Numerical Matrix Analysis

Notes #23 — The Singular Value Decomposition Application to Signal and Data Analysis

Peter Blomgren
⟨blomgren@sdsu.edu⟩

Department of Mathematics and Statistics
Dynamical Systems Group
Computational Sciences Research Center
San Diego State University
San Diego, CA 92182-7720

http://terminus.sdsu.edu/

Spring 2024

(Revised: April 16, 2024)

Outline

Searches on scholar.google.com

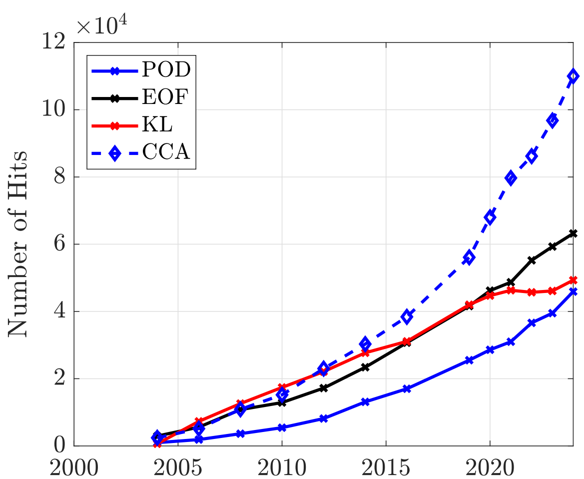


Figure: The many names, faces, and close relatives of the Singular Value Decomposition... Number of hits for "Proper.Orthogonal.Decomposition", "Empirical.Orthogonal.(Function|Functions)", "Karhunen.Loeve", "Canonical.Correlation.Analysis"

Searches on scholar.google.com

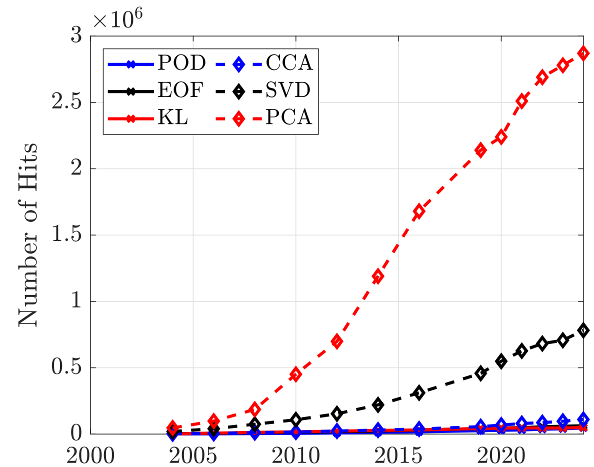


Figure: The many names, faces, and close relatives of the Singular Value Decomposition... Number of hits for "Proper.Orthogonal.Decomposition", "Empirical.Orthogonal.(Function|Functions)", "Karhunen.Loeve", "Canonical.Correlation.Analysis", "Singular.Value.Decomposition", "Principal.Component.Analysis"

Searches on scholar.google.com

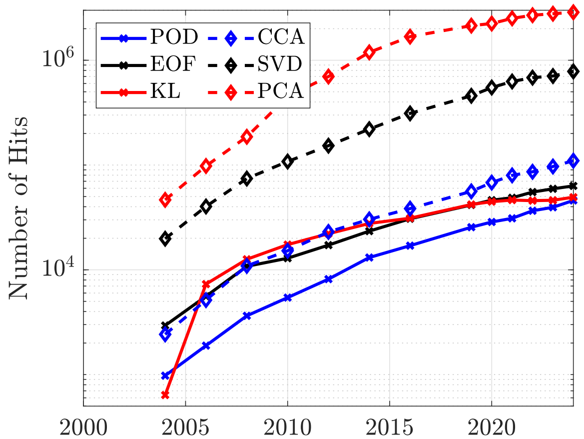


Figure: The many names, faces, and close relatives of the Singular Value Decomposition... Number of hits for "`Proper.Orthogonal.Decomposition`", "`Empirical.Orthogonal.(Function|Functions)`", "`Karhunen.Loeve`", "`Canonical.Correlation.Analysis`", "`Singular.Value.Decomposition`", "`Principal.Component.Analysis`"

Applications/Keywords "Random" results from scholar.google.com

Principal.Component.Analysis
Positron Emission Tomography (PET), Gene Clustering, fMRI, Dynamics of the Bovine Pancreatic Trypsin Inhibitor (BPTI), ...

Singular.Value.Decomposition
Genome data processing, Orthogonal Frequency Division Multiplexing (OFDM) channel estimation, Information retrieval, Hamiltonian mechanics, ...

Empirical.Orthogonal.(Function—Functions)
Statistical weather prediction, Atlantic Ocean surface temperatures, Acoustic classification of zoo-plankton, ...

Applications/Keywords "Random" results from scholar.google.com

Canonical.Correlation.Analysis
fMRI, Neural activity, Climate forecasts, Identification of hydrological neighborhoods, El Niño/Southern Oscillation (ENSO) prediction, ...

Proper.Orthogonal.Decomposition
Turbulent flows, Vibroimpact oscillations, Cavity flows, Optimal control of fluids, Magneto-Hydro-Dynamics (MHD) flows, ...

Karhunen.Loeve
Characterization of human faces, Cosmology, Turbulence Modeling, Multi-spectral image restoration, Universal image compression, ...

Application: Analysis of Dynamic Pattern Formation

The **Kuramoto-Sivashinsky** equation, here in polar coordinates

$$\begin{aligned} u_t &= -u_{rrrr} - \frac{1}{r^4}u_{\phi\phi\phi\phi} - \frac{2}{r^2}u_{rr\phi\phi} - \frac{2}{r}u_{rrr} + \frac{2}{r^3}u_{r\phi\phi} \\ &\quad - \left[2 - \frac{1}{r^2}\right]u_{rr} - \left[\frac{4}{r^4} + \frac{2}{r^2}\right]u_{\phi\phi} - \left[\frac{1}{r^3} + \frac{2}{r}\right]u_r \\ &\quad + \eta_1 u - \eta_2\left[u_r^2 + \frac{1}{r^2}u_\phi^2\right] - \eta_3 u^3, \end{aligned}$$

is a model for the behavior of cellular flames stabilized on a circular porous plug burner. For different simulation parameters $(\eta_1, \eta_2, \eta_3, R)$ it exhibits a wide array of complex flame patterns; — mimicking patterns observed in physical experiments.

∃ Movies.

Integrating the Kuramoto-Sivashinsky Equation

- We defer all discussion on how to time-integrate the Kuramoto-Sivashinsky equation to [MATH 693B].
- We note that each time step (from t to $t+\delta t$, where δt is "small"), requires the solution of several non-Hermitian linear systems $A\vec{x} = \vec{b}$, where in our set-up $A \in \mathbb{R}^{m \times m}$, with $m = 2048$.
- In what follows, we keep the parameters $(\eta_1, \eta_2, \eta_3) = (0.32, 1.00, 0.017)$ constant, and vary **only** the radius of the circular burner.
- For the majority of radii, we get static (non-moving) patterns, which are quite easy to classify.
- However, for some fairly narrow parameter ranges we get time-dependent (dynamic) patters. We will use the SVD to analyze and classify these patterns.

Static Patterns Observed in the Kuramoto-Sivashinsky Simulations

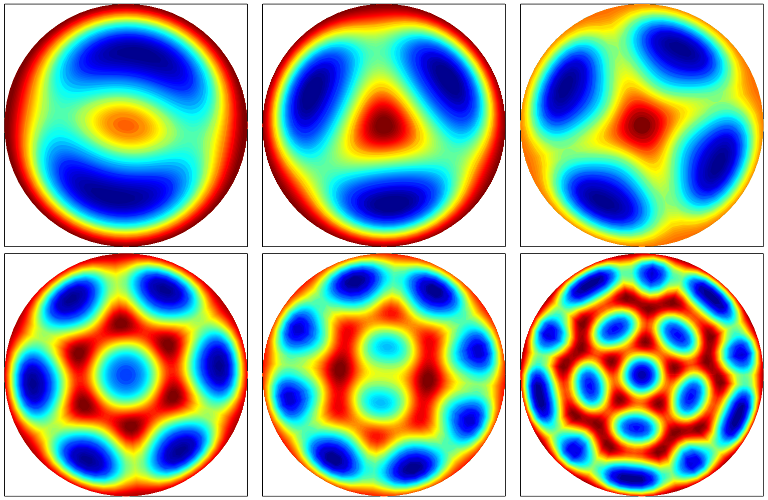
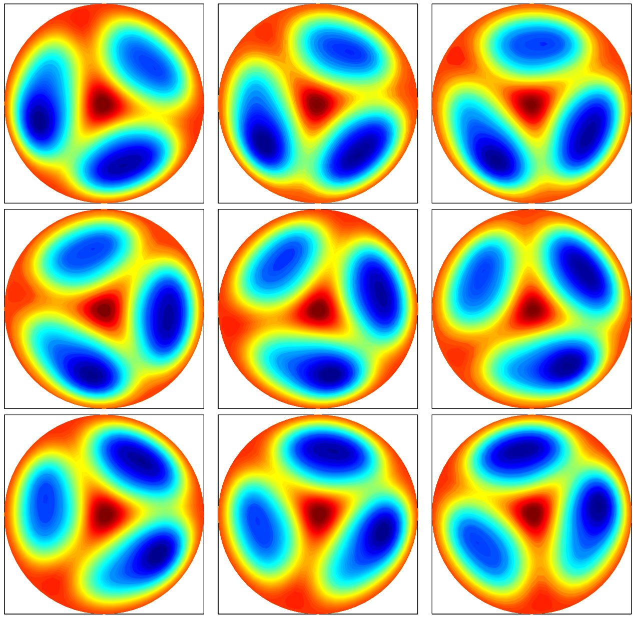
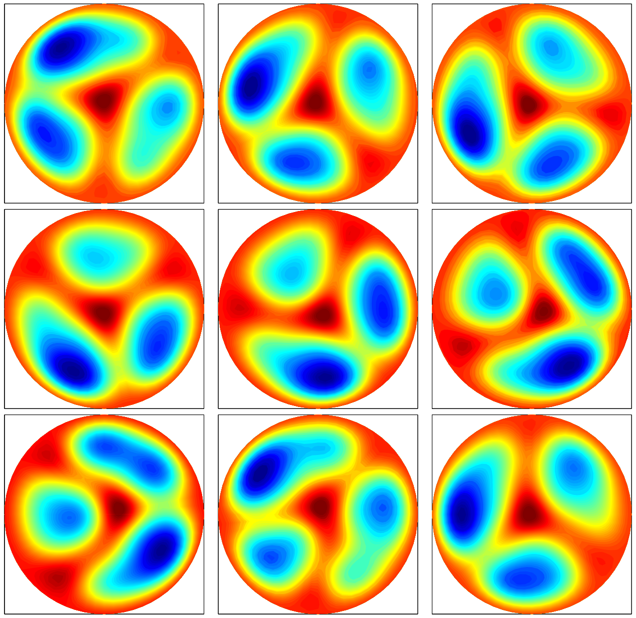


Figure: Some of the static patterns observed using the Kuramoto-Sivashinsky integration scheme. 2-cell pattern, $R = 5.0$; 3-cell pattern, $R = 6.0$; 4-cell pattern, $R = 8.0$; 6/1-cell pattern, $R = 10.0$; 8/2-cell pattern, $R = 12.0$; 10/5/1-cell pattern, $R = 14.5$; *Common simulation parameters:* $(\eta_1, \eta_2, \eta_3) = (0.32, 1.00, 0.017)$.

Dynamic Pattern #1: 3-Cell (Nearly) Rigid Rotation



Dynamic Pattern #2: 3-Cell "Hopping Pattern"



Dynamic Pattern #1–2: 3-Cell Dynamic States (Captions)

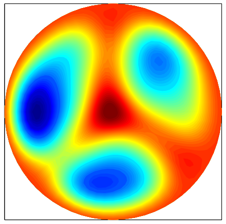
Figure (Slide 9): Snapshots of a three-cell nearly rigid rotation state from a simulation of the Kuramoto-Sivashinsky equation. The pattern is shown at times $t \in \{0, 15, 30, 45, 60, 75, 90, 105, 120\}$. The simulation parameter values are: $(\eta_1, \eta_2, \eta_3;\ R) = (0.32, 1.00, 0.017;\ 7.36)$. In this sequence we see how the pattern stays the same, but rotates counter-clockwise.

Figure (Slide 10): Snapshots of a three-cell hopping state from a simulation of the Kuramoto-Sivashinsky equation. The pattern is shown at times $t \in \{0, 15, 30, 45, 60, 75, 90, 105, 120\}$. The simulation parameter values are: $(\eta_1, \eta_2, \eta_3;\ R) = (0.32, 1.00, 0.017;\ 7.7475)$. In this sequence we see how the "front-runner" of the two-cell formation bridges the gap to the solitary cell.

Analyzing the Dynamic Patterns "The Method of Snapshots"

We use the SVD in order to analyze and classify these dynamic patters.

Each "frame", $u^{(i)}(r, \phi)$ with 32 radial, and 64 azimuthal points, of the sequence defines a 2048×1-vector $\tilde{f}_i$:



$$\rightarrow \quad \tilde{f}_i = \begin{bmatrix} u^{(i)}(r_1, \phi_1) \\ \vdots \\ u^{(i)}(r_1, \phi_{64}) \\ u^{(i)}(r_2, \phi_1) \\ \vdots \\ \vdots \\ u^{(i)}(r_{32}, \phi_{64}) \end{bmatrix}$$

Analyzing the Dynamic Patterns: The Snapshot Matrices

For both the rigidly rotating, and the hopping pattern, we have computed 7200 frames, hence for each simulation we can build a 2048×7200 matrix of snapshots

$$\tilde{A} = \left[\ \tilde{f}_1 \quad \tilde{f}_2 \quad \dots \quad \tilde{f}_{7200}\ \right].$$

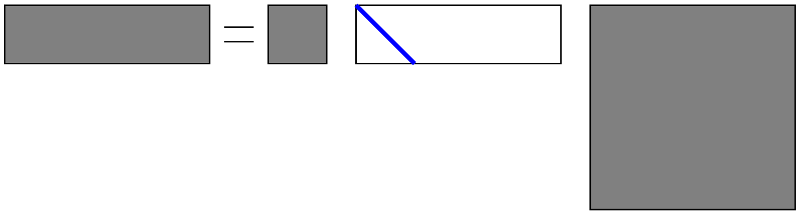
It turns out that for this application (and many others) it is advantageous to view each snapshot as a perturbation from the mean, so with $\vec{m}_f = \mathrm{mean}_{i=1,\dots,7200}(\tilde{f}_i)$, we define new vectors $\vec{f}_i = \tilde{f}_i - \vec{m}_f$, and a new "snapshot perturbation matrix"

$$A = \left[\ \vec{f}_1 \quad \vec{f}_2 \quad \dots \quad \vec{f}_{7200}\ \right].$$

Interpreting $U\Sigma V^* = A$

We now compute the SVD of the snapshot perturbation matrix, so that

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

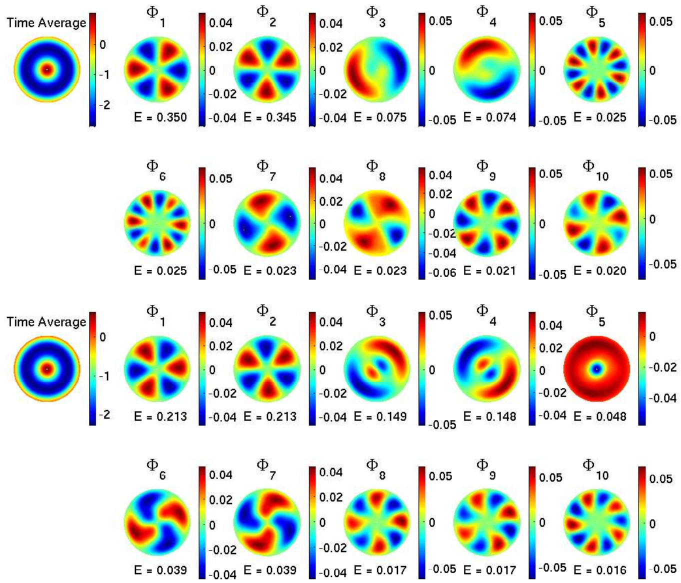


Restatement of the obvious: U is orthonormal, and has the same column space as A, *i.e.* it is an orthonormal basis for $\mathrm{range}(A)$.

The singular values σ_i tell us how "important" each column in U is, *i.e.* how much perturbation "energy" is controlled by the ith column of U.

In this application the left singular vectors (columns of U) are of interest.

$\vec{m}_f$ and $\vec{u}_1, \ldots, \vec{u}_{10}$ for Rigid Rotation and Hopping



Some Discussion...

For the **rigid rotation** we see that

$\sim$ 70% of the energy is controlled by $\vec{u}_1$-$\vec{u}_2$, which express rotations of 3-cell perturbations from the mean.

There is $\sim$ 15% of the energy in the $\vec{u}_3$-$\vec{u}_4$ pair (rotations of 1-cell perturbations), and

$\sim$ 5% of energy in 6-cell, and 2-cell perturbations;

the first 10 columns catch in excess of 98% of the energy, and hence provide an almost complete description of the motion.

Some Discussion...

For the **hopping motion**

we first notice that the rotations of 3-cell perturbations from the mean now only control $\sim$ 42% of the motion, and

about 5 times as much energy ($\sim$ 10%) has "leaked" outside the first 10 columns.

— All of this is an indication that the motion is much more complex.

Further, the $\vec{u}_3$-$\vec{u}_4$ pair (of the rigid rotation) has formed a more complex $\vec{u}_3$-$\vec{u}_4$-$\vec{u}_5$ triple, and

the 2-, 4-, and 5-cell rotations have overtaken the importance of the 6-cell rotations (which is no longer in the "top 10.")

Expressing the Motion Using the Orthogonal Basis

Since U is an orthonormal basis, it is very straight-forward to write any frame as a linear combination of the basis vectors $\vec{u}_k$:

$$\vec{f}_i = \sum_{k=1}^{2048} a_{ik}\vec{u}_k, \quad \text{where} \quad a_{ik} = \vec{u}_k^* \vec{f}_i$$

Observation: Since, in both cases, the first 10 basis vectors control at least 90% of the motion, therefore

$$\vec{f}_i \approx \sum_{k=1}^{\mathbf{10}} a_{ik}\vec{u}_k, \quad \text{where} \quad a_{ik} = \vec{u}_k^* \vec{f}_i$$

should be a good approximation.

$\rightsquigarrow$ **Compression:** By storing 10 (2048×1) basis vectors, 7200×10 coefficients, and the average vector (2048×1) $\vec{m}_f$ for a total of 94,528 values instead of the full $7200 \times 2048 = 14,745,600$-value dataset, we get a **compression ratio of $\frac{1}{156}$**.

The Coefficients a_{ik}

The coefficients $a_{ik} = \vec{u}_k^* \vec{f}_i$ give us a lot of useful information.

If the rotation is completely rigid then when $\vec{u}_k$-$\vec{u}_{k+1}$ describe the rotation of some n-cell pattern, the points $(a_{i,k}, a_{i,k+1})$ should form a circle in $\mathbb{R}^2$, usually referred to as *phase space*...

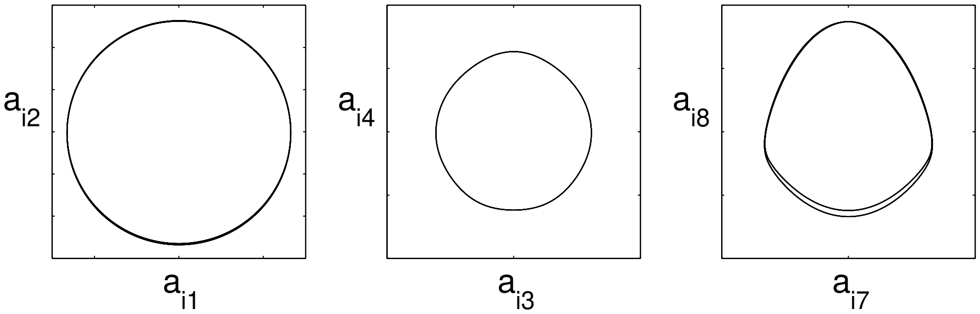


Figure: The phase plots for $\vec{u}_1$-$\vec{u}_2$, $\vec{u}_3$-$\vec{u}_4$, and $\vec{u}_7$-$\vec{u}_8$ corresponding to the nearly rigid rotation. We notice a very small deformation for the $\vec{u}_3$-$\vec{u}_4$ phase portrait, and see that the $\vec{u}_7$-$\vec{u}_8$ phase portrait (controlling $\sim 4.6\%$ energy) is quite egg-shaped and has period 2.

The Coefficients a_{ik} — Hopping State

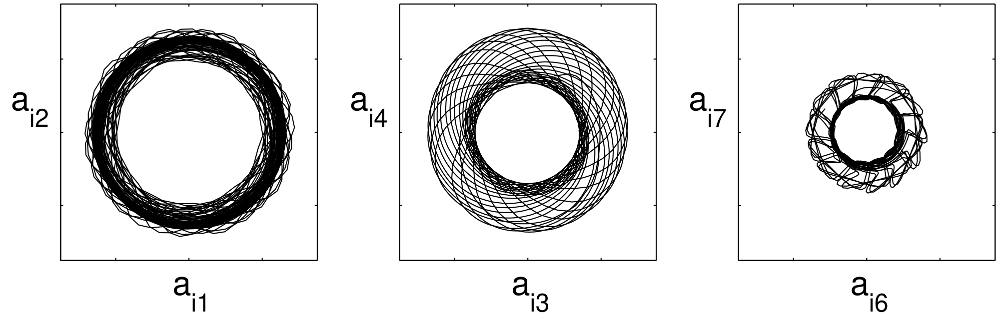


Figure: The phase plots for the hopping state looks very different. The three pairs: $\vec{u}_1$-$\vec{u}_2$, $\vec{u}_3$-$\vec{u}_4$, and $\vec{u}_7$-$\vec{u}_8$ all display quasi-periodic behavior.

The comparison of the phase-diagrams for the nearly rigid rotation and the hopping state is the most straight-forward way of classifying (and distinguishing) these dynamic patterns.

Looking at all the phase-diagrams for motions in the range $R \in [7.3600, 7.7475]$ may give us an insight into how the hopping state is "born."

Analyzing Other Types of Data

Clearly, the SVD does not care what kind of data we encode in the matrix A, we can think of many applications...

$\tilde{f}_i =$ Passport/DMV photographs (face recognition)

$\tilde{f}_i =$ Finger-prints

$\tilde{f}_i =$ DNA-(sub)sequence — Gattaca

$\tilde{f}_i =$ Multiple simultaneous temperature readings

$\tilde{f}_i =$ Demographic data

$\tilde{f}_i =$ Netflix data

$\tilde{f}_i =$ Purchase history

For time-dependent data, we can look that the phase-portraits; for other types of data, the k-tuple of coefficients $(a_{i1}, \dots, a_{ik})$ defines a **"signature"** of $\vec{f}_i$ expressed in the orthogonal basis. The signature may be useful for identification purposes.

Principal Component Analysis

Since Principal Component Analysis is the main(?) application area of the SVD, we should probably say something about it?

We borrow from [Wikipedia]...

> *"Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called principal components."*

Principal Component Analysis

"PCA can be done by eigenvalue decomposition of a data covariance (or correlation) matrix or singular value decomposition of a data matrix"

"$X^T X$ itself can be recognized as proportional to the empirical sample covariance matrix of the dataset X."

From our experience, conditioning strongly "suggests" we compute `svd`(A), where $\kappa(A) = \sigma_1/\sigma_n$, since the problem `eig`($A^*A$) suffers from $\kappa(A^*A) = (\sigma_1/\sigma_n)^2$.

We note (formally)

$$A^*A = X \Lambda X^{-1}, \quad A = U \Sigma V^* \Leftrightarrow A^*A = V \Sigma^2 V^*$$

which means that the right singular vectors (columns of V) contain the principal components.

Principal Component Analysis

Using $A = U \Sigma V^*$, the *Score Matrix* $T = U \Sigma$. (incidentally, U and Σ form a Polar Decomposition [Math 524 (Notes#7.2)] of T).

"Full" Principal Component Analysis involves (among other things) making sure your data is properly organized and scaled. It is common to extract the mean values, and describe the variations from the mean in terms of z-scores.

Whereas the "core" computation is "just the SVD," the rest of the statistical explanations are best left to a statistician!