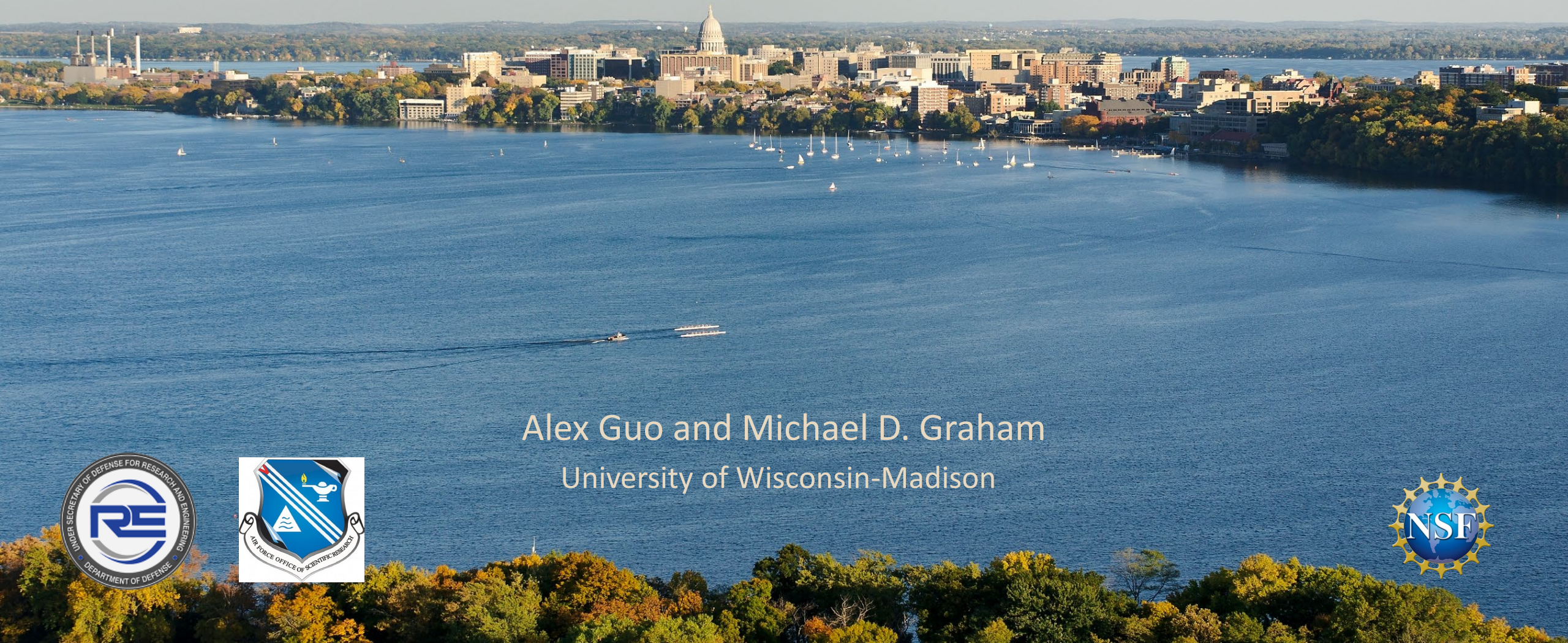
Blending data and physics for data-driven reduced-order modeling of systems with complex dynamics

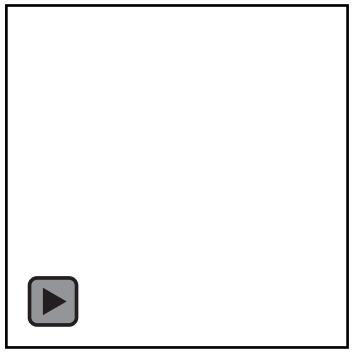


Alex Guo and Michael D. Graham
University of Wisconsin-Madison

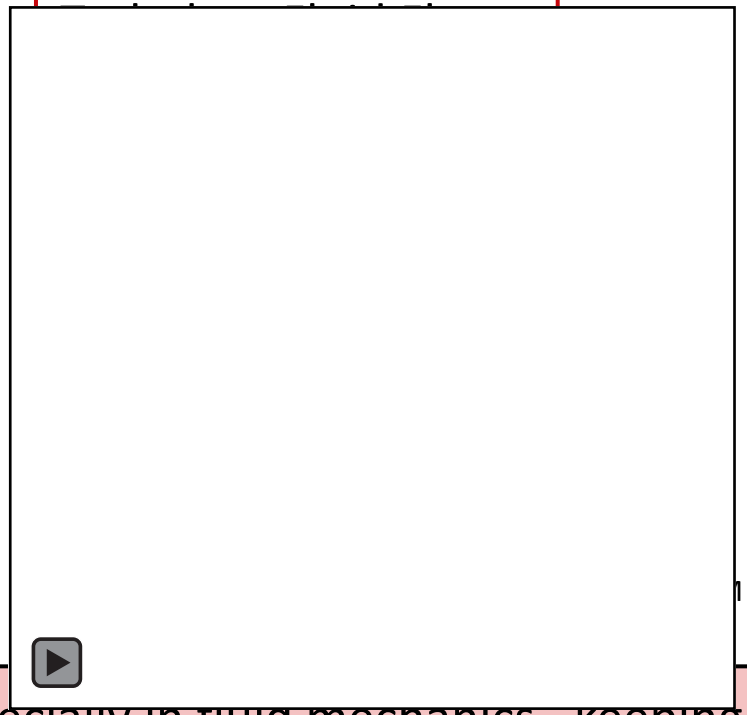
Complex dynamics with many degrees of freedom

- In many applications we have large amounts of high-dimensional data from experiments or high-resolution simulations
- Dynamics may be very complex in space and time
- Modeling is computationally expensive and limits simulation-based design and control applications

Microstructure in a complex fluid



Young et al. Rheol. Acta 2023



1 (2023)

Aim: Data-based modeling of complex systems, especially in fluid mechanics –keeping only *essential* degrees of freedom ➔ efficient models for design, control, optimization

Data, dynamics, and manifolds

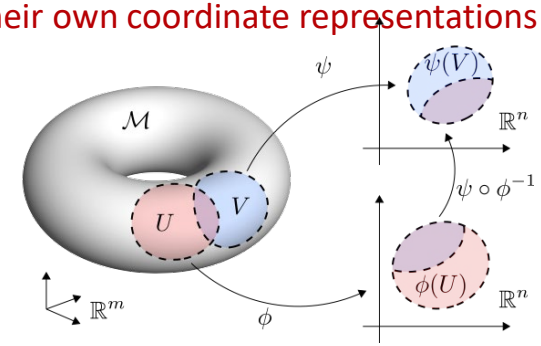
Manifold: generalization of curve or surface – can *locally* represent with Cartesian coords.

- **Dynamics:** invariant manifolds are surfaces in state space that trajectories stay on forever, and a stable manifold is approached by nearby trajectories
  - Invariant manifolds organize dynamics and contain long-time behavior
  - Diffusion/viscosity → strong damping of short wavelengths → long-time dynamics of dissipative PDEs lie on a finite-dimensional invariant manifold

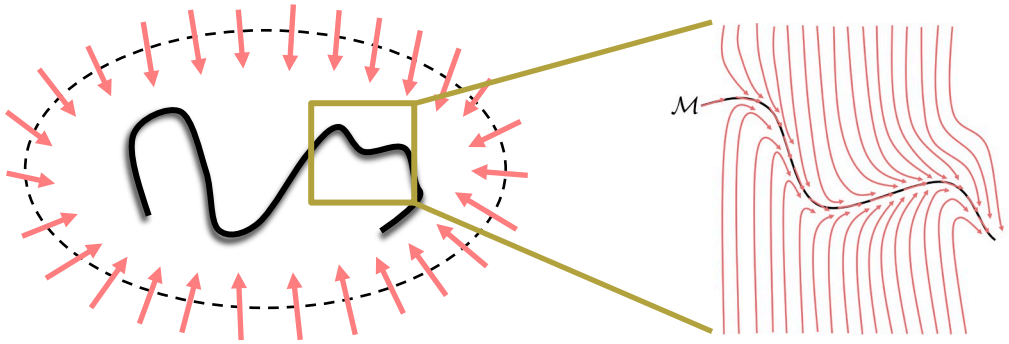
- **Machine learning:** "Manifold hypothesis": idea that real-world data in *d* dimensions lies on a manifold with far fewer dimensions – partial explanation for the success of neural networks to represent and classify data.

- Unless the data is very simple, we don't know in advance the dimension on the manifold it lies on.

Manifold and local regions ("charts") with their own coordinate representations



State space of a dynamical system with a stable manifold – arrows are trajectories



Initial conditions approaching a stable oscillatory solution (1D manifold)



Overview: toward minimal data-driven models of complex dynamics

"Data-driven manifold dynamics (DManD)"

Overall Aim: **From data**, efficiently/*minimally* approximate and the dynamics on it for complex/chaotic processes $\mathcal{M}$

- Short-time predictions (always limited quantitatively for chaotic systems)
- Long-time statistics – overall "shape" of the dynamics in state space

➢ **Coordinate transformations** between full state space *H* and lower-dimensional manifold $\mathcal{M}$

  ➢ Reduced description contains the essential structures of the flow -- autoencoders

  ➢ Combine clustering with local representations to obtain global minimal-dimensional representations

➢ **ODE models** for the dynamics on the manifold (again in neural network form: "Neural ODEs" (NODEs))

  ○ Respect Markovian nature of the original dynamical system

  ○ Allow effective use of widely spaced data – no time derivatives should need to be estimated from data

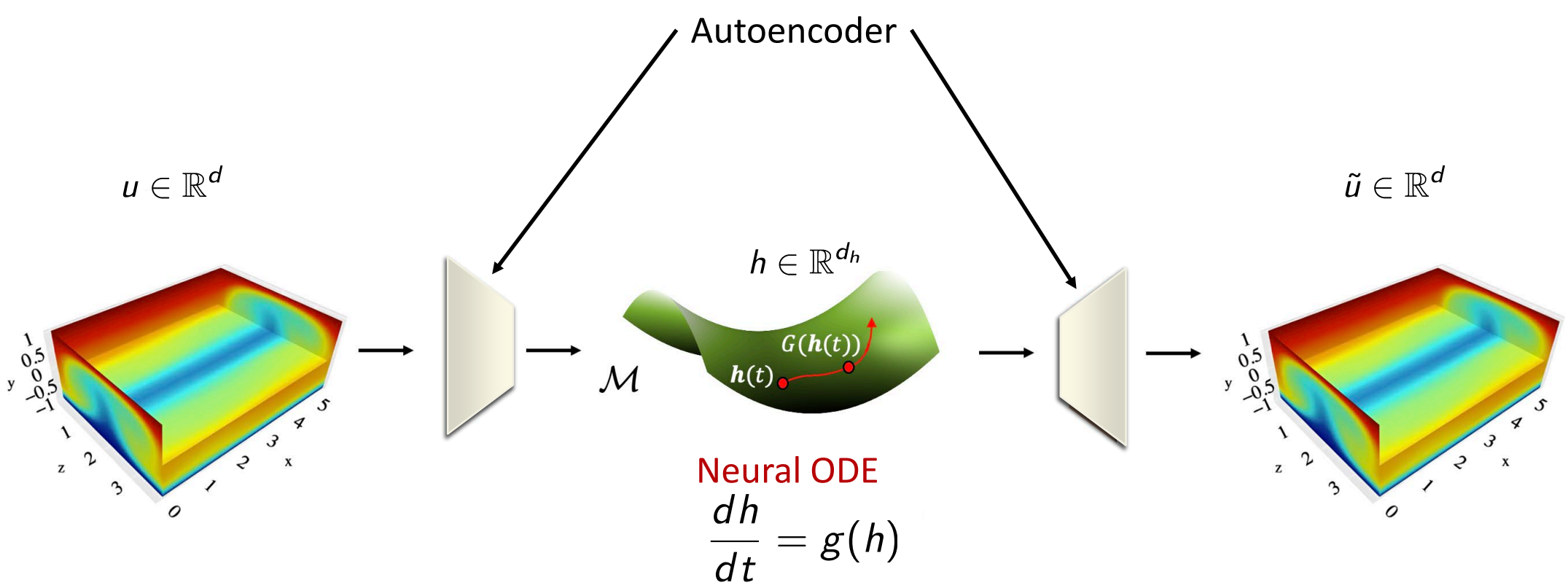
➢ Exploit continuous and discrete symmetries (Linot et al, PRE 2020, Perez De Jesus & G. 2025)

➢ Allow for actuation and exploit low-dimensional representation for control

➢ Apply to drag reduction (Linot et al IJHFF (2023)), evolution equations for complex fluids (Young et al. Rheol. Acta 2023), …

Other manifold-based or "latent dynamics" approaches: Kevrekidis, Koumoutsakos, Haller,…

DManD: Basic Framework



Notes:

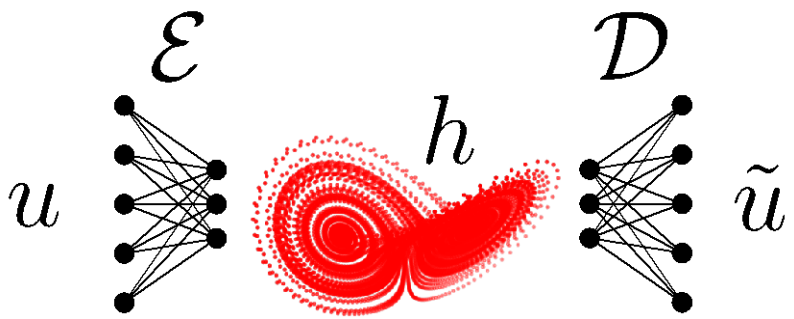
1. How do we estimate d_h? See e.g. Zeng et al. MLST 5, 025053 (2024)
2. If there's a one-to-one mapping between *M* and a region of a linear subspace *S*, then we can use a linear encoder → better interpretability (always true for $d_h > 2d_M$ where d_M is intrinsic dimension of *M*)

Data-driven manifold dynamics (DManD)

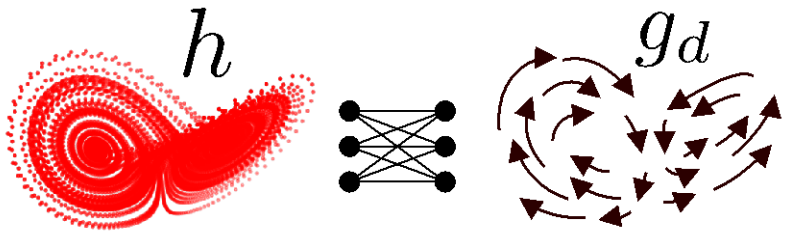
DManD is a **solely data-driven ROM**

1. Uses autoencoder to perform dimension reduction from state space vector u to latent space vector h

2. Trains neural ODE (NODE) to learn vector field in the latent space



Autoencoder learns manifold coordinates h



Neural ODE learns $dh/dt = g_d(h)$

How do we incorporate exact or approximate knowledge of the governing equations?

Physics-based model: Manifold Galerkin (MG)

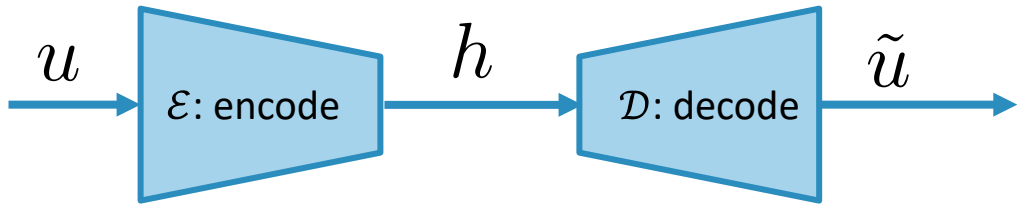
- Assume we have a full-order model

- Manifold Galerkin: express RHS of full-order model (FOM) in manifold coordinates *h*
- While POD-Galerkin projects the FOM onto a linear subspace, MG does so onto a nonlinear manifold:

Define as $J_{\mathcal{E}}(u)$ and calculate with automatic differentiation

➔ g_m is the approximate vector field on the manifold

Autoencoder



- Encoder:
- Decoder:

Notes:
- Retains the expense of solving the FOM
- Lacks robustness to errors in FOM or autoencoder
- Does not exploit the time-series nature of data

Lee & Carlberg, JCP 2020

Approximating the MG vector field: the MG-N method

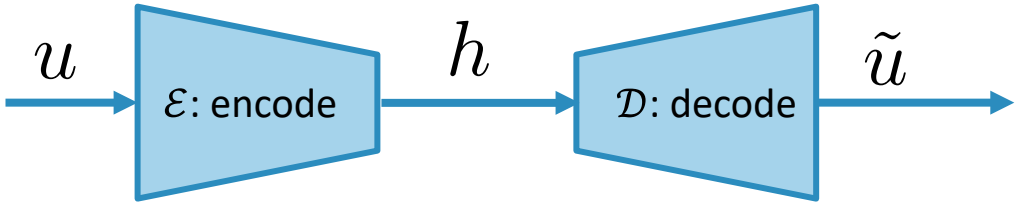
- Assume we have a full-order model

- Manifold Galerkin: express RHS of full-order model (FOM) in manifold coordinates *h*
- While POD-Galerkin projects the FOM onto a linear subspace, MG does so onto a nonlinear manifold:

Define as $J_{\mathcal{E}}(u)$ and calculate with automatic differentiation

➔ g_m is the approximate vector field on the manifold

Autoencoder



- Encoder:
- Decoder:

MG – Neural network (MG-N)

- Use deep neural network to approximate $g_m(h)$:

- MG-N approximates the physics-based vector field given by MG
- Avoids computation of f_{FOM} at each time step

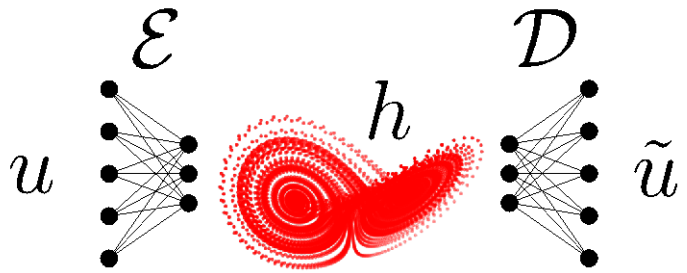
Still lacks robustness and does not fully exploit dataset

Physics-informed, data-driven manifold dynamics (PI-DManD)

- MG/MG-N do not exploit the availability of data as *time series*
- We combine DManD with MG-N in two ways:

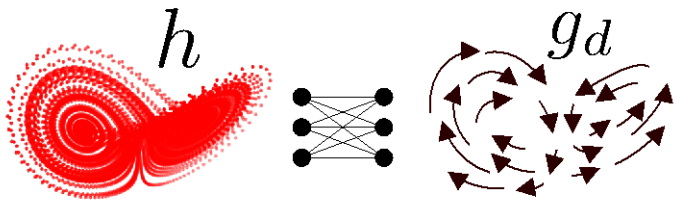
1. **PI-DManD-c**: NODE with additive NN to correct for numerical and FOM errors
2. **PI-DManD-B**: Train NODE g_B with weight initialization θ_{mn} and additional loss term $\lambda\|\theta_{mn} - \theta_B\|^2$ (motivated by Bayes's theorem – treat MG-N model as prior. Equivalently, view as transfer learning)

PI-DManD combines data and physics for forecasting



Autoencoder learns manifold coordinates h

DManD: data only

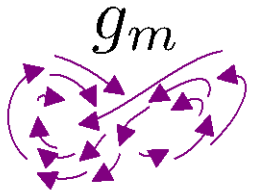


Neural ODE learns $dh/dt = g_d(h)$

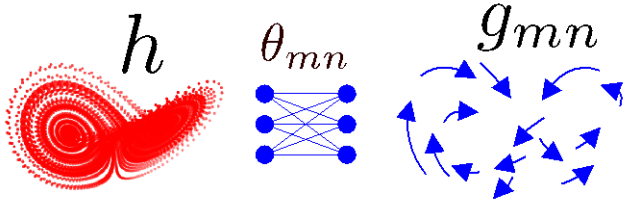
PI-DManD: physics and data

Physics

$$\frac{\partial u}{\partial t} = f_{\text{FOM}}(u)$$

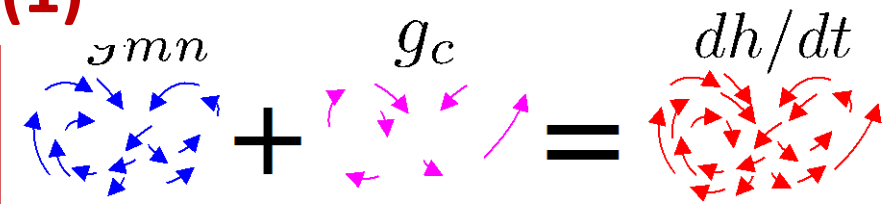


Manifold Galerkin gives
$g_m(h) = J_{\mathcal{E}}(\mathcal{D}(h)) f_{\text{FOM}}(\mathcal{D}(h))$



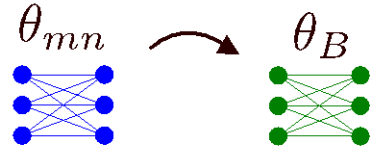
Learn $g_{mn}(h)$ that approximates g_m

(1)



PI-DManD-c: neural ODE g_c corrects for numerical errors and errors in f_{FOM}

(2)



PI-DManD-B: use g_{mn} as Bayesian prior for neural ODE $dh/dt = g_B(h)$

Guo & G. (ArXiv 2025)

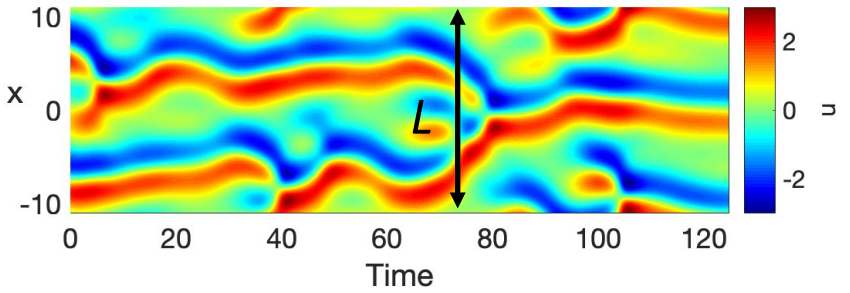
Results: example system

- Kuramoto-Sivashinsky Equation (KSE) exhibits spatiotemporal chaos
  - Domain size L=22 with periodic boundary conditions
  - Invariant manifold dimension is known to be 8 at L=22 (Zeng et al. 2024)
  - Similar results for complex Ginzburg-Landau eq.

- Given an initial condition for the state space vector u, we encode it, use a model (DManD, PI-DManD-c, etc.) to time-integrate it, then the trajectory $h(t)$ is decoded to

- A good forecasting model should have low error for ~1 Lyapunov time (22.2 time units) for test data. We quantify error with

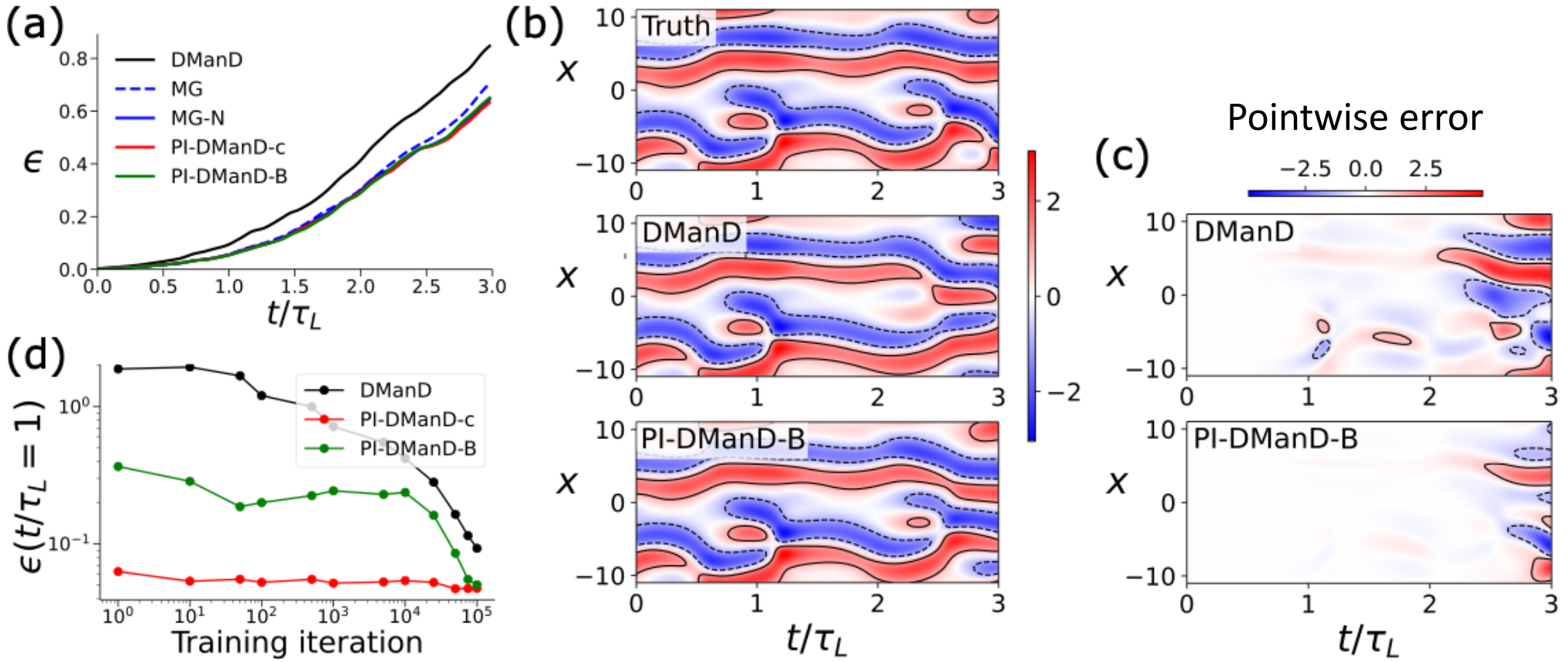
- Several cases are considered: abundant or sparse data, exact or approximate FOM



We test our methods on a chaotic 1D PDE

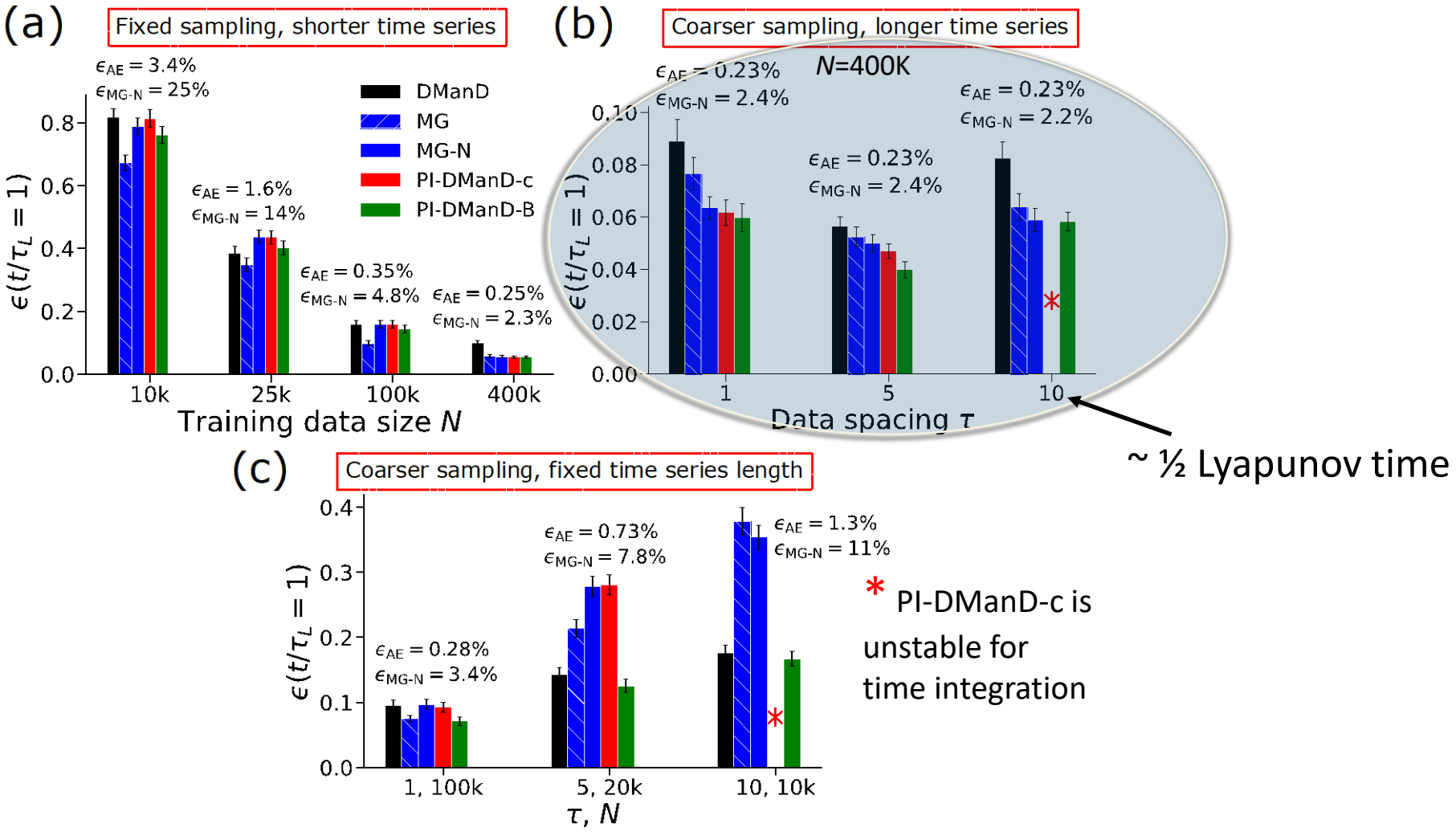
Results: Abundant data, exact FOM

- Abundant data (*N*=400000 training snapshots with data spacing $\tau = 0.25$)
- Exact FOM



Both PI-DManD methods outperform DManD in predictive performance: 40% smaller error at 1 Lyapunov time

Results: scarcer data, exact FOM



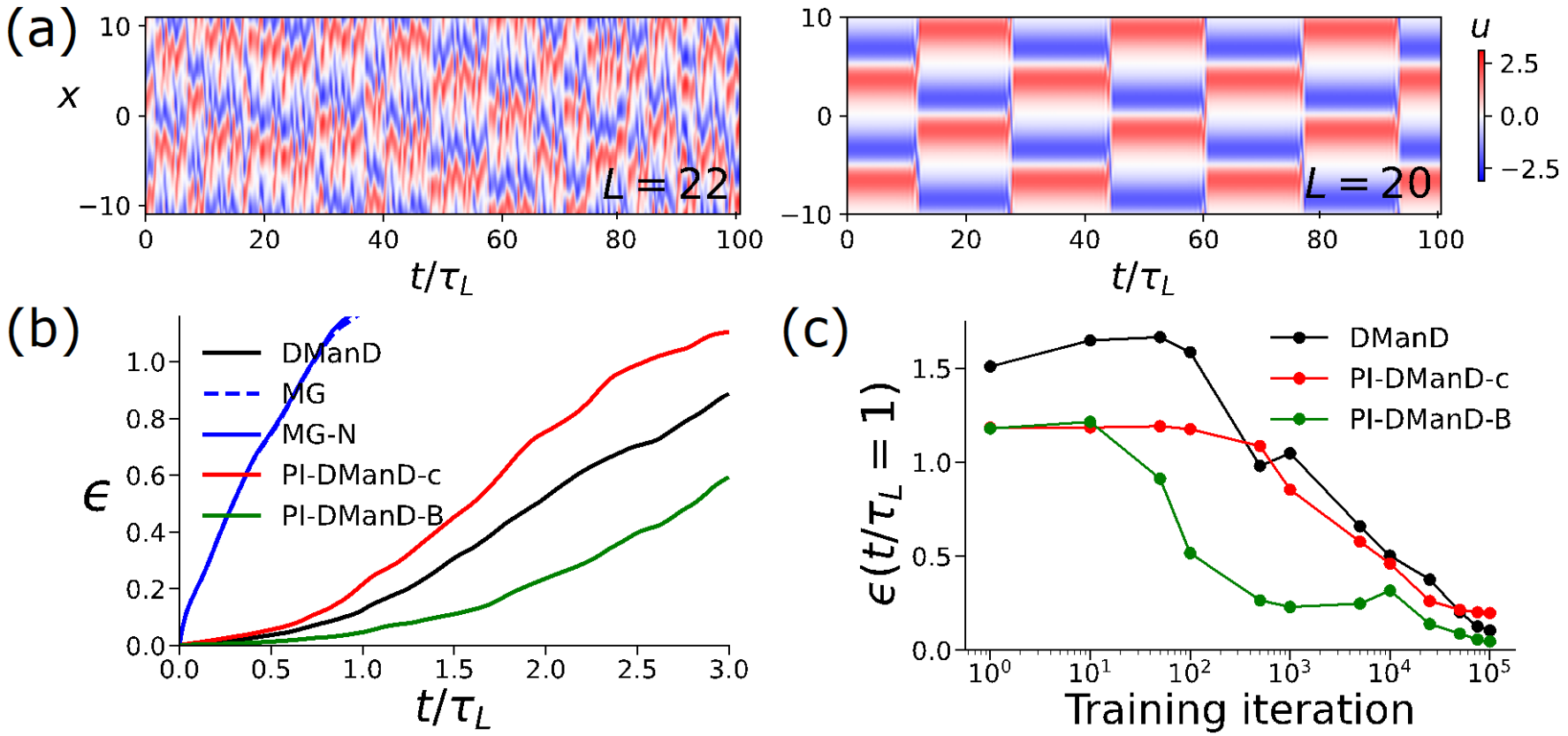
* PI-DManD-c is unstable for time integration

PI-DManD-B performs substantially better for widely-spaced data

Results: FOM with incorrect parameter value

- Abundant, closely spaced data ($N = 400{,}000$ training snapshots with data spacing $\tau = 0.25$)

- Erroneous FOM
  - f_{FOM} uses incorrect domain size of $L = 20$
  - Training+test data still use $L = 22$
  - Represents scenario where exact physics isn't known



Even with inaccurate physics-based information, PI-DManD-B still substantially outperforms DManD

Conclusions and directions

- DManD learns a “low”-dimensional evolution equation (ROM) solely from data
- PI-DManD-B is a hybrid ROM
  - Uses physics by projecting the governing equations onto the underlying invariant manifold
  - Achieves marked improvement over DManD, especially for
    - Widely-spaced data
    - Erroneous model – i.e. even an approximate model can be useful!
  - Yields some improvement in presence of noise if temporal filtering is applied

- What’s next?
  - Higher-dimensional systems – here AE error seems to be limiting (e.g. in turbulent Couette flow)
  - Partial observations
  - Other physical systems esp. soft matter dynamics ➔ data-driven microstructural evolution models
  - Distributed/hierarchical models for systems with multiscale dynamics
  - Control

Thanks!



DManD: Basic Framework

Mapping to Manifold Coordinates (Autoencoder)

$$u \in \mathbb{R}^d$$
$$h \in \mathbb{R}^{d_h}, d_h \ll d$$

Data (time series)

Input $[u(t_1), u(t_2), \ldots, u(t_M)]$

Output $[u(t_1), u(t_2), \ldots, u(t_M)]$

Loss

NN parameters

reconstruction $\tilde{u}_i$

- Symmetry reduction (split out spatial phase from pattern, build in discrete symmetries) improves representation
- Ways to choose a good d_h (estimate of manifold dimension) are evolving

Linot and G., PRE (2020), Chaos (2022). Perez De Jesus and G. PRF (2023, 2025), Zeng et al ML:S&T (2024),

Evolution in Manifold Coordinates (Neural ODE)

$$\frac{dh}{dt} = g(h)$$

Data (already processed through autoencoder)

Input $[h(t_1), h(t_2), \ldots, h(t_M)]$

Output $[h(t_1 + \tau), h(t_2 + \tau), \ldots, h(t_M + \tau)]$

Loss

NN parameters

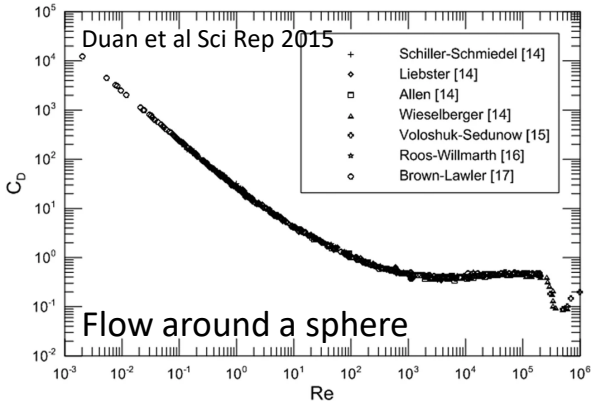
Time-integration of ODE

- To determine gradient of loss
  - Treat ODE as constraint, use adjoint method
  - Or just do automatic differentiation through an ODE solver
- Adding an explicit dissipative term can stabilize (Linot et al 2023)
- Multi-step loss can also be used

Chen et al., NeurIPS (2018)

Data, dynamics, and symmetry

Exploiting symmetry enables more effective use of data



Independence of physical laws from units of measurements = a form of dilation symmetry → dimensional analysis

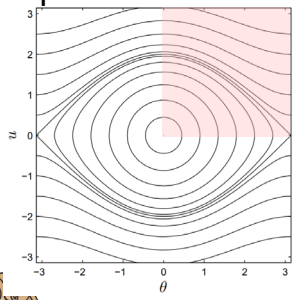
Drag force, velocity, viscosity, density, diameter → drag coefficient, Reynolds number

Periodic domain & translation invariance

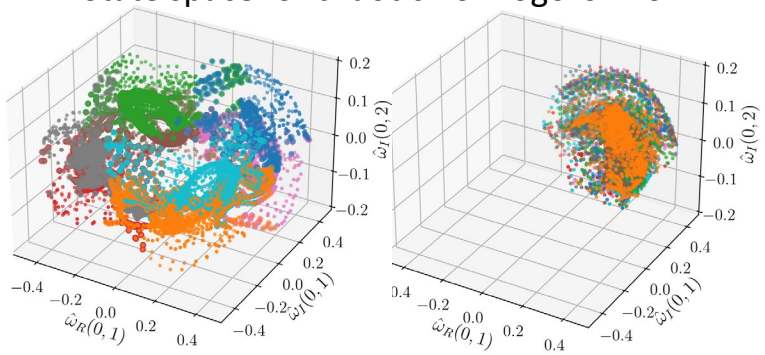
→ data can be separated into pattern and phase -- i.e. we don't have to learn separate representations of the same pattern at different rotation angles



State space for pendulum



State space for chaotic Kolmogorov flow



Symmetries in physical space lead to symmetries between different parts of state space

→ we only have to capture dynamics in part of state space and symmetry gives us the rest

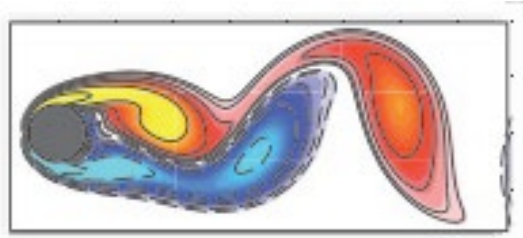
Linot & G, PRE 2020, Zeng & G. PRE 2021, Perez de Jesus & G. 2023

Other approaches to time-evolution prediction with data-driven models

DMD/EDMD/Koopman:

- linear dynamical model, but formally infinite dimensional
- Very successful for weakly nonlinear problems on low-d manifolds– periodic, quasiperiodic
- Need to introduce nonlinearity to capture chaotic dynamics → becomes equivalent to NODEs!

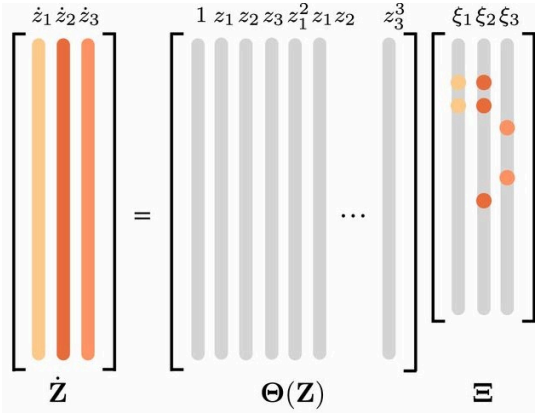
$$\psi_{k+1} = K\psi_k$$



Schmid, Rowley, Mezic, Kutz, Brunton, Kevrekidis, Bollt,…

SINDy/symbolic regression:

- learns form of vector field from a dictionary
- ➢ Limited/pre-specified structure
- ➢ Low-dimensional systems

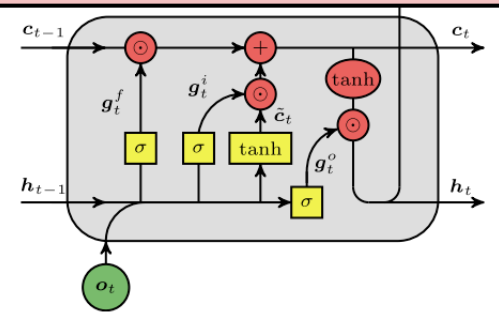


Champion et al., PNAS (2019)

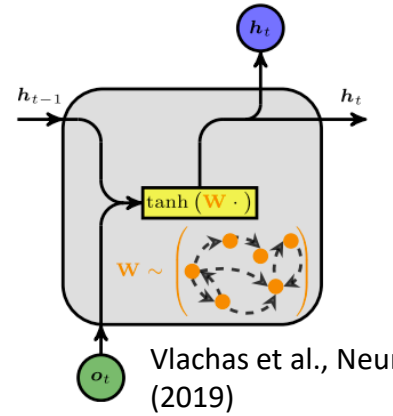
LSTM/Reservoir Computing/Transformers:

- can model high-dimensional chaotic systems (KSE)
- Recurrent non-Markovian structure – state dimension generally increases – *not reduced order*

LSTM Cell



RC Cell

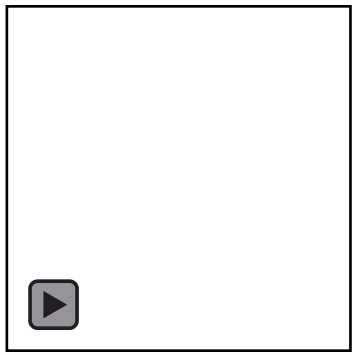


Vlachas et al., Neural Networks (2019)

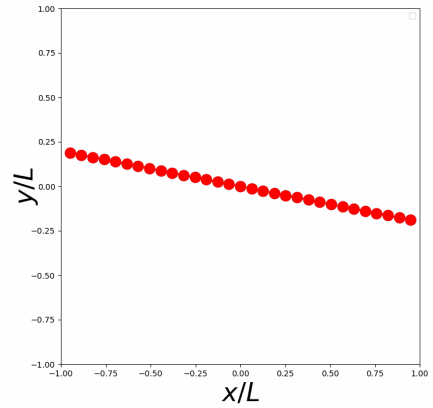
Our approach:

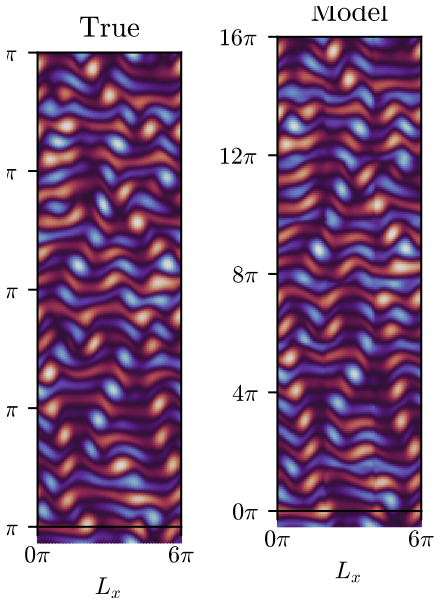
- Accurate reduced state representation.
- Retains Markov property of ODEs.
- Can use data that is sparse in time – no derivatives

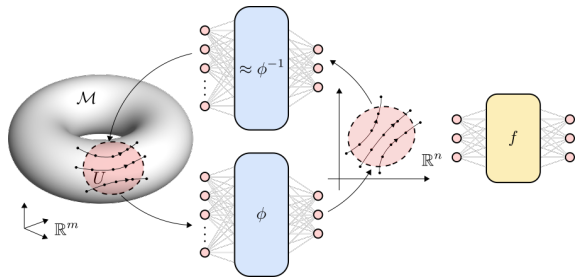
What's next?



- More complex systems – higher Re, other flow geometries, elastoinertial turbulence, microstructure in complex fluids
- Longer prediction horizons
- Parameter-dependence
- Coupled local models -- "CANDyMan" (Floryan et al. Nature MI 2022)
- Precise and robust dimension estimates – other approaches?
- Distributed/hierarchical models for systems with multiscale dynamics
- Include/enforce geometric information on data/dynamics
- Approximate methods for dynamics with very high dimension – keep the "thick" directions, model the "thin" ones (this would be a version of LES)
- Physics-Informed/Bayesian DManD – use data and governing equations to find evolution on $\mathcal{M}$

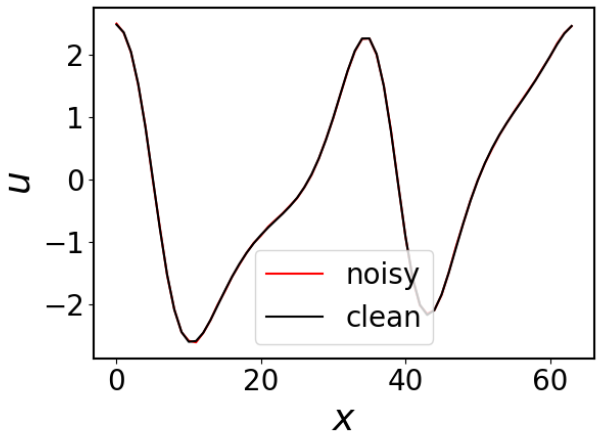




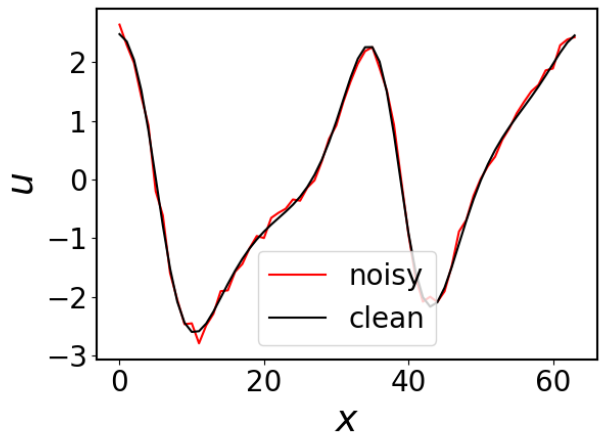


KSE: noisy vs. clean data (in space and time)

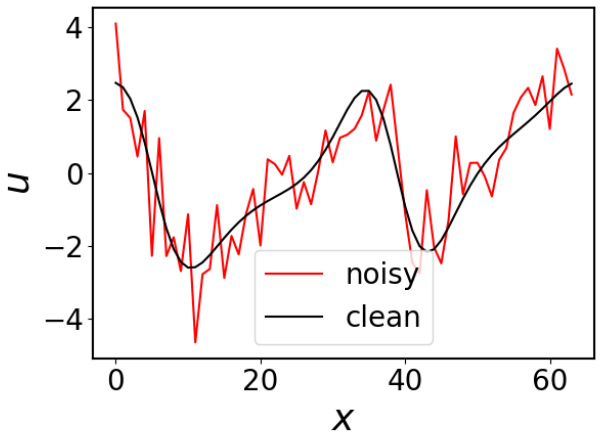
$\sigma = 0.01$



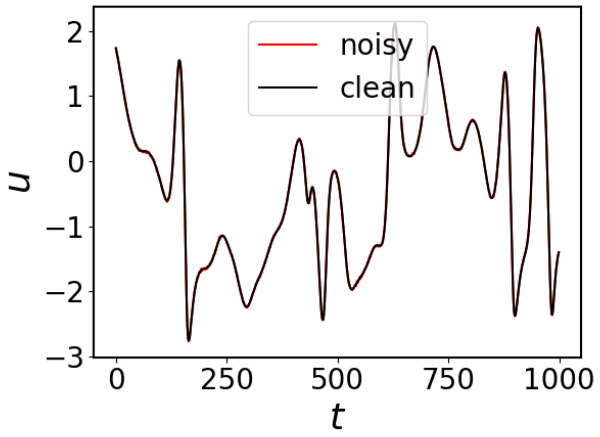
$\sigma = 0.1$

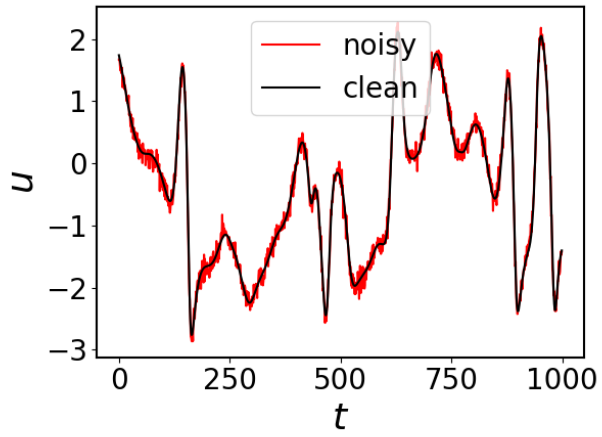


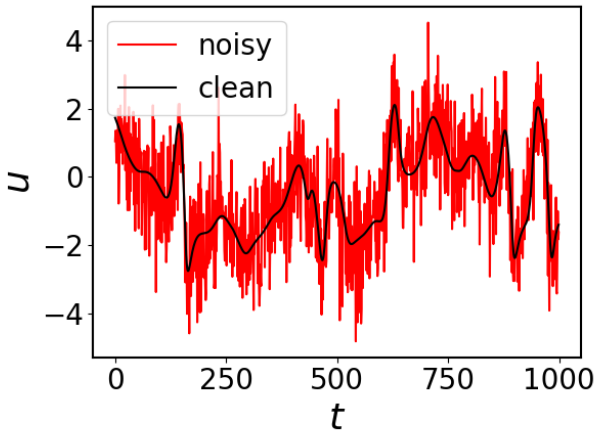
$\sigma = 1$



How robust are our models to varying levels of noise?

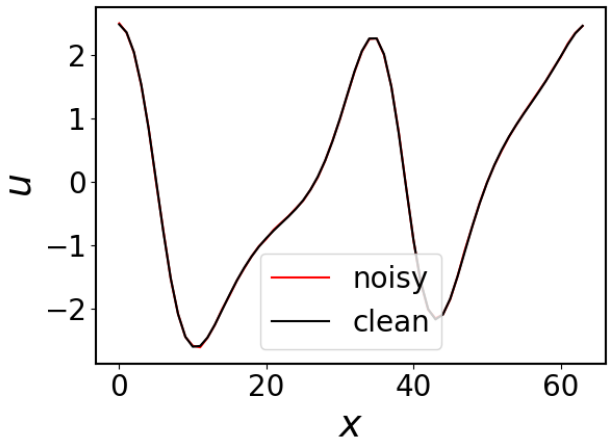




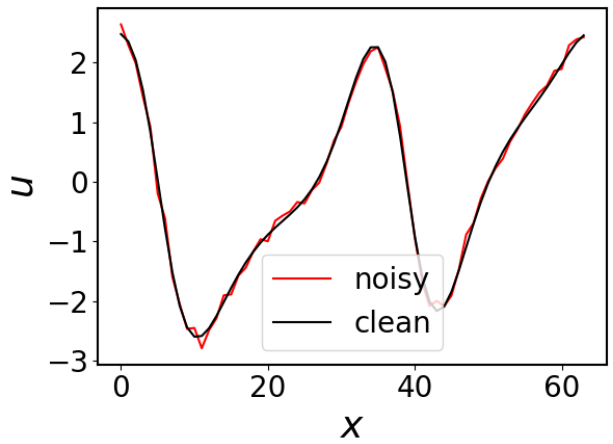


AE reconstruction (space)

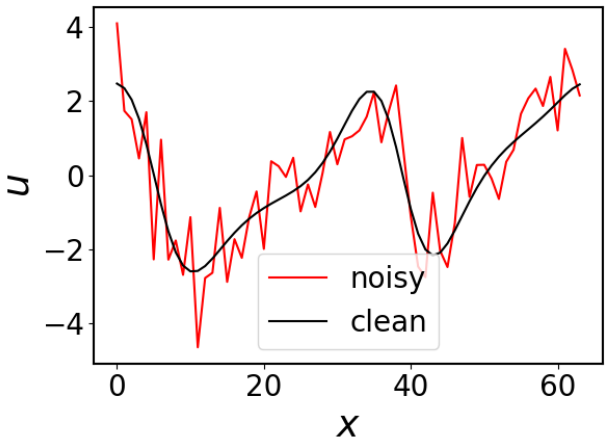
$\sigma = 0.01$



$\sigma = 0.1$

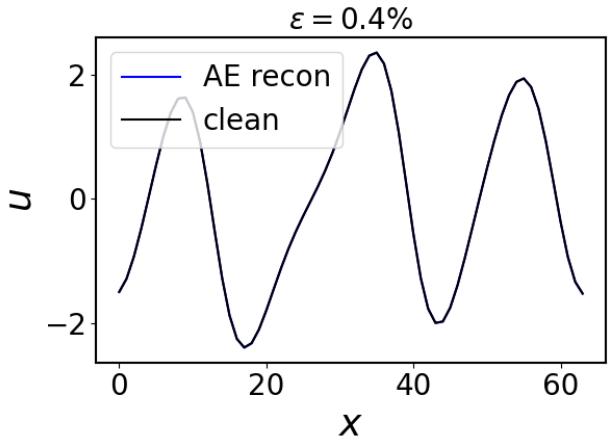


$\sigma = 1$

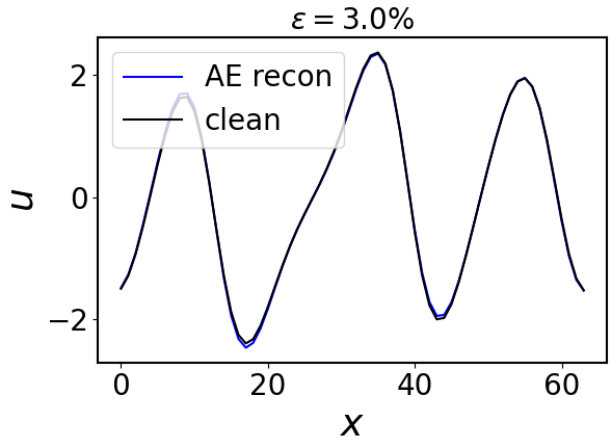


AE gets rid of some noise in space

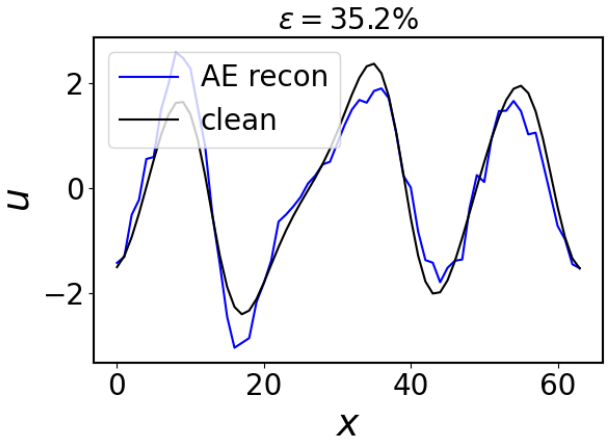
$\varepsilon = 0.4\%$



$\varepsilon = 3.0\%$

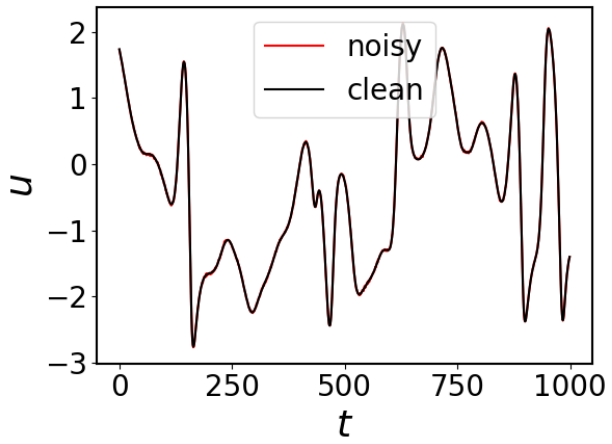


$\varepsilon = 35.2\%$

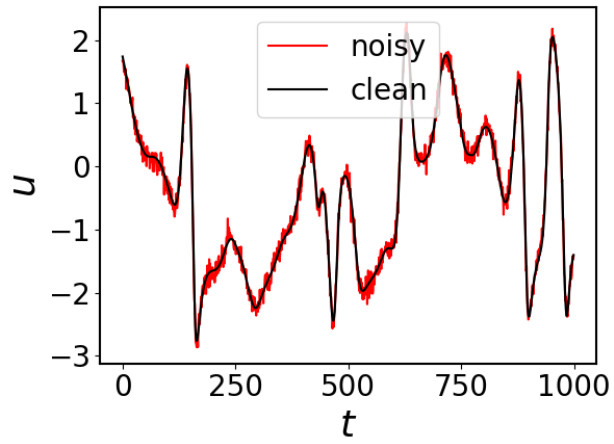


AE reconstruction (time)

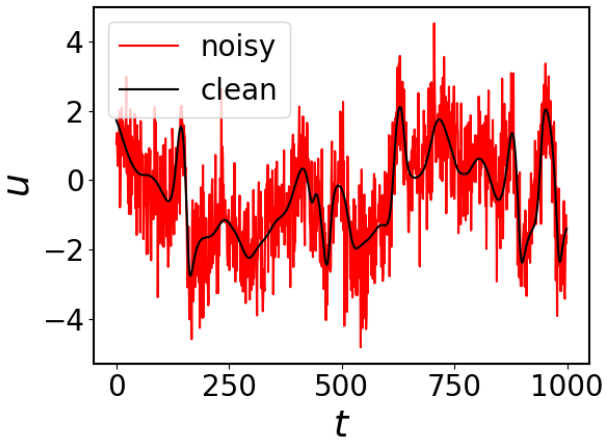
σ = 0.01



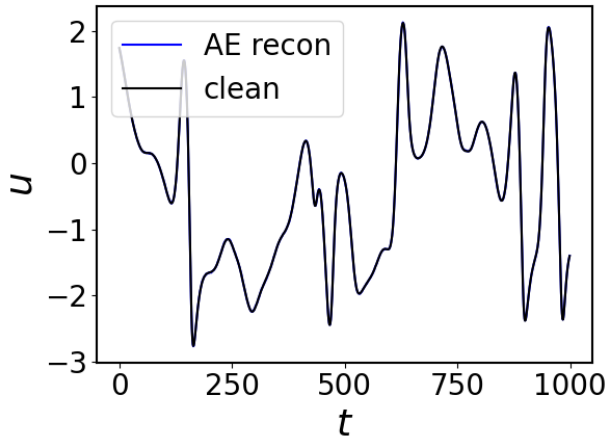
σ = 0.1

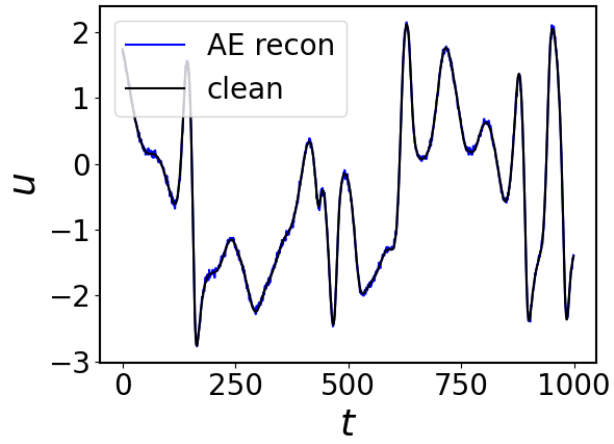


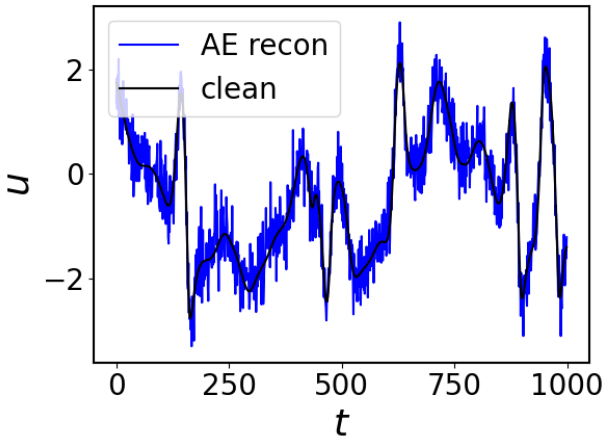
σ = 1



AE gets rid of some noise in time (less so than in space)

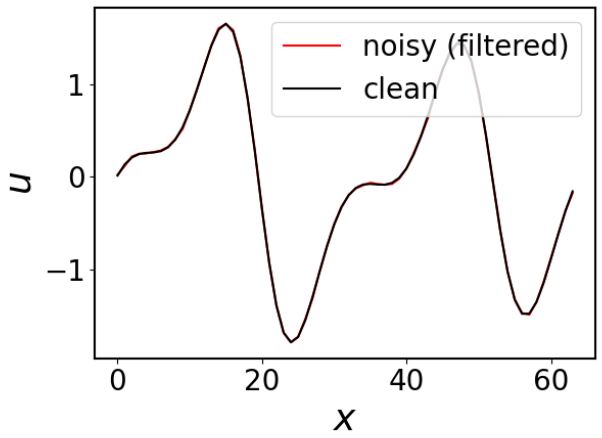




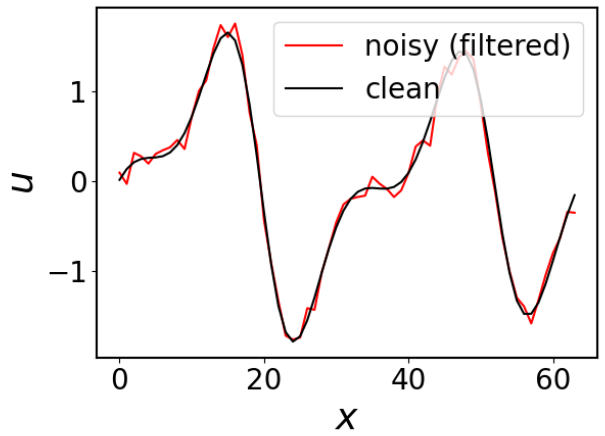


Noise (filtered)

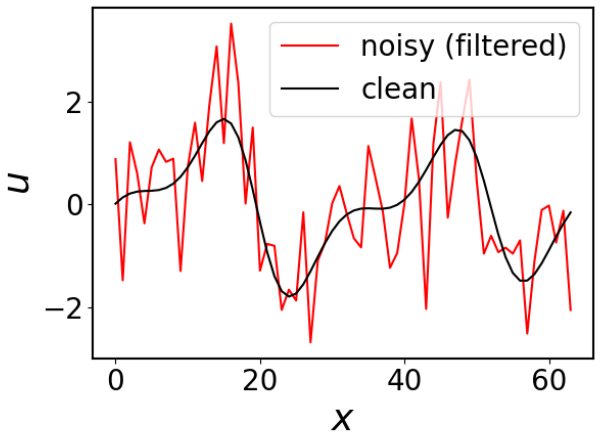
σ = 0.01



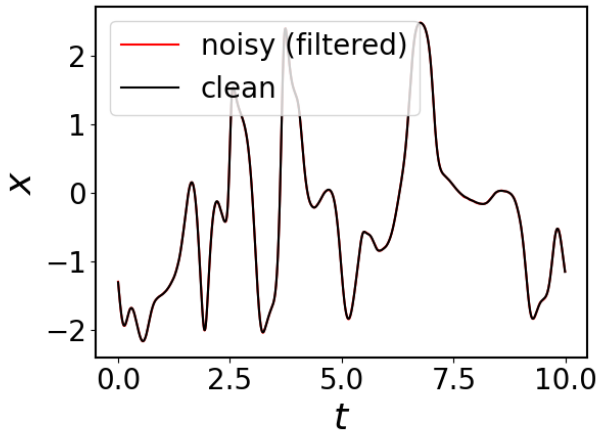
σ = 0.1

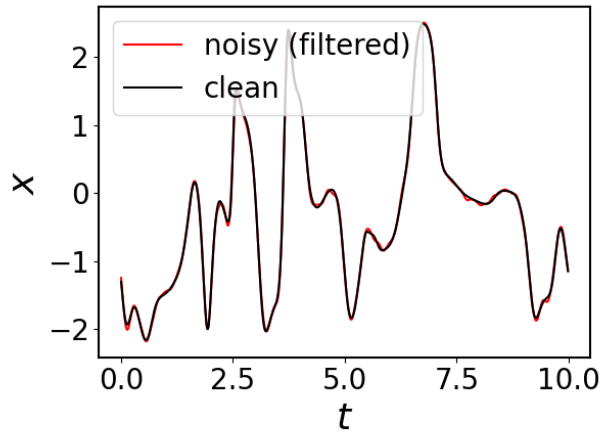


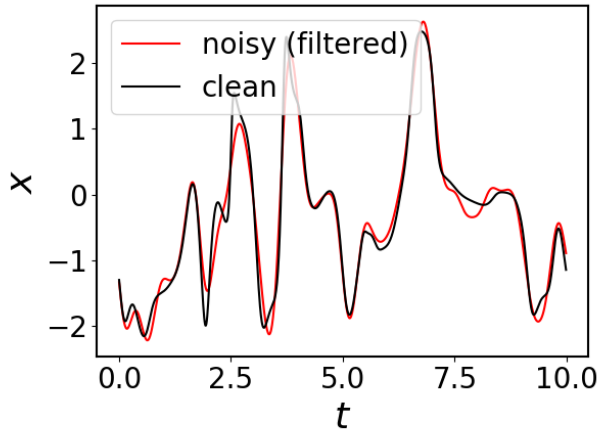
σ = 1



How robust are our models to varying levels of noise if we filter in time?

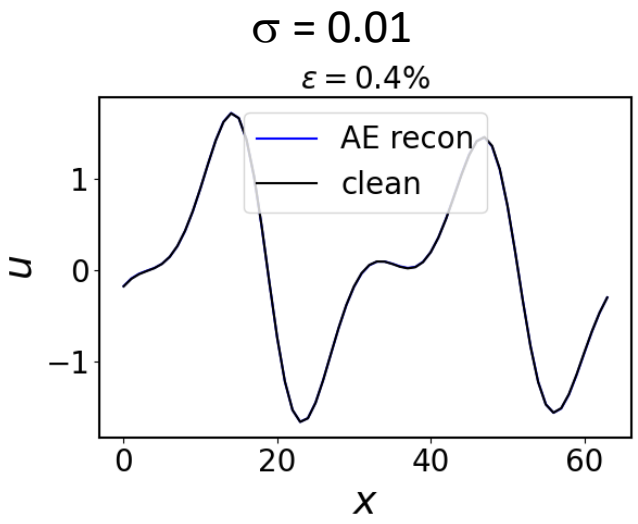


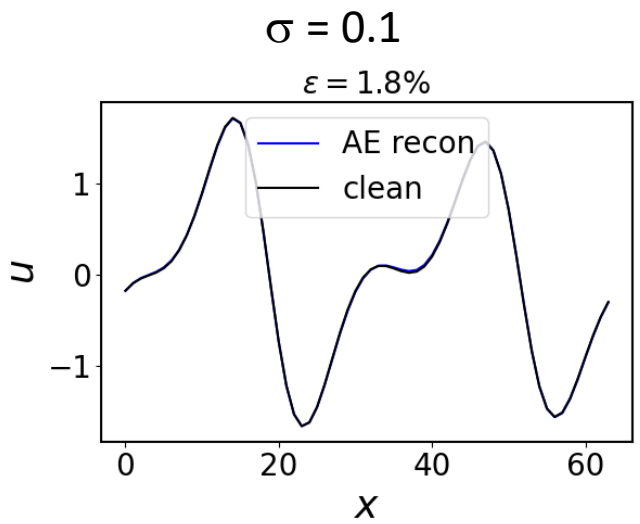


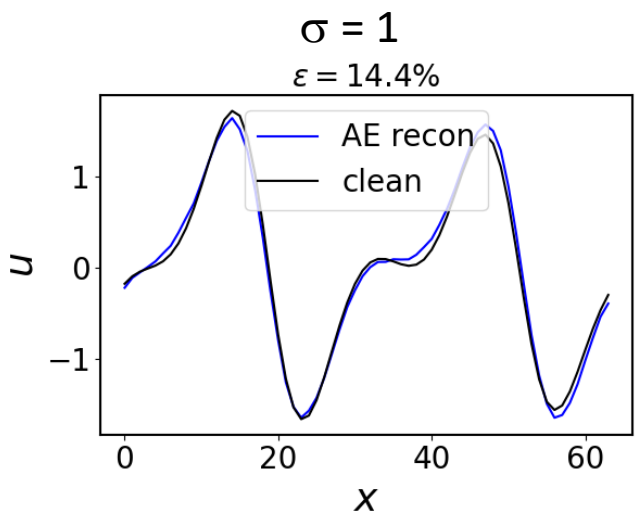


Butterworth filter with cutoff frequency at"bend" in PSD

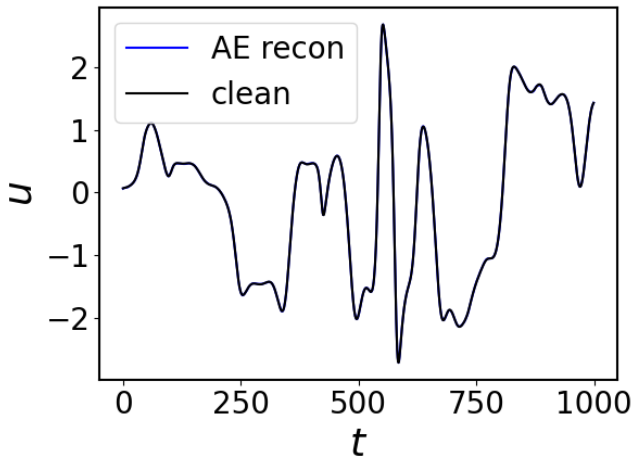
AE reconstruction

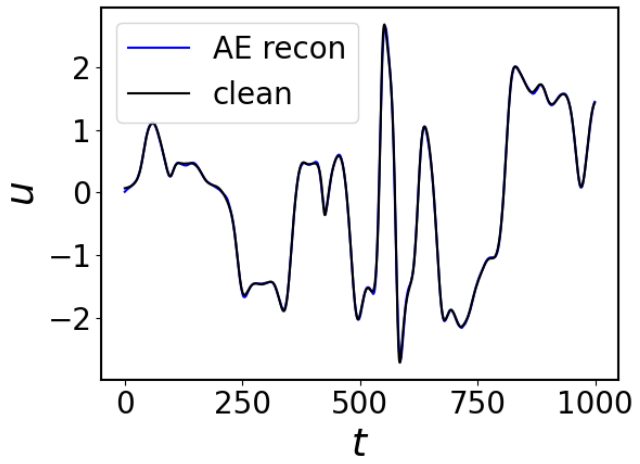


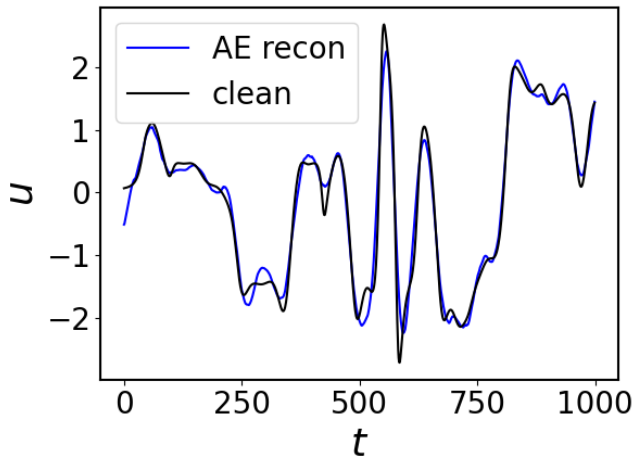




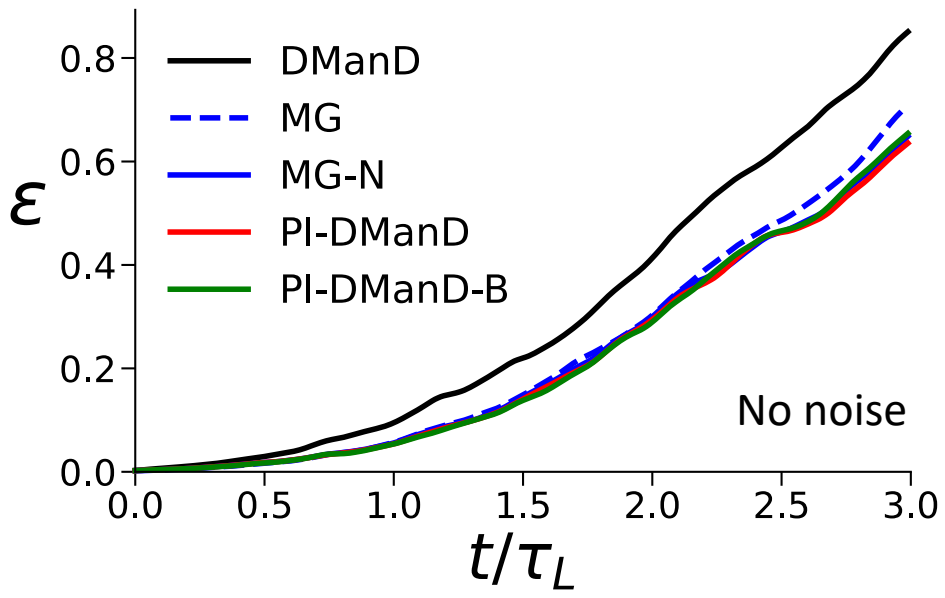
Filtering in time and using the AE gets rid of most noise in both space and time

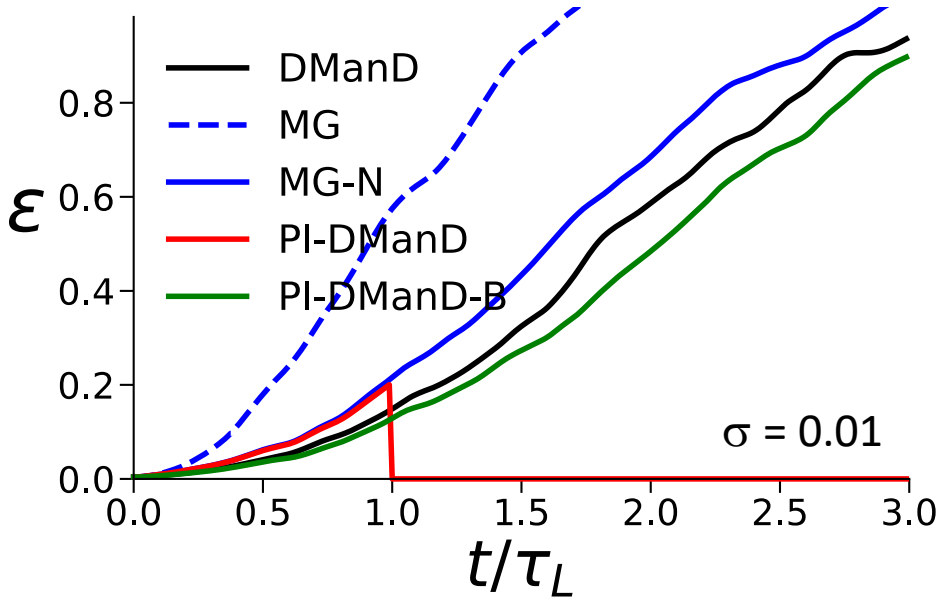




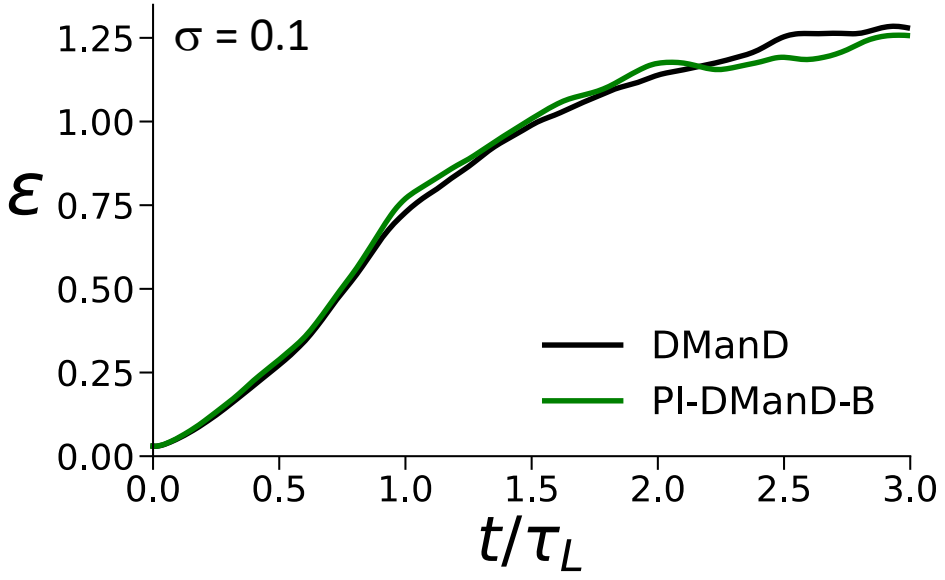


Prediction error (no temporal filtering)





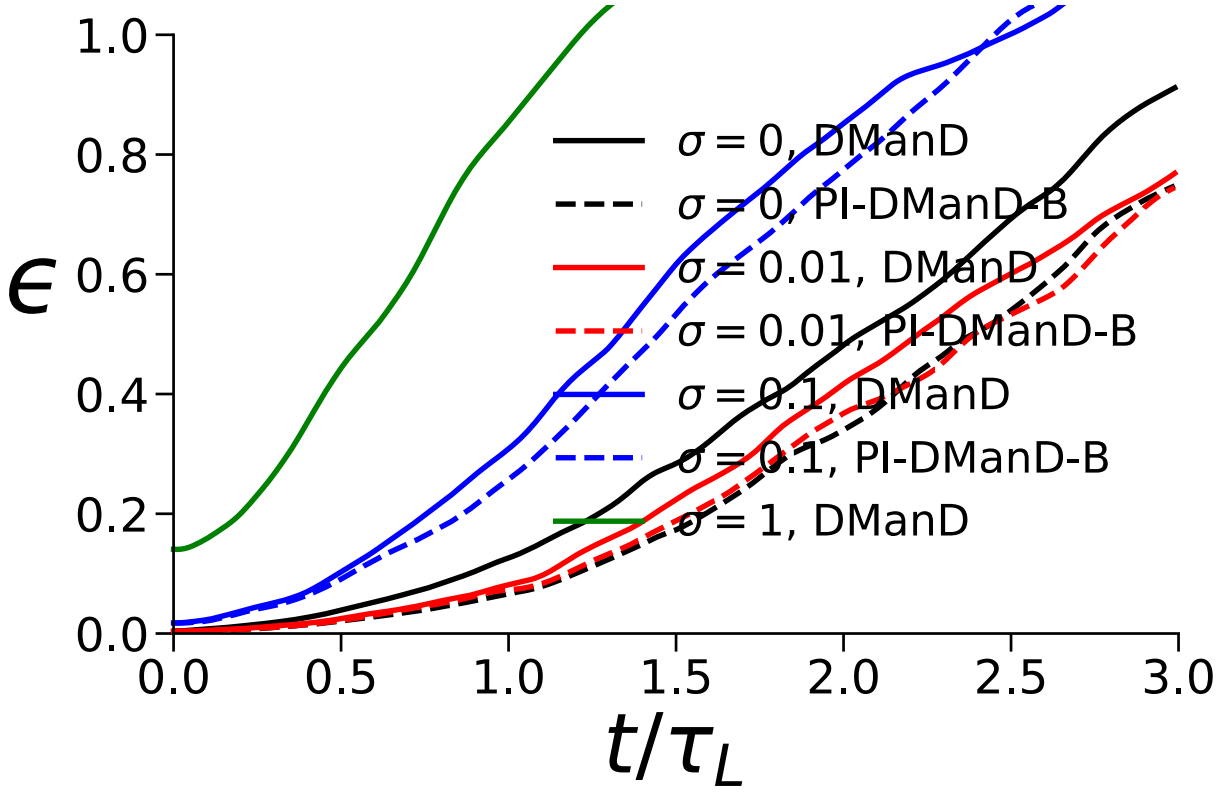
Tracking error increases with increasing noise. PI-DManD-B performs about the same as or slightly better thanDManD



Prediction error (temporal filtering)

- All data: IC+ground truth are noisy+filtered signals
- For earlier results without filtering, $\epsilon(t/\tau_L = 1)$ was 0.75 for $\sigma = 0.1$ for DManD – filtering in time helps

PI-DManD-B model doesn't train for $\sigma = 1$



PI-DManD-B outperforms DManD for no noise, low noise, and moderate noise