Machine learning to efficiently linearize system for control

Initial goal

- System $\dot{\boldsymbol{x}} = F(\boldsymbol{x})$
- Observe $\mathrm{h}(\boldsymbol{x_k})$ k=1:K
- Compute $\dot{\boldsymbol{x}} = W\ \boldsymbol{x}$

Linear system

- Autoencoder
- Measure $(\boldsymbol{x_k}, \boldsymbol{y_k})$
- Architecture $\boldsymbol{y_k} = \varphi^{-1}(K(\varphi(\boldsymbol{x_k}))$
- Difficult in finite dimension,

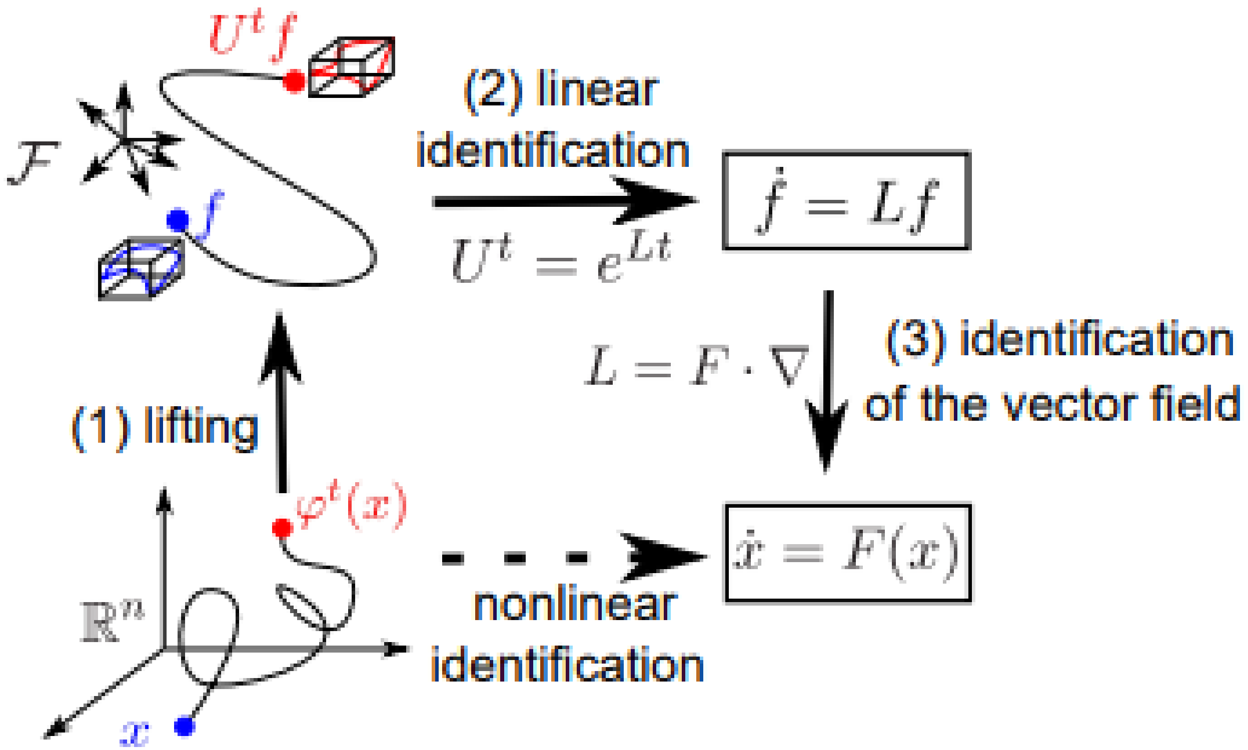
Chaos as an intermittently forced linear system Steven L. Brunton1, Bingni W. Brunton2, Joshua L. Proctor3, Eurika Kaiser1 & J. Nathan Kutz4

"However, we have shown that it is quite rare for a dynamical system to admit a finite-dimensional Koopman-invariant subspace that includes the state variables explicitly, so that exact linear models to propagate the state dynamics exist only for systems with a single isolated fixed point."

New goal

- System $\dot{\boldsymbol{x}} = F(\boldsymbol{x})$
- Observe $\mathrm{h}(\boldsymbol{x}_{\boldsymbol{k}})$ k=1:K
- Compute $\boxed{\dot{\boldsymbol{x}} = W\ \varphi(\boldsymbol{x})}$ linear in the parameters

Framework



- *Koopman-based lifting techniques for nonlinear systems identification* (A. Mauroy and J. Goncalves 2019)

Problem statement

- System $\dot{\boldsymbol{x}} = F(\boldsymbol{x})$

- Assume $\boxed{\boldsymbol{F}(\boldsymbol{x}) = \sum_{k=1}^{nF} \boldsymbol{w_k} h_k(\boldsymbol{x})}$

- Goal : *compute* $\boldsymbol{w_k}$

1) Lifting

- Data : (x_k, y_k) $k = 1{:}K$ $\boxed{\text{vector : } X, Y}$ $size\ (n, K)$
- Basis : $(g_1, g_2, \ldots, g_{nF})$

- Lifted vector $\boxed{f(\boldsymbol{X}) = (g_1(\boldsymbol{X}), g_2(\boldsymbol{X}), \ldots, g_{nF}(\boldsymbol{X}))}$ $size\ (nF, K)$

- Paper basis : monomials (useful for next steps)
- Data sampling period dt ; multiple experience ; noisy

2) Koopman theory

- Find Discrete time Koopman Operator K_{dt}

$$f(\boldsymbol{Y}) = K_{dt}\, f(\boldsymbol{X}) \qquad \Rightarrow K_{dt} = f(\boldsymbol{X})^{\dagger} f(\boldsymbol{Y}) \qquad size\ (nF, nF)$$

- Infinitesimal generator $\quad L = F.\nabla \qquad\qquad K_{dt} = e^{L\,dt}$

-

$$\boxed{L = \frac{\log(f(\boldsymbol{X})^{\dagger} f(\boldsymbol{Y}))}{dt}}$$

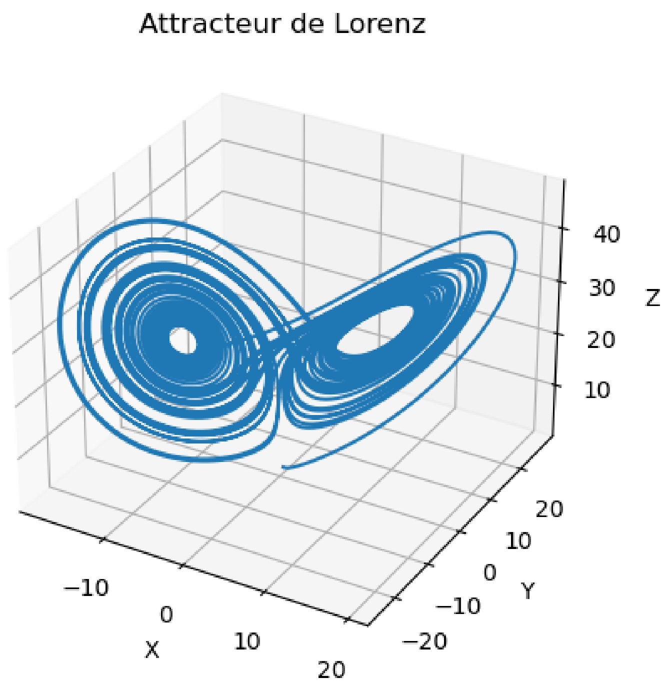
3) Coefficient identification

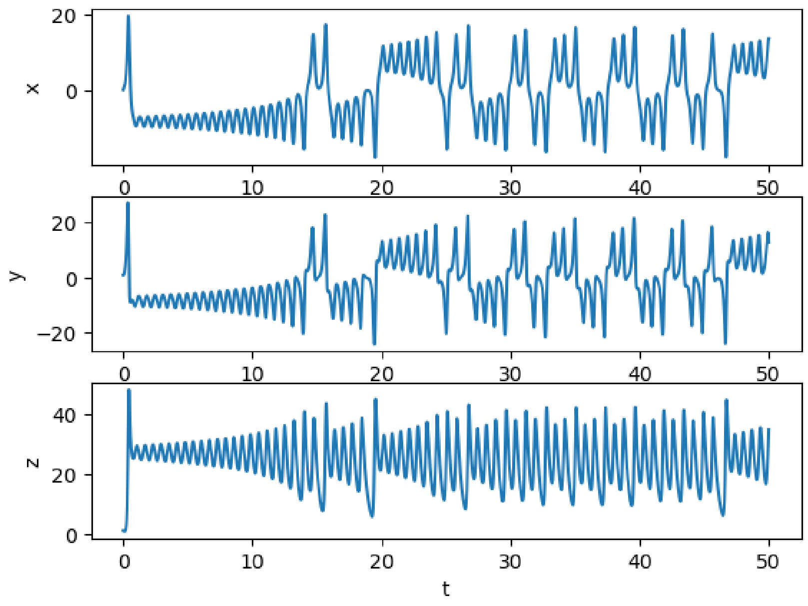
- Equation $f(\dot{\boldsymbol{X}}) = L\, \boldsymbol{f}(\boldsymbol{X})$
- $(\dots ,\ \dots \qquad ,\ x^1\ ,\dots \qquad ,\ \dots)$
- $\begin{pmatrix} \dots \\ \vdots \\ \vdots \\ \dots \end{pmatrix} \begin{bmatrix} L_{11} & \cdots & L_{1i} & \cdots & L_{1nF} \\ \vdots & & \vdots & & \vdots \\ L_{nF1} & \cdots & L_{nFi} & \cdots & L_{nFnF} \end{bmatrix}$
- *i : indice of deg(x) =1* $\boxed{w_k = L_{ki}}$

Lorentz System

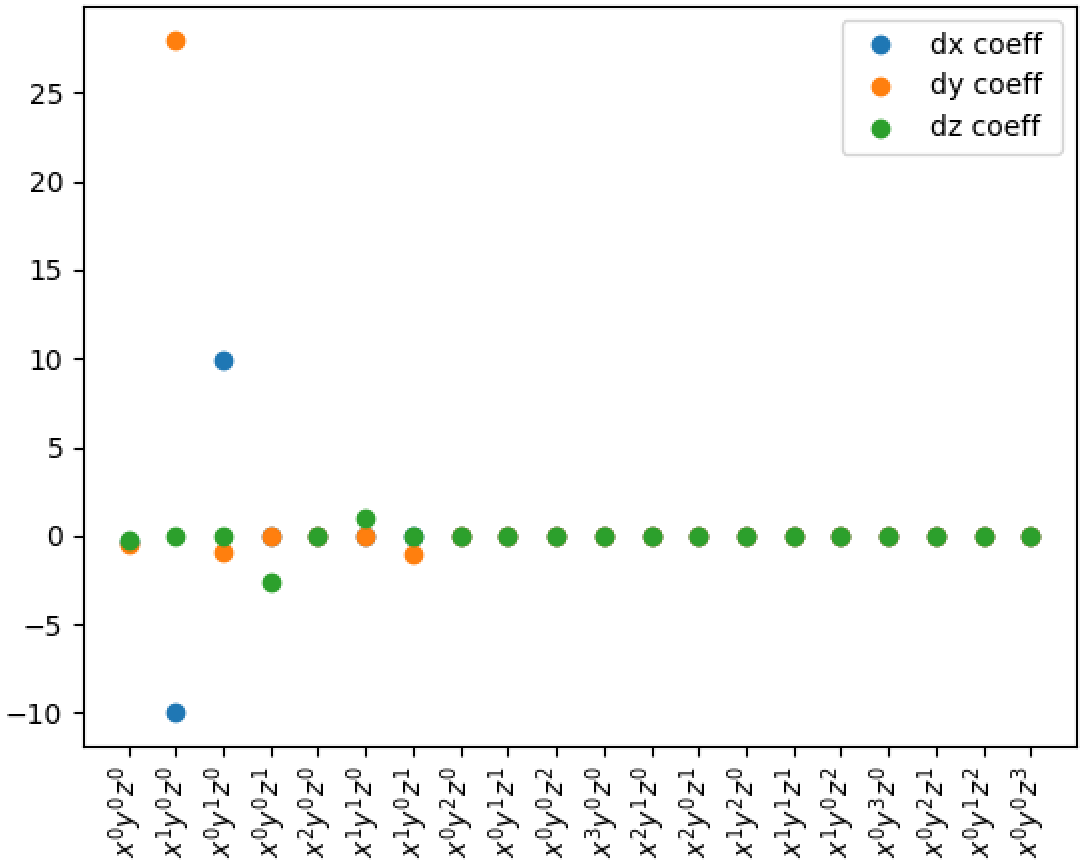
$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = xy - \beta z$$

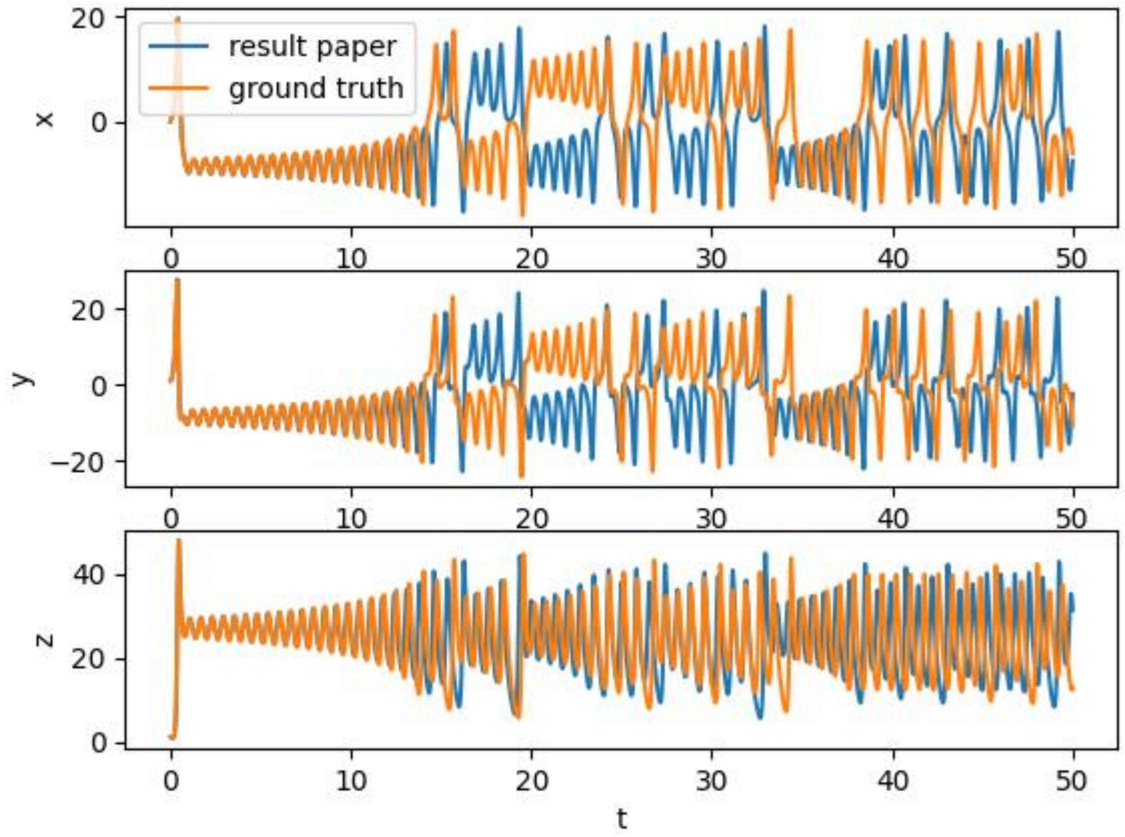
$\sigma = 10\,; \quad \rho = 28; \quad \beta = 8/3$





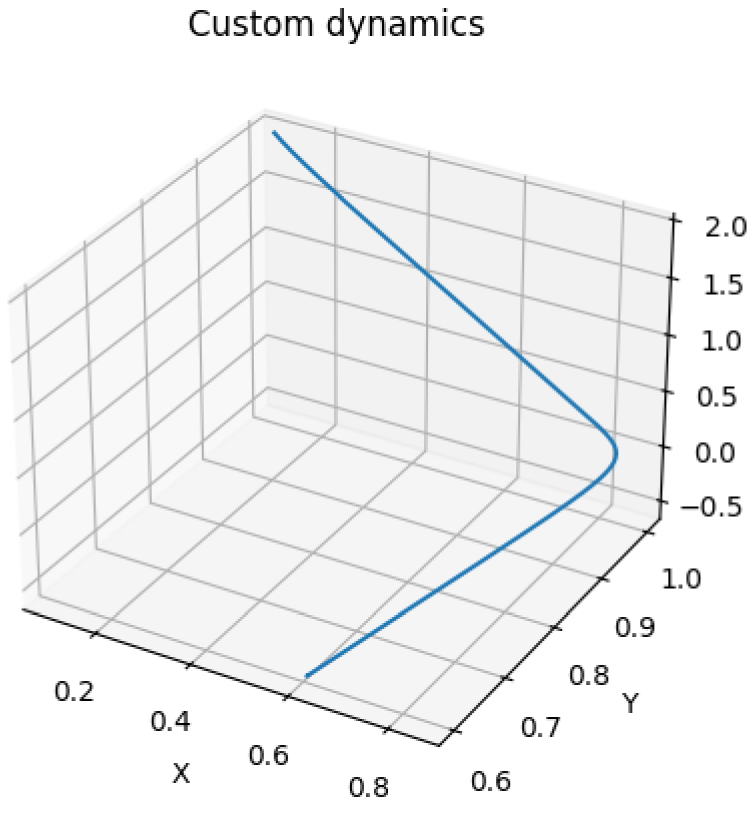
Results Lorentz

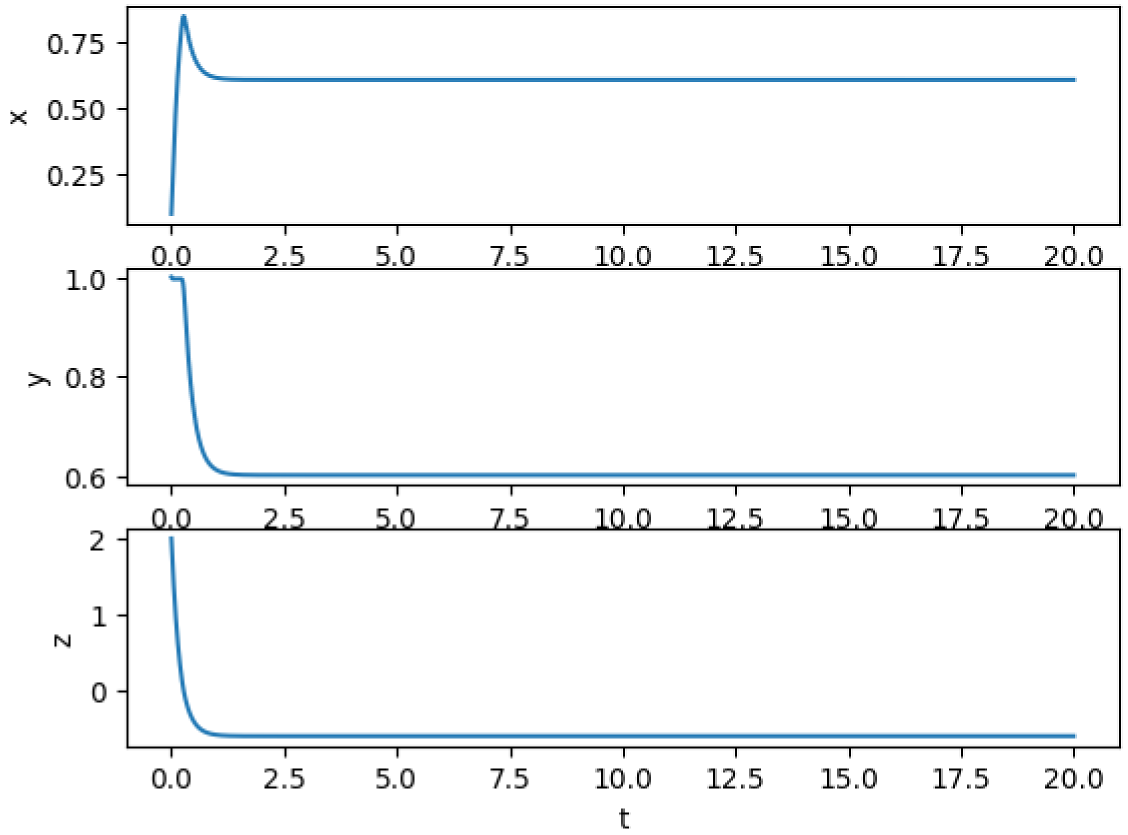




System linear in the parameters

$$\dot{\boldsymbol{x}} = -\lambda\,\boldsymbol{x} + W f(\boldsymbol{x})$$
$$f(x) = \tanh(10\,x)$$





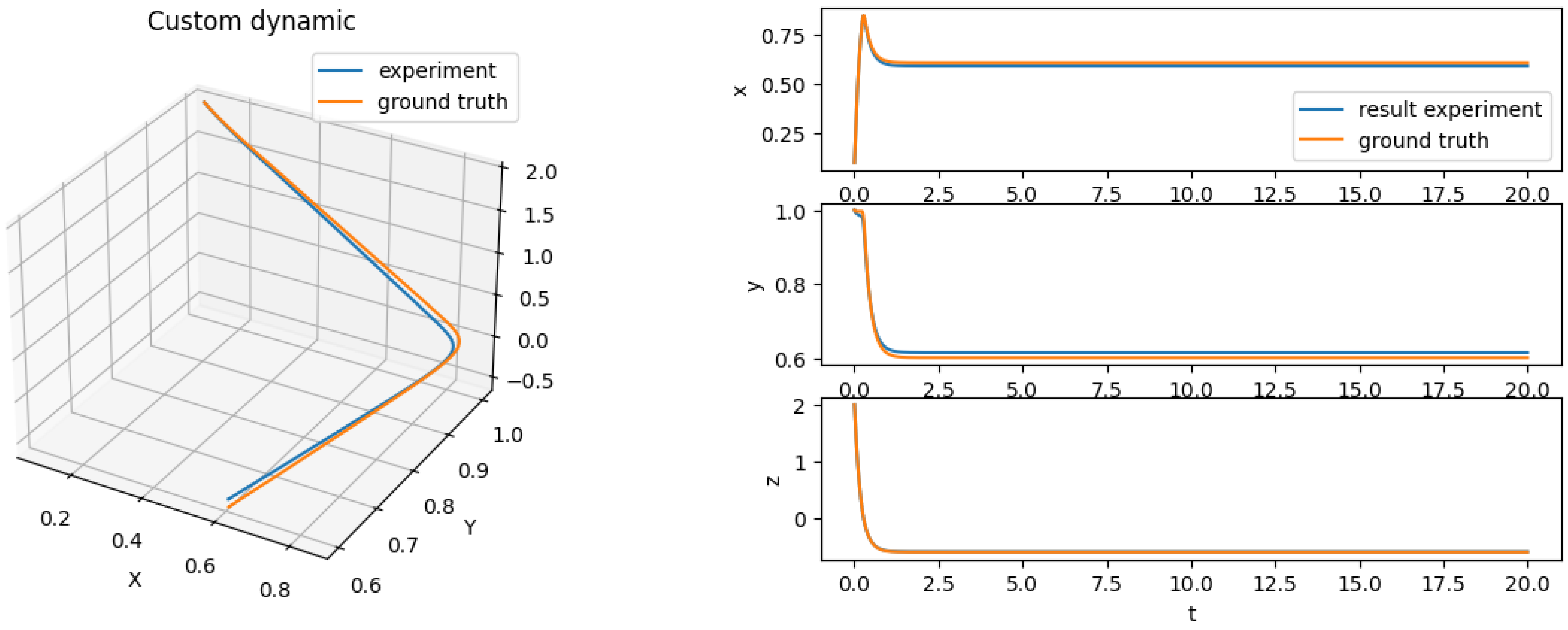
Result linear in the parameters

Lift basis : $(1, x, f(x))$

$$W_{exp} = \begin{matrix} 3.82990212 & 0.64017437 & 1.42006624 \\ 1.54285035 & 2.49518633 & 0.96473717 \\ -2.53540673 & -1.0017014 & -0.23836165 \end{matrix} ; \qquad \lambda = diag(-5.028\,, -4.996, -4.987)$$

$$W_{ground\ truth} = \begin{matrix} 3.75752499 & 0.62070398 & 1.34797359 \\ 1.49608366 & 2.49995341 & 0.9920835 \\ -2.39647322 & -0.92307815 & -0.2574784 \end{matrix} ; \qquad \lambda = diag(-5\,, -5, -5)$$

Result linear in the parameters



f(x) as input

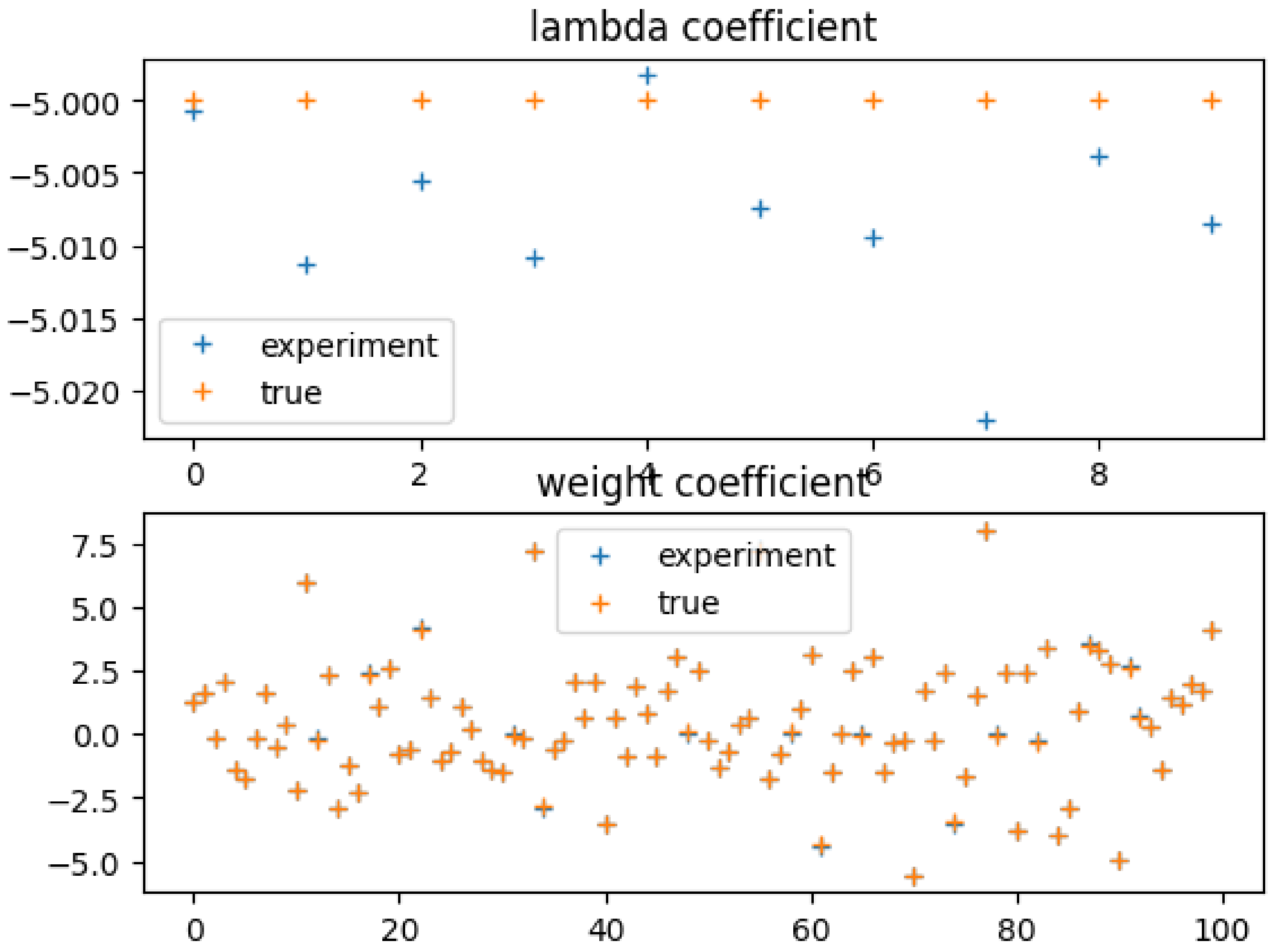
- x as input

- f(x) as input

- Lift basis $\left(1, x, f(x)\right)$
- Data : $\mathrm{X} = \ (1, x, f(x))$

- Lift basis $\left(1, x, f^{-1}(x)\right)$
- Data : $\mathrm{fX} = \ (1, f(x), f^{-1}(f(x)))$
- Permute column $fX_2 = \ (1, x, f(x)) = X$

Results f(x) as input



Partial observation

- Takens's theorem and delay embeddings
- Complete state $x = (x_1, x_2, \dots, x_n)$
- Observe $x = (x_1, x_2, \dots, x_k) \quad k < n$

In progress …

Spiking neural network (SNN)

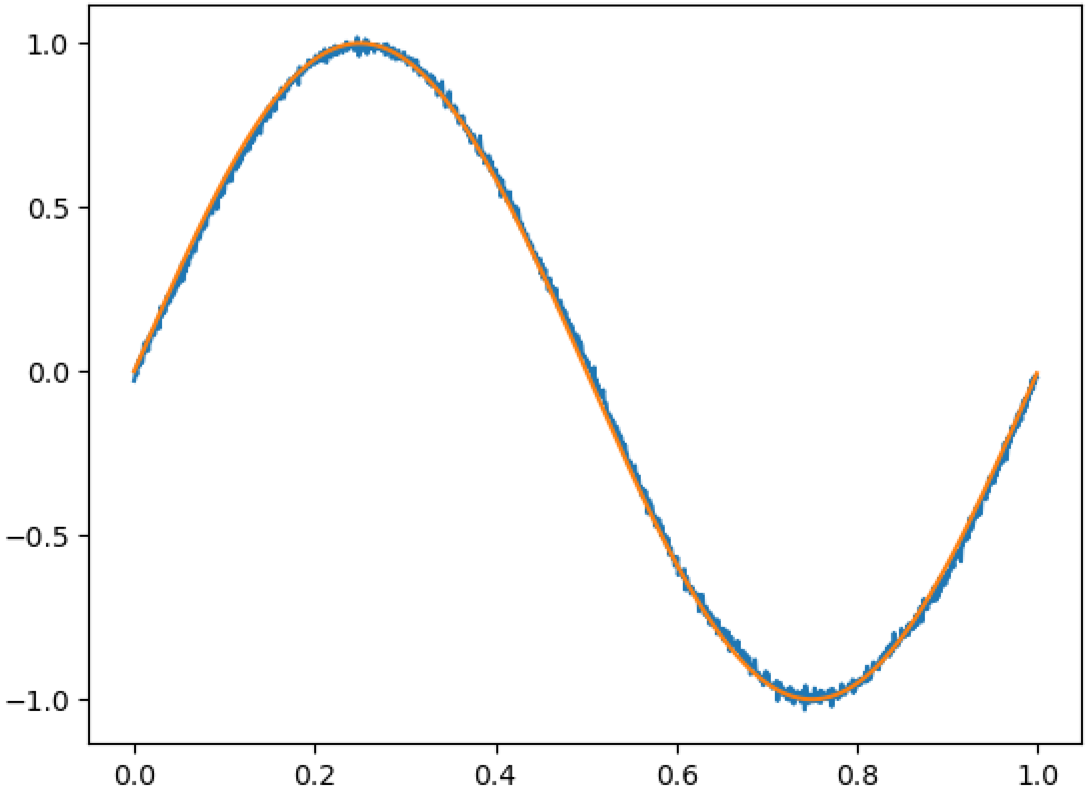
snnTorch

$$U[t+1] = \beta U[t] + WX[t+1]$$

$$\beta = e^{-\frac{\Delta t}{\tau}}$$

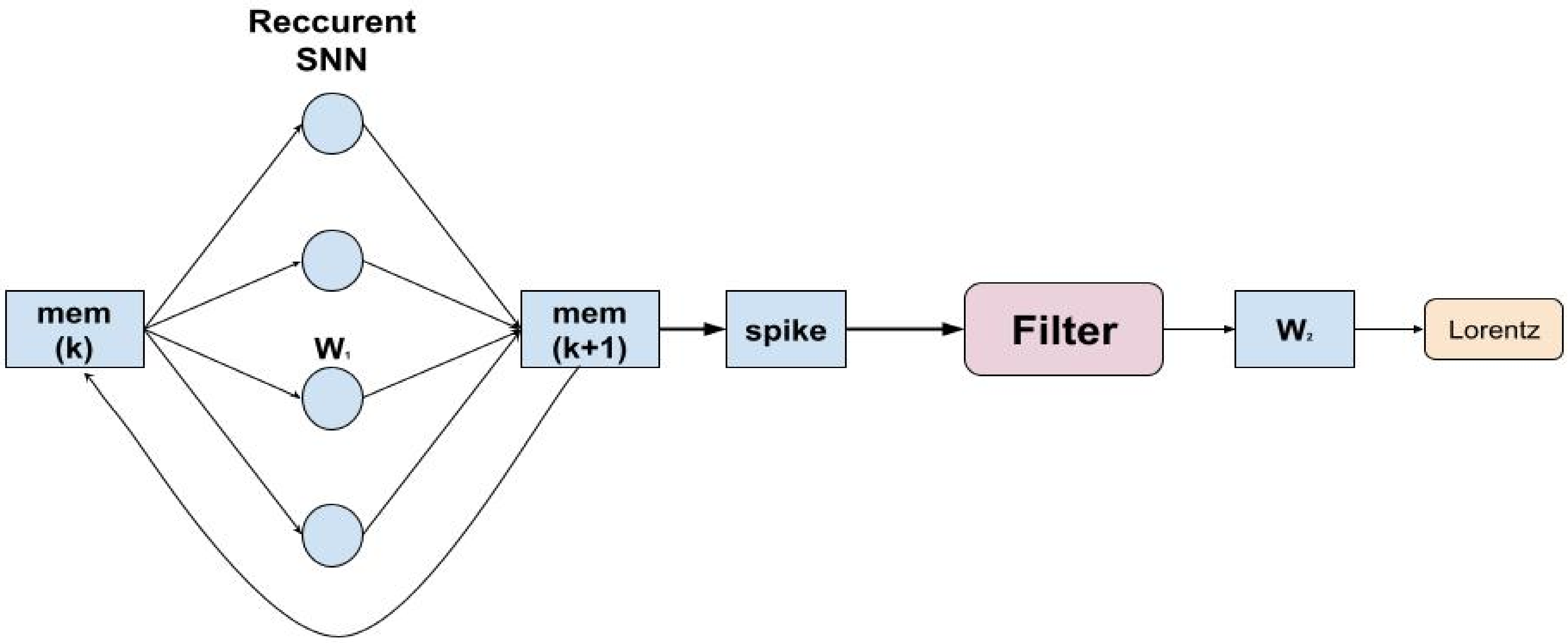
- https://snntorch.readthedocs.io/en/latest/
- *Jason K. Eshraghian, Max Ward, Emre Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu "Training Spiking Neural Networks Using Lessons From Deep Learning". Proceedings of the IEEE, 111(9) September 2023.*

Fit a sinus with SNN



Framework + SNN

- Goal: *find recurrent SNN that generates Lorentz dynamic*



Questions ?