Coding Project 3: Principal Component Analysis of a Mass-on-a-spring System

Wietse Vaes

Abstract

People love collecting data. That's a fact. But is there something as too much data? Someone with three phones decided to film a mass on a spring experiment from three different angles in order to solve a mass-on-a-spring system described by differential equations. One might think this is overkill, but using the benefits of a simple decomposition we will be able to figure out which vectors describe these three linearly independent movements! In doing so, we create a better way of redimensionalising a six dimensional setting into a three dimensional setting.

1 Introduction

One of the most important sectors in the world is the big data sector. They try to analyze data and make predictions on everything. This can go from geological data to social media to healthcare. But, what usually happens when you have a lot of something, you might have redundant data, data that can easily be found using another set in the data. The height and body weight are linearly dependent, thus with one, we might be able to describe the data. Another example is the motion of an object. Looking at this object from three different angles and recording data, would give you data that would probably tell you the same thing: the height and it's movements in an x-y plane. Therefore we would like to go from a high-dimensional data set to a lower dimensional data set. One might think there's no other way to do this than to throw away data, but there is! In this paper we will go over Principal component analysis (PCA) and how to use it using a singular value decomposition of a data set.

This method will be used on a data set gathered by professor Nathan J. Kutz describing the a mass on a spring. This seems to be a random movement, but maybe there is an underlying notion going on. In order to record this data set, we take videos of a certain object on a spring from three different angles, while fixing an axis setup. The setup can be found in figure 1.

Before looking at putting up this experiment, one must look at how we could mathematically describe it. The movement in the z-axis results from Newton's well known second law:

$$F = ma,$$

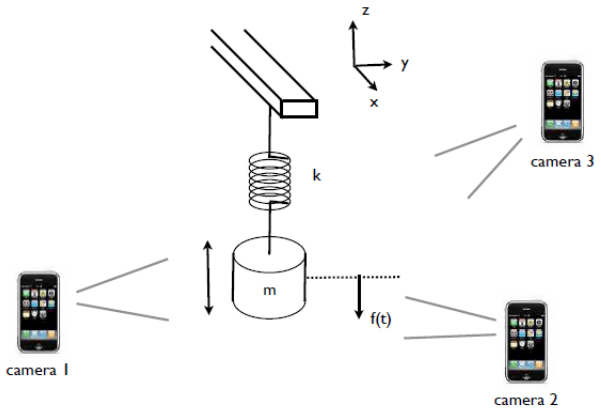

Figure 1: Object with mass m hangs on a spring with Hook's constant k and downward movement $f(t)$, with t time while fixing the axis as presented. In order to obtain data from the movement of the object, we record it from three, not necessarily perpendicular, different angles using cameras.

with F the force on an object with mass m and acceleration a. Here we have that the force worked on the mass is given by

$$F = -kz(t).$$

Since the acceleration is equal to the second derivative of $z(t)$, we get that our system might be governed by the ordinary differential equation

$$m\frac{d^2z(t)}{dt^2} + kz(t) = 0.$$

This ODE is easy to solve, but does not represent our experiment well enough. A lot of assumptions have been made, such as, ignoring friction of air, the energy loss of the spring because of warmth, the fact that our mass can also move in x and y direction with the littlest push, etc. The world is a lot more complicated that this simple ODE represents. One must ask himself: "What best describes the real world?" The answer is of course the real world itself! That is why we will be doing the experiment and since we have too high of a dimension, we must lower this by using the principal component analysis!

2 Theoretical Background

Three data sets, giving information about two movement directions each, have been gathered in order to find out how the mass on a spring moves in our three dimensional plane, therefore there must be redundant data. The set up at which the experiment is done is, however, not optimal. Therefore, using only a couple of data sets does not necessarily give us an optimal result. In order to reduce the data more optimally we look at the singular value decomposition (SVD), after which we will be able to approximate the data by an n-rank approximation. In order to further enforce that this is better than ignoring a sub group of the data, we look at the principal component analysis.

2.1 Singular Value Decomposition

There are multiple ways of representing a matrix $A \in \mathbb{C}^{m\times n}$, for example, the diagonilization of A, LU-decomposition, QU-decomposition, etc. Each has it's own pros and cons, but for data-compression/smoothing we start by looking at the singular value decomposition (SVD). This essentially describes the matrix A as the stretching/compressing and rotating vectors $\mathbf{v}_1, \ldots, \mathbf{v}_p \in \mathbb{C}^n$ into orthogonal directions $\mathbf{u}_1, \ldots \mathbf{u}_p \in \mathbb{C}^m$, with $p = \min(m, n)$. Thus,

$$A\mathbf{v}_j = \sigma_j \mathbf{u}_j, \ 1 \leq j \leq p, \tag{1}$$

here we have σ_j added such that we have that $\mathbf{u}_j$ orthonormal directions, we will call these the singular values. A compact notation of this is

$$A\hat{\mathbf{V}} = \hat{\mathbf{U}}\hat{\Sigma},$$

where the $j-$th column of $\hat{\mathbf{V}} \in \mathbb{C}^{n\times p}$ is the vector v_j, the $j-$th column of $\mathbf{U} \in \mathbb{C}^{m\times p}$ is the vector u_j and $\hat{\Sigma} \in \mathbb{R}^{p\times p}$ a diagonal matrix with the σ_j on the diagonal. It is desirable to be able to find A or Σ separately, therefore we should be able to take the inverses of $\hat{\mathbf{U}}$ and $\hat{\mathbf{V}}$ and they thus need to be square. Take as an example that $m < n$, we then have that $\hat{U}$ is already square, in order to make $\hat{V}$ square, we add $n-m$ zero columns. However, we still want the equations (1) to hold and $\hat{U}\hat{\Sigma}\hat{V}^{-1}$ to exist. Therefore, we make $\hat{\Sigma} \in \mathbb{R}^{m,n}$ by filling it with zeros. In conclusion we have

$$A = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{-1}.$$

In order to compute all of these matrices, we will assume that V and U are unitary, thus their inverse are equal to their conjugate transpose:

$$A = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{*}. \tag{2}$$

Ponder now about three things, first of all, this is not unique. Switching two columns of V and their respective two columns of U results in the same idea. Secondly, the m by n matrix hasn't been restricted by anything and thus we don't know if this decomposition, let alone with unitary matrices, even exists for any matrix. Thirdly, and more specifically to the second note, should A be invertible? From now on we will talk about a matrix A of rank r. The rank means that we are able to reduce A into an invertible sub matrix $\tilde{A} \in \mathbb{C}^{r\times r}$, therefore $\tilde{A}$ consists of r linear independent rows or columns. Therefore, we would only be able to find r orthonormal vectors u for which the equalities can hold, afterwards we will not be able to create a new orthogonal vector u since it will always be a combination of the other vectors. Say that $r < p$, the simple solution to this is to just add $p-r$ and $p-r$ zero columns to get $\hat{U}$ and $\hat{V}$. We will also have $p-r$ zeros on the diagonal of $\hat{\Sigma}$. With this reasoning, existence to the decomposition, without the assumption of unitary matrices is simple to see. Proving the unitary characteristic and realness of the σ_j is trickier and outside the scope of this paper.

Now one must also be able to compute this decomposition. This is a rather simple

process given a few tricks. Note that

$$\begin{aligned} A^*A &= (U\Sigma V^*)^* U\Sigma V^* \\ &= V\Sigma^T U^* U\Sigma V^* \\ &= V\hat{\Sigma}^2 V^*, \end{aligned}$$

and, likewise,

$$AA^* = U\hat{\Sigma}^2 U^*.$$

Multiplying both equations respectively by V and U we get 2 systems of eigenvalue problems:

$$\begin{cases} A^*A\mathbf{V} &= \mathbf{V}\hat{\Sigma}^2, \\ AA^*\mathbf{U} &= \mathbf{U}\hat{\Sigma}^2. \end{cases}$$

This gives us a way of non-negative eigenvalues, and thus we can choose non-negative σ_j and, after normalizing, unique eigenvectors u_j and v_j. In order to make this fully unique, we choose to order the σ_j in an decreasing order: $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$.

2.2 Rank-n Approximation

This σ_j can be seen as the importance of u_j and v_j^* when constructing A. This is intuitively seen when looking at the Frobenius norm of a matrix given by

$$\|A\|_F^2 = \sum_{j=1}^{r} \sigma_j^2,$$

with these sigmas the singular values. Here, since the sigmas are decreasing, we have that the Frobenius norm is best approximated using only a hand full of the first singular values. The last couple might not even add a lot in the Frobenius norm sense. Therefore we can look at approximating A using these singular values. This can be done by seeing that (2) can be written as

$$A = \sum_{j=1}^{p} \sigma_j u_j v_j^*,$$

and then refusing to use the last singular values, therefore

$$A \approx \sum_{j=1}^{n} \sigma_j u_j v_j^*,$$

with n not necesarily equal to the n from the dimension of A. We call this the rank-n approximation of A.

It is important to remember that all u_j were orthogonal and so were the v_j. Therefore, if we only include n of these orthogonal basis, we will get a data set on an n-dimensional plane in the original r-dimensional plane. We can thus also see these rank-n approximations as some sort of projections from the $r-$plane to the n-plane!

2.3 Principal Component Analysis

As shown in the previous section, the singular value decomposition is good for compressing data, but would it also be able to compress data, that may has redundant data, in an efficient way? This is especially important since most of the time there's a lot of redundant data, but it still might be important and help with noise. For this we move on to the principal component analysis (PCA). Redundant data means that a subset of the gathered data essentially informs us of the same information another subset, or combination of subsets, of the data would do. Therefore we can look at these subsets to be linear combinations of other subsets and thus not being linearly independent. Of course, since there's noise, it won't be perfect linear dependence. Say that we look at this subset of the data as a variable gather while gathering data. Therefore there would be linear dependence between these variables.

We would like to find a way to put a quantitative way of looking at this dependence and the best way to do this, is by looking at the covariance between the data sets, or variables. Assuming that this, seemingly random, data set is determined by a mean and variance, we'd like to abuse this variance. With N the length of the centralized variables $\vec{a}$ and $\vec{b}$, we have that the covariance is given by

$$\sigma_{ab}^2 = \frac{1}{N-1}\vec{a}\vec{b}^T.$$

One can also look at the variance of $\vec{a}$ by $\sigma_a^2 = \frac{1}{N-1}\vec{a}\vec{a}^T$. Given a data set $\mathbf{X}$, the covariance matrix, filled with the variances and covariances of the variables is given by

$$\Sigma_{\mathbf{X}}^2 = \frac{1}{N-1}\mathbf{X}\mathbf{X}^T,$$

where $\Sigma_{\mathbf{X}}^2 \in \mathbb{R}^{M\times M}$, with M the amount of variables. If a variable is linearly independent of another variable we have that the covariance between these will be zero. This, however, does not give us just a zero or one as a result, it gives us a range of values, making it possible to see if they are almost linearly independent, but not fully due to noise. It also tells us that if a number is large, the variance between these variables is large. This is particularly important on the variance of one variable. If this is large, we know that it varies a lot, and thus changes a lot. This makes it important to track, assuming there isn't a big amount of noise. This is essentially what SVD does, after ranking the variables by largest variance. Also note that this covariance matrix is diagonizable since it's symmetric, which is trivial, positive definite:

$$x\Sigma_{\mathbf{X}}^2 x^T = \frac{1}{N-1}x\mathbf{X}\mathbf{X}^T x^T = \frac{1}{N-1}\|x\mathbf{X}\|_2^2 \geq 0.$$

The diagonilization of the covariance matrix gives us a diagonalisation with eigenvectors and eigenvalues, where we call the eigenvalues the principal components with their respective eigenvalues, the covariance of this basis, and this is comparable to the SVD decomposition. This link can be found rather easily by substituting $\frac{1}{\sqrt{N-1}}\mathbf{X}$ by A, where A is real:

$$\Sigma_{\mathbf{X}}^2 = AA^T = U\Sigma U^T U\Sigma^T U^T = U\hat{\Sigma}^2 U^T,$$

therefore, we have that

$$\Sigma^2_{\mathbf{U^T X}} = U^T U \hat{\Sigma}^2 U^T U = \hat{\Sigma}^2.$$

In conclusion, we have that the singular values is directly related to the covariance matrix of $U^T X$, which means that the singular values are closely related to the variance of vectors in $U^T X$. Further confirming that the SVD method is good for reducing redundant data, based on using n-rank approximations.

3 Implementation

Using MATLAB and three given video clips we will now try to find and process the movement of the mass on a spring. Before doing anything, we would like to track the mass in the video itself. We do this by using code given by dr. Amin Rahman. Because of noise and his implementation, this also might influence the movement of the mass matrix which we will process. Therefore we now use the singular value decomposition on these, already processed, data.

Since the videos weren't the exact same length, we first cut off the ends of the video for which we don't know what happened on the shortest video. After which we center the data by computing the difference between every variable, the position of the mass in a direction, and it's mean. This so that we can rely on the PCA. Thereafter we put all data into one matrix and find the singular value decomposition of this matrix using function $svd()$.

Finally we compute the transformed data into vectors $U^T X$, These are all linearly independent, since the covariance matrix is a diagonal matrix, as seen in the previous section. Thereafter we calculate the energies of the singular values in order to find out which n-rank approximation we will choose as the approximation. We define the energies of the singular values as the square of the relative approximation of the n-rank approximation A_n to the true A according to the Frobenius norm:

$$n\text{-rank energy} = \frac{\|A_n\|_F^2}{\|A\|_F^2}.$$

This gives us an indication of how much information remains in the n-rank approximation. We will be choosing the n-rank approximations when 90% of the information remains. This implementation is done for all four video throuples.

4 Results

In order to interpret the results from our process, we need to understand what was being processed. Remind yourself that we received twelve video clips, four groups of three different angles, one group was relatively perfectly recorded, one had noise, one was off centered and in the last group our mass rotated while going up and down. Let us put an axis on the real world now, where z is the height from the ground, and x and y are yet to be determined, but perpendicular to the other directions. In the first of three videos, we can make out the height of the mass in the y-direction of the

video and a sideways motion on the x-y plane on the x- direction. The second clip gives us the same, but from a different angle. The last clip gives us the height in the x direction and another last angle of the x-y plane on the y direction of the video. One can look back at figure 1 to get a better perspective on the camera angles. In conclusion, our combined data set gives us 3 vectors describing the height, note that these were centered and should thus resemble each other, and 3 different angles on the x-y direction. Sadly these angles were not perpendicular from each other, thus not giving the proper motion in three dimensions.

Even though we approximated this data set by an n-rank approximation, the physical meaning of these vectors remain the same. First we show all n-rank approximations for one vector specifically: the ideal case, x-direction of the video. This can be found in figure 2.

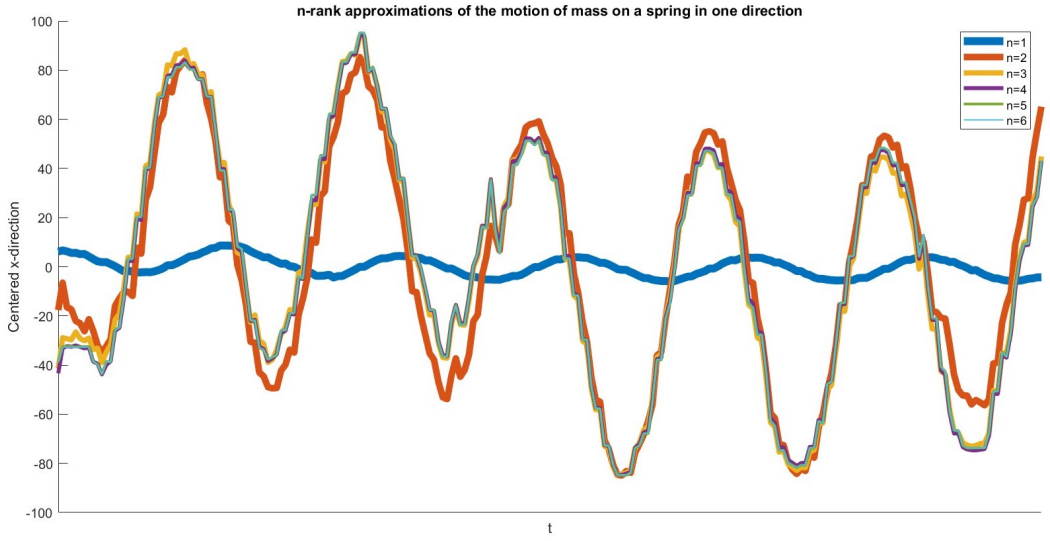

Figure 2: The approximations to a data set using n-rank approximations, in particular with SVD, is shown above. Starting out weak, a clear convergence can be seen going to the actual data. This indicates that just using the first singular value here is not enough.

Here one can clearly see that the first rank gives us a simple, not so noisy, sine like form. This is what we expect the motions to be in the x-y plane: going back and forth. Also note that we clearly see convergence of the line. The thicker the line, the least amount of singular values used. To further drive the point home that the the value of the singular values is important we compare result using only the highest and lowest singular value in figure 3.

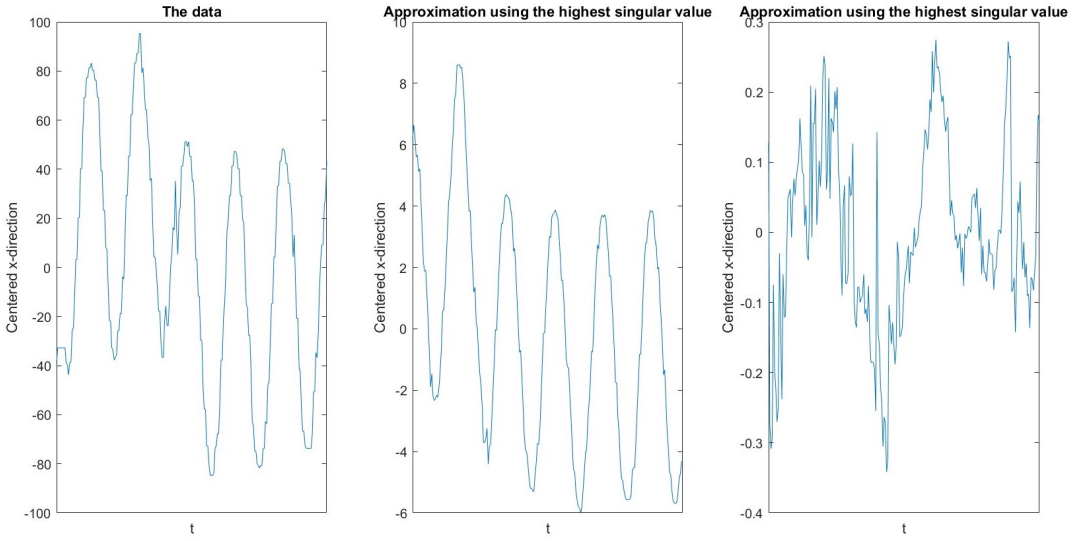

Figure 3: The effect caused by using the first, highest, singular value and last, lowest, singular value compared to the actual data. The highest singular value shows good form of the data, but lacks value and the last singular value gives nothing but giberish alone. It thus clearly only adds detail.

The approximation using the highest singular value has a clear form and bigger values. This is not as big as the actual data, that is why we will be using more than one singular value. The approximation using the lowest singular value, however, is very oscillatory, has a bad form and is small in value.

Therefore, it is clear that the higher singular value is more important, but clearly not good enough on it's own. In figure 2, however, we saw that it converged to the actual data. We choose to use the singular values which are cumulatively about 90% of our information. In order to figure out at which rank this is, we plot the energies in figure 4.

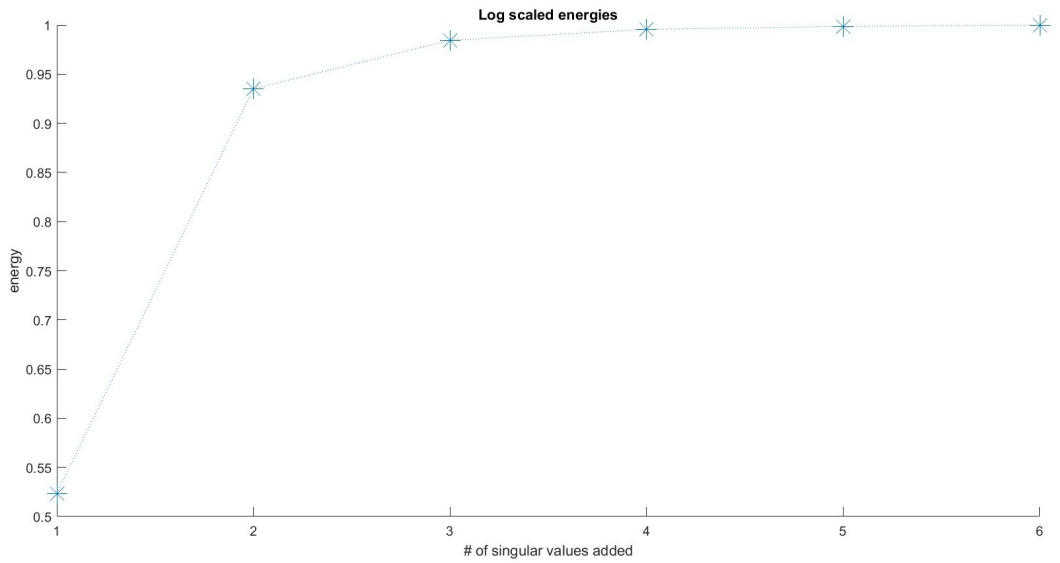

Figure 4: The energies, or the percentage of information kept in the approximation, is shown above, here we see that the first energies are important since the plot rises fast.

Here it is clear that we reach this threshold at rank 2, thus we will use the 2-rank approximation of the data to describe it better. Using the information on the x-y plane from videos one and three and the height from video two, we get the motion described in figure 5.

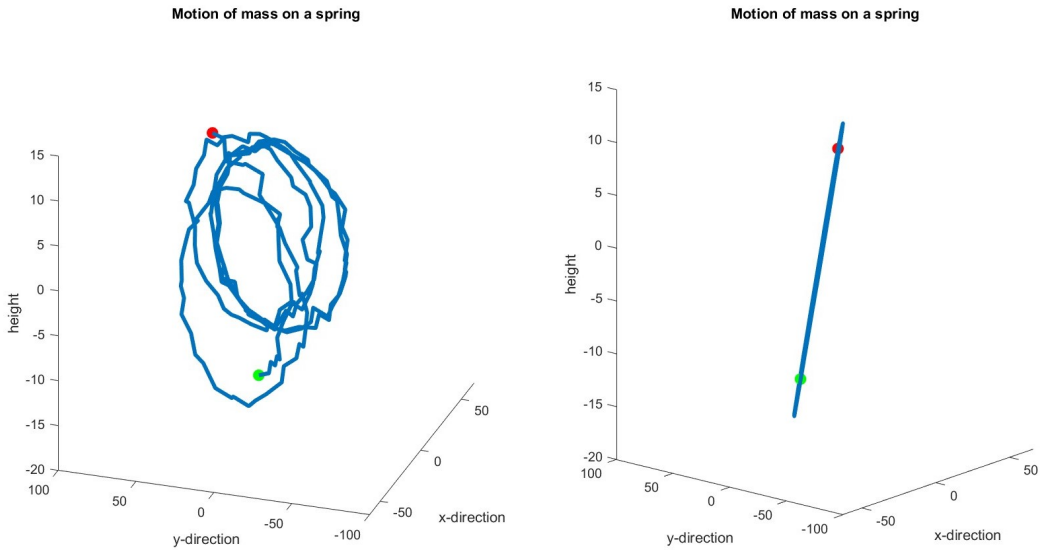

Figure 5: The motion of the mass on a spring according to a 2-rank approximation of the data plotted in three dimensions. The mass starts at the green dot and ends at the red dot. The results are promising, but lack the three dimensional characteristic of the real world. A good choice of n-approximation thus not only relies on the energy.

First note that it seems like changes in the x and y plane is huge, while we know it shouldn't be. I contribute this fact to how the data is processed before the SVD

process. Then more importantly: it looks like the mass on a spring moves linearly over the x-y plane while of course oscillating in height. The latter is what we expect, but the first is very strange one might think. Then again, one must remember that n-rank approximation project the data set into an n sub dimension of the rank. Here we choose to have the second rank and therefore we project our data into a two dimensional plane! As to why it choose to link the x-y plane, one must go back to the thought process using the covariance. When the mass is moving on the spring, it's mainly moving in the height direction and the x-y plane barely changes, thus they have a higher covariance than with the height. Using only this energy thresh hold might not be an ideal idea then. For this reason we plot all n-approximations in figure 6.

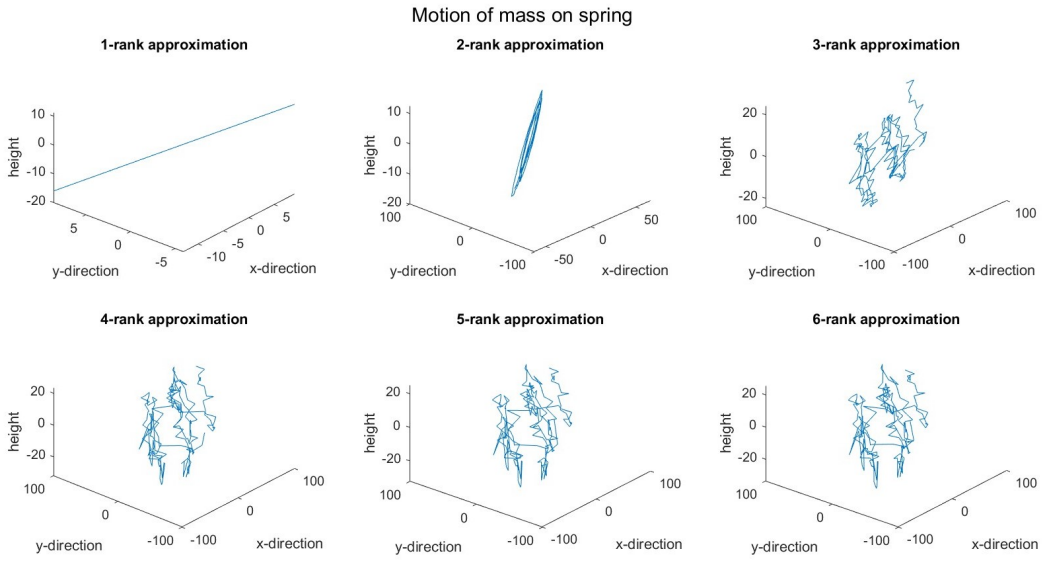

Figure 6: The motion of the mass on a spring using all n-rank approximations. The dimensionality of the first couple of approximations are clearly related to the rank, afterwards only details are added and big differences would be made if we were able to plot a higher dimension.

Here we see that the first three are indeed forms living on their respective dimensions, afterwards they all live on the three dimensional domain, since we're restricted to using three dimensions. Also note that the last plots are beginning to be very noisy, which is to be expected.

To conclude this results section, we plot the motions of the three other video throuples in figure 7 using the 3-rank approximations.

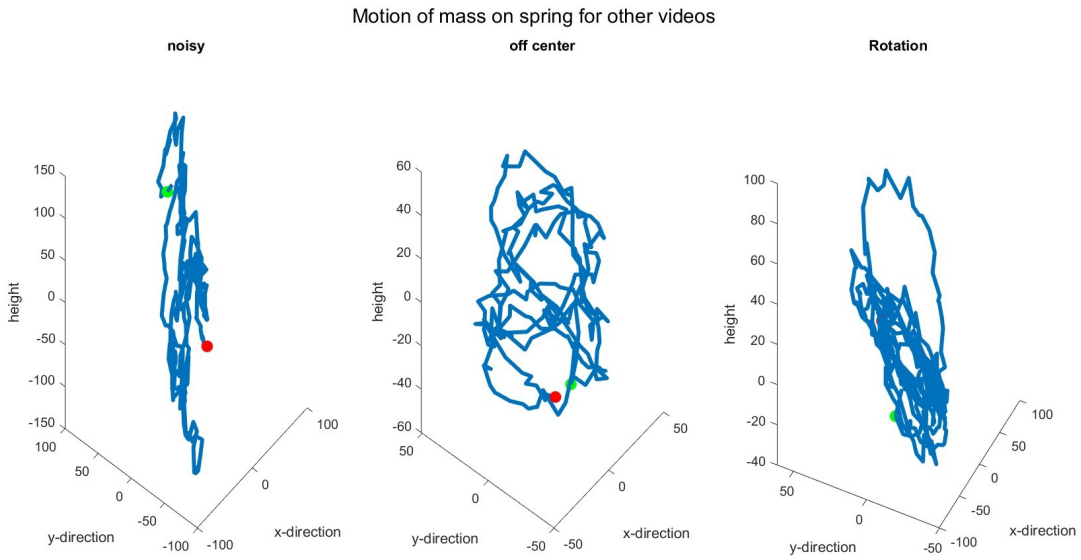

Figure 7: The motion of the mass on a spring using 3-rank approximations of the data. The mass starts at the green dot and ends at the red dot. Noise does have effect on the method, but it is not as bad. Off center has no effect since we center the data and rotation seems to enhance a weakness of the svd method: not minimizing outliers because of noise, in this case the movement.

Here we see that the noisy data is a bit compact, there are no clear oscillations in the x-y plane and it is clearly not as good as the second approximation. This indicates that the n-rank approximations are limited by what you give them. The original data also shows some form like this, likely due to how this initial guess is made. Therefore we see that the SVD method does not reduce the effects of noise on another process. The off centered video did not get influenced, this as we calculated the mean off of the data set and thus the displacement had no effect. It seems like the best motion in my opinion. The Rotation on the other hand also shows promise, but it shows that the SVD method might be susceptible to movement and rotation as there is a clear outlying movement. One must also remember that it relies on the covariances between the movements. Here all three movements have a lot of movement, thus for the 2-rank approximation it's a guessing game which direction will be filtered out. Lastly, The bucket with noise and rotation seem to also be compact on two dimensions, but we're unsure if this is because of the SVD method or the videos itself.

5 Improvements

In our naive opinion, improvements can be made in order to get better more realistic motions. As, this is not the main topic of the paper, we will make it short:

- In order to get a better output, the input might need to improve. One should process the videos before finding the initial motion of the mass. This can be done by, for example, cutting away excess space on the video by finding the initial domain over which the mass moves and then cutting away the rest.

Given that the mass is one color, it might be able to be found using Gabor transformations. Thereafter, one should scale the videos so that the bucket occupies approximately the same area over all videos. One can also rotate the video using a Givens rotation matrix in order to have the height consistently on the y-direction. The effects of noise can be reduced by smoothing the video with, for example, Guassian frequency filtering.

- Assume that for some reason we cannot put two cameras on perpendicular sides of the mass, one could, perhaps, interpolate two perpendicular projections of the bucket using the given angles of data. This will result a better plot of the motion.

- The rank of the approximation should not be solely based on the energies, It should also take into account the dimensionality over which our experiment is living. Say we are playing in dimensions d and the energies reach the wanted thresh hold at $n = \tilde{n}$, then we advised to choose the $\max(d, \tilde{n})$-rank approximation.

6 Conclusion

After receiving an initial guess at tracking a mass in a three dimensional domain in the form of six data sets, the motion can be better approximated using singular value decomposition. The data is over determined, we have six data sets, while we're only really interested in the three components of motion. The decomposition attempts to describe a matrix in rotations and scaling and in doing so it gives us a decomposition which tells us the singular values, or most important orthogonal vectors describing the matrix and even their respective importance! This was further proven by looking at its link to principal component analysis. Therefore telling us that this is a good method to efficiently compress data with redundant data without deleting data. Eventually, after finding the perfect $n-$rank approximations where enough information is being kept, while throwing away the less important singular values, through energies, we found a better way of describing the motion using videos. This method is still effected by the initial given data and the motion itself, more precisely the outliers. For this we're sure better methods exist in the world!

Acknowledgment

I would like to acknowledge the bottle of wine and homemade spaghetti that helped me get through the last stretches of this project.