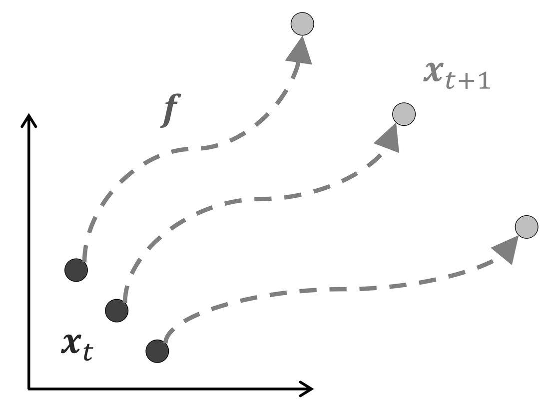
Kernel Learning for Data-Driven Spectral Analysis of Koopman Operators

Naoya Takeishi (RIKEN AIP)

TL;DR For spectral analysis of dynamics based on the Koopman operator, features extracted using diffusion map have good properties in the *infinite-data* regime. To improve the suboptimality in *finite-data* regime, we adjust the kernel function used in diffusion map by making the diffusion operator commute with the Koopman operator as well as possible. We empirically confirmed that this strategy worked well.

What is the Koopman operator?

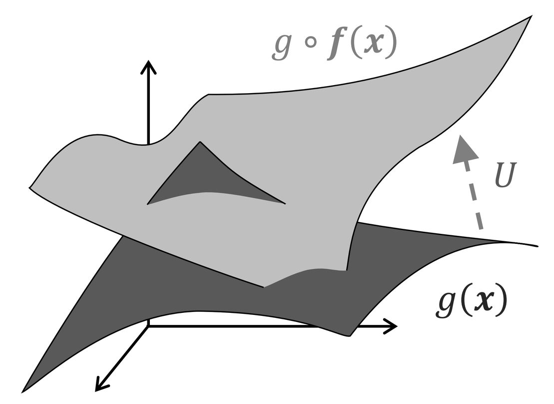
The Koopman operator represents a dynamical system. Using this, we can analyze **nonlinear dynamical systems** using **linear operator** theories.



Discrete-time dynamical system:

$$x_{t+1} = f(x_t), \qquad \begin{array}{ll} x \in M & \text{state vector,} \\ M & \text{state space, and} \\ f\colon M \to M & \text{transition map.} \end{array}$$

When f is nonlinear, analysis is challenging.



Let $g\colon M \to \mathbb{R}$ an *observable* on the state space, in some Hilbert space, say $g \in L^2(M, \mu)$.

The transition of g (instead of x) is described as

$$Ug(x) = g(f(x)).$$

U is a linear (but infinite-dimensional) operator called the *Koopman operator*.

Koopman spectral analysis

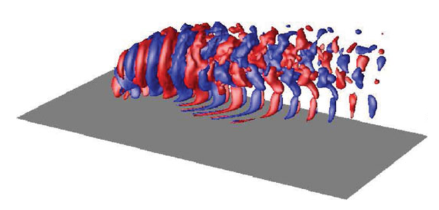
By the spectral decomposition of Koopman operator, we can **extract quasi-periodic components** of dynamics.

$$U^t g(x) = \overbrace{\sum_j \lambda_j^t \varphi_j(x) c_j(g)}^{\text{discrete spectra part}} + \underbrace{\int \sigma^t E_c(\mathrm{d}\sigma) g(x)}_{\text{continuous spectra part}}$$

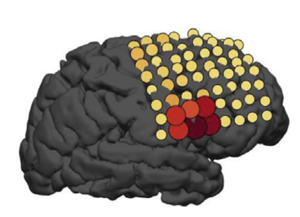
$\lambda_j \in \mathbb{C}$, $\varphi_j\colon M \to \mathbb{C}$ — eigenvalues and eigenfunctions of U

$c_j\colon L^2(M,\mu) \to \mathbb{C}$ — coefficients of g in $\operatorname{span}\{\varphi_j\}$ (*modes*)

A canonical usage Considering concatenation of multiple observables, $\boldsymbol{g} = [g_1 \ \dots \ g_d]\colon M \to \mathbb{R}^d$, watch modes $\{\boldsymbol{c}_j \in \mathbb{C}^d\}$.



[Rowley+ 09]



[Brunton+ 16]

[Takeishi+ 17]

Data-driven computation

There are several data-driven methods for Koopman spectral analysis. In this work, we focus on the use of **diffusion map**.

time-series data

$(\boldsymbol{g}(x_0), \boldsymbol{g}(x_1), \dots, \boldsymbol{g}(x_T))$

↓

basis function [Williams+ 15]
kernel method [Kawahara 16]
neural network [Takeishi+ 17; etc.]
diffusion map [Giannakis 19; etc.]

feature extraction

→

dynamic mode decomposition (DMD)
/ Galerkin method

spectra computation

↑

$\{\boldsymbol{c}_j\}$ Koopman modes
$\{\lambda_j\}$ eigenvalues, and
$\{\varphi_j\}$ eigenfunctions

Review: Diffusion map A kernel integral operator P called diffusion operator, which depicts the geometry of data space, is computed and used for feature extraction [Coifman&Lafon 06].

$$Ph(x) := \int_M \frac{k\big(\boldsymbol{g}(x), \boldsymbol{g}(x')\big)}{d(x)} h(x') \mu(\mathrm{d}x')$$

Proposed method for kernel learning

We learn the kernel function k used in diffusion map so that **the diffusion operator commutes with the Koopman operator** as well as possible.

Important fact In the infinite-data regime, P and U commutes. Hence, their eigenspaces coincide, which is why the feature extraction using diffusion map is good for Koopman spectral analysis [Giannakis 17,19; Giannakis&Das 19].

→ However, in the finite-data regime, this is not the case.

Proposed method We try to *minimize the commutator* $\|KU - UK\|$, where K is the unnormalized version of P, as their properties are common. To this end, we use the fact $\|KU - UK\| \leq \|D\|\|U\|$, where D is defined as

$$Dh(x) := \int_M \Big(k\big(\boldsymbol{g}(f(x)), \boldsymbol{g}(f(x'))\big) - k\big(\boldsymbol{g}(x), \boldsymbol{g}(x')\big) \Big) h(x') \mu(\mathrm{d}x')$$

Finally, we solve

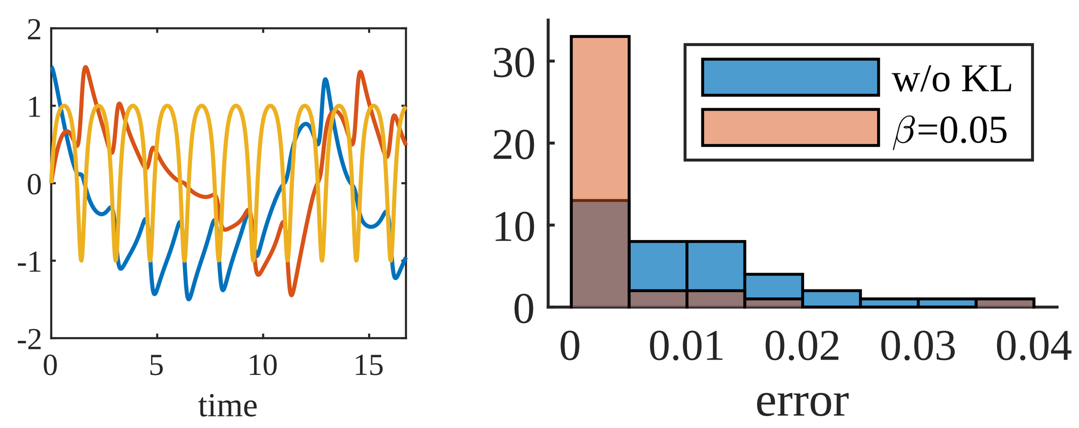
$$\underset{k}{\text{minimize}} \quad \sum_{t,t'} \Big| k\big(\boldsymbol{g}(x_{t+1}), \boldsymbol{g}(f(x_{t'+1}))\big) - k\big(\boldsymbol{g}(x_t), \boldsymbol{g}(x_{t'})\big) \Big|^2 + \beta R(k),$$

where $R(k)$ is a regularization term to prevent trivial solutions.

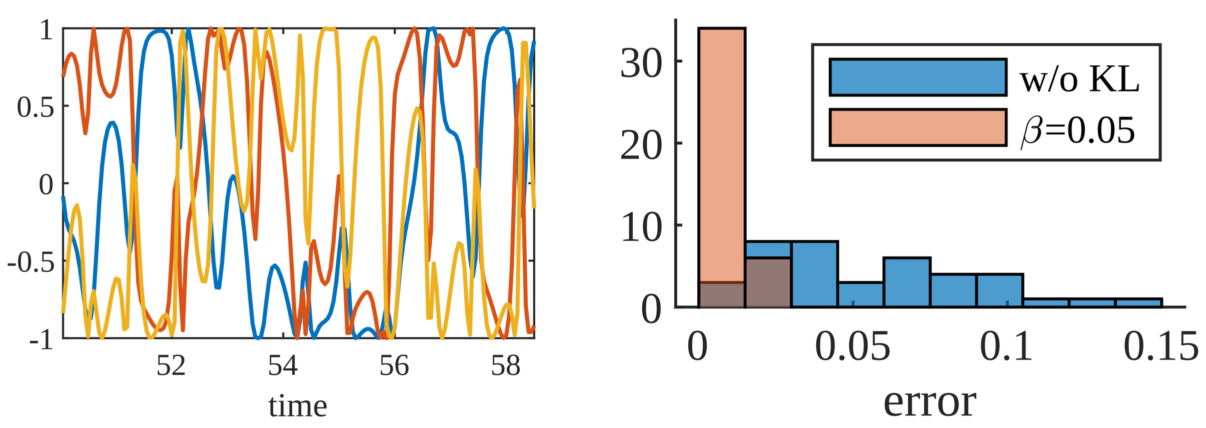
Numerical examples

Evaluated the error between estimation of λ on **large data** and that on **small data**, using k adjusted by the proposed method.

k was modeled as a linear combination of some base kernels (i.e., multiple kernel learning).



(*left*) Torus data, (*right*) histogram of errors over random trials



(*left*) Lorenz data, (*right*) histogram of errors over random trials