Forecasting without a model and with an imperfect model

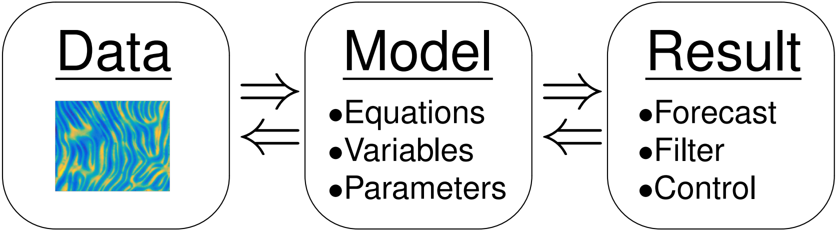
Tyrus Berry

George Mason University

March 23, 2018

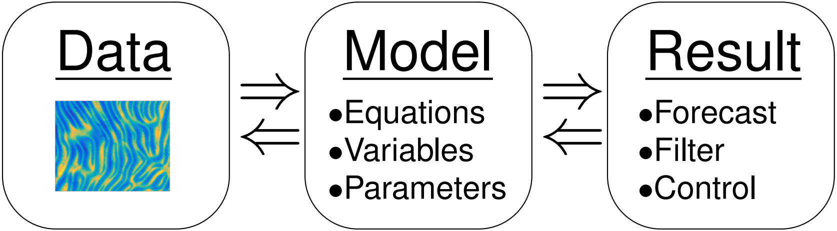
Joint work with John Harlim, PSU and Dimitris Giannakis, NYU

PARAMETRIC MODELING



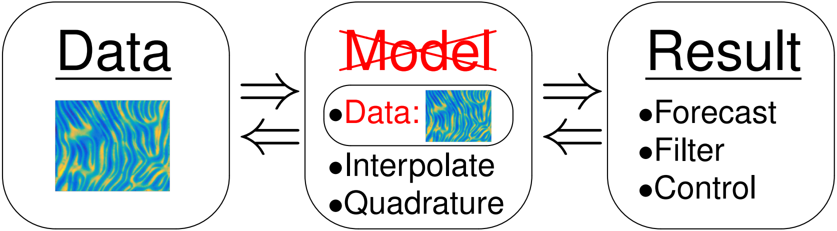
- **Design Model:** Limited resolution and complexity
- **Assimilate Data:** Fit Parameters/Variables
  - Kalman Filter, EKF, EnKF, Variational methods
  - Observability and noise
  - Model error
- **Study/Apply:** Ensemble Forecast

PARAMETRIC MODELING



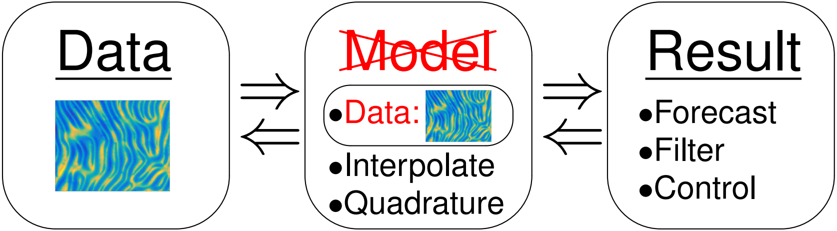
- **Model error:**
  - Trade off resolution and complexity
  - Stationarity/Homogeneity of parameters
- **Assimilate Data:** Fit Parameters/Variables
  - Lumps together noise and model error

NONPARAMETRIC MODELING



- **Data IS the model:**
  - Assume a model exists
    - Data lies on/near an unknown sub-manifold
    - Data obeys an unknown dynamical system
  - Represent the model using training data

NONPARAMETRIC MODELING



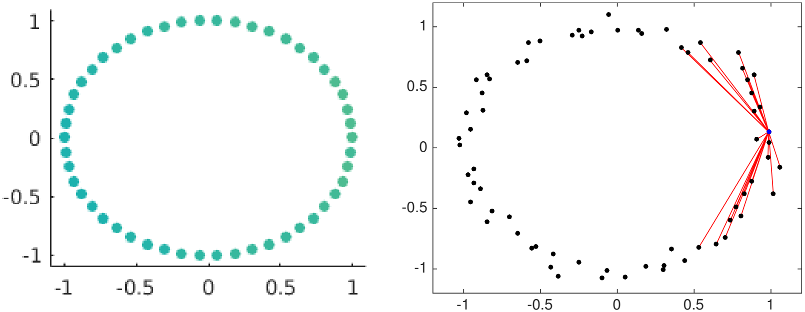
- **Tools:** Distributions $f \in \mathcal{H}$
  - Interpolation: $f(x) = \sum_j \langle f, \varphi_j \rangle \varphi_j(x)$
  - Quadrature: $\langle f, \varphi_i \rangle \approx \sum_i f(x_i)\varphi(x_i)$
  - Operator Representation: $\mathbf{A}_{jk} = \langle \varphi_j, \mathcal{A}\varphi_k \rangle$
- All require a basis $\{\varphi_j\}$!

ROADMAP: CORRECTING MODEL ERROR

- What is manifold learning? $\Rightarrow$ Custom Fourier Basis
- Nonparametric methods (no model)
  - Diffusion Forecast
- Semiparametric methods (model error)

WHAT IS MANIFOLD LEARNING?

- **Geometric prior:** Data lie on/near a manifold $\mathcal{M} \subset \mathbb{R}^m$
- **Goal:** Represent all the information about a manifold
- **Example:** Data on or near the unit circle:



HOW DO WE REPRESENT A MANIFOLD?

- Geometric prior: Data lie on/near a manifold $\mathcal{M} \subset \mathbb{R}^m$
- Every manifold has a Laplacian operator:
- Euclidean space: $\Delta f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2}$
- Unit circle: $\Delta f = \frac{d^2 f}{d\theta^2}$
- **Intrinsic:** In geodesic coordinates $\Delta f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial s_i^2}$

Why the Laplacian?

- **Manifold learning $\Leftrightarrow$ Estimating Laplace-Beltrami**
- Euclidean space:
  - Eigenfunctions of Δ are $e^{i\vec{\omega}\cdot\vec{x}}$
  - Basis for Fourier transform
- Unit circle:
  - Eigenfunctions of Δ are $e^{ik\theta}$
  - Basis for Fourier series
- **Theorem**: Eigenfunctions of Δ form a basis for square integrable functions (Hilbert space $L^2(\mathcal{M})$).

Why the Laplacian?

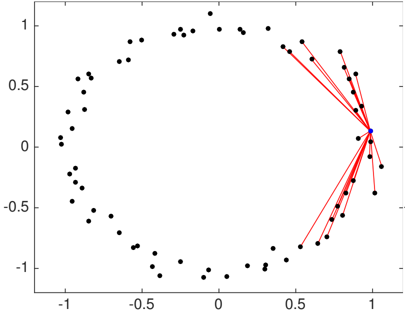
- **Manifold learning $\Leftrightarrow$ Estimating Laplace-Beltrami**
- Eigenfunctions $\Delta\varphi_i = \lambda_i\varphi_i$ orthonormal basis for $L^2(\mathcal{M})$
- Smoothest functions: φ_i minimizes the functional

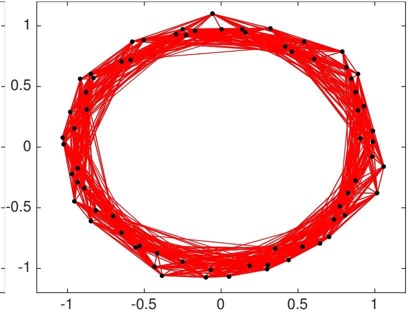
$$\lambda_i = \min_{\substack{f\perp\varphi_k \\ k=1,\dots,i-1}} \left\{ \frac{\int_{\mathcal{M}} ||\nabla f||^2 \, dV}{\int_{\mathcal{M}} |f|^2 \, dV} \right\}$$

- Eigenfunctions of Δ are custom Fourier basis
  - Smoothest orthonormal basis for $L^2(\mathcal{M})$
  - Can be used to define wavelets
  - Define the Hilbert/Sobolev spaces on $\mathcal{M}$

SO HOW DO WE FIND THE LAPLACIAN FROM DATA?

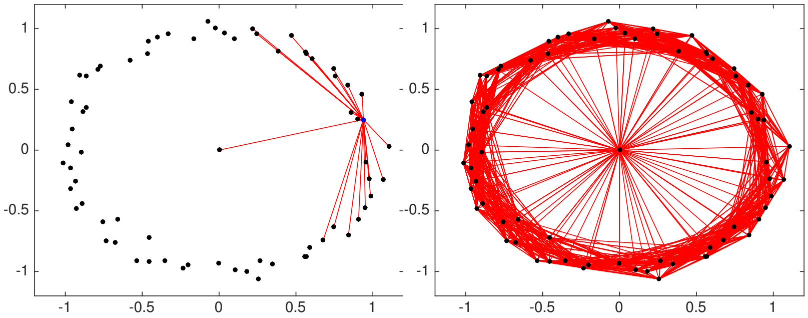
- Assume data lies on (or at least near) a manifold
- Approximate manifold with graph $\Rightarrow$ Connect nearby points





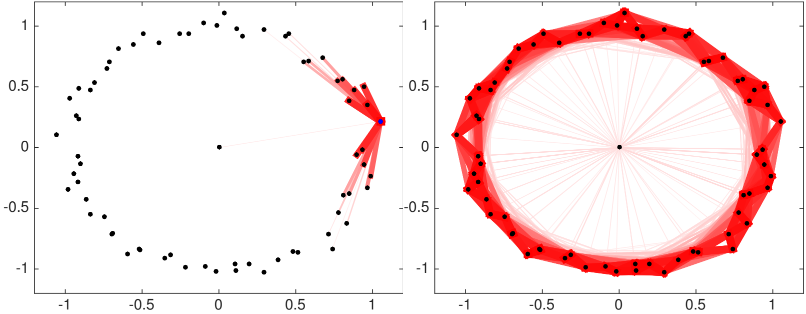
SO HOW DO WE FIND THE LAPLACIAN FROM DATA?

- **Problem:** Noise and outliers can lead to *bridging*



SO HOW DO WE FIND THE LAPLACIAN FROM DATA?

- To prevent bridging we weight the edges
- Edges are given weights $K_\delta(x,y) = e^{-\frac{||x-y||^2}{4\delta^2}}$



SO HOW DO WE FIND THE LAPLACIAN FROM DATA?

- Data set $\Rightarrow$ *weighted graph*
- Edge Weights defined by a kernel function

$$K_\delta(x_i, x_j) = e^{-\frac{||x_i - x_j||^2}{4\delta^2}}$$

- Bandwidth δ determines localization
- 'Adjacency' matrix: $\mathbf{K}_{ij} = K(x_i, x_j)$
- 'Degree' matrix: $\mathbf{D}_{ii} = \sum_j \mathbf{K}_{ij}$
- Normalized graph Laplacian: $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{K}$

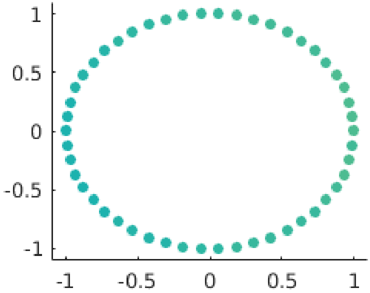
SO HOW DO WE FIND THE LAPLACIAN FROM DATA?

- Data set $\Rightarrow$ *weighted graph*
- Normalized graph Laplacian: $\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{K}$
- **Theorem:** In the limit of $\delta \to 0$ and $N \to \infty$ we have

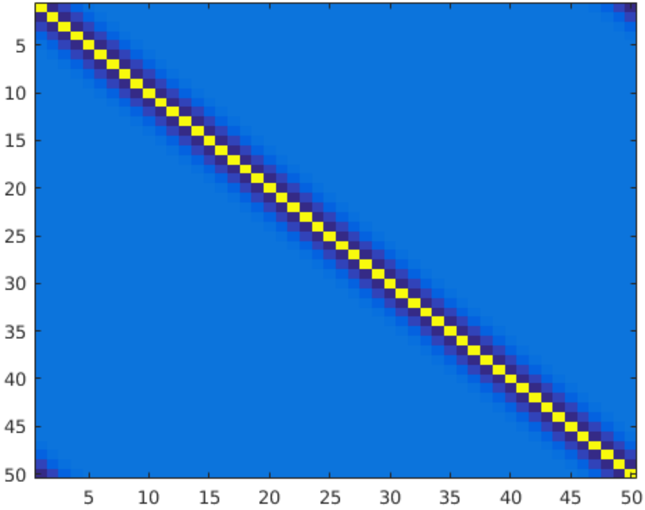
$$L \to \Delta$$

- Eigenvectors $\vec{\phi}$ of L converge to eigenfunctions φ of Δ

EXAMPLE: 50 DATA POINTS ON S^1

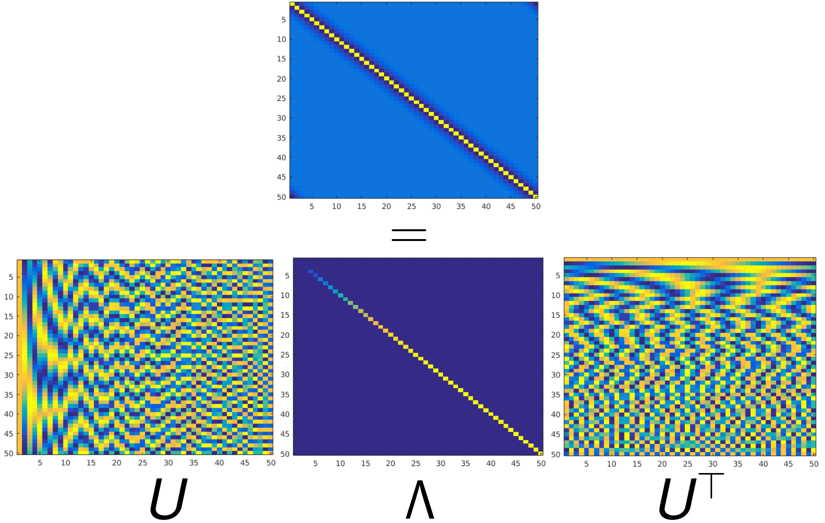


EXAMPLE: L MATRIX FOR S^1

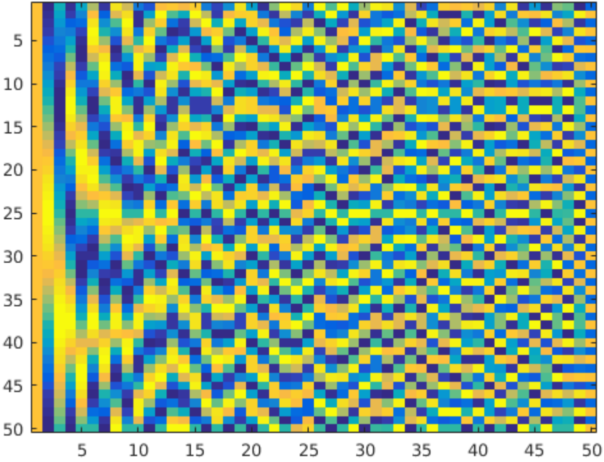


Example S^1: Eigenvector Decomposition

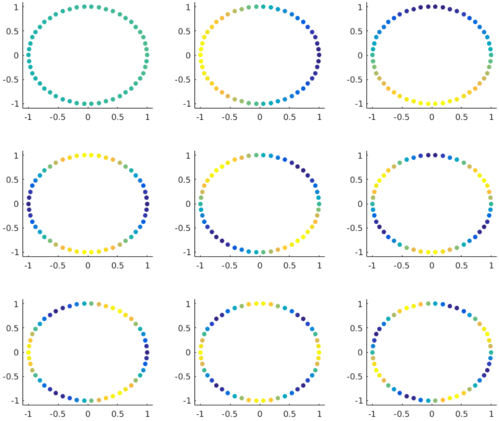
$$L = U \Lambda U^{\top}$$



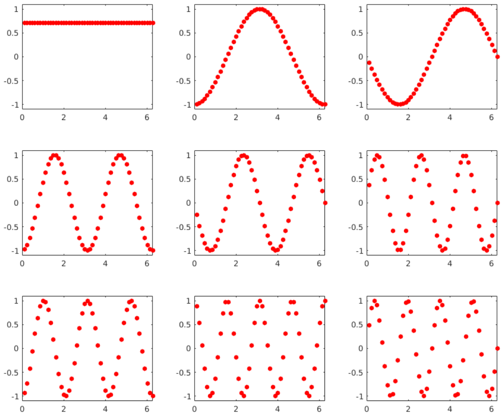
EXAMPLE S^1: MATRIX OF EIGENVECTORS, U



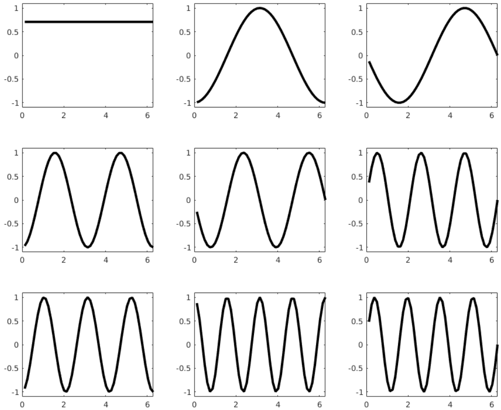
EXAMPLE S^1: EIGENVECTORS ON DATA



Example S^1: Eigenvectors vs. θ



EXAMPLE S^1: CONNECTING THE DOTS

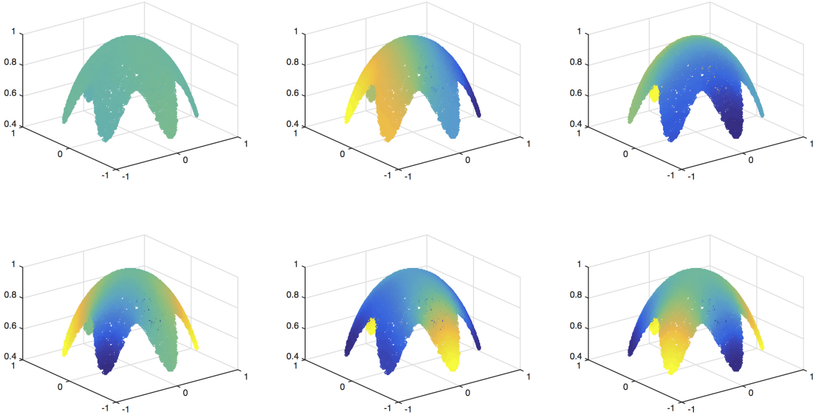


DISCRETE ANALOGS OF CONTINUOUS OBJECTS

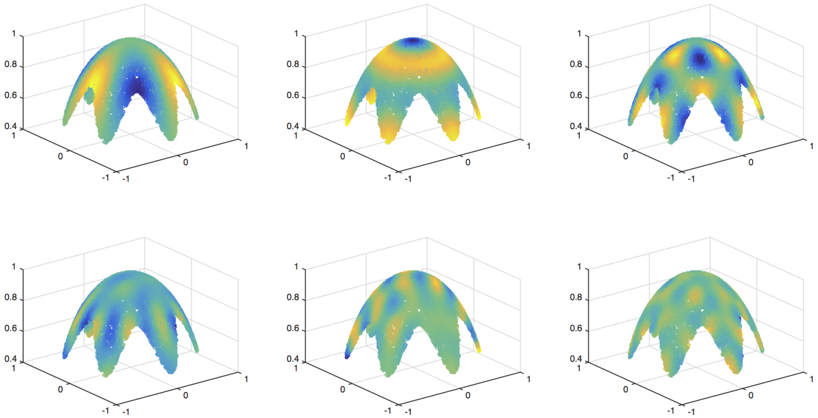
Continuous, $\mathcal{M}$	Discrete, $\{x_i\}_{i=1}^N$
$L^2(\mathcal{M})$	$\mathbb{R}^N$
Functions, $f : \mathcal{M} \to \mathbb{R}$	Vectors, $\vec{f}_i = f(x_i)$
'Basis', δ_x	Basis, $\vec{e}_i = \delta_{x_i}$
Laplace-Beltrami, Δ	Normalized Graph Laplacian, $\mathbf{L}$
Eigenfunctions, $\Delta\varphi_j = \lambda_j\varphi_j$	Eigenvectors, $\mathbf{L}\vec{\varphi}_j = \lambda_j\vec{\varphi}_j$
Inner product, $\langle f, h\rangle_{L^2}$	Dot Product, $\frac{1}{N}\vec{f} \cdot \vec{h}$

$$\frac{1}{N}\vec{f} \cdot \vec{h} = \frac{1}{N}\sum_{i=1}^{N} f(x_i)h(x_i) \to_{N\to\infty} \int_{\mathcal{M}} f(x)h(x)\, dV(x)$$

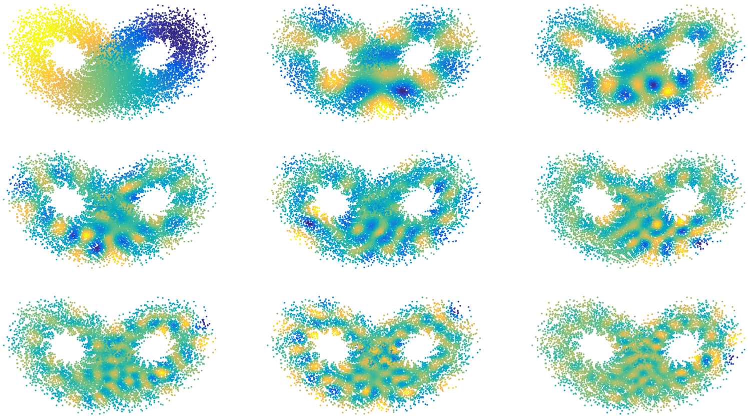
Harmonic Analysis on Manifolds/Data Sets



Harmonic Analysis on Manifolds/Data Sets



HARMONIC ANALYSIS ON MANIFOLDS/DATA SETS



Diffusion Forecast

- Autonomous SDE: $dx = a(x)\,dt + b(x)\,dW_t$
- Density solves Fokker-Planck PDE: $\frac{\partial}{\partial t} p = \mathcal{L}^* p$
- Project onto custom Fourier basis (spectral method)

$$\begin{array}{ccc}
p(x,t) & \xrightarrow{\text{Diffusion Forecast}} & p(x,t+\tau) = e^{\tau\mathcal{L}^*} p(x,t) \\
\big\downarrow \langle p, \varphi_j \rangle & & \big\uparrow \sum_j c_j \varphi_j q \\
\vec{c}(t) & \xrightarrow{A_{lj} \equiv \mathbb{E}[\langle \varphi_j, S\varphi_l \rangle_{p_{\mathrm{eq}}}]} & \vec{c}(t+\tau) = A\vec{c}(t).
\end{array}$$

The Shift Map

- Autonomous SDE: $dx = a(x)\,dt + b(x)\,dW_t$
- Density solves Fokker-Planck PDE: $\frac{\partial}{\partial t}p = \mathcal{L}^* p$
- Given data $x_i = x(t_i)$ with $\tau = t_{i+1} - t_i$
- Using the Itô lemma we can show:

$$Sf(x_i) = f(x_{i+1}) = e^{\tau\mathcal{L}} f(x_i) + \int_{t_i}^{t_{i+1}} \nabla f^\top b\, dW_s + \int_{t_i}^{t_{i+1}} Bf\, ds$$

- Feynman-Kac: $\mathbb{E}[S(f)] = e^{\tau\mathcal{L}} f$
- Shift map estimates the solution of the Fokker-Planck PDE!

Representing the Shift Map

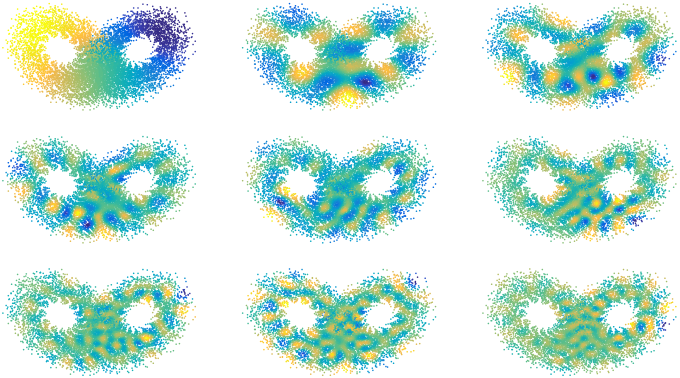
- Choose a basis $\{\varphi_j\}$ orthonormal with respect to $\langle \cdot, \cdot \rangle_{p_{\text{eq}}}$
- Evolution of Fourier coefficients $c_l(t) = \langle p(x,t), \varphi_l \rangle$

$$c_l(t+\tau) = \langle p(x, t+\tau), \varphi_l \rangle = \left\langle e^{\tau \mathcal{L}^*} p(x,t), \varphi_l \right\rangle = \left\langle p(x,t), e^{\tau \mathcal{L}} \varphi_l \right\rangle$$
$$= \sum_j c_j(t) \left\langle \varphi_j, e^{\tau \mathcal{L}} \varphi_l \right\rangle_{p_{\text{eq}}} = \sum_j A_{lj} c_j(t)$$

- So $\vec{c}(t+\tau) = A\vec{c}(t)$
- Where $A_{lj} = \left\langle \varphi_j, e^{\tau \mathcal{L}} \varphi_l \right\rangle_{p_{\text{eq}}}$

MANIFOLD LEARNING ⇒ CUSTOM 'FOURIER' BASIS

- **Optimal basis:** Minimum variance $A_{lj} \equiv \mathbb{E}[\langle \varphi_j, S\varphi_l \rangle_{p_{\mathrm{eq}}}]$



Forecast Operator is Linear

$$p(x,t) \xrightarrow{\text{Diffusion Forecast}} p(x,t+\tau)$$

$$\Big\downarrow \langle p, \varphi_j \rangle \qquad\qquad \Big\uparrow \textstyle\sum_j c_j \varphi_j p_{\mathrm{eq}}$$

$$\vec{c}(t) \xrightarrow{A_{lj} \equiv \mathbb{E}[\langle \varphi_j, S\varphi_l \rangle_{p_{\mathrm{eq}}}]} \vec{c}(t+\tau) = A\vec{c}(t)$$

- Shift Map: $S(\varphi_l)(x_i) = \varphi_l(x_{i+1})$
- Forecast Operator: $A_{lj} = \overline{\varphi_j(x_i)\varphi_l(x_{i+1})}$
- **Theorem:** Eigenfunctions of Δ optimal estimates of $e^{\tau\mathcal{L}}$

DIFFUSION FORECAST LORENZ-63 EXAMPLE

(Loading Video...)

NONPARAMETRIC FORECAST ON A TORUS

- Stochastic dynamics on a torus $(\theta, \phi) \in [0, 2\pi)^2$

$$d(\theta, \phi)^\top = a(\theta, \phi)\, dt + b(\theta, \phi)\, dW_t$$

- Drift and diffusion coefficients,

$$a(\theta, \phi) = \begin{pmatrix} \frac{1}{2} + \frac{1}{8}\cos(\theta)\cos(2\phi) + \frac{1}{2}\cos(\theta + \pi/2) \\ 10 + \frac{1}{2}\cos(\theta + \phi/2) + \cos(\theta + \pi/2)) \end{pmatrix},$$

$$b(\theta, \phi) = \begin{pmatrix} \frac{1}{4} + \frac{1}{4}\sin(\theta) & \frac{1}{4}\cos(\theta + \phi) \\ \frac{1}{4}\cos(\theta + \phi) & \frac{1}{40} + \frac{1}{40}\sin(\phi)\cos(\theta) \end{pmatrix}.$$

DIFFUSION FORECAST TORUS EXAMPLE

(Loading Video...)

PROBLEM: CURSE OF DIMENSIONALITY

- Learning the basis → Data exponential in manifold dim
- Monte-Carlo type estimates $\mathcal{O}(N^{-1/2})$:
  - Coefficients:

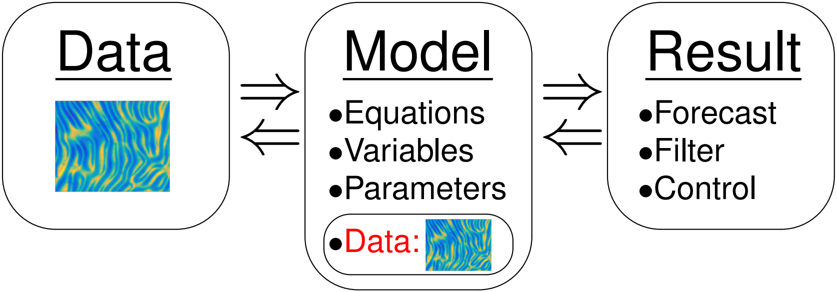
$$c_l(t) = \langle p(x,t), \varphi_l \rangle \approx \frac{1}{N} \sum_{i=1}^{N} \varphi_l(x_i) p(x_i,t)/p_{\text{eq}}(x_i)$$

  - Markov Matrix:

$$A_{lj} = \left\langle \varphi_j, e^{\tau \mathcal{L}} \varphi_l \right\rangle_{p_{\text{eq}}} \approx \frac{1}{N} \sum_{i=1}^{N} \varphi_j(x_i) \varphi_l(x_{i+1})$$

- Maybe we shouldn't throw out the model...
- Use diffusion forecast to fix model error!

Semiparametric Modeling



- **Data becomes part of the model:**
  - Start with imperfect parametric model
  - Fit training data with time-varying parameters
  - Query data as part of running model
- **Compensate for model error:**
  - Truncated resolution and complexity
  - Non-analytic expressions
  - Non-stationarity/Inhomogeneity

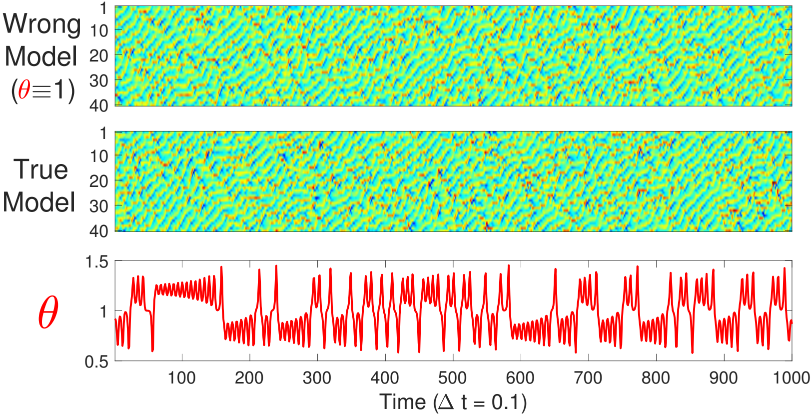
Semiparametric Forecast Model

- Partially known model $\dot{x} = f(x, \theta)$
- Unknown: $d\theta = a(\theta)\, dt + b(\theta)\, dW_t$
- Apply the Diffusion Forecast to $p(\theta, t)$
- Sample $\theta^k(t) \sim p(\theta, t)$ and pair with ensemble $x^k(t)$

$$
\begin{array}{ccc}
(x^k(t), \theta^k(t)) & \xrightarrow{\dot{x}=f(x,\theta)} & (x^k(t+\tau), \theta^k(t+\tau)) \\
\uparrow \theta^k(t) & & \uparrow \theta^k(t+\tau) \\
p(\theta, t) & \overset{\text{Diffusion Forecast}}{\dashrightarrow} & p(\theta, t+\tau)
\end{array}
$$

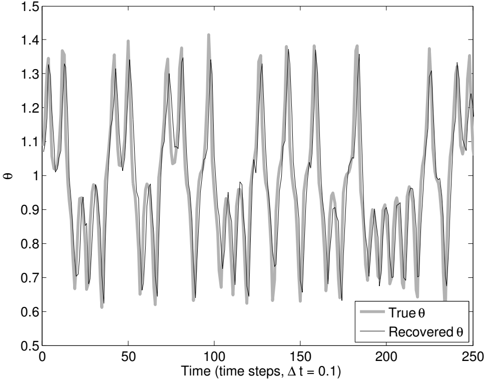
Example: 40-dimensional Lorenz-96 system

$$\dot{x}_i = \theta x_{i-1}x_{i+1} - x_{i-1}x_{i-2} - x_i + 8$$



EXAMPLE: 40-DIMENSIONAL LORENZ-96 SYSTEM

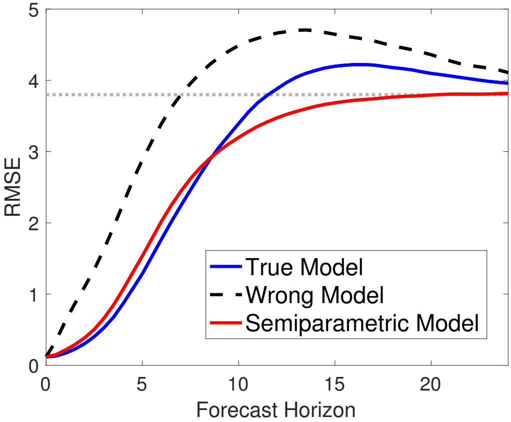
Kalman filter ⇒ Estimate time series of θ (training period)



Using this data, build a diffusion forecast model for θ

Example: 40-dimensional Lorenz-96 system

$$\dot{x}_i = \theta x_{i-1}x_{i+1} - x_{i-1}x_{i-2} - x_i + 8$$



SEMIPARAMETRIC FILTER: PUT IT ALL TOGETHER...

$$
\begin{array}{ccccc}
\begin{pmatrix} x^{k,a}(t-\tau) \\ \theta^{k,a}(t-\tau) \end{pmatrix} & \xrightarrow{\dot{x}=f(x,\theta)} & \begin{pmatrix} x^{k,f}(t) \\ \theta^{k,f}(t) \end{pmatrix} & \xrightarrow{\text{EnKF } y^o(t)} & \begin{pmatrix} x^{k,a}(t) \\ \theta^{k,a}(t) \end{pmatrix} \\
\Big\downarrow \theta^a & & \Big\uparrow \theta^{k,f}(t) & & p(\theta^a(t) \mid \theta(t)) \Big\downarrow \\
p^a(\theta, t-\tau) & \overset{\text{Diffusion Forecast}}{- - - - - \rightarrow} & p^f(\theta, t) & \xrightarrow{p^f(\theta) p(y \mid \theta)} & p^a(\theta, t) \\
\Big\downarrow \langle p^a, \varphi_j \rangle & & \Big\uparrow \sum_j c_j^f \varphi_j p_{\mathrm{eq}} & & \langle p^a, \varphi_j \rangle \Big\downarrow \\
\vec{c}^{\,a}(t-\tau) & \xrightarrow{A_{lj} c^a(t-\tau)} & \vec{c}^{\,f}(t) & \overset{\text{Bayesian Update}}{- - - - - \rightarrow} & \vec{c}^{\,a}(t)
\end{array}
$$

For more information: http://math.gmu.edu/~berry/

Building the basis

- Coifman and Lafon, *Diffusion maps.*
- B. and Harlim, *Variable Bandwidth Diffusion Kernels.*
- B. and Sauer, *Local Kernels and Geometric Structure of Data.*

Nonparametric forecast

- B., Giannakis, and Harlim, *Nonparametric forecasting of low-dimensional dynamical systems.*
- B. and Harlim, *Forecasting Turbulent Modes with Nonparametric Diffusion Models.*

Semiparametric forecast

- B. and Harlim, *Semiparametric forecasting and filtering: correcting low-dimensional model error in parametric models.*