MATH 4280

Lecture Notes 8: Data-driven dynamical systems

Dynamical systems

- We consider dynamical systems of the form

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t), t; \boldsymbol{\beta})$$

where $\mathbf{x}$ is the state, $\mathbf{f}$ is a given vector field and $\boldsymbol{\beta}$ is a set of parameters

- We will also consider simpler case of an autonomous system

$$\frac{d}{dt}\mathbf{x}(t) = \mathbf{f}(\mathbf{x}(t))$$

where the right hand side has no time dependence or parameter

Discrete-time systems

- We also consider discrete-time systems

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

- Discrete-time dynamics may be induced from continuous-time dynamics, where $\mathbf{x}_k = \mathbf{x}(k\Delta t)$ is obtained the sampling in discrete times
- In this case, the discrete-time propagator $\mathbf{F}_{\Delta t}$ is parameterized by the time step Δt.
- For an arbitrary time, we define the flow map $\mathbf{F}_t$ by

$$\mathbf{F}_t(\mathbf{x}(t_0)) = \mathbf{x}(t_0) + \int_{t_0}^{t_0+t} \mathbf{f}(\mathbf{x}(\tau))\, d\tau$$

This expression transition continuous-time systems into discrete time systems

Linear dynamics

- We consider linear dynamics of the form

$$\frac{d}{dt}\mathbf{x} = \mathbf{A}\mathbf{x}$$

- The solution is given by

$$\mathbf{x}(t_0 + t) = e^{\mathbf{A}t}\mathbf{x}(t_0)$$

- The matrix A has the following spectral decomposition

$$\mathbf{A}\mathbf{T} = \mathbf{T}\mathbf{\Lambda}$$

- When the matrix A has distinct eigenvalues, we have $\mathbf{A} = \mathbf{T}\mathbf{\Lambda}\mathbf{T}^{-1}$ and

$$\mathbf{x}(t_0 + t) = \mathbf{T}e^{\mathbf{\Lambda}t}\mathbf{T}^{-1}\mathbf{x}(t_0)$$

where $\mathbf{\Lambda}$ is a diagonal matrix, columns of $\mathbf{T}$ are eigenvectors

Goals

- Future state prediction: e.g. weather forecasting
- Design and optimization: tune the parameters of a system for improved performance
- Estimation and control: actively control a dynamical system through feedback
- Physical understanding: obtain insights of a system by analyzing the solutions

Challenges

- Nonlinearity: the main challenge
- Unknown dynamics: a lot of problems do not have known governing equations
- High dimensional dynamics: difficult to find patterns to uncover intrinsic coordinates which the dominant behavior evolves

Data-driven approaches

- Traditionally, physical systems are analyzed by making ideal approximations, resulting in simple models
- With increasing complex systems, the paradigm is shifting from classical approaches to data-driven methods to discover governing equations
- Determining the correct model is becoming more subjective, and there is a need for automated model discovery technique
- Thus, identifying unknown dynamics and learning intrinsic coordinates are the most pressing goals

Dynamic mode decomposition (DMD)

- DMD is used to identity spatio-temporal coherent structures in high dimensional data
- DMD is based on proper orthogonal decomposition (POD)
- DMD identifies the best linear dynamical system that advances high dimensional measurement forward in time
- DMD is based purely on measurement data

Basic ideas of DMD

- It is a data-driven approach
- First, collect a number of pairs of snapshots as they evolve in time
- We denote the snapshots as $\{(\mathbf{x}(t_k), \mathbf{x}(t_k'))\}_{k=1}^{m}$ where $t_k' = t_k + \Delta t$
- Here Δt is the time step, small enough to capture highest frequencies in the dynamics
- The snapshot is a state of the system sampled at a number of discretized locations

- The snapshots can be arranged into data matrices

reshape a snapshot into a 1D array

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}(t_1) & \mathbf{x}(t_2) & \cdots & \mathbf{x}(t_m) \\ | & | & & | \end{bmatrix}$$

$$\mathbf{X}' = \begin{bmatrix} | & | & & | \\ \mathbf{x}(t'_1) & \mathbf{x}(t'_2) & \cdots & \mathbf{x}(t'_m) \\ | & | & & | \end{bmatrix}$$

- Note, if we assume uniform sampling in time $t_k = k\Delta t$ and $\boxed{t'_k = t_k + \Delta t = t_{k+1}\ ,}$ we use the notation $\mathbf{x}_k = \mathbf{x}(k\Delta t)$

shifting the data entries by one

- The DMD algorithm seeks the best $\boxed{\text{linear operator } \mathbf{A}}$ that relates the two snapshot matrices in time

$$\mathbf{X}' \approx \mathbf{A}\mathbf{X}$$

- This best fit linear operator gives a linear dynamical system that advances snapshot measurements in time
- If we assume uniform sampling in time

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

- Mathematically, the linear operator is defined as

$$\mathbf{A} = \underset{\mathbf{A}}{\operatorname{argmin}} \|\mathbf{X}' - \mathbf{A}\mathbf{X}\|_F = \mathbf{X}'\mathbf{X}^\dagger$$

This is the .fit() step in the python script

- Note that $\mathbf{A}$ and its eigenvectors are expensive to compute due to large dimension

DMD algorithm

- Assume $\mathbf{X}$ is $m \times n,\ m \ll n$
- Step 1: Compute the SVD of $\mathbf{X}$

$$\mathbf{X} \approx \tilde{\mathbf{U}}\tilde{\mathbf{\Sigma}}\tilde{\mathbf{V}}^*$$

  where $\tilde{\mathbf{U}} \in \mathbb{C}^{n\times r},\quad \tilde{\mathbf{\Sigma}} \in \mathbb{C}^{r\times r}$ and $\tilde{\mathbf{V}} \in \mathbb{C}^{m\times r}$
- Here $r \leq m$ is the exact or approximate rank of the data matrix $\mathbf{X}$
- Recall that the columns of $\tilde{\mathbf{U}}$ are called POD modes

(= PCA modes)

- Step 2: Recall that the full matrix $\mathbf{A}$ is computed as

$$\mathbf{A} = \mathbf{X}'\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}\tilde{\mathbf{U}}^*$$

- Note that we are interested in the leading r eigenvectors
- We project $\mathbf{A}$ onto the POD modes (the space that spanned by U \tilde)

$$\tilde{\mathbf{A}} = \tilde{\mathbf{U}}^*\mathbf{A}\tilde{\mathbf{U}} = \tilde{\mathbf{U}}^*\mathbf{X}'\tilde{\mathbf{V}}\tilde{\mathbf{\Sigma}}^{-1}$$

- Note that the reduced matrix $\tilde{\mathbf{A}}$ has the same nonzero eigenvalues as the full matrix $\mathbf{A}$
- The reduced matrix $\tilde{\mathbf{A}}$ defines a linear dynamic for POD modes

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

x_{k+1} = Ax_k , \tilde u^* x_{k+1} = \tilde u^* A x_k ; \tilde x_{+1} = \tilde A \silde x_k

- The full state can be recovered by $\mathbf{x} = \tilde{\mathbf{U}}\tilde{\mathbf{x}}$

W is eigenvector matrix of \tilde A. And columns of W are eigenvectors

- Step 3: Perform the spectral decomposition of $\tilde{\mathbf{A}}$

$$\tilde{\mathbf{A}}\mathbf{W} = \mathbf{W}\boldsymbol{\Lambda}$$

- Note that the diagonal matrix $\boldsymbol{\Lambda}$ contains the DMD eigenvalues, which are also eigenvalues of the full matrix $\mathbf{A}$
- The columns of $\mathbf{W}$ are eigenvectors, which are linear combination of POD modes amplitudes
- Step 4: The high dimensional DMD modes $\boldsymbol{\Phi}$ are reconstructed by

$$\boldsymbol{\Phi} = \mathbf{X}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1}\mathbf{W}$$

(Which has no explanation ....)

- Note that

$$\begin{aligned}\mathbf{A}\boldsymbol{\Phi} &= (\mathbf{X}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1}\underbrace{\tilde{\mathbf{U}}^{*})(\mathbf{X}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1}}_{\tilde{\mathbf{A}}}\mathbf{W}) \\ &= \mathbf{X}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1}\tilde{\mathbf{A}}\mathbf{W} \\ &= \mathbf{X}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1}\mathbf{W}\boldsymbol{\Lambda} \\ &= \boldsymbol{\Phi}\boldsymbol{\Lambda}.\end{aligned}$$

DMD modes are eigenvectors

DMD expansion

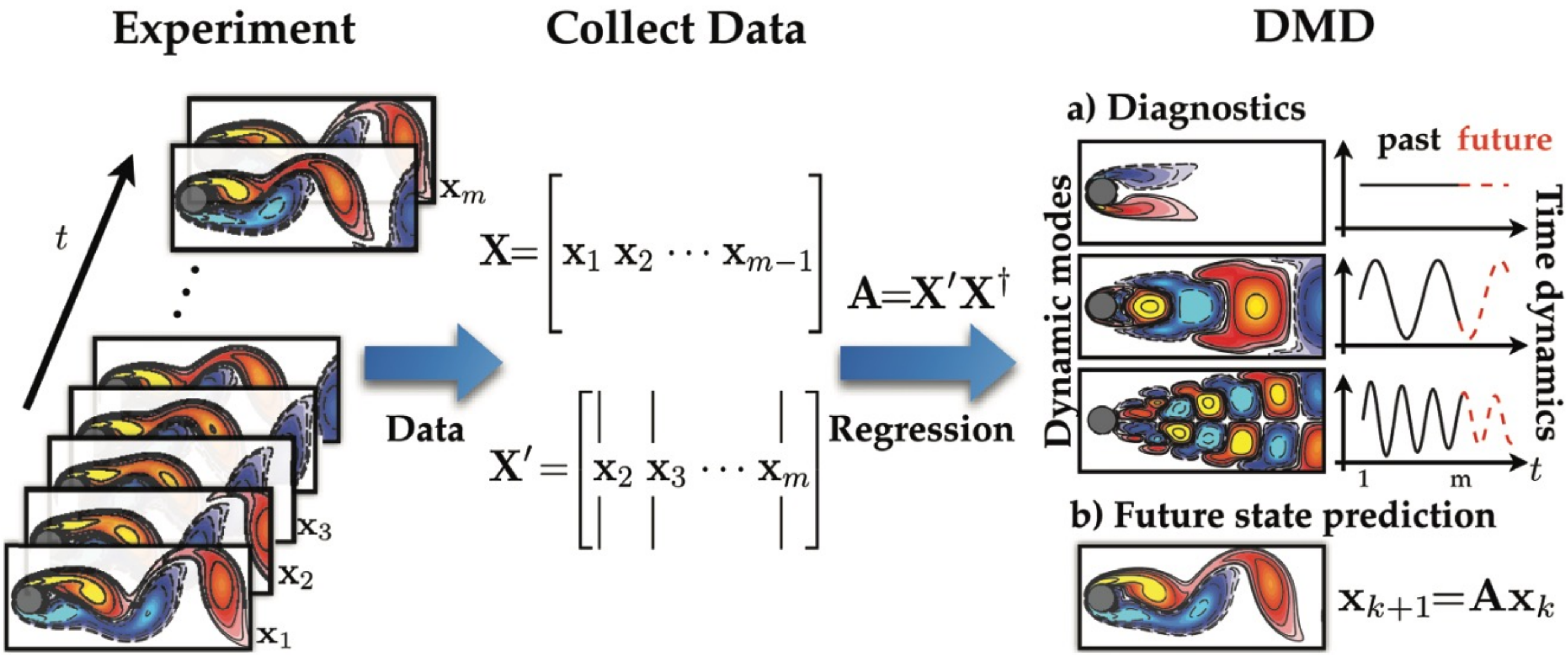
- We can use DMD modes to represent the state of the system

$$\boxed{\mathbf{x}_k = \sum_{j=1}^{r} \boldsymbol{\phi}_j \lambda_j^{k-1} b_j = \boldsymbol{\Phi}\boldsymbol{\Lambda}^{k-1}\mathbf{b}}$$

- The vector $\mathbf{b}$ is computed by $\mathbf{b} = \boldsymbol{\Phi}^{\dagger}\mathbf{x}_1$
- A more convenient way to compute $\mathbf{b}$

$$\begin{aligned}
& \mathbf{x}_1 = \boldsymbol{\Phi}\mathbf{b} \\
\Longrightarrow \quad & \tilde{\mathbf{U}}\tilde{\mathbf{x}}_1 = \mathbf{X}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1}\mathbf{W}\mathbf{b} \\
\Longrightarrow \quad & \tilde{\mathbf{x}}_1 = \tilde{\mathbf{U}}^*\mathbf{X}'\tilde{\mathbf{V}}\tilde{\boldsymbol{\Sigma}}^{-1}\mathbf{W}\mathbf{b} \\
\Longrightarrow \quad & \tilde{\mathbf{x}}_1 = \tilde{\mathbf{A}}\mathbf{W}\mathbf{b} \\
\Longrightarrow \quad & \tilde{\mathbf{x}}_1 = \mathbf{W}\boldsymbol{\Lambda}\mathbf{b} \\
\Longrightarrow \quad & \mathbf{b} = (\mathbf{W}\boldsymbol{\Lambda})^{-1}\tilde{\mathbf{x}}_1.
\end{aligned}$$

Overview of using DMD algorithm



Applications: fluid dynamics, neuroscience, video processing, etc

Sparse identification of nonlinear dynamics

- Discovering dynamical systems from data is a central challenge
- Sparse identification of nonlinear dynamics (SIND) considers

$$\frac{d}{dt}\mathbf{x} = \mathbf{f}(\mathbf{x})$$

- We seek the following approximation

$$\mathbf{f}(\mathbf{x}) \approx \sum_{k=1}^{p} \theta_k(\mathbf{x})\xi_k = \mathbf{\Theta}(\mathbf{x})\boldsymbol{\xi}$$

with the fewest nonzero terms in $\boldsymbol{\xi}$

- We collect the following data for the states and their derivatives

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}(t_1) & \mathbf{x}(t_2) & \cdots \mathbf{x}(t_m) \end{bmatrix}^T$$

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{\mathbf{x}}(t_1) & \dot{\mathbf{x}}(t_2) & \cdots \dot{\mathbf{x}}(t_m) \end{bmatrix}^T$$

- A library of candidate functions is used

$$\mathbf{\Theta}(\mathbf{X}) = \begin{bmatrix} \mathbf{1} & \mathbf{X} & \mathbf{X}^2 & \cdots & \mathbf{X}^d & \cdots & \sin(\mathbf{X}) & \cdots \end{bmatrix}$$

- The dynamical system is represented using data matrices

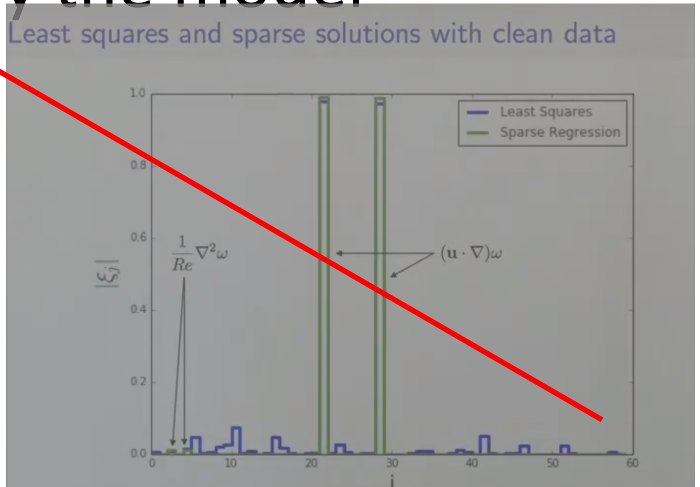
$$\dot{\mathbf{X}} = \mathbf{\Theta}(\mathbf{X})\mathbf{\Xi}.$$

- The following optimization problem is used to identify the model

$$\boxed{\boldsymbol{\xi}_k = \mathrm{argmin}_{\boldsymbol{\xi}'_k} \|\dot{\mathbf{X}}_k - \mathbf{\Theta}(\mathbf{X})\boldsymbol{\xi}'_k\|_2 + \lambda\|\boldsymbol{\xi}'_k\|_1}$$

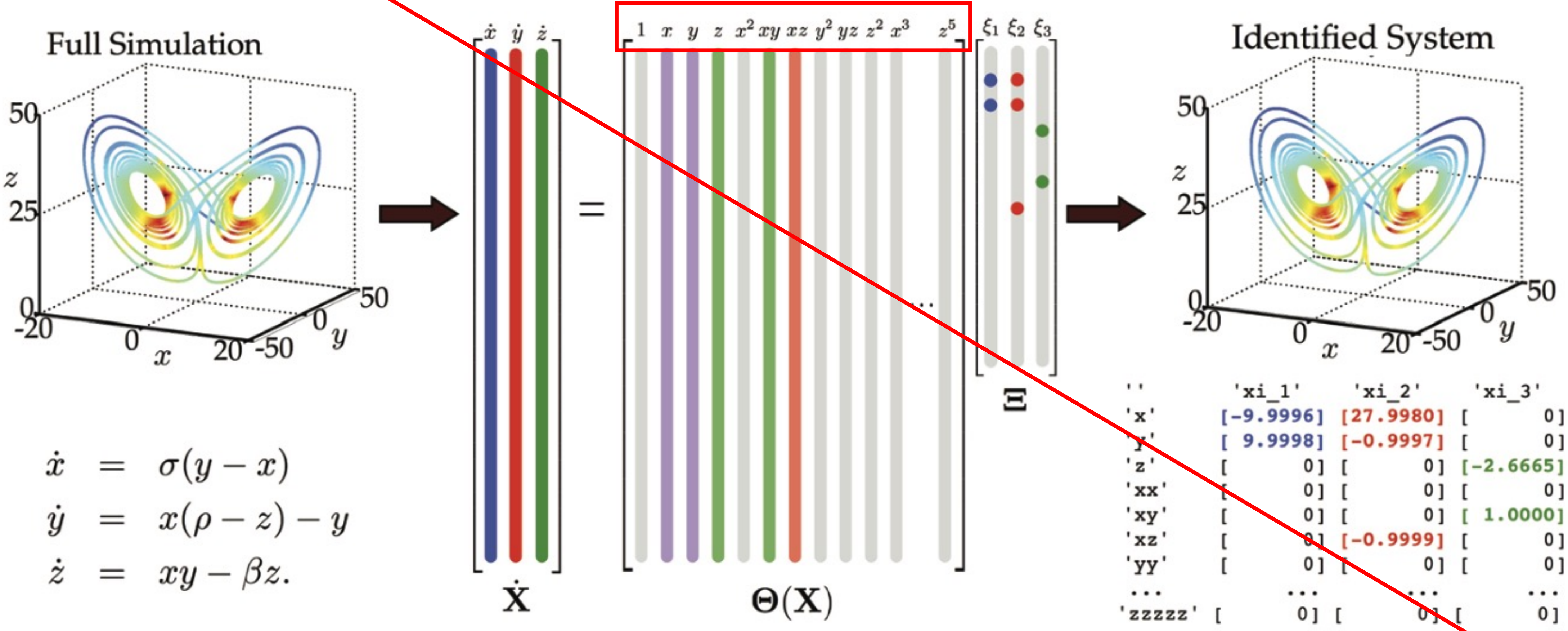
Bad approach: least squares - It is bad because \xi is almost surely dense.
Better approach: sparse regression

Green column: generated by sparse regression



Schematic of SIND

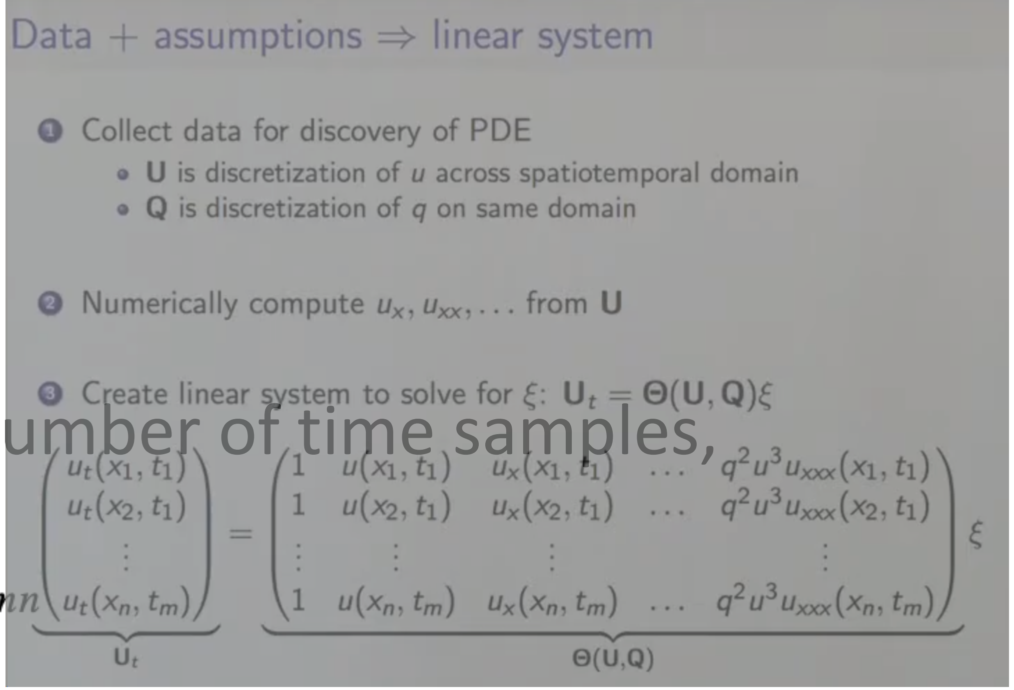
Kernel?



$$
\begin{aligned}
\dot{x} &= \sigma(y - x) \\
\dot{y} &= x(\rho - z) - y \\
\dot{z} &= xy - \beta z.
\end{aligned}
$$

''	'xi_1'	'xi_2'	'xi_3'
'x'	[-9.9996]	[27.9980]	[0]
'y'	[9.9998]	[-0.9997]	[0]
'z'	[0]	[0]	[-2.6665]
'xx'	[0]	[0]	[0]
'xy'	[0]	[0]	[1.0000]
'xz'	[0]	[-0.9999]	[0]
'yy'	[0]	[0]	[0]
...	...	...	...
'zzzzz'	[0]	[0]	[0]

Discovering PDE



Data: if the data is not clean, we need to smoothen it.

- Let $\Upsilon \in \mathbb{C}^{mn}$ be space-time data, m is the number of time samples, and n is the number of spatial locations
- Additional input can be included in $\mathbf{Q} \in \mathbb{C}^{mn}$
- Next, a library $\boldsymbol{\Theta}(\Upsilon, \mathbf{Q}) \in \mathbb{C}^{\boxed{mn \times D}}$ with D candidates of linear and nonlinear terms and partial derivatives is used, and it has the form

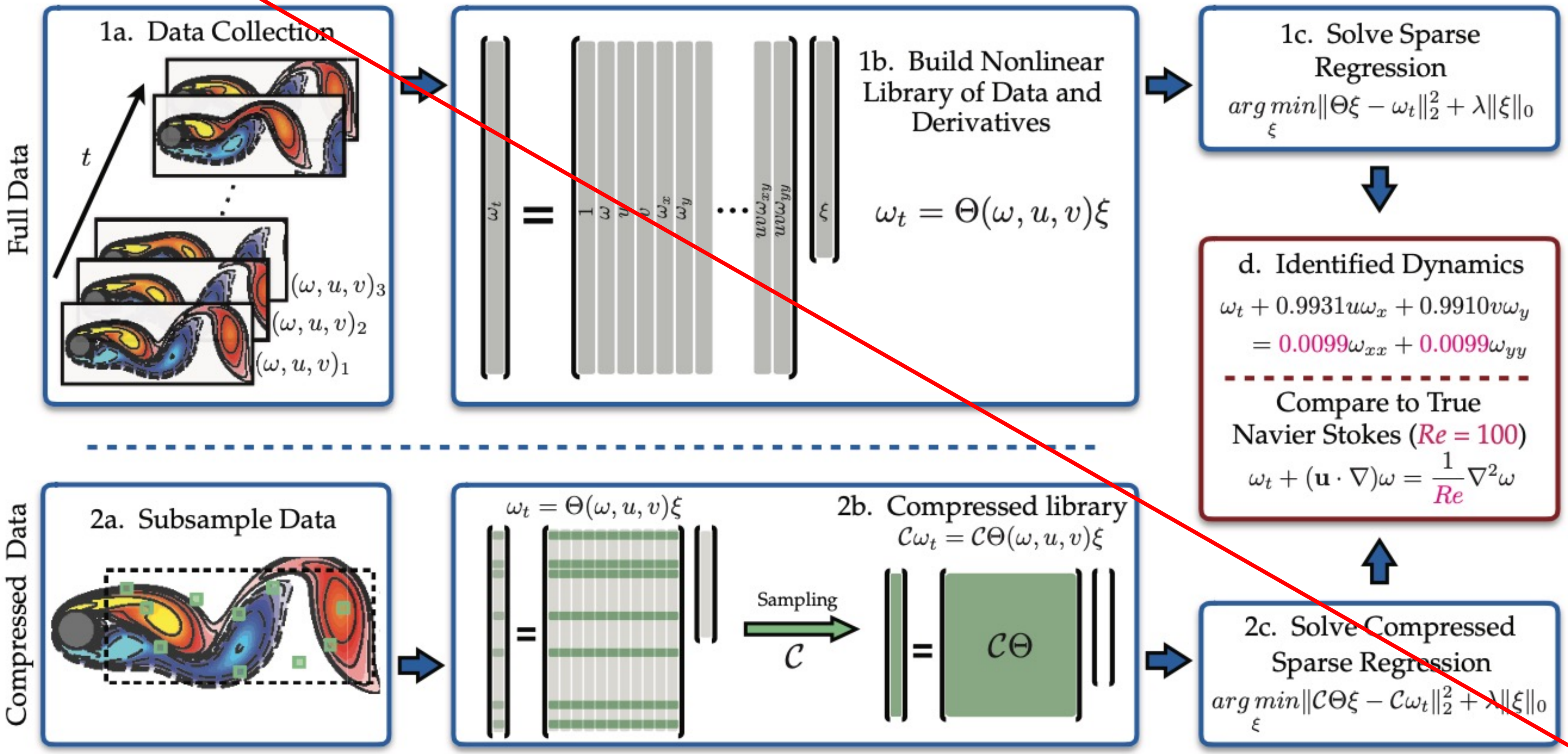
$$\boldsymbol{\Theta}(\Upsilon, \mathbf{Q}) = \left[\boxed{1 \quad \Upsilon \quad \Upsilon^2 \quad \ldots} \; \boxed{\mathbf{Q} \quad \ldots} \; \boxed{\Upsilon_x \quad \Upsilon\Upsilon_x \quad \ldots}\right]$$

- The PDE can be represented in this library by

$$\Upsilon_t = \boldsymbol{\Theta}(\Upsilon, \mathbf{Q})\boldsymbol{\xi}$$

- Then, we find the sparse vector $\boldsymbol{\xi}$

Schematic



Full Data
1a. Data Collection
t
$(\omega, u, v)_3$
$(\omega, u, v)_2$
$(\omega, u, v)_1$
1b. Build Nonlinear Library of Data and Derivatives
$\omega_t = \Theta(\omega, u, v)\xi$
1c. Solve Sparse Regression
$\arg\min_\xi \|\Theta\xi - \omega_t\|_2^2 + \lambda\|\xi\|_0$
d. Identified Dynamics
$\omega_t + 0.9931 u\omega_x + 0.9910 v\omega_y = 0.0099\omega_{xx} + 0.0099\omega_{yy}$
Compare to True Navier Stokes ($Re = 100$)
$\omega_t + (\mathbf{u}\cdot\nabla)\omega = \frac{1}{Re}\nabla^2\omega$
Compressed Data
2a. Subsample Data
$\omega_t = \Theta(\omega, u, v)\xi$
2b. Compressed library
$\mathcal{C}\omega_t = \mathcal{C}\Theta(\omega, u, v)\xi$
Sampling
$\mathcal{C}$
$\mathcal{C}\Theta$
2c. Solve Compressed Sparse Regression
$\arg\min_\xi \|\mathcal{C}\Theta\xi - \mathcal{C}\omega_t\|_2^2 + \lambda\|\xi\|_0$

Koopman operator

- Represent a nonlinear dynamical system in terms of an infinite dimensional linear operator
- This Koopman operator is linear, and its spectral decomposition completely characterizes the nonlinear dynamics
- Obtaining finite dimensional approximation of the Koopman operator is the focus of research

Linear matrix that advances measurements forward in time

Mathematical formulation

composition of function

g could be any measurements of x in this Herbert space

- We let $g: M \to \mathbb{R}$ be a real-valued measurement function
- The Koopman operator $\mathcal{K}_t$ is an infinite dimensional linear operator acting on measurement functions defined by

$$\mathcal{K}_t g = g \circ \mathbf{F}_t$$

(F_t we considered before)

- For discrete time system, we have

from previous properties

$$\mathcal{K}_{\Delta t} g(\mathbf{x}_k) = g(\boxed{\mathbf{F}_{\Delta t}(\mathbf{x}_k)}) = g(\mathbf{x}_{k+1})$$

we could have many choices of g

- Thus, the $\boxed{\text{Koopman operator advances the observation of a state}}$

$$g(\mathbf{x}_{k+1}) = \mathcal{K}_{\Delta t} g(\mathbf{x}_k)$$

- For smooth continuous time dynamical system, we can define the following Koopman dynamical system

$$\frac{d}{dt}g = \mathcal{K}g.$$

- The operator $\mathcal{K}$ is defined as

$$\mathcal{K}g = \lim_{t\to 0} \frac{\mathcal{K}_t g - g}{t} = \lim_{t\to 0} \frac{g \circ \mathbf{F}_t - g}{t}$$

Koopman eigenfunctions

- Identify key measurement functions, that evolves linearly
- Discrete-time Koopman eigenfunctions are given by

$$\varphi(\mathbf{x}_{k+1}) = \mathcal{K}_{\Delta t}\varphi(\mathbf{x}_k) = \lambda\varphi(\mathbf{x}_k)$$

- Continuous-time Koopman eigenfunctions are given by

$$\frac{\mathrm{d}}{\mathrm{d}t}\varphi(\mathbf{x}) = \mathcal{K}\varphi(\mathbf{x}) = \lambda\varphi(\mathbf{x})$$

- Taking derivatives, we have

$$\frac{\mathrm{d}}{\mathrm{d}t}\varphi(\mathbf{x}) = \nabla\varphi(\mathbf{x}) \cdot \dot{\mathbf{x}} = \nabla\varphi(\mathbf{x}) \cdot \mathbf{f}(\mathbf{x})$$

- So, we obtain a PDE for the eigenfunctions

$$\nabla\varphi(\mathbf{x}) \cdot \mathbf{f}(\mathbf{x}) = \lambda\varphi(\mathbf{x})$$

page 32 discusses how to obtain eigenfunction

Mathematical formulation

g could be any measurements of x in this Herbert space

composition of function

- We let $g: M \to \mathbb{R}$ be a real-valued measurement function
- The Koopman operator $\mathcal{K}_t$ is an infinite dimensional linear operator acting on measurement functions defined by

$$\mathcal{K}_t g = g \circ \mathbf{F}_t$$ (F_t we considered before)

- For discrete time system, we have

from previous properties

$$\mathcal{K}_{\Delta t} g(\mathbf{x}_k) = g(\mathbf{F}_{\Delta t}(\mathbf{x}_k)) = g(\mathbf{x}_{k+1})$$ we could have many choices of g

- Thus, the Koopman operator advances the observation of a state

$$g(\mathbf{x}_{k+1}) = \mathcal{K}_{\Delta t} g(\mathbf{x}_k)$$

- Any conserved quantity is a Koopman eigenfunction corresponding to the eigenvalue $\lambda = 0$
- The constant function $\varphi = 1$ is the trivial eigenfunction
- Product of two eigenfunctions is still an eigenfunction:

$$\begin{aligned}
\mathcal{K}(\varphi_1\varphi_2) &= \frac{d}{dt}(\varphi_1\varphi_2) \\
&= \dot{\varphi}_1\varphi_2 + \varphi_1\dot{\varphi}_2 \\
&= \lambda_1\varphi_1\varphi_2 + \lambda_2\varphi_1\varphi_2 \\
&= (\lambda_1 + \lambda_2)\varphi_1\varphi_2.
\end{aligned}$$

- The continuous and discrete time eigenvalues are related: if λ is a continuous time eigenvalue, then $e^{\lambda t}$ is the corresponding discrete time eigenvalue:

$$\lim_{t\to 0} \frac{\mathcal{K}_t\varphi(\mathbf{x}) - \varphi(\mathbf{x})}{t} = \lim_{t\to 0} \frac{e^{\lambda t}\varphi(\mathbf{x}) - \varphi(\mathbf{x})}{t} = \lambda\varphi(\mathbf{x})$$

We dont have eigenfunction(s) in the beginning

Koopman mode decomposition

- Usually, we like to take multiple measurements of a system
- We denote these measurements as a vector

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \vdots \\ g_p(\mathbf{x}) \end{bmatrix}$$

- Each individual measurement can be expanded in terms of eigenfunctions

$$\boxed{g_i(\mathbf{x}) = \sum_{j=1}^{\infty} v_{ij}\varphi_j(\mathbf{x})} \qquad \mathbf{g}(\mathbf{x}) = \begin{bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \vdots \\ g_p(\mathbf{x}) \end{bmatrix} = \sum_{j=1}^{\infty} \varphi_j(\mathbf{x})\mathbf{v}_j$$

where $\mathbf{v}_j$ is the j-th Koopman mode associated with φ_j

- Using the above decomposition, we can represent the dynamics of the measurement function as follows

$$\mathbf{g}(\mathbf{x}_k) = \mathcal{K}_{\Delta t}^k \mathbf{g}(\mathbf{x}_0) = \mathcal{K}_{\Delta t}^k \sum_{j=0}^{\infty} \varphi_j(\mathbf{x}_0)\mathbf{v}_j$$

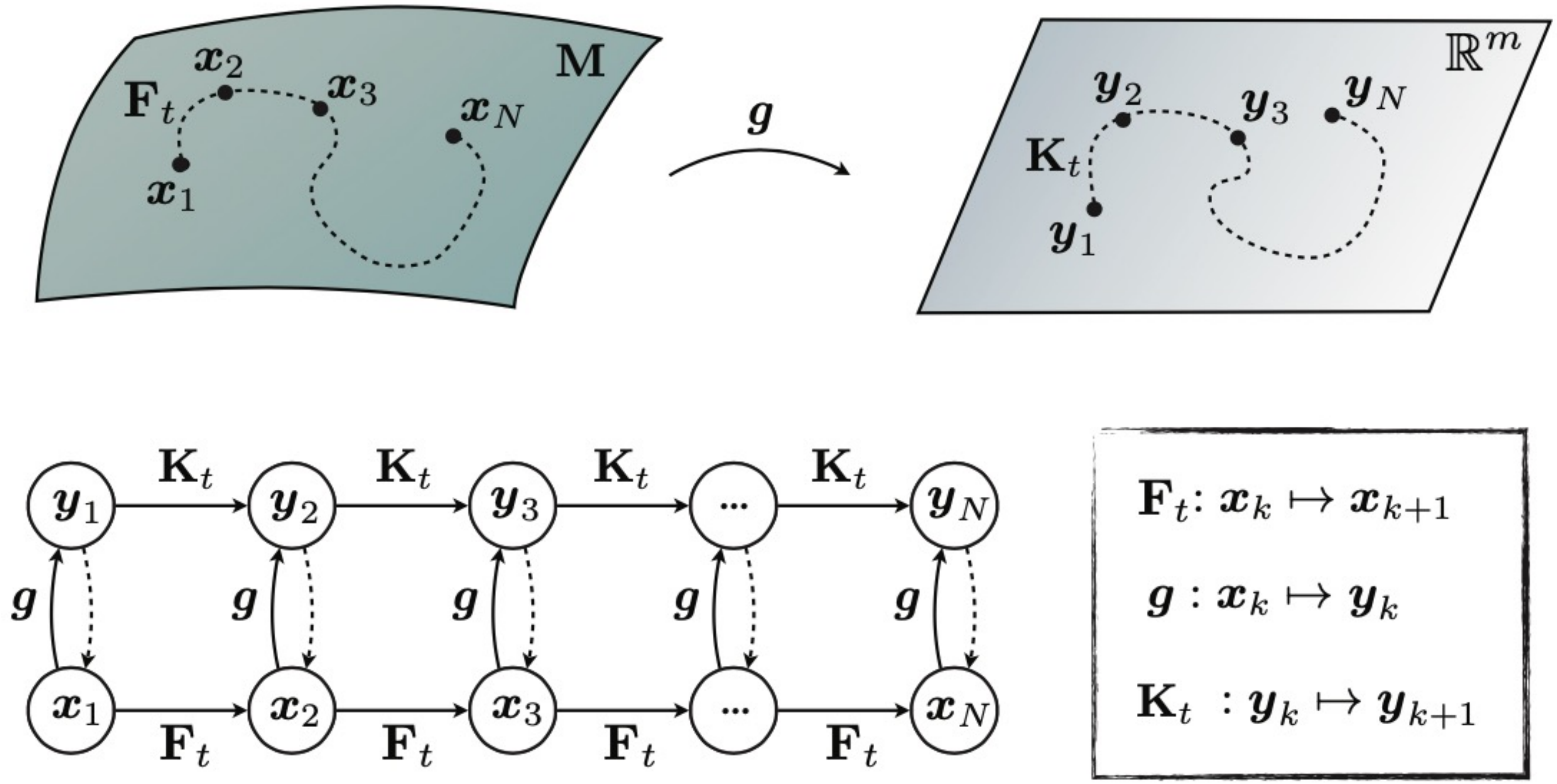
$$= \sum_{j=0}^{\infty} \mathcal{K}_{\Delta t}^k \varphi_j(\mathbf{x}_0)\mathbf{v}_j$$

$$= \sum_{j=0}^{\infty} \lambda_j^k \varphi_j(\mathbf{x}_0)\mathbf{v}_j$$

- The triple $\{(\lambda_j, \varphi_j, \mathbf{v}_j)\}_{j=0}^{\infty}$ is called the Koopman mode decomposition

Koopman mode
eigenvalues
eigenvectors
eigenfunctions

- Schematic for Koopman operator



(invariant: after mapping, you are still within the subspace)

Invariant eigenspace

- A Koopman invariant subspace is the span of functions $\{g_1, g_2, \cdots, g_p\}$ if all functions g in this space

$$g = \alpha_1 g_1 + \alpha_2 g_2 + \cdots + \alpha_p g_p$$

remain in the space after applied the Koopman operator

$$\mathcal{K}g = \beta_1 g_1 + \beta_2 g_2 + \cdots + \beta_p g_p$$

- Find a finite dimensional matrix representation of the Koopman operator restricted to the invariant subspace
- This gives finite dimensional systems

$$g(\mathbf{x}_{k+1}) = \mathcal{K}_{\Delta t} g(\mathbf{x}_k) \qquad \frac{d}{dt} g = \mathcal{K} g.$$

A simple example

- Consider the following nonlinear dynamical system

$$\dot{x}_1 = \mu x_1$$
$$\dot{x}_2 = \lambda(x_2 - x_1^2)$$

- We introduce the nonlinear measurement $g = x_1^2$

$$\frac{d}{dt}\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} \mu & 0 & 0 \\ 0 & \lambda & -\lambda \\ 0 & 0 & 2\mu \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \quad \text{for} \quad \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \end{bmatrix}$$

Examples of Koopman eigenfunctions

- Consider the linear dynamics

$$\frac{d}{dt}x = x$$

- Using Taylor's expansion

$$\varphi(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3 + \cdots$$

$$\nabla\varphi = c_1 + 2c_2 x + 3c_3 x^2 + 4c_4 x^3 + \cdots$$

$$\nabla\varphi \cdot f = c_1 x + 2c_2 x^2 + 3c_3 x^3 + 4c_4 x^4 + \cdots$$

- We use the equation $\nabla\varphi(\mathbf{x}) \cdot \mathbf{f}(\mathbf{x}) = \lambda\varphi(\mathbf{x})$
- We observe that $c_0 = 0$
- For any positive integer $\lambda = \mathrm{k}$, $\varphi(x) = cx^k$ is an eigenfunction

???

- Consider the nonlinear dynamical system

$$\frac{d}{dt} = x^2$$

- Assume the Laurent series

$$\varphi(x) = \cdots + c_{-3}x^{-3} + c_{-2}x^{-2} + c_{-1}x^{-1} + c_0 + c_1x + c_2x^2 + c_3x^3 + \cdots .$$

$$\nabla\varphi = \cdots - 3c_{-3}x^{-4} - 2c_{-2}x^{-3} - c_{-1}x^{-2} + c_1 + 2c_2x + 3c_3x^2 + 4c_4x^3 + \cdots$$

$$\nabla\varphi \cdot f = \cdots - 3c_{-3}x^{-2} - 2c_{-2}x^{-1} - c_{-1} + c_1x^2 + 2c_2x^3 + 3c_3x^4 + 4c_4x^5 + \cdots .$$

- We observe $c_k = 0$ for $k \geq 1$, and $\lambda c_{k+1} = kc_k$ for $k \leq -1$
- Thus, for any $\lambda \in \mathbb{C}$, we have

$$\varphi(x) = c_0\left(1 - \lambda x^{-1} + \frac{\lambda^2}{2}x^{-2} - \frac{\lambda^3}{3}x^{-3} + \cdots\right) = c_0 e^{-\lambda/x}$$

Extended DMD: data-driven approach

- Goal: identify $\boxed{\text{Koopman eigenfunctions}}$ from data
- We construct an augmented state

$$\mathbf{y} = \mathbf{\Theta}^T(\mathbf{x}) = \begin{bmatrix} \theta_1(\mathbf{x}) \\ \theta_2(\mathbf{x}) \\ \vdots \\ \theta_p(\mathbf{x}) \end{bmatrix}$$

which contains the original state and nonlinear measurements, $p \gg n$

- Define the data matrices

$$\mathbf{Y} = \begin{bmatrix} | & | & & | \\ \mathbf{y}_1 & \mathbf{y}_2 & \cdots & \mathbf{y}_m \\ | & | & & | \end{bmatrix}, \qquad \mathbf{Y}' = \begin{bmatrix} | & | & & | \\ \mathbf{y}_2 & \mathbf{y}_3 & \cdots & \mathbf{y}_{m+1} \\ | & | & & | \end{bmatrix}$$

- The best fit matrix is obtained by

$$\mathbf{A_Y} = \underset{\mathbf{A_Y}}{\operatorname{argmin}} \|\mathbf{Y}' - \mathbf{A_Y}\mathbf{Y}\| = \mathbf{Y}'\mathbf{Y}^\dagger$$

- Equivalently

$$\mathbf{A_Y} = \underset{\mathbf{A_Y}}{\operatorname{argmin}} \|\mathbf{\Theta}^T(\mathbf{X}') - \mathbf{A_Y}\mathbf{\Theta}^T(\mathbf{X})\| = \mathbf{\Theta}^T(\mathbf{X}')\left(\mathbf{\Theta}^T(\mathbf{X})\right)^\dagger$$

Approximating Koopman eigenfunctions

- For discrete time dynamics, Koopman eigenfunction satisfies

$$\begin{bmatrix} \lambda\varphi(\mathbf{x}_1) \\ \lambda\varphi(\mathbf{x}_2) \\ \vdots \\ \lambda\varphi(\mathbf{x}_m) \end{bmatrix} = \begin{bmatrix} \varphi(\mathbf{x}_2) \\ \varphi(\mathbf{x}_3) \\ \vdots \\ \varphi(\mathbf{x}_{m+1}) \end{bmatrix}$$

- Expand the eigenfunction in terms of some candidate functions

$$\boldsymbol{\Theta}(\mathbf{x}) = \begin{bmatrix} \theta_1(\mathbf{x}) & \theta_2(\mathbf{x}) & \cdots & \theta_p(\mathbf{x}) \end{bmatrix}$$ set of basis

- So, we have

$$\varphi(\mathbf{x}) \approx \sum_{k=1}^{p} \theta_k(\mathbf{x})\xi_k = \boldsymbol{\Theta}(\mathbf{x})\boldsymbol{\xi}$$

- We obtain the following using the above

$$\left(\lambda \boldsymbol{\Theta}(\mathbf{X}) - \boldsymbol{\Theta}(\mathbf{X}')\right) \boldsymbol{\xi} = \mathbf{0}.$$

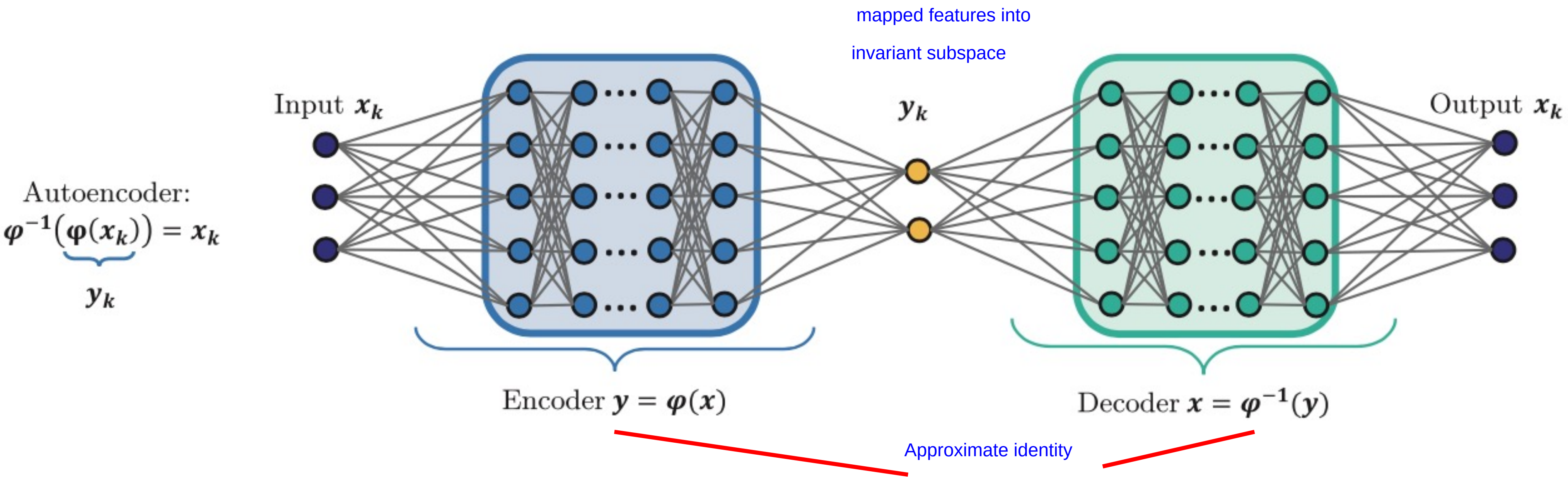
- If we seek the least squares fit

$$\lambda \boldsymbol{\xi} = \boldsymbol{\Theta}(\mathbf{X})^{\dagger} \boldsymbol{\Theta}(\mathbf{X}') \boldsymbol{\xi}$$

- So, we need to find eigenvectors of the above matrix

Using neural networks

- The eigenfunctions of the Koopman operator are complicated
- Deep learning provides a convincing approach
- The auto encoder structure is used naturally, that is, a few latent variables $\mathbf{y} = \boldsymbol{\varphi}(\mathbf{x})$ are used to parameterize the dynamics



- An additional constraint is enforced so that the dynamic is linear on these latent variables
- This constraint can be enforced using the loss function $\|\boldsymbol{\varphi}(\mathbf{x}_{k+1}) - \mathbf{K}\boldsymbol{\varphi}(\mathbf{x}_k)\|$ where K is a matrix