Chatterjee, 2000

~~Proper~~ PROPER ORTHOGONAL DECOMPOSITION | 13

Chatterjee, Current Science, 2000

"Introduction to the proper orthogonal decomposition"

~~Words of Caut~~ ~~Rep~~ POD: obtain low-dimensional approximate descriptions of high-dim. ~~sys~~ processes

a.k.a./~~ren~~ related:

- Principle Component Analysis
- Karhunen-Loéve decomposition
- Single-value ~~de~~decomposition

Words of Caution: →

*** POD is sensitive to coord. changes

note that rank ≠ amount of information contained

*** Inappropriate scaling of variables measured (very possible in systems w/ mixed measurements e.g. acceleration, displacement, & strain) → can lead to misleading &/or meaningless results

→ Using delay coordinate embedding implicitly involves strongly nonlinear change of variables

based on linear combos of basis functions

*** Some physical systems may be poorly suited to naive POD analysis

- Measurement & process noise:
  & for single variable where $\dot{x}$ is numerically calculated → differentiation noise. Counter act w/ total variation regularization.

- Algorithms for sparse regression
  SI → • LASSO works, * may be computationally expensive
  [other disadvantages?] - potentially sensitive to noise?]
  • Alternately (as implemented in SI examples)
  * Sequential Thresholded Least-Squares algorithm
    → start w/ least squares solution
    → threshold all coefficients smaller than some cutoff value λ.
    → Repeat Least-Squares regression on remaining coeff. indices.
    → Repeat thresholding of resulting coeff.
    → Iterate until non-zero coeff. converge
  Efficient and simple, only parameter is λ.
  "Remarkably" robust to noise.

- Filtering or regularization for noisy input derivatives
  esp. if $\dot{x}$ is calculated instead of measured
  ↳ use total variation regularized derivative

LINEAR ALGEBRA

$$A = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \qquad B = \begin{bmatrix} 3 & 0 \\ 1 & 5 \end{bmatrix}$$

$$A * B = \begin{bmatrix} 6 & 15 \\ 10 & 20 \end{bmatrix}$$

$$B * A = \begin{bmatrix} 3 & 9 \\ 11 & 23 \end{bmatrix}$$

$$A \backslash B \longrightarrow Ax = B$$

$$= \begin{bmatrix} -4.5 & 7.5 \\ 2.5 & -2.5 \end{bmatrix}$$

$$\begin{array}{cc} & \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix} \\ \underset{A}{\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}} & \underset{B}{\begin{bmatrix} 3 & 0 \\ 1 & 5 \end{bmatrix}} \end{array}$$

$$A / B \longrightarrow xB = A$$

$$= \begin{bmatrix} 0.13 & 0.6 \\ 0.4 & 0.8 \end{bmatrix}$$

$$\begin{array}{cc} & \overset{B}{\begin{bmatrix} 3 & 0 \\ 1 & 5 \end{bmatrix}} \\ \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix} & \underset{A}{\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}} \end{array}$$

$$B / A \longrightarrow xA = B$$

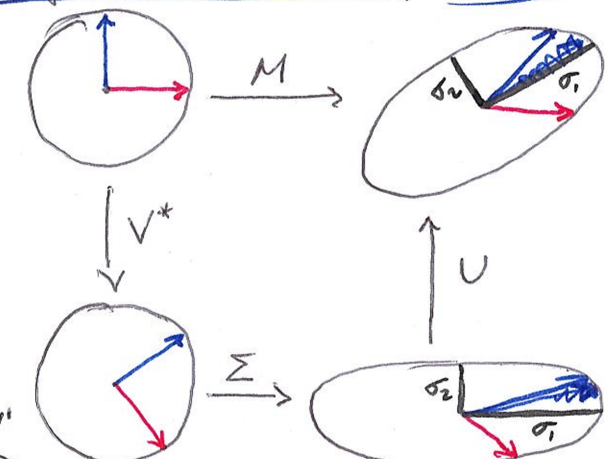
$$= \begin{bmatrix} -6 & 4.5 \\ 3 & -1 \end{bmatrix}$$

$$\begin{array}{cc} & \overset{A}{\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}} \\ \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix} & \underset{B}{\begin{bmatrix} 3 & 0 \\ 1 & 5 \end{bmatrix}} \end{array}$$

~~Noort & Kalta, J. Hydrol., 2016~~

SVD — Singular Value Decomposition

~~SWAT~~



also notated as V^T or V'
"V transpose"

$$M = U \Sigma V^*$$

* indicates conjugate transpose of V

σ_1 & σ_2 are singular values of M

U: rotation/reflection
Σ: scaling
V^*: rotation/reflection

if M is $m \times n$:

U is $m \times m$ unitary matrix

Σ is $m \times n$ diagonal matrix, entries are the "singular values" of M arranged in decreasing order

V^* is $n \times n$ unitary matrix

unitary, so columns form a set of orthonormal vectors which can be regarded as (orthonormal) basis vectors

e.g. M maps basis vector V_i to the stretched unit vector $\sigma_i U_i$

SVD can be applied to any $m \times n$ matrix whereas eigenvalue decomposition can only be applied to certain classes of square matrices (although SVD & EVD are related).

16| SVD (Matlab & Chatterjee, 2000)

in Matlab:

$$[U, S, V] = svd(A, 'econ')$$

'econ' produces economy-size decomposition of $m \times n$ matrix A such that $A = U*S*V'$

→ $m > n$: only first n columns of U are computed; S is $n \times n$

→ $m = n$: svd(A, 'econ') is equivalent to svd(A)

→ $m < n$: only the first m columns of V are computed; S is $m \times m$

Chatterjee, 2000.

Geometric Interpretation of SVD for POD applications:

A ($N \times m$ matrix) is a list of the coordinates of N points in m-dimentional space.

For any $k \leq m$, we seek a k-dimensional sub-space for which the mean square distance of the points from the sub-space is minimized. A basis for this sub-space is given by the first k columns of V.

The practice of subtracting the column mean from each column ensures that the N-point "cloud" is centered around the origin.

1 | SINDy   Brunton et al. 2016

Sparse identification - $\boxed{SI}$

SI - Fig. 8: ? ABC/vel. u_{xyz}

L_1/L_2

genetic programming: evolutionary algorithm that builds & tests candidate fns out of simple building blocks

Proper Orthogonal Decomposition

Figs 10/11/12 : data included in model vs. results

* SI 4.5 - SINDy with time-delay coordinates
  Extract dynamics in the Lorenz system if only the first variable $x(t)$ is measured. It is well-known that time delay coordinates allow us to synthesize additional dynamic variables using time-series measurement from a single variable $x(t)$.

  [Ye, H, et al. (2015) Equation-free ~~model~~ mechanistic ecosystem forecasting using empirical dynamic modeling. PNAS 112: E1569-E1576.]

↳ eigen-time-series from "singular value decomp."
use these new time delay coordinates & derivatives as input to SINDy ⟶ model coeff.

⇒ "For short times, the identified dynamics are qualitatively similar to the true time-delay embedding, capturing the skeleton of the attractor."
Fig. 18

? ↳ for short times meaning for the first few steps in the time series?
↳ I think yes, see pg SI-28 & facing line Z

illustrated in Lorenz example: "ability to predict a specific trajectory is not as important as the ability to capture attractor dynamics."

method demonstrated on systems with: chaos, big data w/ low coherence, parameterized dynamics [and time-delay embedding].

[Trade-off btw fit & over fit]

* Open Problems: Trade-off w/ choice of λ (sparsifying parameter) → decreasing λ increases model fit w/ data, but model then includes higher-order non-linearities

- dynamical symmetries & conserved quantities may alter the form of the identified dynamics.
  - for example, degenerate identification of a linear system in a space of high-order polynomial linearities suggest a connection with near-identity transformations & dynamic similarity (?)
- how to choose measurement coordinates & sparsifying function basis for the dynamics.
  - draw on expert knowledge, feature extraction, & inference-based ~~[illegible]~~ methods.

I.e.: How do you choose feature testing to include in your model on which to fit?

- too few measurements → augment with time-delay
- too many measurements → extract coherent structures using advanced methods from dimensionality reduction & ML. Maybe also sparsify using subsequent coordinate transformations.

* sparse identification requires ~~you want to have~~ measurements in a sensible coord. system where the dynamics are sparse in the chosen function basis.

* e.g. trying sparse ID on Lorenz in non-linearly transformed coord → sparse ID fails to ID accurate model.

[How would you know the sparse ID had failed? Compare w/ results from ~~eigen-time-delay~~ other coord. syst. choices?]

↳ Using eigen-time-delay coordinates may help to find a natural coord. syst. → using these w/ transformed Lorenz results in much better model-observed match...

→

SINDy

transformed Lorenz system w/ eigen-time-series
→ better correlation btw measured & modeled
* → but resulting dynamical systems do not reproduce attractor dynamics
⇒ open question ~~to~~ on how to identify natural coordinate systems to measure in & natural function bases to represent dynamics sparsely.

Example: Glycolytic oscillator
- SINDy IDs ~~features~~ terms ~~w/~~ that ~~have~~ are sparse in polynomial search basis
- Does not ID ~~features~~ for terms w/ rational fn in their dynamics.
- ID'ed model accurately matches measured derivatives, but dynamical model ~~does~~ not agree w/ true system, except ~~at~~ for a very short time at the beginning of the simulation.

* [Questions ~~&~~ / Thoughts]
- In the case of ~~a~~ poor ~~fit~~ suitability / ~~of~~ compatability btw underlying system coordinate space & function bases w/ those chosen for SINDy.

it is stated:
1) eigen-time-series transformation may ~~increase~~ improve short-term correlation w/ ~~measured~~ "true system"
2) this can capture the "skeleton" of the attractor
3) but resulting dynamical systems do not reproduce attractor dynamics.

*** ⇒ If eigen-time-series embedding is a non-linear transform: that can potentially break ability ~~of~~ SINDy to ID syst. dynamics, how is it also a method that potentially helps ID appropriate coord. syst.? →

- This seems akin to the example of the decay of the ability to predict partially/not determined chaotic weather systems with increase of time step into the future you want to predict.
- You have some idea where the system will go in the near future, but you ~~don't~~ haven't effectively ~~identi~~ identified the underlying system (attractor) & therefore cannot predict past a certain point.
- What is the balance between the statements that ~~transformed~~ (imperfectly) transformed coordinates (eg. eigen-time-series) can ~~repre~~ capture the skeleton of the attractor, but do not successfully/sufficiently represent the dynamical system (attractor)? or level of success
  → Is this just a difference in example results?
  → How do you (or can you) know if/how your SINDy id'ed model is accurate/ representative or not?

→ What are the implications of the limitations of SINDy ~~to~~ in the application of SINDy to malaria systems?

→ Do you need to know the full shape of the attractor or just what the system is currently doing/will do in the near future? →

- This seems akin to the example of the decay of the ability to predict partially/not determined chaotic weather systems with increase of time step into the future you want to predict.
- You have some idea where the system will go in the near future, but you ~~don't~~ haven't effectively identified the underlying system (attractor) & therefore cannot predict past a certain point.
- What is the balance between the statements that ~~transformed~~ (imperfectly) transformed coordinates (eg. eigen-time-series) can ~~repre~~ capture the skeleton of the attractor, but do not successfully/sufficiently represent the dynamical system (attractor)? or level of success
  - → Is this just a difference in examples results?
  - → How do you (or can you) know if/how your SINDy id'ed model is accurate/representative or not?

→ What are the implications of the limitations of SINDy in ~~to~~ the application of SINDy to malaria systems?

→ Do you need to know the full shape of the attractor or just what the system is currently doing/will do in the near future? →

SINDy

- I think you need to know the full shape of the attractor given the strong seasonality of dynamics in the malaria syst.
- One possible approach might be to do a SINDy analysis by season (do you clip out season & trade length of time series for # of iterations?)
- Is the dependency of SINDy accuracy on near-time predictions (for poorly chosen coordinates) a matter of ~~where in the time series you start~~ ~~when in time you choose to start your time series~~ (e.g. season) ~~or a~~ constraint on # of time steps into the future? (Probably option B).
- Is there a ~~result~~ use for changing how you clip/order/aggregate e.g. by season?

★★★ NOTE:
⇒ Sensitivity of method to coordinate system & variable scaling
→ What about constructing SINDy analysis with iterations on diff coord systems?
→ ~~Running~~ ~~e.g.~~ calculate on a number of different transformations & normalizations to compare
→ Compare sensitivity of results
★ → How does coord. syst. transf. affect interpretation of results?

SINDy

(Josh)

★★★ Order of magn. of feature values across features should be the same

examples: Dennis, Brainstorm

look at ★ ~~dist of~~ ~~data~~ values of a feature. It should look "nice"

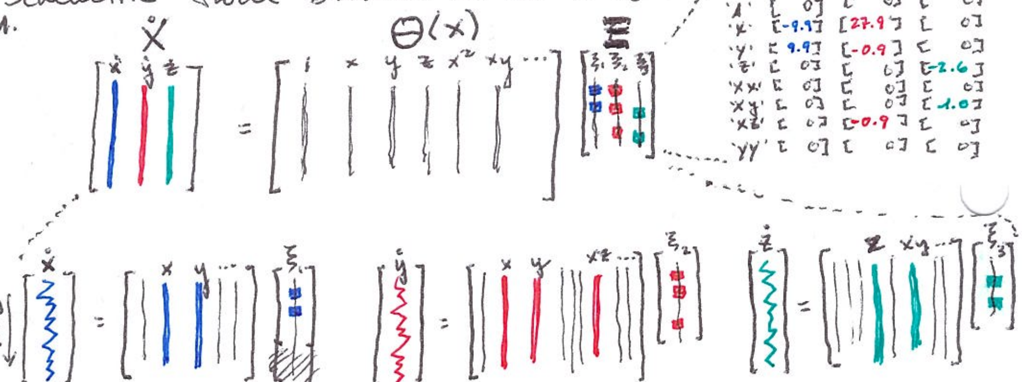
design with interpretebility in mind
↳ spatial decomposition of dynamics
(DMD?) Dynamical Mode Decomposition

✓ [talk to Josh's post-doc?] ~~Nate~~ Neil (Neal?)

★ Set up pipeline for model dev ★

Schematic from Brunton et al. 2016

Fig 1.



"	'xi_1'	'xi_2'	'xi_3'
'1'	[0]	[0]	[0]
'x'	[-9.9]	[27.9]	[0]
'y'	[9.9]	[-0.9]	[0]
'z'	[0]	[0]	[-2.6]
'xx'	[0]	[0]	[0]
'xy'	[0]	[0]	[1.0]
'xz'	[0]	[-0.9]	[0]
'yy'	[0]	[0]	[0]

True Lorenz System

$\dot{x} = \sigma(y - x)$

$\dot{y} = x(\rho - z) - y$

$\dot{z} = xy - \beta z$

$\sigma = 10,\ \beta = 8/3,\ \rho = 28$

Identified System

$\dot{x} = \Theta(x^T)\xi_1$

$\dot{y} = \Theta(x^T)\xi_2$

$\dot{z} = \Theta(x^T)\xi_3$

Singular value decomp:

$M = U\Sigma V^T$

↳ $x = V$

58 | SINDy - MATLAB implementation

SI examples & MATLAB functions

⇒ poolData.m % [build library of features]

yout = poolData(yin, nVars, polyorder, usesin)

yout — Theta: non-linear feature library

⇒ sparsifyDynamics.m % [Compute Sparse regression]

Xi = sparseDynamics(Theta, dXdt, Lambda, n)

EX01a_Linear2D.m

% generate data

polyorder = 5; % search space up to 5th order polynom.

usesin = 0; % no trig fns

n = 2; % 2D system

$$A = \begin{bmatrix} -.1 & 2 \\ -2 & -.1 \end{bmatrix} \quad \text{\% dynamics}$$

rhs = @(x) A*x; % ODE rhs ← matrix multiplic.

tspan = [0 : .01 : 25]; time span

$x0 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$; % initial conditions

[t, x] = {ode integrate over time}

$$A * x0 = \begin{bmatrix} -.1(2) + 0 \\ -4 + 0 \end{bmatrix} = \begin{bmatrix} -.2 \\ -4 \end{bmatrix}$$

% Compute derivative:

for i = 1 : length(x)

dx(i, :) = A * x(i, :)'

end

2 layers of Dim reduction:

POD var space transformation

& Sparse linear regression

reducing features

⇒ feature reduction depends on basis choice

EX02_LorenzTVDiff.m – Total Variation Regularized ~~Dif.~~ Differentiation

generate data

no noise : xclean

noise : x

clean derivative :

compute derivative on xclean

☆ Numerical differentiation :

Total Variation Regularized Differentiation

Numerical calculation of derivative ~~(e.g.~~ (e.g. finite difference) fundamentally introduces error into resulting derivative (in addition to calculation precision error).

function : **TVRegDiff**

parameters:

- data = vector of data to be diff.
- iter = # of iterations ← (no defaults)
- alph = regularization parameter

→ This is the main parameter to play with. Start by varying by orders of magnitude until reasonable results are obtained. A value to the nearest power of 10 is usually ~~accurate~~ adequate. Higher values increase regulariz. strength & improve conditioning.

- u0 = initiallization of the iteration
- scale = 'large' or 'small' (default) ~~(small~~

→ 'small' has somewhat better boundary behavior, but >1k obs may be better w/ 'large'

- ep = param for avoiding ~~d~~ div by 0
- dx = grid spacing (default = reciprocal of data size
- diagFlag = flag to run diagnostics

output u is est. regularized derivative, 1 entry larger than input data for 'small' scale, same if 'large'

$\boxed{\text{sparsifyDynamics}}$.m

Xi = sparsifyDynamics(Theta, dXdt, λ, n)

→ lambda is sparsification knob.

→ n is # of dimensions of system

→ Theta is matrix wherein each column represents a feature in the basis & each row is the value of that feature at a particular time step.

★→ Xi is sparse matrix of feature coefficients of ~~size(featureBasis~~ size(features, sysDim)

$\boxed{\text{sparseGalerkin}}$ .m ~~computes new sparse rhs function using spars~~

dy = sparseGalerkin(t, y, ahat, polyord, usesin)

coefficients

↳ use this w/ ode45 integration to ~~find~~ ~~identi~~ calculate system behavior ~~dynamics~~ according to identified model

i.e.

```
[t, xt] = ode45(@(t,x) sparseGalerkin(t, x, Xi, ...
    polyorder, usesin), tspan, x0, odeOptions)
```

odeOptions e.g. ~~odeset('RelTol~~

= odeset('RelTol'), 1e-8, 'AbsTol', 1e-8*ones(1,3))

or 10 or 12 ... 10, 12, ... n

★

[long, lat, value]