mcgill.ca; Paul Kry, McGill University, Canada, kry@cs.mcgill.ca.

SA Conference Papers '24, December 03–06, 2024, Tokyo, Japan
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-1131-2/24/12
https://doi.org/10.1145/3680528.3687628

CCS Concepts: • **Computing methodologies → Computer graphics**; **Modeling and simulation**; **Machine learning**.

Additional Key Words and Phrases: fluid simulation, reduced order modelling, dynamics, latent space, implicit neural representation, deep learning

ACM Reference Format:
Yuanyuan Tao, Ivan Puhachov, Derek Nowrouzezahrai, and Paul Kry. 2024. Neural Implicit Reduced Fluid Simulation. In *SIGGRAPH Asia 2024 Conference Papers (SA Conference Papers '24), December 03–06, 2024, Tokyo, Japan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3680528.3687628

1 INTRODUCTION

Reduced-order neural physics offers a change of perspective to physical simulations that would otherwise be based on high-dimensional discretized partial differential equations (PDEs). On one hand, for an n-dimensional problem, spatial discretization transforms a continuous spatial field, e.g., velocity, pressure, and density, to a P-resolution mesh, grid, or point cloud, yielding a $(P \cdot n)$-dimensional matrix. Memory and computation cost become intractable as P grows large [Museth 2013]. In fluid simulations, one can have undesirably small time steps imposed for stability, or errors from numerical dissipation [Fedkiw et al. 2001], viscosity [Roache 1998], and diffusion [Lantz 1971]. On the other hand, within a subspace such as a fluid phenomenon, an implicit neural representation (INR) can represent continuous spatial fields and connect to a latent space that permits dynamics in potentially simpler forms than the PDEs. An INR is parameterized with a latent vector and permits continuous queries for any point within the field. INRs have been proven invaluable to visual computing for their adaptive and differentiable data representation, low memory footprint, and arbitrary resolution.

The latent spaces of INRs, however, have lacked structure and interpretability. While black box neural networks, such as long short-term memories (LSTMs) [Vlachas et al. 2018; Wiewel et al. 2019] and graph neural networks (GNNs) [Pfaff et al. 2021], can be used to agnostically navigate latent spaces, latent space exploration, extrapolation, and even interpolation present challenges [Liu et al. 2022; Maesumi et al. 2023]. Nevertheless, such challenges are not inherent to latent spaces but only stem from the failure to construct latent spaces with desired structures through training.

In the end, latent spaces are what they are shaped to be. This plasticity enables latent space physics to take alternative forms to the fundamental PDEs, such as Hamiltonian formulations. Hamiltonian mechanics forms the foundation of theoretical mechanics. Instead of using vectorial properties of individual particles as Newtonian mechanics, theoretical mechanics characterizes a system using global scalar properties of motion, such as kinetic energy and potential energy. This global view coincides with INRs that encode a system with a single latent vector. Additionally, the Hamiltonian structure offers the explicit characterization of conservation laws, geometric insights through symplectic structures, and the convenience of canonical transformations and Poisson brackets. Hamiltonian counterparts or mathematical equivalents of many fundamental PDEs have been derived. By adopting the Hamiltonian structure, latent space physics can potentially preserve the nature of real physics while taking simpler forms than the PDEs.

In this work, we advance reduced-order modelling and propose to simulate fluid dynamics in smooth, coherent, and structured Hamiltonian latent spaces for specific scenarios, i.e. parameter spaces and boundary conditions. Consolidating INRs and neural ordinary differential equations (ODEs), our neural implicit reduced fluid simulation (NIRFS) features full spatiotemporal differentiability, low memory footprint, and arbitrary resolution. While maintaining high fidelity, NIRFS achieves tens of thousands times speedup, empowering real-time surrogates of traditional methods. As a fundamental framework for latent space physics, NIRFS holds the potential for generalization to other physical regimes.

We focus on simulating directly the dynamics of fluid geometries without evolving velocity and pressure fields. The geometries are represented as signed distance functions (SDFs) through an INR. The evolution of fluid geometries is reduced to latent damped Hamiltonian trajectories and simulated in the latent space of the INR. An initial condition encoder (ICE), a latent damped Hamiltonian neural ODE (DHNODE), and an INR decoder are jointly trained to shape a Hamiltonian latent space. In many cases, e.g., generative applications and control tasks, where simulations for specific physical conditions are not necessary, the ICE can be omitted. We constrain the latent steady state for realistic temporal extrapolation. We evaluate our method by learning and simulating three fluid phenomena and examine the conservation of mass and momentum. We demonstrate our methods on the generation of novel simulations and an inverse design problem involving a fluid control task.

In summary, our contributions include the following:

- The first solution to the major challenge of learning a smooth, coherent, and structured latent space;
- Latent space physics simulation of a drastically simpler alternative formulation to the fundamental PDEs;
- Animating INRs as a simulation primitive;
- An extremely cheap surrogate for differentiable simulation that is useful for tasks like fluid control;
- Insights into latent space landscape, e.g., convexity and latent dynamics stiffness.

2 RELATED WORK

We first present previous and recent advances in fluid simulation that exploit data, reduction, or machine learning to accelerate computation or produce reduced models. Then, we discuss relevant work on topics directly related to the techniques we employ for neural implicit fluid simulation, specifically, implicit neural representations, latent space dynamics, and Hamiltonian mechanics.

2.1 Reduced and Data-driven Simulation

While we may formulate the simulation of physical systems in a general manner that allows its degrees of freedom to take on arbitrary values, many states correspond to high energy configurations or are states that are unlikely to arise under typical scenarios of user interaction and boundary conditions. Using prescribed interaction or user input have proven useful in guiding data collection of elastic solids [Barbič and James 2005] and fluids [Treuille et al. 2006], where principal component analysis can then provide a reduced set of degrees of freedom. In the absence of data, energy can also guide the sampling and construction of reduced models [Sharp et al. 2023]. Our work focuses on the case where a set of initial conditions identifies the subset of configuration space that we wish to model. We run simulations at sampled initial conditions to produce data.

Indeed, within many scenarios, it is possible to comprehensively collect and compress trajectory data under expected user interaction [James and Fatahalian 2003], and likewise refine as new states are discovered [Stanton et al. 2014]. In more recent work, machine learning techniques have been applied to the compression problem, for instance, work of Kim et al. [2019] which proposes a generative model using a convolutional neural network (CNN) to produce the velocity fields of a reduced family of fluid simulations. Related to this is the LSTM-based approach of Wiewel et al. [2019], which does not simply generate a plausible trajectory but aims to model the latent space dynamics.

In contrast to building reduced models, data-driven approaches show promise in accelerating expensive computations of full (unreduced) systems. For example, Tompson et al. [2017] train a convolutional neural network to solve the pressure projection step of an incompressible fluid simulation. Similarly, in the context of smoothed-particle hydrodynamics (SPH) simulations, Ladický et al. [2015] propose using a regression forest to reduce the cost of computing acceleration vectors of particles.

2.2 Implicit Neural Representations

Though broadly adopted in volumetric data representations, spatial discretization comes with drawbacks. Point-based methods lack connectivity and fail to represent topology. Mesh-based methods face challenges in mesh quality for complex geometries and conditions, and in mesh adaptivity for dynamic processes [Pfaff et al. 2021]. Voxel-based methods are often restricted to low resolutions by computation and memory requirements.

Without explicit parameterizations, INRs are neural networks that, through network weights and latent vectors, encode arbitrary data, such as 3D geometries [Chen and Zhang 2019; Mescheder et al. 2019; Park et al. 2019], radiance fields [Mildenhall et al. 2021], and physical fields [Chen et al. 2023a,b; Zehnder et al. 2021]. For example, an MLP representing SDFs takes as input a latent vector and an arbitrary query point, and outputs the signed distance [Park et al. 2019]. As such, the neural networks transcend spatial discretization and empower continuously differentiable neural fields. By adaptively allocating network weights based on signal complexity, INRs avoid aggressive scaling of memory and computation with resolution [Dupont et al. 2022; Takikawa et al. 2021]. Advantages of INRs have been exploited by many applications, including geometry processing [Sharp and Jacobson 2022; Yang et al. 2021], scene deformation [Park et al. 2021a,b], and 3D scene and geometry generation [Chan et al. 2022; Schwarz et al. 2020; Wu and Zheng 2022]. Our method not only leverages the compact but expressive and flexible encoding by INRs, but also shapes the encoding and latent space based on the intrinsic dynamics in fluid features. In this way, we generate new fluid processes from the latent space.

2.3 Latent Space Dynamics

To capture dynamics in a primal (data) space is to navigate the latent space of an INR and to observe latent space dynamics. Smooth primal dynamics does not necessarily parallel smooth latent dynamics. This inconsistency in smoothness causes friction in data interpolation and extrapolation from a latent space. Methods like variational autoencoder [Kingma and Welling 2013] and Lipschitz regularization [Liu et al. 2022] shape a smooth latent space, so that smooth latent dynamics yields smooth primal dynamics. Alternatively, an arbitrary latent space has been re-parameterized to an exploration space that preserves primal geodesics [Maesumi et al. 2023].

Agnostic to latent space smoothness or dynamics, black boxes, including LSTMs [Vlachas et al. 2018; Wiewel et al. 2019], GNNs [Pfaff et al. 2021], and time-lagged autoencoders [Wehmeyer and Noé 2018], have been employed to jointly learn latent dynamics with INRs. Otherwise, a structure is enforced to some extent on latent dynamics and thus on the latent space. A neural ODE parameterizes an ODE with a neural network and time steps the network with an ODE solver, continuously transforming states [Chen et al. 2018]. Consequently, a neural ODE defines a continuous vector field and outputs continuous state trajectories. Fully parameterized neural ODEs are agnostic to the dynamics but could smooth a latent space if training is properly handled. Linear Koopman models [Lusch et al. 2018], polynomials [Champion et al. 2019], Hamiltonian neural networks (HNNs) [Greydanus et al. 2019], and partially parameterized neural ODEs [Massaroli et al. 2020] prescribe general structures in the dynamics. With less flexibility, reduced equations of motion specify latent dynamics [Fulton et al. 2019]. Balancing structure and flexibility, our method constructs a smooth, coherent, and structured latent space with a partially parameterized neural ODE to explicitly simulate latent space fluid dynamics.

2.4 Hamiltonian Mechanics

The Hamiltonian formulation of Euler equations has been extensively studied [Arnold 2014; Arnold and Khesin 2008; Camassa et al. 2014; Ebin and Marsden 1970; Kolev 2007; Miles 1977; Morrison 1998; Olver 1982]. With zero viscosity and thermal conductivity, the Euler equations are a special case of the Navier-Stokes equations, describing the motion of an inviscid and incompressible fluid. Due to viscous dissipation, a viscous fluid, on the other hand, is generally considered a non-Hamiltonian system and the resulting dynamics are often not amenable to a straightforward Hamiltonian formulation. However, the Hamiltonian and port-Hamiltonian formulations of Navier-Stokes equations have been derived [Altmann and Schulze 2017; Haine and Matignon 2021; Jones 2014; Kaufman and Morrison 1982; Morrison 1984; Oseledets 1989; Sanders et al. 2023].

The Hamiltonian structure has been incorporated into neural networks to regress simple Hamiltonian systems [Greydanus et al. 2019] and to perform classification and function approximation tasks [Massaroli et al. 2020]. Our method learns latent Hamiltonian analogs of fluid dynamics.

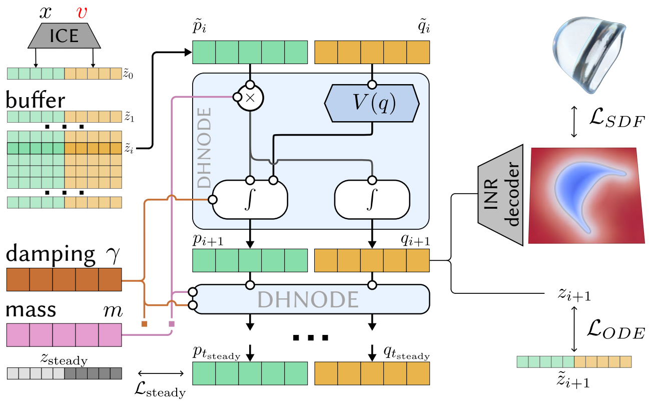


Fig. 2. An ICE, a DHNODE, and an INR decoder are jointly trained. To achieve training stability and convergence, we keep latent states for each frame in training simulations in a buffer $\tilde{z}$. The ICE is trained to map physical initial conditions of each full-length simulation to latent initial conditions in the buffer $\tilde{z}_0$. For each scenario, the DHNODE learns an energy function, a mass matrix, and a damping matrix. The DHNODE is integrated from uniformly sampled latent states until (after) t_{steady}. Over a short segment at the beginning of each integrated trajectory, $\mathcal{L}_{ODE}$ measures the difference between the rollouts and states in the buffer, and $\mathcal{L}_{SDF}$ measures the differencz'e between the decoded and ground truth SDF geometries. For the states rolled out at or beyond t_{steady}, $\mathcal{L}_{\text{steady}}$ measures the difference between the rolled out states and the determined latent steady states.

3 MODEL

NIRFS consists of three parts: an initial condition encoder (ICE) for physical parameters, a damped Hamiltonian neural ODE (DHNODE) for latent dynamics, and an INR decoder for geometry representation. The ICE allows NIRFS to simulate from physical inputs by mapping initial position/dimension and velocity to latent initial geometry and geometry momentum with separate MLPs. Optionally, the ICE can be omitted when physical inputs are not required by tasks. The DHNODE provides a means of generating physically correct shape dynamics in the latent space. The INR decoder maps latent space to shape space. Figure 1 shows an overview of how the components work together to produce a fluid simulation, with more details in Figure 2. The state at each time step is described by a latent geometry vector and a latent geometry momentum vector.

3.1 Reduced Representation of Fluid Shapes

We represent fluid geometries as SDFs. Given a location $\boldsymbol{x} \in \mathbb{R}^3$, the function $SDF(\boldsymbol{x})$ provides the signed distance to the surface of a closed geometry, where negative values indicate points inside the geometry. The zero isosurface of the SDF defines the surface geometry.

Inspired by DeepSDF [Park et al. 2019], we encode fluid geometries as zero-level sets of an MLP that regresses a collection of SDFs from point samples. With the MLP conditioned on a latent geometry vector for each fluid geometry, we employ the MLP as an INR decoder to approximate signed distances of a fluid geometry from the latent vector. That is, given decoder $\mathcal{D}$, we obtain signed distance approximation $\mathcal{D}(\boldsymbol{q}, \boldsymbol{x})$ at a query coordinate $\boldsymbol{x} \in \mathbb{R}^3$ for the shape encoded with latent geometry vector $\boldsymbol{q} \in \mathbb{R}^d$. We can compute an explicit representation of the full surface, for instance, with multiple queries and marching cubes.

We enhance our decoder to accommodate NIRFS:

Gaussian embedding. To encourage the latent space to be locally smooth and better tolerate ODE integration error, we take inspiration from VAE and use a Gaussian embedding. We structure the latent space as a cluster of distributions and train the decoder to learn a local probabilistic mapping. At training time, for a fluid geometry with associated latent geometry vector $\boldsymbol{q}$ the decoder is provided input $\boldsymbol{q}'$ drawn from Gaussian $\mathcal{N}(\boldsymbol{q}, \boldsymbol{\sigma}^2)$ with mean $\boldsymbol{q}$ and a fixed variance $\boldsymbol{\sigma}$.

Fourier features. NIRFS must accurately reconstruct shapes from the latent space with high fidelity. However, fluid geometries can have high spatial frequencies, for instance the crown splash example in Fig. 1 – these shapes are challenging for MLPs to learn. Fourier feature mapping of input coordinates is one of the most common techniques to enable an MLP to learn high frequency functions [Tancik et al. 2020]. We therefore use Fourier features to help the decoder learn high-frequency fluid geometries in this example:

$$\boldsymbol{x}'(\boldsymbol{x}) = \left[\cos(2\pi \boldsymbol{b}_1^\top \boldsymbol{x}), \sin(2\pi \boldsymbol{b}_1^\top \boldsymbol{x}), \cdots \cos(2\pi \boldsymbol{b}_m^\top \boldsymbol{x}), \sin(2\pi \boldsymbol{b}_m^\top \boldsymbol{x})\right]^\top .$$

Here, $\boldsymbol{b}_m$ is a frequency vector sampled from a Gaussian distribution. As indicated by Tancik et al. [2020], the variance of the distribution needs to be tuned to fit the data. $\boldsymbol{x}'$ is then input to the decoder in place of $\boldsymbol{x}$.

Importance sampling. Because some high-frequency structures occupy a limited volume and surface area (Fig. 1e top), sampling SDF training points with equal importance for all structures often results in imbalanced data, making the high-frequency structures difficult to learn (Fig. 3a). To balance the samples, we isolate high-frequency structures so that we can assign them greater weights in sampling. First, we find points inside the geometry on the medial surface that have a small medial radius less than a threshold r, which we set to be 1% of the domain size. These medial points will be located near sharp features or centered within thin sheets of fluid. Samples fall inside these medial spheres are identified as important for recreating high-frequency details and given a greater weight $w_{\text{importance}}$ for drawing samples during training (Sec. 4.1). This sampling approach ensures that the decoder correctly reconstructs these thin and high frequency details (Fig. 1, 8, 10, and 9).

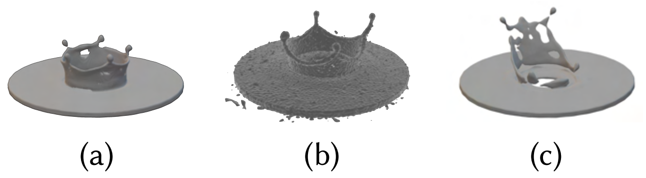


Fig. 3. Geometric defects can arise when an INR (a) is trained without importance sampling, (b) uses excessive variance in Fourier feature mapping, or (c) uses insufficient variance in Fourier feature mapping.

3.2 Latent Damped Hamiltonian Neural ODEs

We reduce the computation of a PDE that locally describes the dynamics for a fluid phenomenon in physical space to a second-order neural ODE that globally describes the dynamics of that phenomenon in latent space. Learning a latent neural ODE not only models the latent dynamics, but also regularizes the decoder and latent space, resulting in a smooth, coherent, and well structured latent space. We use $\boldsymbol{z}(t) = (\boldsymbol{q}(t), \boldsymbol{p}(t))$ to denote the latent phase space trajectory, where $\boldsymbol{q}$ is a latent geometry vector, and $\boldsymbol{p}$ is a latent geometry momentum vector.

Instead of deriving a Hamiltonian formulation explicitly from the Navier-Stokes equations, we aim to learn a latent Hamiltonian formulation of fluid geometry dynamics. We embed a Hamiltonian structure in the neural ODE:

$$\frac{d\boldsymbol{q}}{dt} = \boldsymbol{m}^{-1}\boldsymbol{p}, \tag{1}$$

$$\frac{d\boldsymbol{p}}{dt} = -\boldsymbol{\gamma}\boldsymbol{m}^{-1}\boldsymbol{p} - \frac{\partial V(\boldsymbol{q})}{\partial \boldsymbol{q}}, \tag{2}$$

where $V(\boldsymbol{q})$ is a latent potential energy function that is parameterized by an MLP, $\boldsymbol{m}$ is a diagonal latent mass matrix, and $\boldsymbol{\gamma}$ is a diagonal latent damping coefficient matrix. The energy function defines latent space landscape and drives latent dynamics (Sec. 4.4 and 5.1). Including a linear damping term is necessary to capture energy dissipation and to produce simulations that, in the long term, go to a steady state in the latent space. Damping also generally encourages stable integration of the latent dynamics. A single $V(\boldsymbol{q})$, $\boldsymbol{m}$, and $\boldsymbol{\gamma}$ are learned for each scenario.

We exploit the latent space plasticity, using Hamiltonian dynamics for Navier-Stokes systems along with further simplifications, including Cartesian latent space, linear damping/dissipation, and diagonal damping and mass matrices. While these simplifications may not generally be true for fluid systems, we note that the latent space effectively adapts to these simplifications, which we believe are acceptable for the purpose of animating fluids such as those in our results.

Steady state. Damping merely ensures the presence of steady states in the latent space. Physical and latent steady states should be reached synchronously. In some cases, e.g., binary droplet collisions, training data terminate before reaching a steady state. While latent

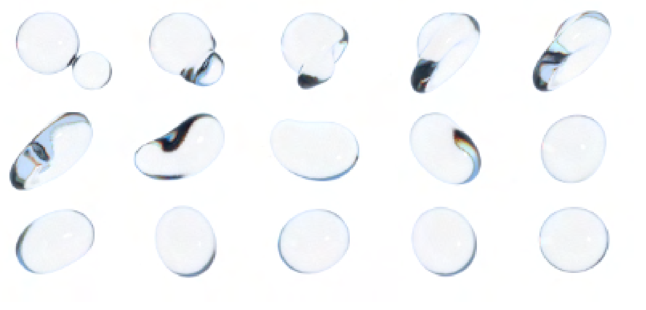


Fig. 4. A novel simulation of binary droplet collision is generated by NIRFS. A latent initial condition is randomly sampled from its domain. Two droplets of new sizes collide at new angle and speed. The novel simulation reaches a plausible steady state beyond training time scale.

dynamics can be learned to match the training data, the learned latent space steady state shape can correspond to physically incorrect geometry (e.g., a non-spherical droplet). In other cases, e.g., crown splash, we have training data that end with a steady state. Again, latent dynamics can be learned to match the training data, but without the guarantee that the latent steady state is synchronized with that of the training data. In this case, the simulated latent space trajectory will continue to evolve past the end of the training data with reconstructed shapes that demonstrate non-physical motion.

To resolve this issue, we constrain latent dynamics to chosen latent steady states at time when physical steady states are reached, i.e., $z(t_{\text{steady}}) = (\boldsymbol{q}_{\text{steady}}, \mathbf{0})$. Constraining only the latent state suffices in most cases as a training set should contain near-steady state data that defines a neighbourhood around the steady state.

4 TRAINING

We jointly train an ICE, a DHNODE, and an INR decoder to simultaneously shape a latent space and explore latent dynamics. However, due to the joint optimization, every update may result in an substantial variation that hampers training convergence. We propose an approach that ensures training stability (Fig. 2).

4.1 Data Preparation

We precompute a set of simulations that demonstrate the varying dynamics under a given scenario. From the simulated fluid geometries, we prepare ground truth SDF samples that each contains a spatial coordinate $\boldsymbol{x}$ and a signed distance value $SDF(\boldsymbol{x})$ for training.

Simulation sets for three fluid phenomena were generated - binary droplet collision (Fig. 4), crown splash (Fig. 1), and slosh in a cylinder (Fig. 5). High fidelity ground truth simulations were performed with Blender [Blender Foundation 2021] FLIP Fluids add-on [Guy and Fassbaender 2022]. Each binary droplet collision simulation took approximately 8 hours to produce (250×250×150 grid, approximately 873 000 particles); each crown splash, approx. 17h (350×350×200 grid, approx. 2 097 150 particles); each slosh in a cylinder, approx. 6h (92×92×100 grid, approx. 1 075 600 particles). Frames are sampled from the simulations at high rates. The training set contains, for binary droplet collision, 25 570 frames from 51 simulations of approx. 500 frames each; for crown splash, 11 700 frames from 39 simulations

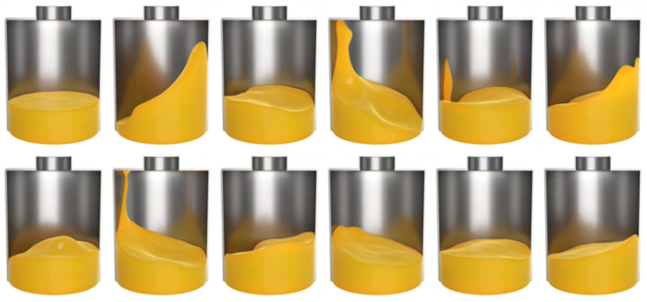


Fig. 5. NIRFS learns and simulates the latent dynamics of a weakly damped fluid slosh in a cylinder, reproducing high-fidelity sloshing orange juice.

of approx. 300 frames each; for slosh in a cylinder, 4 800 frames from 32 simulations of approx. 150 frames each.

For SDF dataset, equal numbers of SDF points are sampled inside and outside each mesh. Meshes are extracted and normalized to a unit sphere, hemisphere, and cylinder, respectively. Dense sampling for each scenario is within the normalized domain, using different sample counts depending on the complexity of the scenario. For each mesh in the binary droplet collisions and slosh in a cylinder sequences, we uniformly draw 250 000 SDF samples inside and outside the mesh, yielding a total of 500 000 samples per mesh. For crown splash, we uniformly draw 375 000, 375 000, and 250 000 samples inside, outside, and on the mesh, respectively, yielding a total of one million samples per mesh. The excessive sampling for crown splash is only to safeguard that sufficient samples are collected for high-frequency structures.

In training, SDF samples for each geometry are drawn from the SDF dataset according to a weight w_{point}. Given signed distance $s = SDF(\boldsymbol{x})$ for a point $\boldsymbol{x}$, for binary droplet collision and slosh in a cylinder, $w_{\text{point}} = (|s| + 1)^{-n}$; for crown splash, $w_{\text{point}} = w_{\text{importance}} (|s| + 1)^{-n}$, where $w_{\text{importance}} = 128$ for samples inside high-frequency structures and $w_{\text{importance}} = 1$ for all other samples (Sec. 3.1, *Importance sampling*). $n = 10$ for all experiments. We did not fine-tune any sampling hyperparameters.

4.2 Joint Training

We maintain a buffer of latent vectors $\tilde{z} = (\tilde{\boldsymbol{q}}, \tilde{\boldsymbol{p}})$, with one vector for each frame in ground truth simulations. The buffer directly provides inputs to the decoder and DHNODE. We can train the ICE to map physical initial conditions of each full length fluid sequence to latent space. The output of the ICE is directly input to the DHNODE and decoder; in this case, the first latent vectors for each fluid sequence in the buffer is replaced by the output of the ICE.

We conduct training with short fluid sequence segments uniformly sampled from the full length sequences. The buffer and DHNODE is updated based on every frame of each segment. To fit more segments in one batch, the decoder is updated based on frames uniformly down-sampled from each segment. In one epoch, we iterate through the same number of frames of SDF samples as the total number of frames in a training set.

We asynchronously optimize the ICE, buffer, DHNODE, and decoder. The ICE is updated per epoch. Because each full length fluid sequence only contains one segment starting from the initial condition, those segments present in each batch at a low frequency. The buffer and decoder are updated at each batch/step. The neural ODEs are updated on the order of every 10 batches/steps. Learning rates for the decoder and latent vectors, and learning rate schedules are adapted from DeepSDF [Park et al. 2019]. Initial learning rates for the DHNODE is around 0.001 and for the ICE 0.0004. Adam optimizer is used [Kingma and Ba 2014]. The training of neural ODE is considered converged when full trajectories rolled out by neural ODE match the buffer. Gradually increasing segment length may assist training.

The latent vector buffer can be initialized to zero or, as in DeepSDF [Park et al. 2019], random values from a normal distribution with a small variance and mean of zero. The latent mass matrix was initialized as $\mathbb{I}$. The initialization of the latent damping matrix is discussed in Sec. 4.4.

4.3 Loss function

For geometry reconstruction, we formulate a mean squared error (MSE) loss between decoder outputs and ground truth SDF samples:

$$\mathcal{L}_{SDF}(\boldsymbol{x}) = w_{\text{geometry}} \left(\mathcal{D}(\tilde{\boldsymbol{q}}, \boldsymbol{x}) - SDF(\boldsymbol{x})\right)^2 , \tag{3}$$

where w_{geometry} balances the difficulty of learning each geometry. While different scenarios may necessitate different weighting strategies, the weighting follows the intuition that more complex geometries require larger weights and we did not fine-tune any weighting heuristic. For binary droplet collision, $w_{\text{geometry}} = 5$ from the pre-collision initial geometry to the end of radial expansion [Roisman 2004] and $w_{\text{geometry}} = 1$ for later geometries. The conclusion of radial expansion is approximated to be the earliest local maximum of the cross-sectional area orthogonal to the collision axis. For crown splash and slosh in a cylinder, $w_{\text{geometry}} = h$, where h is the height of each fluid geometry.

During latent dynamics exploration, we roll out a segment using the DHNODE from a latent state $\tilde{z}_i$ in the buffer. We then compare this segment of latent states at discrete time points $(\tilde{z}_i, z_{i+1}, ...z_{i+k})$ to its counterpart in the buffer $(\tilde{z}_i, \tilde{z}_{i+1}, ...\tilde{z}_{i+k})$. We define an MSE loss between the ODE rollout and the buffer at the evaluated frames:

$$\mathcal{L}_{ODE} = \|\tilde{z} - z\|^2 . \tag{4}$$

Additionally, we fulfill the steady state constraint (Sec. 3.2) at a time point t_{steady} far enough in future for the fluid to be fully at rest. An MSE loss for the steady state of every trajectory segment is defined, with a geometry and a geometry momentum vector at t_{steady}:

$$\mathcal{L}_{\text{steady}} = \phi_1 \|\boldsymbol{q}(t_{\text{steady}}) - \boldsymbol{q}_{\text{steady}}\|^2 + \phi_2 \|\boldsymbol{p}(t_{\text{steady}})\|^2 ; \tag{5}$$

or alternatively with multiple geometry vectors beyond t_{steady}:

$$\mathcal{L}_{\text{steady}} = \phi_1 \frac{\sum_{n=0}^{N} \|\boldsymbol{q}(t_{\text{steady}} + n\Delta t) - \boldsymbol{q}_{\text{steady}}\|^2}{N+1} , \tag{6}$$

where Δt should be sufficiently small to avoid aliasing, because latent dynamics may contain oscillation, and $N\Delta t$ should be sufficiently large to ensure that the system remains in the steady state. In many cases including our examples, where a single steady state exists for a given scenario or boundary condition, we can simply set $\boldsymbol{q}_{\text{steady}} = \boldsymbol{0}$. No significant difference in effect was found between Equation 5 and 6.

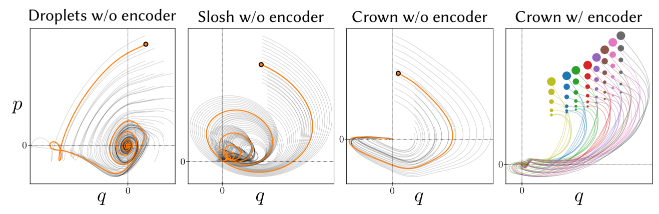


Fig. 6. Reduced phase portraits of the first principal component of latent states reveal physically-structured behavior. For experiments without an ICE, orange trajectories are reconstructed simulations from the training set, and grey trajectories are uniformly sampled from the latent spaces to illustrate the Hamiltonian structure. For the crown splash experiment with an ICE, uniformly sampled initial positions/offsets of the droplet is color coded; the size of ● scales with uniformly sampled initial speed.

The resulting total loss is a weighted sum of the loss terms:

$$\mathcal{L} = \mathcal{L}_{SDF} + \lambda \mathcal{L}_{ODE} + \mathcal{L}_{\text{steady}} , \tag{7}$$

We note that the training is not sensitive to the hyperparameters in Eq. 5 - 7. We provide our choice of the hyperparameters as a result of coarse tuning by altering the orders of magnitudes or significant steps, just to balance the magnitude of loss terms. For example, SDF values are orders of magnitude different for thin geometries (e.g., crown slash) and for bulk volumes, (e.g., slosh in a cylinder); therefore, SDF losses also differ in magnitude, requiring balancing. For binary droplet collision, $\lambda = 2$ is gradually increased to $\lambda = 2000$, $\phi_1 = \phi_2 = 10$, and $t_{steady} = 1200$ frames; for crown splash, $\lambda = 200$, $\phi_1 = 10$, $\phi_2 = 1$, and $t_{steady} = 300$ frames; for slosh in a cylinder, $\lambda = 200$ is gradually increased to $\lambda = 20\,000$, $\phi_1 = 100$, $\phi_2 = 10$, and $t_{steady} = 200$ frames. For all scenarios, we use $N = 10$ and $\Delta t = 10$ frames.

4.4 Neural ODE Stiffness

We consider a dynamics to be stiff when it changes rapidly with respect to one or more components or directions. Generally, the stiffness of Hamiltonian dynamics increases with the stiffness (steepness) of the energy function and decreases with the mass. For explicit solvers, high stiffness necessities a large number of small time steps and thus high cost. Implicit solvers can handle larger time steps for stiff systems, but also at the expense of higher cost; besides, only a very limited number of implicit solvers for neural ODEs has been proposed [Baker et al. 2022]. Neural ODEs may learn a stiff latent dynamics for an implicit system and become very expensive for training and inference.

Unlike a fully parameterized black box neural ODE, the explicit structure of our DHNODE offers intuitive control over the learning of latent dynamics. The stiffness of DHNODEs can be reduced by initializing damping coefficients with small values. Large damping restricts the change of states; to gain complexity in latent dynamics, a DHNODE learns a stiff energy function to gain forcing and a small mass to gain acceleration, yielding stiff dynamics. Low damping allows DHNODEs to have complexity with low stiffness.

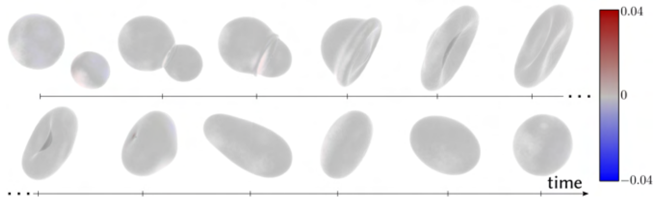


Fig. 7. Overlays of NIRFS reconstructions and ground truth reveal minimum geometry discrepancy, indicating high fidelity of reconstructed simulation. The ground truth is shown in grey surface and reconstructions are colored for signed distance error.

Initializing the damping to $0.00001\mathbb{I}$ achieved, compared to $\mathbb{I}$, a threefold acceleration in training with reduced number of integration time steps. We confirmed the decrease of the stiffness of the energy function with the decrease of the eigenvalues of the Hessian matrix of the energy function $\boldsymbol{H}(V(\boldsymbol{q}))$. We also confirmed the decrease of the mass and no side effects.

4.5 Implicit and Temporal Regularization

We generally desire and expect smoothness along and across individual fluid geometry sequences in the latent space. The Gaussian embedding enforces local smoothness, but not smoothness along or across latent trajectories. $\mathcal{L}_{ODE}$ explicitly maps temporal distance among fluid geometries to the latent space, achieving temporarily coherent and smooth latent trajectories. This temporal constraint anisotropically regularizes and smooths the latent space along the integral curves. It can be viewed as similar to applying Lipschitz style regularization on the decoder, but only for states that are part of the same trajectory.

During training phase, the neural ODE dynamically explores the latent space in various directions before converging to the final latent trajectories. This gradual and structured exploration of the latent space is driven not by discrete steps, as is the case for vanilla autodecoders, but rather by continuous-time dynamics, discouraging abrupt or discontinuous changes in the latent space. As a result, throughout the training process, the latent space and the decoder are implicitly regularized over the entire latent and primal space.

5 RESULTS AND EVALUATION

We examine the utility, generalizability and latent space properties of NIRFS with three fluid phenomena. We follow the convention of reduced-order modelling and train separate models for different scenarios. We did not fine-tune any model hyperparameters except for the variance of the frequencies in the Fourier feature mapping.

Compared to DeepSDF [Park et al. 2019], our decoders have lower complexity, use lower-dimensional latent spaces, and reconstruct surfaces without facets. We use eight fully connected layers with less nodes per layer. Notably, because ReLU activation produces faceted implicit surfaces, we use smooth activation functions for the decoder. In principle, different scenarios do not require different activation functions. For crown slash, however, we encountered "crack" artifacts in all geometries with tanh. Other functions, e.g., ReLU, ELU, and leaky-ReLU, yielded no such artifacts.

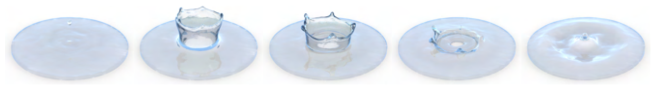


Fig. 8. NIRFS produces the geometric details of a novel crown splash. A latent initial condition is obtained by interpolating between two close latent initial conditions in the training set. The droplet hits the pool at a new direction and speed.

Our latent Hamiltonian energy functions have two fully connected layers with 128 neurons in each layer and tanh activation. The ICEs have the same architecture as the energy functions except for the input layer that depends on the dimensions of the physical initial condition parameters. The ICE is only demonstrated in the crown splash scenario.

Binary droplet collision. Two spherical droplets of varying volume collide at different relative speed and angle in zero gravity. The initial internal velocity is uniform. All simulations have the same total volume. The decoder has 256 neurons for each layer and uses tanh activation. The latent space has 32 dimensions.

Crown splash. A spherical droplet of constant volume and uniform internal velocity descends and impacts a shallow pool of zero internal velocity at different speed and angle. The droplet is offset from the horizontal centre so that it always lands at the centre of the pool. Gravity is included. Fourier feature mapping is used with 256 frequencies sampled from $\mathcal{N}(0, 49)$. Greater variance produces high frequency noise on geometry surfaces (Fig. 3b) and smaller variance leads to fragmental high-frequency structures (Fig. 3c). The decoder has 256 neurons for each layer except the middle layer of 800 neurons to which the input is concatenated. ELU activation is used. The latent space has 32 dimensions.

Slosh in a cylinder. In a stationary vertical cylindrical container, liquid with a leveled surface is initialized with different uniform internal velocity. Gravity is included. The decoder employs 256 neurons for the first four layers and 128 neurons for the remaining four layers, all using tanh activation. The latent space has eight dimensions.

5.1 Latent Space Physics

We visually characterize the trajectories and longitudinal behaviors of latent space dynamics through phase portraits, revealing stability, periodicity, and attractors of the Hamiltonian latent space (Fig. 6). For visualization, the phase portraits are projected to 2D with principal component analysis (PCA) by taking the first component of latent geometry and latent geometry momentum. The 2D phase portraits show steady states and oscillations or undulations that parallel those in fluid geometries. Notably, an ICE further structures the latent space by enforcing a deterministic relationship between physical initial conditions and latent states (Fig. 6 right-most). Without an ICE, multiple latent states may correspond to one physical initial conditions, and the latent spaces appear to be less structured, although some correlations between latent and physical states were observed.

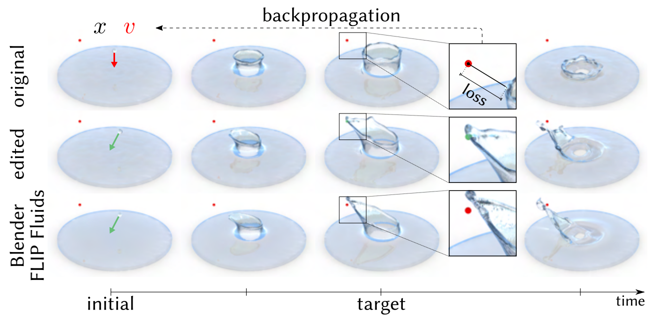


Fig. 9. We edit an existing novel simulation by setting a target surface point at a target time. With an ICE, the initial droplet position and velocity are optimized for the target. From the new initial condition, NIRFS and Blender FLIP Fluids produce similar results.

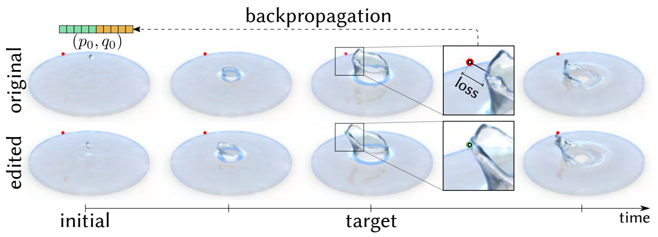
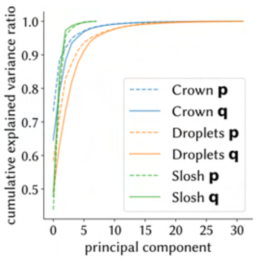


Fig. 10. We obtain a targeted novel simulation by editing an existing novel simulation. Without an ICE, we optimize the latent initial condition so that the fluid reaches a target surface point at a target time.

We examined the landscape of the latent spaces through the eigenvalues of the Hessian of the learned energy function $\boldsymbol{H}(V(\boldsymbol{q}))$. Without imposing any constraints, the energy function naturally learns to be convex around an equilibrium point. This convexity with respect to latent fluid geometry ensures that greater deformation produces greater forcing and that numerical integration in the latent space is stable for proper time steps. Interestingly, when the energy function is the squared output of an MLP, the convexity is not preserved and the energy function is flat in all but one direction around an equilibrium point.



We note that a large portion of the variance of position and momentum can be explained with just a few latent space dimensions, i.e., 90% with less than 5 components. Analogous to low-dimension subspace selection of reduced elastics [Barbič and James 2005], latent space dimensionality relates to expressivity of the reduced model. Complex geometries benefit from higher-dimensional latent spaces. We observed no significant effects when the dimensionality is higher than necessary, e.g., 32 rather than 8 dimensions for slosh in the cylinder.

5.2 Trained Simulation and Novel Simulation

NIRFS integrates the DHNODE in the latent space and decode the trajectory of implicitly represented fluid shapes. We reconstruct the trajectory of a simulation in the training set from a learned latent initial condition $\boldsymbol{z}_0 = (\boldsymbol{q}_0, \boldsymbol{p}_0)$ (Fig. 5). We observe that the reconstructed simulations are visually indistinguishable from ground truth (Fig. 7).

We generate novel simulations not included in the training set from new latent initial conditions, that can be obtained in different ways. With an ICE, we simply set a new physical initial condition (Fig. 1). Otherwise, we can optimizing latent geometry vectors to match provided initial geometry SDFs and then compute an initial latent geometry momentum vector. We can also interpolate between known latent initial conditions that are close in the physical parameter space (Fig. 8). Another strategy is to sample within the the convex hull of a 2D embedding of latent initial conditions of the training set (Fig. 4). Novel simulations produced in these manners are physically realistic.

5.3 Fluid Control

Fluid trajectory editing tasks is another case where we can create novel simulations that satisfy a desired objective. Empowered by the full differentiability of NIRFS, we solve an inverse fluid design as an optimization problem. The design variable is the physical initial condition (Fig. 9), or if an ICE is not applicable, the latent initial condition $\boldsymbol{z}_0 = (\boldsymbol{q}_0, \boldsymbol{p}_0)$ (Fig. 10). We edit a given crown splash trajectory by setting a target point for the fluid surface to reach at a target time. We then optimize the design variable through gradient descent on the squared signed distance (squared decoder output) at the target point and time. Optimization cost varies by task but is generally less than five minutes and as low as a few seconds. From the optimized physical initial condition, NIRFS and Blender FLIP Fluids produce similar results (Fig. 9).

5.4 Conservation of Volume and Momentum

Our incompressible fluid simulations do not feature sources nor sinks and thus volume should be preserved along all trajectories in all of our simulations. Furthermore, in the case of binary droplet collisions, we also expect the center of mass to be constant (i.e., a conservation of linear momentum).

Figure 11 (left) shows volume conservation for training data, reconstructions, and novel trajeoctires. For the ground truth and reconstruction, we compute the ratio of the volume of each fluid geometry to the initial volume of the ground truth simulation. For novel simulation, we compute the ratio of the volume of each fluid geometry to the initial volume of the novel simulation.

Figure 11 (right) shows centre of mass trajectories for the binary droplet collision scenario. We reset the origin of each ground truth geometry to its center of mass during normalization (Section 4.1). With geometries normalized to a unit sphere, these deviations in the center of mass position is very small, i.e., below 0.25% the domain size.

With no direct constraints on either volume or center of mass in the loss function, the conservation of the physical properties depends on the quality of INRs. Deviation from the conserved quantity

Table 1. NIRFS offers a significant speedup compared to Blender FLIP Fluids. + denotes latent trajectory integration; ++ denotes latent trajectory integration and SDF evaluation; +++ denotes latent trajectory integration, SDF evaluation, and marching cubes.

	Number of frames	SDF grid resolution	NIRFS (s)			FLIP Fluids (10^3 s)	Speedup ($\times$)		
			+	++	+++		+	++	+++
Droplets	500	256^3		500.12	625.12	28.8	2.4×10^5	58	46
Crown	300	512^3	0.12	1350.12	1980.12	61.2	5.1×10^5	45	31
Slosh	150	512^3		675.12	990.12	21.6	1.8×10^5	32	22

originates from geometry representation error. The conservation is well maintained by our method, with deviation concentrated at the beginning of simulations where geometries are more complicated.

5.5 Cost

NIRFS demonstrates a substantial speed advantage over conventional fluid simulation (Table 1). The cost arises from latent trajectory integration, SDF evaluation, and surface extraction.

We use the fifth-order Runge-Kutta method of Tsitouras [2011] with adaptive time step for latent trajectory integration. Solution at specified time instants is evaluated through interpolation. All experiments are implemented with PyTorch [Paszke et al. 2019] and torchode [Lienen and Günnemann 2022]. The approximate cost per time step is 2.2 milliseconds on an AMD EPYC 7532 32-core CPU and from 9.5 to 19.2 milliseconds on an NVIDIA A100 GPU. Respectively, the approximate cost per trajectory is 0.12 ± 0.05 and 0.55 ± 0.01 seconds for all three scenarios. Massive parallelization of the integration of trajectories can be done on a GPU with almost no additional cost.

INRs are continuous and support arbitrary grid resolution. The cost for the decoder to evaluate on a 128^3 grid is approximately 0.147 seconds on an NVIDIA A100 GPU. We evaluated SDF for demonstrations on 256^3 (Fig. 4) and 512^3 (Fig. 1 and 5) grids, taking 1s and 4.5s per frame. While we used marching cubes to extract a surface, which takes 0.25s and 2.1s per frame, we can avoid meshes and render the INRs with sphere tracing.

6 DISCUSSION

NIRFS admits a number of design decisions and is demonstrated through specific scenarios under varying initial conditions. First, we discuss ablation experiments to further justify the design decisions

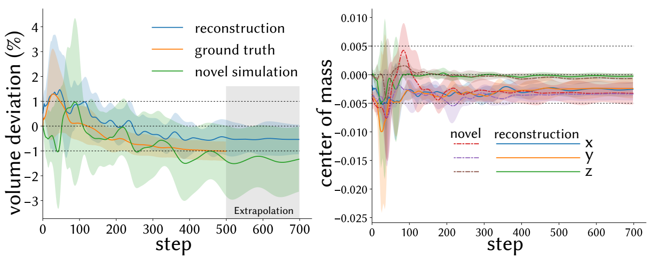


Fig. 11. NIRFS conserves volume and momentum throughout binary droplet collisions. Volume is conserved comparably to the ground truth. Center of mass exhibits limited deviation from the origin of the unit sphere.

presented in Sections 3 and 4. Subsequently, we discuss the scope of generalization for NIRFS.

We validate the importance of the an explicit Hamiltonian model by replacing it with a fully MLP-parameterized neural ODE. This alternative learns smooth latent dynamics without an embedded structure: latent trajectories do not exhibit steady states, oscillations, nor undulations that parallel fluid dynamics. Notably, the lack of a latent steady state results in free extrapolation of latent states (and thus geometries) beyond the training regime.

We experimentally tested the shortcomings of decoupled training. First, we remove the latent neural ODE entirely, training only a decoder. After densely sampling geometries from continuous geometric sequences, latent trajectories from different sequences displayed varying degrees of smoothness. Some trajectories were piecewise smooth (i.e., with intermittent irregularities), while others lacked any (even partial) smoothness. Here, we observed that neither Gaussian embeddings nor Lipschitz regularization [Liu et al. 2022] significantly impacted the smoothness of latent trajectories. Subsequently, we attempted to train a Hamiltonian and a fully parameterized neural ODE to model the latent trajectories for the (separately trained) decoder; however, we were unable to successfully train such a model: discontinuities in the latent trajectories destabilize the neural ODE during training, due to the unconstrained structure of these latent trajectories likely belonging to function classes ill-suited to neural ODE representation, i.e., those functions with discontinuities and intersections [Dupont et al. 2019].

We use a Gaussian embedding to encourage a locally smooth latent space that better tolerates integration error. Training the decoder without a Gaussian embedding can result in a decoder that maps significantly different geometries from nearby latent vectors. Consequently, the ablation of Gaussian embedding introduces instability with respect to ODE integration error.

The generalizability of NIRFS is demonstrated across initial conditions for specific scenarios through novel simulations produced in different ways (Sec. 5.2 and 5.3). We note that the demonstrations used small numbers of simulations for training, i.e., 51 for binary droplet collision, 39 for crown splash, and 32 for slosh in a cylinder. Future work can explore larger physical parameter spaces and boundary conditions to enhance generalization, with more data collection and larger networks. Scenarios involving other materials (e.g., smoke), interactions, and moving boundaries are all excellent avenues for future work.

Generalizing over simulation resolution is also feasible by training with data of different simulation resolutions. However, because the cost of NIRFS is unaffected by training simulation resolution, it is reasonable to train models on accurate simulations of a fixed high resolution.

While we only demonstrated INR for SDF and Hamiltonian latent dynamics, we note that the latent space is agnostic to the type of physical dynamics and data (e.g., SDFs, radiance fields, density maps, occupancy fields, etc.), and that the latent dynamics can have formulations other than Hamiltonian. The value here is that latent space physics can have alternative forms to the physical PDE.

Simplifications in the latent Hamiltonian model, including Cartesian latent space, linear damping/dissipation, and diagonal damping and mass matrices, trade physical fidelity for reduced complexity. Future studies may seek if a more complex physical system requires a more faithful latent space model. The damping term emulates viscous dissipation. Inviscid fluid dynamics without steady state can be explored in the future by, e.g., removing the damping term and adjusting learning strategy without the steady state loss.

7 CONCLUSION

Learning reduced neural-implicit representations combined with latent space neural ordinary differential equations is a viable technique for efficiently modeling the dynamics of costly fluid simulations. Our results show that the learned latent space fluid dynamics is smooth and stable, and conserves volume and momentum. Producing fluid trajectories with NIRFS is many orders of magnitude faster than computing a full simulation. Just the same, we acknowledge that there are opportunities to make NIRFS computation even faster with smaller latent space, and smaller networks.

We explored single-fluid settings but believe NIRFS can be applied to scenes with multi-scale fluid interactions, such as multi-droplet interactions that merge and split. As a surrogate model, NIRFS can be applied to control applications (e.g., what external force moves a bottle to a desired location without spilling its juice). Here, we are excited to explore how the core NIRFS concepts can be applied to other physical systems, such as elastic solids or thin shells.

References

R. Altmann and P. Schulze. 2017. A port-Hamiltonian formulation of the Navier-Stokes equations for reactive flows. *Systems & Control Letters* 100 (2017), 51–55.

V. I. Arnold. 2014. *Hamiltonian nature of the Euler equations in the dynamics of a rigid body and of an ideal fluid.* Springer Berlin Heidelberg, Berlin, Heidelberg, 175–178. https://doi.org/10.1007/978-3-642-31031-7_16

V. I. Arnold and B. A. Khesin. 2008. *Topological methods in hydrodynamics.* Vol. 125. Springer Science & Business Media.

J. Baker, H. Xia, Y. Wang, E. Cherkaev, A. Narayan, L. Chen, J. Xin, A. L. Bertozzi, S. J. Osher, and B. Wang. 2022. Proximal implicit ODE solvers for accelerating learning neural ODEs. *arXiv preprint arXiv:2204.08621* (2022).

J. Barbič and D. L. James. 2005. Real-Time subspace integration for St. Venant-Kirchhoff deformable models. *ACM Trans. Graph.* 24, 3 (jul 2005), 982–990. https://doi.org/10.1145/1073204.1073300

Blender Foundation. 2021. Blender. https://www.blender.org. Version 2.92.0.

R. Camassa, G. Falqui, G. Ortenzi, and M. Pedroni. 2014. On variational formulations and conservation laws for incompressible 2D Euler fluids. *Journal of Physics: Conference Series* 482, 1 (mar 2014), 012006. https://doi.org/10.1088/1742-6596/482/1/012006

K. Champion, B. Lusch, J. N. Kutz, and S. L. Brunton. 2019. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences* 116, 45 (2019), 22445–22451. https://doi.org/10.1073/pnas.1906995116

E. R. Chan, C. Z. Lin, M. A. Chan, K. Nagano, B. Pan, S. De Mello, O. Gallo, L. J. Guibas, J. Tremblay, S. Khamis, et al. 2022. Efficient geometry-aware 3D generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 16123–16133.

H. Chen, R. Wu, E. Grinspun, C. Zheng, and P. Y. Chen. 2023a. Implicit neural spatial representations for time-dependent PDEs. In *International Conference on Machine Learning.* PMLR, 5162–5177.

P. Y. Chen, J. Xiang, D. H. Cho, Y. Chang, G. A. Pershing, H. T. Maia, M. M. Chiaramonte, K. T. Carlberg, and E. Grinspun. 2023b. CROM: Continuous Reduced-Order Modeling of PDEs Using Implicit Neural Representations. In *The Eleventh International Conference on Learning Representations.* https://openreview.net/forum?id=FUORz1tG8Og

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. 2018. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems*, Vol. 31.

Z. Chen and H. Zhang. 2019. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 5939–5948.

E. Dupont, A. Doucet, and Y. W. Teh. 2019. Augmented neural ODEs. *Advances in neural information processing systems* 32 (2019).

E. Dupont, H. Kim, S. Eslami, D. Rezende, and D. Rosenbaum. 2022. From data to functa: Your data point is a function and you can treat it like one. *arXiv preprint arXiv:2201.12204* (2022).

D. G. Ebin and J. Marsden. 1970. Groups of diffeomorphisms and the motion of an incompressible fluid. *Annals of Mathematics* (1970), 102–163.

R. Fedkiw, J. Stam, and H. W. Jensen. 2001. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques.* 15–22.

L. Fulton, V. Modi, D. Duvenaud, D. I. Levin, and A. Jacobson. 2019. Latent-space dynamics for reduced deformable simulation. In *Computer graphics forum*, Vol. 38. Wiley Online Library, 379–391.

S. Greydanus, M. Dzamba, and J. Yosinski. 2019. Hamiltonian neural networks. *Advances in neural information processing systems* 32 (2019).

R. Guy and D. Fassbaender. 2022. FLIP Fluids addon for Blender. https://flipfluids.com/. 1.4.0.

G. Haine and D. Matignon. 2021. Incompressible Navier-Stokes Equation as port-Hamiltonian systems: velocity formulation versus vorticity formulation. *IFAC-PapersOnLine* 54, 19 (2021), 161–166.

D. L. James and K. Fatahalian. 2003. Precomputing interactive dynamic deformable scenes. *ACM Trans. Graph.* 22, 3 (2003), 879–887. https://doi.org/10.1145/882262.882359

B. D. Jones. 2014. Navier-Stokes Hamiltonian for the Similarity Renormalization Group. *arXiv preprint arXiv:1407.1035* (2014).

A. N. Kaufman and P. J. Morrison. 1982. Algebraic structure of the plasma quasilinear equations. *Physics Letters A* 88, 8 (1982), 405–406. https://doi.org/10.1016/0375-9601(82)90664-8

B. Kim, V. C. Azevedo, N. Thuerey, T. Kim, M. Gross, and B. Solenthaler. 2019. Deep Fluids: A Generative Network for Parameterized Fluid Simulations. *Computer Graphics Forum* 38, 2 (2019), 59–70. https://doi.org/10.1111/cgf.13619

D. P. Kingma and J. Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

D. P. Kingma and M. Welling. 2013. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114* (2013).

B. Kolev. 2007. Poisson brackets in Hydrodynamics. *Discrete and Continuous Dynamical Systems - A* 19, 3 (2007), 555–574. https://doi.org/10.3934/dcds.2007.19.555

L. Ladický, S. Jeong, B. Solenthaler, M. Pollefeys, and M. Gross. 2015. Data-driven fluid simulations using regression forests. *ACM Trans. Graph.* 34, 6, Article 199 (nov 2015), 9 pages. https://doi.org/10.1145/2816795.2818129

R. Lantz. 1971. Quantitative evaluation of numerical diffusion (truncation error). *Society of Petroleum Engineers Journal* 11, 03 (1971), 315–320.

M. Lienen and S. Günnemann. 2022. torchode: A Parallel ODE Solver for PyTorch. In *The Symbiosis of Deep Learning and Differential Equations II.* https://openreview.net/forum?id=uiKVKTiUYB0

H.-T. D. Liu, F. Williams, A. Jacobson, S. Fidler, and O. Litany. 2022. Learning smooth neural functions via Lipschitz regularization. In *ACM SIGGRAPH 2022 Conference Proceedings.* 1–13.

B. Lusch, J. N. Kutz, and S. L. Brunton. 2018. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications* 9, 1 (2018), 4950.

A. Maesumi, P. Guerrero, V. G. Kim, M. Fisher, S. Chaudhuri, N. Aigerman, and D. Ritchie. 2023. Explorable Mesh Deformation Subspaces from Unstructured Generative Models. *arXiv preprint arXiv:2310.07814* (2023).

S. Massaroli, M. Poli, M. Bin, J. Park, A. Yamashita, and H. Asama. 2020. Stable Neural Flows. arXiv:2003.08063 [cs.LG]

L. Mescheder, M. Oechsle, M. Niemeyer, S. Nowozin, and A. Geiger. 2019. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition.* 4460–4470.

B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

J. W. Miles. 1977. On Hamilton's principle for surface waves. *Journal of Fluid Mechanics* 83, 1 (1977), 153–158.

P. J. Morrison. 1984. Some observations regarding brackets and dissipation. *Center for Pure and Applied Mathematics Report PAM-228, University of California, Berkeley* (1984).

P. J. Morrison. 1998. Hamiltonian description of the ideal fluid. *Reviews of modern physics* 70, 2 (1998), 467.

K. Museth. 2013. VDB: High-resolution sparse volumes with dynamic topology. *ACM transactions on graphics (TOG)* 32, 3 (2013), 1–22.

P. J. Olver. 1982. A nonlinear Hamiltonian structure for the Euler equations. *J. Math. Anal. Appl.* 89, 1 (1982), 233–250.

V. I. Oseledets. 1989. On a new way of writing the Navier-Stokes equation. The Hamiltonian formalism. *Russ. Math. Surveys* 44 (1989), 210–211.

J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove. 2019. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*

K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R. Martin-Brualla. 2021a. Nerfies: Deformable Neural Radiance Fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).* 5865–5874.

K. Park, U. Sinha, P. Hedman, J. T. Barron, S. Bouaziz, D. B. Goldman, R. Martin-Brualla, and S. M. Seitz. 2021b. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228* (2021).

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32.* 8024–8035.

T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. Battaglia. 2021. Learning Mesh-Based Simulation with Graph Networks. In *International Conference on Learning Representations.* https://openreview.net/forum?id=roNqYL0_XP

P. J. Roache. 1998. *Fundamentals of computational fluid dynamics.* Hermosa Publishers, Albuquerque, NM.

I. V. Roisman. 2004. Dynamics of inertia dominated binary drop collisions. *Physics of Fluids* 16, 9 (08 2004), 3438–3449. https://doi.org/10.1063/1.1777584

J. W. Sanders, A. C. DeVoria, N. J. Washuta, G. A. Elamin, K. L. Skenes, and J. C. Berlinghieri. 2023. A canonical Hamiltonian formulation of the Navier-Stokes problem. *arXiv preprint arXiv:2310.07085* (2023).

K. Schwarz, Y. Liao, M. Niemeyer, and A. Geiger. 2020. Graf: Generative radiance fields for 3d-aware image synthesis. *Advances in Neural Information Processing Systems* 33 (2020), 20154–20166.

N. Sharp and A. Jacobson. 2022. Spelunking the deep: Guaranteed queries on general neural implicit surfaces via range analysis. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–16.

N. Sharp, C. Romero, A. Jacobson, E. Vouga, P. G. Kry, D. I. Levin, and J. Solomon. 2023. Data-Free Learning of Reduced-Order Kinematics. (2023).

M. Stanton, B. Humberston, B. Kase, J. F. O'Brien, K. Fatahalian, and A. Treuille. 2014. Self-Refining Games Using Player Analytics. *ACM Trans. Graph.* 33, 4, Article 73 (jul 2014), 9 pages. https://doi.org/10.1145/2601097.2601196

T. Takikawa, J. Litalien, K. Yin, K. Kreis, C. Loop, D. Nowrouzezahrai, A. Jacobson, M. McGuire, and S. Fidler. 2021. Neural geometric level of detail: Real-time rendering with implicit 3d shapes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 11358–11367.

M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng. 2020. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems* 33 (2020), 7537–7547.

J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin. 2017. Accelerating Eulerian Fluid Simulation With Convolutional Networks. In *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70. PMLR, 3424–3433. https://proceedings.mlr.press/v70/tompson17a.html

A. Treuille, A. Lewis, and Z. Popović. 2006. Model reduction for real-time fluids. *ACM Transactions on Graphics* 25, 3 (July 2006), 826–834.

C. Tsitouras. 2011. Runge–Kutta pairs of order 5 (4) satisfying only the first column simplifying assumption. *Computers & Mathematics with Applications* 62, 2 (2011), 770–775.

P. R. Vlachas, W. Byeon, Z. Y. Wan, T. P. Sapsis, and P. Koumoutsakos. 2018. Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 474, 2213 (2018), 20170844.

C. Wehmeyer and F. Noé. 2018. Time-lagged autoencoders: Deep learning of slow collective variables for molecular kinetics. *The Journal of Chemical Physics* 148, 24 (03 2018). https://doi.org/10.1063/1.5011399 241703.

S. Wiewel, M. Becher, and N. Thuerey. 2019. Latent space physics: Towards learning the temporal evolution of fluid flow. In *Computer graphics forum*, Vol. 38. Wiley Online Library, 71–82.

R. Wu and C. Zheng. 2022. Learning to generate 3d shapes from a single example. *arXiv preprint arXiv:2208.02946* (2022).

G. Yang, S. Belongie, B. Hariharan, and V. Koltun. 2021. Geometry processing with neural fields. *Advances in Neural Information Processing Systems* 34 (2021), 22483–22497.

J. Zehnder, Y. Li, S. Coros, and B. Thomaszewski. 2021. Ntopo: Mesh-free topology optimization using implicit neural representations. *Advances in Neural Information Processing Systems* 34 (2021), 10368–10381.