UNIVERSITY OF CAMBRIDGE

Deep Learning for Koopman Operator Optimal Control

Tom Lu

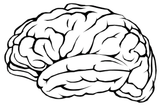
3 June 2020

Motivation

Part IB Paper 6: Information Engineering

LINEAR SYSTEMS AND CONTROL

$$\mathbf{x}_{k+1} = A\mathbf{x}_k$$



$$\mathbf{x}_{k+1} = F(\mathbf{x}_k)$$

Koopman Framework Overview

The Koopman Operator

$$\mathbf{x}_{k+1} = F\left(\mathbf{x}_k\right) \qquad \mathbf{x} \in \mathbb{R}^n$$

$$? \; \boxed{\mathbf{y} = g(\mathbf{x})} \qquad g : \mathbb{R}^n \rightarrow \mathbb{C}^m$$

$$\mathbf{y}_{k+1} = A\mathbf{y}_k \qquad A \in \mathbb{C}^{m \times m}$$

$$\begin{tikzcd}
\mathbf{y}_0 \arrow[r, "A"] \arrow[d, dotted, bend left] & \mathbf{y}_1 \arrow[r, "A"] \arrow[d, dotted, bend left] & \mathbf{y}_2 \arrow[r, "A"] \arrow[d, dotted, bend left] & {} \\
\mathbf{x}_0 \arrow[u, bend left, "g"] \arrow[r, "F"'] & \mathbf{x}_1 \arrow[u, bend left, "g"] \arrow[r, "F"'] & \mathbf{x}_2 \arrow[u, bend left, "g"] \arrow[r, "F"'] & {}
\end{tikzcd}$$

The Koopman Operator Optimal Control

$$\mathbf{x}_{k+1} = F\left(\mathbf{x}_k, \mathbf{u}_k\right)$$

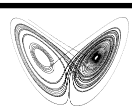
$$\mathbf{y}_{k+1} = A\mathbf{y}_k + B\mathbf{u}_k$$

$$\hat{\mathbf{x}}_{k+1} = C\mathbf{y}_{k+1}$$

$$J = \int_0^\infty \left(\mathbf{y}^T Q \mathbf{y} + \mathbf{u}^T R \mathbf{u}\right) dt$$

$$\mathbf{u} = -K\mathbf{y}$$

Nonlinear System



$\mathbf{x}_0$

$\mathbf{x}_{k+1} = F(\mathbf{x}_k, \mathbf{u}_k)$

→ **Linear Predictor**

$\mathbf{y}_{k+1} = A\mathbf{y}_k + B\mathbf{u}_k$

$\mathbf{y}_0 = g(\mathbf{x}_0)$

$dim(\mathbf{y}) > dim(\mathbf{x})$

→ $\mathbf{x} = C\mathbf{y}$ → Easy, LQR

→ $\mathbf{x} = g^{-1}\mathbf{y}$ → Difficult. KMPC

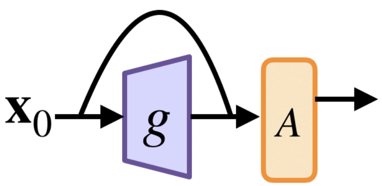
Linear Control Design

Methods

$$\mathbf{y} = g(\mathbf{x})$$
$$\mathbf{y}_{k+1} = A\mathbf{y}_k$$

LREN:

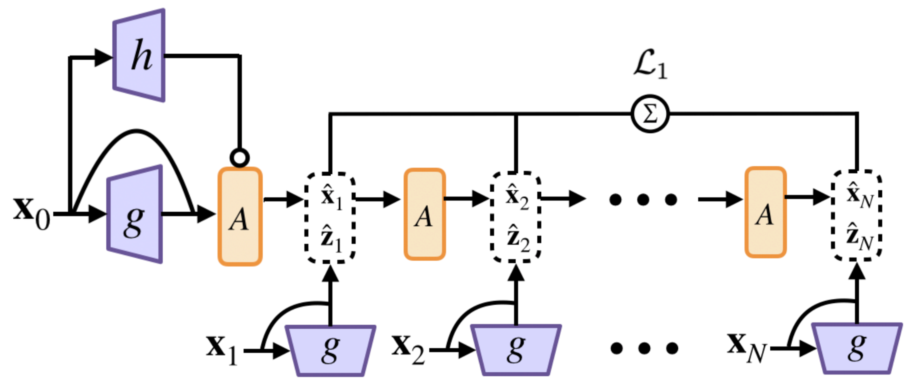
Linearly **R**ecurrent **E**ncoder **N**etwork



5X faster to train [Otto et al. 2016]

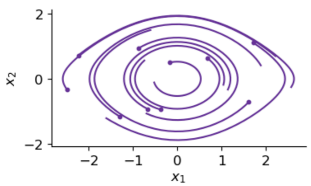
DENIS:

Deep **E**ncoder **N**etwork with
Initial **S**tate **P**arameterisation

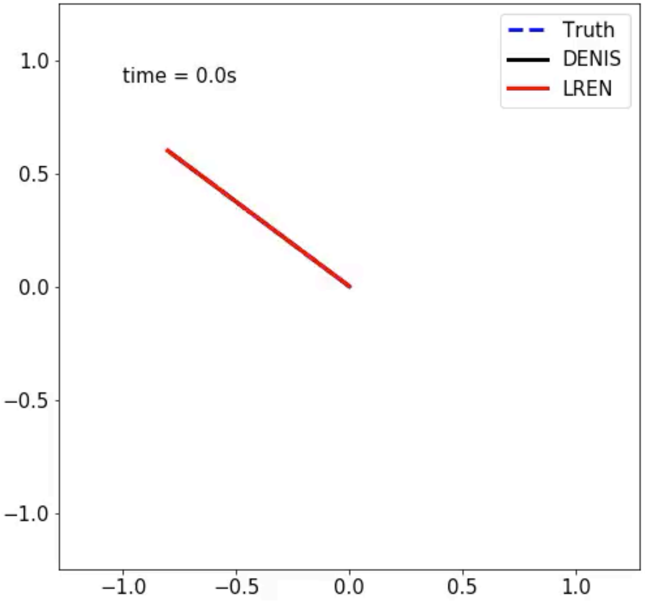


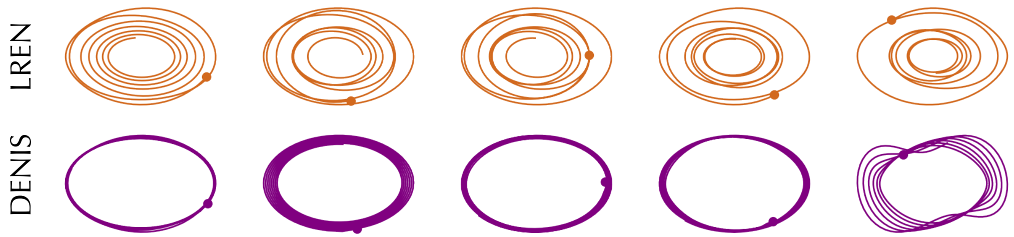
Prediction performance evaluation

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -\sin(x_1)$$

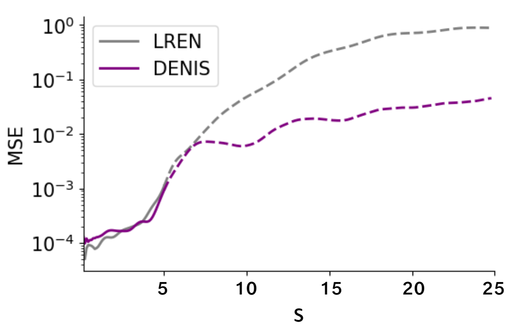


Nonlinear Pendulum

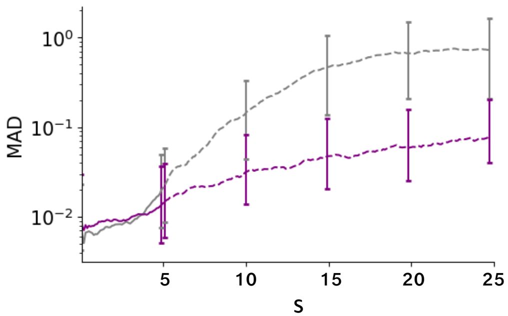




Mean Squared Error VS Time

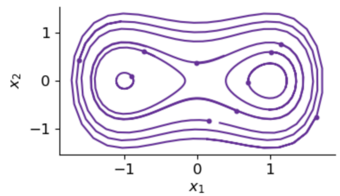


Median of Absolute Deviation VS Time

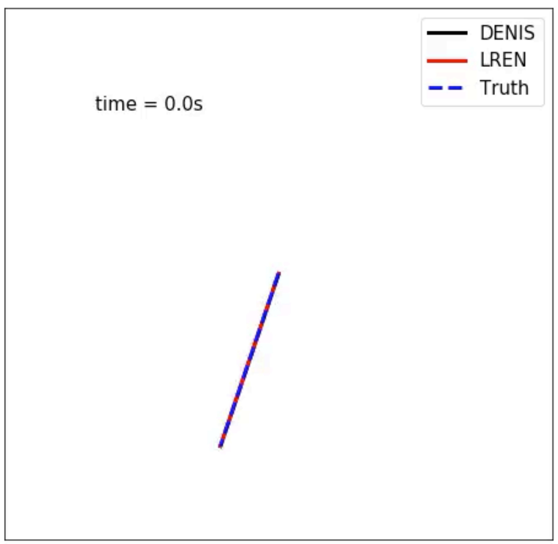


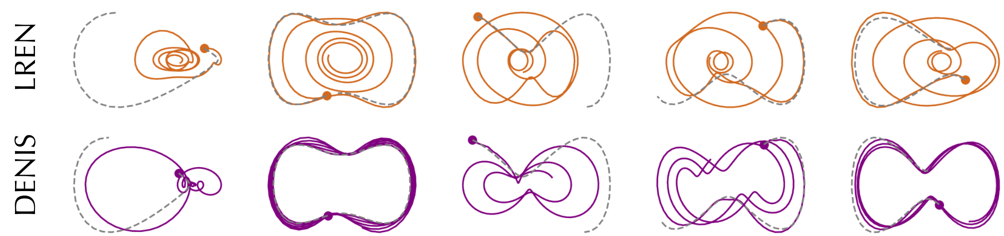
Prediction performance evaluation

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = x_1 - x_1^3$$

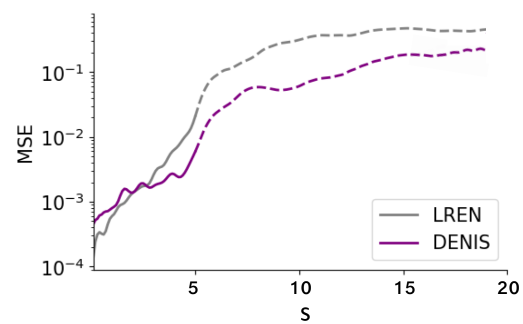


Duffing Oscillator

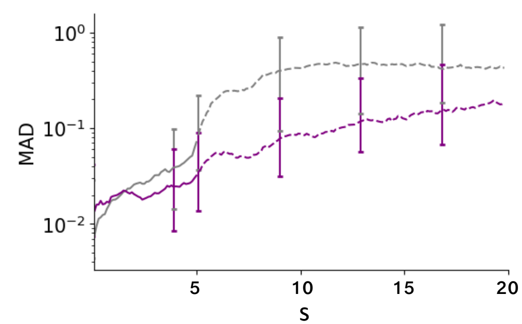




Mean Squared Error VS Time

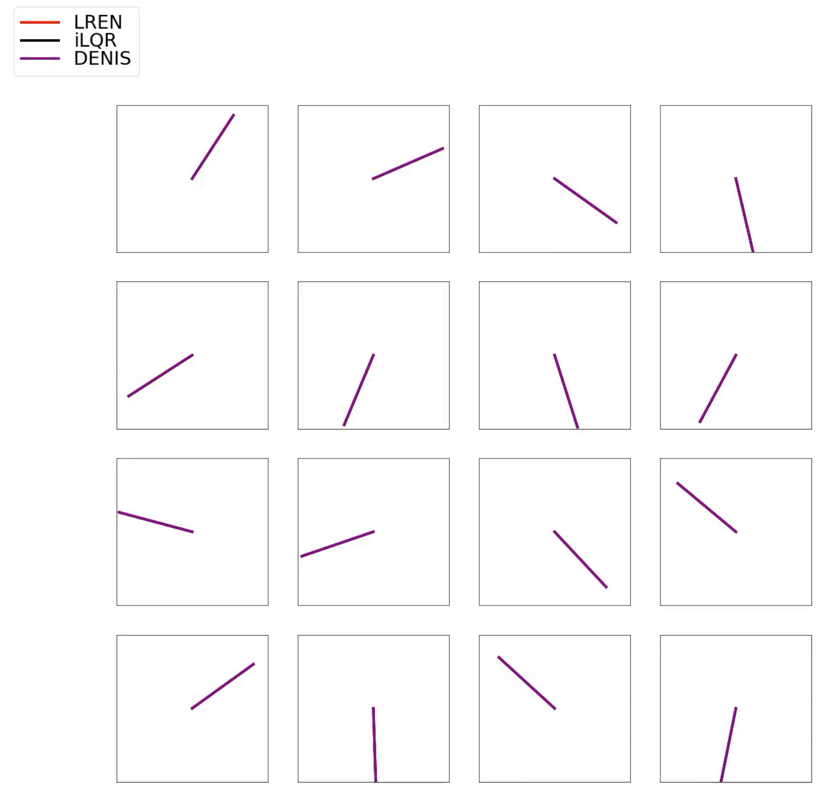


Median of Absolute Deviation VS Time



Control performance evaluation

Closed loop control driving a **pendulum** to up/down positions

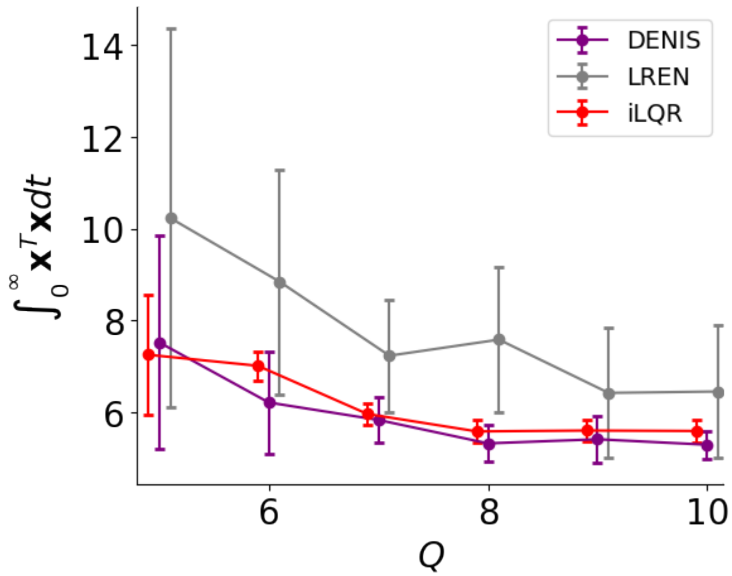


Control performance evaluation

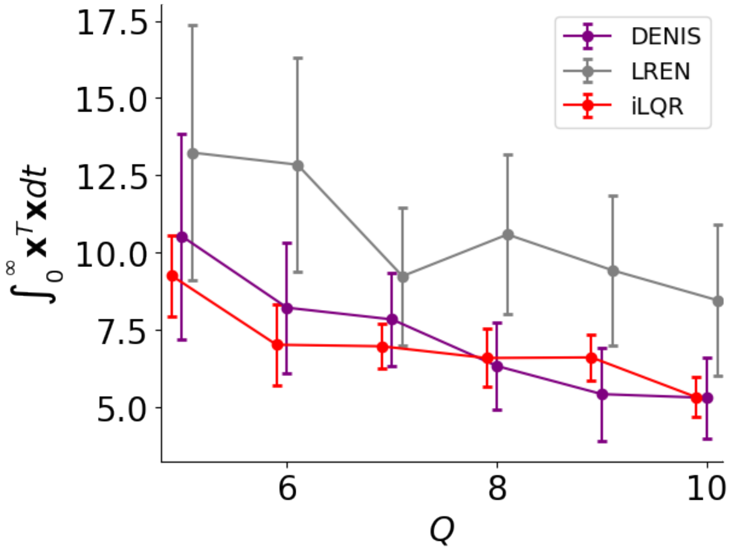
How do we compare different control schemes? $J = \int_0^\infty \left(\mathbf{x}^T Q \mathbf{x} + \underbrace{\mathbf{u}^T R \mathbf{u}}_{\text{fix}} \right) dt$

Fix total input $\int_{t=0}^{\infty} \mathbf{u}^T \mathbf{u} dt = C_u$ find total optimal state cost $\int_{t=0}^{\infty} \mathbf{x}^T \mathbf{x} dt$

Pendulum

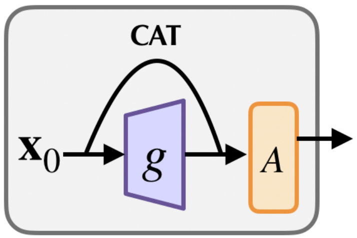


Duffing Oscillator



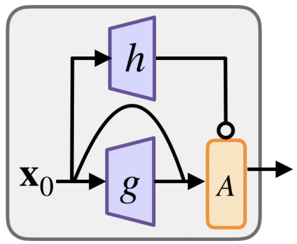
Summary

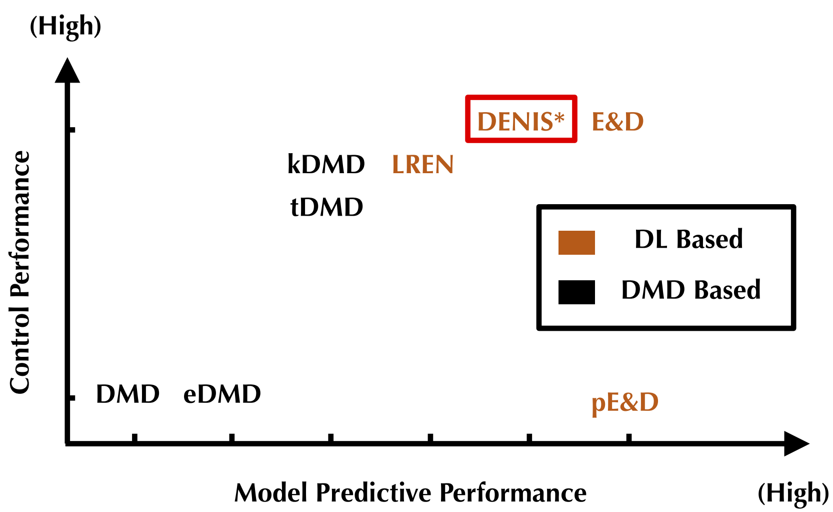
LREN

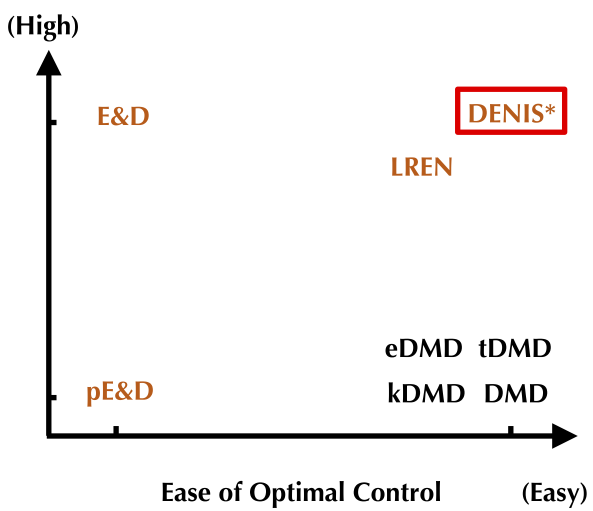


5X faster to train

DENIS*







Supplementary Information

Loss function for all networks:

$$\mathscr{L} = \sum_{i=1}^{5} \alpha_i \mathscr{L}_i$$

Weighted state reconstruction loss:

$$\mathscr{L}_1 = \frac{1}{TB\sum_k c_k} \sum_{b=1}^{B} \sum_{k=1}^{T} c_k \left\| \hat{\mathbf{x}}_{k,b} - \mathbf{x}_{k,b} \right\|_2$$

Weighted latent linear loss:

$$\mathscr{L}_2 = \frac{1}{TB\sum_k c_k} \sum_{b=1}^{B} \sum_{k=1}^{T} c_k \left\| \hat{\mathbf{z}}_{k,b} - \mathbf{z}_{k,b} \right\|_2$$

Maximum deviation loss:

$$\mathscr{L}_3 = \frac{1}{B} \sum_{b=1}^{B} \| \|\hat{\mathbf{x}}_{k,b} - \mathbf{x}_{k,b} \|_2 \|_\infty$$

Zero loss:

$$\mathscr{L}_4 = \|g(0)\|_2$$

Energy budget analytical solution:

Theorem 1: The expected total controller input across all initialization is:

$$\int_0^\infty \left\langle \mathbf{u}^T \mathbf{u} \right\rangle_{x_0} dt = R^{-1}\,\mathrm{Tr}(P - M) \quad \text{(IV.31)}$$

where M is the solution through the Lyapunov equation:

$$(A + BK)M + M(A + BK)^T + Q = 0 \quad \text{(IV.32)}$$

References:

S. E. Otto and C. W. Rowley, "Linearly Recurrent Autoencoder Networks for Learning Dynamics," *SIAM Journal on Applied Dynamical Systems, 2019*

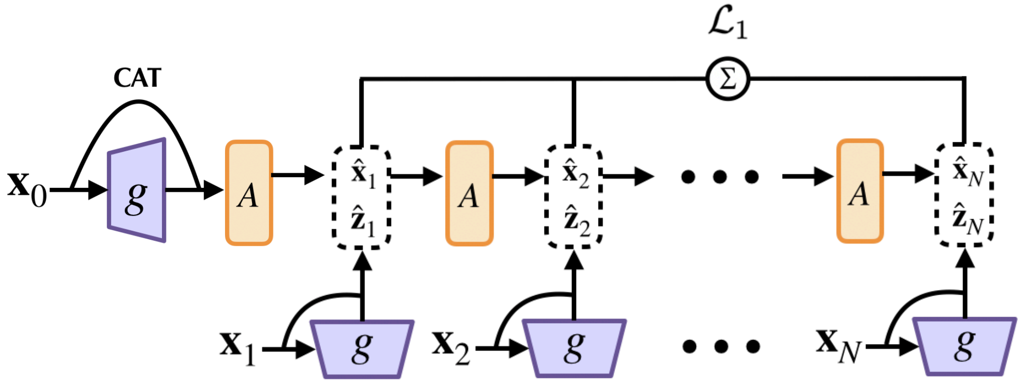
B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature Communications, 2018*

B. O. Koopman, "Hamiltonian Systems and Transformation in Hilbert Space," Proceedings of the National Academy of Sciences, vol. 17, no. 5, pp. 315–318, May 1931

M. Korda and I. Mezic, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control," Automatica, vol. 93, pp. 149–160,

LREN

Linearly **R**ecurrent **E**ncoder **N**etwork

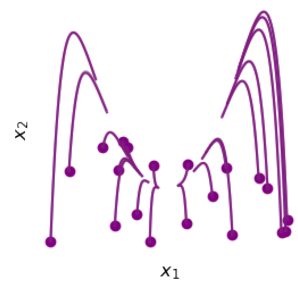


$$\dot{x}_1 = \mu x_1$$

$$\dot{x}_2 = \lambda \left(x_2 - x_1^2 \right)$$

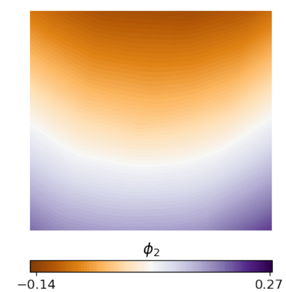
$$\mathbf{y} = \begin{bmatrix} x_1 & x_2 & x_1^2 \end{bmatrix}^T$$

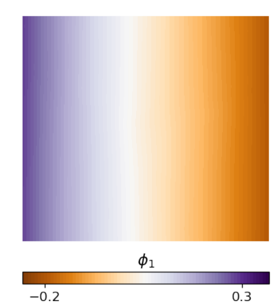
$$\dot{\mathbf{y}} = \begin{bmatrix} \mu & 0 & 0 \\ 0 & \lambda & -\lambda \\ 0 & 0 & 2\mu \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = A\mathbf{y}$$



$$\phi_1 = x_1$$

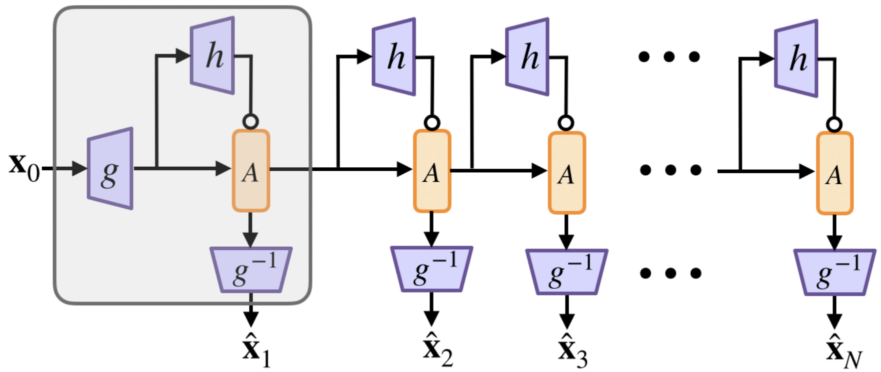
$$\phi_2 = x_2 - b x_1^2$$





DENIS

Deep **E**ncoder **N**etwork with **I**nitial **S**tate Parameterisation

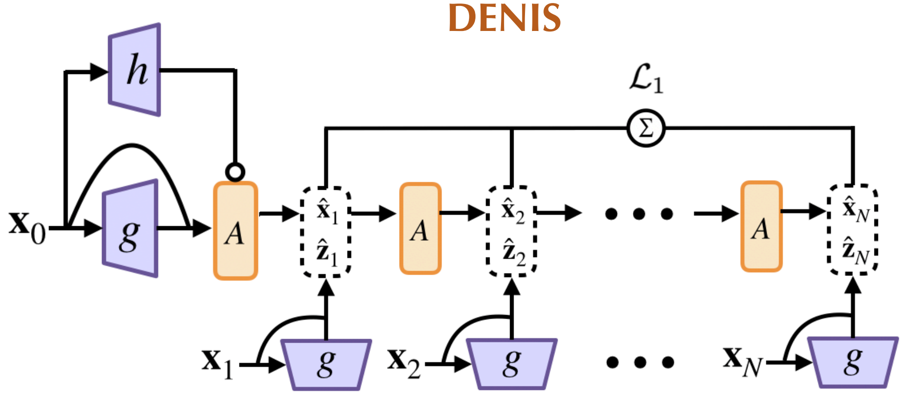


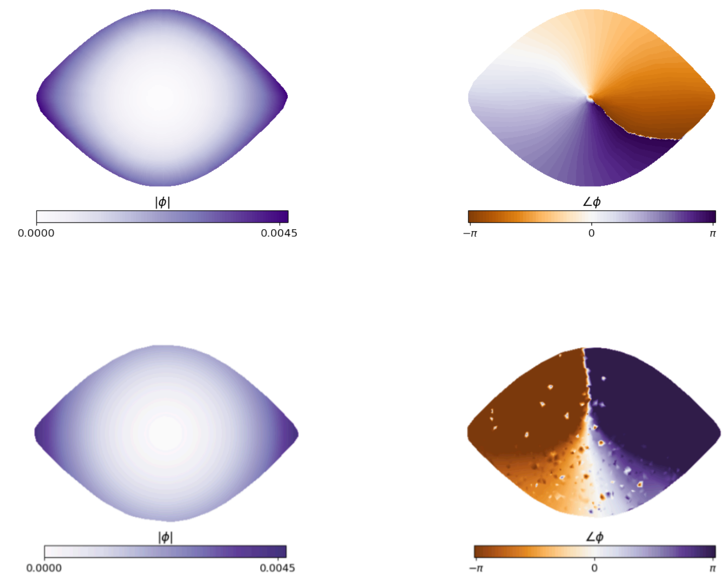
[Lusche et al. 2018]

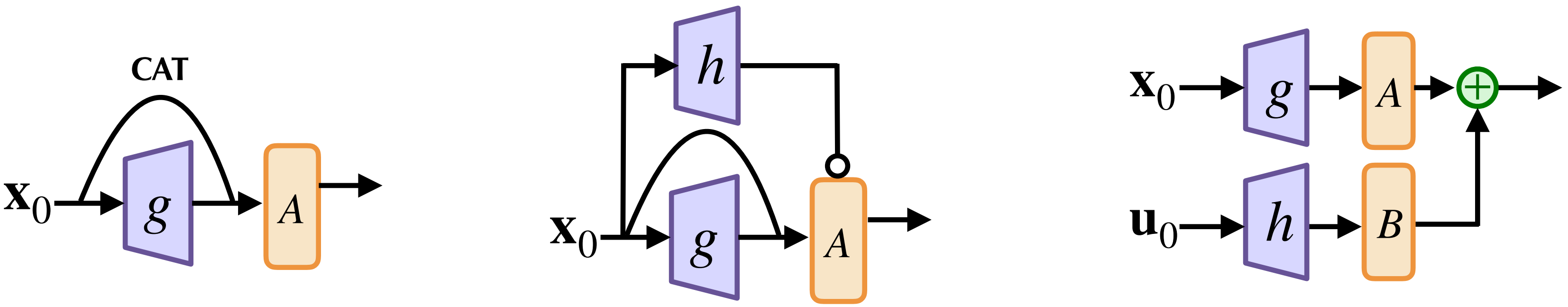
- $h(\mathbf{y}) = h(g(\mathbf{x})) = \tilde{h}(\mathbf{x})$ Hence h can be placed before g.
- For systems with preserved energy, there is *no need* to parameterise the Koopman operator at each time step:

  $A(\mathbf{x}) = A\left(\mathbf{x}_0\right) \quad \forall \mathbf{x} \,|\, \mathscr{H}(\mathbf{x}) = \mathscr{H}\left(\mathbf{x}_0\right)$

DENIS






CAT
x0
g
A
h
g
x0
A
x0
g
A
u0
h
B