

# Fast Subspace Fluid Simulation with a Temporally-Aware Basis

SIYUAN CHEN, University of Toronto, Canada and Shanghai Jiao Tong University, China

YIXIN CHEN, University of Toronto, Canada

JONATHAN PANUELOS, University of Toronto, Canada

OTMAN BENCHEKROUN, University of Toronto, Canada

YUE CHANG, University of Toronto, Canada

EITAN GRINSPUN, University of Toronto, Canada

ZHECHENG WANG, University of Toronto, Canada

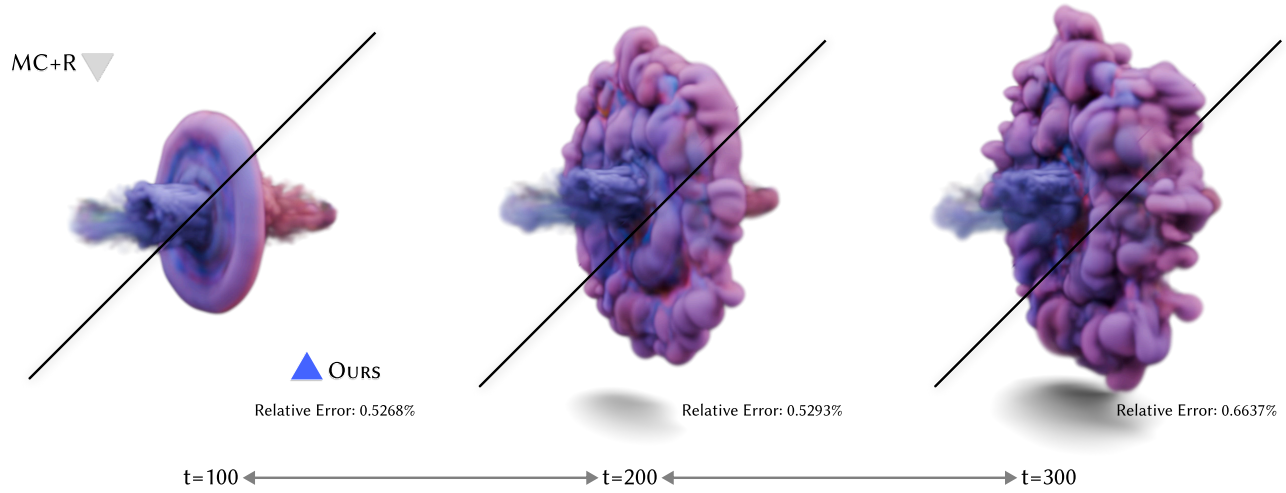


Fig. 1. **Accurate Koopman-based Fluid Prediction of Vortex Ring Colliding.** We compare (above) the ground truth MacCormack + Reflection (MC+R) solver [Zehnder et al. 2018] to (below) our reduced Koopman-based reconstruction, which leverages a reduced-order temporally-aware subspace with  $r = 150$  basis functions. Our approach achieves efficient, near-perfect prediction of arbitrary future fluid states via the  $k$ -th power of the eigenvalue matrix. In this experiment, parameter  $k = 100$ .

We present a novel reduced-order fluid simulation technique leveraging Dynamic Mode Decomposition (DMD) to achieve fast, memory-efficient, and user-controllable subspace simulation. We demonstrate that our approach combines the strengths of both spatial reduced order models (ROMs) as well as spectral decompositions. By optimizing for the operator that *evolves* a system state from one timestep to the next, rather than the system state itself, we gain both the compressive power of spatial ROMs as well as the intuitive physical dynamics of spectral methods. The latter property is of particular interest in graphics applications, where user control of fluid phenomena is of high demand. We demonstrate this in various applications including spatial and temporal modulation tools and fluid upscaling with added turbulence.

We adapt DMD for graphics applications by reducing computational overhead, incorporating user-defined force inputs, and optimizing memory usage with randomized SVD. The integration of OptDMD and DMD with Control

(DMDc) facilitates noise-robust reconstruction and real-time user interaction. We demonstrate the technique’s robustness across diverse simulation scenarios, including artistic editing, time-reversal, and super-resolution.

Through experimental validation on challenging scenarios, such as colliding vortex rings and boundary-interacting plumes, our method also exhibits superior performance and fidelity with significantly fewer basis functions compared to existing spatial ROMs. Leveraging the inherent linearity of the DMD formulation, we demonstrate a range of diverse applications. This work establishes another avenue for developing real-time, high-quality fluid simulations, enriching the space of fluid simulation techniques in interactive graphics and animation.

CCS Concepts: • **Computing methodologies** → **Modeling and simulation**; **Physical simulation**.

Additional Key Words and Phrases: Model-Reduction, Koopman Theory, Dynamic Mode Decomposition

Authors’ addresses: Siyuan Chen, siyuangraphics.chen@mail.utoronto.ca, University of Toronto, Toronto, Canada and Shanghai Jiao Tong University, Shanghai, China; Yixin Chen, yixinc.chen@mail.utoronto.ca, University of Toronto, Toronto, Canada; Jonathan Panuelos, jonathan.panuelos@mail.utoronto.ca, University of Toronto, Toronto, Canada; Otman Benckekroun, otman.benckekroun@mail.utoronto.ca, University of Toronto, Toronto, Canada; Yue Chang, changyue.chang@mail.utoronto.ca, University of Toronto, Toronto, Canada; Eitan Grinspun, eitan@cs.toronto.edu, University of Toronto, Toronto, Canada; Zhecheng Wang, zhecheng@cs.toronto.edu, University of Toronto, Toronto, Canada.

## 1 INTRODUCTION

Fast, responsive, and visually realistic fluid simulation remains a significant challenge in physics-based animation. Traditional full-resolution simulations deliver highly detailed fluid flows but are

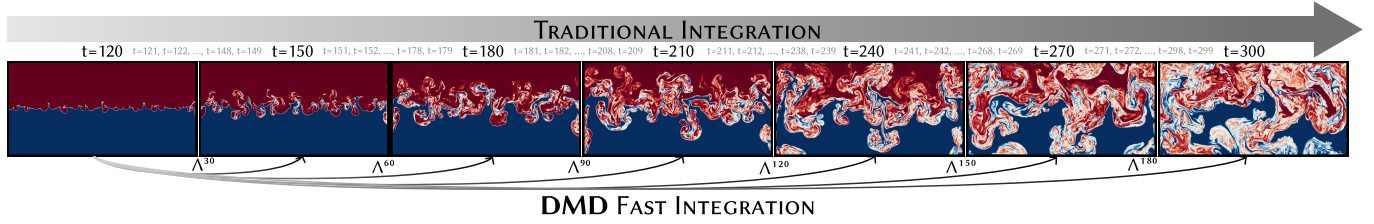


Fig. 2. **Long-time Single Step Integration.** We demonstrate that our method can perform integration into arbitrary points in time via the *exponential integration* of a single matrix (Sec. 4.4). Since the DMD operator is diagonal within the reduced basis, it is trivial to find the matrix that evolves the initial velocity field to the field at any point in time, significantly accelerating the integration as compared to traditional methods required by PCA. Although the DMD operator is trained over the velocity field  $\mathbf{u}$ , we show the corresponding density field at each time point for clearer visualization.

computationally expensive, limiting their use to offline cinematic visual effects rather than interactive virtual reality applications.

Reduced-Order Models (ROMs) address this challenge by accelerating simulations through dimensionality reduction. In the context of fluid dynamics, ROMs approximate the *full, high-dimensional* fluid simulation solution space by effectively operating within a *reduced, lower-dimensional* subspace. The dynamics of the reduced fluid simulation are then obtained by projecting the high-dimensional solution space into the reduced space and solving the physical Partial Differential Equation (PDE) within this subspace. As a result, the subspace needs to provide a compressed representation of the fluid state, while also capturing realistic and highly detailed fluid behavior.

Picking which subspace to use for simulation is non-trivial, and after committing to a subspace, using it for reduced simulation opens up separate difficulties. First, subspace fluid simulation is notorious for dissipating high-frequency detail, the kind that is commonly desired in turbulent flows. Second, it involves computing a static subspace that can generalize to a diversity of simulation states expected across a variety of scene interactions and configurations. Finally, even if one makes use of a linear subspace for simulation, the advection component of inviscid Euler equations is non-linear with respect to the fluid state, which requires full-space computation even with the use of a precomputed subspace.

Instead of committing to the static subspace methodology, we make use of Dynamic Mode Decomposition (DMD) [Schmid 2010], a modern flow analysis technique that takes an alternative simplification to the fluid simulation problem. This formulation linearly approximates the *Koopman operator*, which encodes the temporal evolution of the fluid flow, and performs reduction directly on this operator [Schmid 2010]. An immediate advantage of reduction on this operator is that the resulting subspace is imbued with information regarding the temporal dynamics of the flow. This temporal awareness allows us to quickly evaluate the fluid flow at *any* point in time directly, without performing any time integration or discrete fluid advection whatsoever, as shown in Fig. 2.

While DMD has shown great success for use in flow analysis from the engineering community, we are the first to show how it can be adapted for use in fluid animation applications in graphics, such as fluid editing, guiding, interaction, and artistic fluid control.

Our key insight is to leverage the inherent spatio-temporal nature of the linear DMD operator. By encoding *both* the *spatial* nature of flows in the eigenvector basis and their *temporal* evolution in the

eigenvalues, each eigenvector-eigenvalue pair becomes representative of a distinct wave mode.

This spectral-like decomposition allows for directable control of each mode separately, enabling new forms of creative expression, such as adjusting the amplitude of temporal frequency bands to control look-and-feel in interactive editing of fluid animation.

**Contributions.** In summary, in this paper we:

- introduce Dynamic Mode Decomposition (DMD) for fast control of fluid simulations with exceptional accuracy, computational speed, and memory efficiency;
- identify that the inherent spatiotemporal nature of the DMD operator enables control of each wave mode by modification of eigenvector-eigenvalue pairs;
- demonstrate DMD’s versatility and practicality in the context of interaction and artistic control, such as frequency editing, time-reversal, super-resolution and simulation styling.

## 2 RELATED WORKS

Following pioneering work by Foster and Metaxas [1996], Stam [1999], and Fedkiw et al. [2001], the graphics community has made significant progress in the visual simulation of fluids. Our method straddles literature from both fluid control and runtime acceleration. From foundational work, we aim to demonstrate that DMD arises naturally as a combination of spatial reduced-order modelling and spectral decompositions, a unique position that makes it well suited for fluid control applications.

### 2.1 Fluid Control

The chaotic nature of fluid behaviour makes artist-directed control a particularly difficult problem, as it often requires balancing accuracy, efficiency, and the user intuition. Of particular interest for us are applications in fluid snapshotting, where control *frames* are provided that fluid flows into and out of. Numerous prior works have been done using optimization-based techniques, aiming to compute optimal control forces that guide fluid flows to match user-specified keyframes [Inglis et al. 2017; McNamara et al. 2004; Pan and Manocha 2017; Thürey et al. 2009; Treuille et al. 2003]. However, the high-dimensional optimization problems make most prior methods computationally expensive, limiting their application to real-time and interactive control. To further improve the efficiency, reduced-order fluid control methods have been proposed, such as



frequency-aware force field reduction [Tang et al. 2021] and control with Laplacian Eigenfunctions [Chen et al. 2024]. In this work, we demonstrate that the reversible nature of the DMD operator makes it particularly suitable for these problems. Time-reversibility has been prior explored in graphics [Oborn et al. 2018], but requires the solution of a modified Poisson problem. In comparison, we demonstrate that the DMD operator is trivially reversible.

We also take influence from resolution-based smoke control [Nielsen et al. 2009; Sato et al. 2021], which uses an artist-directed low-resolution input to produce a high-resolution simulation with secondary motion, such as smaller-scale turbulence. We demonstrate that the modes provided by DMD allow for direct application to these problems. We also similarly can preserve the input features via projection into the input velocity space similarly to Nielsen et al. [2009], but show that our quadratic problem is precomputable and resolves into a simple matrix-vector multiply at runtime.

## 2.2 Fast Fluid Simulation

A major computational bottleneck in most Navier-Stokes (NS) solvers is the global pressure projection step, which is computationally expensive when solved directly, especially for high-resolution simulations. To mitigate this cost, iterative solvers such as Conjugate Gradient (CG) [Saad 2003] and multigrid methods [Briggs et al. 2000] are commonly employed to exploit the sparsity structure of the linear operators.

Avoiding this global pressure projection step is a key focus of real-time methods. To this end, Smoothed-Particle Hydrodynamics (SPH) [Müller et al. 2003], and later Position-Based Fluids (PBF) [Macklin and Müller 2013], have become the *de facto* standard for GPU-based fluid simulation. More recently, the Lattice Boltzmann Method (LBM) has emerged as a powerful grid-based alternative [Chen and Doolen 1998; Chen et al. 2021; Li et al. 2020]. All these methods avoid a global solve, and are thus embarrassingly parallelizable and highly suitable for modern GPU architectures. In exchange, however, these methods often require extremely high resolutions and equivalently high memory consumption during runtime. By comparison, our method also avoids this global solve, evolving the system via a single matrix vector multiply, while simultaneously keeping the memory footprint low via reduction to a small basis.

## 2.3 Spatial Order Reduction

In addition to the above parallelization approaches, reduced-order models present another promising direction for accelerating simulations. These methods aim to simplify the underlying computational models while preserving essential physical dynamics, thus improving speed and scalability. ROMs generally fall into two categories: *data-free* and *data-driven* methods.

**2.3.1 Data-free ROMs.** Data-free ROMs rely on simplified physics-based formulations to reduce computational complexity without requiring precomputed datasets. Laplacian Eigenfluids [Cui et al. 2018; De Witt et al. 2012; Liu et al. 2015] has emerged as a class of data-free reduced-order methods for fluid simulations, representing states using a set of basis functions that best span a given spatial domain.

This approach demonstrates the feasibility of order reduction for fluid simulation. However, because their basis attempts to represent *any* divergence-free field, handling intricate boundaries or capturing fine details often requires a large number of basis functions, and correspondingly large runtime and memory. If the general type of flow is known *a priori*, which is often the case when an artist wants to apply simulation tools, these priors can be a powerful tool for improving computational efficiency.

**2.3.2 Data-driven ROMs.** Data-driven ROMs, in contrast, leverage statistical techniques and precomputed datasets to approximate complex fluid dynamics. Essentially, precomputed data already encodes essential structures of the fluid flow; one only needs to extract that flow and simplify the representation elsewhere.

Treuille et al. [2006] used principal component analysis (PCA) to find this reduced basis by minimizing reconstruction error given some target dimension. Wicke et al. [2009] introduced spatial generalization by replacing the global basis with ‘tiles’ of local bases coupled with shared boundary bases. A particular drawback for existing data-driven methods is the lack of intuition for these spatial bases. This makes it difficult to pick out individual bases and modify them to manipulate the resulting fluid flow, a key application that we aim to tackle. General adoption has thus been limited for existing data-driven ROMs, and are largely relegated to playback.

## 2.4 Spectral Methods

A key strength of spectral methods is the physical intuition of the basis functions. Whereas spatial order reductions construct basis functions based on the domain shape and are thus more physically intuitive for non-advection phenomena such as elastic modes [Brandt and Hildebrandt 2017; Sellán et al. 2023], spectral methods construct basis functions based on how the field evolves over time, leveraging the periodicity of fluid flows to represent the system’s dynamics in the frequency domain. Each mode thus corresponds to the propagation of a different wave, or coherent groups of waves. Kim et al. [2008] leveraged this to add smaller-scale turbulence as a post-processing step, and Chern et al. [2017] used this to directly modify vortex rings. This type of fluid control greatly motivates much of our application domains presented in Sec. 6.

Prominently among these are Fourier-based spectral methods. Stam [2001] solves stable fluids in Fourier space, though only in a periodic domains due to the global nature of Fourier bases. In contrast, wavelet-based methods localize these bases into individual waves [Jeschke et al. 2018], allowing for better expressivity as well as multiscale features. Fourier methods also greatly simplify advection, which becomes a linear transform in Fourier space [Chern et al. 2016], further speeding up computation. We adopt a similar scheme, where advection is encoded directly in our DMD operator.

## 2.5 Dynamic Mode Decomposition

DMD is a data-driven method that approximates the Koopman operator, a linear operator that describes the temporal evolution of a dynamical system [Schmid 2010]. Prior usage of DMD for fluids application has been limited, and have not been explored at all in computer graphics and creative applications. Usage has primarily been limited to playback and short-term forecasting [Proctor et al.

2016]. Outside of fluid simulation, DMD has been used to process large datasets in high-dimensional systems [Williams et al. 2015] and construct usable operators from noisy data [Askham and Kutz 2018]. This latter extension, called OptDMD, is of significant interest to us, as the turbulence inherent in fluid fields naturally produces noisy training data.

Similar to other data-driven ROMs, DMD constructs spatial bases that encode the different flows present in the training set. Because it approximates the *operator* that evolves a state forward in time rather than the state itself, it also gains the advantages of spectral methods. In particular, each basis becomes associated with different turbulent scales, opening the door to fluid control and upscaling applications. By computing the eigenvalues and eigenvectors of the Koopman operator, DMD can identify the dominant modes of a system and predict its future behavior.

We propose the use of Dynamic Mode Decomposition (DMD) as a tool for compressing and manipulating fluid simulations, combining the spatial compressive power of data-driven methods with the physical intuition and control of spectral methods.

### 3 KOOPMAN OPERATOR AND DYNAMIC MODE DECOMPOSITION

Consider a system of the form,

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(\mathbf{u}), \quad (1)$$

where  $\mathbf{u}(\mathbf{x}, t) \in \mathbb{R}^N$  is a time-varying state vector in  $\mathbf{x} \in \mathbb{R}^d$ ,  $t \in \mathbb{R}$ , and  $\mathbf{f} : \mathbb{R}^N \rightarrow \mathbb{R}^N$  is a nonlinear operator. Since the right-hand side is not explicitly dependent on time, this *autonomous* system evolves according to the current state  $\mathbf{u}$ , not when  $\mathbf{u}$  is evaluated.

Suppose we collect a finite dictionary of observables, and stack them to yield the measurement vector,

$$\mathbf{g}(\mathbf{u}(t)) = \begin{bmatrix} g_1(\mathbf{u}(t)) \\ g_2(\mathbf{u}(t)) \\ \vdots \\ g_n(\mathbf{u}(t)) \end{bmatrix} = \begin{bmatrix} \mathbf{u}(\mathbf{x}_1, t) \\ \mathbf{u}(\mathbf{x}_2, t) \\ \vdots \\ \mathbf{u}(\mathbf{x}_n, t) \end{bmatrix} \in \mathbb{R}^{Nn}, \quad (2)$$

where  $n$  is the total spatial degrees of freedom of our measurements, and we take  $T \leq n+1$  independent measurements of  $\mathbf{g}(\mathbf{u})$  at different times. There exists an operator  $\mathbf{K}$  on this basis that applies the following *linearized* time-stepping operation [Brunton et al. 2022],

$$\mathbf{g}_{k+1} = \mathbf{K} \mathbf{g}_k, \quad (3)$$

where the subscripts  $\{k \in \mathbb{N} \mid 0 \leq k < T\}$  indicate the time level of a given measurement.  $\mathbf{K}$  is known as the restricted Koopman operator, and is guaranteed to exist and exactly recover the given input measurements as long as the number of time samples  $T$  is fewer or equal to  $n + 1$ , where  $n$  is the size of the spatial degrees of freedom. Moving forward, for practical use, we only require the understanding that  $\mathbf{K}$  exists, and we defer discussion of Koopman theory on why this is true to App. A.

#### 3.1 Approximating the Koopman Operator with Dynamic Mode Decomposition

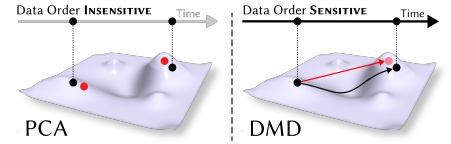
The restricted Koopman operator  $\mathbf{K}$  has a basis with dimension on the number of degrees of freedom of the data (*i.e.* the number

of grid points). In most reasonable simulation resolutions, this is computationally intractable, considering  $\mathbf{K}$  is almost certainly dense in this grid basis.

To address this, we follow a strategy similar to prior dimensionality reduction approaches such as PCA [Kim and Delaney 2013; Treuille et al. 2006], and instead approximate the Koopman operator using *Dynamic Mode Decomposition* (DMD) [Schmid 2010], constructing a *reduced, low-dimensional* basis to represent the function space spanned by the input observables. Note that while  $\mathbf{K}$  will exactly reproduce input measurements (and reproduce the continuous system in Eq. (1) in a least-squares sense on the finite measurement basis), the dimensionality reduction offered by DMD serves to approximate  $\mathbf{K}$ , and thus will introduce extra error in exchange for computational tractability.

As illustrated on the right, a key distinction between PCA and DMD is their relationship to time. Here, two consecutive snapshots are shown as two black dots (states recorded by frames) connected by a timeline. Conventional *PCA* is data order insensitive as its singular-value decomposition depends only on the covariance of the snapshot set, so re-ordering the frames leaves the low-rank surface where the projected states (red dots) reside unchanged. *DMD*, in contrast, fits an

operator  $\mathbf{K}$  that advances the earlier state to the next predicted state



(hollow red dot), as indicated by the red arrow. Because this least-squares fit relies on aligned pairs of samples, shuffling the data alters  $\mathbf{K}$ . This built-in temporal sensitivity lets DMD capture system dynamics rather than merely the geometry of the solution manifold, making it better suited for modelling time-evolving systems. Intuitively, PCA models a subspace of the solution space, whereas DMD models the operator that maps a state to the next—an objective that naturally requires temporal information.

Given two *time-shifted* sequence of snapshots  $\mathbf{X}$  and  $\mathbf{X}'$ , a discrete Koopman operator is defined by:

$$\begin{aligned} & \arg \min_{\mathbf{K}} \|\mathbf{X}' - \mathbf{K}\mathbf{X}\|_F, \\ & \text{where } \mathbf{X} = \underbrace{\begin{bmatrix} \mathbf{u}(0) & \mathbf{u}(1) & \cdots & \mathbf{u}(T-1) \end{bmatrix}}_{T \text{ Frames}}, \\ & \mathbf{X}' = \underbrace{\begin{bmatrix} \mathbf{u}(1) & \mathbf{u}(2) & \cdots & \mathbf{u}(T) \end{bmatrix}}_{T \text{ Frames}} \end{aligned} \quad (4)$$

Intuitively, this optimizes  $\mathbf{K} \in \mathbb{R}^{(Nn) \times (Nn)}$  to minimize the difference between the next states  $\mathbf{u}(t + \Delta t)$  and its prediction of the next states  $\mathbf{K}\mathbf{u}(t)$  under a linear model.

We can solve this minimization as follows:

$$\mathbf{K} = \mathbf{X}' \mathbf{V} \mathbf{\Sigma}^{-1} \mathbf{U}^T, \quad (5)$$

where  $\mathbf{X} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T$  is the singular value decomposition (SVD) of the snapshot matrix  $\mathbf{X}$ , with  $\mathbf{V} \in \mathbb{R}^{(Nn) \times r}$  being the left eigenvectors,  $\mathbf{\Sigma}$  being the eigenvalues matrix, and  $\mathbf{U} \in \mathbb{R}^{(Nn) \times r}$  being the right

eigenvectors. We can truncate this eigensystem, taking the top  $r$  singular values corresponding vectors to form the truncated Koopman operator  $\tilde{K}$ . We can then project this truncated operator to the reduced basis  $U^T \tilde{K} U = U^T X' V \Sigma^{-1} U^T U = \tilde{K}$  to produce the reduced Koopman operator  $\tilde{K}$ .

We can see that as long as a mode is representable in the reduced space,  $\tilde{K}$  and  $\tilde{K}$  share that mode's eigenvalue,  $\tilde{K} \mathbf{w}_i = \lambda_i \mathbf{w}_i$ , with the eigenvector simply being  $\tilde{K}$ 's eigenvector projected onto the reduced space,  $\mathbf{w}_i = U^T \phi_i$ . Here,  $\lambda_i$  is a mode's eigenvalue,  $\phi_i$  is its corresponding eigenvector in full space, and  $\mathbf{w}_i$  is its projection onto reduced space. We thus can find a spectral decomposition of the truncated Koopman operator:  $\tilde{K} = \Phi \Lambda \Phi^*$ .

Since applying the truncated Koopman operator  $\tilde{K}$  on the full space has complexity  $O(N^2 n^2)$ , we *project* the full-space velocity field  $\mathbf{u}$  onto the reduced space spanned by the basis  $\Phi$ :

$$\mathbf{u}(t + \Delta t) = \tilde{K} \Phi \Phi^+ \mathbf{u}(t) = \tilde{K} \Phi \mathbf{z}(t), \quad (6)$$

where  $\Phi^+ = (\Phi^* \Phi)^{-1} \Phi^* \in \mathbb{R}^{(Nn) \times r}$  is the Moore-Penrose pseudoinverse of  $\Phi$  and  $\mathbf{z}(t) = \Phi^+ \mathbf{u}(t) \in \mathbb{R}^r$  is the reduced state of the fluid system at time  $t$ . This projection step reduces the complexity to  $O(Nnr)$ .

Notice now that taking  $\Lambda = \Phi^* \tilde{K} \Phi \in \mathbb{R}^{r \times r}$  gives exactly a matrix that advances the reduced state forward in time by  $\Delta t$ :

$$\mathbf{z}(t + \Delta t) = \Lambda \mathbf{z}(t), \quad (7)$$

We note that  $\Lambda$  is the diagonal matrix of eigenvalues of  $\tilde{K}$ , and  $\Phi$  are the corresponding eigenvectors. That is to say,  $\tilde{K} = \Phi \Lambda \Phi^*$  is exactly the spectral decomposition of the reduced Koopman operator. This low-rank model allows linear simulation of the reduced state, offering an efficient surrogate for the nonlinear underlying system.

From here, we can once again simply apply the basis  $\Phi$  to return back to full space:

$$\mathbf{u}(t + \Delta t) = \Phi \Lambda \mathbf{z}(t). \quad (8)$$

We would like to point out that despite both  $\Lambda$  and  $\tilde{K}$  are referred to as Koopman operators in the literature, we remark that in our formulation they act on a subspace of *linear* observables derived from state components. Therefore, the dynamics we capture are strictly linear, and our model is best interpreted as a linear approximation of the full underlying nonlinear system.

$\Lambda$  in particular gives particular insight into the theory; notice here that it acts on the function space of  $g = \Phi$ . Each eigenfunction is thus a special observable, which clearly behaves linearly by definition. Since we know that  $\Phi^*$  maps observables to observables,  $\tilde{K}$  must necessarily also be a Koopman operator, this time acting on the identity operator (or more accurately, the  $\mathbf{u}$  observable in Hilbert space).

In addition, the  $\Lambda$  representation exposes the modes in an easily manipulatable manner. As a Koopman operator, it represents the time evolution of some observable. By being a diagonal matrix of eigenvalues, it turns out that these observables are exactly spatial modes that rotate with a particular frequency given by the imaginary part of the eigenvalues.

## 4 ADAPTING DMD FOR GRAPHICS APPLICATIONS

### 4.1 Constructing Training Data

We point out that the system solved by Eq. (4) is agnostic to the structure of the input state  $\mathbf{u}$ . As stated, the DMD operator learns a mapping that evolves the state  $\mathbf{u}$  from one point in time to its state at a future moment, and the natural description of a fluid state is to represent  $\mathbf{u}$  as a stacked vector of fluid velocities, which implies that the DMD operator learns the *PDE* that governs the evolution of the fluid velocities over time.

For NS solvers, the evolution of a fluid is described by the inviscid Euler equations,

$$\begin{cases} \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\frac{1}{\rho} \nabla p + \mathbf{g} \\ \nabla \cdot \mathbf{u} = 0 \end{cases} \quad (9)$$

where  $\mathbf{u}$  is the velocity field,  $p$  is the pressure,  $\rho$  is the density, and  $\mathbf{g}$  is the gravitational force. We discretize the domain  $\Omega$  with the staggered grid, a standard approach in computer graphics [Harlow et al. 1965], where the velocity field  $\mathbf{u}$  is defined at the cell faces and the pressure  $p$  is defined at the cell centers. This thus provides snapshots of the fluid velocity field  $\mathbf{u}(t)$ , at any point in time  $t$ . Note that  $\mathbf{u}$  is a vector of size  $\mathbb{R}^{Nn}$ . This vector space henceforth will be our *fullspace*, as it includes spatially every point in our full resolution simulation.

Since our training data can be generated from any simulation method (or even observed from the real world), we utilize several fluid simulation algorithms to construct our dataset. Specifically, we include the standard stable fluids [Stam 1999], the MacCormack [Selle et al. 2008] (MC), the MacCormack + Reflection [Zehnder et al. 2018] (MC+R) and the Lattice Boltzmann Method with the Bhatnagar-Gross-Krook collision model [Chen and Doolen 1998] (LBM-BGK).

### 4.2 Learning from Noisy Data with Nonlinear Optimization

Fluid behaviour is inherently chaotic, particularly in highly turbulent systems. Additionally, discretization of such states leads to noisy data, particularly as structures approach the Nyquist limit of the sampling grid. Further, simulation speed and artist directability is of primary concern in graphics application above simulation accuracy, often leading to simulations with CFD-condition-violating large timesteps and early termination of iterative algorithms. This further leads to degraded simulation results. As such, we note that input fluid data in general will be highly noisy.

Standard DMD, as shown in Eq. (5), directly fits the data from the  $i$ -th frame to the  $(i+1)$ -th frame without considering the (potentially coherent) relationships between the  $i$ -th frame and the  $(i+2)$ -th or subsequent frames. This short time horizon makes it highly sensitive to noise, both spatially as the fluid state changes rapidly at the Nyquist limit, and at the operator level. As a result, standard DMD imposes high requirements on data quality.

To address this issue, we choose to use OptDMD [Askham and Kutz 2018] for our graphics applications. OptDMD processes all snapshots simultaneously, significantly reducing the impact of noise by looking at the signal over a longer time horizon. By considering the  $i$ -th frame, the  $(i+1)$ -th frame, the  $(i+2)$ -th frame, and so on

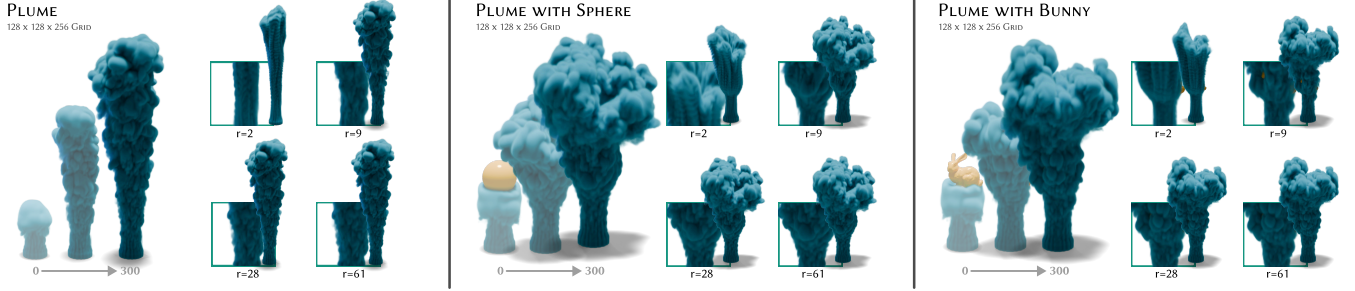


Fig. 3. **Reconstruction of 3D Plume Simulations.** From left to right: standalone plumes, plumes interacting with a sphere, and plumes interacting with a bunny. For each configuration, the last frame of the original MacCormack [Selle et al. 2008] fluid simulation is shown on the left, alongside the last frame of our subspace simulation with SVD rank ranging from  $r = 2$  to  $r = 61$ . Remarkably, the fluid dynamics demonstrate strong resilience to low-rank bases, highlighting a key advantage of our proposed reduced-order pipeline. Additionally, these scenarios illustrate the robustness of our method in handling increasingly complex boundary conditions. The reconstruction quality improves as the number of basis functions increases, enabling more accurate capture of finer details around the boundaries. While reasonable results are achieved with a small basis ( $r=9$ ), increasing the basis significantly enhances the fidelity of the simulations. More details on the temporal evolution of the flow and other basis configurations can be found in the Fig. 10, Fig. 11 and Fig. 12.

together, the influence of random noise is mitigated. To achieve this, OptDMD transforms the training of the reduced Koopman operator into an exponential data fitting problem and utilizes the variable projection method to solve this optimization problem. Similar to the procedure described in Sec. 3, we first perform Singular Value Decomposition (SVD) on the snapshot matrix  $X$  (see Eq. (4)) to obtain  $U, \Sigma, V$  such that  $X = U\Sigma V^T$ . Here,  $\Sigma$  is the diagonal singular value matrix, and  $U, V$  are orthonormal matrices whose columns are the left and right singular vectors, respectively.

With this decomposition, we can construct the optimization problem as:

$$\arg \min_{\alpha, B} \|\bar{V}\Sigma - \Phi(\alpha)B\|_F \quad (10)$$

where  $\bar{V}$  denotes the element-wise complex conjugate of  $V$ ,  $\Phi(\alpha)$  is the parameterized basis matrix with optimizable parameters  $\alpha$ , and  $B$  is the coefficient matrix corresponding to the basis matrix. Here  $\Phi(\alpha) \in \mathbb{C}^{T \times r}$  is the Vandermonde matrix whose  $(i, j)$ -entry is  $[\Phi(\alpha)]_{ij} = \alpha_j^{i-1}$  for  $i = 1, \dots, T$  and  $j = 1, \dots, r$ ; the vector  $\alpha = [\alpha_1, \dots, \alpha_r]^T$  collects the (discrete-time) DMD eigenvalues  $\alpha = \Lambda$ , initialized from the eigenvalue matrix  $\Lambda$  of a standard DMD solve (Eq. (4)).

Once the coefficient matrix  $B$  is estimated, we could compute each column of the basis matrix  $\Phi$  using:

$$\phi_i = \frac{U_r B^T(:, i)}{\|U_r B^T(:, i)\|_2} \quad (11)$$

where  $\phi_i$  denotes the  $i$ -th column of matrix  $\Phi$ .

#### 4.3 Memory Overhead Optimization

DMD can effectively reconstruct datasets, but its training process requires a significant amount of memory due to the SVD. When the dataset is large, it becomes necessary to store an  $Nn \times Nn$  matrix in memory, where  $Nn$  represents the dimensionality of a snapshot, thus demonstrating a significant memory requirement.

We note that previous DMD literature either has much smaller datasets afforded by 2D data, or perform training on large clusters [Askham and Kutz 2018; Proctor et al. 2016; Sashidhar and Kutz

2022; Schmid 2010]. Uniquely in graphics, we expect algorithms to be able to simulate 3D examples (i.e. large degrees of freedom) but run on consumer hardware (i.e. limited memory).

To reduce the memory requirements of DMD, we employ randomized SVD. Randomized SVD approximates the range of the original matrix  $X$  by constructing a matrix  $Q$ . This is done by multiplying a randomly initialized low-dimensional matrix  $Q$  with the original matrix  $X$  and performing decompositions iteratively until a stable vector matrix  $Q$  is obtained. The goal is to ensure that  $X \approx QQ^T X$ . We then construct the matrix  $B = Q^T X$ , which applies to the space spanned by  $Q$ . Traditional SVD can thus be performed on the much smaller  $B$  to obtain  $B = U\Sigma V^T$ , and finally be used to approximate  $X \approx QU\Sigma V^T$ .

#### MODULI EDITING 128 x 128 x 256 Grid

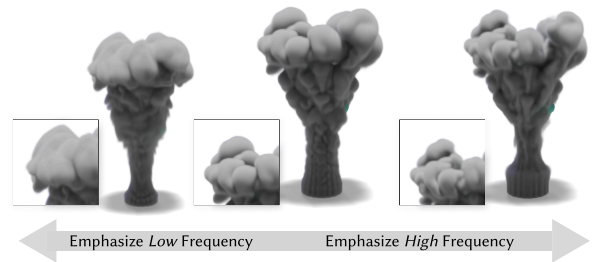


Fig. 4. **Editing Temporal Dynamics of the Plume with Bunny with the Koopman Operator Approximation.** In this experiment, we demonstrate the impact of changing the moduli of low-frequency and high-frequency modes in a 4:1 (left) and 1:4 (right) ratio. As mentioned in Sec. 6.1, in real applications, users can modify and manipulate the dynamic of different scales of vorticity by adjusting the reduced-order parameters.

#### 4.4 Arbitrary Time Step

Since the reduced Koopman operator is a linear operator, as shown in Fig. 2 we could easily pre-compute the power of the reduced



Koopman operator and apply it to a reduced state  $z$ , enabling *large* time step  $k$  times of the time step  $\Delta t$  that  $A$  is trained on:

$$\mathbf{u}(t + k\Delta t) = \Phi \Lambda^k z(t) \quad (12)$$

where  $k$  is an integer of how many time steps we want to simulate forward. This can map any state at time  $t$  to the state at time  $t + k\Delta t$  in **one** matrix multiplication.

Notice additionally that because the operator maps velocity fields to velocity fields, advection of the velocity variable is achieved for *free*. Nonlinear advection of the field is encoded directly in the operator.

#### 4.5 Boundary Conditions and Divergence Free Constraint

The truncated Koopman operator  $\tilde{K}$  is trained on a full-space velocity field  $\mathbf{u}$  which satisfies some boundary conditions and the divergence free constraint. We follow a similar proof to that provided by Treuille et al. [2006] to demonstrate that the Koopman operator also satisfies the space boundary conditions and constraints.

First, the boundary conditions and constraints can be represented as some linear constraint matrix  $C$  on the time series data  $X$  such that  $CX = 0$ . Any column  $\mathbf{b}_i$  in  $U_r$  by definition satisfies  $XX^T \mathbf{b}_i = \lambda_i \mathbf{b}_i$ . We can then conclude that  $CXX^T \mathbf{b}_i = \lambda_i C \mathbf{b}_i = 0$ , and thus claim that  $\mathbf{b}_i$  satisfies the constraint matrix as long as  $\lambda_i \neq 0$ .

As this is true for all columns, then  $C U_r = 0$  and consequently  $C \Phi = 0$ . This means that any velocity field  $\mathbf{u}$  reconstructed from its reduced representation satisfies any linear constraint satisfied by the initial data.

#### 4.6 Encompassing External Forces in Reduced Space

The OptDMD method in CFD can only reconstruct the dataset and cannot respond to user inputs of new external forces. However, in graphics applications, it is very common to edit or manipulate existing datasets, such as adding or reducing forces. To address this, we combine DMD with Control (DMDc) [Proctor et al. 2016] and OptDMD, marking the first instance of integrating these two methods. This integration allows user-input external forces to be incorporated into the OptDMD framework. Our improved framework is as follows:

$$\mathbf{z}(t + \Delta t) = \Lambda \mathbf{z}(t) + \Phi^+ \sum_{i=1}^m \mathbf{B}_i \mathbf{q}_i(t) \Delta t + \Phi^+ \sum_{j=1}^n \mathbf{f}_j(t) \Delta t \quad (13)$$

where  $\mathbf{B}_i$  and  $\mathbf{q}_i$  can be constructed differently based on the settings of various scenarios.  $\mathbf{f}_j \in \mathbb{R}^{(Nn) \times 1}$  represents a user-modifiable external force. For example, in the 2D standalone plume scenario, we can define  $\mathbf{q}_1 \in \mathbb{R}^{M \times 1}$  as the density field, where  $M$  represents the number of grid cells, and rewrite its impact on the velocity field in matrix form  $\mathbf{B}_1 \in \mathbb{R}^{(Nn) \times M}$ . Similarly, we can define  $\mathbf{q}_2 \in \mathbb{R}^{M \times 1}$  as the temperature field, and rewrite its impact on the velocity field in matrix form  $\mathbf{B}_2 \in \mathbb{R}^{(Nn) \times M}$ . These effects can then be projected onto the reduced space as  $(\Phi^+ (\mathbf{B}_1 \mathbf{q}_1 + \mathbf{B}_2 \mathbf{q}_2))$ .

## 5 EVALUATION AND RESULTS

In this section, we demonstrate the results of our approach on baseline examples and provide detailed numerical evaluations. We conduct all numerical experiments on a Linux workstation equipped

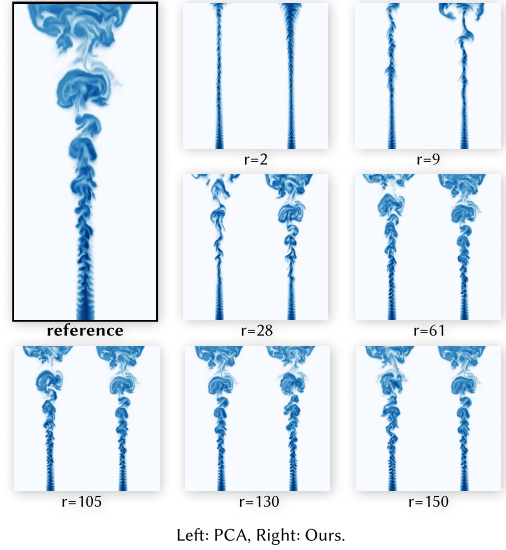


Fig. 5. **Comparison of PCA and Our Method for Low-Rank Flow Reconstruction.** The leftmost column shows the reference high-resolution simulation, while the right grid presents reconstructions at different rank truncations ( $r$ ). Each pair in the grid compares Principal Component Analysis (PCA) (left) and our method (right). Lower ranks ( $r = 2, 9$ ) fail to capture large-scale turbulence while increasing  $r$  improves accuracy. Our method retains more detailed structures at lower ranks than PCA, demonstrating improved efficiency in capturing complex flow dynamics.

with a 2.10GHz 32-core Intel CPU featuring 120GB of RAM. We implement our method exclusively in Python, and our DMD implementation is a heavily optimized, memory-efficient version of PyDMD [Ichinaga et al. 2024]. We use Taichi [Hu et al. 2019] for interactive examples and parallelization of DMD model inference, and NumPy [Harris et al. 2020] for numerical validation. Please refer to Tab. 2 for the experiment setup of the examples presented in this section.

#### 5.1 Comparison of 2D Plume Reconstruction

To evaluate the reconstruction performance of our method in comparison with the prior work [Kim and Delaney 2013], we conducted experiments on a 2D plume scenario using varying numbers of basis functions: 2, 9, 28, 61, 105, 130, and 150; shown on Fig. 5. With only 9 basis functions, our method is already capable of reconstructing the contour of the plume's top, while the prior method can only capture basic upward motion of the plume structure without any vortical structure details. This shows the superior reconstruction capability of our method even at low ranks. When using 28 basis functions, our method closely matches the ground truth, whereas the prior method still misses many details, particularly in the upper regions of the plume. When the number of basis functions is increased to 61, our method achieves a near-perfect reconstruction of the ground truth, and the prior method requires 130 basis functions to produce a similar level of detail. These results effectively demonstrate that our method can achieve better reconstruction with significantly fewer basis functions.

**RUNTIME: 2D PLUME**  $256 \times 512$  GRID ( $\triangleright$  Fig. 5)

<b>Fullspace Solve</b>	754 ms
<b>Subspace Solve</b>	
PCA-based [2013]	16.15 ms (47 $\times$ ) / 0.95 ms (794 $\times$ ) <sup>†</sup>
<b>Ours</b>	<b>8.9 <math>\mu</math>s (84,719<math>\times</math>)</b>
<b>Precomputation</b>	
PCA-based [2013]	114 s
<b>Ours</b>	<b>79 s</b>

Table 1. **Breakdown of Experiment Runtime.** The table compares the runtime and precomputation costs of the full-space ground truth simulation, the PCA-based approach [2013] and our method on the 2D plume example (Fig. 5). Leveraging the linear Koopman operator, our method achieves significantly faster runtime by requiring only a single matrix multiplication for each reduced-space simulation step.

<sup>†</sup> = with/without external force.

## 5.2 Generalization With Vorticity Confinement

To further evaluate generalization and artist-directability of our method, we revisited the 2D plume example following the experimental setup from the prior work [Kim and Delaney 2013]. Specifically, we incorporated vorticity confinement [Fedkiw et al. 2001] into the original MacCormack solver [Selle et al. 2008] and then applied our method to the modified solver. For training, the vorticity confinement value was set to 1.5, and we tested both our approach and the prior method at vorticity confinement values of 1.51, 1.6, and 2.5. To quantify generalization efficacy, we calculated the relative error of the velocity field with respect to ground truth MacCormack[Selle et al. 2008] simulation results. The results, as shown in Fig. 6, reveal that our method achieves a lower relative error compared to the prior method. Notably, the prior method required 150 basis functions, whereas our method achieved comparable results with only 50 basis functions. These findings highlight the reliability and effectiveness of our Koopman-based approach in adapting to changes in simulation parameters.

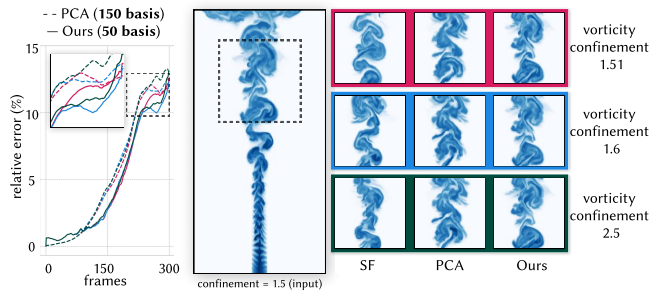


Fig. 6. **Comparison of Relative Error and Vorticity Confinement Between PCA [2013] and DMD (Ours).** Left to right: (1) relative error over time for PCA (150 basis) (dashed) and Ours (50 basis) (solid), showing comparable or lower error with fewer basis functions. (2) reference high-resolution simulation with vorticity confinement 1.5, with a zoomed-in region marked. (3) comparison of the zoomed region in (2) under novel unseen vorticity confinement force 1.51 (magenta), 1.6 (blue), 2.5 (dark green).

## 5.3 3D Plume Baselines

To benchmark our Koopman-based fluid simulation pipeline in 3D scenarios, we present simulation results across three progressively complex scenarios: a standalone plume, a plume interacting with a sphere, and a plume interacting with a bunny. All original simulations were generated using the MacCormack solver [Selle et al. 2008] at a resolution of  $128 \times 128 \times 256$ . We demonstrate the reconstruction results using varying numbers of basis functions. These cases were designed to benchmark the method’s ability to handle increasingly intricate fluid dynamics, from simple turbulence in the standalone plume to complex boundary interactions with the sphere and bunny.

**Plume.** The standalone plume serves as a fundamental benchmark, as tested in prior work [Kim and Delaney 2013]. Fig. 3 highlights the effectiveness of our Koopman-based method in simulating a plume without solid obstacles. While [Kim and Delaney 2013] experienced significant high-frequency dissipation at reduced ranks, our method well captures the swirling and rising behavior, closely matching the original MacCormack simulations even with a small number of basis functions ( $r = 9$ ). Increasing the basis amount to  $r = 28$  or  $r = 61$  further enhances the reconstruction, better preserving turbulent vortices and fine flow structures. These results demonstrate the robustness of our method in capturing the essential dynamics of the fluid simulation, maintaining high fidelity even at low ranks.

**Plume with Sphere and Bunny.** Building on the baseline, we introduce two more scenarios to evaluate the robustness of our method in handling complex boundary conditions. When using a small set of basis ( $r = 9$ ), our Koopman-based method still achieves a reasonable reconstruction of the overall flow dynamics. However, finer details, especially those near boundaries, are less accurately represented. Notably, at  $r = 9$ , there exist noticeable discrepancies in regions around the bunny, which will be resolved as the number of basis functions increases. This enhancement demonstrates how additional basis functions help the Koopman operator to reconstruct higher-frequency components of the flow and better capture complex boundary interactions.

## 5.4 3D Colliding Vortex Rings

The reduced-space Koopman operator is linear and has demonstrated that it can accurately reconstruct scenarios such as Kármán vortex street and plumes. However, these datasets feature velocity fields with relatively smooth variations over time. To test whether our method can adapt to scenarios with abrupt changes in the velocity field, we selected a more challenging scenario: colliding vortex rings. In this experiment, two point vortices are initialized and collide head-on. When the vortices meet, the velocity field undergoes a sudden change, resulting in finer vortex structures. We tested our method on this dataset using 150 basis functions. As shown in Fig. 1, our method can reconstruct the transition from the two point vortices before the collision to the rapid formation of a divergent velocity field during the impact, as well as the subsequent emergence of numerous vortical structures around the periphery. This

experiment shows that our method can effectively handle datasets with significant and sudden velocity variations.

### 5.5 Independence on Simulation Schemes

Our reduced simulation method is inherently *simulator-agnostic*, allowing it to work seamlessly with a variety of fluid solvers. This flexibility arises from the fact that our method models only the transitions between successive fluid states in the reduced space, rather than being tied to the specific equations or numerical schemes of a given solver. This allows us to apply our method to any fluid simulator without the need for additional adjustments. For instance, in our experiments, we used both the MacCormack [Selle et al. 2008] + Reflection [Zehnder et al. 2018] (MC+R) solver and a Lattice Boltzmann Method [Chen and Doolen 1998] (LBM) solver. They are fundamentally different in their discretization and numerical operations. Specifically, the LBM solver does not directly solve the Navier-Stokes equations; instead, it solves the approximated Boltzmann equation, which models fluid dynamics at the mesoscopic scale using particle distribution functions. Our method overcame these challenges by relying solely on the data produced by the solver. In other words, our method is *equation-free*. We detailed the base simulators used for each example in Tab. 2, where our approach successfully reconstructs the velocity fields in all cases.

In contrast, previous data-driven methods such as [Kim and Delaney 2013; Treuille et al. 2006] developed a specific numerical scheme for the subspace constructed from data, essentially binding the formulation to one specific base simulation framework. As a result, when switching to a different solver, their method also requires corresponding adjustments. This will introduce significant limitations for users, as they can only input datasets corresponding to a specific solver. In particular, this limits artists usage to only solutions of such solvers, whereas our method accepts hand-modified, or even real-world measured data.

## 6 APPLICATION: HARNESSING THE LINEARITY

Leveraging the *linearity* of DMD operator, as well as the intuition of bases exposed by the spectral decomposition, we have developed several novel applications that extend the capabilities of our Koopman-based reduced-order simulation pipeline. In this section, we explore these applications, demonstrating that our method’s unique strengths translate into practical tools for graphics and simulation.

### 6.1 Direct Editing Temporal Dynamics

Since our method approximates Eq. (9) with a linear operator in the full space, this allows us to transform the operator acting on the velocity field into the evolution of different modes under a linear operator. Therefore, we can directly edit the temporal dynamics of the fluid system by modifying the modes of the reduced Koopman operator  $\hat{K}$ : we set  $t_0$  to be the initial time,  $\Omega = \log(\Lambda)/\Delta t$ , where  $\Delta t$  is the time step of the dataset. With this, we can rewrite Eq. (7) in

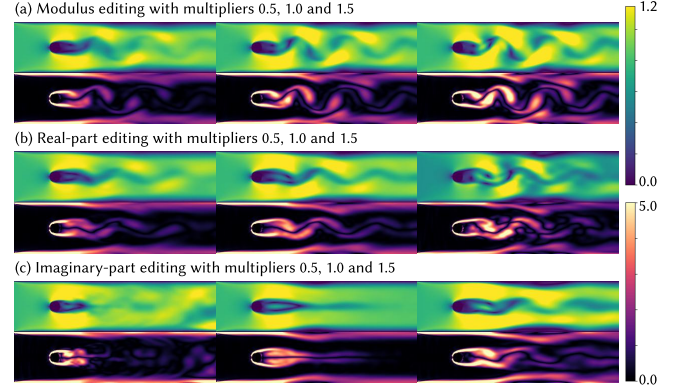


Fig. 7. **Editing temporal dynamics of Kármán Vortex Street with the Koopman Operator Approximation.** The modifications are applied to the DMD basis coefficients: (a) Scaling the modulus of the DMD basis by factors of 0.5, 1.0, and 1.5, affecting overall amplitude; (b) Adjusting the real part of  $\Omega$ , influencing growth and decay rates of modal contributions; (c) Modifying the imaginary part, altering phase dynamics and wave propagation characteristics.

the following form:

$$\begin{aligned} \mathbf{u}(t_0 + k\Delta t) &= \Phi \exp(\Omega t) \mathbf{z}(t_0) \\ &= \Phi \exp(k(\log(r) + i\theta)) \mathbf{z}(t_0) \\ &= \sum_{i=1}^n w_i \Phi_i r_i^k (\cos(k\theta_i) + \sin(k\theta_i)) \mathbf{z}_i(t_0) \end{aligned} \quad (14)$$

where  $w_i$  is a user-defined scalar weight,  $r_i = \sqrt{\Re(\lambda_i)^2 + \Im(\lambda_i)^2}$  is the *modulus* and  $\theta_i = \arctan(\Im(\lambda_i), \Re(\lambda_i))$  is the *phase* of the  $i$ -th eigenvalue  $\lambda_i$  in the diagonal *complex* eigenvalue matrix  $\Lambda$ . Notice that this implies that the modes of the spectral decomposition represent different scales of vorticity, completing the physical intuition of the reduced space modes.

As shown in Eq. (14), our method decomposes a simulation sequence into modes with different growth/decay rates and frequencies. The growth/decay rate of a mode is reflected in  $r_i$ , where a larger  $r_i$  indicates a higher growth rate (or a lower decay rate), and vice versa. The frequency of a mode is represented by the absolute value of  $\theta_i$ , with a larger absolute value corresponding to a higher frequency mode, and vice versa. Furthermore, the different modes are decoupled, allowing for the adjustment of the relative proportions between modes. As a result, these properties provide the artist with powerful tools to edit the simulation playback. The artist can modify the overall velocity field by adjusting the proportion ( $w_i$ ), growth/decay rate ( $r_i$ ), and frequency ( $\theta_i$ ) of specific modes. In the experiments, we directly adjust the real part of  $\Omega_i$  to control  $r_i$ , modify the imaginary part of  $\Omega_i$  to control  $\theta_i$ , and vary the modulus of  $\Phi_i$  to control  $w_i$ .

*Editing the Kármán Vortex Street.* The first example is editing on the classic Kármán vortex street. We filter the imaginary part of  $\Omega$  and cluster modes with an absolute value smaller than 0.01 as *low-frequency cluster*, and the rest as *high-frequency cluster*. The low-frequency mode manifests as a laminar flow, with its phase changing



Examples	Resolution	Dim.	B.C.	Amount of Basis	Base Simulator
Smoke Ring ( $\triangleright$ Fig. 1)	$128 \times 128 \times 256$	3D	Open	150	MC+R
Rayleigh–Taylor Instability ( $\triangleright$ Fig. 2)	$1024 \times 512$	2D	Dirichlet	100	MC
Plume ( $\triangleright$ Fig. 3)	$128 \times 128 \times 256$	3D	Open	2, 9, 28, 61, 105, 130, 150	MC
Plume w/ Sphere ( $\triangleright$ Fig. 3)	$128 \times 128 \times 256$	3D	Dirichlet	2, 9, 28, 61, 105, 130, 150	MC
Plume w/ Bunny ( $\triangleright$ Fig. 3)	$128 \times 128 \times 256$	3D	Dirichlet	2, 9, 28, 61, 105, 130, 150	MC
Reversibility ( $\triangleright$ Fig. 8)	$512 \times 512$	2D	Open	20	MC
Editing (Kármán Vortex Street) ( $\triangleright$ Fig. 7)	$512 \times 512$	2D	Periodic	100	LBM-BGK
Editing (Bunny) ( $\triangleright$ Fig. 4)	$128 \times 128 \times 256$	3D	Dirichlet	50	MC

Table 2. **Breakdown of Experiment Setup.** The result and experiment setup are detailed in this table, including grid resolution, dimensionality, boundary conditions (B.C.), the number of basis functions used and the base simulator for each result. As for the base simulators, we employ the MacCormack [Selle et al. 2008] (MC), the MacCormack + Reflection [Zehnder et al. 2018] (MC+R) and the Lattice Boltzmann Method with the Bhatnagar-Gross-Krook collision model (LBM-BGK)[Chen and Doolen 1998].

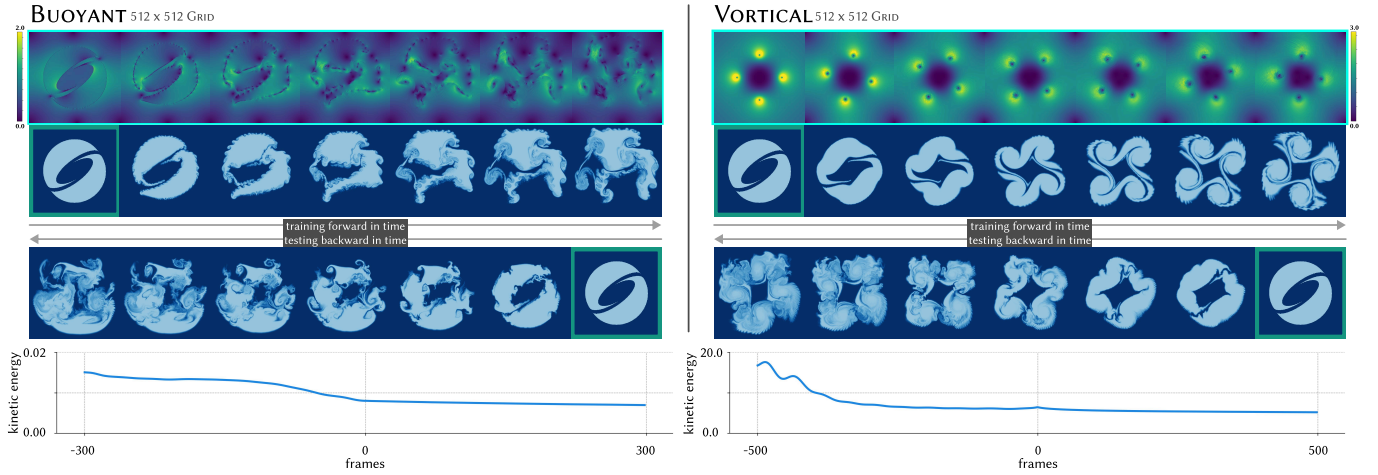


Fig. 8. **Reversibility of Flows with Inversed DMD Operator.** We compare the reconstruction of two distinct fluid flows using Dynamic Mode Decomposition (DMD). The top row in each panel shows the velocity L2-norm of the field used to train the DMD, while the second and third rows depict the temporal evolution of the reconstructed flow fields as applied to an initial density field. The forward-time training phase is followed by a backward-time testing phase to assess predictive accuracy when advecting backward in time. The bottom plots show the evolution of kinetic energy over time. From the buoyant case, we observe the inversed DMD operator  $A^{-1}$  can still reasonably trace backward in time without compromising much visual quality. The vortical case exhibits a more challenging example where the symmetry should be reconstructed backwards in time. We see that the inverse operator indeed recovers this symmetry, with some acceptable levels of incurred noise. Bottom plots show the evolution of the total kinetic energy over time, demonstrating that our inverse operator actually correctly reverses the arrow of time, reversing the dissipation-related entropy increase over time. Decreasing kinetic energy also validates the *physical plausibility* of our result.

very slowly over time. The high-frequency mode is represented by vortical structures distributed on both sides of the cylinder, where the phase of this mode changes relatively quickly over time. As seen in Fig. 7, when we adjust the modulus of the high-frequency cluster from 0.5 to 1.5, the intensity of the vortices increases, which is as we expected. When we set the real part of  $\Omega$  to 0.5, it can be observed that the high-frequency motion decays faster than user input. When we set the real part of  $\Omega$  to 1.5, it can be observed that the high-frequency motion decays slower than user input. Similarly, when we tune the imaginary part of  $\Omega$  from 0.5 to 1.5, we could observe the oscillation frequency of the fluid trail transitions from slow to fast compared to user input.

*Editing the Plume with Bunny.* To evaluate the editing capability of our method, we scale our editing scenario to 3D. With the same filtering procedure as in the Kármán vortex street example, we set the low-frequency cluster to high-frequency cluster ratio to 4 : 1, 2 : 1, 1 : 2, and 1 : 4, and compared the results with the user input. From the results, we observe that when the proportion of low-frequency cluster is increased, with a ratio of 4 : 1, the top of the plume lacks "wrinkles" and appears more "fluffy". This is because the velocity field is dominated by smoother, lower-frequency modes than the original user input. Conversely, when the proportion of high-frequency cluster is increased, with ratios of 1 : 4, the plume develops more detailed plume structure around the top, as the velocity field now emphasizes more high-frequency details compared to the user input.



## 6.2 Reversibility of the Reduced Simulation

Although physically-based fluid simulations have the capability to generate stunning visuals, when artists aim to direct the fluid’s evolution toward a predefined target shape, challenges arise. It is a long standing problem in the community that people aim to enable users with *spatial control*. In this example, we aim to enable users to do *temporal control*, motivated by a prior work [Oborn et al. \[2018\]](#). Compared to previous work [2018] where the authors employ a self-attraction force to replace the arbitrary external forces, providing a stable, physics-motivated, but time-consuming approach, we propose a data-driven, fast, and easy to implement method to address the same problem.

We observe that that given  $\tilde{K} = \Phi \Lambda \Phi^+$ , we could easily compute the *inverse* of the truncated Koopman operator  $\tilde{K}^{-1} = (\Phi \Lambda \Phi^+)^{-1} = \Phi \Lambda^{-1} \Phi^+$ , which is essentially the approximate inverse time evolution  $f^{-1}(\mathbf{u})$  of the fluid system. This allows us to reverse the simulation by applying the inverse truncated Koopman operator to the current state of the fluid system:

$$\begin{aligned} \mathbf{u}(t) &= A^{-1} \mathbf{u}(t + \Delta t), \\ \mathbf{u}(t) &= \Phi \Lambda^{-1} \Phi^+ \mathbf{u}(t + \Delta t), \\ \mathbf{u}(t) &= \Phi \Lambda^{-1} \mathbf{z}(t + \Delta t). \end{aligned} \quad (15)$$

Similar to Eq. (8), we could train the reduced Koopman operator on the forward simulation data and then apply the inverse reduced Koopman operator to reverse the simulation, given a state of the fluid system.

*Reversibility of Buoyant Flow.* We experiment our approach on a simple buoyant flow setup (Fig. 8, left). Our dataset was initialized with a density field shaped like the SIGGRAPH logo, with the density value set to 1. A density value of 1 density field was driven by a velocity field where an upwards velocity of 0.3 is set within the SIGGRAPH logo and downwards elsewhere. We run the simulation for 300 frames to construct the dataset, and trained the DMD operator on this dataset. The inverse operator  $\tilde{K}^{-1}$  was then applied to the initial velocity field of the dataset at  $t = 0$  (frame 0). By iteratively applying the inverse operator, we obtained the velocity fields for the preceding frames, starting from frame  $-1$ , frame  $-2$ , and all the way back to frame  $-300$ . When examining the evolution of the density field from frame  $-300$  to frame 300, it is evident that the velocity field remains consistently upward and smooth, indicating that our method is both reasonable and effective. Further analysis of the energy of the velocity field obtained through the inverse process and the velocity field from the dataset reveals a downward trend in energy, with a smooth and reasonable curve, consistent with fluids with dissipative properties. This demonstrates that our inverse operator has the ability to predict a *physically-plausible* velocity field prior to the dataset.

*Reversibility of Vortical Flow.* To challenge the method with a scene of nontrivial vortical structure, we initialized a vortex sheet by placing four vortices at the corners of the domain (Fig. 8, right). We generated the dataset using the same procedure as in the previous experiment, resulting in a collection of 500 frames. Subsequently, we constructed the inverse operator to recover the velocity fields

preceding the dataset. The results show that the density field (counterclockwise) and the dataset (clockwise) rotate in the opposite direction, which indicates that the velocity field predicted by the inverse operator is correct. This is because the vortex sheet velocity field continuously rotates in a clockwise direction, and by examining the density field from frame  $-500$  to frame 500, we observe that the field indeed undergoes continuous clockwise rotation. From the energy field analysis, the results show that, except for the significant energy fluctuation between frames  $-500$  and  $-450$ , the energy consistently decreases in the remaining frames, with a consistent slope. This further demonstrates the robustness of our method.

## 6.3 Upsampling with Reduced Koopman Operator

The scale of the imaginary part of eigenvalues in  $\Lambda$  encode different scales of turbulent modes, enabling us to use a trained DMD operator to add in secondary motion to an existing fluid simulation. This is particularly useful for *upsampling* a low-resolution fluid, simulated using stable fluid for example, leveraging the DMD basis to add in turbulent modes that were too small for the low-res sim to capture. This upscaling problem has been explored in prior work [Kim et al. 2008; Nielsen et al. 2009], but we show that due to the linearity of the Koopman operator, and the physical intuition on each of its reduced bases, this upscaling is essentially attained for *free*, amounting to nothing more than a linear combination of two matrix multiplications.

**6.3.1 Evolution.** Suppose we have frames of a low-res input velocity field  $\{L_0, L_1, L_2, \dots, L_T\}$ , a high-res initial condition  $H_0$ . Additionally, we have some DMD basis  $\Phi$  trained on some high-res simulation distinct from the low-res simulation, with corresponding eigenvalues  $\Lambda$ , sorted by the length of their imaginary parts in increasing order. At the first frame, we can generate the reduced-space initial condition by simply using our basis mapping  $R_0 = \Phi^T H_0$ .

Now, for every subsequent frame  $t$ , we generate  $R_t$  by first applying the DMD evolution on the previous reduced space frame to produce an intermediate state  $R_t^* = \Lambda R_{t-1}$ . We also produce a representation of the current frame of the low-res input in reduced space  $P_t = \Phi^T L_t$ . Now, we have a representation of the *current* frame of the low-res input, and the DMD *time evolution* of the *previous* reduced space frame. We want to keep the low-order bulk flow of the low-res input, and augment it with the high-order turbulent flow learned by the DMD basis. To that end, we split each reduced-space vector into a low-order and high-order part:  $R_t^* = [R_t^{*L} \ R_t^{*H}]$ ,  $P_t = [P_t^L \ P_t^H]$ . Now, we take only the low-order modes of the input flow, and the high-order modes of the DMD-evolved flow, to produce our new reduced space velocity field  $R_t = [P_t^L \ R_t^{*H}]$ . From here, we can just apply the basis to return to high-resolution full-space  $H_t = \Phi R_t$ .

We note that the composition operators here are linear. We can simply represent them with selection matrices  $S^H, S^L$ , for the high- and low-order bases respectively, such that  $R_t = S^L P_t + S^H R_t^*$ . Since the DMD operator is also linear, we note that this entire upscaling method is linear by construction.

Results are shown on Fig. 9. We see that even if the initial velocity field is significantly different from the input field, the low-order basis is able to capture the bulk flow of the low-resolution input,

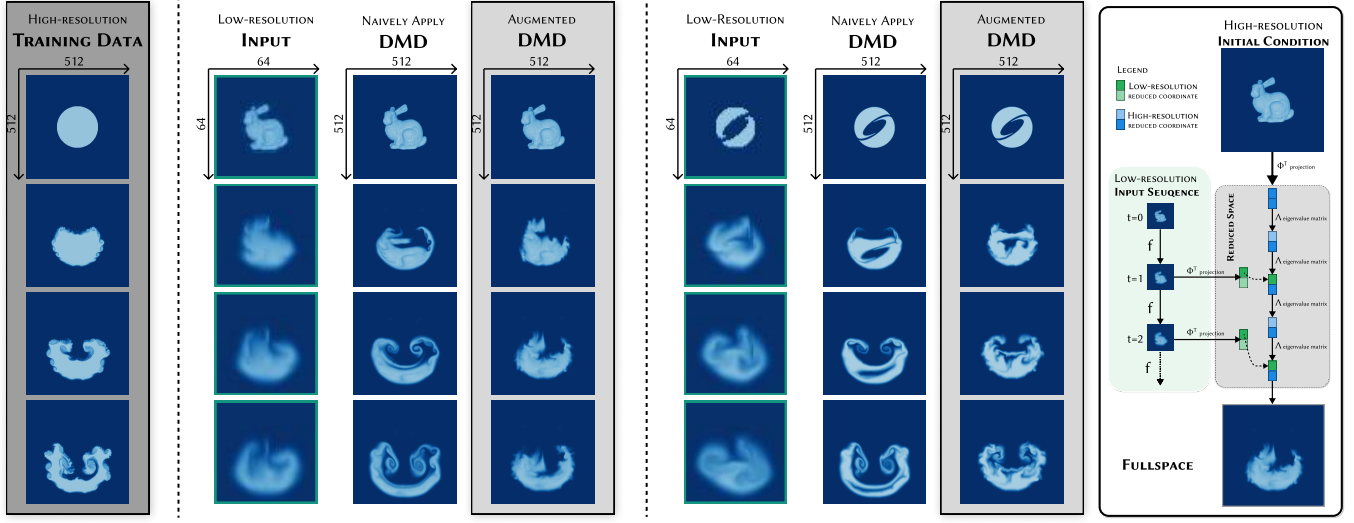


Fig. 9. **Upsampling and Generalization to Unseen Sequences with Trained DMD Operator.** Two different input low-resolution fluid simulations (bunny and SIGGRAPH logo) are upscaled using the same DMD operator trained on a different velocity field. Initial velocity fields are seeded as moving down based on the input density field. Naive application of DMD shown in each middle column, and our *augmented DMD upresolution* method shown on the right columns. Schematic of our method presented on the far right. At each frame, we project the low-resolution artist-directed input into the low-order bases of our reduced representation, using these to replace the low-order terms of the DMD field. Notice that naive application of DMD simply moves towards the known input training data, while our augmented field matches the low-resolution input more closely, with extra high-order detail gained from the DMD operator.

and modify the DMD-produced field accordingly. In particular, we note that naively applying the DMD operator, without passing the low-resolution input field into the low-order bases, ends up reconstructing the original training set, rather than a velocity field directed by our input. This is demonstrated by the results for the two initial conditions being very similar, whereas our augmented field matches the input much closer.

**6.3.2 Projection.** The above governs the time evolution of the velocity field. In some cases, where the input velocity field differs significantly from the training data used for the DMD basis, the above as written will still produce velocity fields that are unacceptably different from the input velocity field. This is largely representation error, fields that are far away from the training data are less representable by the reduced space. In these cases, we can again leverage our input low-res field, this time as a constraint.

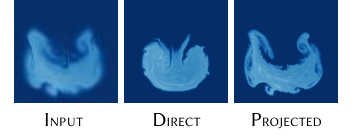
Essentially, we would like to project our velocity field  $H_t$  onto the space of velocity fields that are identical to the input low-res field when downsampled to that resolution. This can be represented as an equality-constrained quadratic problem,

$$\arg \min_x \frac{1}{2} (x - H_t)^T (x - H_t) \quad (16)$$

$$\text{subject to } Ax = L_t, \quad (17)$$

where  $A$  is a downsampling operator that converts from high-res to low-res. Notice that because the downsampling operator does not change for the duration of the simulation. Thus, the KKT (Karush-Kuhn-Tucker) matrix can be precomputed making the projection a single matrix multiply during runtime.

As a sanity check, we show the effect of this projection here: it is apparent with the projection step, we



can recover fields that are much closer to the input, yet retaining extra high-order detail. And, of course, because these are all linear, linear combinations of the direct and projected fields can be taken. In particular, because the basis functions of reduced space are orthogonal, a diagonal matrix of linear weights can be taken, preferring projected for low-order modes and direct for high-order modes, for example.

## 7 DISCUSSION

DMD, similar to PCA, performs dimensionality reduction on some dataset, and provides a reduced basis for representing the fluid flow. The key difference is that in constructing this basis, DMD optimizes for the *action* of some linear operator that maps the data from one time state to the next, while PCA only looks at the state snapshots individually. This temporal information allows DMD to find a reduced space that can represent the evolution of the field as a linear operator within this reduced space, as opposed to PCA that only seeks to find a reduced space that best minimizes reconstruction error.

We also note that as a spectral decomposition of the reduced Koopman operator, much of the *directability* power of DMD comes from the exposure of the complex-valued eigenvalues. Because each eigenvalue represents a different speed of rotation, and their corresponding eigenvectors represent their spatial bases, we gain manipulability over different scales of turbulence for free. This is similar

in concept to other spectral methods [Kim et al. 2008], but with corresponding spatial bases that learned from input data rather than predefined bases. We thus show that DMD straddles the realm between *spatial* order reduction (PCA and co.) and *temporal* order reduction (spectral methods), gaining strengths of both methodologies.

The other interesting observation is that *dynamic mode decomposition* (DMD) shares the same algebraic backbone as a first-order *vector autoregression*, VAR(1) [Stock and Watson 2001]: both estimate a linear map that advances a state from  $t$  to  $t+1$ . Under a *noise-free* assumption, the two time-shifted snapshot matrices  $X'$  and  $X$  introduced in Eq. (4) satisfy the VAR(1) relation

$$X' = AX, \quad A \in \mathbb{R}^{(Nn) \times (Nn)}, \quad (18)$$

with  $A$  interpreted as the *state-transition matrix*. DMD solves the identical least-squares problem but replaces the full  $A$ —the restricted Koopman operator  $K$  in our notation—with its truncated SVD, yielding a compact low-rank approximation that captures the dominant dynamical modes.

## 7.1 Limitations

*SVD for Dimensionality Reduction.* Like other low-rank approximation approaches, DMD requires performing SVD for dimensionality reduction. This demands high memory consumption for large-dimensional data, such as 3D fluid simulation, on the order of  $O(n^2)$ , where  $n$  is the number of state variables. Despite applying randomized SVD to reduce this memory footprint, larger simulations are still costly to train. In the future, we will explore the use of other dimensionality reduction methods, such as autoencoders, to reduce the dimensionality of the dataset.

*Unseen External Force Generalization.* It is well-known that reduced-order model trades *generalization for performance*; our method is no exception. We have demonstrated that we can represent velocity fields that were unseen by DMD, as well as interactively push current states towards modified states. Despite this, representability is still a key limitation—there just may not be the spatial basis and the reduced Koopman operator to support certain velocity fields and their nonlinear evolution. Thus, users may find that interacting with the fields, especially in states far away from observed data, behave strangely. This warrants further work on improving spatial generalization of these methods.

## 7.2 Future Work

We have identified various potential avenues for extension, particularly ones that exploit the linearity of the DMD operator.

*Augmenting Fluid State with Density Field.* The DMD framework allows us to trivially evaluate the velocity field at any point in time via a low-dimensional matrix exponentiation (as shown in Fig. 2), unfortunately evaluating an immersed *density field* at any point in time is not so trivial. To achieve the time-evolved density field, we’ve first had to resort to classical numerical integration, first evaluating the velocity field with our fast DMD model, and then using that velocity field to advect the density field forward. As an exciting avenue of future work, this could potentially be sidestepped by defining the fluid state as including the density field (as opposed

to just the velocity field), and forming a Koopman operator on this augmented system. This would allow us to use the exact same DMD framework to quickly query the density of the flow at any point in time.

*Adaptive Blending between DMD Operators.* As a linear operator, linear combinations of DMD operators construct a valid linear operator. Therefore, we can train multiple DMD operators and flexibly combine them to obtain a richer set of new motion patterns. Similar to building an ensemble, once we have multiple pre-trained DMD operators, we solve a minimal optimization problem to adaptively select the optimal linear combination of DMD operators based on user input, achieving the desired motion effects in the simulation.

*Inverse Design.* Since the DMD operator decomposes the energy information ( $\Phi$ ) and frequency information ( $\Lambda$ ) of a simulation sequence, we can adjust the eigenbasis of the DMD operator based on the user-defined objectives, thereby modifying the energy and frequency information represented by the DMD operator. We take gradients of the DMD operator with respect to some target images at prescribed points in time, and use this as an optimizer to construct a new DMD operator that represents flow that evolving through the specified snapshots.

*Learning Control Space Enrichment.* Inspired by DMD with control [Proctor et al. 2016], we could train a neural network to enrich the linear control space of the DMD operator with nonlinear responses. This network could be a direct way to expand the expressiveness of the DMD operator, allowing us to learn more diverse motion patterns and handle more complex user inputs.

## 8 CONCLUSION

We introduce *Dynamic Mode Decomposition* (DMD) to the field of graphics. Unlike previous methods that approximate the solution space, our approach is an *equation-free, data-driven* method that learns a low-rank approximation of the state transition matrix directly from the dataset. Its linear nature enables rapid reconstruction of the dataset and shows superior reconstruction quality compared to previous data-driven methods. The operator linearity also unlocks a collection of artist-centric graphics applications: (1) by decomposing the dataset into a linear combination of temporally oscillating modes of different frequencies, our method enables artists to edit the dataset by adjusting the mode’s modulus, growth/decay rate, and frequency, all without requiring knowledge of the solver’s details; (2) with the inverse of the Koopman operator, we can go back in time and rollout animations at negative frame ranges; (3) users can supplement low-resolution flow sequences and generate new enriched flow details with the trained DMD operator, taking advantage of the high-frequency information stored within the operator.

## ACKNOWLEDGMENTS

We are grateful to Jonathan Chalaturnyk for illuminating discussions that shaped the early stages of this research. We thank Xinwen Ding for helpful proofreading feedback. This research was made possible with the administrative support of our lab’s system administrator, John Hancock, and financial officer, Xuan Dam. This research was



supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) through grant RGPIN-2021-03733, whose funding made this work possible.

## REFERENCES

- Travis Askham and J Nathan Kutz. 2018. Variable projection methods for an optimized dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems* 17, 1 (2018), 380–416.
- Christopher Brandt and Klaus Hildebrandt. 2017. Compressed vibration modes of elastic bodies. *Computer Aided Geometric Design* 52 (2017), 297–312.
- William L Briggs, Van Emden Henson, and Stephen F McCormick. 2000. A Multigrid Tutorial. *SIAM* 3 (2000).
- Steven L. Brunton, Marko Budišić, Eureka Kaiser, and J. Nathan Kutz. 2022. Modern Koopman Theory for Dynamical Systems. *SIAM Rev.* 64, 2 (2022), 229–340. <https://doi.org/10.1137/21M1401243> arXiv:[https://doi.org/10.1137/21M1401243](https://arxiv.org/abs/https://doi.org/10.1137/21M1401243)
- Shiyi Chen and Gary D Doolen. 1998. Lattice Boltzmann method for fluid flows. *Annual review of fluid mechanics* 30, 1 (1998), 329–364.
- Yixin Chen, David Levin, and Timothy Langlois. 2024. Fluid Control with Laplacian Eigenfunctions. In *ACM SIGGRAPH 2024 Conference Papers*. 1–11.
- Yixin Chen, Wei Li, Rui Fan, and Xiaopei Liu. 2021. GPU optimization for high-quality kinetic fluid simulation. *IEEE Transactions on Visualization and Computer Graphics* 28, 9 (2021), 3235–3251.
- Albert Chern, Felix Knöppel, Ulrich Pinkall, and Peter Schröder. 2017. Inside fluids: Clebsch maps for visualization and processing. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–11.
- Albert Chern, Felix Knöppel, Ulrich Pinkall, Peter Schröder, and Steffen Weißmann. 2016. Schrödinger’s smoke. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–13.
- Qiaodong Cui, Pradeep Sen, and Theodore Kim. 2018. Scalable laplacian Eigenfluids. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.
- Tyler De Witt, Christian Lessig, and Eugene Fiume. 2012. Fluid simulation using laplacian eigenfunctions. *ACM Transactions on Graphics (TOG)* 31, 1 (2012), 1–11.
- Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. 2001. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 15–22.
- Nick Foster and Dimitri Metaxas. 1996. Realistic animation of liquids. *Graph. Models Image Process.* 58, 5 (Sept. 1996), 471–483. <https://doi.org/10.1006/gmip.1996.0039>
- Francis H Harlow, J Eddie Welch, et al. 1965. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of fluids* 8, 12 (1965), 2182.
- Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. 2020. Array programming with NumPy. *Nature* 585, 7825 (2020), 357–362.
- Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. 2019. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–16.
- Sara M Ichinaga, Francesco Andreuzzi, Nicola Demo, Marco Tezzele, Karl Lapo, Gianluigi Rozza, Steven L Brunton, and J Nathan Kutz. 2024. PyDMD: A Python package for robust dynamic mode decomposition. *arXiv preprint arXiv:2402.07463* (2024).
- Tiffany Inglis, M-L Eckert, James Gregson, and Nils Thuerey. 2017. Primal-dual optimization for fluids. In *Computer Graphics Forum*, Vol. 36. Wiley Online Library, 354–368.
- Stefan Jeschke, Tomáš Skřivan, Matthias Müller-Fischer, Nattapong Chentanez, Miles Macklin, and Chris Wojtan. 2018. Water surface wavelets. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–13.
- Theodore Kim and John Delaney. 2013. Subspace fluid re-simulation. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 1–9.
- Theodore Kim, Nils Thürey, Doug James, and Markus Gross. 2008. Wavelet turbulence for fluid simulation. *ACM Transactions on Graphics (TOG)* 27, 3 (2008), 1–6.
- Bernard O Koopman. 1931. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences* 17, 5 (1931), 315–318.
- Bernard O Koopman and J v Neumann. 1932. Dynamical systems of continuous spectra. *Proceedings of the National Academy of Sciences* 18, 3 (1932), 255–263.
- Wei Li, Yixin Chen, Mathieu Desbrun, Changxi Zheng, and Xiaopei Liu. 2020. Fast and scalable turbulent flow simulation with two-way coupling. *ACM Trans. Graph.* 39, 4 (2020), 47.
- Beibei Liu, Gemma Mason, Julian Hodgson, Yiyi Tong, and Mathieu Desbrun. 2015. Model-reduced variational fluid simulation. *ACM Transactions on Graphics (TOG)* 34, 6 (2015), 1–12.
- Miles Macklin and Matthias Müller. 2013. Position based fluids. *ACM Trans. Graph.* 32, 4, Article 104 (July 2013), 12 pages.
- Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. 2004. Fluid control using the adjoint method. *ACM Transactions On Graphics (TOG)* 23, 3 (2004), 449–456.
- Matthias Müller, David Charypar, and Markus Gross. 2003. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Citeseer, 154–159.
- Michael B Nielsen, Brian B Christensen, Nafees Bin Zafar, Doug Roble, and Ken Museth. 2009. Guiding of smoke animations through variational coupling of simulations at different resolutions. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 217–226.
- Jeremy Oborn, Sean Flynn, Parris K Egbert, and Seth Holladay. 2018. Time-Reversed Art Directable Smoke Simulation.. In *Eurographics (Short Papers)*. 1–4.
- Zherong Pan and Dinesh Manocha. 2017. Efficient solver for spacetime control of smoke. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1.
- Joshua L Proctor, Steven L Brunton, and J Nathan Kutz. 2016. Dynamic mode decomposition with control. *SIAM Journal on Applied Dynamical Systems* 15, 1 (2016), 142–161.
- Yousef Saad. 2003. *Iterative methods for sparse linear systems*. SIAM.
- Diya Sashidhar and J Nathan Kutz. 2022. Bagging, optimized dynamic mode decomposition for robust, stable forecasting with spatial and temporal uncertainty quantification. *Philosophical Transactions of the Royal Society A* 380, 2229 (2022), 20210199.
- Syuhui Sato, Yoshinori Dobashi, and Theodore Kim. 2021. Stream-guided smoke simulations. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–7.
- Peter J Schmid. 2010. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics* 656 (2010), 5–28.
- Silvia Sellán, Jack Luong, Leticia Mattos Da Silva, Aravind Ramakrishnan, Yuchuan Yang, and Alec Jacobson. 2023. Breaking good: Fracture modes for realtime destruction. *ACM Transactions on Graphics* 42, 1 (2023), 1–12.
- Andrew Selle, Ronald Fedkiw, Byungmoon Kim, Yingjie Liu, and Jarek Rossignac. 2008. An unconditionally stable MacCormack method. *Journal of Scientific Computing* 35 (2008), 350–371.
- Jos Stam. 1999. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. 121–128.
- Jos Stam. 2001. A simple fluid solver based on the FFT. *Journal of graphics tools* 6, 2 (2001), 43–52.
- James H Stock and Mark W Watson. 2001. Vector autoregressions. *Journal of Economic perspectives* 15, 4 (2001), 101–115.
- Jingwei Tang, Vinicius C. Azevedo, Guillaume Cordonnier, and Barbara Solenthaler. 2021. Honey, I Shrunk the Domain: Frequency-aware Force Field Reduction for Efficient Fluids Optimization. In *Computer Graphics Forum*, Vol. 40. Wiley Online Library, 339–353.
- Nils Thürey, Richard Keiser, Mark Pauly, and Ulrich Rüde. 2009. Detail-preserving fluid control. *Graphical Models* 71, 6 (2009), 221–228.
- Adrien Treuille, Andrew Lewis, and Zoran Popović. 2006. Model reduction for real-time fluids. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 826–834.
- Adrien Treuille, Antoine McNamara, Zoran Popović, and Jos Stam. 2003. Keyframe control of smoke simulations. In *ACM SIGGRAPH 2003 Papers*. 716–723.
- Martin Wicke, Matt Stanton, and Adrien Treuille. 2009. Modular bases for fluid dynamics. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 1–8.
- Matthew O Williams, Ioannis G Kevrekidis, and Clarence W Rowley. 2015. A data-driven approximation of the koopman operator: Extending dynamic mode decomposition. *Journal of Nonlinear Science* 25 (2015), 1307–1346.
- Jonas Zehnder, Rahul Narain, and Bernhard Thomaszewski. 2018. An advection-reflection solver for detail-preserving fluid simulation. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–8.

## A KOOPMAN THEORY

We briefly summarize Koopman theory, following the presentation in Brunton et al. [2022], to motivate the the linearization introduced in Eq. (3). Once again, consider the autonomous system:

$$\frac{du}{dt} = f(u). \quad (19)$$

The time-dependent state  $u(t)$  subject to this dynamical system and some initial condition  $u(0) = u_0$  may be represented with a time-dependent family of flow maps  $\{F_t\}_{t>0} : \mathbb{R}^{Nn} \rightarrow \mathbb{R}^{Nn}$  such that:

$$u(t) = F_t(u_0). \quad (20)$$

That is, it maps some initial state  $u_0$  from time 0 to its state at time  $t$ . In general,  $F(u)$  is not a linear map, but as per Koopman and Neumann [1932] may be lifted in a higher space that admits a linear mapping in that space. In particular, take a Hilbert space



$\mathcal{H} \subseteq \{g : \mathbb{R}^N \rightarrow \mathbb{C}\}$  of scalar observation functions  $g \in \mathcal{H}$ . With these observables, Koopman [1931] defines a family of *Koopman operators*  $\mathcal{K}_t : \mathcal{H} \rightarrow \mathcal{H}$  such that

$$\mathcal{K}_t g = g \circ F_t, \forall g \in \mathcal{H}, \quad (21)$$

forming a trajectory in  $\mathcal{H}$ . That is to say,  $\mathcal{K}_t$  maps the measurement operator  $g$  to its state at time  $t$ . Consequently, rather than following the nonlinear flow according to  $F_t$  of some initial state  $\mathbf{u}_0$ , we can instead take an observation of  $\mathbf{u}_0$ , and linearly evolve it according to  $\mathcal{K}_t$ . Notice that because  $\mathcal{H}$  is a linear space, it follows from Eq. (21) that:

$$\mathcal{K}_t(\alpha g_\alpha + \beta g_\beta) = \alpha \mathcal{K}_t g_\alpha + \beta \mathcal{K}_t g_\beta, \quad (22)$$

irrespective of whether  $F_t$  is linear in state space  $\mathbb{R}^N$ .

An infinitesimal generator  $\mathcal{L}$  of the family  $\{\mathcal{K}_t\}$  exists given  $f$  is an autonomous system (Eq. (19)), and can be found via the limit,

$$\mathcal{L}(g(\mathbf{u}(t))) = \lim_{\tau \rightarrow 0} \frac{\mathcal{K}_\tau g(\mathbf{u}(t)) - g(\mathbf{u}(t))}{\tau}. \quad (23)$$

Notice that the limit is exactly the definition of a derivative, so that:

$$\frac{d}{dt} g(\mathbf{u}(t)) = \mathcal{L}g(\mathbf{u}(t)). \quad (24)$$

This is analogous to Eq. (19), but with the finite nonlinear operator  $f$  replaced by an infinite-dimensional linear operator  $\mathcal{L}$ . This has the matrix exponential solution  $g(\mathbf{u}(t+\tau)) = e^{\mathcal{L}(t+\tau)} g(\mathbf{u}(t))$ . Thus, for discrete time  $t \in \mathbb{N}$ , we can simply take some constant timestep  $\tau$  to find a single Koopman operator  $\mathcal{K} = e^{\mathcal{L}\tau}$ , notably acting linearly in  $\mathcal{H}$ , whose  $m$ -times repeated application  $\{\mathcal{K}^m = \mathcal{K} \circ \mathcal{K} \circ \dots \circ \mathcal{K}\}$  constructs the family of operators at discrete multiples of  $\tau$ . This discrete-time Koopman operator, often called *the* Koopman operator, thus giving the timestepping scheme:

$$g(\mathbf{u}((k+1)\tau)) = \mathcal{K}g(\mathbf{u}(k\tau)). \quad (25)$$

Now, suppose we have some input data consisting of a (necessarily) finite set of measurements. We can consider each one of these measurements to be exactly one evaluation of the observation function. We can thus restrict our set of observation functions only to those required for the given measurements. That is, we only need to consider observation functions spanned by the set  $\{g_i(\mathbf{u}) = \mathbf{u}(\mathbf{x}_i)\}$ , where  $\mathbf{x}_i$ 's are all the spatial degrees of freedom of our measurements. In other words, we limit  $\mathcal{H}$  only to the subspace of observables spanned by input data. As a convenient representation of the entire system observation function, we define the stacked vector,

$$\mathbf{g}(\mathbf{u}) = \begin{bmatrix} g_1(\mathbf{u}) \\ g_2(\mathbf{u}) \\ \vdots \\ g_n(\mathbf{u}) \end{bmatrix} = \begin{bmatrix} \mathbf{u}(\mathbf{x}_1) \\ \mathbf{u}(\mathbf{x}_2) \\ \vdots \\ \mathbf{u}(\mathbf{x}_n) \end{bmatrix} \quad (26)$$

where  $n$  is the total spatial degrees of freedom of our measurements. We then define  $\mathbf{K}$  to be the Koopman operator restricted to this subspace, providing a *linearized* timestepping operator,

$$\mathbf{g}_{k+1} = \mathbf{K} \mathbf{g}_k, \quad (27)$$

analogous to Eq. (25).

We have thus shown that the finite-dimensional *nonlinear* system provided in Eq. (19) may be lifted into a *linear* system in an *infinite-dimensional* Hilbert space  $\mathcal{H}$ , then restricted to a finite linear system respecting a finite set of measurements. We would like to emphasize here, as a key limitation of the theory, the requirement of the nonlinear system to be *autonomous*. That is, the dynamics is necessarily only state-dependent and time-independent. This means that we cannot hope to train an operator that includes some time-dependent source term. Consequently, our moduli-editing schemes cannot be seen as solving the original system given by Eq. (19), but rather provides a perturbed version of the original solution.

## B ADDITIONAL RESULTS

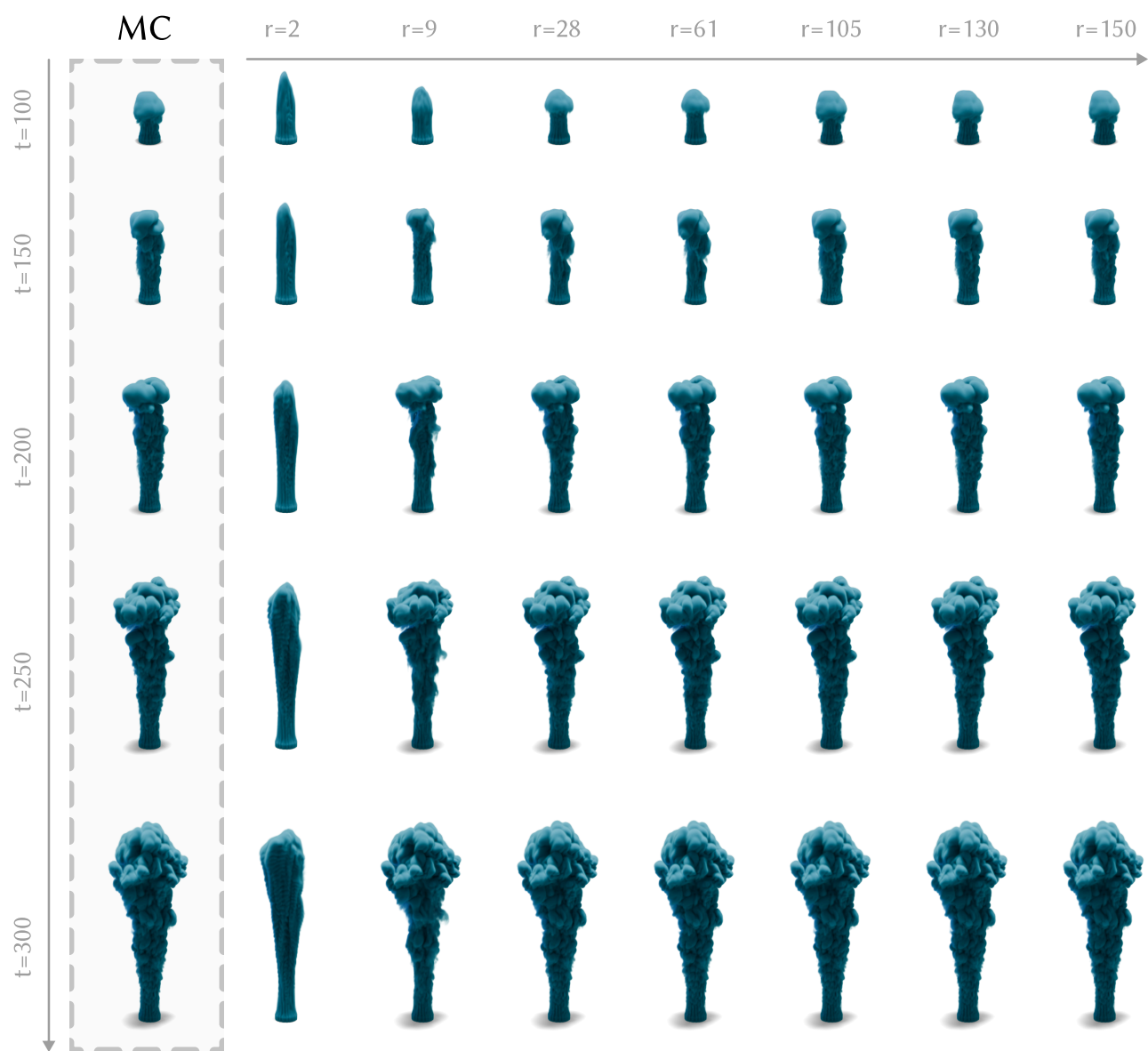


Fig. 10. Additional plume simulation results in Fig. 3.

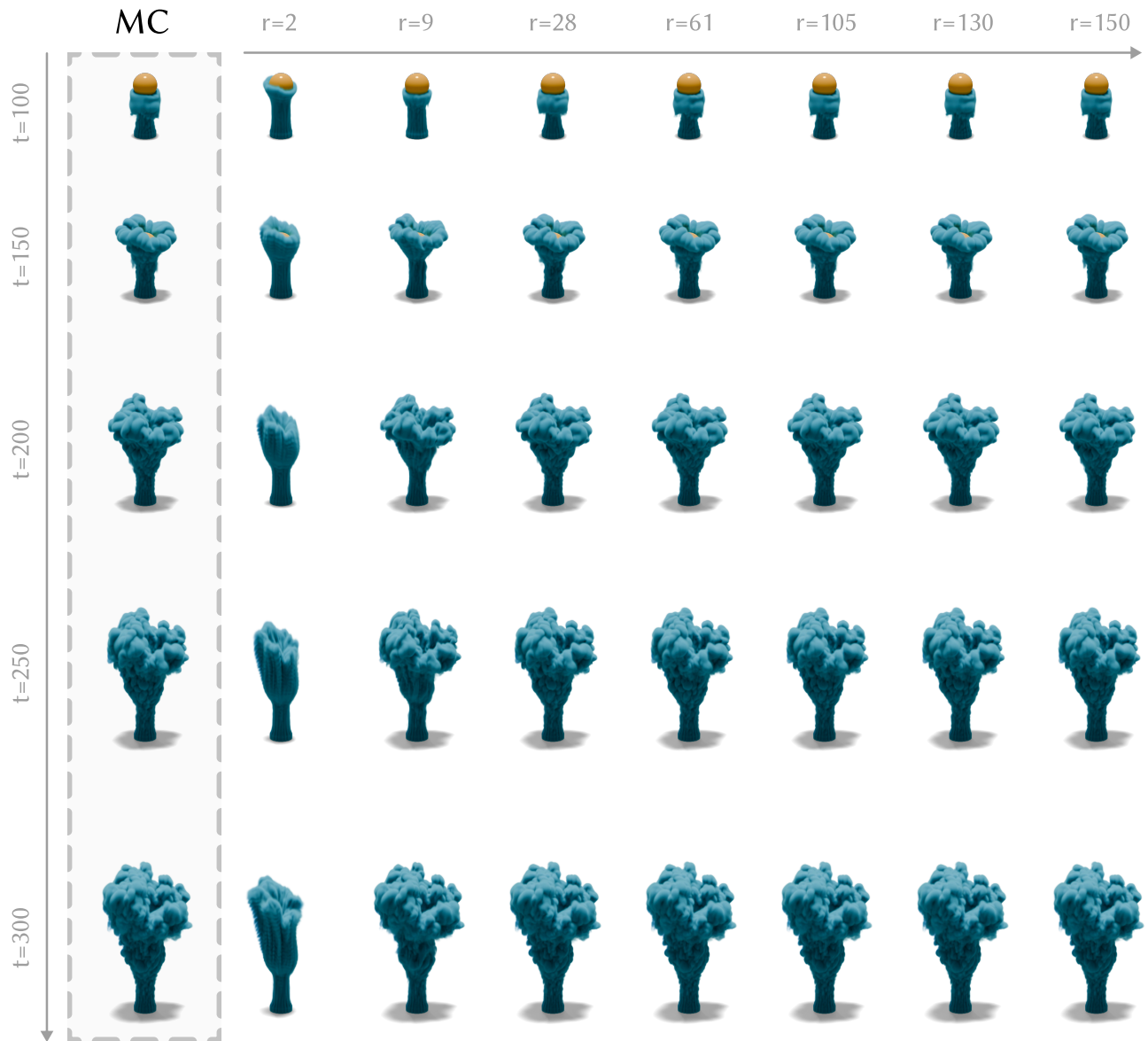


Fig. 11. Additional plume with sphere simulation results in Fig. 3.

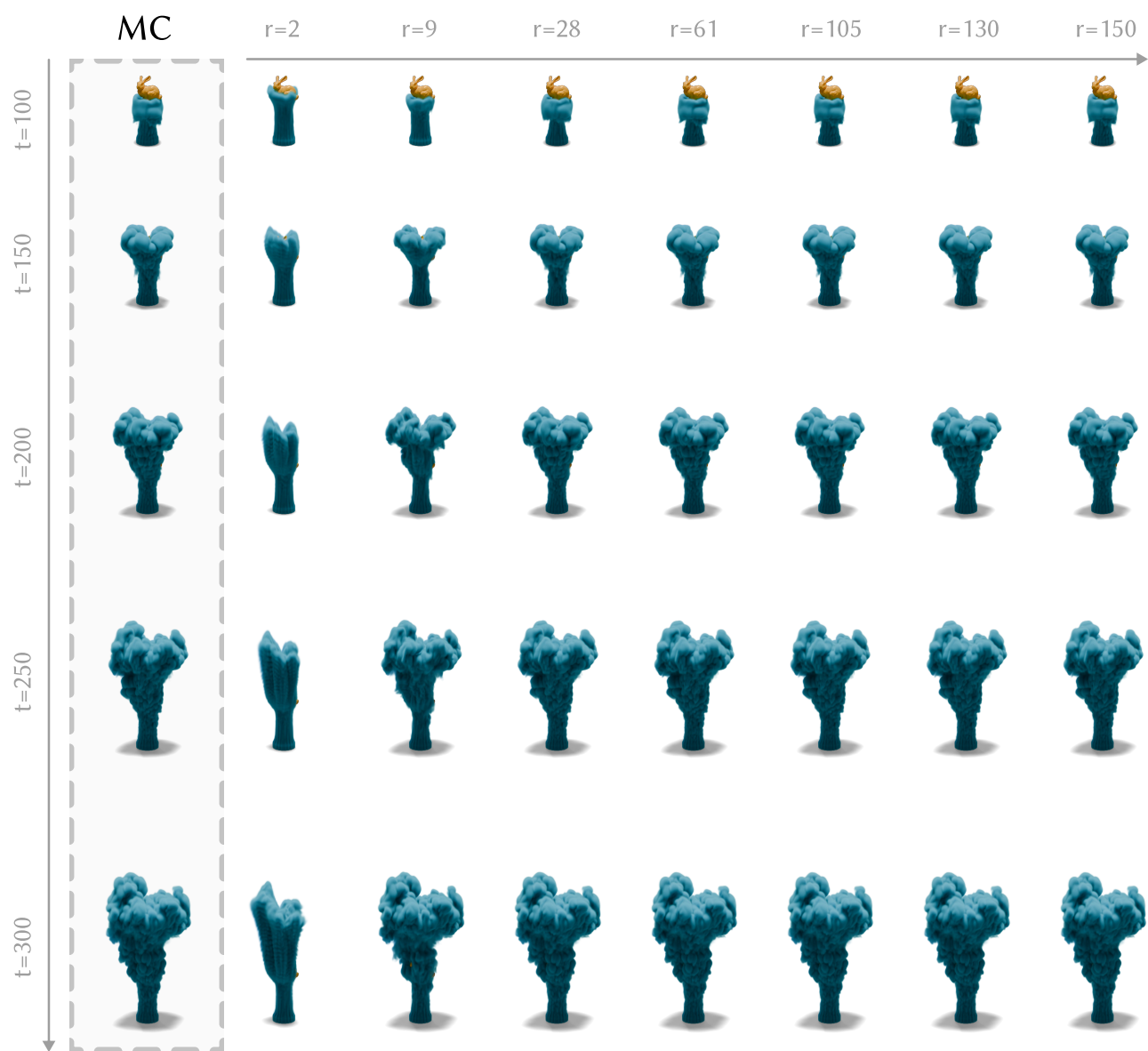


Fig. 12. Additional plume with bunny simulation results in Fig. 3.