Coherent structure identification using flow map composition and spectral interpolation

Clancy Rowley

Joint work with S. Brunton, M. Luchtenburg, and M. Williams

Princeton University

BIRS: Uncovering Transport Barriers in Geophysical Flows
September 23, 2013

Outline

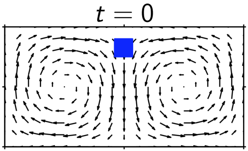
Acknowledgments

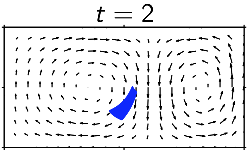
- Steve Brunton (U. Washington)
  - Finite-time Lyapunov exponents
- Mark Luchtenburg (Princeton)
  - Uncertainty quantification
  - Perron-Frobenius
- Matt Williams (Princeton)
  - Koopman eigenfunctions via Dynamic Mode Decomposition

Outline

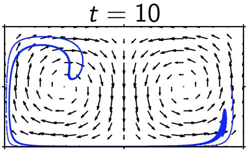
Goal

- Efficient, accurate representation of nonlinear maps
- Example: double gyre





Simple deformation



Complex deformation

Two simple ideas

- Flow map composition
  - Represent a long-time flow map as a composition of short-time flow maps
  - Each short-time flow map should be relatively easy to describe

Two simple ideas

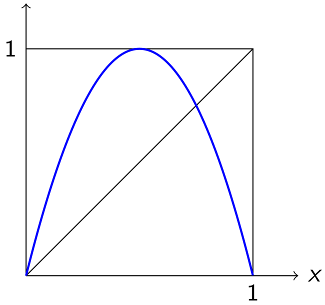
- Flow map composition
  - Represent a long-time flow map as a composition of short-time flow maps
  - Each short-time flow map should be relatively easy to describe
- Spectral interpolation
  - Expand each short-time flow map in terms of orthogonal functions (e.g., Legendre polynomials)
  - Can determine coefficients from values at collocation points

Flow map composition: example

Consider the logistic map

$$x_{k+1} = f(x_k)$$

$$f(x) = 4x(1-x)$$

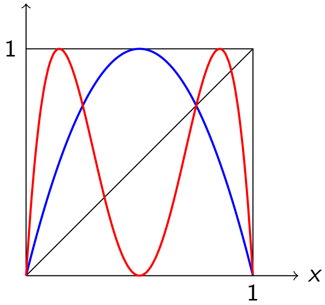


Flow map composition: example

Consider the logistic map

$$x_{k+1} = f(x_k)$$

$$f(x) = 4x(1-x)$$
$$f^2(x) = 16x(1-x)(1-2x)^2$$

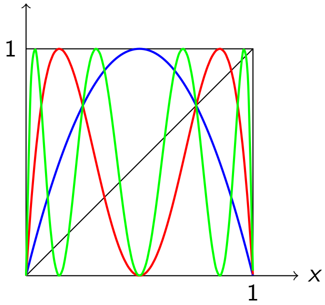


Flow map composition: example

Consider the logistic map

$$x_{k+1} = f(x_k)$$

$$\begin{aligned} f(x) &= 4x(1-x) \\ f^2(x) &= 16x(1-x)(1-2x)^2 \\ f^3(x) &= -2^{14}x^8 + \cdots \end{aligned}$$

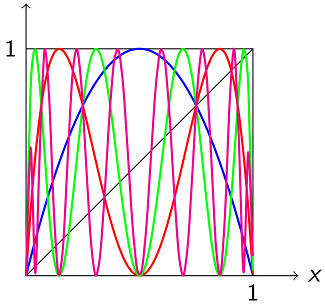


Flow map composition: example

Consider the logistic map

$$x_{k+1} = f(x_k)$$

$$\begin{aligned} f(x) &= 4x(1-x) \\ f^2(x) &= 16x(1-x)(1-2x)^2 \\ f^3(x) &= -2^{14}x^8 + \cdots \\ f^4(x) &= -2^{30}x^{16} + \cdots \end{aligned}$$

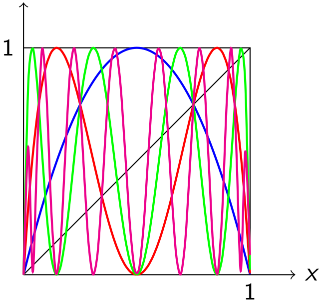


Flow map composition: example

Consider the logistic map

$$x_{k+1} = f(x_k)$$

$$f(x) = 4x(1-x)$$
$$f^2(x) = 16x(1-x)(1-2x)^2$$
$$f^3(x) = -2^{14}x^8 + \cdots$$
$$f^4(x) = -2^{30}x^{16} + \cdots$$
$$f^k(x) = -cx^{2^k} + \cdots$$



- Degree of polynomial increases exponentially in the number of compositions
- Leads to complex long-time map, though short-time map is simple

Representing short-time flow maps

- Short-time flow maps are reasonably "well behaved"
- Represent them with relatively low-order polynomials
- Use orthogonal polynomials
  - Expand flow map ϕ in terms of orthogonal polynomials ψ_j (e.g., Legendre polynomials):

$$\phi(x) = \sum_{j=1}^{n} a_j \psi_j(x) \qquad a_j = \langle \phi, \psi_j \rangle$$

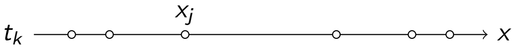
  - Can compute coefficients a_j by evaluating ϕ at collocation points, using Gauss quadrature
  - Simply propagate the collocation points through the flow map to obtain the corresponding coefficients

Representing short-time flow maps

- Short-time flow maps are reasonably "well behaved"
- Represent them with relatively low-order polynomials
- Use orthogonal polynomials
  - Expand flow map ϕ in terms of orthogonal polynomials ψ_j (e.g., Legendre polynomials):

$$\phi(x) = \sum_{j=1}^{n} a_j \psi_j(x) \qquad a_j = \langle \phi, \psi_j \rangle$$

  - Can compute coefficients a_j by evaluating ϕ at collocation points, using Gauss quadrature
  - Simply propagate the collocation points through the flow map to obtain the corresponding coefficients

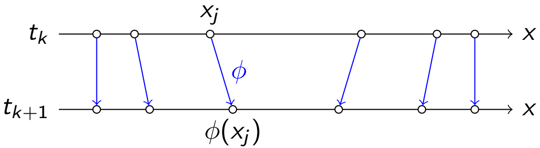


Representing short-time flow maps

- Short-time flow maps are reasonably "well behaved"
- Represent them with relatively low-order polynomials
- Use orthogonal polynomials
  - Expand flow map ϕ in terms of orthogonal polynomials ψ_j (e.g., Legendre polynomials):

$$\phi(x) = \sum_{j=1}^{n} a_j \psi_j(x) \qquad a_j = \langle \phi, \psi_j \rangle$$

  - Can compute coefficients a_j by evaluating ϕ at collocation points, using Gauss quadrature
  - Simply propagate the collocation points through the flow map to obtain the corresponding coefficients



Why flow map composition and spectral interpolation?

- Accurate long-time behavior

Why flow map composition and spectral interpolation?

- Accurate long-time behavior
- Minimal storage needed to represent flow map
  - Degree of the flow map polynomial grows *exponentially* with number of compositions: if short-time flow map is approximated by a degree-p polynomial, after k compositions the degree is p^k
  - For a non-autonomous system, number of parameters grows *linearly* with number of compositions.
  - For an autonomous system, number of parameters is *constant*, independent of number of compositions.

Why flow map composition and spectral interpolation?

- Accurate long-time behavior
- Minimal storage needed to represent flow map
  - Degree of the flow map polynomial grows *exponentially* with number of compositions: if short-time flow map is approximated by a degree-p polynomial, after k compositions the degree is p^k
  - For a non-autonomous system, number of parameters grows *linearly* with number of compositions.
  - For an autonomous system, number of parameters is *constant*, independent of number of compositions.
- Spectral interpolation is accurate and efficient
  - Typically $p+1$ collocation points for a degree-p approximation

Outline

Two simple ideas

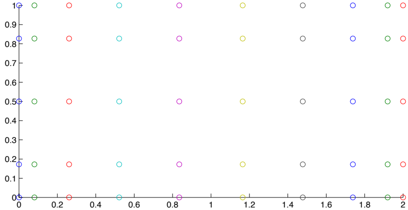
Computing FTLE fields

Uncertainty quantification and Perron-Frobenius

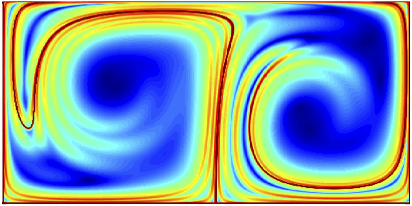
Approximating Koopman eigenfunctions using DMD

Spectral interpolation for FTLE of double gyre

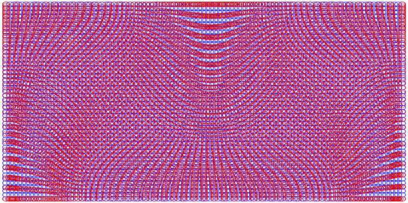
Gauss-Lobatto collocation points



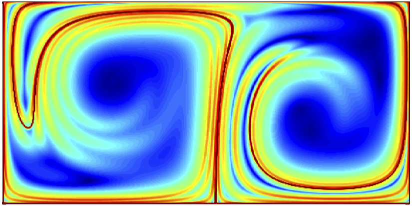
512×256 uniform grid (exact)



Short-time flow map ($\Delta t = 0.1$)



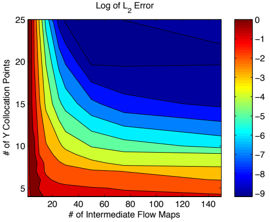
10×5 collocation points ($\Delta t = 0.1$)



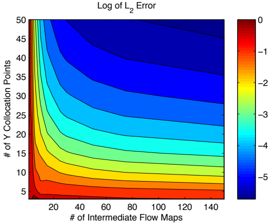
Error comparison

- Measure errors as a function of number of flow map compositions and number of collocation points
- Compare spectral interpolation with cubic spline and linear interpolation
  - Spectral is the most accurate, and uses the least memory
  - Cubic spline faster; a good alternative
  - Linear interpolation is fast, but poor accuracy

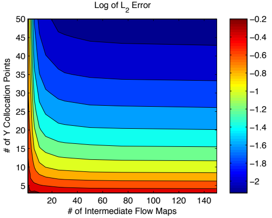
Spectral



Cubic



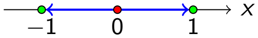
Linear



Outline

Simple ODE example

$$\dot{x} = x(1 - x^2), \qquad x \in [-1, 1]$$



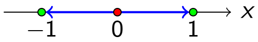
- Want flow map ϕ_t for large times.
- Approximate in terms of Legendre polynomials $\psi_i(x)$:

$$\phi_t \approx \sum_{i=0}^{P} a_i(t)\psi_i(x)$$

- Same as *polynomial chaos* expansion, for an uncertain initial condition uniformly distributed in $[-1, 1]$.

Simple ODE example

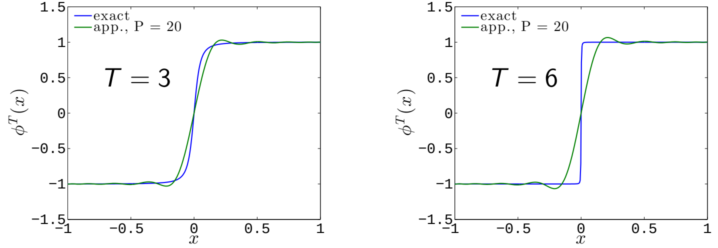
$$\dot{x} = x(1 - x^2), \qquad x \in [-1, 1]$$



- Want flow map ϕ_t for large times.
- Approximate in terms of Legendre polynomials $\psi_i(x)$:

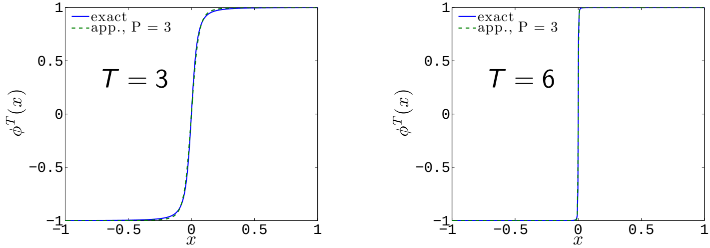
$$\phi_t \approx \sum_{i=0}^{P} a_i(t)\psi_i(x)$$

- Same as *polynomial chaos* expansion, for an uncertain initial condition uniformly distributed in $[-1, 1]$.



Flow map composition: ODE example

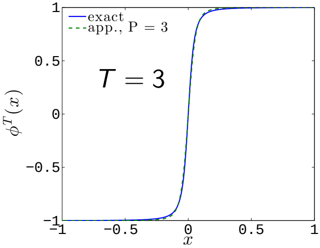
- Compare with results for flow map composition
  - Degree-3 polynomial for $\phi_{\Delta t}$, $\Delta t = 0.2$

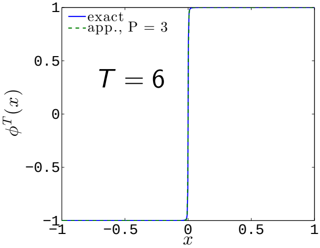


- Greatly improved accuracy, with spectral convergence

Flow map composition: ODE example

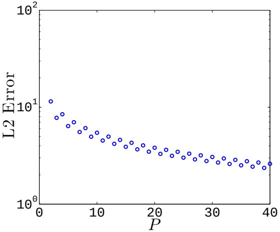
- Compare with results for flow map composition
  - Degree-3 polynomial for $\phi_{\Delta t}$, $\Delta t = 0.2$



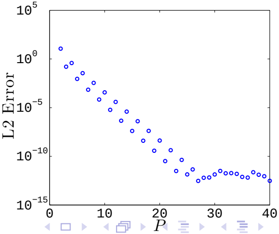


- Greatly improved accuracy, with spectral convergence

Standard PC: poor convergence

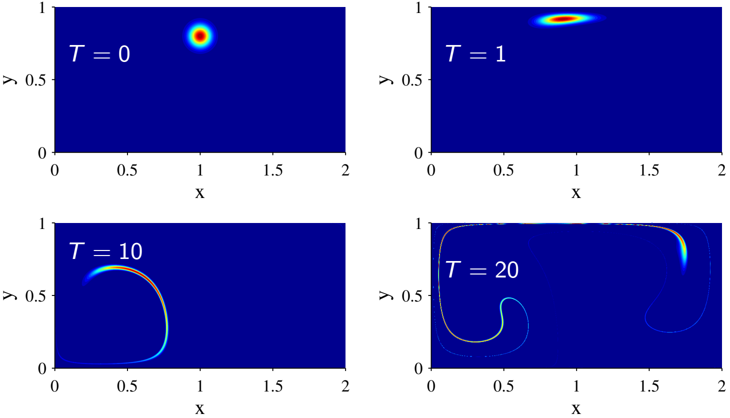


Composition: Spectral convergence



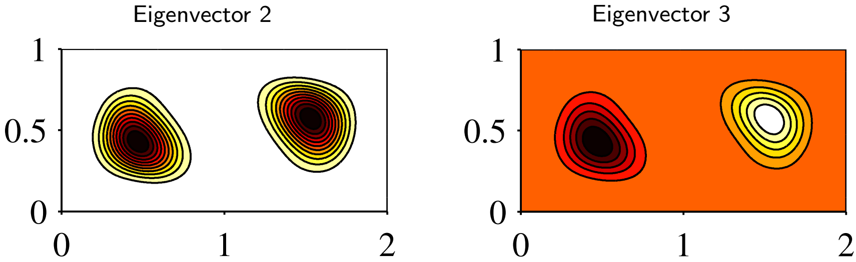
Propagating a PDF in the double gyre

- Propagation of a probability density function using flow map composition
  - Double gyre parameters: $A = 0.25$, $\epsilon = 0.25$, $\omega = 2\pi$
  - Legendre polynomial basis with 11×6 collocation points



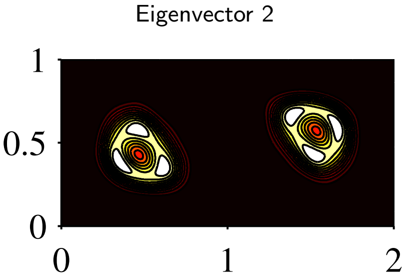
Almost invariant sets: low resolution

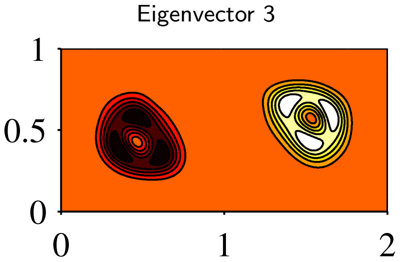
- Calculate eigenvectors of the approximation of Perron-Frobenius
  - 22×12 collocation points
  - Double gyre: $A = 0.25$, $\epsilon = 0.25$, $\omega = 2\pi$
  - Eigenvectors corresponding to near-unity eigenvalues reveal almost-invariant sets



Almost invariant sets: high resolution

- Same calculation at higher resolution reveals islands
  - 43×22 collocation points

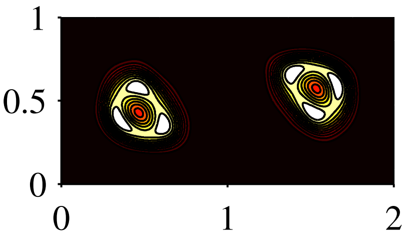




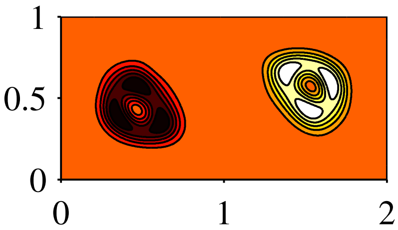
Almost invariant sets: high resolution

- Same calculation at higher resolution reveals islands
  - 43×22 collocation points

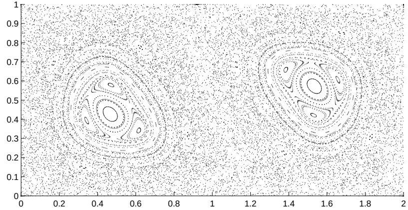
Eigenvector 2



Eigenvector 3



Poincare map



Outline

Approximating a few Koopman eigenfunctions using Dynamic Mode Decomposition

Given a discrete-time dynamical system $\vec{x}_{n+1} = F(\vec{x}_n)$ with $\vec{x}_n \in \mathbb{R}^N$, the action of the Koopman operator $\mathcal{K}$ on $\psi : \mathbb{R}^N \to \mathbb{C}$ is

$$(\mathcal{K}\psi)(\vec{x}_n) = \psi(F(\vec{x}_n)) = \psi(\vec{x}_{n+1}).$$

Approximating a few Koopman eigenfunctions using Dynamic Mode Decomposition

Given a discrete-time dynamical system $\vec{x}_{n+1} = F(\vec{x}_n)$ with $\vec{x}_n \in \mathbb{R}^N$, the action of the Koopman operator $\mathcal{K}$ on $\psi : \mathbb{R}^N \to \mathbb{C}$ is

$$(\mathcal{K}\psi)(\vec{x}_n) = \psi(F(\vec{x}_n)) = \psi(\vec{x}_{n+1}).$$

Our goal is to approximate a few Koopman eigenfunctions, $\varphi(\vec{x})$, using two sets of data,

$$X = \begin{bmatrix} \vec{x}_1 & \vec{x}_2 & \dots & \vec{x}_M \end{bmatrix}, \quad Y = \begin{bmatrix} \vec{y}_1 & \vec{y}_2 & \dots & \vec{y}_M \end{bmatrix},$$

where $\vec{y}_n = F(\vec{x}_n)$.

Approximating a few Koopman eigenfunctions using Dynamic Mode Decomposition

Given a discrete-time dynamical system $\vec{x}_{n+1} = F(\vec{x}_n)$ with $\vec{x}_n \in \mathbb{R}^N$, the action of the Koopman operator $\mathcal{K}$ on $\psi : \mathbb{R}^N \to \mathbb{C}$ is

$$(\mathcal{K}\psi)(\vec{x}_n) = \psi(F(\vec{x}_n)) = \psi(\vec{x}_{n+1}).$$

Our goal is to approximate a few Koopman eigenfunctions, $\varphi(\vec{x})$, using two sets of data,

$$X = \begin{bmatrix} \vec{x}_1 & \vec{x}_2 & \dots & \vec{x}_M \end{bmatrix}, \quad Y = \begin{bmatrix} \vec{y}_1 & \vec{y}_2 & \dots & \vec{y}_M \end{bmatrix},$$

where $\vec{y}_n = F(\vec{x}_n)$.
Using *Dynamic Mode Decomposition*, the approximations of the *Koopman modes* and eigenvalues are obtained by solving the eigenvalue problem:

$$A\vec{v} = \lambda\vec{v},$$

with $A = YX^{\dagger}$, where the rank of A is the smaller of N or M.

Extending Dynamic Mode Decomposition

Instead of operating on raw data, we define M observables, $\psi_m(\vec{x}) : \mathbb{R}^N \to \mathbb{C}$, and form the transformed data matrices

$$\Psi_X = \begin{bmatrix} \psi_1(\vec{x}_1) & \dots & \psi_1(\vec{x}_M) \\ \vdots & \vdots & \vdots \\ \psi_M(\vec{x}_1) & \dots & \psi_M(\vec{x}_M) \end{bmatrix}, \quad \Psi_Y = \begin{bmatrix} \psi_1(\vec{y}_1) & \dots & \psi_1(\vec{y}_M) \\ \vdots & \vdots & \vdots \\ \psi_M(\vec{y}_1) & \dots & \psi_M(\vec{y}_M) \end{bmatrix},$$

and compute the left-eigenvectors of

$$\vec{w}^*(\Psi_Y \Psi_X^\dagger) = \lambda \vec{w}^*.$$

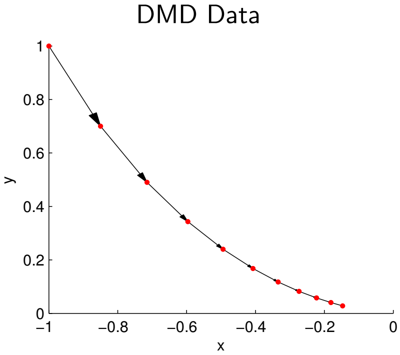
For a given left-eigenvector, the approximation of the Koopman eigenfunction is

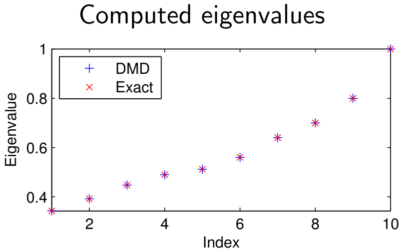
$$\tilde{\varphi}(\vec{x}) = \sum_{j=1}^{M} w_j^* \psi_j(\vec{x}), \tag{1}$$

where w_j^* is the complex conjugate of the j-th element of $\vec{w}$. Note: using regular DMD $\psi_j(\vec{x}) = u_j^* x$, where u_j is a basis vector for the image of X.

Computing Koopman eigenfunctions: a linear example

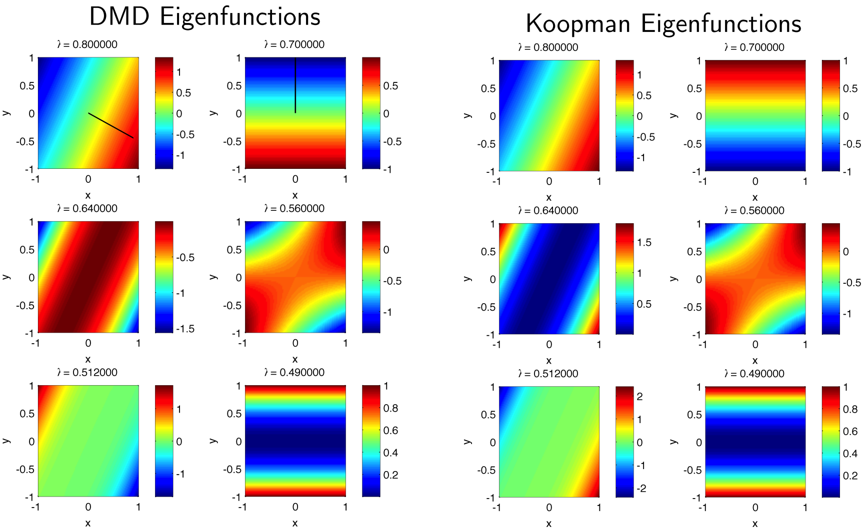
- $\vec{x}_{n+1} = \begin{bmatrix} 0.8 & -0.05 \\ 0 & 0.7 \end{bmatrix} \vec{x}_n$, with $\lambda = 0.8, 0.7$.
- Data are a time series of 11 snapshots
- Basis functions (observables) are $\psi_{i,j}(x,y) = x^i y^j$ for $i,j = 0,1,2,3$.





- Desired eigenfunctions: $\varphi_{i,j}(x,y) = (2x-y)^i y^j$ for $i,j \in \mathbb{N}$
- $\lambda_{i,j} = (0.8)^i (0.7)^j$

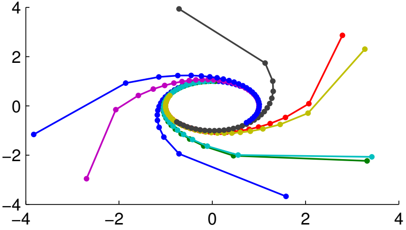
Comparing the eigenfunctions: a linear example



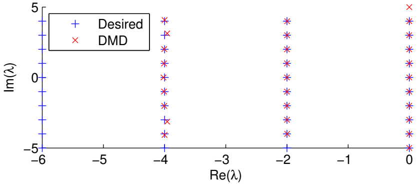
A nonlinear example: the Stuart-Landau equation

- $\frac{dA}{dt} = a_0 A - a_1 |A|^2 A$, with $A \in \mathbb{C}$, $a_0 = 1$, $a_1 = 1 + i$
- Eight time series ($\Delta t = 0.1$) with 29 snapshots each
- Choose $\psi_{m,n}(r, \theta) = r^m e^{in\theta}$ with $A = r\exp(i\theta)$
- $m = -4, \ldots, 3$ and $n = -16, \ldots, 16$.

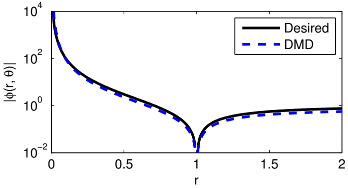
DMD Data



Computed eigenvalues



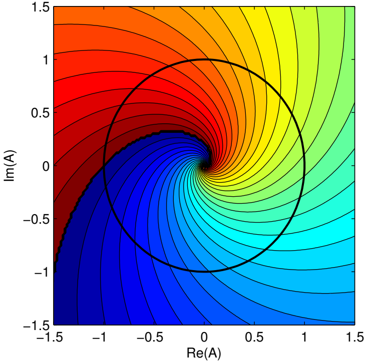
Computed eigenfunction ($\lambda = -2$)



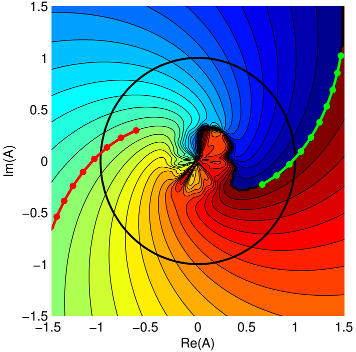
[0] S. Bagheri. *Koopman Mode Decomposition of the Cylinder Wake*, JFM **726**, 2013.

Computing isochrons in the Stuart-Landau equation

Analytic $\angle\phi_{0,1}$



Computed $\angle\phi_{0,1}$



- Koopman eigenfunctions:

$$\phi_{m,n} = \left(\frac{1}{r^2} - 1\right)^m \exp\left(in\left(\theta + \ln\left(\frac{1}{r}\right)\right)\right)$$

- Plot of the level sets of $\angle\phi_{0,1}$
- Good agreement with the analytical results

[0] A. Mauroy, I. Mezic, J. Moehlis. *Isostables, isochrons, and Koopman spectrum for the action-angle representation of stable fixed point dynamics.* arXiv:1302.0032 [math.DS]

Summary

- Efficient representation of long-time flow maps
  - Compose short-time flow maps
  - Represent short-time flow maps by spectral interpolation
- Examples
  - Computing FTLE fields
  - Propagating probability density functions
  - Computing eigenfunctions of Perron-Frobenius
- Approximate Koopman eigenfunctions using Dynamic Mode Decomposition (DMD)
  - Sample several observables from different points in phase space
  - Reconstructs Koopman eigenfunctions for both linear and nonlinear problems