Exploring Physical Latent Spaces for High-Resolution Flow Restoration

Chloé Paliard¹, Nils Thuerey², Kiwon Um¹

¹LTCI, Télécom Paris, Institut Polytechnique de Paris, France
²Technical University of Munich, Germany

Figure 1: We propose a model composed of neural components and a physics solver, that autonomously discovers the reduced representation (a) that best fulfills the goal of restoring a fine reference simulation (b, c) from a unique coarse frame. This leads to relative improvements with respect to the baseline of 91% on average for the Karman vortex street case, 74% for the forced turbulence case and 35% for the smoke plume case.

Abstract
We explore training deep neural network models in conjunction with physics simulations via partial differential equations (PDEs), using the simulated degrees of freedom as latent space for a neural network. In contrast to previous work, this paper treats the degrees of freedom of the simulated space purely as tools to be used by the neural network. We demonstrate this concept for learning reduced representations, as it is extremely challenging to faithfully preserve correct solutions over long time-spans with traditional reduced representations, particularly for solutions with large amounts of small scale features. This work focuses on the use of such physical, reduced latent space for the restoration of fine simulations, by training models that can modify the content of the reduced physical states as much as needed to best satisfy the learning objective. This autonomy allows the neural networks to discover alternate dynamics that significantly improve the performance in the given tasks. We demonstrate this concept for various fluid flows ranging from different turbulence scenarios to rising smoke plumes.

CCS Concepts
• Computing methodologies → Physical simulation; Learning latent representations;

1. Introduction
Realistic simulation of natural phenomena is one of the ultimate goals of Computer Graphics research. Modeling and recreating such phenomena typically involves partial differential equations (PDEs) and thus numerical methods that aim for efficient computation of their solution. Despite the recent advances in numerical methods and computing power, many PDE problems of real-world scenarios, such as fluid simulation, require extremely costly calculations to resolve fine details, which are yet essential in the graphics context. Thus, using traditional numerical methods, which often require super-linear scaling in computation, remains challenging in practice. To minimize the costs of the computation, one can consider resolving the PDE in a reduced space, yet sacrificing the desired fine degrees of freedom both temporally and spatially. As an attractive compromise between computation and resolution, data-driven meth-
In this paper, we present a data-driven simulation technique for PDE problems with a novel training method that explores physical states as latent space for deep learning. In contrast to many previous studies [WKA*20; FMZF21], our latent space is not composed of the output or intermediate states of a neural network, but is rather made of the physical states of a PDE solver, such as velocity fields. We train a deep neural network (DNN) to exploit the content of a reduced PDE solver and shape it in a way that best satisfies the given learning objective, i.e., achieving solutions that are as accurate as possible at our target high resolution space. This shaping of the physical latent space gives the neural network a chance to discover modified dynamics and allows our model to better restore accurate high resolution solutions from them. Examples of the reduced and restored solutions are shown in Fig. 1.

Our training method consists of an encoder model transforming a coarse physical state using the degrees of freedom of a learnable latent space, a physics solver corresponding to a given PDE followed by an adjustment DNN, both operating in the reduced space, and a decoder turning the reduced state into the target high resolution space. To train our models with a physics solver, we adopt a differentiable simulator approach [HAL*20; HKT20; THM*21]. We let the encoder model learn the latent space representation without any other constraint than the restoration of the target solution. Therefore, an end-to-end training of this pipeline gives the encoder the complete autonomy to shape the reduced representation.

This paper demonstrates that the autonomy of our training method leads to a better performance than previous work, especially in terms of generalization. We apply our method to various complex, non-linear PDE problems, based on the Navier-Stokes equations, which are essential in the context of modeling fluid flows. For all the scenarios, our model produces more accurate high-resolution results in a longer temporal horizon than conventional and more tightly constrained models.

2. Related Work

Learning a PDE The study of machine learning (ML) techniques for PDEs is getting more and more popular [CM87; KGH*03; BPK16]. A conventional direction when using ML for PDEs is to aim for the replacement of entire PDE solvers by neural network models that can efficiently approximate the solutions as accurately as possible [LKB18; KAT*19; WKM*20; BHKS21]. In this context, Fourier Neural Operator [LKA*21] and Neural Message Passing [BWW22] models have been introduced for learning PDEs, aiming at a better representation of full solvers with neural network models.

Instead of the pure ML-driven approach to solve target PDEs, an alternative approach exists in the form of hybrid methods that combine ML with traditional numerical methods. Among many different PDEs, fluid problems have received great attention due to their complex nature. For example, a learned model can replace the most expensive part of an iterative PDE solver for fluids [TSSP17; XYY18] or supplement inexpensive yet under-resolved simulations [UHT18; SMF20]. A regression forests model was also proposed for fast Lagrangian liquid simulations [LJS*15]. For smoke simulations in particular, efficient DNNs approaches synthesize high-resolution results from low-resolution versions [CT17; XFCT18; BDL20] and convert low frame rate results into high frame rate versions [OL21].

Differentiable solvers Recently, differentiable components for ML have been studied extensively, particularly when training neural network models in recurrent setups for spatio-temporal problems [AK17; dASA*18; TAST18; CRBD18; SF18; LLK19; WAG20; UBF*20; KSA*21; ZKB*21]. Consequently, a variety of differentiable programming frameworks have been developed for different domains [SC19; HAL*20; IEF*19; HKT20]. These differentiable frameworks allow neural networks to closely interact with PDE solvers, which provides the model with important feedback about the temporal evolution of the target problem from the recurrent evaluations. Targeting similar problems for temporal evolution, we employ a differentiable framework in our training procedure.

Latent space representations Effectively utilizing latent spaces lies at the heart of many ML-based approaches for solving PDEs. A central role of the latent space is to embed important (often non-linear) information for the given training task into a set of reduced degrees of freedom. For example, with an autoencoder network architecture, the latent space can be used for discovering interpretable, low-dimensional dynamical models and their associated coordinates from high-dimensional data [CLKB19]. Moreover, thanks to their effectiveness in terms of embedding information and reducing the degrees of freedom, latent space solvers have been proposed for different problems such as advection-dominated systems [MLB21] and fluid flows [WKA*20; FMZF21]. While those studies typically focus on training equation-free evolution models, we focus on latent states that result from the interaction with a PDE solver. Neural network models have also been studied for the integration of a dynamical system with an ordinary differential equation (ODE) solver in the latent space [CRBD18]. This approach targets general neural network approximations with a simple physical model in the form of an ODE, whereas we focus on learning tasks for complex non-linear PDE systems.

Reduced solutions The ability to learn underlying PDEs has allowed neural networks to improve reduced, approximate solutions. Residual correction models are trained to address numerical errors of PDE solvers [UBF*20]. Details at sub-grid scales are improved via learning discretizations of PDEs [BHBB19] and learning solvers [KSA*21; SFK*22] from high-resolution solutions. Moreover, multi-scale models with downsampling skip-connections have been used for super-resolution tasks of turbulent flows [FTF19]. These methods, however, typically employ a constrained solution manifold for the reduced representation. Indeed, the reduced solutions are produced using coarse-grained simulations with standard numerical methods, while our work shows the advantages of autonomously exploring the latent space representation through our joint training methodology.

3. Exploring Physical Latent Spaces

For a given learning objective, our training method explores how neural network models can leverage the physical states of a PDE
as latent space. Let \( f \in \mathbb{R}^{d_f} \) and \( r \in \mathbb{R}^{d_r} \) denote two discretized solutions of a PDE, a fine and a coarse version respectively, with \( d_r \ll d_f \). We focus on the numerical integration of this target PDE problem and indicate the temporal evolution of each state as a subscript. A reference solution trajectory integrated from a given initial state \( f_1 \) at time \( t \) for \( n \) steps is represented by the finite set of states \( \{ f_1, f_{t+1}, \ldots, f_{t+n} \} \). Each reference state is integrated over time with a fixed time-step size using a numerical solver \( P_{f_{t+1}} \), i.e., \( f_{t+1} = P_{f_{t}}(f_t) \). Similarly, we integrate a reduced state \( r_1 \) over time using a corresponding numerical solver \( P_r \) at the reduced space, which we will call reduced solver henceforth, i.e., \( r_{t+1} = P_r(r_t) \).

In this paper, we focus on cases where the solver \( P_r \) is the same for both reduced and fine discretizations.

Our model takes the linear down-sampling of \( f_i \), i.e., \( s_i = \text{lerp}(f_i) \), as input, and transforms it with the help of an encoder function \( \mathcal{E}(\cdot; \theta_E) : \mathbb{R}^{d_r} \rightarrow \mathbb{R}^{d_r} \), thus \( \mathcal{E}(s_i; \theta_E) = \hat{s}_i \). Then, we can obtain the next reduced state \( r_{t+1} = \mathcal{P}_{f_{t}}(\hat{s}_i) \). Moreover, in order to keep the reduced solution consistent with the encoded representation over time, the output of the reduced solver is transformed by an adjustment function, \( A(r_{t+1} ; \theta_A) = \hat{r}_{t+1} \). Thus, each reduced state \( \hat{r}_{t+1} \) is obtained by recurrent evaluations of the reduced solver and the adjustment function. Finally, a decoder function \( D(\cdot; \theta_D) : \mathbb{R}^{d_r} \rightarrow \mathbb{R}^{d_f} \) restores a fine solution trajectory \( \{ f_{t}, f_{t+1}, \ldots, f_{t+n} \} \) from the reduced trajectory \( \{ \hat{f}_{t}, \hat{f}_{t+1}, \ldots, \hat{f}_{t+n} \} \), thus \( \hat{f}_{t+i} = D(\hat{r}_{t+i} ; \theta_D) \). We model the encoder, adjustment, and decoder functions as DNNs in which trainable weights are denoted by \( \theta_E \), \( \theta_A \), and \( \theta_D \), respectively.

The joint learning objective of the three DNNs is to minimize the error between the approximate solutions and their corresponding reference solutions, i.e., \( ||\hat{f}_{t+i} - f_{t+i}||_2 \). To guide the adjustment model, we additionally minimize \( ||\hat{r}_{t+i} - \mathcal{E}(s_{t+i} ; \theta_E)||_2 \). Thus, the final loss of our model is as follows:

\[
\mathcal{L} = \sum_{i=1}^{N} \lambda_{\text{hires}} \times ||\hat{f}_{t+i} - f_{t+i}||_2 + \lambda_{\text{latent}} \times ||\hat{r}_{t+i} - \mathcal{E}(s_{t+i} ; \theta_E)||_2
\]

(1)

where \( N \) denotes the number of integrated time-steps for training. Hence, at each training iteration, the gradients through all \( N \) steps are computed for back-propagation and, consequently, all the models get jointly updated.

4. Experiments

For each of the following scenarios, which are represented in 2D, the reference solution trajectories are generated for 200 steps from different initial conditions with a fixed time-step size. We focus on the velocity field for our restoration task and consider a four times coarser discretization for the reduced representation. More details about the experimental setups are given in the appendix.
Karman vortex street Forced turbulence
Decaying turbulence

4.1. Karman vortex street

We first consider a complex constrained PDE problem in the form of the Navier-Stokes equations. This problem is modeled as follows:

\[
\frac{\partial v}{\partial t} = -(v \cdot \nabla)v - \nabla p + \nu \nabla^2 v \\
\text{subject to } \nabla \cdot v = 0
\]

where \( v \) is the velocity, \( p \) is the pressure, \( \rho \) is the density and \( \nu \) is the viscosity.

In this scenario, shown in Fig. 3 (left), a continuous inflow collides with a fixed circular obstacle. It creates an unsteady wake flow, which evolves differently depending on the Reynolds number. For the reference solutions, we use a numerical fluid solver that adopts the operator splitting scheme, Chorin projection for implicit pressure solve [Cho67], semi-Lagrangian advection [Sta99], and explicit integration for diffusion. We choose Reynolds numbers between \( Re = 90 \) and \( Re = 1190 \) for our training data-set, and Reynolds numbers from \( Re = 450 \) to \( Re = 1400 \) for testing. The encoder of our ATO setup takes the Reynolds number as an additional input in order to guide the exploration of the reduced space. The DilResNet and SOL setups also receive the Reynolds number to let the models learn different physics evolutions. Finally, in order to be fair in our comparisons, the obstacle mask is applied to each state condition problems and zero padding for the others. The architecture of all target solutions, hence an improvement of 100% would mean that the reduced solver without interactions with any neural network and a baseline simulation. To make the baseline solutions, we apply the reduced solver for an arbitrary density distribution in a circular shape. The marker field is then passively advected by the Boussinesq approximation, that is influencing the velocity evolution. Therefore, the marker and velocity fields are tightly coupled. This scenario considers a more challenging problem of the Navier-Stokes equations than before, naturally making the fluid flow more interesting and providing a harder task for our model. The training and test data-sets are composed of smoke volumes initialized with random noise, with a fixed radius and position. The passive marker field is given as an input to our encoder and adjustment models, but its linearly down-sampled version is used in the reduced solver. Then, only the velocity field is up-sampled, and the high-resolution marker field is advected by the predicted velocity. The simulation domain is discretized with \( 128^2 \) cells adopting a centered layout for the marker field, a staggered layout for the velocity field, and open boundary conditions.

4.4. Network architecture and training procedure

The encoder \( E(\cdot; \theta_l) \) is implemented as a simple convolutional neural network (CNN) and the adjustment model is composed of convolutional layers that are interleaved with skip-connections. For the decoder, we adapt the multi-scale model for turbulent flows [FFT19], and the separate super-resolution network used in Dil-ResNet + SR and SOL + SR employs this same model. For all models, every convolutional layer except for the last one is followed by the Leaky ReLU activation function, except for Dil-ResNet which uses ReLU activations. We adopt circular padding for the periodic boundary condition problems and zero padding for the others. The architectures of the models are detailed in the appendix.

At each training iteration, for a given batch size, we randomly sample the initial states from the reference solution trajectories and integrate the approximate solution trajectories for \( N \) steps. All our trainings use an Adam optimizer [KB14] and a decaying learning rate scheduling.

5. Results

We evaluate the trained models based on relative improvements over a baseline simulation. To make the baseline solutions, we apply the reduced solver without interactions with any neural network and up-sample the reduced states into the reference space with a linear interpolation. Errors are computed with respect to the reference solutions, hence an improvement of 100% would mean that the restored solutions are identical to the reference. We evaluate each model using the mean absolute error (MAE) and mean squared error.
(MSE) metrics, which we measure in both velocity and vorticity. We present the results of the models trained with the highest number of integrated steps for each scenario, as they show better performance in general.

5.1. Reduced representations

The images of Fig. 4 show visual examples of the reduced representations for the Karman vortex street and forced turbulence scenarios, for different time-steps. The graphs of Fig. 5 show the quantified differences between the reduced states produced by the different trained models and the conventionally down-sampled reference states. We observe that our training procedure leads the latent representation to have vortex structures that are very similar to conventional down-sampling, while being considerably different quantitatively. Thus, we believe that the reduced representation of the ATO model stays physically meaningful for the numerical solver yet adds signals for accurately decoding high resolution states. We note that different training initializations of the same scenario produce latent representations that stay close to each other, which indicates that there exists a manifold of latent solutions that our ATO model converges to in order to get the best performance.

5.2. Karman vortex street

This example considers different vortex shedding behaviors depending on the Reynolds number of each simulation. We evaluate the models trained with 16 integrated steps on six test simulations with Reynolds numbers ranging from 450 to 1400, consisting of 2000 time-steps each. In this scenario, we test the extrapolation ability of the models both physically and temporally, with higher Reynolds number thus more turbulent simulations than for training, and ten times longer sequences.

Table 1 shows that ATO outperforms the other models with a relative improvement of 91% (and 88%) in terms of velocity MAE (and vorticity), while SOL + SR improves the baseline by 84% (and 83%). On the other hand, the Dil-ResNet + SR model fails to retrieve the target simulation for more than 200 time-steps, and thus is incapable of generalization in this scenario. The temporal metrics shown in the appendix demonstrate the capability of ATO to correctly restore a solution for longer time ranges than the other models. More specifically, the distance between the reduced states and lerp(ref) show that the ATO model is the only one to have a consistent latent representation over time, which proves its better temporal extrapolation capabilities.

5.3. Decaying turbulence

In this example, we consider initially chaotic turbulent flows that slowly decay over time. We evaluate the models trained with 16 integrated steps, on five random initializations, lasting 200 steps each.

Table 1 shows that the ATO model yields greatly improved results with a relative improvement of 83% (and 80%) in terms of velocity MAE (and vorticity). However, in this more simple case, the SOL + SR model also improves the baseline significantly with 82% (and 78%) of relative improvement. Dil-ResNet + SR, however, only yields 53% (and 6%) of improvement. Examples frames for this scenario are shown in the appendix.

5.4. Forced turbulence

This complex fluid flow scenario considers the same experimental setup as in the previous case but with external forces, which leads to highly chaotic turbulent flows. We evaluate the models trained with 16 integrated steps, on five random initializations both in velocity and forcing, for 200 steps.

© 2023 The Authors. Proceedings published by Eurographics - The European Association for Computer Graphics.
Table 1 shows that the ATO model significantly improves the baseline with a relative improvement of 74% (and 69%) in terms of velocity MAE (and vorticity). In comparison, SOL + SR improves by only 49% (and 43%) and Dil-ResNet + SR by 46% (and 38%). Therefore, in this complex case with external forcing and more turbulent flows, the ATO model particularly stands out. Fig. 6 (right) shows examples of high-resolution frames produced by each model along with the spatial distribution of the absolute error in velocity.

5.5. Smoke plume

In this last example, we consider complex flow behaviors created by hot smoke plumes that evolve from random circular densities. We evaluate our model trained with 32 integrated steps on five test simulations with different initial marker fields from which we perform a “warm-up” of 50 time-steps, in order to get interesting plume shapes.

Fig. 7 shows that, despite the increased difficulty of this challenging scenario, our ATO model succeeds at reconstructing a complex high-resolution plume of good quality. Indeed, our method presents an improvement of 35% on average over the baseline for 100 steps.

5.6. Ablation study

In order to see the effects of each of its components, we evaluate our ATO model with differently ablated training setups for the forced turbulence scenario. Our ablation study includes the following models:

- No latent loss: we remove the second term of the loss in Eq. 1; consequently, our training does not constrain the adjusted states to match the encoder-induced latent space.
- No encoder: we omit the encoder such that the latent representation is constrained to be conventional linear down-sampling.
- No encoder & no latent loss: since the previous model’s reduced space is constrained to linear down-sampling, we test the same setup without the encoder and with no latent constraint.
- No solver: we replace the solver + adjustment part of our ATO model with the Dil-ResNet NN-solver in order to study the effect of a non-physical latent space.
- No adjustment: we evaluate a setup where the reduced simulation evolves without being adjusted.
- lerp(forces): we input a simple linear down-sampling of the force fields to the reduced solver, instead of their encoded representation.

Table 2 shows that the encoder, physics solver, and adjustment components of our ATO model are essential for its good performance. Firstly, the no encoder and no encoder & no latent loss experiments confirm that, with lerp(ref) as initial reduced representation and without our encoder, the adjustment network was not able to find a latent representation that would lead to an optimal performance. Furthermore, the no latent loss ablation shows that the latent loss guiding the adjustment model via the encoder results in a better performance. Note that the performance of ATO significantly decreased when the encoder was absent, whereas the performance drop due to omitting the latent loss was relatively less significant. Secondly, the no solver and no adjustment experiments show that using a reduced physics solver in conjunction with an adjustment model is crucial for the good performance of our ATO model. Finally, the lerp(forces) experiment indicates that our encoder model failed to find a latent representation for the velocity that was compatible with an external factor conditioned to lerp.

All of the models tested in this ablation study gave comparable standard deviation values within the test set; thus, we did not include them in the table.

5.7. Runtime performance

For each scenario, we compare the runtime performance of the trained models with the reference’s, measuring timing for one simulation of 100 frames, averaged over ten different runs. For ATO, the computations start with the initial velocity inference by the encoder model and stop when all 100 frames are output by the decoder. All timings were computed using a single GeForce RTX 2080 Ti with 11GiB of VRAM.

Table 1 shows the summary of computational timings for the reference, baseline (reduced solver without any DNN model), and trained models. For all four cases, our ATO model yields improvement in runtime compared to the reference. For the Karman vortex street scenario, our ATO model speeds up the computations by 28%, against 29% for SOL+SR. Yet, as shown in Sec 5.2, ATO shows an improvement of the baseline MAE that is 7% better than SOL+SR. Similarly, for the forced turbulence case, the ATO model speeds up the computations by 18%, against 9% for SOL+SR, and improves the baseline MAE of 25% more than SOL+SR. For the smoke plume scenario, our ATO model speeds up the reference by 20% compared to 28% for the baseline, while improving the baseline MAE by 35%. We note that the Dil-ResNet + SR model often has the best runtime performance because it does not contain any numerical solver, but it has errors at least 50% higher than ATO and shows very poor temporal extrapolation capabilities.

Training our ATO model takes between one and three days depending on the physical scenario, on a Tesla V100 with 16GiB of VRAM.
6. Limitations and Future Work

These results show that our training method using the states of physics simulations as latent space of DNNs can facilitate the learning task for complex simulations. This provides a starting point for the exploration of physical latent spaces in many different problems. However, we note that our ATO model is not particularly standing out in a simple scenario like the decaying turbulence. Therefore, we can presume that the benefits of its unconventional reduced space are truly visible only when the PDE system is complex enough. In addition to the distance metric, more thorough analysis of latent space contents via, e.g., perceptual metrics, also remains our future work.

Moreover, our method has proven its capabilities in scenarios where force fields were inferred by our networks besides the velocity work. In the forced turbulence case, the forces were external factors that were independent from the velocity data, thus our ATO model had no difficulty finding a latent representation that led to a superior performance. In Sec. 5.5, we showed that our model gave promising results in a scenario where the forces were internal, i.e. created by a marker field that was dependent of the latent velocity. That case opens interesting future work, such as finding the best reduced representations for the coupled marker and velocity fields.

Although we evaluated our model on various scenarios, its generalization for broader applications still remains a challenge. As our model allows for the efficient production of high-resolution simulations with a reduced solver, it is potentially attractive for editing physics simulations within the learned reduced space in real-time. Indeed, once a coarse initial frame is transformed into ATO’s latent space, it is easy to tweak the physical properties of the reduced solver (e.g., viscosity) or to add external factors, such that it can produce high-resolution simulations in a more interactive way. Accordingly, the adaptation of our ATO setup for three-dimensional problems is a promising topic for future work.

7. Conclusion

We have presented ATO, a model that leverages interactions between neural networks and a differentiable physics solver to autonomously explore reduced representations for high-resolution fluid restoration purposes. Our results show that deep neural networks can learn to develop new dynamics for specific learning objectives by using the simulated degrees of freedom as latent space. Our approach opens the path to the exploration of physical latent spaces for other PDEs, as well as different learning tasks than the restoration of details of fluid simulations.

8. Acknowledgments

This project has been funded by the Futur & Ruptures PhD program of the Fondation Mines-Telecom.

References


