Learning dynamics with dynamical distances

From diffusion maps to commute maps and coherence

Ralf Banisch, with Cecilia Clementi, Frank Noé, and Péter Koltai
IPAM December 5th 2016

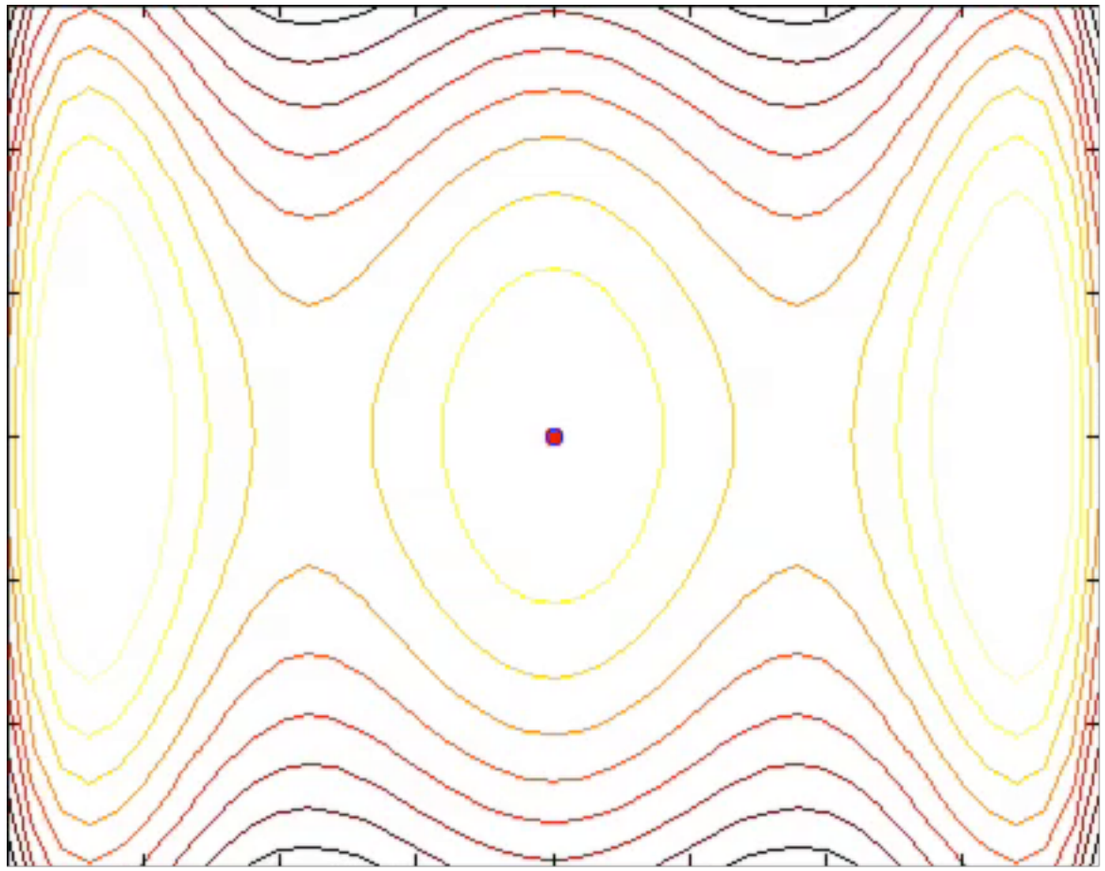
papers:
F. Noé, RB and C. Clementi. Commute maps: Separating slowly mixing molecular configurations for kinetic modeling. J. Chem. Theory Comput. 2016 12(11). DOI: 10.1021/acs.jctc.6b00762

RB and Péter Koltai. Understanding the geometry of transport: diffusion maps for Lagrangian trajectory data unravel coherent sets. arXiv:1603.04709 (accepted in CHAOS)

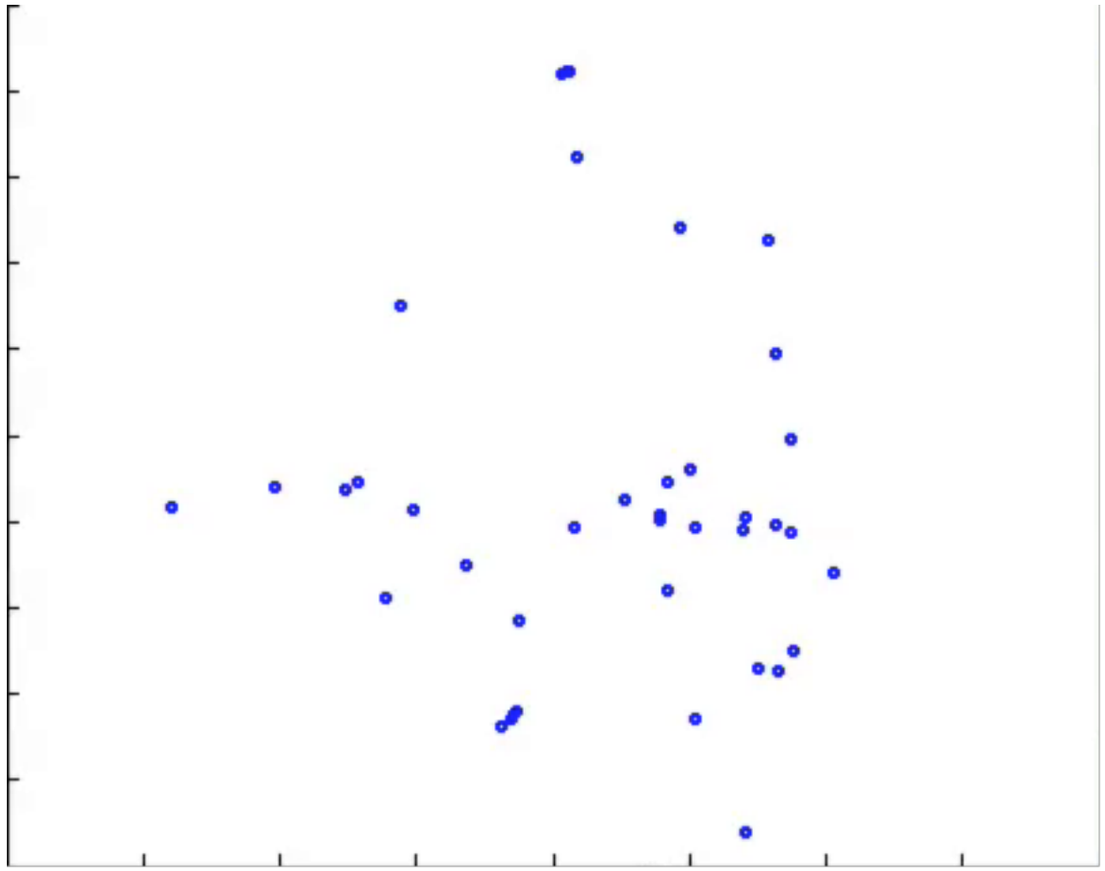
SFB 1114

Freie Universität Berlin

Motivation: Time series



Brownian particle in a potential landscape



earthquake events in Southern California

Outline of the talk

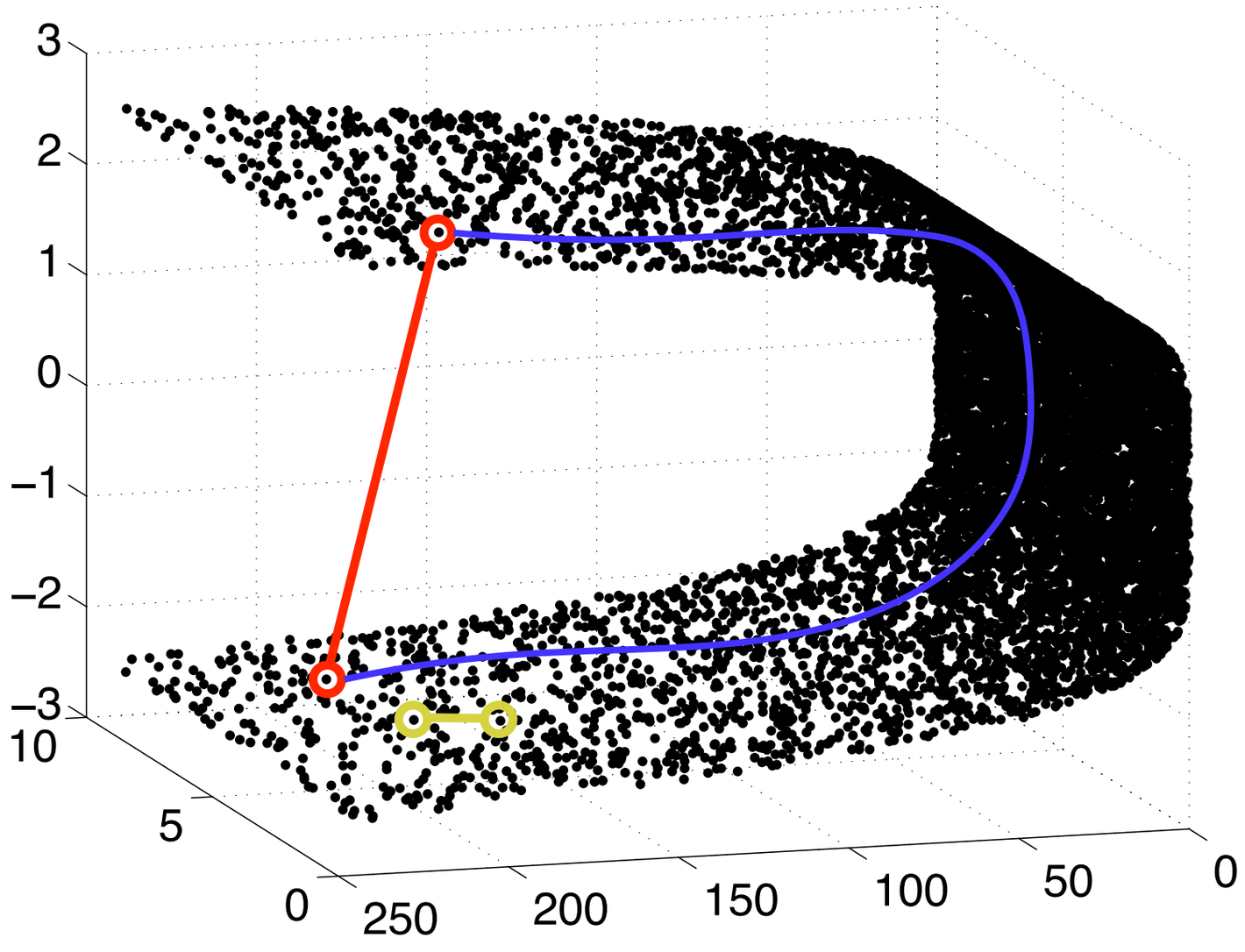
- learning the geometry of point clouds: diffusion maps
- dynamical distances I: commute maps
- dynamical distances II: Time-averaged diffusion maps

Outline of the talk

- **learning the geometry of point clouds: diffusion maps**
- dynamical distances I: commute maps
- dynamical distances II: Time-averaged diffusion maps

Learning geometry: local vs. global

The game: Given data points in $\mathbb{R}^n$ that lie on an **unknown** submanifold ***M*** and are distributed according to an **unknown** distribution ***q***, learn the geometry of ***M***!

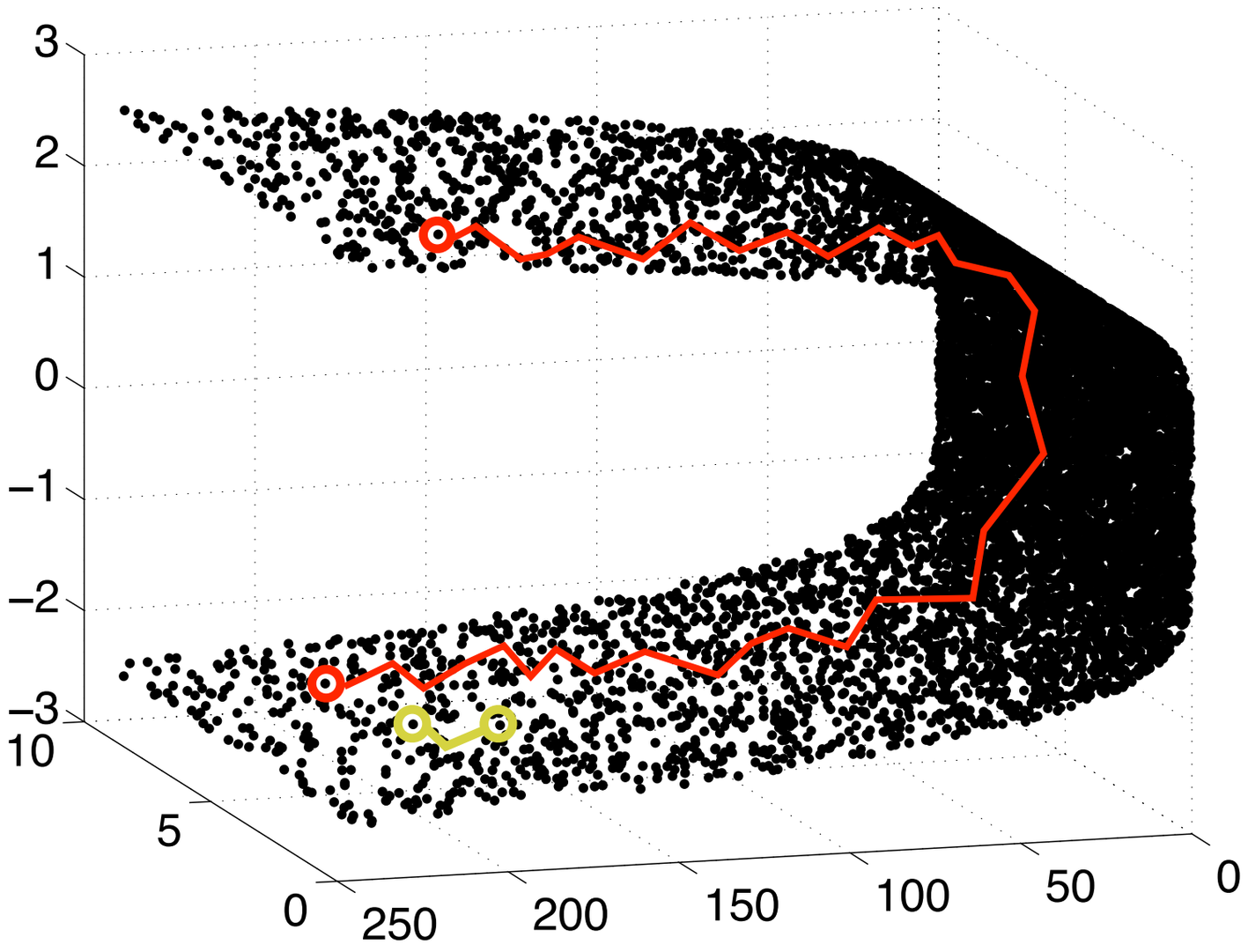


When we play this game, we can trust the Euclidean distance **locally**, but **not globally**!

Learning geometry: local vs. global

[Coifman & Lafon]

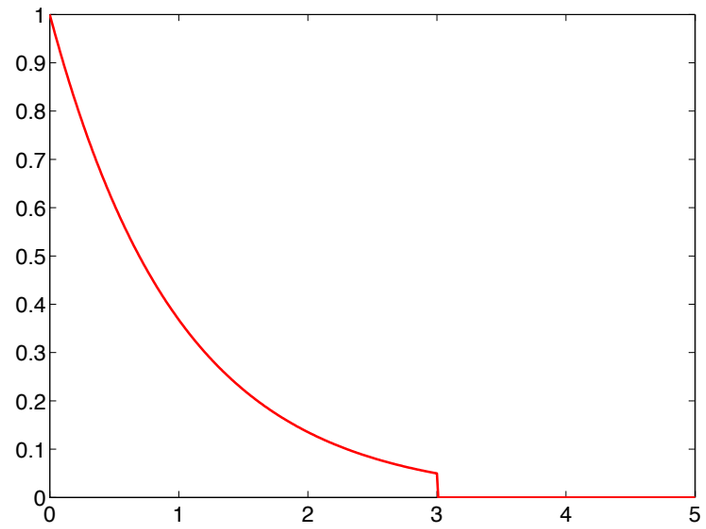
Diffusion maps idea: Build a Markov chain on the data points. Base jump probabilities on Euclidean distance, but only allow local jumps. Then compute distances based on how the Markov chain traverses the dataset.



Learning geometry: constructing the Markov chain

Our data: *m* data points $\{x_i\}_{i=1}^m \in \mathbb{R}^n$.

$$h(x) \sim \exp(-x)\mathbf{1}_{x \leq r}$$



build a similarity matrix based on kernel function ***h***

$$k_\varepsilon(x_i, x_j) = h\left(\frac{\|x_i - x_j\|^2}{\varepsilon}\right)$$

normalize

row-stochastic matrix

$$\mathbf{P}_{\varepsilon,\alpha}(i,j) := \frac{k_\varepsilon^{(\alpha)}(x_i, x_j)}{d_\varepsilon^{(\alpha)}(x_i)}$$

Parameters:

- ε>0 scaling parameter
- α∈[0,1]

Learning geometry: from points to manifolds

Central result in diffusion maps: In the limit of infinite data and for ε→0, the generator of the Markov chain converges to a differential operator on ***M***. `[Coifman & Lafon 2006]`

$$\lim_{\varepsilon \to 0} \lim_{m \to \infty} \mathbf{L}_{\varepsilon,\alpha} f = \Delta f + (2 - 2\alpha) \nabla f \cdot \frac{\nabla q}{q}$$

$$\mathbf{L}_{\varepsilon,\alpha} = \varepsilon^{-1} (\mathbf{P}_{\varepsilon,\alpha} - \mathbf{I})$$

Learning geometry: from points to manifolds

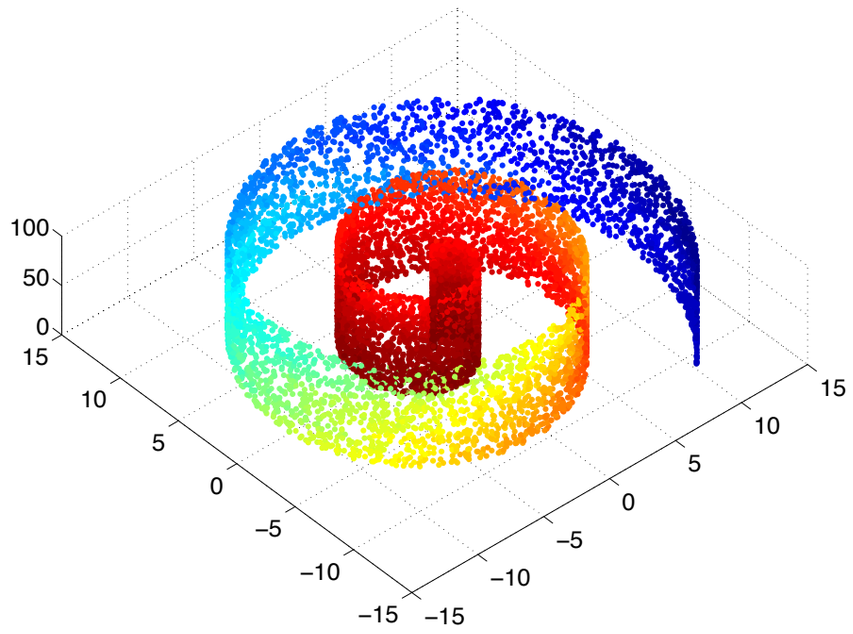
Central result in diffusion maps: In the limit of infinite data and for ε→0, the generator of the Markov chain converges to a differential operator on ***M***. `[Coifman & Lafon 2006]`

$$\lim_{\varepsilon \to 0} \lim_{m \to \infty} \mathbf{L}_{\varepsilon,\alpha} f = \Delta f + (2 - 2\alpha) \nabla f \cdot \frac{\nabla q}{q}$$

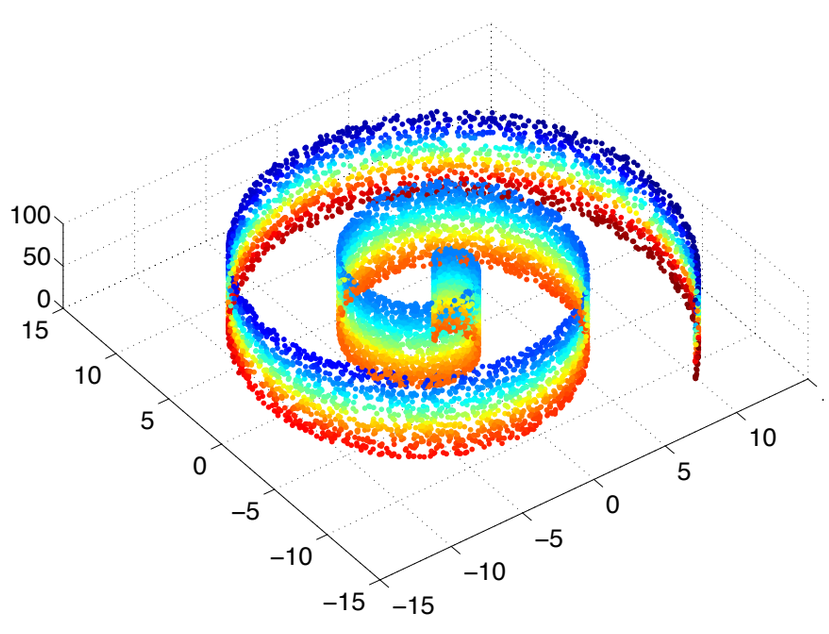
$$\mathbf{L}_{\varepsilon,\alpha} = \varepsilon^{-1} \left(\mathbf{P}_{\varepsilon,\alpha} - \mathbf{I} \right)$$

Why is this nice? Dominant eigenfunctions of Δ are good coordinates on ***M***, and we can approximate the eigenfunctions of Δ by eigenfunctions of $\mathbf{L}_{\varepsilon,1}$.

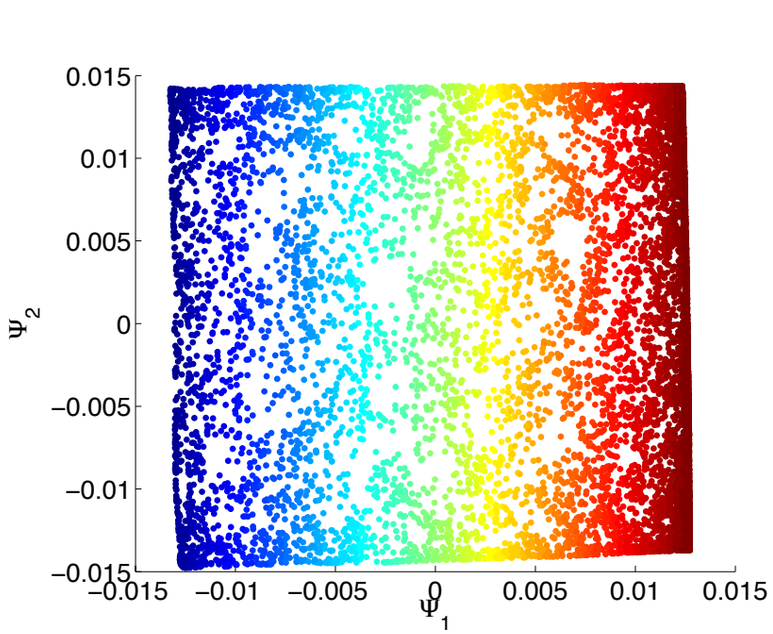
2nd eigenfunction of $\mathbf{L}_{\varepsilon,1}$



3rd eigenfunction of $\mathbf{L}_{\varepsilon,1}$



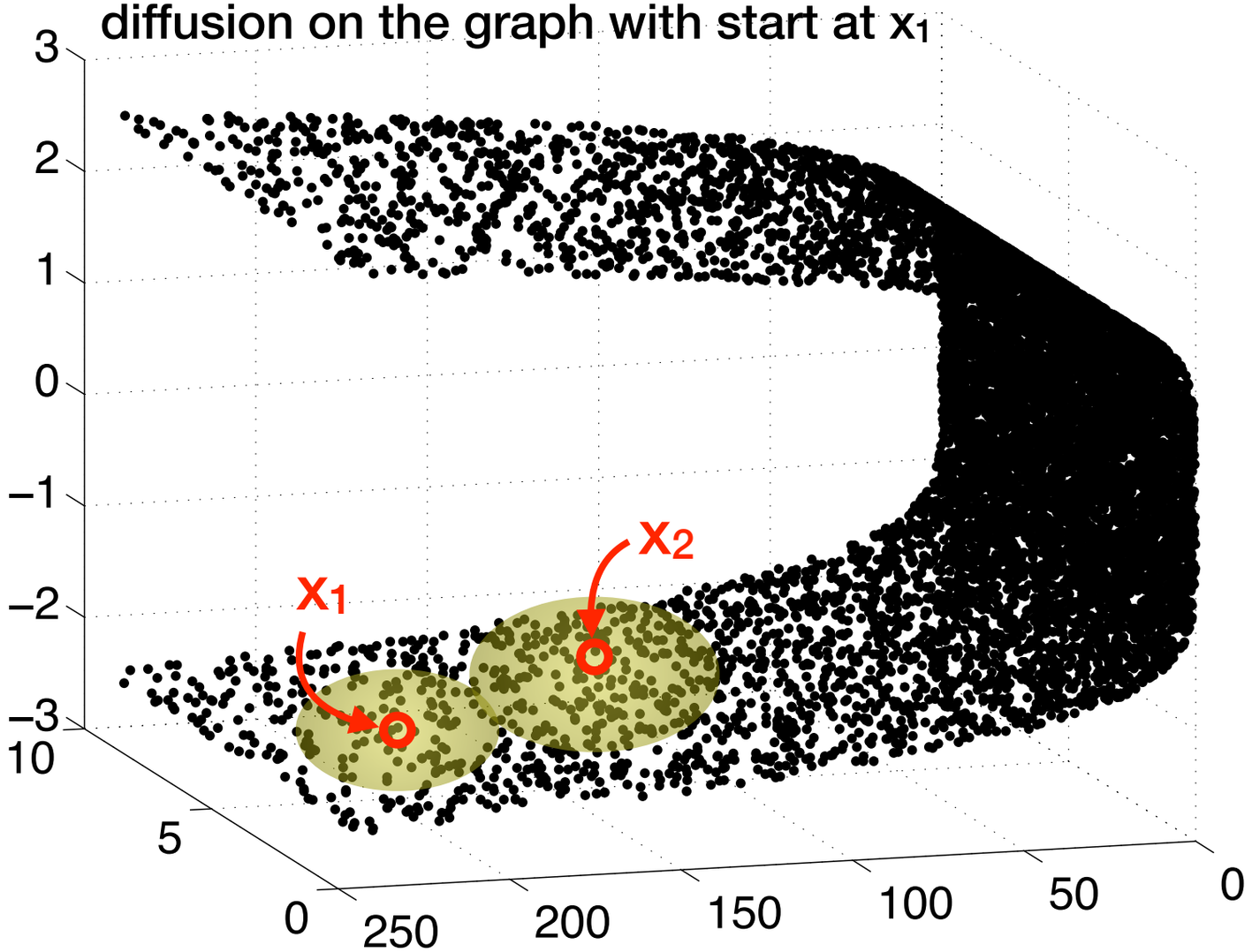
embedding



Diffusion distance

$$D_\tau^2(\mathbf{x}_1, \mathbf{x}_2) = \|p_\tau(\cdot|\mathbf{x}_1) - p_\tau(\cdot|\mathbf{x}_2)\|_{\pi^{-1}}^2 = \sum_{j=1}^{m} \lambda_j^{2\tau} \left(\psi_j(\mathbf{x}_1) - \psi_j(\mathbf{x}_2)\right)^2$$

probability distribution after τ steps of diffusion on the graph with start at x_1



Outline of the talk

- learning the geometry of point clouds: diffusion maps
- **dynamical distances I: commute maps**
- dynamical distances II: Time-averaged diffusion maps

Technical slide: Transfer operators

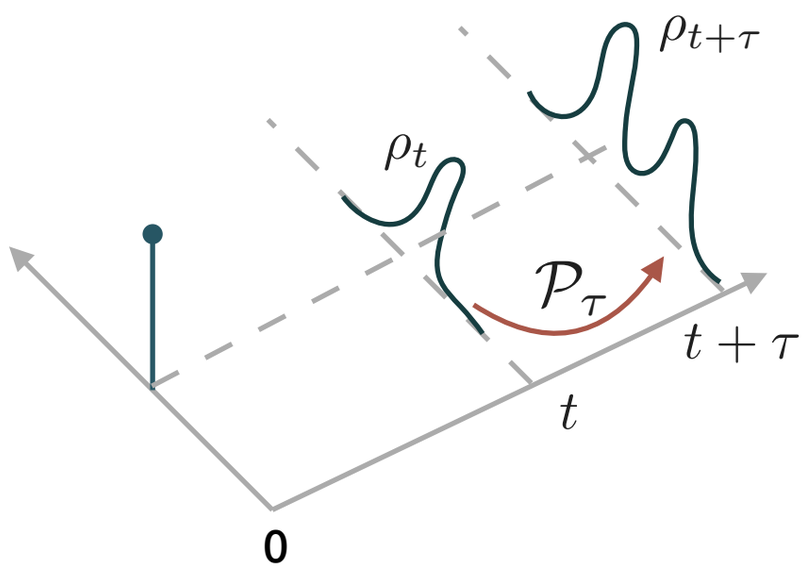
- we can describe the evolution of the dynamics of $\boldsymbol{X_t}$ via a family of transition functions:

$$p_t(x, A) = \mathbf{P}\left[X_t \in A | X_0 = x\right]$$

- **invariant** measure π: $\int p_t(x, A)\pi(dx) = \pi(A)$

- the transition functions induce a one-parameter family of **propagators**:

$$\rho_{t+\tau}(y) = \mathcal{P}_\tau \rho_t(y) = \int \rho_t(x) p_\tau(x, y) dx$$



Technical slide: Transfer operators

$$\rho_{t+\tau}(y) = \mathcal{P}_\tau \rho_t(y) = \int \rho_t(x) p_\tau(x,y) dx$$

- in $L^2(\pi)$ and under appropriate conditions, the propagators have a spectral decomposition with **dominant** isolated real eigenvalues:

$$1 = \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$$

Technical slide: Transfer operators

[Deuflhard, Huisinga, Fischer, Schütte 2000]

$$\rho_{t+\tau}(y) = \mathcal{P}_\tau \rho_t(y) = \int \rho_t(x) p_\tau(x,y) dx$$

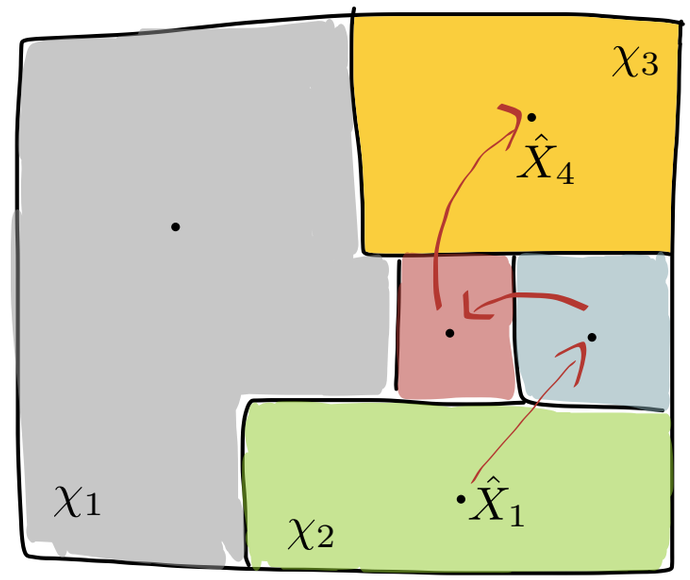
- in $L^2(\pi)$ and under appropriate conditions, the propagators have a spectral decomposition with **dominant** isolated real eigenvalues:

$$1 = \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$$

- **reduced kinetic model:** Markov State Model (MSM) associated with partition of state space

$$P_\tau = Q^T \mathcal{P}_\tau Q$$
$$P_\tau(i,j) = \mathbf{P}\left[X_{t+\tau} \in \chi_j | X_t \in \chi_i\right]$$

variational principle: $\hat{\lambda}_i(\tau) \leq \lambda_i(\tau)$

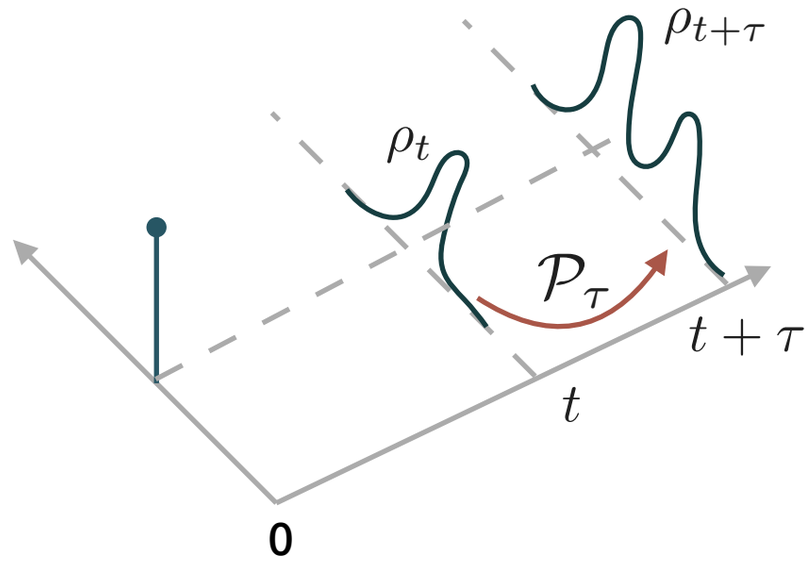


Kinetic distance

[Noé and Clementi 2015]

- the transition functions induce a one-parameter family of **propagators**:

$$\rho_{t+\tau}(y) = \mathcal{P}_\tau \rho_t(y) = \int \rho_t(x) p_\tau(x, y) dx$$



- τ-kinetic distance

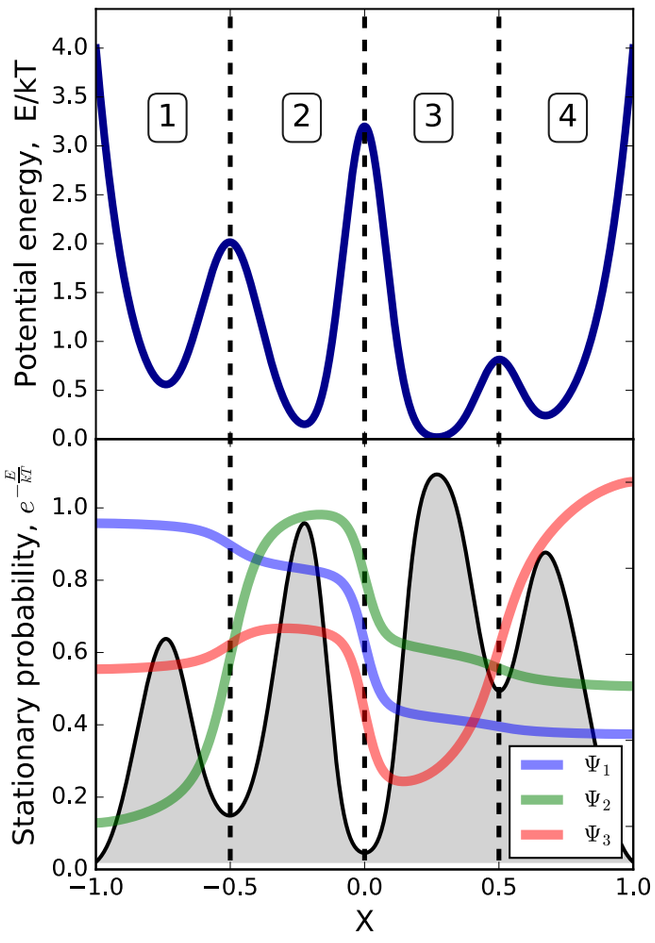
$$D_\tau^2(\mathbf{x}_1, \mathbf{x}_2) = \|p_\tau(\cdot|\mathbf{x}_1) - p_\tau(\cdot|\mathbf{x}_2)\|_{\pi^{-1}}^2 = \sum_{j=1}^{m} \lambda_j^2(\tau) \left(\psi_j(\mathbf{x}_1) - \psi_j(\mathbf{x}_2)\right)^2$$

probability distribution after dynamics is evolved for τ steps with start at x_1

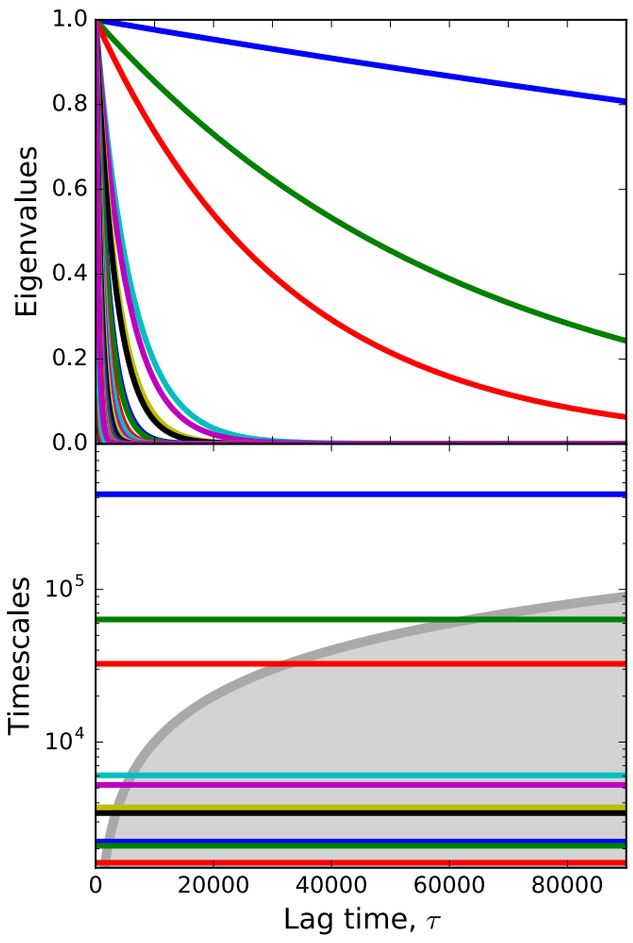
Kinetic distance: Example

[Noé and Clementi 2015]

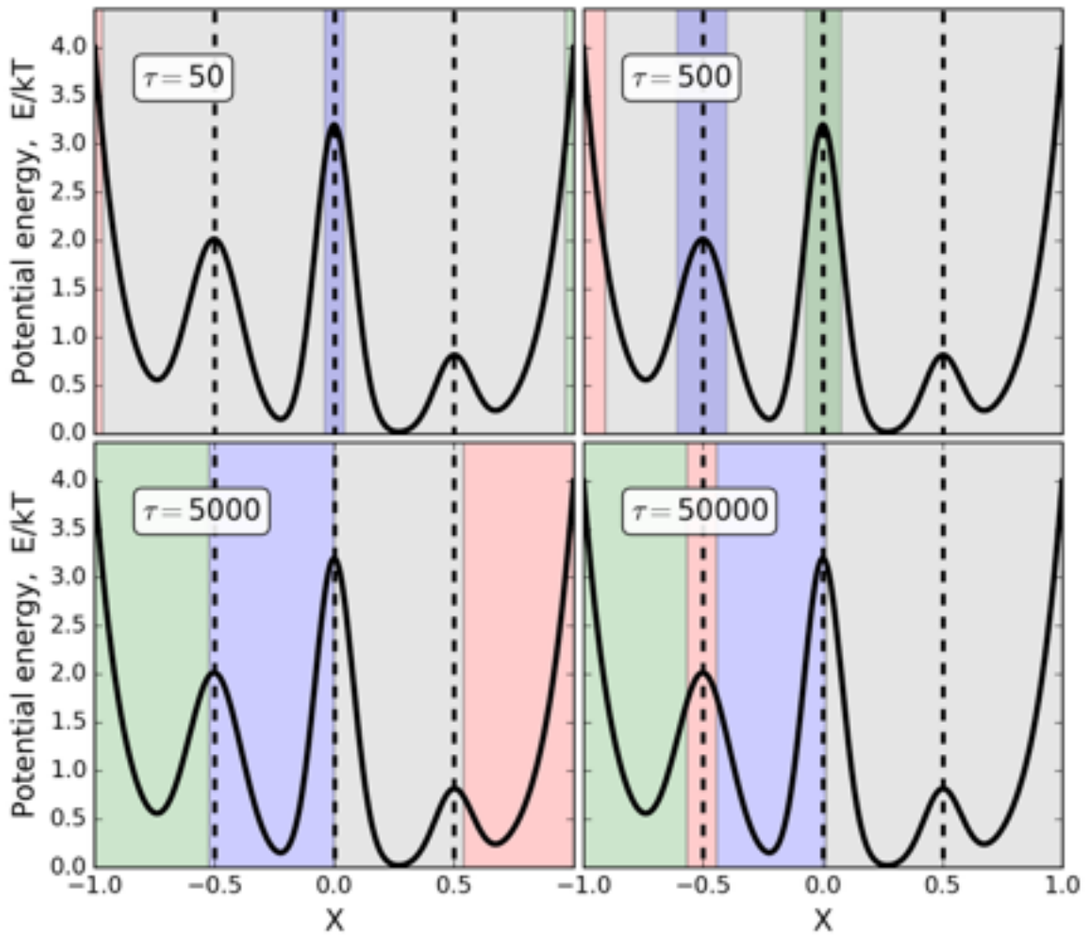
eigenfunctions



time scales



k-means based on $D_\tau(x_1, x_2)$

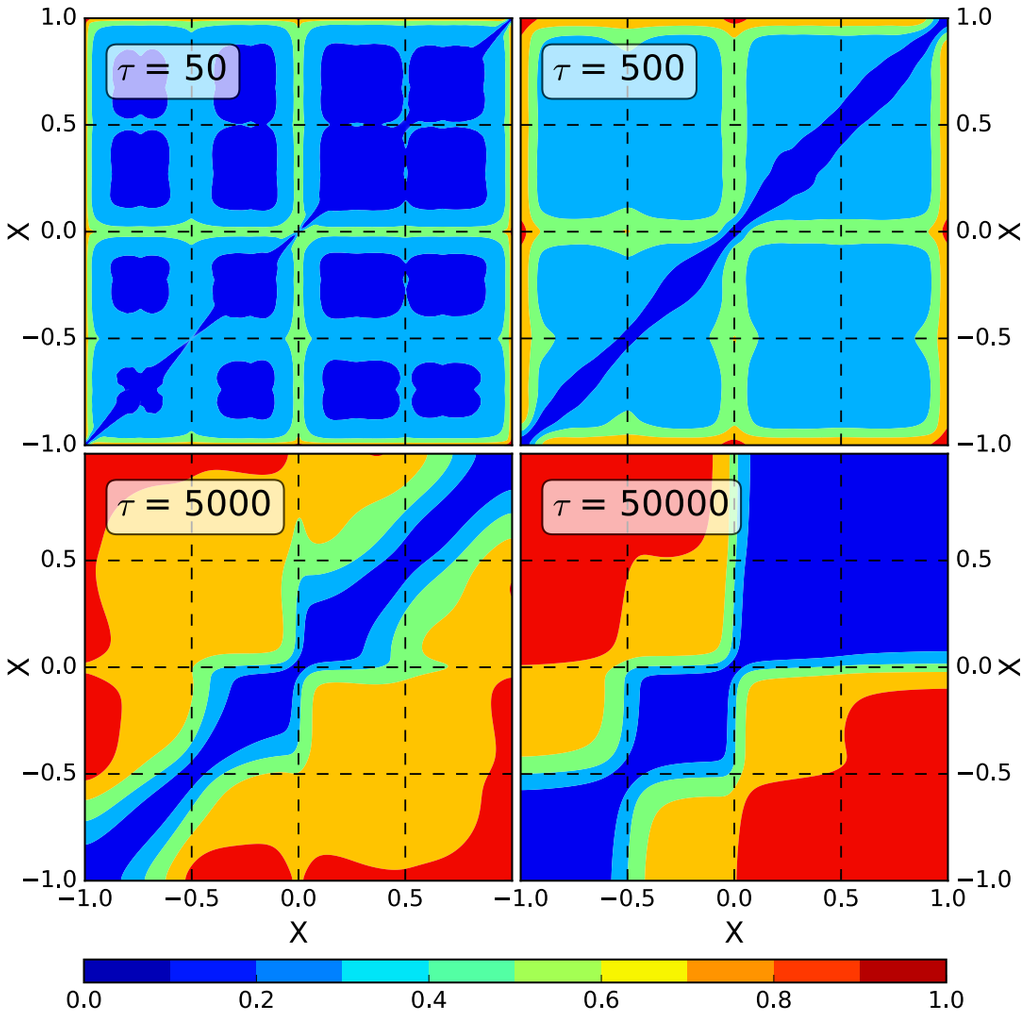


results are strongly τ dependent!

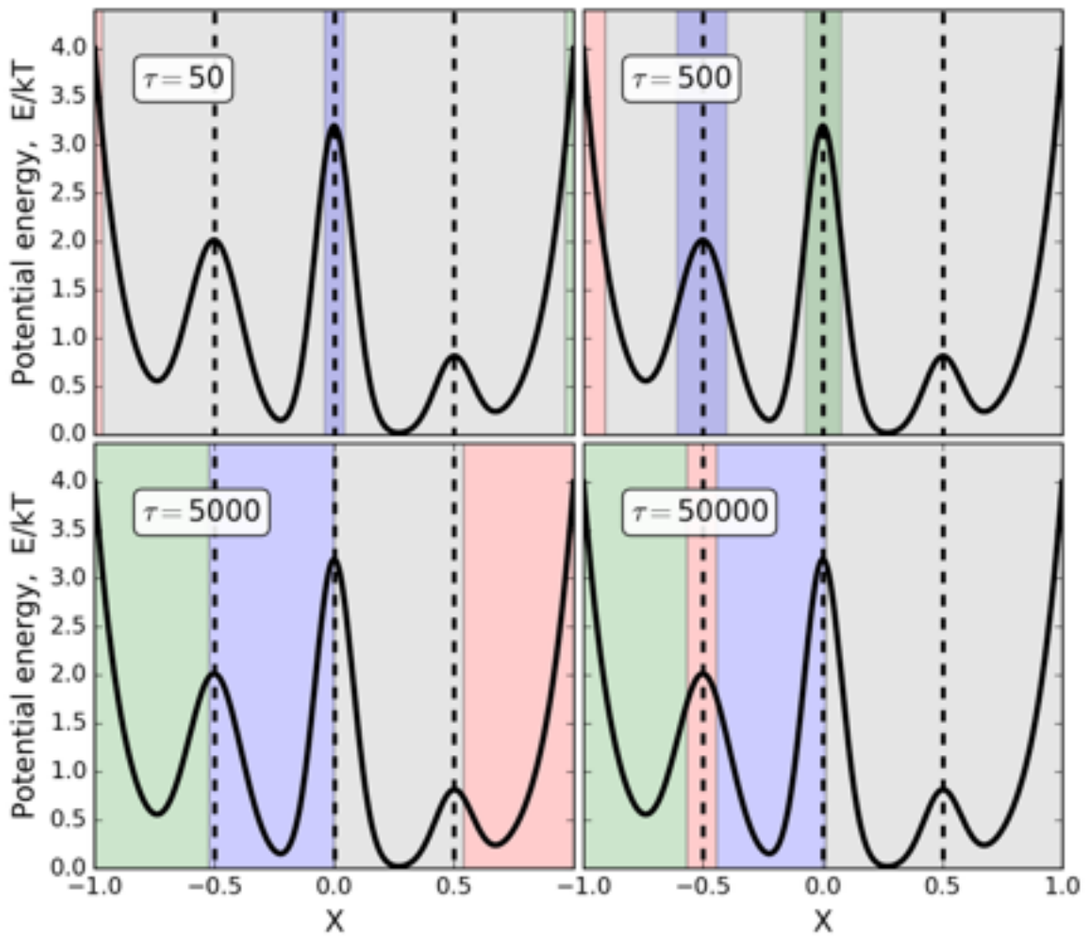
Kinetic distance: Example

[Noé and Clementi 2015]

$D_\tau(x_1, x_2)$



k-means based on $D_\tau(x_1, x_2)$



results are strongly τ dependent!

Commute distance

[Noé, RB and Clementi, JCTC 2016]

- **dynamical** diffusion distance

$$\lambda_j(\tau) = \exp(-t_j/\tau)$$

$$D_\tau^2(\mathbf{x}_1, \mathbf{x}_2) = \|p_\tau(\cdot|\mathbf{x}_1) - p_\tau(\cdot|\mathbf{x}_2)\|_{\pi^{-1}}^2 = \sum_{j=1}^{m} \lambda_j^2(\tau) \left(\psi_j(\mathbf{x}_1) - \psi_j(\mathbf{x}_2)\right)^2$$

- to get rid of dependence on lagtime τ, introduce **commute distance**:

$$d_{\mathrm{comm}}^2(\mathbf{x}_1, \mathbf{x}_2) = \int_0^\infty D_\tau^2(\mathbf{x}_1, \mathbf{x}_2) d\tau = \frac{1}{2} \sum_{j=2}^{m} t_j \left(\psi_j(\mathbf{x}_1) - \psi_j(\mathbf{x}_2)\right)^2$$

- **kinetic content** explained by the first m' eigenvalues:

$$c_{m'} = \frac{\sum_{j=2}^{m'} t_j}{K}, \quad K = \sum_{j=2}^{m} t_j$$

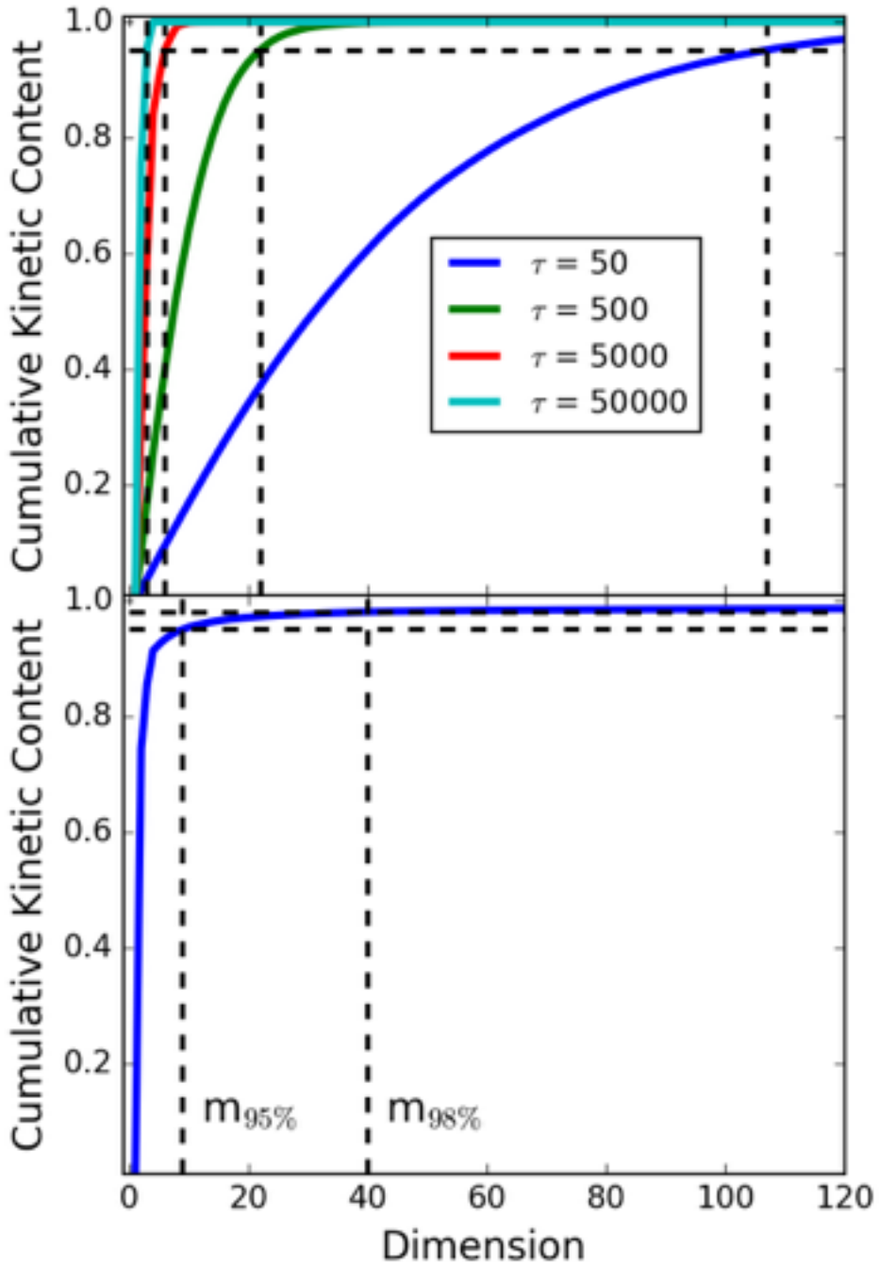
Commute distance

[Noé, RB and Clementi, JCTC 2016]

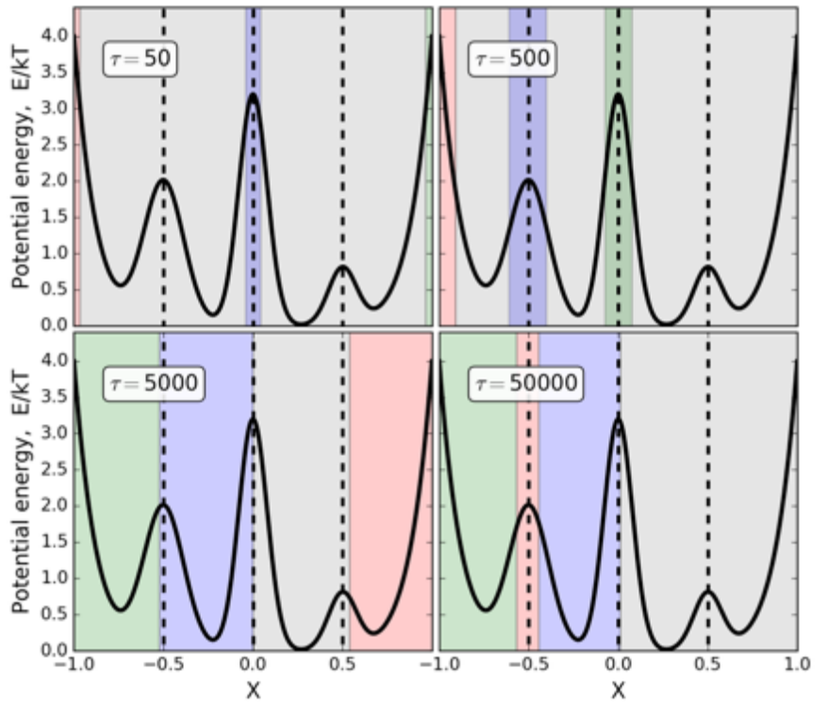
τ-kinetic distance

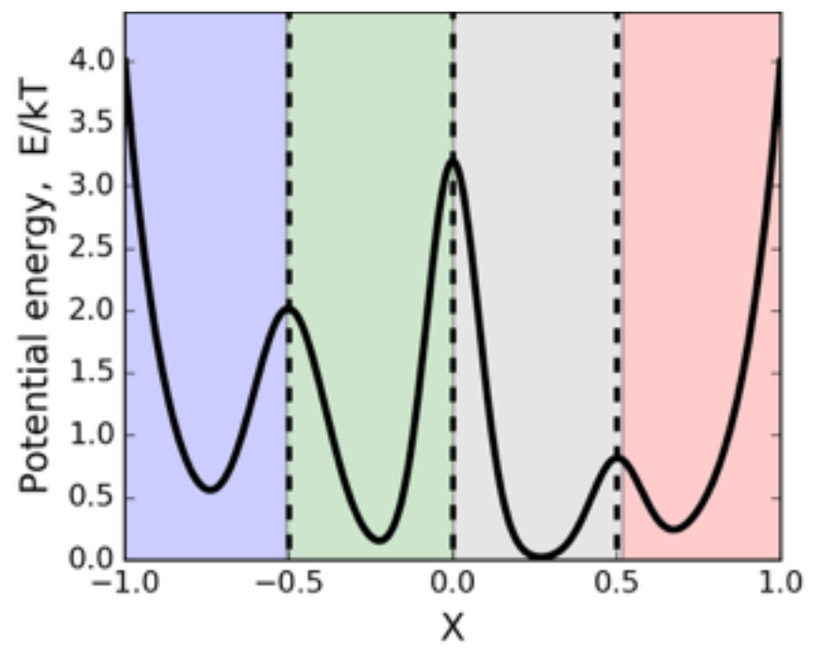
commute distance

kinetic content explained



k-means





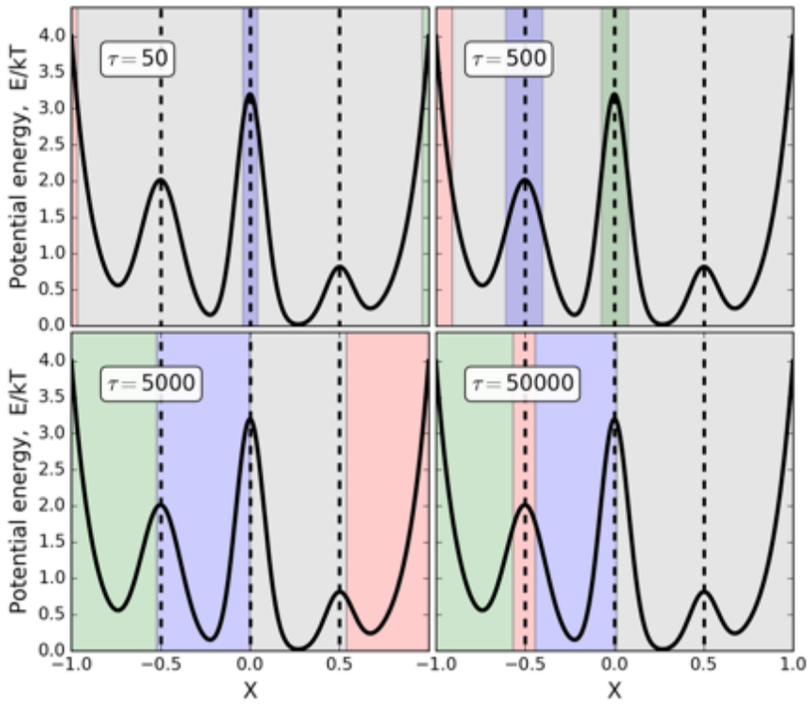
Commute distance

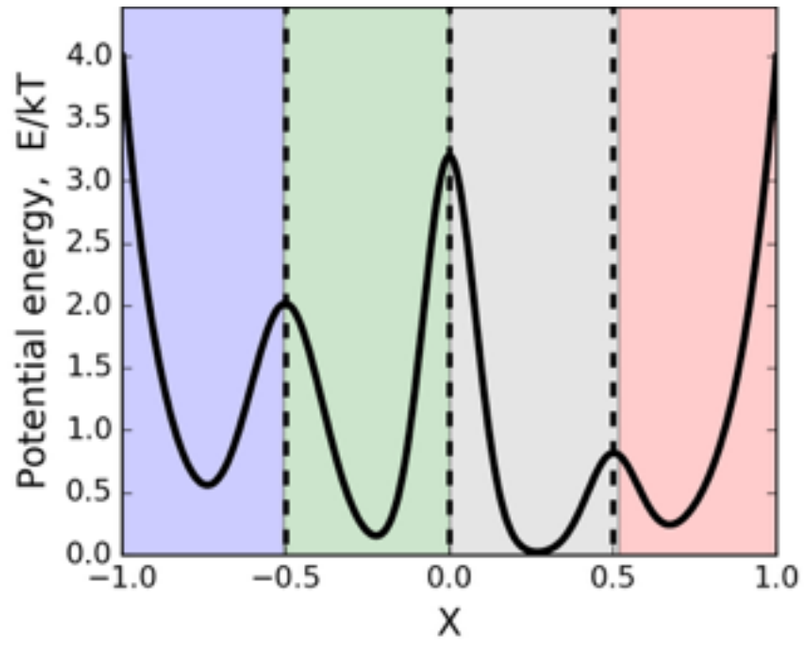
[Noé, RB and Clementi, JCTC 2016]

τ-kinetic distance

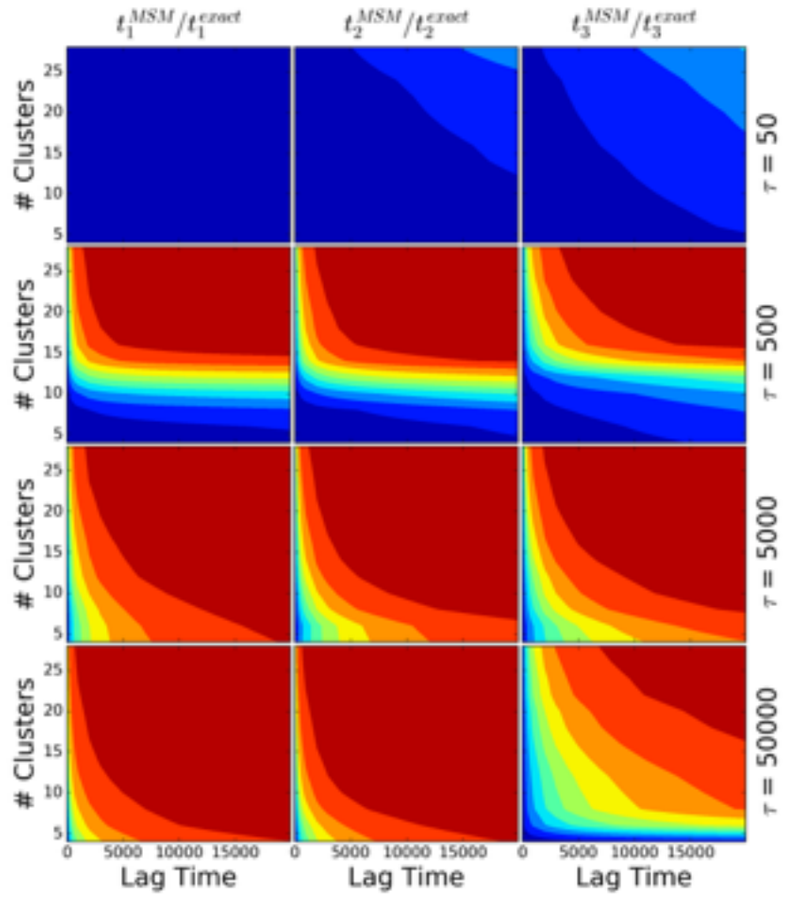
commute distance

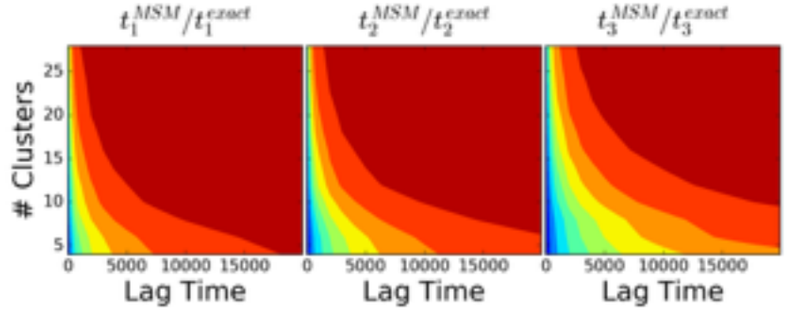
k-means





kinetic model quality





Commute distance vs. commute time

[Noé, RB and Clementi, JCTC 2016]

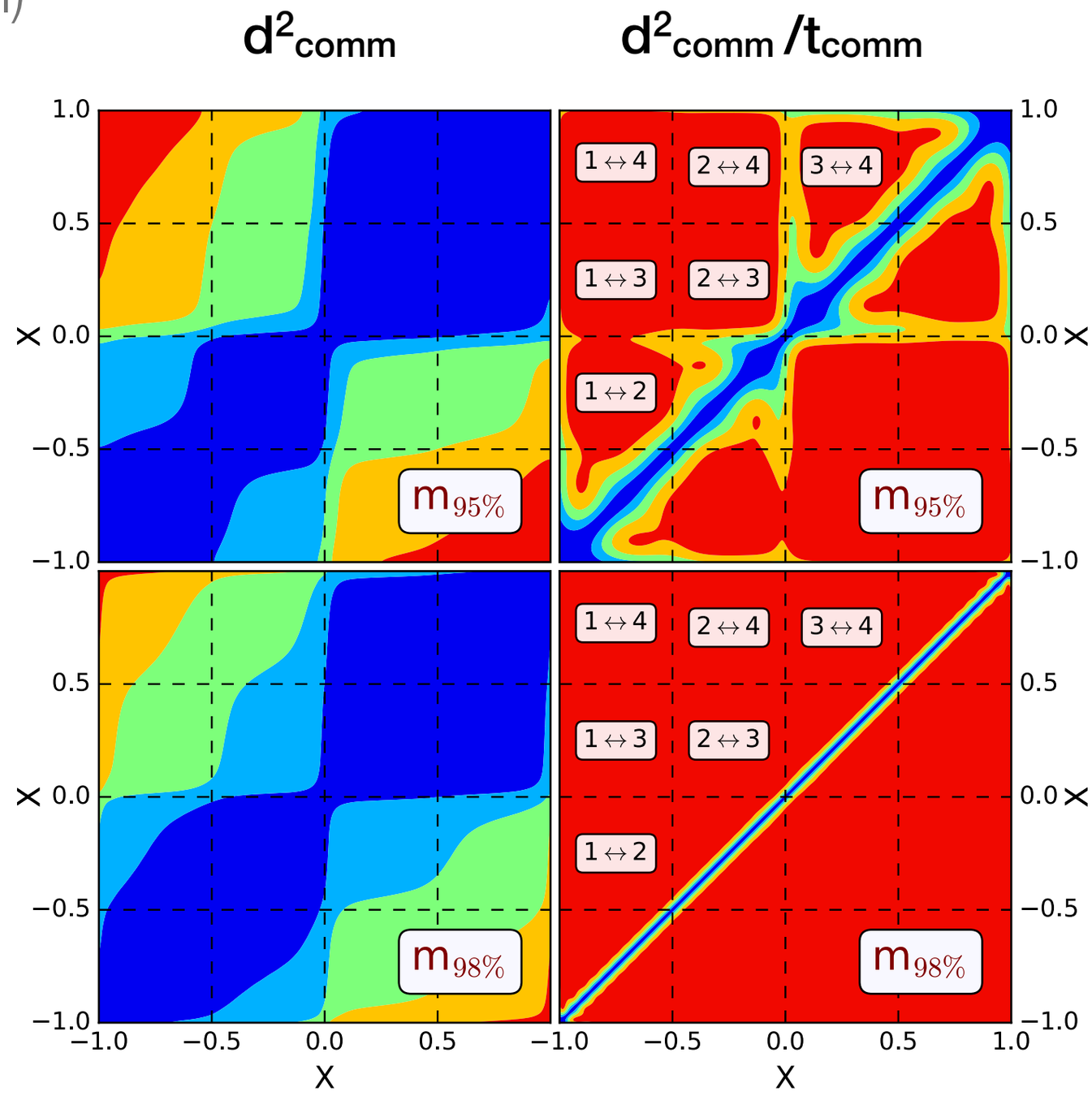
- in discrete state spaces, the (full) commute distance equals the commute time:

$$d^2_{\mathrm{comm}}(\mathbf{x}_i, \mathbf{x}_j) = \frac{t_{ij} + t_{ji}}{2}$$

$$t_{ij} = \mathbf{E}[\text{time to hit } \mathbf{x}_j | \mathbf{x}_i]$$

- if we only use the largest m' eigenvalues, we get an upper bound:

$$\left[d^{(m')}_{\mathrm{comm}}(\mathbf{x}_i, \mathbf{x}_j)\right]^2 \leq \frac{t_{ij} + t_{ji}}{2}$$



Estimation from data

[Noé, Nüske 2013]
[Williams, Kevrekidis, Rowley 2014]

$$d^2_{\mathrm{comm}}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2}\sum_{j=2}^{m} t_j \left(\psi(\mathbf{x}_i) - \psi(\mathbf{x}_j)\right)^2, \quad \lambda_j(\tau) = \exp(-t_j/\tau)$$

VAC / EDMD:

needs estimation

Estimation from data

[Noé, Nüske 2013]
[Williams, Kevrekidis, Rowley 2014]

$$d^2_{\mathrm{comm}}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2}\sum_{j=2}^{m} t_j \left(\psi(\mathbf{x}_i) - \psi(\mathbf{x}_j)\right)^2, \quad \lambda_j(\tau) = \exp(-t_j/\tau)$$

VAC / EDMD:

needs estimation

- given data in form of simulation trajectories $\mathbf{x}_{t,i}$ (t = time, i = coordinate), form the two matrices

$$\mathbf{X}_0 = \begin{bmatrix} x_{0,1} & \cdots & x_{0,n} \\ \vdots & & \vdots \\ x_{T-\tau,1} & \cdots & x_{T-\tau,n} \end{bmatrix} \quad \mathbf{X}_\tau = \begin{bmatrix} x_{0,\tau} & \cdots & x_{\tau,n} \\ \vdots & & \vdots \\ x_{T,1} & \cdots & x_{T,n} \end{bmatrix}$$

- compute the moment matrices $\mathbf{C}_{00} = \mathbf{X}_0^T \mathbf{X}_0, \quad \mathbf{C}_{0\tau} = \mathbf{X}_0^T \mathbf{X}_\tau$
- then solve generalised eigenvalue problem: $\mathbf{C}_{0\tau}\mathbf{R} = \mathbf{C}_{00}\mathbf{R}\mathbf{\Lambda}$
- the eigenvectors are approximated as $\mathbf{\Psi} \approx \mathbf{X}_0\mathbf{R}$
- the commute map is given as $\mathbf{\Psi}' = \mathbf{X}_0\mathbf{R}\mathrm{diag}\left(\sqrt{\hat{t}_i/2}\right)$, with $\hat{t}_i = -\tau/\ln\Lambda_{ii}(\tau)$

Example: BPTI

[F. Noé, RB and C. Clementi, JCTC 2016]

simulation data

↓ VAC/EDMD

approximate eigenfunctions

↓

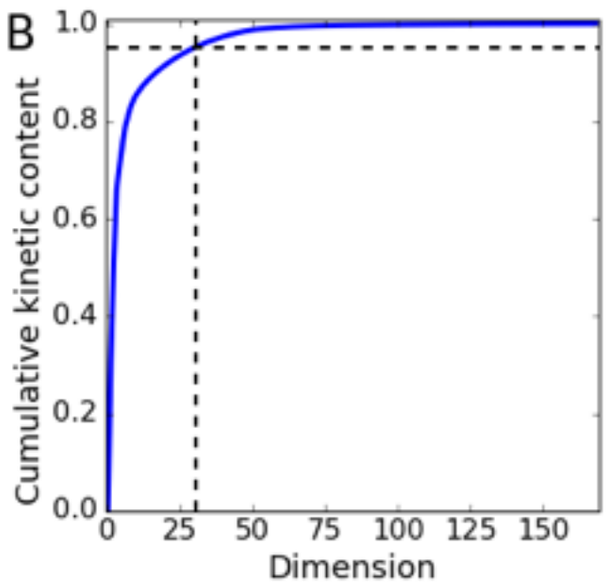
embedding: commute map, τ-kinetic map, …

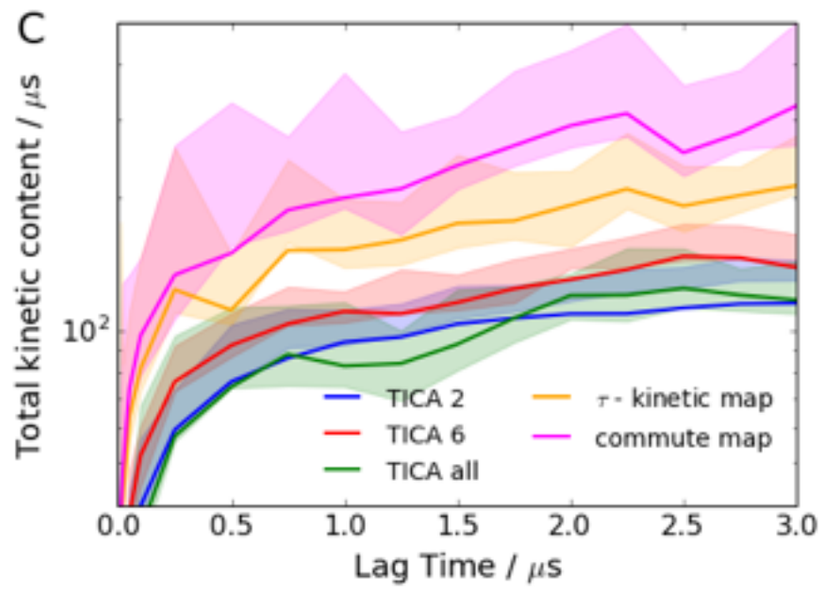
↓ k-means

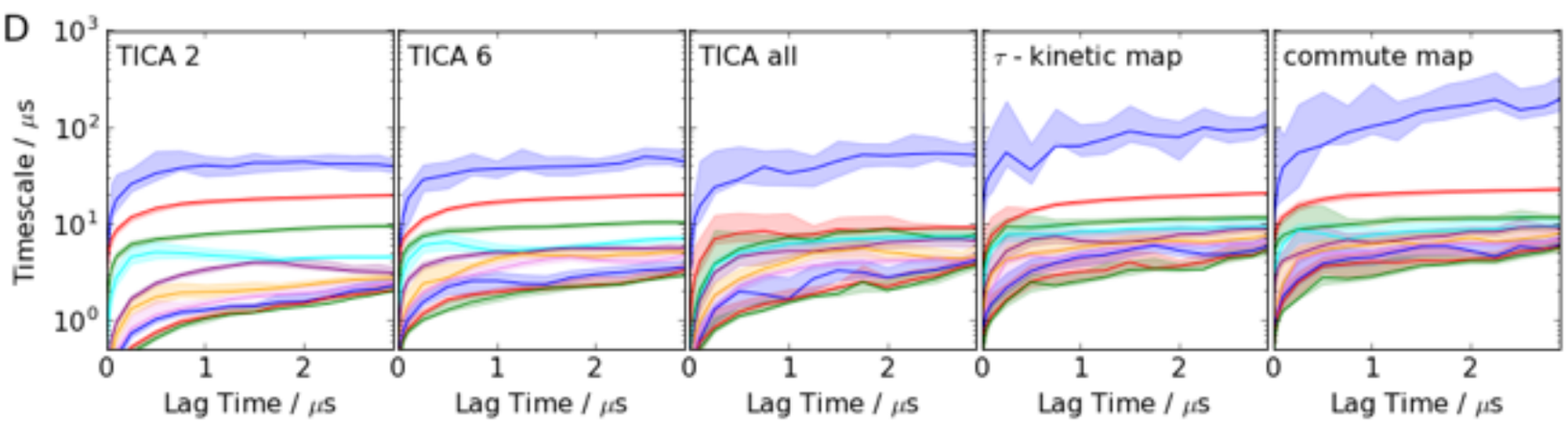
coarse-grained kinetic model (MSM)

data: 1ms simulation of BPTI produced by Anton supercomputer

variational principle $\hat{t}_i \leq t_i$







Summary of Part II

- given data in form of simulation trajectories $\mathbf{x_{t,i}}$, we can use VAC/EDMD to approximate the dominant time scales and eigenfunctions of the propagator.

- from these eigenfunctions, we can compute the commute map embedding.

$$\mathbf{\Psi}' = \mathbf{X}_0 \mathbf{R} \mathrm{diag}\left(\sqrt{\hat{t}_i/2}\right)$$

- the commute distance is the Euclidean distance in the commute map space.

$$d^2_{\mathrm{comm}}(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2} \sum_{j=2}^{m} t_j \left(\psi(\mathbf{x}_j) - \psi(\mathbf{x}_j)\right)^2 = \left\| \psi'_i - \psi'_j \right\|^2$$

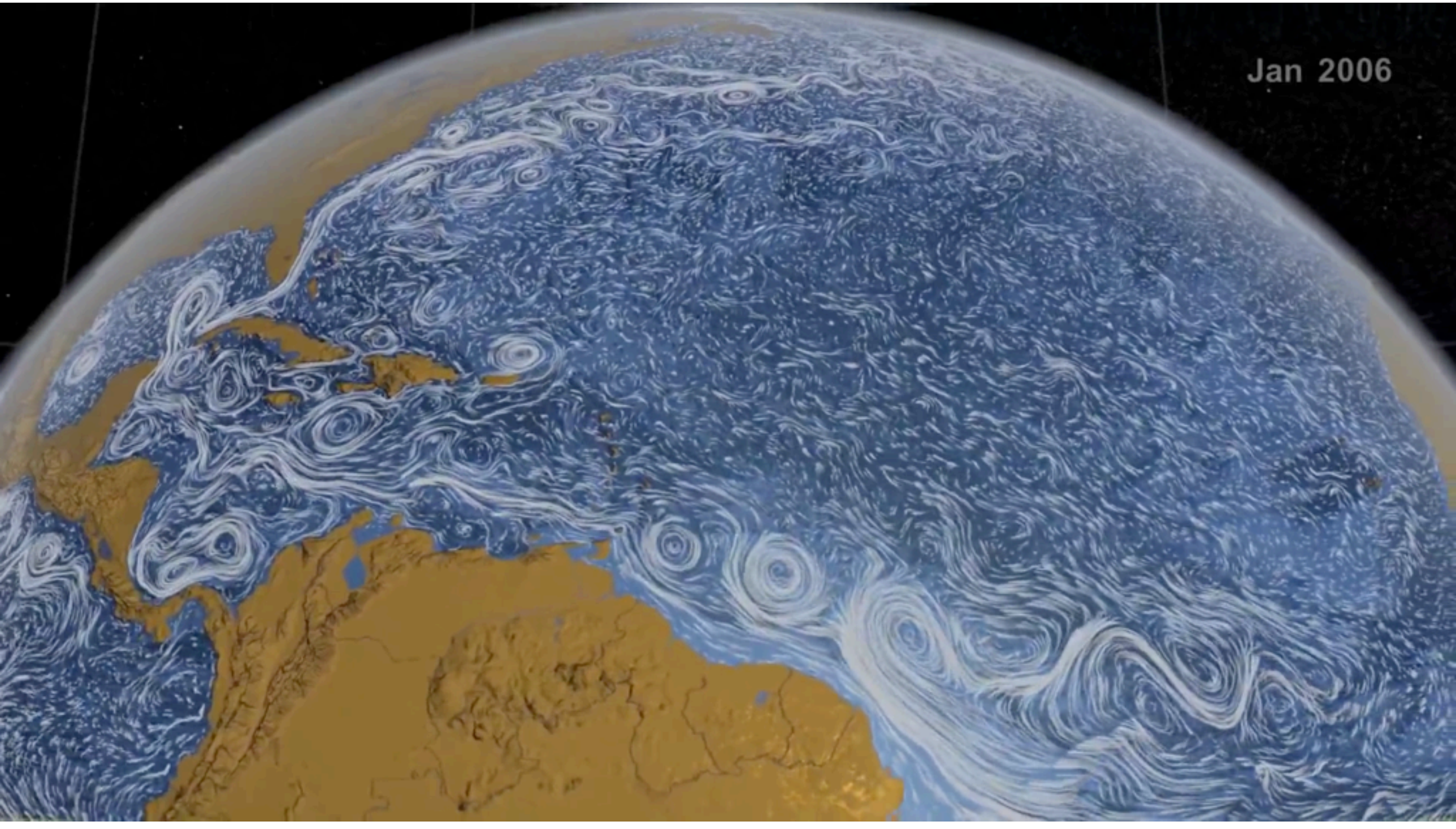
- with geometric clustering techniques such as k-means, we can construct coarse grained kinetic models (MSMs) from the commute map.

- the resulting models outperform models constructed from different embeddings.

Outline of the talk

- learning the geometry of point clouds: diffusion maps
- dynamical distances I: commute maps
- **dynamical distances II: Time-averaged diffusion maps**

Motivation: What are coherent sets?



Time-averaged diffusion maps

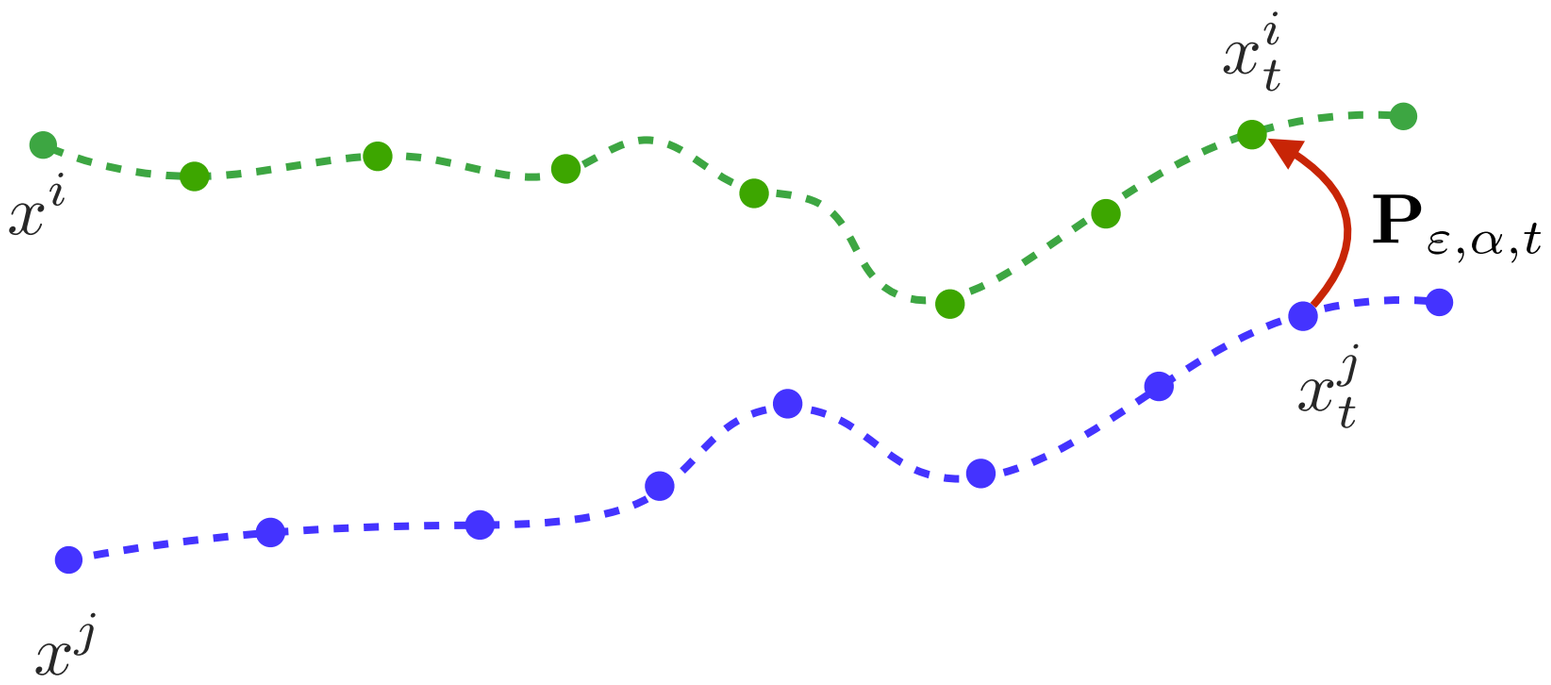
[RB & Koltai 2016]

The game: Given ***mT*** data points which are organised as ***m*** trajectories produced from an **unknown** dynamical system Φ_t sampled at ***T*** time points $I_t = \{t_0, \ldots, t_{T-1}\}$, i.e. given the dataset

$$X = \{x_t^i := \Phi_t x^i \ : \ i = 1, \ldots, m; \ t \in I_t\},$$

learn something about the dynamics of Φ_t !

Idea: Build a Markov chain on the trajectory data! Use diffusion maps to jump between points <u>in the same time slice</u>.



Time-averaged diffusion maps

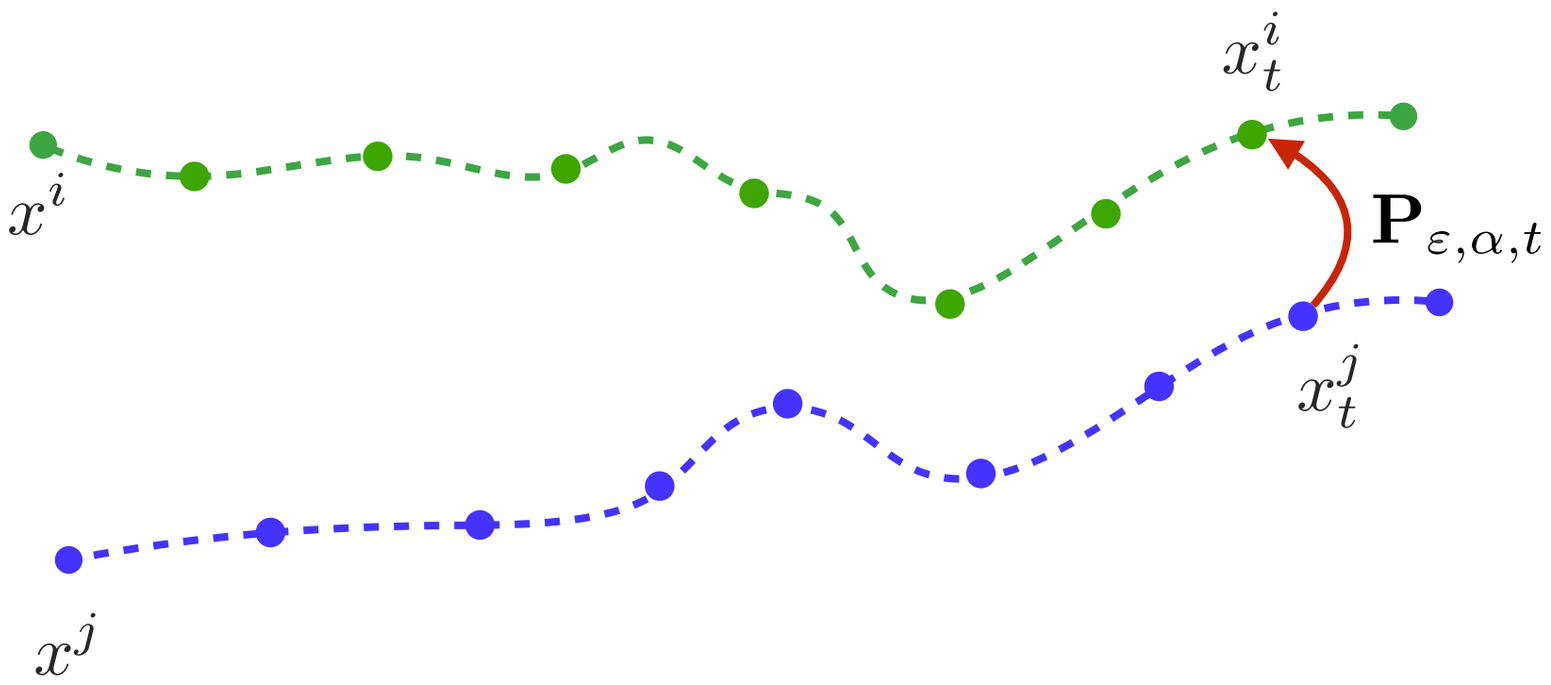
[RB & Koltai 2016]

The game: Given ***mT*** data points which are organised as ***m*** trajectories produced from an **unknown** dynamical system Φ_t sampled at ***T*** time points $I_t = \{t_0, \ldots, t_{T-1}\}$, i.e. given the dataset

$$X = \{x_t^i := \Phi_t x^i \ : \ i = 1, \ldots, m; \ t \in I_t\},$$

learn something about the dynamics of Φ_t !

Idea: Build a Markov chain on the trajectory data! Use diffusion maps to jump between points <u>in the same time slice</u>.



Average over time slices, α = 1/2:

$$\mathbf{Q}_\varepsilon = \frac{1}{T} \sum_{t \in I_t} P_{\varepsilon,t}$$

average over time slices

diffusion maps at time slice t

Time-averaged diffusion maps

[RB & Koltai 2016]

Central result: If Φ_t is a diffeomorphism (i.e. the flow map of an ODE), then the limit of infinite data and for small ε, $\mathbf{Q}_\varepsilon$ converges to an operator (as in diffusion maps):

$$\lim_{\varepsilon\to 0}\lim_{m\to\infty}\frac{1}{\varepsilon}(\mathbf{Q}_\varepsilon - \mathrm{Id}) = \frac{1}{2T}\sum_{t\in I_t}\mathcal{P}_t^*\Delta_{q_t}\mathcal{P}_t$$

$$\Delta_q f = q^{-1}\nabla\cdot(q\nabla f)$$

compare with dynamic Laplacian:

$$\hat{\Delta} = \frac{1}{2}\left(\Delta + \mathcal{P}_t^*\Delta\mathcal{P}_t\right)$$

The dominant eigenfunctions $\{\Psi_1, \ldots, \Psi_d\}$ of $\mathbf{Q}_\varepsilon$ can be used to define a **time-averaged diffusion distance:**

$$d_T^2(\mathbf{x}_1,\mathbf{x}_2) = \sum_{j=1}^{m}\lambda_j^2\left(\psi_j(\mathbf{x}_1) - \psi_j(\mathbf{x}_2)\right)^2$$

Some related work

Coherent Sets:

- Analytical framework: G. Froyland, K. Padberg-Gehle, N. Santitissadeekorn, A. Mohanan, S. Lloyd and others, 2010-2015
- Shape coherence: T. Ma and E. Bollt, 2014

Clustering in trajectory space:

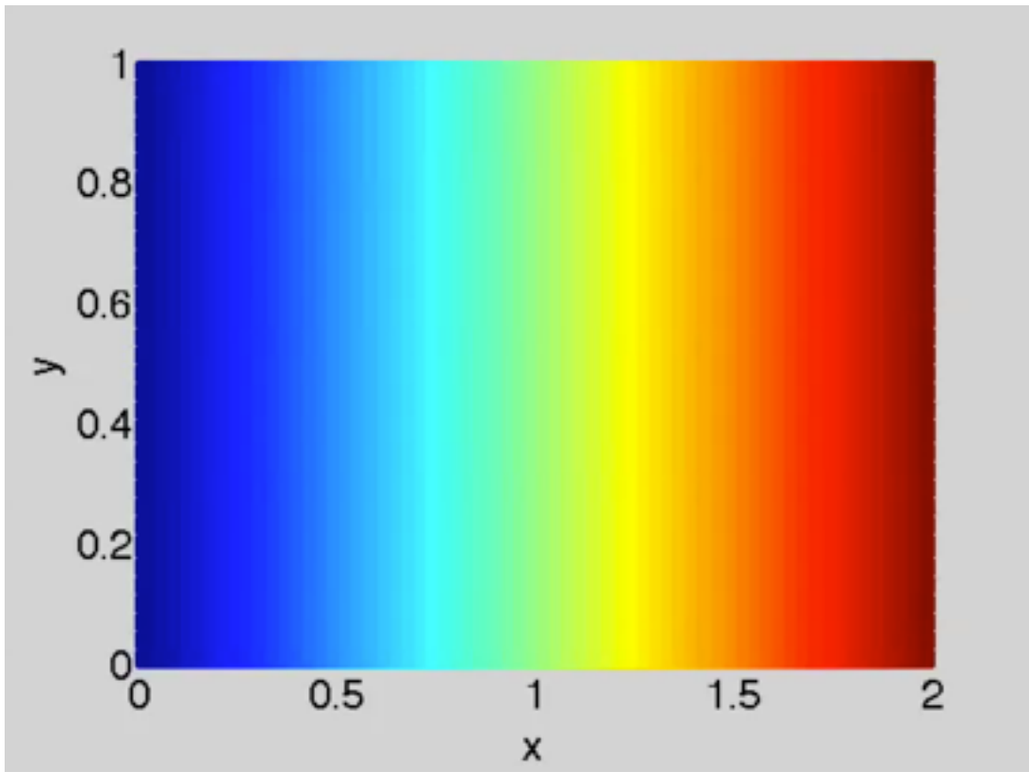
- Dynamical distance + fuzzy c-means: K. Padberg-Gehle and G. Froyland, 2015
- Dynamical distance + spectral clustering: A. Hadjighasem et. al., 2015

Transfer operator approximation:

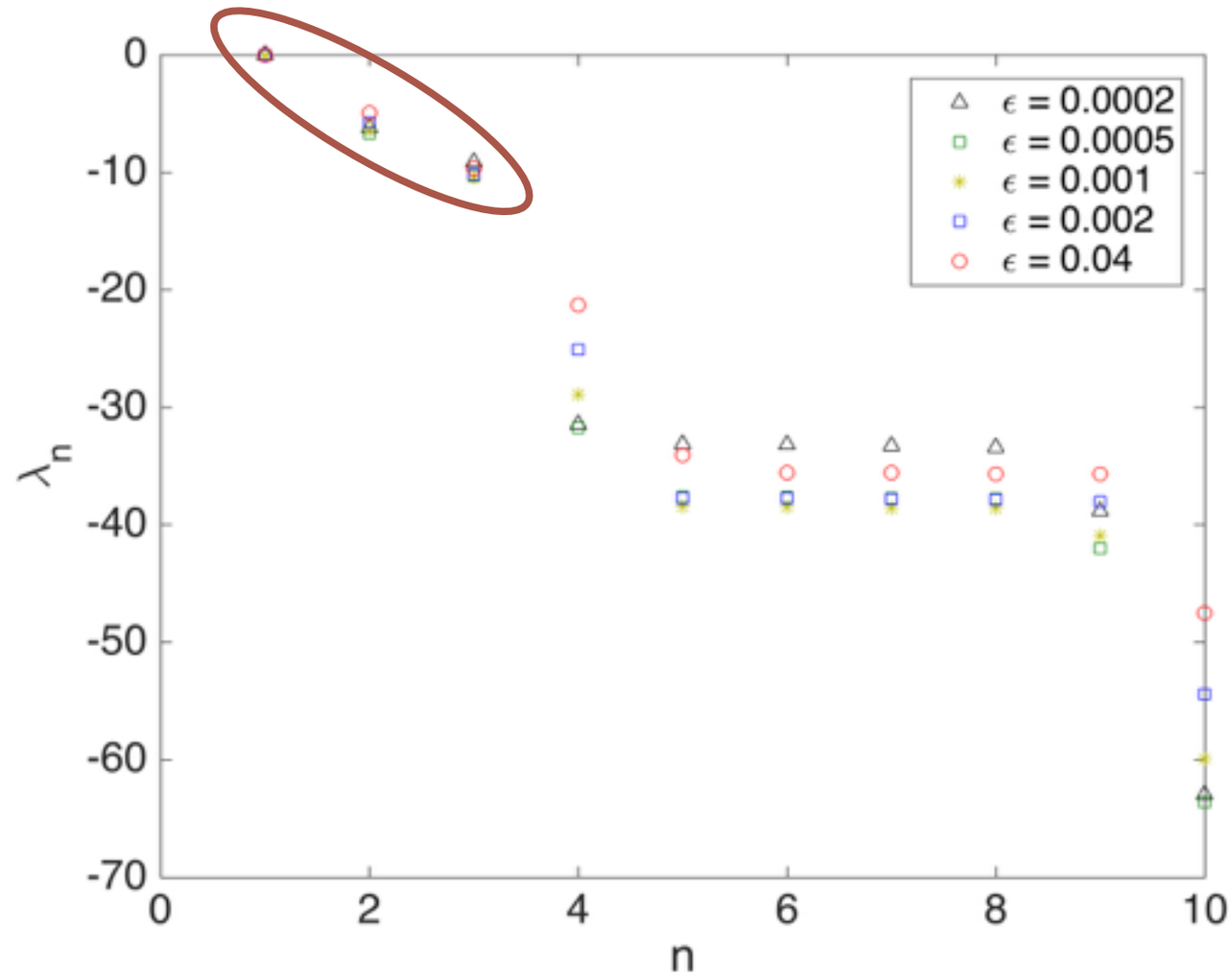
- DMD and EDMD: I. Mezic and others, 2013-2016
- Radial basis function collocation: O. Junge and A. Denner, 2016
- RBF and thin plate splines: M. Williams and C. Rowley, 2015

Double gyre: Eigenvalues

We consider a data-rich case first: 20.000 trajectories with initial conditions on uniform grid.



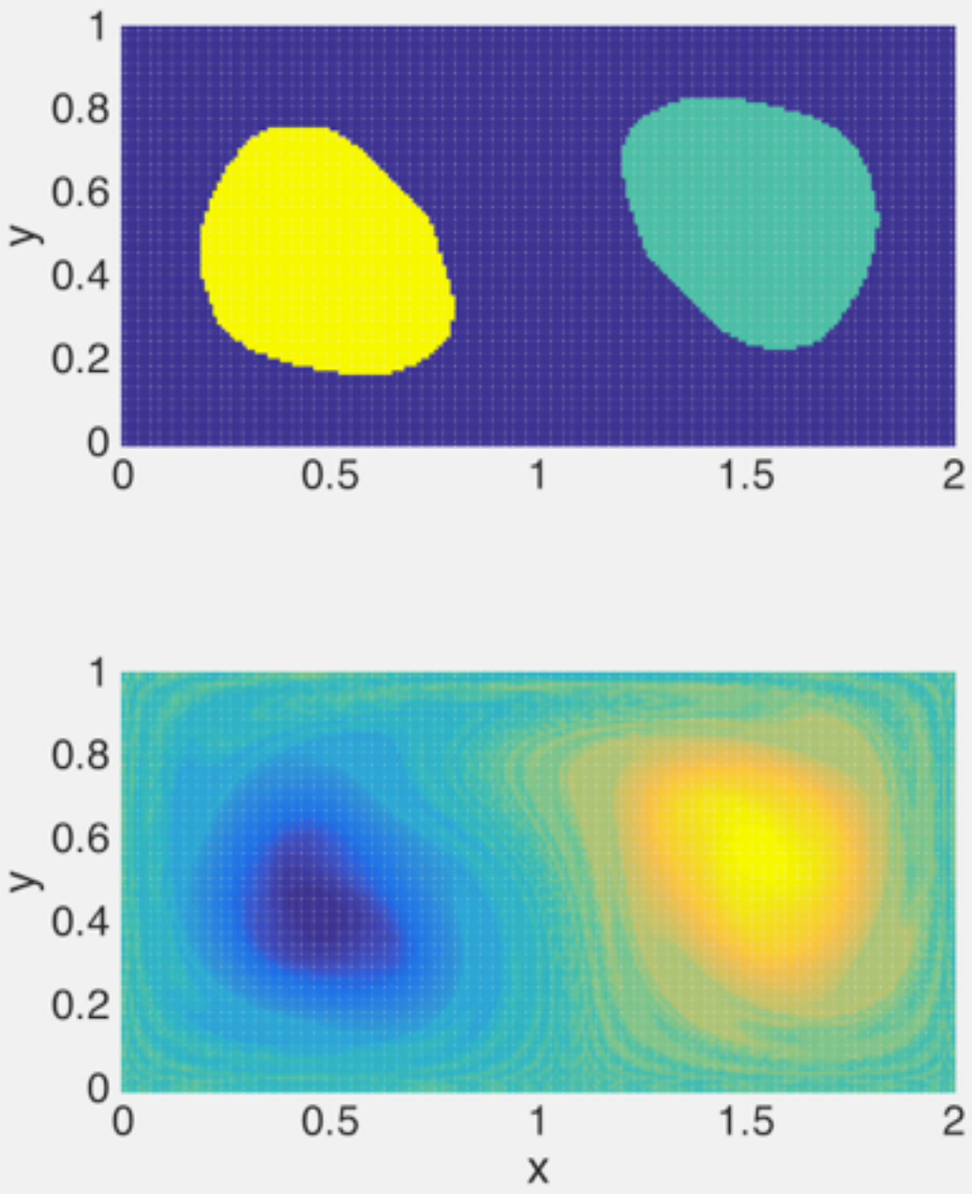
eigenvalues of $\mathbf{L}_\varepsilon = \varepsilon^{-1}\left(\mathbf{Q}_\varepsilon - \mathbf{I}\right)$



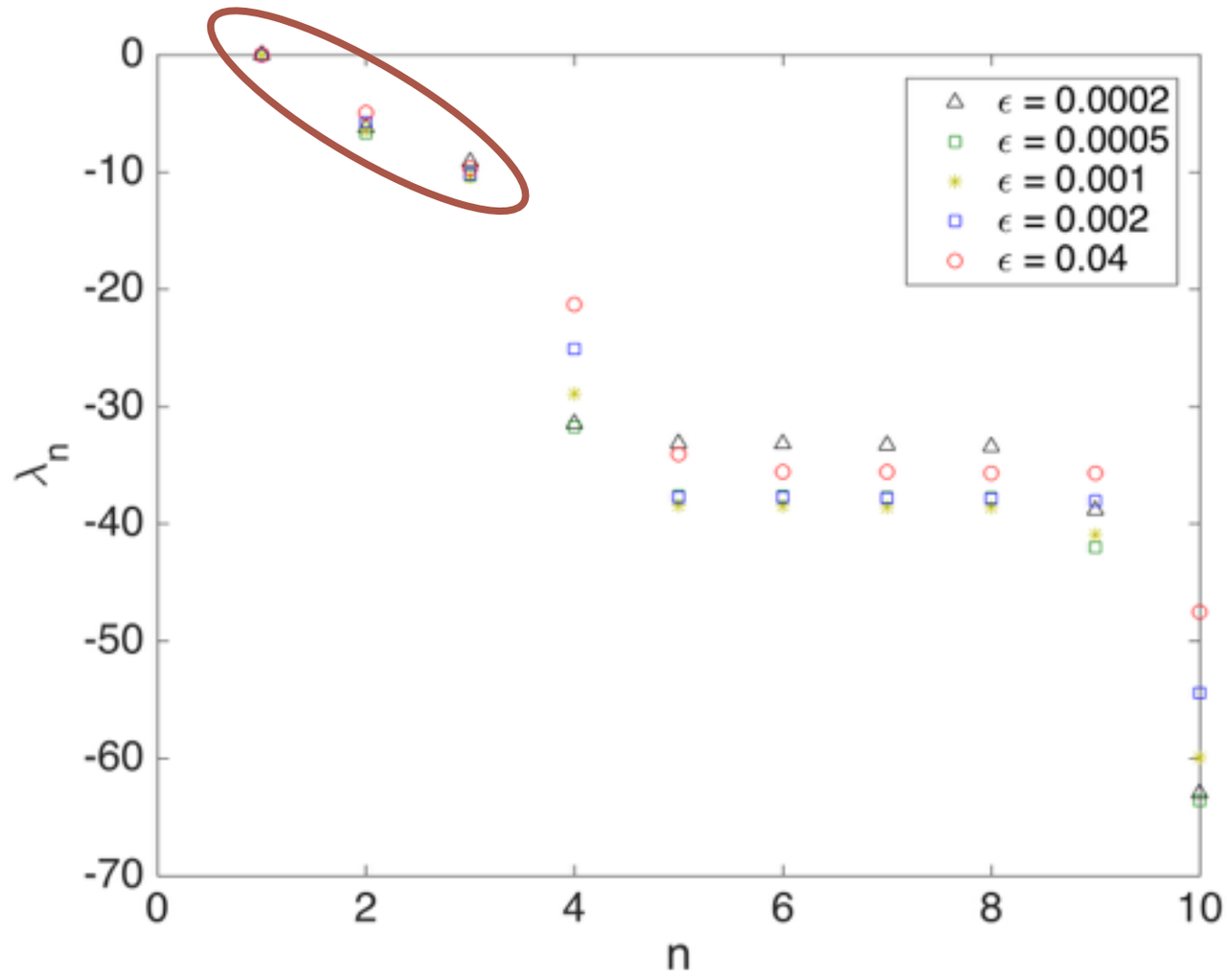
Double gyre: 3-Clustering & Eigenfunctions

We consider a data-rich case first: 20.000 trajectories with initial conditions on uniform grid.

ε = 0.004



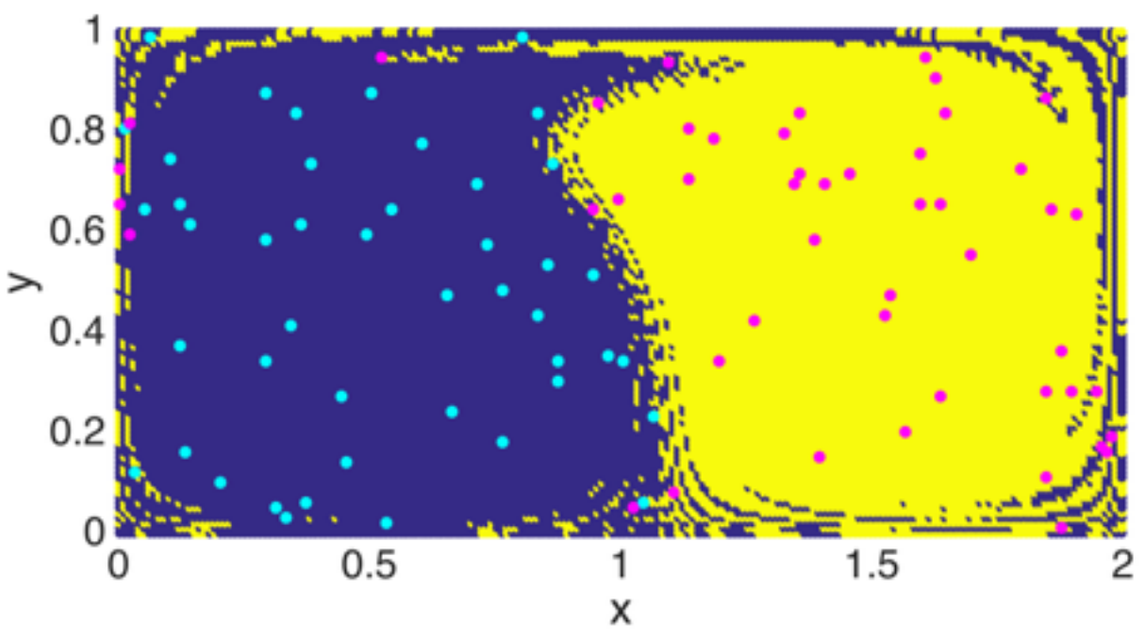
eigenvalues of $\mathbf{L}_\varepsilon = \varepsilon^{-1}(\mathbf{Q}_\varepsilon - \mathbf{I})$



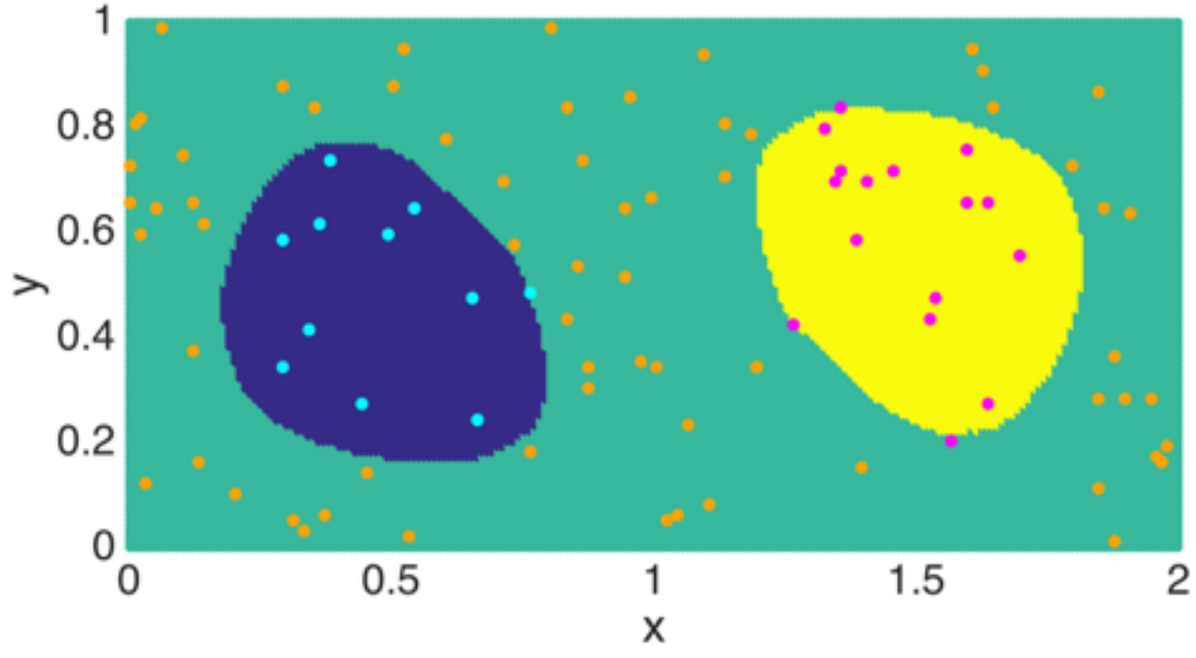
Double gyre: sparse data

We randomly select 500 of the 20.000 trajectories, and for those we destroy 80% of the entries. 97.5% of the data is destroyed!

2-clustering



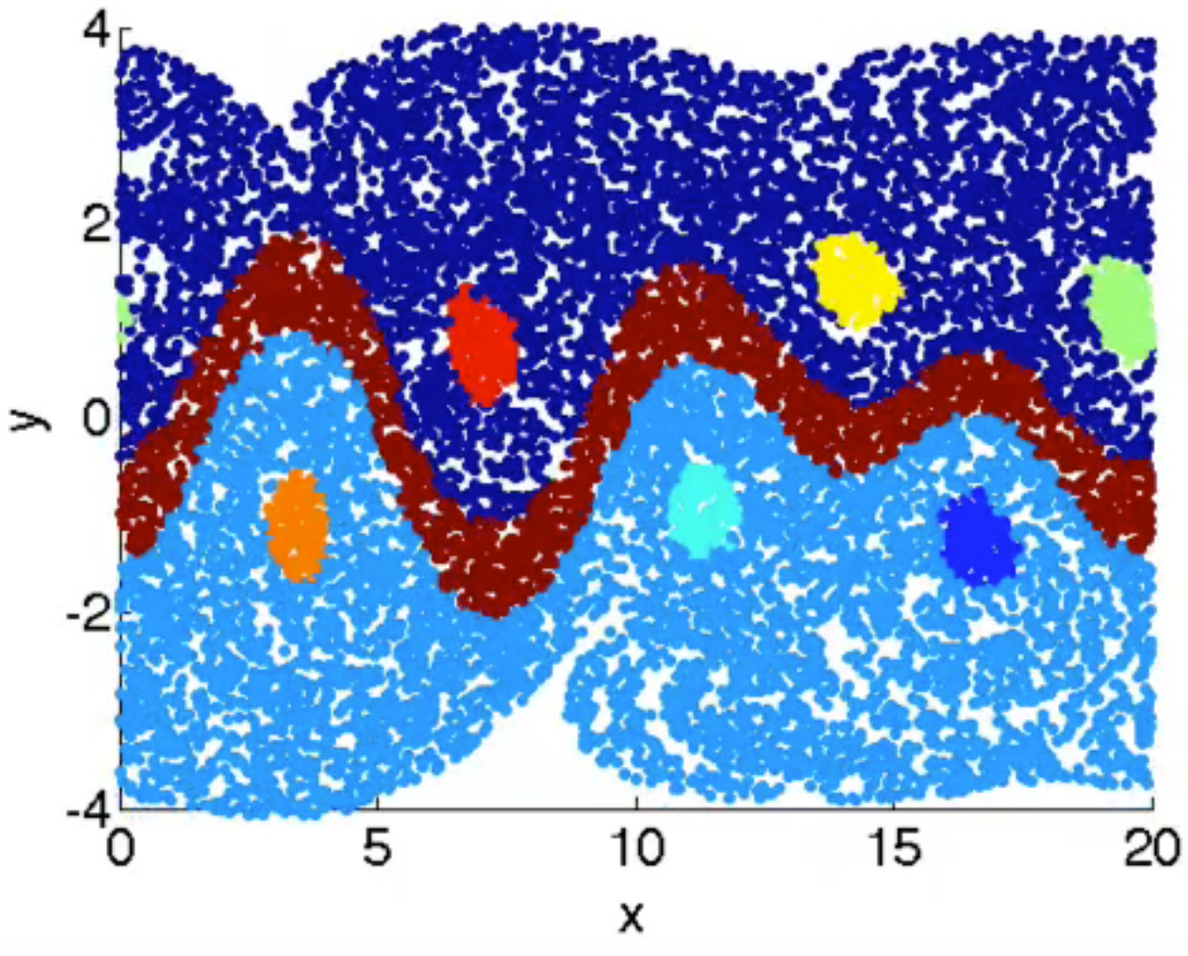
3-clustering



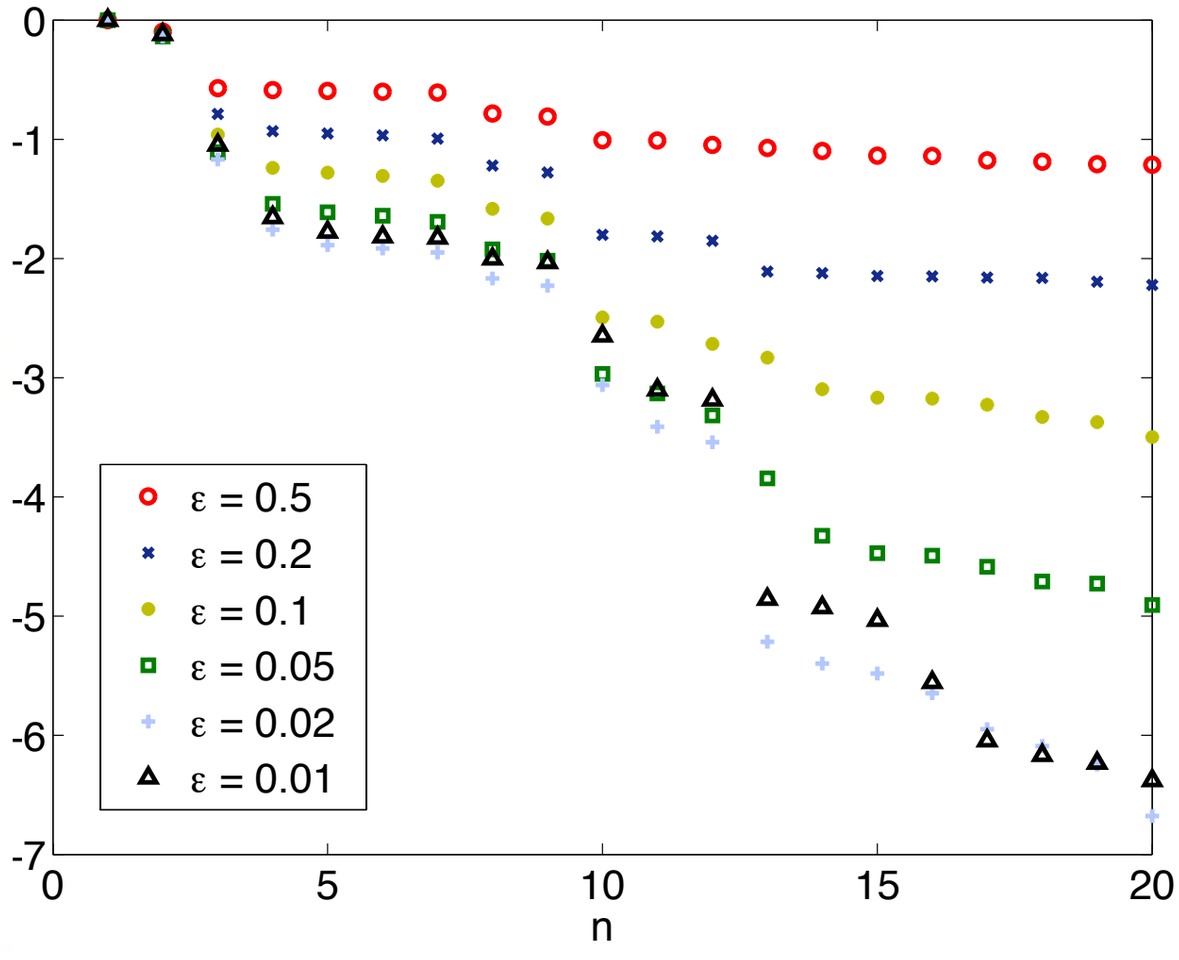
Quasiperiodic Bickley jet: Eigenvalues [RB & Koltai 2016]

12.000 trajectories, flow time t=40 days.

9 clusters



eigenvalues of $\mathbf{L}_\varepsilon = \varepsilon^{-1}(\mathbf{Q}_\varepsilon - \mathbf{I})$



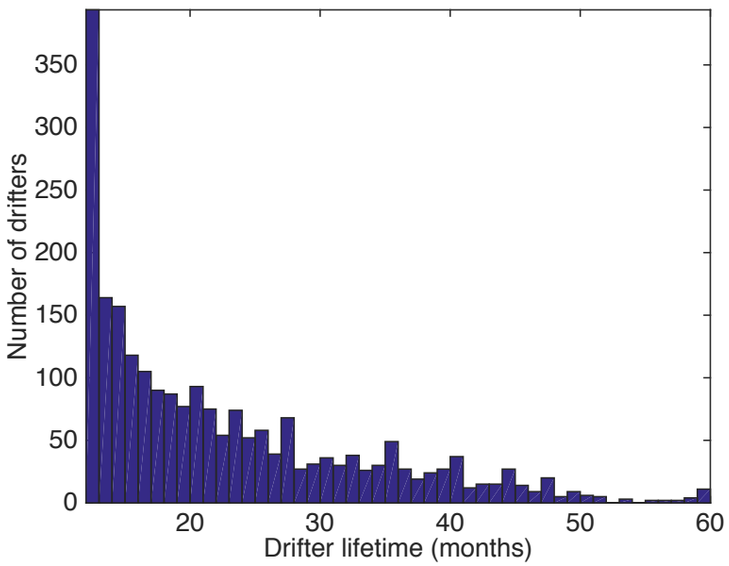
ralfbanisch.shinyapps.io/shiny_bickley/

Ocean drifter data

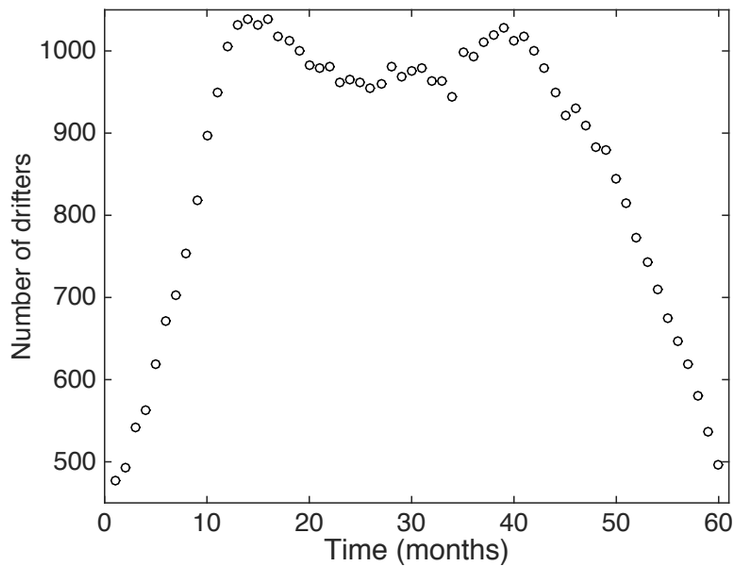
[Froyland & Padberg-Gehle 2015]

Data: 2.267 drifters in the global ocean, with monthly positions from Jan 2005 - Dec 2009.

This data is sparse: On average only 38% of all trajectories available at any time instant.

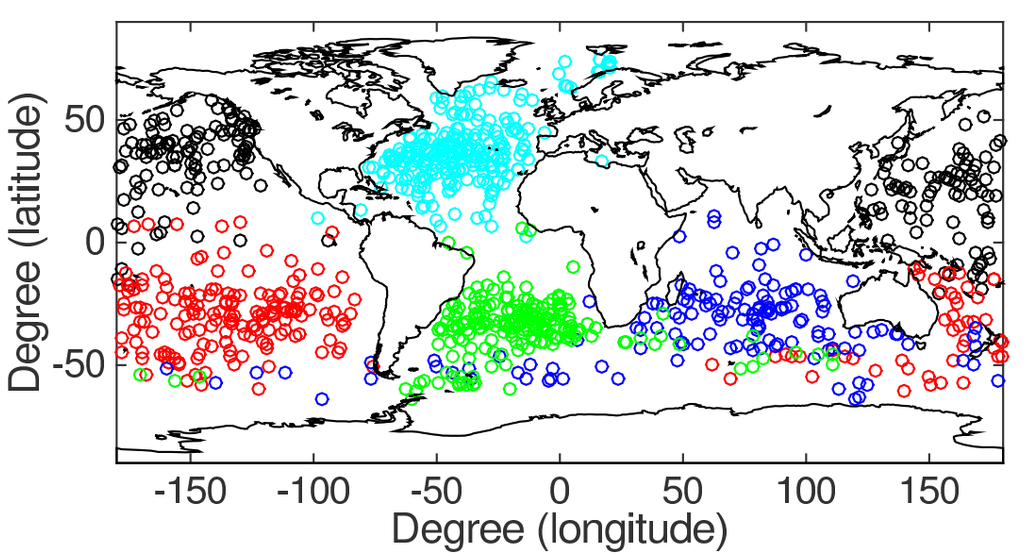


(a)



(b)

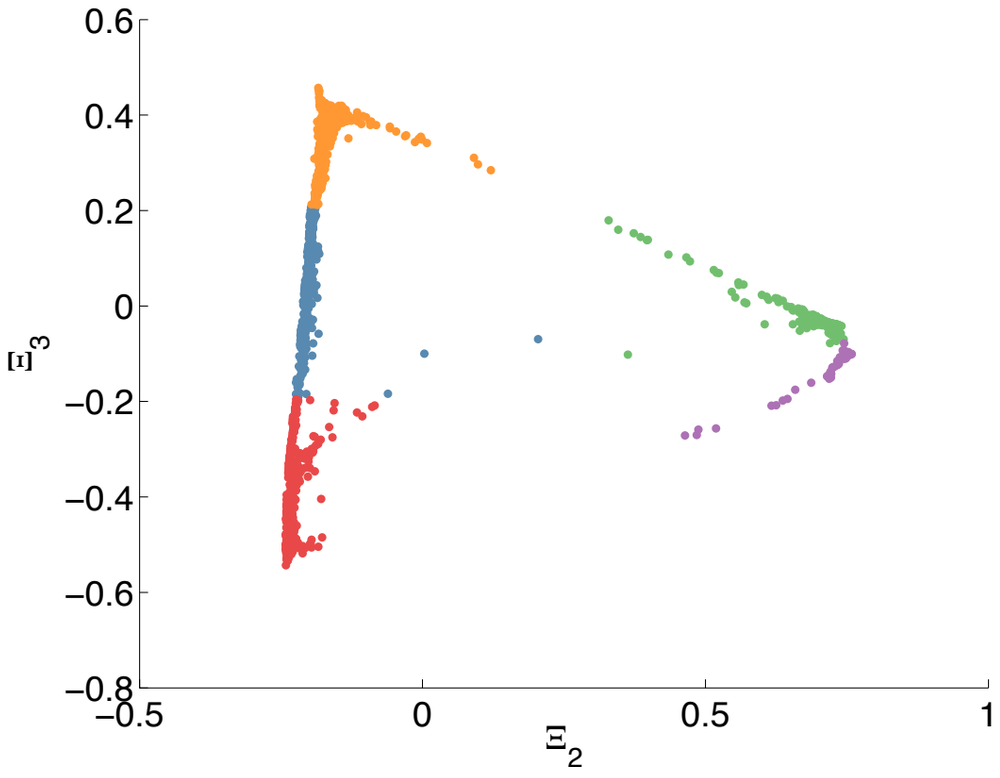
FPG clustering



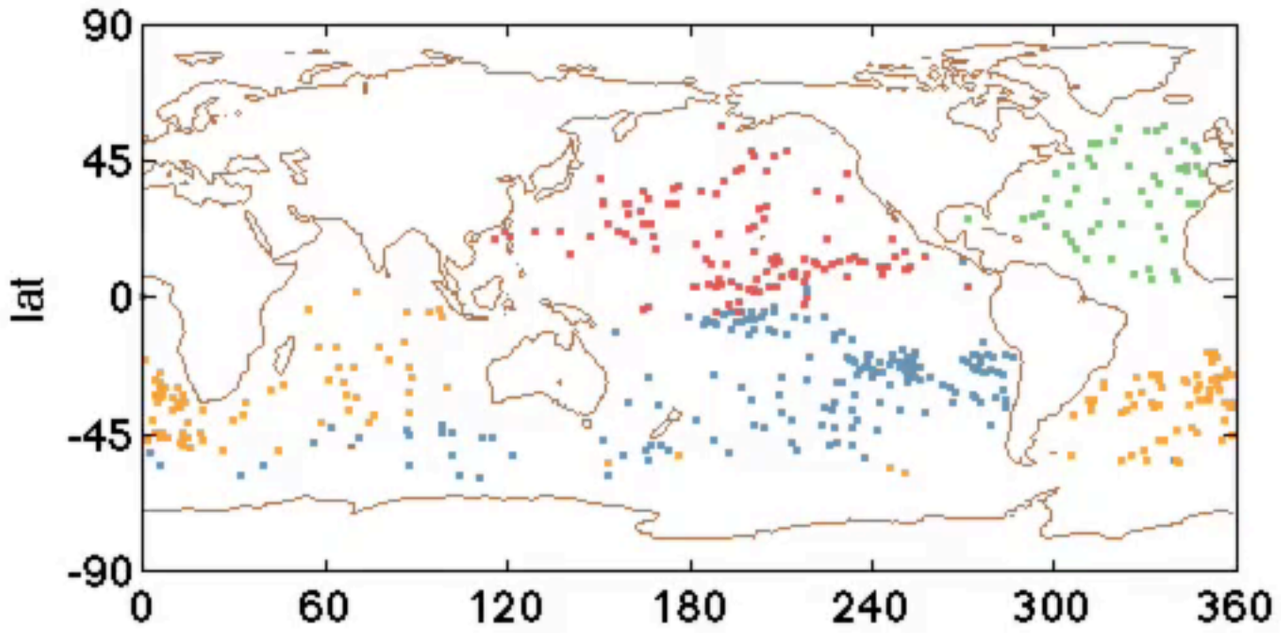
Ocean drifter data

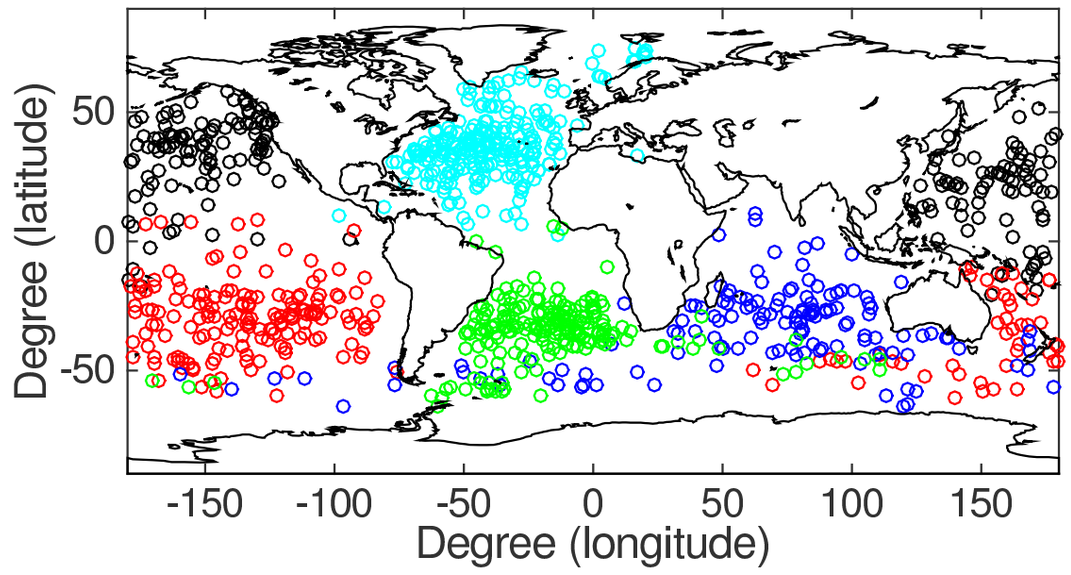
[RB & Koltai 2016, Froyland & Padberg-Gehle 2015]

embedding



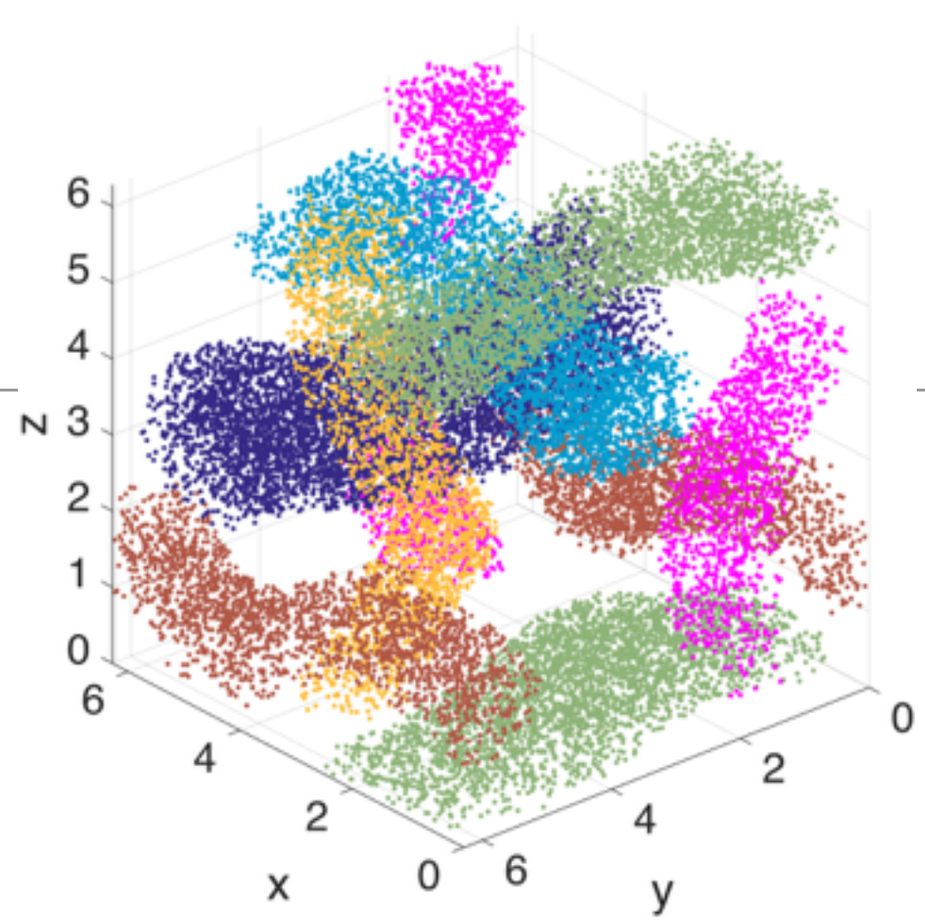
5-clustering



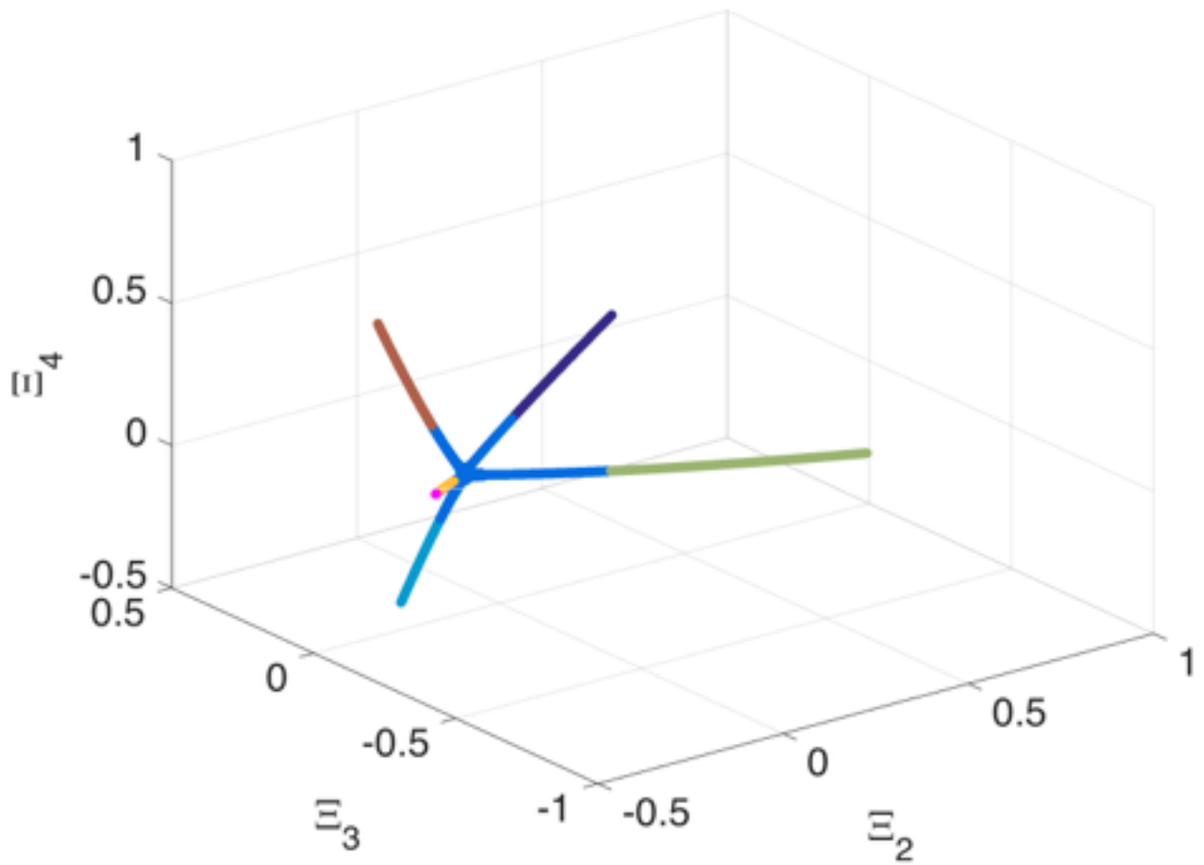


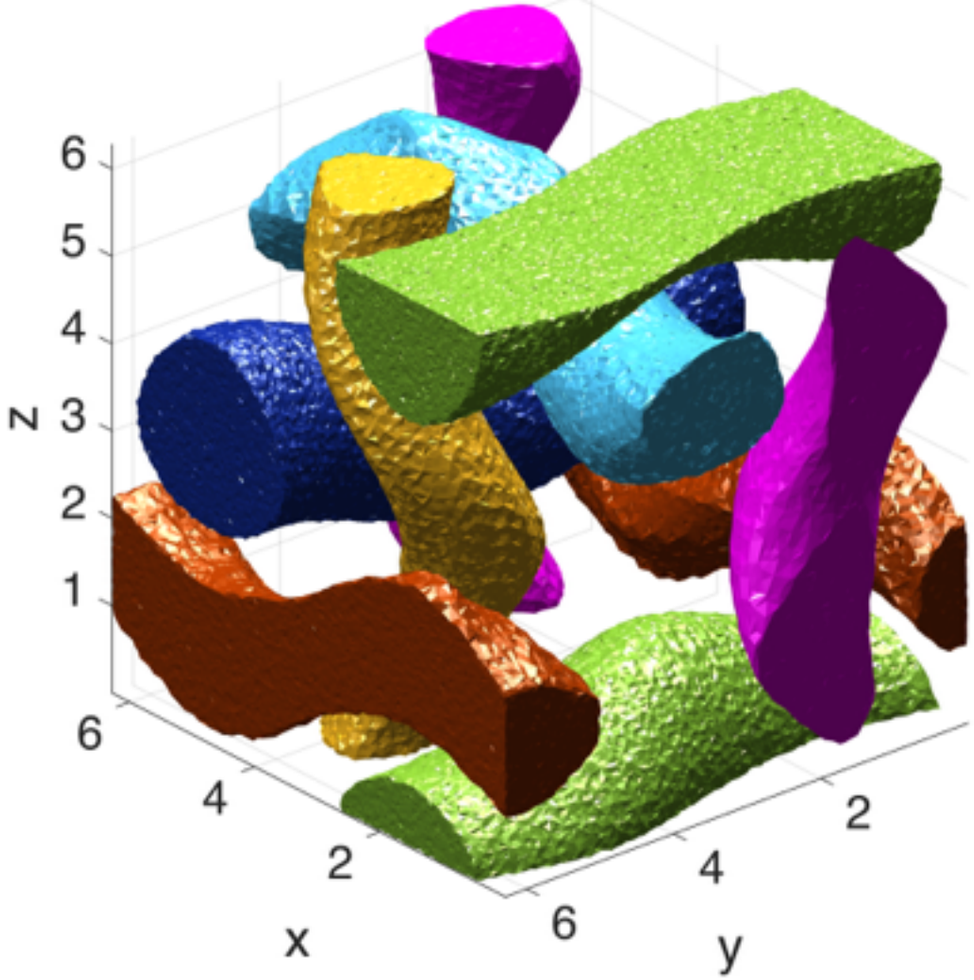
ABC flow [RB & Koltai 2016]

64.000 trajectories, flow time t=20.

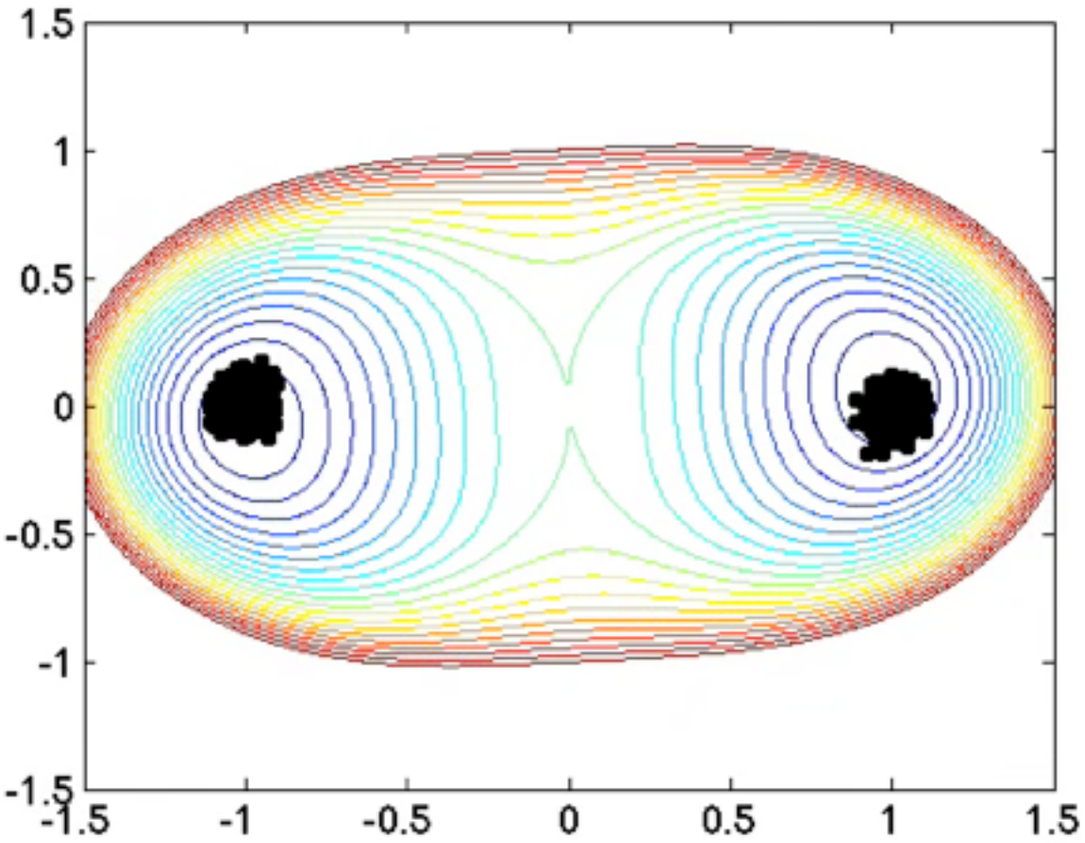


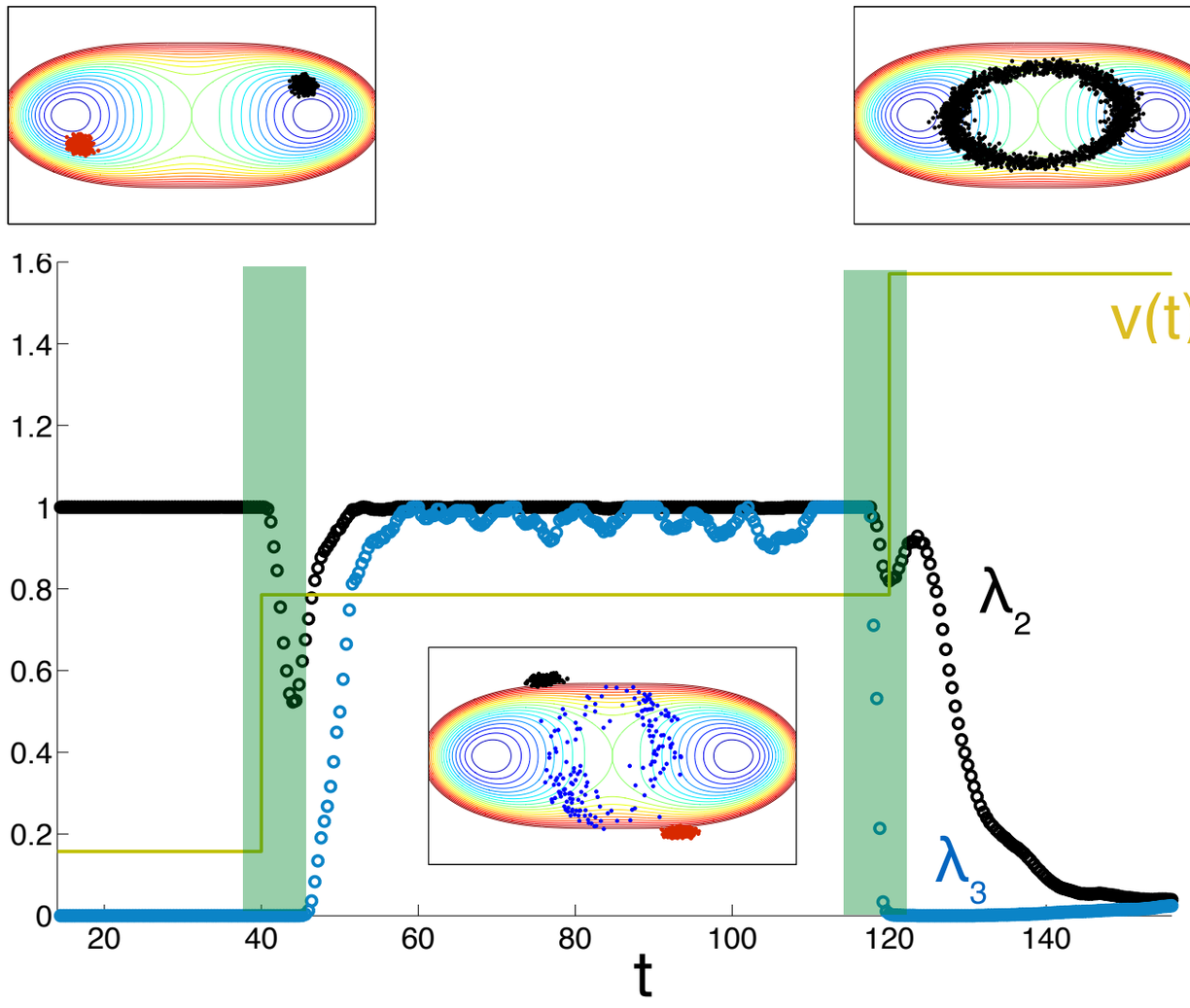
embedding



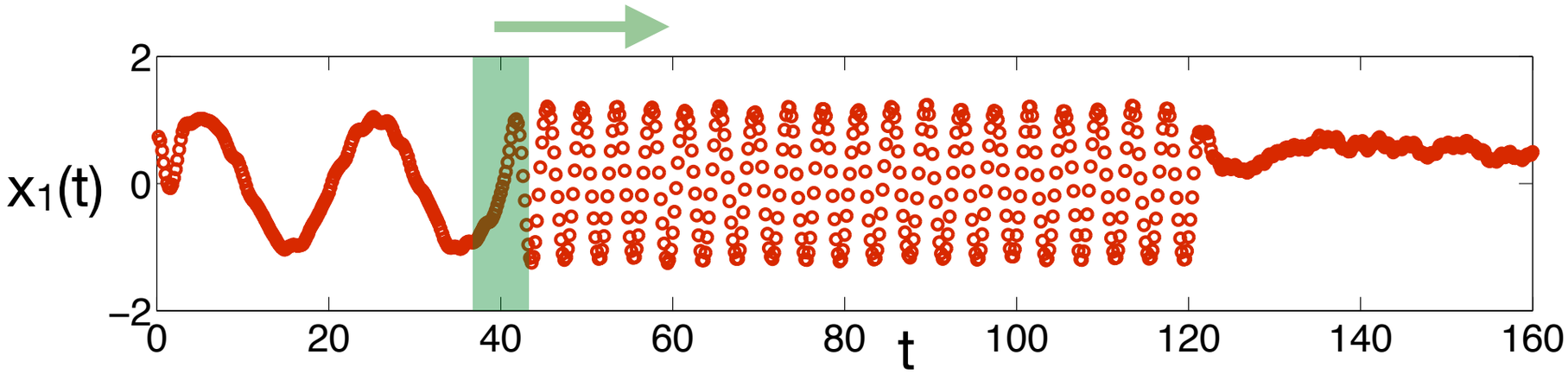


Change point detection

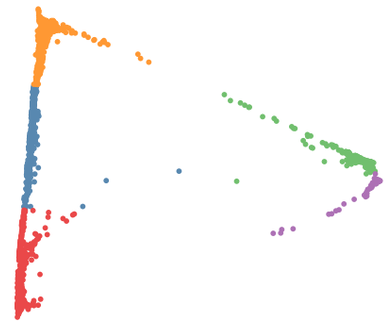




sliding window analysis:



Summary



- it is possible to define **dynamical distances** which capture essential dynamical features of interest: **slow dynamical modes, coherence,…**

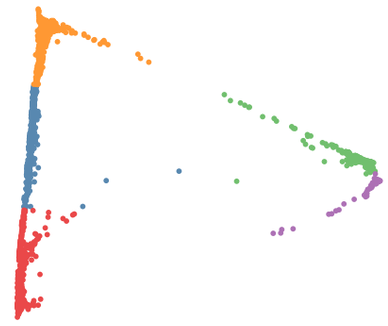
- there is always a connection to an analytical transfer operator framework in terms of dominant eigenfunctions and eigenvalues.

- direct numerical estimation from trajectory data possible.

- the dynamical distances define a mapping to a low-dimensional space in which geometry-based machine learning methods can be applied.

- the resulting reduced-order kinetic models have superior accuracy.

Summary



- it is possible to define **dynamical distances** which capture essential dynamical features of interest: **slow dynamical modes, coherence,…**

- there is always a connection to an analytical transfer operator framework in terms of dominant eigenfunctions and eigenvalues.

- direct numerical estimation from trajectory data possible.

- the dynamical distances define a mapping to a low-dimensional space in which geometry-based machine learning methods can be applied.

- the resulting reduced-order kinetic models have superior accuracy.

useful links:

- papers: arxiv.org/abs/1603.04709 and DOI:10.1021/acs.jctc.6b00762

- app: ralfbanisch.shinyapps.io/shiny_bickley/
  and https://github.com/ralfbanisch/shiny-diffusion-maps

- PyEMMA analysis toolkit: http://pyemma.org