There are a variety of ways to express the $m \times n$ matrix $\mathbf{A}$ as a sum of rank one matrices. The bottom line is this: *the Nth partial sum captures as much of the matrix $\mathbf{A}$ as possible.* Thus the partial sum of the rank one matrices is an important object to consider. This leads to the following theorem:

Theorem: *For any N so that $0 \leq N \leq r$, we can define the partial sum*

$$\mathbf{A}_N = \sum_{j=1}^{N} \sigma_j \mathbf{u}_j \mathbf{v}_j^* \,. \tag{14.2.103}$$

And if $N = \min\{m, n\}$, define $\sigma_{N+1} = 0$. Then

$$\|A - A_N\|_2 = \sigma_{N+1} \,. \tag{14.2.104}$$

Likewise, if using the Frobenius norm, then

$$\|A - A_N\|_N = \sqrt{\sigma_{N+1}^2 + \sigma_{N+2}^2 + \cdots + \sigma_r^2} \,. \tag{14.2.105}$$

Interpreting this theorem is critical. Geometrically, we can ask *what is the best approximation of a hyperellipsoid by a line segment? Simply take the line segment to be the longest axis, i.e. that associated with the singular value σ_1.* Continuing this idea, what is the best approximation by a two-dimensional ellipse? Take the longest and second longest axes, i.e. those associated with the singular values σ_1 and σ_2. After r steps, the total energy in $\mathbf{A}$ is completely captured. **Thus the SVD gives a type of least-square fitting algorithm, allowing us to project the matrix onto low-dimensional representations in a formal, algorithmic way.** Herein lies the ultimate power of the method.

14.3 Introduction to Principal Component Analysis (PCA)

To make explicit the concept of the SVD, a simple model example will be formulated that will illustrate all the key concepts associated with the SVD. The model to be considered will be a simple spring-mass system as illustrated in Fig. 134. Of course, this is a fairly easy problem to solve from basic concepts of $\mathbf{F} = m\mathbf{a}$. But for the moment, let's suppose we didn't know the governing equations. In fact, our aim in this lecture is to use a number of cameras (probes) to extract out data concerning the behavior of the system and then to try extract empirically the governing equations of motion.

This prologue highlights one of the key applications of the SVD, or alternatively a *Principle Component Analysis* (PCA), *Proper Mode Decomposition* (POD), or *Karhunen-Loéve Decomposition* as it also known in the literature. Namely, from seemingly complex, perhaps random, data, can low-dimensional reductions of the dynamics and behavior be produced when the governing equations are not known? Such methods can be used to quantify low-dimensional

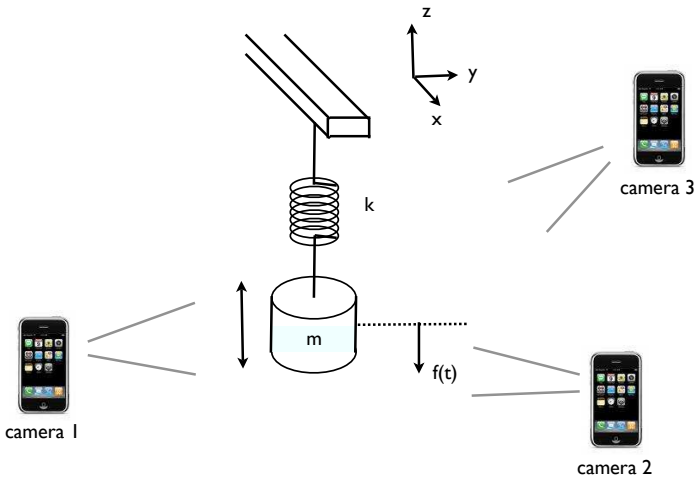

Figure 134: A prototypical example of how we might apply a principal component analysis, or SVD, is the simple mass-spring system exhibited here. The mass m is suspended with a spring with Hook's constant k. Three video cameras collect data about its motion in order to ascertain its governing equations.

dynamics arising in such areas as turbulent fluid flows [23], structural vibrations [24, 25], insect locomotion [26], damage detection [27], and neural decision making strategies [28] to name just a few areas of application. It will also become obvious as we move forward on this topic that the breadth of applications is staggering and includes image processing and signal analysis. Thus the perspective to be taken here is clearly one in which the data analysis of an unknown, but potentially low-dimensional system is to be analyzed.

Again we turn our attention to the simple experiment at hand: a mass suspended by a spring as depicted in Fig. 134. If the mass is perturbed or taken from equilibrium in the z-direction only, we know that the governing equations are simply

$$\frac{d^2 f(t)}{dt^2} = -\omega^2 f(t) \tag{14.3.106}$$

where the function $f(t)$ measures the displacement of the mass in the z-direction as a function of time. This has the well known solution (in amplitude-phase form)

$$f(t) = A\cos(\omega t + \omega_0) \tag{14.3.107}$$

where the values of A and ω_0 are determined from the initial state of the system. This essentially states that the state of the system can be described by a one-

degree of freedom system.

In the above analysis, there are many things we have ignored. Included in the list of things we have ignored is the possibility that the initial excitation of the system actually produces movement in the $x-y$ plane. Further, there is potentially noise in the data from, for instance, shaking of the cameras during the video capture. Moreover, from what we know of the solution, only a single camera is necessary to capture the underlying motion. In particular, a single camera in the $x-y$ plane at $z=0$ would be ideal. Thus we have over-sampled the data with three cameras and have produced redundant data sets. From all of these potential perturbations and problems, it is our goal to extract out the simple solution given by the simple harmonic motion.

This problem is a good example of what kind of processing is required to analyze a realistic data set. Indeed, one can imagine that most data will be quite noisy, perhaps redundant, and certainly not produced from an optimal viewpoint. But through the process of PCA, these can be circumvented in order to extract out the ideal or simplified behavior. Moreover, we may even learn how to transform the data into the optimal viewpoint for analyzing the data.

Data collection and ordering

Assume now that we have started the mass in motion by applying a small perturbation in the z-direction only. Thus the mass will begin to oscillate. Three cameras are then used to record the motion. Each camera produces a two-dimensional representation of the data. If we denote the data from the three cameras with subscripts a, b and c, then the data collected are represented by the following:

$$\text{camera 1: } (\mathbf{x}_a, \mathbf{y}_a) \tag{14.3.108a}$$

$$\text{camera 2: } (\mathbf{x}_b, \mathbf{y}_b) \tag{14.3.108b}$$

$$\text{camera 3: } (\mathbf{x}_c, \mathbf{y}_c) \tag{14.3.108c}$$

where each set $(\mathbf{x}_j, \mathbf{y}_j)$ is data collected over time of the position in the $x-y$ plane of the camera. Note that this is not the same $x-y$ plane of the oscillating mass system as shown in Fig. 134. Indeed, we should pretend we don't know the correct $x-y-z$ coordinates of the system. Thus the camera positions and their relative $x-y$ planes are arbitrary. The length of each vector $\mathbf{x}_i$ and $\mathbf{y}_i$ depends on the data collection rate and the length of time the dynamics is observed. We denote the length of these vectors as n.

All the data collected can then be gathered into a single matrix:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_a \\ \mathbf{y}_a \\ \mathbf{x}_b \\ \mathbf{y}_b \\ \mathbf{x}_c \\ \mathbf{y}_c \end{bmatrix} \tag{14.3.109}$$

Thus the matrix $X \in \mathbb{R}^{m\times n}$ where m represents the number of measurement types and n is the number of data points taken from the camera over time.

Now that the data has been arranged, two issues must be addressed: noise and redundancy. Everyone has an intuitive concept that noise in your data can only deteriorate, or corrupt, your ability to extract the true dynamics. Just as in image processing, noise can alter an image beyond restoration. Thus there is also some idea that if the measured data is too noisy, fidelity of the underlying dynamics is compromised from a data analysis point of view. The key measure of this is the so-called signal-to-noise ratio: SNR$= \sigma^2_{\text{signal}}/\sigma^2_{\text{noise}}$, where the ratio is given as the ratio of variances of the signal and noise fields. A high SNR (much greater than unity) gives almost noiseless (high precision) data whereas a low SNR suggests the underlying signal is corrupted by the noise. The second issue to consider is redundancy. In the example of Fig. 134, the single degree of freedom is sampled by three cameras, each of which is really recording the same single degree of freedom. Thus the measurements should be rife with redundancy, suggesting that the different measurements are statistically dependent. Removing this redundancy is critical for data analysis.

The covariance matrix

An easy way to identify redundant data is by considering the covariance between data sets. Recall from our early lectures on probability and statistics that the covariance measures the statistical dependence/independence between two variables. Obviously, strongly statistically dependent variables can be considered as redundant observations of the system. Specifically, consider two sets of measurements with zero means expressed in row vector form:

$$\mathbf{a} = [a_1\ a_2\ \cdots\ a_n] \text{ and } \mathbf{b} = [b_1\ b_2\ \cdots\ b_n] \tag{14.3.110}$$

where the subscript denotes the sample number. The variances of $\mathbf{a}$ and $\mathbf{b}$ are given by

$$\sigma^2_{\mathbf{a}} = \frac{1}{n-1}\mathbf{a}\mathbf{a}^T \tag{14.3.111a}$$

$$\sigma^2_{\mathbf{b}} = \frac{1}{n-1}\mathbf{b}\mathbf{b}^T \tag{14.3.111b}$$

while the covariance between these two data sets is given by

$$\sigma^2_{\mathbf{ab}} = \frac{1}{n-1}\mathbf{a}\mathbf{b}^T \tag{14.3.112}$$

where the normalization constant of $1/(n-1)$ is for an unbiased estimator.

We don't just have two vectors, but potentially quite a number of experiments and data that would need to be correlated and checked for redundancy. In fact, the matrix in Eq. (14.3.109) is exactly what needs to be checked for covariance. The appropriate *covariance matrix* for this case is then

$$\mathbf{C_X} = \frac{1}{n-1}\mathbf{X}\mathbf{X}^T\,. \tag{14.3.113}$$

This is easily computed with MATLAB from the command line:

```
cov(X)
```

The covariance matrix $\mathbf{C_X}$ is a square, symmetric $m \times m$ matrix whose diagonal represents the variance of particular measurements. The off-diagonal terms are the covariances between measurement types. Thus $\mathbf{C_X}$ captures the correlations between all possible pairs of measurements. Redundancy is thus easily captured since if two data sets are identical (identically redundant), the off-diagonal term and diagonal term would be equal since $\sigma^2_{\mathbf{ab}} = \sigma^2_{\mathbf{a}} = \sigma^2_{\mathbf{b}}$ if $\mathbf{a} = \mathbf{b}$. Thus large diagonal terms correspond to redundancy while small diagonal terms suggest that the two measured quantities are close to statistically independent and have low redundancy. It should also be noted that large diagonal terms, or those with large variances, typically represent what we might consider *the dynamics of interest* since the large variance suggests strong fluctuations in that variable. Thus the covariance matrix is the key component to understanding the entire data analysis.

Achieving the goal: The insight given by the covariance matrix leads to our ultimate aim of

i) removing redundancy
ii) identifying those signals with maximal variance.

Thus in a mathematical sense, we are simply asking to represent $\mathbf{C_X}$ so that the diagonals are ordered from largest to smallest and the off-diagonals are zero, i.e. our task is to diagonalize the covariance matrix. This is *exactly* what the SVD does, thus allowing it to becomes the tool of choice for data analysis and dimensional reduction. In fact, the SVD diagonalizes and each singular directions captures as much energy as possible as measured by the singular values σ_j.

14.4 Principal Components, Diagonalization and SVD

The example presented in the previous section shows that the key to analyzing a given experiment is to consider the covariance matrix

$$\mathbf{C_X} = \frac{1}{n-1}\mathbf{X}\mathbf{X}^T . \tag{14.4.114}$$

where the matrix $\mathbf{X}$ contains the experimental data of the system. In particular, $\mathbf{X} \in \mathbb{C}^{m\times n}$ where m are the number of probes or measuring positions, and n is the number of experimental data points taken at each location.

In this setup, the following facts are highlighted:

- $\mathbf{C_X}$ is a square, symmetric $m \times m$ matrix.
- The diagonal terms of $\mathbf{C_X}$ are the measure variance for particular measurements. By assumption, large variances correspond to *dynamics of interest*, whereas low variances are assumed to correspond to *non-interesting dynamics*.
- The off-diagonal terms of $\mathbf{C_X}$ are the covariance between measurements. Indeed, the off-diagonals captures the correlations between all possible pairs of measurements. A large off-diagonal term represents two events that have a high-degree of redundancy, whereas a small off-diagonal coefficient means there is little redundancy in the data, i.e. they are statistically independent.

Diagonalization

The concept of diagonalization is critical for understanding the underpinnings of many physical systems. In this process of diagonalization, the correct coordinates, or basis functions, are revealed that reduce the given system to its low-dimensional essence. There is more than one way to diagonalize a matrix, and this is certainly true here as well since the constructed covariance matrix $\mathbf{C_X}$ is square and symmetric, both properties that are especially beneficial for standard eigenvalue/eigenvector expansion techniques.

The key idea behind the diagonalization is simply this: there exists an *ideal* basis in which the $\mathbf{C_X}$ can be written (diagonalized) so that in this basis, all redundancies have been removed, and the largest variances of particular measurements are ordered. In the language being developed here, this means that the system has been written in terms of its *principle components*, or in a *proper orthogonal decomposition*.

Eigenvectors and eigenvalues: The most straightforward way to diagonalize the covariance matrix is by making the observation that $\mathbf{X}\mathbf{X}^T$ is a square, symmetric $m \times m$ matrix, i.e. it is self-adjoint so that the m eigenvalues are real

and distinct. Linear algebra provides theorems which state the such a matrix can be rewritten as

$$\mathbf{X}\mathbf{X}^T = \mathbf{S}\mathbf{\Lambda}\mathbf{S}^{-1} \tag{14.4.115}$$

as stated in Eq. (14.2.94) where the matrix $\mathbf{S}$ is a matrix of the eigenvectors of $\mathbf{X}\mathbf{X}^T$ arranged in columns. Since it is a symmetric matrix, these eigenvector columns are orthogonal so that ultimately the $\mathbf{S}$ can be written as a unitary matrix with $\mathbf{S}^{-1} = \mathbf{S}^T$. Recall that the matrix $\mathbf{\Lambda}$ is a diagonal matrix whose entries correspond to the m distinct eigenvalue of $\mathbf{X}\mathbf{X}^T$.

This suggests that instead of working directly with the matrix $\mathbf{X}$, we consider working with the transformed variable, or in the principal component basis,

$$\mathbf{Y} = \mathbf{S}^T\mathbf{X}\,. \tag{14.4.116}$$

For this new basis, we can then consider its covariance

$$\begin{aligned}
\mathbf{C_Y} &= \frac{1}{n-1}\mathbf{Y}\mathbf{Y}^T \\
&= \frac{1}{n-1}(\mathbf{S}^T\mathbf{X})(\mathbf{S}^T\mathbf{X})^T \\
&= \frac{1}{n-1}\mathbf{S}^T(\mathbf{X}\mathbf{X}^T)\mathbf{S} \\
&= \frac{1}{n-1}\mathbf{S}^T\mathbf{S}\mathbf{\Lambda}\mathbf{S}\mathbf{S}^T \\
\mathbf{C_Y} &= \frac{1}{n-1}\mathbf{\Lambda}\,.
\end{aligned} \tag{14.4.117}$$

In this basis, the *principal components* are the eigenvectors of $\mathbf{X}\mathbf{X}^T$ with the interpretation that the jth diagonal value of $\mathbf{C_Y}$ is the variance of $\mathbf{X}$ along $\mathbf{x}_j$, the jth column of $\mathbf{S}$. The following codes of line produce the principal components of interest.

```
[m,n]=size(X);   %  compute data size
mn=mean(X,2); %  compute mean for each row
X=X-repmat(mn,1,n);  % subtract mean

Cx=(1/(n-1))*X*X';   % covariance
[V,D]=eig(Cx);      % eigenvectors(V)/eigenvalues(D)
lambda=diag(D);    % get eigenvalues

[dummy,m_arrange]=sort(-1*lambda);  % sort in decreasing order
lambda=lambda(m_arrange);
V=V(:,m_arrange);

Y=V'*X;  % produce the principal components projection
```

This simple code thus produces the eigenvalue decomposition and the projection of the original data onto the principal component basis.

Singular value decomposition: A second method for diagonalizing the covariance matrix is the SVD method. In this case, the SVD can diagonalize any matrix by working in the appropriate pair of bases $\mathbf{U}$ and $\mathbf{V}$ as outlined in the first lecture of this section. Thus by defining the transformed variable

$$\mathbf{Y} = \mathbf{U}^*\mathbf{X} \tag{14.4.118}$$

where $\mathbf{U}$ is the unitary transformation associated with the SVD: $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$. Just as in the eigenvalue/eigenvector formulation, we then compute the variance in $\mathbf{Y}$:

$$\begin{aligned}
\mathbf{C_Y} &= \frac{1}{n-1}\mathbf{Y}\mathbf{Y}^T \\
&= \frac{1}{n-1}(\mathbf{U}^*\mathbf{X})(\mathbf{U}^*\mathbf{X})^T \\
&= \frac{1}{n-1}\mathbf{U}^*(\mathbf{X}\mathbf{X}^T)\mathbf{U} \\
&= \frac{1}{n-1}\mathbf{U}^*\mathbf{U}\mathbf{\Sigma}^2\mathbf{U}\mathbf{U}^* \\
\mathbf{C_Y} &= \frac{1}{n-1}\mathbf{\Sigma}^2 .
\end{aligned} \tag{14.4.119}$$

This makes explicit the connection between the SVD and the eigenvalue method, namely that $\mathbf{\Sigma}^2 = \mathbf{\Lambda}$. The following codes of line produce the principal components of interest using the SVD (Assume that the you have the first three lines from the previous MATLAB code).

```
[u,s,v]=svd(X'/sqrt(n-1));  % perform the SVD
lambda=diag(s).^2;  % produce diagonal variances
Y=u'*X;  % produce the principal components projection
```

This gives the SVD method for producing the principal components. Overall, the SVD method is the more robust method and should be used. However, the connection between the two methods becomes apparent in these calculations.

Spring Experiment

To illustrate this completely in practice, three experiments will be performed in class with the configuration of Fig. 134. The following experiments will attempt to illustrate various aspects of the PCA and its practical usefulness and the effects of noise on the PCA algorithms.

- **Ideal case:** Consider a small displacement of the mass in the z direction and the ensuing oscillations. In this case, the entire motion is in the z directions with simple harmonic motion being observed.
- **noisy case:** Repeat the ideal case experiment, but this time, introduce camera shake into the video recording. This should make it more difficult to extract the simple harmonic motion. But if the shake isn't too bad, the dynamics will still be extracted with the PCA algorithms.
- **horizontal displacement:** In this case, the mass is released off-center so as to produce motion in the $x-y$ plane as well as the z direction. Thus there is both a pendulum motion and a simple harmonic oscillations. See what the PCA tells us about the system.

In order to extract out the mass movement from the video frames, the following MATLAB code is needed. This code is a generic way to read in movie files to MATLAB for post-processing.

```
obj=mmreader('matlab_test.mov')

vidFrames = read(obj);
numFrames = get(obj,'numberOfFrames');

for k = 1 : numFrames
    mov(k).cdata = vidFrames(:,:,:,k);
    mov(k).colormap = [];
end

for j=1:numFrames
  X=frame2im(mov(j));
  imshow(X); drawnow
end
```

This command multi-media reader command **mmreader** simply characterizes the file type and its attributes. The bulk of time in this is the **read** command which actually uploads the movie frames into the MATLAB desktop for processing. Once the frames are extracted, they can again be converted to double precision numbers for mathematical processing. In this case, the position of the mass is to be determined from each frame. This basic shell of code is enough to begin the process of extracting the spring-mass system information.

14.5 Principal Components and Proper Orthogonal Modes

Now that the basic framework of the principal component analysis and its relation to the SVD has been laid down, a few remaining issues need to be addressed. In particular, the principal component analysis seems to suggest that

we are simply expanding our solution in another *orthonormal* basis, one which can always diagonalize the underlying system.

Mathematically, we can consider a given function $f(x,t)$ over a domain of interest. In most applications, the variables x and t will refer to the standard space-time variables we are familiar with. The idea is to then expand this function in some basis representation so that

$$f(x,t) \approx \sum_{j=1}^{N} a_j(t)\phi_j(x) \tag{14.5.120}$$

where N is the total number of modes, $\phi_j(x)$, to be used. The remaining function $a_j(t)$ determines the weights of the spatial modes.

The expansion (14.5.120) is certainly not a new idea to us. Indeed, we have been using this concept extensively already with Fourier transforms, for instance. Specifically, here are some of the more common expansion bases used in practice:

$$\phi_j(x) = (x - x_0)^j \quad \text{Taylor expansion} \tag{14.5.121a}$$
$$\phi_j(x) = \cos(jx) \quad \text{Discrete cosine transform} \tag{14.5.121b}$$
$$\phi_j(x) = \sin(jx) \quad \text{Discrete sine transform} \tag{14.5.121c}$$
$$\phi_j(x) = \exp(jx) \quad \text{Fourier transform} \tag{14.5.121d}$$
$$\phi_j(x) = \psi_{a,b}(x) \quad \text{Wavelet transform} \tag{14.5.121e}$$
$$\phi_j(x) = \phi_{\lambda_j}(x) \quad \text{Eigenfunction expansion} \tag{14.5.121f}$$

where $\phi_{\lambda_j}(x)$ are eigenfunctions associated with the underlying system. This places the concept of a basis function expansion in familiar territory. Further, it shows that such a basis function expansion is not unique, but rather can potentially have an infinite number of different possibilities.

As for the weighting coefficients $a_j(t)$, they are simply determined from the standard inner product rules and the fact that the basis functions are orthonormal (Note that they are not written this way above):

$$\int \phi_j(x)\phi_n(x)dx = \begin{cases} 1 & j = n \\ 0 & j \neq n \end{cases} . \tag{14.5.122}$$

This then gives for the coefficients

$$a_j(t) = \int f(x,t)\phi_j(x)dx \tag{14.5.123}$$

and the basis expansion is completed.

Interestingly enough, the basis functions selected are chosen often for simplicity and/or their intuitive meaning for the system. For instance, the Fourier transform very clearly highlights the fact that the given function has a representation in terms of *frequency* components. This is fundamental for understanding

many physical problems as there is clear and intuitive meaning to the Fourier modes.

In contrast to selecting basis functions for simplicity or intuitive meaning, the broader question can be asked: what criteria should be used for selecting the functions $\phi_j(x)$. This is an interesting question given that any complete basis can approximate the function $f(x,t)$ to any desired accuracy given N large enough. But what is desired is the best basis functions such that, with N as small as possible, we achieve the desired accuracy. The goal is then the following: find a sequence of orthonormal basis functions $\phi_j(x)$ so that first two terms gives the best two-term approximation to $f(x,t)$, or the first five terms gives the best five-term approximation to $f(x,t)$. These special, ordered, orthonormal functions are called the *proper orthogonal modes* (POD) for the function $f(x,t)$. With these modes, the expansion (14.5.123) is called the POD of $f(x,t)$. The relation to the SVD will become obvious momentarily.

Example 1: Consider an approximation to the following surface

$$f(x,t) = e^{-|(x-0.5)(t-1)|} + \sin(xt) \quad x \in [0,1],\, t \in [0,2]. \tag{14.5.124}$$

As listed above, there are a number of methods available for approximating this function using various basis functions. And all of the ones listed are guaranteed to converge to the surface for a large enough N in (14.5.120).

In the POD method, the surface is first discretized and approximated by a finite number of points. In particular, the following discretization will be used:

```
x=linspace(0,1,25);
t=linspace(0,2,50);
```

where x has been discretized into 25 points and t is discretized into 50 points. The surface is represented in Fig. 135(a) over the domain of interest. The surface is constructed by using the **meshgrid** command as follows:

```
[T,X]=meshgrid(t,x);
f=exp(-abs((X-.5).*(T-1)))+sin(X.*T);
subplot(2,2,1)
surfl(X,T,f), shading interp, colormap(gray)
```

The **surfl** produces a lighted surface that in this case is represented in grayscale. Note that in producing this surface, the matrix **f** is a 25×50 matrix so that the x-values are given as row locations while the t-values are given by column locations.

The basis functions are then computed using the SVD. Recall that the SVD produces ordered singular values and associated orthonormal basis functions that capture as much energy as possible. Thus an SVD can be performed on the matrix **f** that represents the surface.

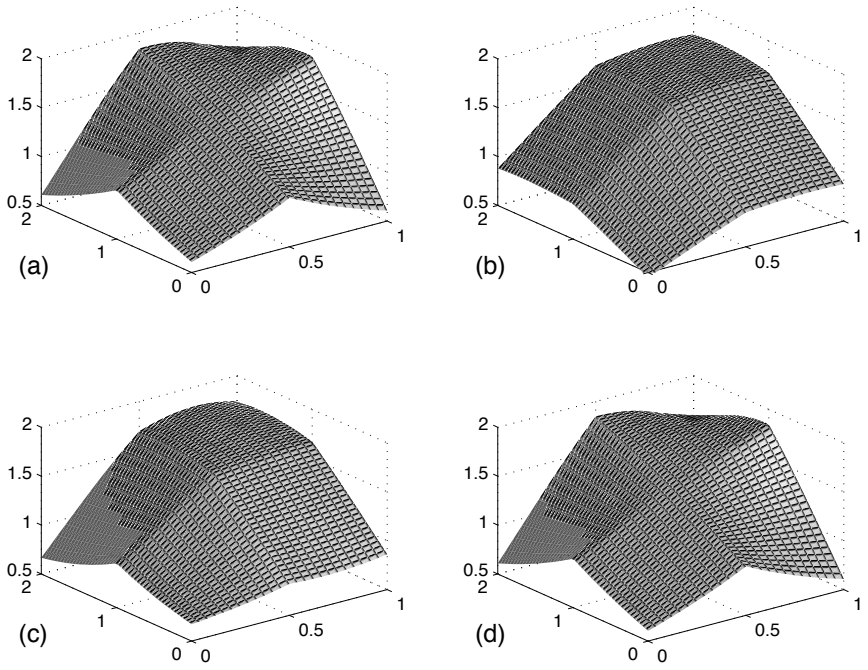

Figure 135: Representation of the surface (a) by a series of low-rank (low-dimensional) approximations with (b) one-mode, (c) two-modes and (d) three-modes. The energy captured in the first mode is approximately 92% of the total surface energy. The first three modes together capture 99.9% of the total surface energy, thus suggesting that the surface can be easily and accurately represented by a three-mode expansion.

```
[u,s,v]=svd(f);  % perform SVD

for j=1:3
   ff=u(:,1:j)*s(1:j,1:j)*v(:,1:j)'; % modal projections
   subplot(2,2,j+1)
   surfl(X,T,ff), shading interp, colormap(gray)
   set(gca,'Zlim',[0.5 2])
end
subplot(2,2,1), text(-0.5,1,0.5,'(a)','Fontsize',[14])
subplot(2,2,2), text(-0.5,1,0.5,'(b)','Fontsize',[14])
subplot(2,2,3), text(-0.5,1,0.5,'(c)','Fontsize',[14])
subplot(2,2,4), text(-0.5,1,0.5,'(d)','Fontsize',[14])
```

The SVD command pulls out the diagonal matrix along with the two unitary matrices $\mathbf{U}$ and $\mathbf{V}$. The loop above processes the sum of the first, second and

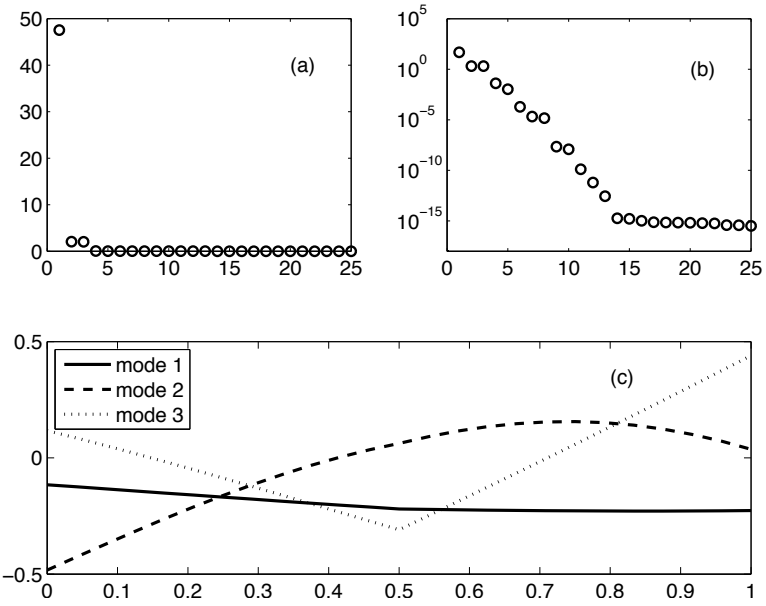

Figure 136: Singular values on a standard plot (a) and a log plot (b) showing the energy in each POD mode. For the surface considered in Fig. 135, the bulk of the energy is in the first POD mode. A three mode approximation produces 99.9% of the energy. The linear POD modes are illustrated in panel (c).

third modes. This gives the POD modes of interest. Figure 135 demonstrates the modal decomposition and the accuracy achieved with representing the surface with one, two and three POD modes. The first mode alone captures 92% of the surface while three modes produce a staggering 99.9% of the energy. This should be contrasted with Fourier methods, for instance, which would require potentially hundreds of modes to achieve such accuracy. *As stated previously, these are the best one, two and three mode approximations of the function $f(x,t)$ that can be achieved.*

To be more precise about the nature of the POD, consider the *energy* in each mode. This can be easily computed, or already has been computed, in the SVD, i.e. they are the singular values σ_j. In MATLAB, the energy in the one mode and three mode approximation is computed from

```
energy1=sig(1)/sum(sig)
energy3=sum(sig(1:3))/sum(sig)
```

More generally, the entire *spectrum* of singular values can be plotted. Figure 136 shows the complete spectrum of singular values on a standard plot and a log plot. The clear dominance of a single mode is easily deduced. Additionally, the

first three POD modes are illustrated: they are the first three columns of the matrix **U**. These constitute the orthonormal expansion basis of interest. The code for producing these plots is as follows:

```
sig=diag(s);
subplot(2,2,1), plot(sig,'ko','Linewidth',[1.5])
axis([0 25 0 50])
set(gca,'Fontsize',[13],'Xtick',[0 5 10 15 20 25])
text(20,40,'(a)','Fontsize',[13])
subplot(2,2,2), semilogy(sig,'ko','Linewidth',[1.5])
axis([0 25 10^(-18) 10^(5)])
set(gca,'Fontsize',[13],'Ytick',[10^(-15) 10^(-10) 10^(-5) 10^0 10^5],...
    'Xtick',[0 5 10 15 20 25]);
text(20,10^0,'(b)','Fontsize',[13])

subplot(2,1,2)
plot(x,u(:,1),'k',x,u(:,2),'k--',x,u(:,3),'k:','Linewidth',[2])
set(gca,'Fontsize',[13])
legend('mode 1','mode 2','mode 3','Location','NorthWest')
text(0.8,0.35,'(c)','Fontsize',[13])
```

Note that the key part of the code is simply the calculation of the POD modes which are weighted, in practice, by the singular values and the their evolution in time as given by the columns of **V**.

Example 2: Consider the following time-space function

$$f(x,t) = [1 - 0.5\cos 2t]\mathrm{sech}x + [1 - 0.5\sin 2t]\mathrm{sech}x\tanh x\,. \qquad (14.5.125)$$

This represents an asymmetric, time-periodic breather of a localized solution. Such solutions arise in a myriad of contexts are often obtained via full numerical simulations of an underlying physical system. Let's once again apply the techniques of POD analysis to see what is involved in the dynamics of this system. A discretization of the system if first applied and made into a two-dimensional representation in time and space.

```
x=linspace(-10,10,100);
t=linspace(0,10,30);
[X,T]=meshgrid(x,t);
f=sech(X).*(1-0.5*cos(2*T))+(sech(X).*tanh(X)).*(1-0.5*sin(2*T));
subplot(2,2,1), waterfall(X,T,f), colormap([0 0 0])
```

Note that unlike before the matrix **f** is now a 30×100 matrix so that the x-values are along the columns and t-values are along the rows. This is done simply so that we can use the **waterfall** command in MATLAB.

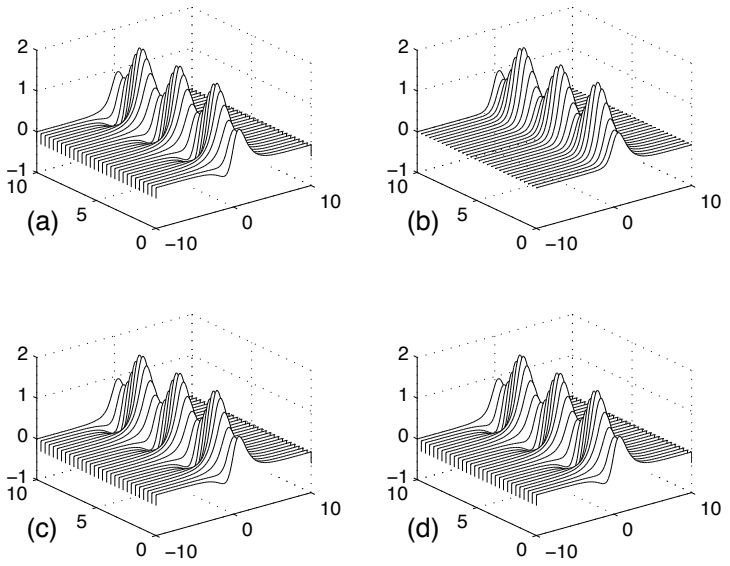

Figure 137: Representation of the time-space dynamics (a) by a series of low-rank (low-dimensional) approximations with (b) one-mode, (c) two-modes and (d) three-modes. The energy captured in the first mode is approximately 83% of the total energy. The first two modes together capture 100% of the surface energy, thus suggesting that the surface can be easily and accurately represented by a simple two-mode expansion.

The singular value decomposition of the matrix **f** produces the quantities of interest. In this case, since we want the x-value in the rows, the transpose of the matrix is considered in the SVD.

```
[u,s,v]=svd(f');
for j=1:3
  ff=u(:,1:j)*s(1:j,1:j)*v(:,1:j)';
  subplot(2,2,j+1)
  waterfall(X,T,ff'), colormap([0 0 0]), set(gca,'Zlim',[-1 2])
end
subplot(2,2,1), text(-19,5,-1,'(a)','Fontsize',[14])
subplot(2,2,2), text(-19,5,-1,'(b)','Fontsize',[14])
subplot(2,2,3), text(-19,5,-1,'(c)','Fontsize',[14])
subplot(2,2,4), text(-19,5,-1,'(d)','Fontsize',[14])
```

This produces the original function along with the first three modal approxima-

tions of the function. As is shown in Fig. 137, the two-mode and three-mode approximations appear to be identical. In fact, they are. The singular values of the SVD show that ≈83% of the energy is in the first mode of Fig. 137(a) while 100% of the energy is captured by a two mode approximation as shown in Fig. 137(b). Thus the third mode is completely unnecessary. The singular values are produced with the following lines of code.

```
figure(2)
sig=diag(s);
subplot(3,2,1), plot(sig,'ko','Linewidth',[1.5])
axis([0 25 0 50])
set(gca,'Fontsize',[13],'Xtick',[0 5 10 15 20 25])
text(20,40,'(a)','Fontsize',[13])
subplot(3,2,2), semilogy(sig,'ko','Linewidth',[1.5])
axis([0 25 10^(-18) 10^(5)])
set(gca,'Fontsize',[13],'Ytick',[10^(-15) 10^(-10) 10^(-5) 10^0 10^5],...
  'Xtick',[0 5 10 15 20 25]);
text(20,10^0,'(b)','Fontsize',[13])

energy1=sig(1)/sum(sig)
energy2=sum(sig(1:2))/sum(sig)
```

As before, we can also produce the first two modes, which are the first two columns of **U**, along with their time behavior, which are the first two columns of **V**.

```
subplot(3,1,2)  % spatial modes
plot(x,u(:,1),'k',x,u(:,2),'k--','Linewidth',[2])
set(gca,'Fontsize',[13])
legend('mode 1','mode 2','Location','NorthWest')
text(8,0.35,'(c)','Fontsize',[13])

subplot(3,1,3)  % time behavior
plot(t,v(:,1),'k',t,v(:,2),'k--','Linewidth',[2])
text(9,0.35,'(d)','Fontsize',[13])
```

Figure 138 shows the singular values along with the spatial and temporal behavior of the first two modes. It is not surprising that the SVD characterizes the total behavior of the system as an exactly (to numerical precision) two mode behavior. Recall that is exactly what we started with! Note however, that our original two modes are different than the SVD modes. Indeed, the first SVD mode is asymmetric unlike our symmetric hyperbolic secant.

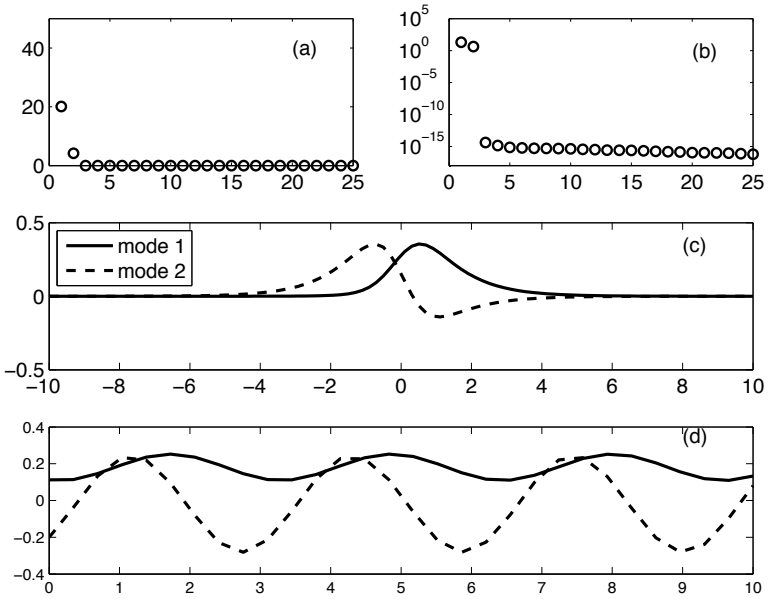

Figure 138: Singular values on a standard plot (a) and a log plot (b) showing the energy in each POD mode. For the space-time surface considered in Fig. 137, the bulk of the energy is in the first POD mode. A two mode approximation produces 100% of the energy. The linear POD modes are illustrated in panel (c) which their time evolution behavior in panel (d).

Summary and Limits of SVD/PCA/POD

The methods of PCA and POD, which are essentially related to the idea of diagonalization of any matrix via the SVD, are exceptionally powerful tools and methods for the evaluation data driven systems. Further, they provide the most precise method for performing low-dimensional reductions of a given system. A number of key steps are involved in applying the method to experimental or computational data sets.

- Organize the data into a matrix $\mathbf{A} \in \mathbb{C}^{m\times n}$ where m is the number of measurement types (or probes) and n is the number of measurements (or trials) taken from each probe.
- Subtract off the mean for each measurement type or row $\mathbf{x}_j$.
- Compute the SVD and singular values of the covariance matrix to determine the principal components.

If considering a fitting algorithm, then the POD method is the appropriate technique to consider and the SVD can be applied directly to the $\mathbf{A} \in \mathbb{C}^{m\times n}$

matrix. This then produces the singular values as well as the POD modes of interest.

Although extremely powerful, and seemingly magical in its ability to produce insightful quantification of a given problem, these SVD methods are not without limits. Some fundamental assumptions have been made in producing the results thus far, the most important being the assumption of *linearity*. Recently some efforts to extend the method using nonlinearity prior to the SVD have been explored [30, 31, 32]. A second assumption is that larger variances (singular values) represent more important dynamics. Although generally this is believed to be true, there are certainly cases where this assumption can be quite misleading. Additionally, in constructing the covariance matrix, it is assumed that the mean and variance are sufficient statistics for capturing the underlying the dynamics. It should be noted that only exponential (Gaussian) probability distributions are completely characterized by the first two moments of the data. This will be considered further in the next section on independent component analysis. Finally, it is often the case that PCA may not produce the optimal results. For instance, consider a point on a propeller blade. Viewed from the PCA framework, two orthogonal components are produced to describe the circular motion. However, this is really a one-dimensional system that is solely determined by the phase variable. The PCA misses this fact unless use is made of the information already known, i.e. that a polar coordinate system with fixed radius is the appropriate context to frame the problem. Thus blindly applying the technique can also lead to non-optimal results that miss obvious facts.

15 Independent Component Analysis

The concept of principal components or of a proper orthogonal decomposition are fundamental to data analysis. Essentially, there is an assertion, or perhaps a hope, that underlying seemingly complex and unordered data, there is in fact, some characteristic, and perhaps low-dimensional ordering of the data. The singular value decomposition of a matrix, although a mathematical idea, was actually a formative tool for the interpretation and ordering of the data. In this section on *independent component analysis* (ICA), the ideas turn to data sets for which there are more than one statistically independent objects in the data. ICA provides a foundational mathematical method for extracting interleaved data sets in a fairly straightforward way. Its applications are wide and varied, but our primary focus will be application of ICA in the context of image analysis.

15.1 The concept of independent components

Independent component analysis (ICA) is a powerful tool that extends the concepts of PCA, POD and SVD. Moreover, you are already intuitively familiar with the concept as some of the examples here will show. A simple way to