Introduction to Data-Science and Data based Methods in Science and Engineering

Manuel R. Wendl
TUM - Munich School of Engineering

Abstract

This papers purpose is to convey basic ideas of Data-Science and data driven methods in connection with science and engineering. It is reduced to simple methods, based on mathematical models, suitable for beginners and intermediates. This first paper will only contain algorithms and methods with Linear Transformations, Change of Basis, as well as dimensionality Reduction. Therefore several coding projects based on Python and Matlab are included in this paper. The used code can be downloaded from .

Keywords: Data-Science; *Science and Engineering*

Contents

1 Introduction

Data-Science applied in Science and Engineering has different tasks to fulfill. The following selection will be discussed in the paper.

1. **Compression**:
   The occurrence of big datasets is very likely in science and engineering, such that it is a common task to compress the data, without losing crucial information.

2. **Denoising**:
   Some measurements or observation data might be corrupted, containing noise and measurement failures. It is crucial to be able to denoise the data, before further analysis.

3. **Prediction**:
   Science also studies the physical and natural Processes evolving over time. In order to predict certain phenomena in the future, data of previous events has to be analyzed and a policy has to be created.

4. **Classification**:
   Classification algorithms are used to distinguish between different objects, behaviors and processes.

2 Mathematical Basis

2.1 LSE - Linear System of Equations

A linear system contains n unknowns $x_1, x_2, x_3, \cdots, x_n$ which satisfy m linear equations of the form:

$$\begin{array}{lccccccccc} I: & a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & + & \cdots & + & a_{1n}x_n = b_1 \\ II: & a_{21}x_1 & + & a_{22}x_2 & + & a_{23}x_3 & + & \cdots & + & a_{2n}x_n = b_2 \\ \vdots & \vdots & & \vdots & & \vdots & & \ddots & \vdots & \vdots & \vdots \\ m: & a_{m1}x_1 & + & a_{m2}x_2 & + & a_{m3}x_3 & + & \cdots & + & a_{mn}x_n = b_m \end{array} \quad \forall a_{ij} \in \mathbb{R}$$

a_{ij} are the coefficients of the linear system.
In this paper such linear systems will be denoted as matrix products of form:

$$A \cdot x = b$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

2.2 Matrix Calculus

2.2.1 Addition of Matrices

Let $A = (a_{ij}) \in \mathbb{R}^{m \times n}, B = (b_{ij}) \in \mathbb{R}^{m \times n}$, then:

$$A + B := (a_{ij} + b_{ij}) \in \mathbb{R}^{m \times n}$$

$$A + B := \begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \cdots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \cdots & a_{2n} + b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \cdots & a_{mn} + b_{mn} \end{pmatrix}$$

2.2.2 Multiplication of a Matrix by a Scalar

Let $A = (a_{ij}) \in \mathbb{R}^{m \times n}$ and $\lambda \in \mathbb{R}$, then:

$$\lambda A := (\lambda a_{ij}) \in \mathbb{R}^{m \times n}$$

$$\lambda A := \begin{pmatrix} \lambda a_{11} & \lambda a_{12} & \cdots & \lambda a_{1n} \\ \lambda a_{21} & \lambda a_{22} & \cdots & \lambda a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda a_{m1} & \lambda a_{m2} & \cdots & \lambda a_{mn} \end{pmatrix}$$

2.2.3 Multiplication of two Matrices

Let $A \in \mathbb{R}^{l \times m}$ and $B \in \mathbb{R}^{m \times n}$.
Two matrices can only be multiplied, if the number of columns of the first matrix is equal to the

number of rows of the second matrix. The computed matrix $\in \mathbb{R}^{l \times n}$.

$$A \cdot B =$$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{l1} & a_{l2} & \cdots & a_{lm} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{pmatrix} = \begin{pmatrix} \sum_{j=1}^{m} a_{1j}b_{j1} & \sum_{j=1}^{m} a_{1j}b_{j2} & \cdots & \sum_{j=1}^{m} a_{1j}b_{jn} \\ \sum_{j=1}^{m} a_{2j}b_{j1} & \sum_{j=1}^{m} a_{2j}b_{j2} & \cdots & \sum_{j=1}^{m} a_{2j}b_{jn} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{j=1}^{n} a_{lj}b_{j1} & \sum_{j=1}^{n} a_{lj}b_{j2} & \cdots & \sum_{j=1}^{n} a_{lj}b_{jn} \end{pmatrix}$$

2.2.4 Transposed Matrices

Let $A \in \mathbb{R}^{m \times n}$ be an arbitrary matrix.
The transposed matrix A^T is the defined as follows:

$$A = (a_{ij}) \quad , \quad \forall i \in \{1, 2, 3, \cdots, m\} \, , \, \forall j \in \{1, 2, 3, \cdots, n\}$$
$$\Rightarrow A^T := (a_{ji}) \in \mathbb{R}^{n \times m}$$

2.2.5 Inverse Matrices

Let $A \in \mathbb{R}^{n \times n}$ be an arbitrary matrix.
The inverse matrix A^{-1} is the defined as follows:

$$A \cdot A^{-1} = A^{-1} \cdot A = I_n$$

I_n is a diagonal matrix of dimension n with only ones on its diagonal. It is called the Identity matrix.

2.3 Orthogonal Projections

Let W be an n-dimensional subspace of an Euclidean Vectorspace V, such that W can be seen as a hyperplane in V.
The aim is to find an element $P(x) \in W$ such that:

$$||x - P(x)|| = min\{||x - y||\} \forall y \in W$$

The orthogonal projection of x on W solves this problem.
Graphically this can be interpreted as in figure 2.1. The pink vector is the shortest possible $x - y$.

Practical use of this method:
This method is used to simplify high dimensional problems to less dimensional problems, such that the simplified problem is depicted as similar as possible to the original problem.
The projection $P(x)$ on W with **orthonormal basis** is determined by the sum of all projections of

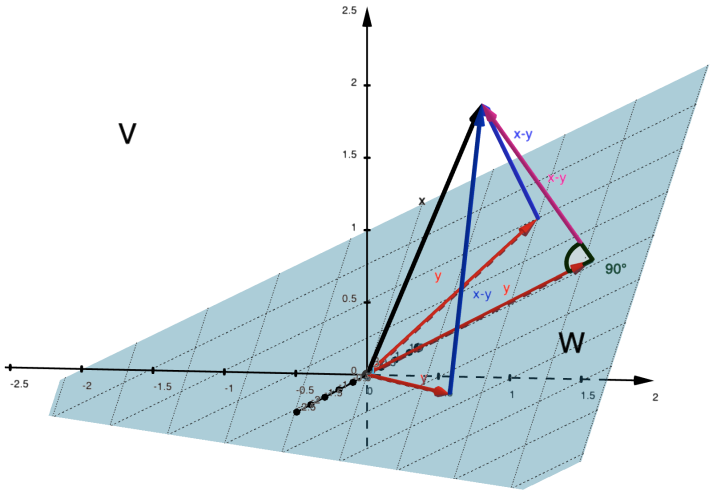

Figure 2.1: Hyperplane W in V. Several elements y $\in W$ with vectors $x - y$ with x $\in V$

x into the direction of the basis vectors.
This transformation P can be expressed as a matrix. This matrix is said to be orthogonal.

$$P : V \mapsto W$$

$$P(x) = \sum_{i=1}^{n} \langle w_i, x \rangle \cdot w_i = \sum_{i=1}^{n} w_i^T \cdot x \cdot w_i$$

$$\rightarrow P(x) = \sum_{i=1}^{n} w_i^T \cdot w_i \cdot x$$

2.4 Definition of Eigenvectors, Eigenvalues and Eigenspace

Let T be a linear map from $\mathbb{R}^n$ to $\mathbb{R}^n$.
T is represented by matrix $A \in \mathbb{R}^{n \times n}$ for some basis $B_{\mathbb{R}^n}$.
Eigenvectors are these vectors that will point into the **exact same direction** before and after the transformation. This means that only their length can be changed during the transformation.

$$Av = \lambda v \qquad \lambda \in \mathbb{C}$$

in this equation the **eigenvalue** is denoted by λ. The eigenvalue is the factor of enlargement or reduction of length of the eigenvector v through the transformation.
The **eigenspace** $Eig(\lambda) = \{v \in \mathbb{R}^n | Av = \lambda v\}$ the subvectorspace of all vectors that are eigenvectors with eigenvalue λ.

2.5 Singular Value Decomposition

With the singular value decomposition, it is possible to decompose any arbitrary matrix $A \in \mathbb{R}^{m \times n}$ into the product of three matrices U, Σ and V^T.

$$A = U\Sigma V^T \text{with } U \in \mathbb{R}^{m \times m} \in O(m), \Sigma \in \mathbb{R}^{m \times n}, V \in \mathbb{R}^{n \times n} \in O(n)$$

U and V are two orthogonal matrices and Σ is a diagonal matrix.

Attributes of Σ:

1. σ_i are called singular values of A.
2. $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k \geq 0$

Distinguish between two cases for Σ:

1. $A \in \mathbb{R}^{m \times n}$ with $m \leq n$

$$\begin{pmatrix} \sigma_1 & & 0 & 0 & \cdots & 0 \\ & \ddots & & \vdots & & \vdots \\ 0 & & \sigma_m & 0 & \cdots & 0 \end{pmatrix}$$

2. $A \in \mathbb{R}^{m \times n}$ with $m \geq n$

$$\begin{pmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_n \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{pmatrix}$$

[1]

As every matrix can be expressed in this **singular value decomposition (SVD)**, also every linear transformation can be seen as a combination of:

1. $V^T \in O(n)$ A orthogonal transformation/ rotation or reflection.
2. $\Sigma \in diag$ Stretch of eigenvectors with factor σ_i
3. $U \in O(n)$ A orthogonal transformation/ rotation or reflection.

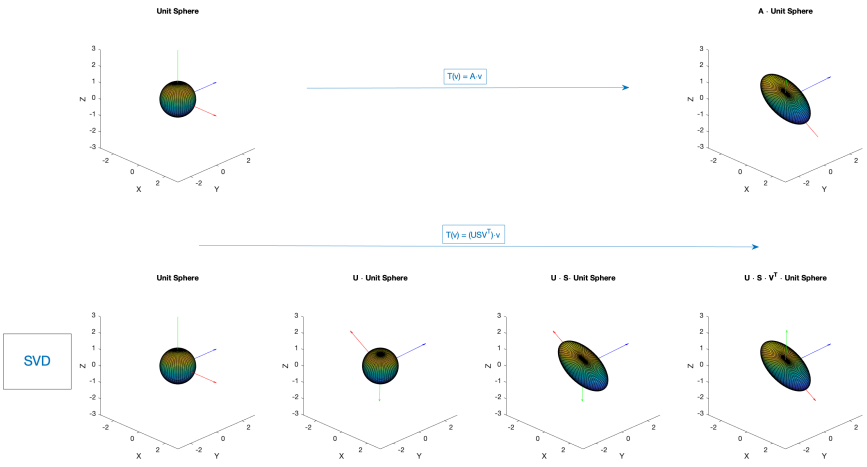

Figure 2.2: SVD visualisation

2.6 Fourier Series

2.6.1 Orthonormal system of sines and cosines

The inner product of two real functions is given by:

$$< f, g >= \int f(x) \cdot g(x)$$

For cosine and sine in the interval $[\pi, \pi]$ it is given by:

$$\forall \alpha \neq \beta : \int_{-\pi}^{\pi} \cos(\alpha x) \sin(\beta x) = \int_{-\pi}^{\pi} \sin(\alpha x) \sin(\beta x) = \int_{-\pi}^{\pi} \cos(\alpha x) \cos(\beta x) = 0$$

$$\int_{-\pi}^{\pi} \cos^2(\alpha x) = \int_{-\pi}^{\pi} \sin^2(\alpha x) = 1$$

This proofs, that sine and cosine functions of different frequencies span an orthonormal system. [2]

2.6.2 Real Fourier Series

Every periodic function can be expressed through a sum of infinitely many sines and cosines of different frequencies. This infinite sum is called the Fourier series.
Let $f(x)$ be $2 \cdot L$ periodic. Then the Fourier Series is given by: [2]

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos(k \frac{2\pi}{L} x) + b_k \sin(k \frac{2\pi}{L} x)$$

a_k, b_b are the Fourier coefficients, which are obtained by projecting the function f onto the sine or cosine function of the certain frequency.

$$a_k = \frac{1}{L} \int_{-\frac{L}{2}}^{\frac{L}{2}} f(x) \cos(k \frac{2\pi}{L} x) dx$$

$$b_k = \frac{1}{L} \int_{-\frac{L}{2}}^{\frac{L}{2}} f(x) \sin(k \frac{2\pi}{L} x) dx$$

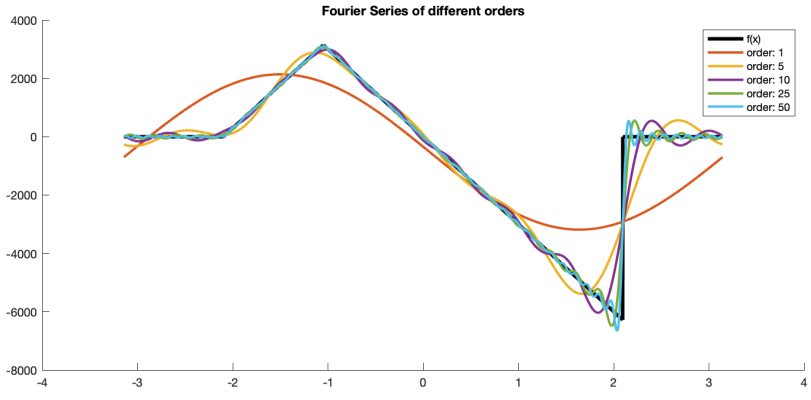

Figure 2.3: Fourier visualisation

2.6.3 Complex Fourier Series

The Fourier Series can also be developed in the set of the complex numbers. This helps to simplify the notation. [2]

$$e^{i\alpha} = \cos(\alpha) + i\sin(\alpha)$$

Then the Fourier-series of a L periodic function f is denoted as

$$f(x) = \sum_{k=-\infty}^{\infty} a_k e^{i+\frac{2\pi}{L}x}$$

Here only the Fourier coefficients a_k have to get calculated.

$$a_k = \frac{1}{L}\int_{-\frac{L}{2}}^{\frac{L}{2}} f(x)e^{i+\frac{2\pi}{L}x}dx$$

2.7 Discrete Fourier Transformation

2.7.1 Discrete Fourier Transformation

As in most cases no mathematical function is given as a Dataset, so there has to be dealt with discrete samples. The Discrete Fourier Transformation is a way to Fourier Transform a sampled data set. Therefore all data pairs are given by a x value and a y value. This y value can be interpreted as a function value $f(x)$ at a discrete x.
Let f be a periodic function in $[-\pi, \pi]$. And the equidistant, discrete n points x_k are defined as: [2]

$$x_l = l \cdot \frac{2\pi}{n} \quad \forall l = 0, 1, \cdots, n-1$$

The approximated complex Fourier coefficients are determined by applying the following formula.

$$c_k \approx \hat{c_k} = \frac{1}{n}\sum_{l=0}^{n-1} f(\frac{2\pi}{n}) \cdot e^{\frac{-2\pi i}{n}kl} \quad \forall k = 0, \cdots, N-1$$

Simplified with $v_l = f(l\frac{2\pi}{n})$ and $\zeta = e^{\frac{-2\pi i}{n}}$ the equation can be written as:

$$c_k \approx \hat{c_k} = \frac{1}{n}\sum_{l=0}^{n-1} v_l \cdot \zeta^{kl} \quad \forall k = 0, \cdots, N-1$$

The equations of the coefficients then have the following form.

$$\begin{aligned}
\hat{c_0} &= \frac{1}{N}(v_0\zeta^0 + v_1\zeta^0 + v_2\zeta^0 + \cdots v_{n-1}\zeta^0) \\
\hat{c_1} &= \frac{1}{N}(v_0\zeta^0 + v_1\zeta^1 + v_2\zeta^2 + \cdots v_{n-1}\zeta^{n-1}) \\
\hat{c_2} &= \frac{1}{N}(v_0\zeta^0 + v_1\zeta^2 + v_2\zeta^4 + \cdots v_{n-1}\zeta^{2(n-1)}) \\
&\vdots \\
\hat{c_{n-1}} &= \frac{1}{N}(v_0\zeta^0 + v_1\zeta^{n-1} + v_2\zeta^{2(n-1)} + \cdots v_{n-1}\zeta^{(n-1)(n-1)})
\end{aligned}$$

These equations can be written as a matrix product.

$$\hat{c} = \frac{1}{n} F_n \cdot v$$

$$\begin{pmatrix} \hat{c_0} \\ \hat{c_1} \\ \hat{c_2} \\ \vdots \\ \hat{c_{n-1}} \end{pmatrix} = \frac{1}{n} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \zeta^1 & \zeta^2 & \cdots & \zeta^{n-1} \\ 1 & \zeta^2 & \zeta^4 & \cdots & \zeta^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \zeta^{n-1} & \zeta^{2(n-1} & \cdots & \zeta^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ \vdots \\ v_{n-1} \end{pmatrix}$$

F_n is called the n^{th} Fourier matrix.

2.7.2 Inverse Discrete Fourier Transformation

To regain the vector v from the calculated coefficients, this transformation has to be inverted. [2]

$$\hat{c} = \frac{1}{n} F_n \cdot v \Rightarrow (\frac{1}{n} F_n)^{-1} \hat{c} = (\frac{1}{n} F_n)^{-1} \frac{1}{n} F_n \cdot v$$
$$\Rightarrow v = n \cdot F_n^{-1} \hat{c}$$

As following propostion is true: $F_n \cdot \bar{F_n} = n \cdot I_n$

$$F_n^{-1} F_n \bar{F_n} = n F_n^{-1} \Rightarrow F_n^{-1} = \frac{1}{n} \bar{F_n}$$

Then the Inverse DFT is given by:

$$v = \overline{\frac{1}{n} F_n \hat{c}}$$

3 Compression

3.1 Matrix Approximation with SVD

3.1.1 Theory

Every SVD can be written as a sum:

$$U\Sigma V^T = \frac{1}{\sqrt{10}}\begin{pmatrix}1 & -3\\ 3 & 1\end{pmatrix}\begin{pmatrix}3\sqrt{5} & 0\\ 0 & \sqrt{5}\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ -1 & 1\end{pmatrix}$$
$$= \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T = \frac{3}{2}\begin{pmatrix}1 & 3\\ 1 & 3\end{pmatrix} + \frac{1}{2}\begin{pmatrix}3 & -1\\ -3 & 1\end{pmatrix} = \begin{pmatrix}3 & 0\\ 4 & 5\end{pmatrix}$$

As the singular values decrease, the smallest summands can be eliminated without changing the matrix values dramatically.

$$A = \sum_{i=1}^{j} \sigma_i u_i v_i^T \overset{\text{rank reduced}}{\Rightarrow} \tilde{A} = \sum_{i=1}^{k} \sigma_i u_i v_i^T \text{ with } k < j$$

$$A = U\begin{pmatrix}\sigma_1 & & 0 & 0 & \cdots & 0\\ & \ddots & & \vdots & & \vdots\\ 0 & & \sigma_j & 0 & \cdots & 0\end{pmatrix}V^T$$

$$\Rightarrow U\begin{pmatrix}\sigma_1 & & 0 & 0 & \cdots & 0 & \cdots & 0\\ & \ddots & & \vdots & & \vdots & & \vdots\\ 0 & & \sigma_k & 0 & \cdots & 0 & & \vdots\\ 0 & \cdots & \cdots & 0 & & \vdots & & \vdots\\ \vdots & & & & \ddots & \vdots & & \vdots\\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & \cdots & 0\end{pmatrix}V^T$$

Here the red σ_i were reduced. In fact now the columns $\{k+1, \cdots, j\}$ of U and the rows $\{k+1, \cdots, j\}$ of V^T will always be multiplied with zero, such that they no longer need to be stocked. Remark: In most cases the economy SVD is used as it is less expensive to calculate. The economy SVD has got a quadratic Σ. Consequently also the size of either U or V has to be adapted. [3]

3.1.2 Low Rank Approximation on Black and White Picture

The picture used is a random black and white picture with size 408×612 in png format.
The following steps were made to process the compression:

1. The picture is converted into a three dimensional matrix with the RGB values stocked for every pixel.
2. The RGB format is converted into a double matrix with the gray values stored inside.
3. As we now have the matrix with the gray values, it is possible to do the singular value decomposition.

4. Modification of the diagonal matrix $S(\Sigma)$. S get's multiplied with another diagonal matrix D, which has got all ones on the diagonal elements, except of the elements which multiply to the small σ_is, which shall be set to zero.

$$D = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{pmatrix}$$

5. The matrix with the gray values get's calculated again through $A = USV^T$

6. The picture will get displayed.

In figure 3.4 the development of quality with respect to the rank can be seen. There can be said that the quality of the picture does barely decrease from $rank(\Sigma) = 408$ to $rank(\Sigma) = 120$.

This can also be seen in the rapid decrease of singular values. The figure 3.1 shows the first 200 singular values in a logarithmic scaling.

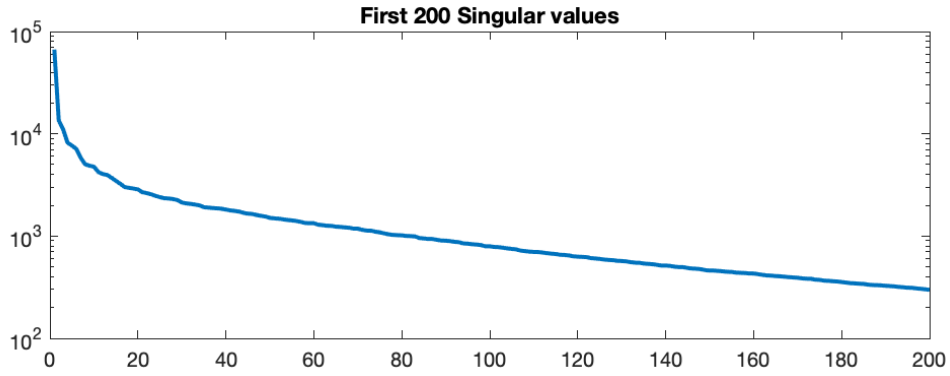

Figure 3.1: Development of picture rank reduced.

If the original and reduced amount of data is compared we get the following data:

$$\text{Data } rank(\Sigma) = 408 : 408 \cdot 612 = 247860$$
$$\text{Data } rank(\Sigma) = 120 : 120 \cdot 408 + 120 \cdot 612 + 120 = 122520$$
$$\Rightarrow \text{Ratio of dispensable data} : \frac{247860 - 122520}{247860} = \frac{125340}{247860} = 50.6\%$$

Consequently only half of the data needs to be kept, to get a nearly identical image.

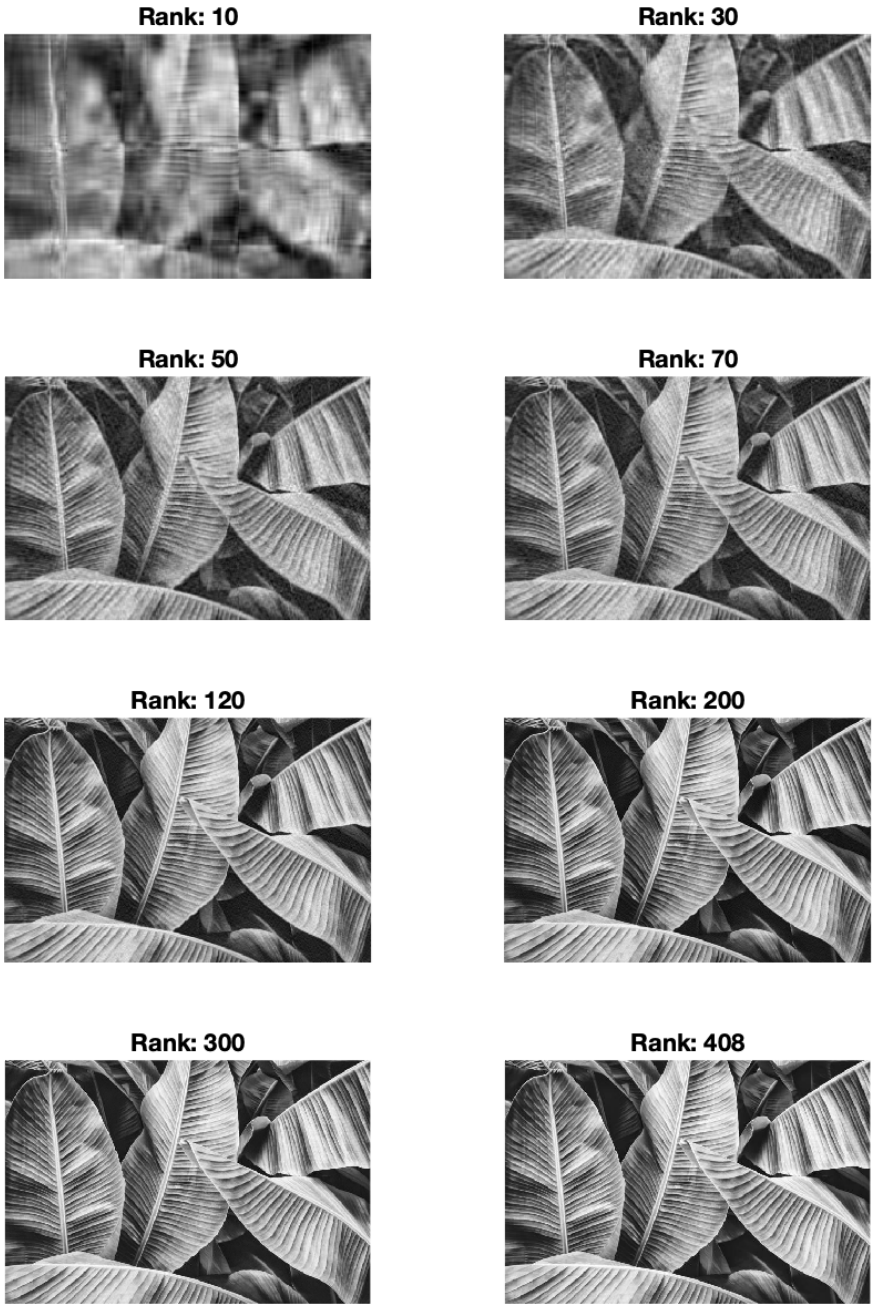

Figure 3.2: Development of picture rank reduced.

3.1.3 Low Rank Approximation on Colored Picture

The picture used is a random colored picture with size 563×750 in png format.
Most of the following steps have already been discussed in 3.1.2.
The following steps were made to process the compression:

1. The picture is converted into a three dimensional matrix with the RGB values.
2. The three dimensional array is split into three two dimensional matrices with double entries.
3. The singular value decomposition of the three matrices are calculated individually.
4. Modification of the diagonal matrices $S(\Sigma)$. This has been discussed in 3.1.2.
5. The three matrices get approximated with the rank reduced $S(\Sigma)$ and merged again to a three dimensional matrix.
6. The figures are displayed.

In figure 3.4 the development of quality with respect to the rank can be seen.

As already in the black and white example $rank(\Sigma) = 120$ also seems to be a good estimation here, as with lower ranks the sharpness decreases rapidly. Although the singular of each color decrease faster. However all the information is attained by the combination of all three.

This can again be seen in the rapid decrease of singular values. The figure 3.3 shows the first 200 singular values in a logarithmic scaling.

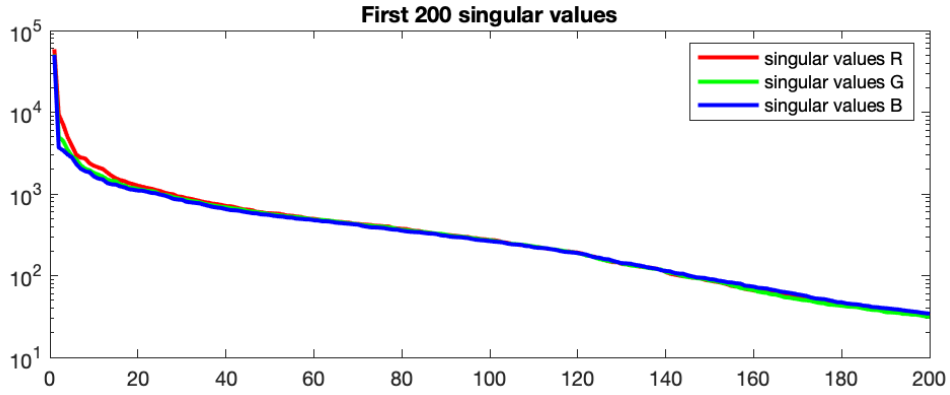

Figure 3.3: Development of picture rank reduced.

If the original and reduced amount of data is compared we get the following data:

$$\text{Data } rank(\Sigma) = 536 : (536 \cdot 750) \cdot 3 = 1206000$$
$$\text{Data } rank(\Sigma) = 120 : (120 \cdot 536 + 120 \cdot 750 + 120) \cdot 3 = 463320$$
$$\Rightarrow \text{Ratio of dispensable data} : \frac{1206000 - 463320}{1206000} = \frac{742680}{1206000} = 61.6\%$$

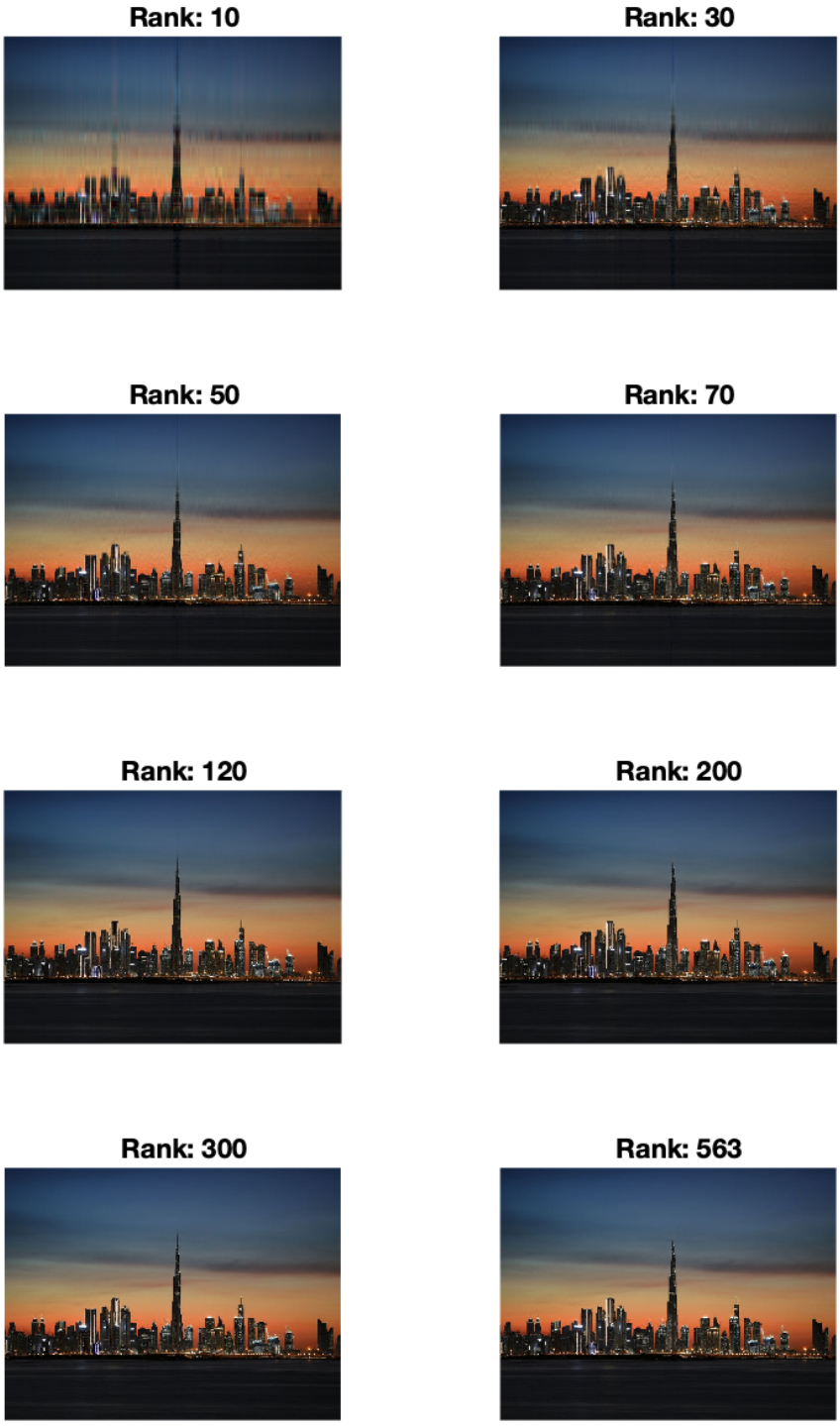

Figure 3.4: Development of picture rank reduced.

3.1.4 Effect of Alignment on Image Compression with SVD

Applying the same compression intensity (same rank reduction) on different, equally sized images results in varying image quality. Such that there has to be a factor of 'how well a picture can be approximated with the SVD'.

Therefore the SVD has to be approached mathematically again.
Let matrix A be decomposed into U, Σ, V with the economy SVD.

$$A = \begin{pmatrix} | & & | \\ u_1 & \cdots & u_n \\ | & & | \end{pmatrix} \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{pmatrix} \begin{pmatrix} | & & | \\ v_1 & \cdots & v_m \\ | & & | \end{pmatrix}^T$$

A matrix is good approximatable with less eigenvalues, if it can be written as a linear combination of less columns of U and V. This means that a matrix with 'alignment' (patterns) in rows and columns is approximatable through less columns and rows and therefore less singular values.

In the following code example (figure 3.5) a strongly aligned matrix is getting less aligned from left to right. Of all those matrices the economy SVD is computed and the first 20 eigenvalues are plotted. It is obvious from this plot, that the number of singular values and their value increases

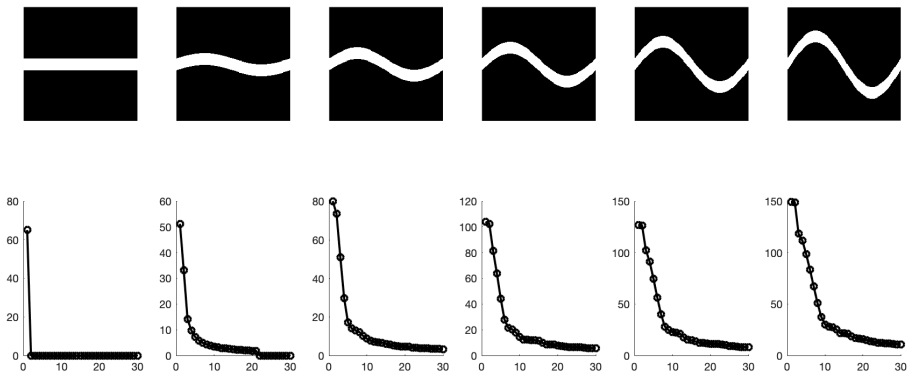

Figure 3.5: Development of picture rank reduced.

with respect to the decreasing alignment in the matrix.

As this has been a simplified example, the theory will be demonstrated on two images. The images and their first 200 singular values plotted on a logarithmic scale can be found in figure 3.6. The left image is strongly aligned, as the city blocks are ordered quadratically. In contrast, the right picture has less alignment. This should lead to a faster decrease of the singular values of the left picture, than the right picture. Exactly this expectation is satisfied, as the graphs below show.

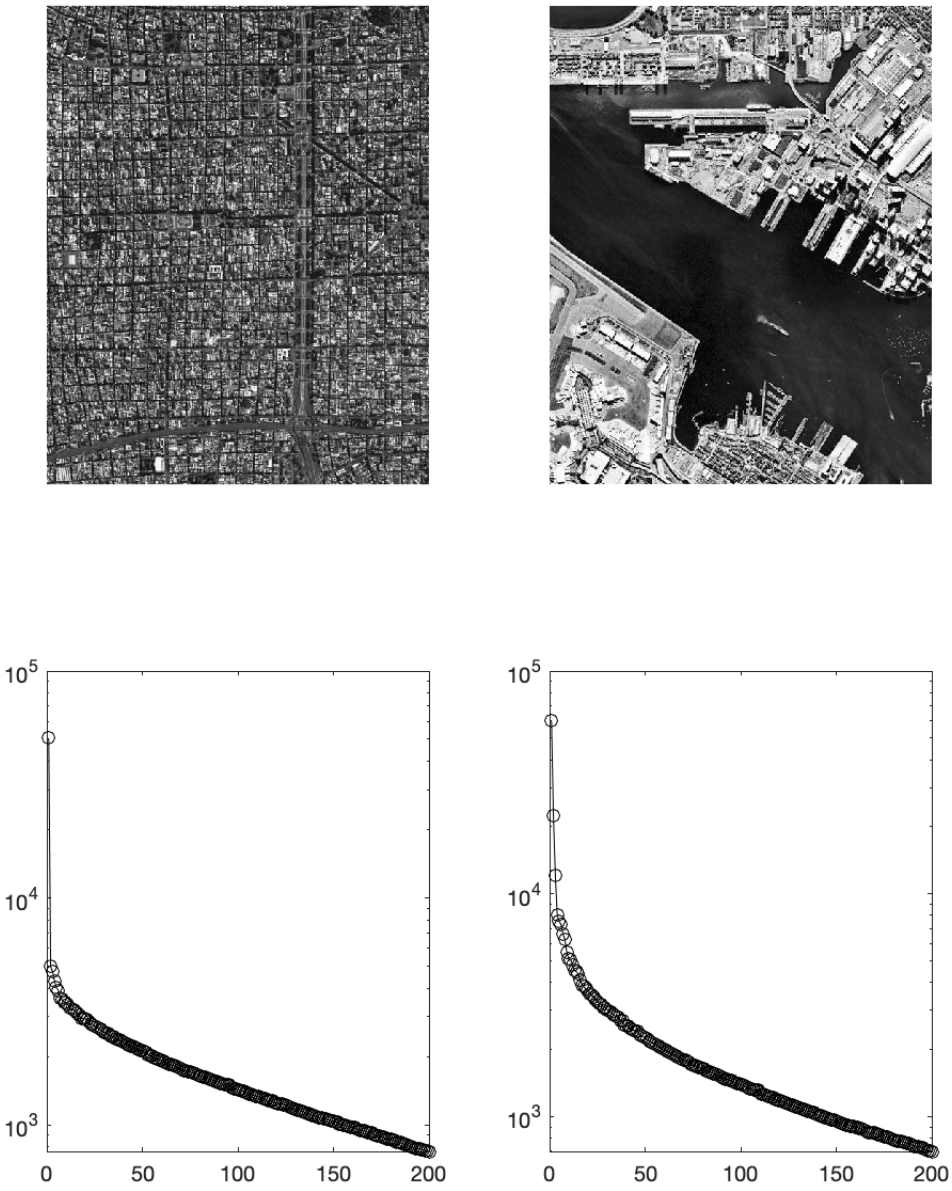

Figure 3.6: Development of picture rank reduced.

3.1.5 Conclusion

A matrix can be approximated with a diagonal matrix S of lower rank, such that there isn't a drastic loss of information. Because of singular values of higher rank being zero, columns of U and V are getting multiplied with zero, such that they are redundant to store. This leads to a reduction of data, however the outcome of the approximation (even with same rank) differs with respect to the alignment in rows and columns of the matrix.

3.2 Compression with FFT - Fast Fourier Transformation

3.2.1 Theory

The DFT was already introduced in 2.7. However it is a very computational expensive algorithm, as the matrix product requires n^2 multiplications. For faster transformation a less expensive algorithm is needed.

This leads us to the Fast Fourier Transformation (FFT). Its runtime is of order $O(nlog(n))$.

Let the data-vector in this example be of length $2^5 = 32$. Then the Fourier matrix F_{32} is of size 32×32. The equation can be written as: [3]

$$\hat{c} = F_{32}v$$

However if the entries of v get reordered to even and odd indices, the matrix multiplication can be simplified.

$$\hat{c} = \begin{pmatrix} I_{16} & -D_{16} \\ I_{16} & -D_{16} \end{pmatrix} \begin{pmatrix} F_{16} & 0 \\ 0 & F_{16} \end{pmatrix} \begin{pmatrix} v_{even} \\ v_{odd} \end{pmatrix}$$

For ordering the elements of v a permutation matrix is used.

$$\hat{c} = \begin{pmatrix} I_{16} & -D_{16} \\ I_{16} & -D_{16} \end{pmatrix} \begin{pmatrix} F_{16} & 0 \\ 0 & F_{16} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ & & & \vdots & & \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 1 & \cdots & 0 \\ & & & \vdots & & \end{pmatrix} v$$

The matrix D_16 is a diagonal matrix of form:

$$\begin{pmatrix} 1 & & & & \\ & \zeta & & & \\ & & \zeta^2 & & \\ & & & \ddots & \\ & & & & \zeta^{15} \end{pmatrix}$$

This can now be applied recursively and the F_{16} matrices replaced by two F_8 matrices.

$$\hat{c} = \begin{pmatrix} I_8 & D_8 & & \\ I_8 & -D_8 & & \\ & & I_8 & D_8 \\ & & I_8 & -D_8 \end{pmatrix} \begin{pmatrix} F_8 & & & \\ & F_8 & & \\ & & F_8 & \\ & & & F_8 \end{pmatrix} \begin{pmatrix} v_{even-even} \\ v_{even-odd} \\ v_{odd-even} \\ v_{odd-odd} \end{pmatrix}$$

In case the vector is of length $2^n \forall n \in \mathbb{N}$ this can be done until the Fourier matrix is reduced to size 2×2. In case v isn't a power of two the vector gets padded with zeros to reach the next power of two. [3]

For two dimensional data, there is used the $FFT2$. This algorithm first determines the Fourier coefficients for each row, before applying the Fast Fourier Transformation on the columns of the matrix gained from the first step.

3.2.2 Image Compression FFT2

The in the example used picture is a black and white picture of size 3668×2945. This means that it is has a high resolution. The following steps were made to process the compression and edge detection:

1. The image is imported in 'jpg' format. The matrix entries are converted to doubles.
2. For demonstration the image is visualised as a surface plot.
3. The Discrete Fourier Transformation matrix is plotted. This however is only for demonstration purpose as well. As all compression and analysis will be done with the much faster FFT2.
4. Next the FFT2 is applied on the matrix.
5. For the compression the most significant frequencies have to be found. Therefore the matrix get's reshaped into a vector and sorted. The most redundant (smallest) coefficients will get eliminate by setting them to zero.
6. The compressed images are reconstructed from the Fourier domain by applying the Inverse Fourier Transform with the specific ratio of coefficients that we want to keep.
7. For the edge detection, the most significant frequencies are removed (low pass filter). The reduced FFT2 matrices are shown additionally to the filtered images.

The FFT2 can be simply seen as an approximation of a surface with sine and cosine waves in row and column dimension. Therefore the target image can be interpreted as a surface, which is shown in figure 3.7. The colour scaling shows the value of the single pixels.

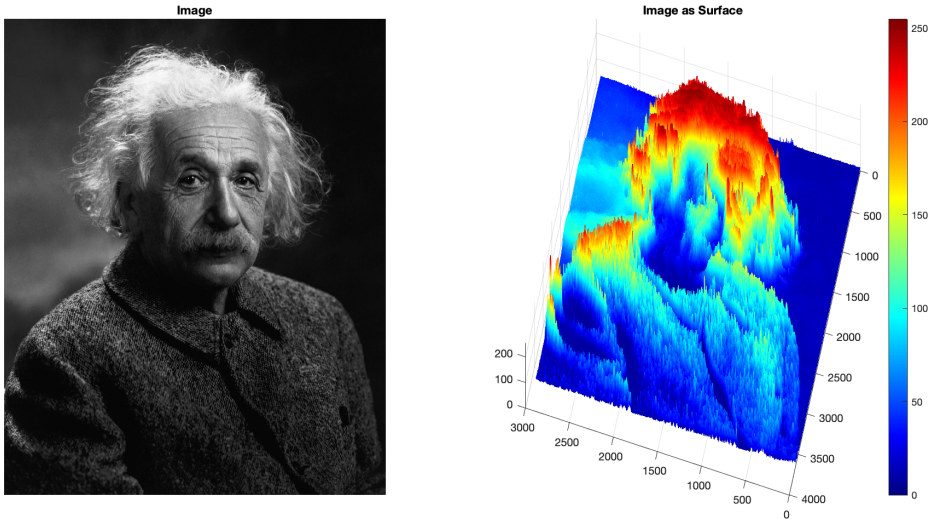

Figure 3.7: Surface plot of image

In figure 3.8 the DFT matrix of size 3668×3668 is shown. There can actually be seen the periodically patterns in this matrix and why it can be approximated with these matrices of smaller sizes.

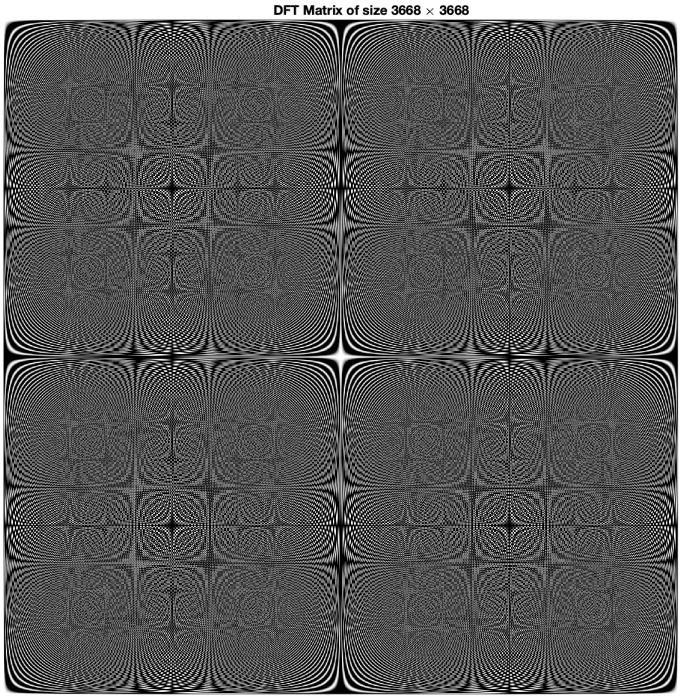

Figure 3.8: DFT matrix size 3668×3668

In figure 3.9 the compressed images are plotted. Additionally to every subplot the percentage of Fourier coefficients, that were kept for compression, is given.

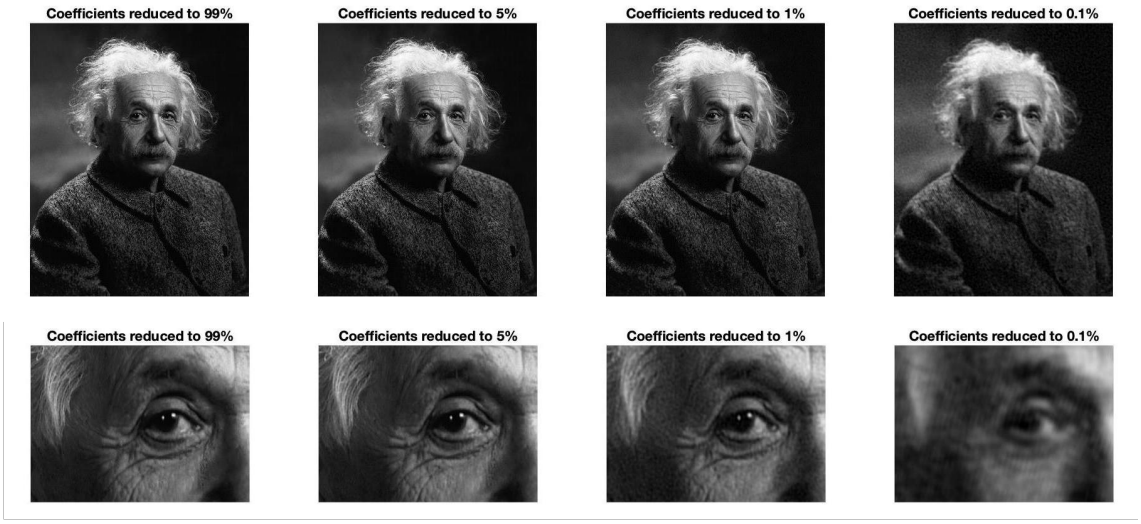

Figure 3.9: FFT2 compressed pictures to certain percentage of coefficients

With the FFT2 there are also applications in Image Analytics and filtering. In figure 3.10 there is applied a so called low pass filter, in which the frequencies of the dominant Fourier coefficients are eliminated. This leads to an effect considered as edge detection. The elimination causes a loss of information about big surfaces of the picture and keeps all detailed information. Consequently only the minor changes in the picture are displayed, these are the edges. In figure 3.10 there are plotted the filtered images with respect to the number of eliminated Fourier coefficients, as well as the FFT2 coefficient matrix on a log scale. The FFT2 matrix first has to be fft-shifted, such that the lowest frequencies are displayed in the middle.

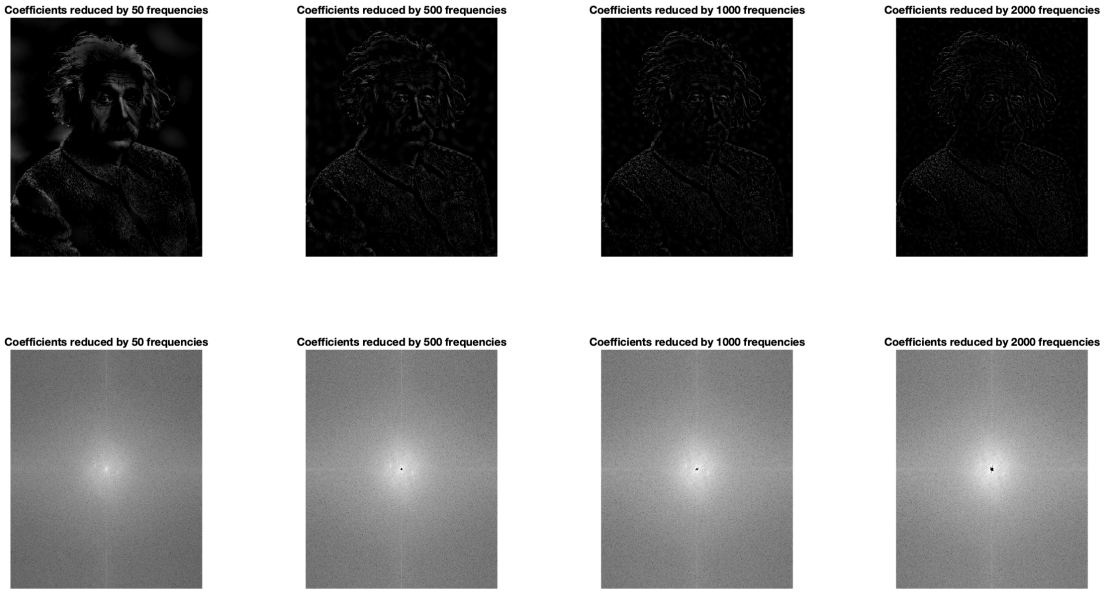

Figure 3.10: FFT2 Matrix logarithmic plot with filter

A general observation is, that the most information of the image is stored in low frequencies, located in the middle, whereas the high frequent part are nearly zero. Because of the logarithmic scale of the FFT2 matrix plot, the gray regions may actually be zero. Additional there can be seen an increasing black point (from left to right). These are the frequencies eliminated by the filter and set to zero.

3.2.3 Audio Compression FFT and Powerspectrum of Frequencies

In the following abstract the aim is to compress an audio File, by transforming the data of the time series into the Fourier domain, reducing the coefficients and reconstructing it on a reduced data basis.

In the following Code example the first line of the famous piano solo 'Für Elise' by Ludwig van Beethoven is used for compression.
Here are listed the required steps in order to perform the compression.

1. The audio and the sampling rate (measurements per second) is imported and a sequence is extracted.

2. The sample get's transformed into the Fourier Domain. (FFT applied)

3. All the coefficients below a threshold, set by the intensity of compression, are eliminated.

4. Next the audio sample is reconstructed with the reduced data. The compressed audio samples are plotted, as well as played. Additionally the Fourier coefficients are displayed.

In the first row of figure 3.11 the compressed audio samples are plotted. Below the audio samples, the norms of the corresponding Fourier coefficients are plotted on a log scale. There can be observed, that there is a strong correlation between the number of frequencies with large coefficients and the audio quality.

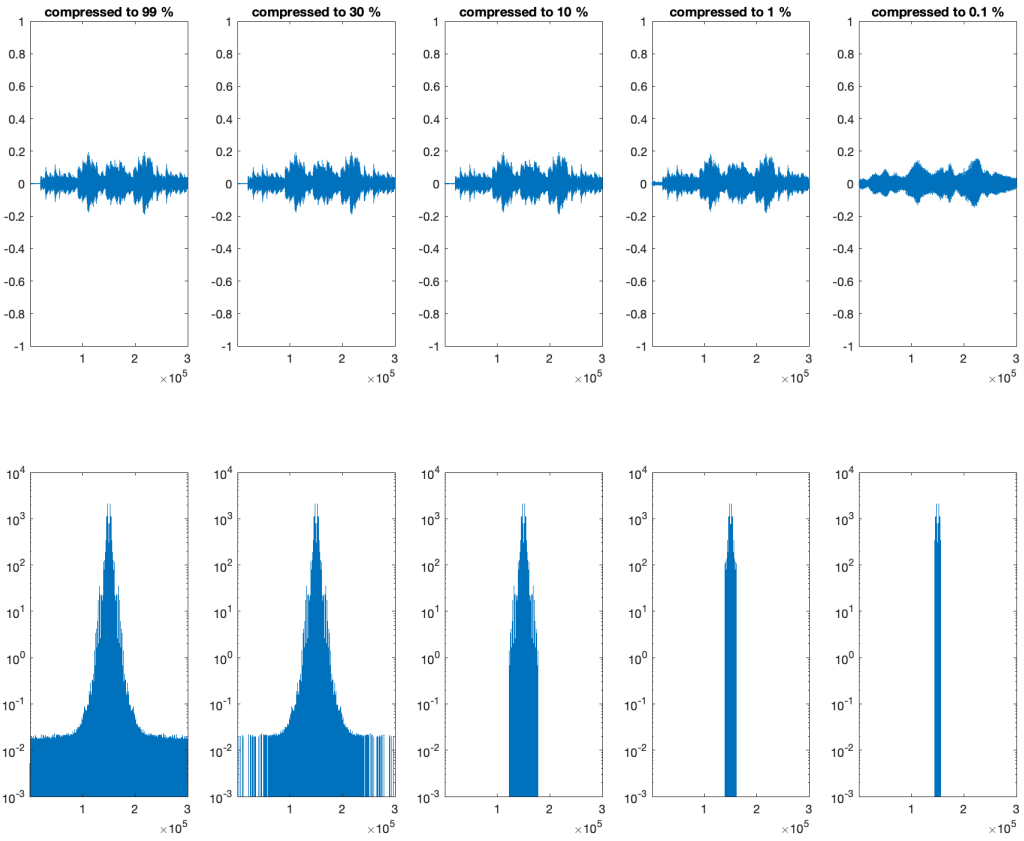

Figure 3.11: FFT compressed audio samples

Additional to the compression it is very interesting to investigate the powerspectrum of the frequencies in that audio file. As every Fourier coefficient is connected to a special frequency of sine and cosine waves, it is possible to say which frequencies dominate. If the piano, that was recorded in this sample, is tuned correctly, the dominant frequencies should refer to dominant tones in the piano solo. Therefore the one sided powerspectrum is plotted in figure 3.12. The power of the frequencies are computed by taking the norm of the complex Fourier coefficients . For better visualisation they are scaled in Hz.

In the powerspectrum (upper subplot) there were added the note lines (vertical black lines) and the most significant frequencies are marked in colours. It is very remarkable, that these frequencies directly correlate with certain tones of the piano.
In the first line of notes (bottom subplot) the correlating tones are coloured identically, such that it is possible to conclude, that the most frequently played notes are the most dominant frequencies in the powerspectrum.

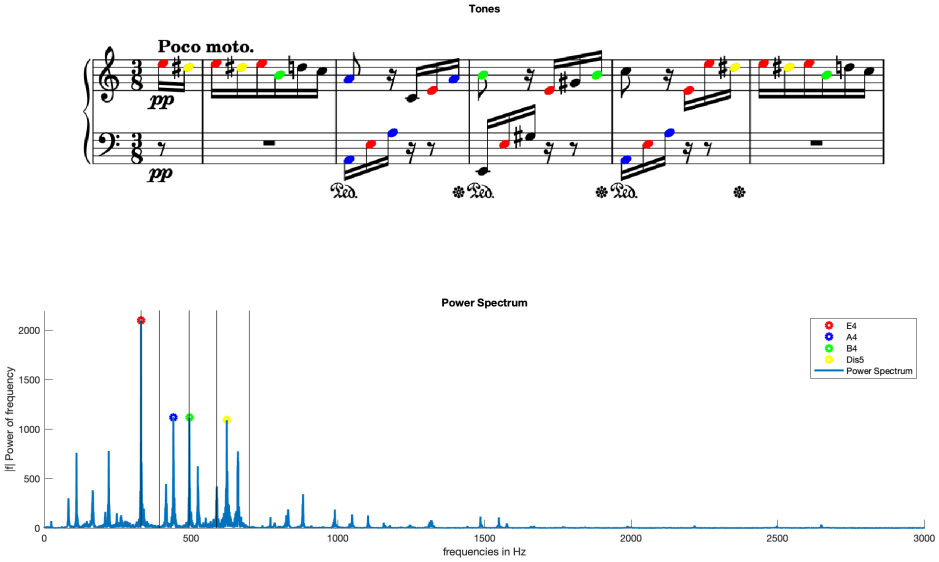

Figure 3.12: FFT compressed audio samples

3.2.4 Conclusion

It is a very effective way to compress Data with the FFT, as the runtime is reduced with the Fast Fourier Algorithm to $n\log(n)$. In comparison to the SVD, the calculations are less expensive.
From both compressions (image and audio) it is possible to observe, that the most information is conveyed by the low frequencies. Whereas details are contained in higher frequencies.

4 Denoising Data

4.1 Denoiseing Data with Truncated SVD

4.1.1 Theory

The idea of this section is to analyse the singular value decomposition in order to find the singular values corresponding to an evenly distributed noise in the data. After identifying these modes it will be the task to reconstruct the data with the correct truncation of modes, in order to differentiate between important information and noise.
The basic algorithm (the SVD) behind this section is already explained in detail in 2.5 as well as the low rank approximation in 3.1. Basically there will be performed a low rank approximation, however on specifically chosen singular values.

4.1.2 Image Denoising with SVD Truncation

In this section a black and white picture containing normally distributed Gaussian white noise is denoised by applying the SVD truncation.
Before focusing on the chosen example, there has to be said, that this method works best for images with strong alignment of data in rows and columns. The effect of alignment on performance of the SVD is described in 3.1.4.

Low Rank System
Because of this reason an image with these requirements is chosen. The original image is without the white noise, as it is used for validation of the optimal truncation. Consequently there has to be added white noise in the first step. In figure 4.1 there is shown the original picture, the added Gaussian white noise and the noisy image. Additionally there are plotted the singular values of each image on a logarithmic scale.

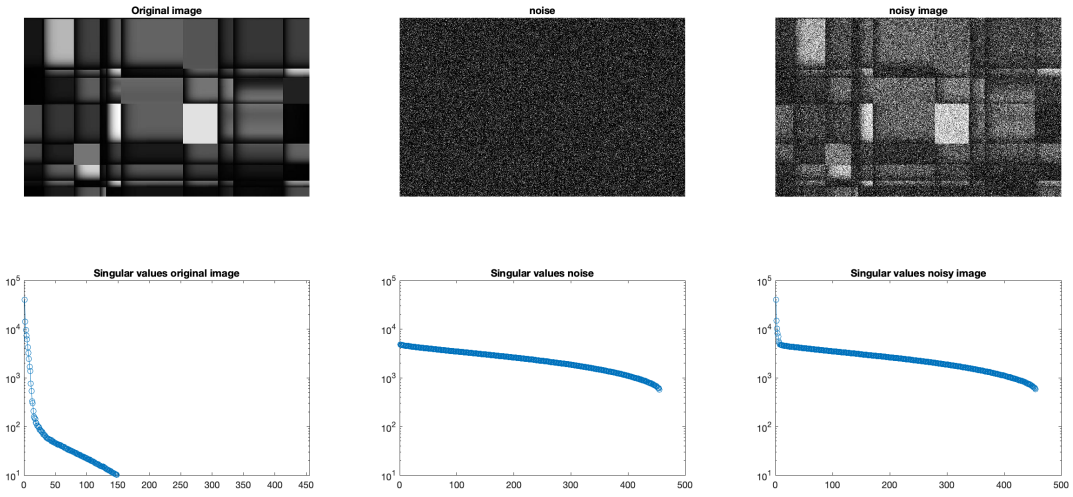

Figure 4.1: Original image, noise and noisy image with singular values

From figure 4.1 can we conclude, that the singular values of the original plot decrease rapidly, such that the requirement of strong alignment is fullfilled.
Further there is a characteristic singular value distribution for white noise. This white noise is not aligned at all, as ist was generated randomly, such that there is a slow decrease in singular values.
Last there are plotted the singular values of the noisy picture. The graph of the singular values seems to have a strong correlation with the original image for the first few values, whereas it is clearly dominated by the noise distribution in the further course.

From this observation, there was developed a mathematical approach by myself, in order to predict a threshold τ for truncation.
Therefore the idea is, to approximate the singular values s of noise in the logarithmic scale with a linear function.

$$\log(s) = \alpha \cdot x + t$$
$$s = e^{\alpha \cdot x + t}$$

The gradient of the slope is determined by the following quotient, in which θ is the index of the median Θ of all m singular values.

$$\alpha = \frac{\log(f(\theta - \frac{\theta}{2}) - f(\theta + \frac{m-\theta}{2}))}{\frac{\theta}{2} + \frac{m-\theta}{2}}$$

Then the threshold τ can be approximated as the highest singular value of the noise from the linear approximation. The largest singular value is always found with index $1(0)$.

$$\tau = e^{\alpha \cdot \theta + \Theta}$$

In this specific case the value (log scale) and index of the median are $\Theta = 7.7835$, $\theta = 228$. Such that the slope can be determined to be:

$$\alpha = \frac{\log(f(228 - 114) - f(228 + 114))}{228} = \frac{8.1170 - 7.3493}{228} = 0.0034$$

All values in this equation are the logarithmic values, such that it actually is the visual slope in the plot.

From this the threshold τ (not in logarithmic scale) is determined to be:

$$\tau = e^{0.0034 \cdot 228 + 7.7835} = 5.2 \cdot 10^3$$

With the determined truncation threshold, all singular values above are kept. In this specific case these are the first six singular values.
In figure 4.2 this linear approximation is visualised.
In the figure 4.3 the original image, the noisy image and the truncated image is shown. The truncation is performed with the mathematically determined threshold τ. Additionally there is plotted the error weighted with respect to the size of the image, for different truncations. On the x axis the number of singular values is given.
It is pretty remarkable, that it was possible to find the optimal truncation with the linear approximation of the noise. The optimal truncation with the least error is marked in red (bottom left subplot).

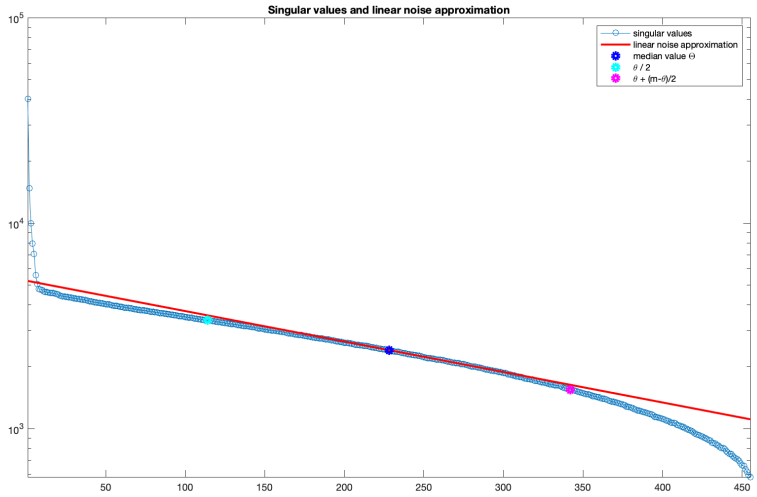

Figure 4.2: Linear noise approximation

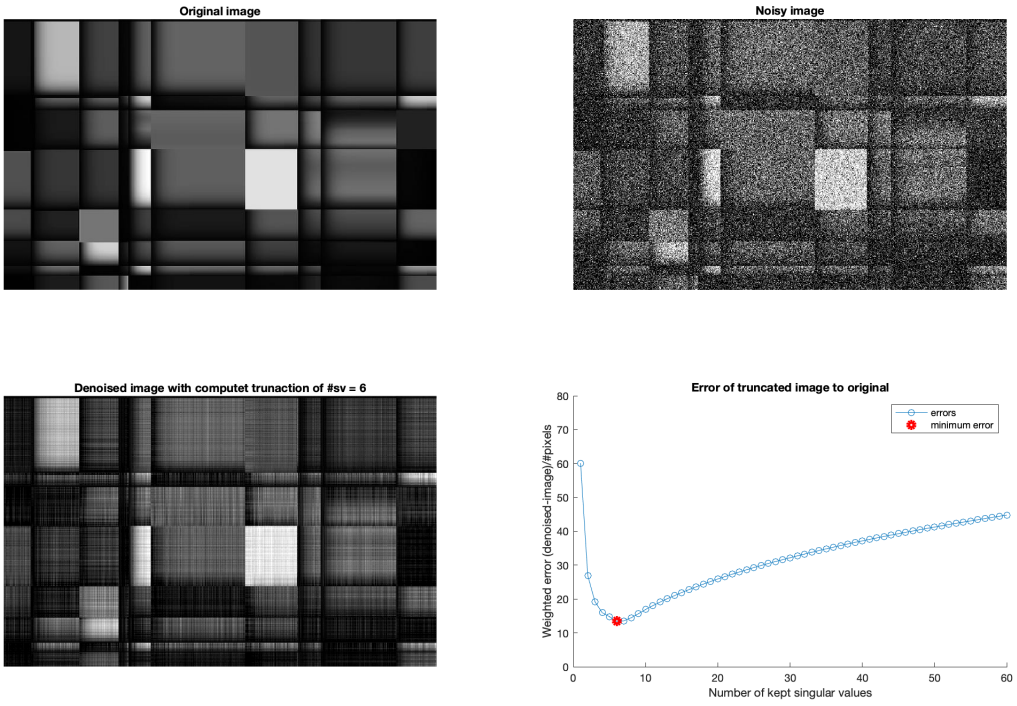

Figure 4.3: Denoised image and optimal truncation on low rank system

High Rank System

As the previous example has been with a low rank picture, a **high rank system** is dealt with next. Therefore the portrait of Einstein from the compression chapter is used again. It is a high resolutional picture, with high rank.

With this picture the same steps are run through as with the low rank system. Again the linear approximation from above is used to calculate the optimal truncation. The determined value is . In the following figure 4.4 there are again plotted the original image, the noisy picture, the denoised picture with the calculated optimal truncation, and the error between the denoised and the original image. In this last plot the minimum is again marked in red.

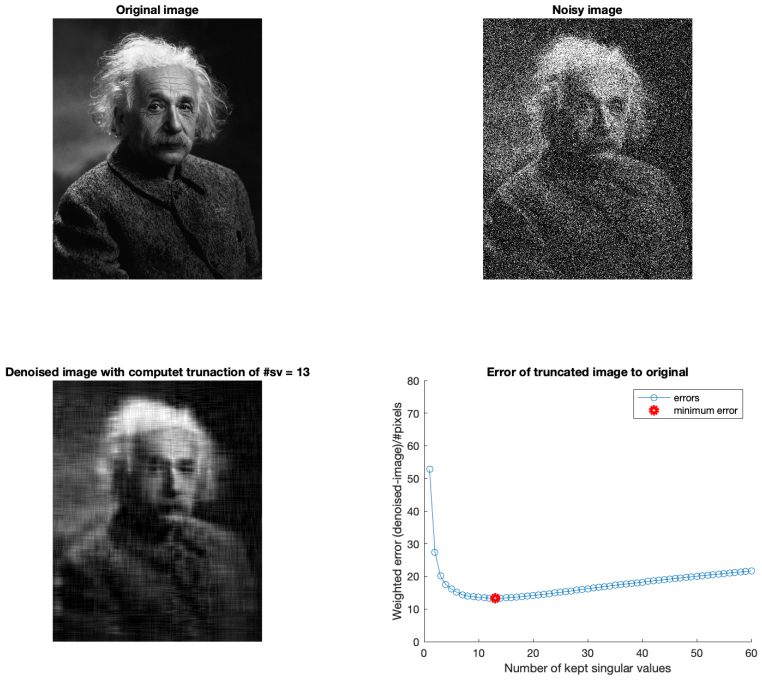

Figure 4.4: Denoised image and optimal truncation on high rank system

From figure 4.4 can be concluded, that also for high rank systems this linear approximation in order to find the optimal truncation (with Gaussian white noise) performs unexpectedly well, as the calculated truncation coincides with the best truncation again. However the result isn't as good as in the low rank system.

4.1.3 Conclusion

With the SVD truncation method, it is possible to truncate noisy data to specific thresholds in order to denoise it. This method performs very good for strongly aligned matrices, as mentioned before. However if the data is satisfying the requirement, it is possible to achieve a good 'best truncation prediction' with linear approximation of the noise distribution in the singular values.

4.2 Denoising Data with FFT

4.2.1 Theory

From 4.1.2 the conclusion was, that the SVD truncation denoising method has a limited performance to low rank systems. Therefore another method has to be spoken about in order to denoise high rank data. This is once more the FFT. The basic idea behind the following is, a similar approach as in the compression chapter. Therefore the data is transformed in the Fourier domain. From 3.2.2 the conclusion was, that the main information is captured in some main frequencies, which are in case of images the low frequencies. If all other redundant high frequencies capturing minor details are filtered out, it is hopefully possible to reconstruct a denoised version. In the following section this will first be processed on a simple signal.

4.2.2 Signal Denoising with FFT

The signal which is used in the following example is a combination of 4 sin and cosine functions with different amplitudes and frequencies.

$$signal(x) = 4 \cdot sin(2\pi \cdot f_1 \cdot x) + 8 \cdot sin(2\pi \cdot f_2 \cdot x) + 4 \cdot cos(2\pi \cdot f_3 \cdot x) + 6 \cdot cos(2\pi \cdot f_4 \cdot x)$$

In this specific example the frequencies wer chosen to be:
$f_1 = 100Hz, f_2 = 60Hz, f_3 = 20Hz, f_4 = 5Hz$.
This signal is taken as the initial data. In the next step normal distributed white noise is added, such there was created some noisy data. In figure 4.5 the upper plot shows the initial and the noisy data. Further this noisy data is Fourier transformed and the powerspectrum is calculated. In the second plot of figure 4.5, the one sided powerspectrum with the frequencies scaled in Hz is shown.

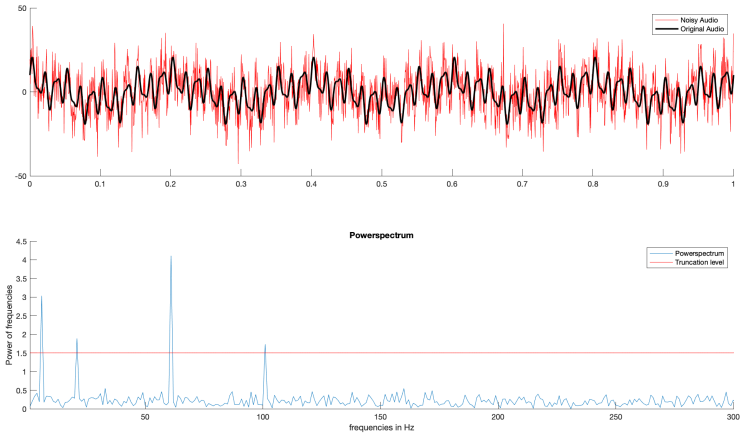

Figure 4.5: Top: Noisy signal and initial signal Bottom: Powerspectrum of noisy signal

There can be observed, that in the power spectrum actually there are four frequencies, that are by far the most dominant frequencies. The frequencies of the peaks in the powerspectrum in Hz coincide with the four frequencies f_1, f_2, f_3, f_4 from our initial signal. As well there can be seen a correlation between the coefficients in the term of the initial signal and the height of the peaks.

By filtering out all frequencies under the red line, the redundant ones are eliminated, such that it is now possible to reconstruct the initial signal with the inverse FFT. Figure 4.6 shows the overlaid initial and denoised signal. There can barely seen a difference those two graphs.

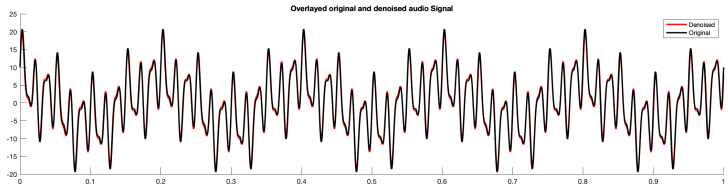

Figure 4.6: Overlaid initial and denoised signal

4.2.3 Denoising Image with FFT2

In 4.1.2 there was tried to denoise a high ranked system (example picture of Einstein), with the SVD truncation method. However the conclusion was that only low rank systems can effectively be denoised with this method.
Following the same noisy picture was used again to show the improvement in performance, with the FFT filtering method. As already mentioned in 4.2.1 the noise in images is usually represented in the high frequency modes. Therefore we make use of it and eliminating all Fourier coefficients of the high frequencies. Such that we build a filter only letting pass the low frequency spectrum. In the code example this filter is constructed as a simple square shape letting pass the center coefficients of the FFT shifted frequencies. In figure 4.7 the denoised image, the original image as well as the noisy image are shown. Additionally there are shown the fft shifted Fourier coefficients below the corresponding picture.

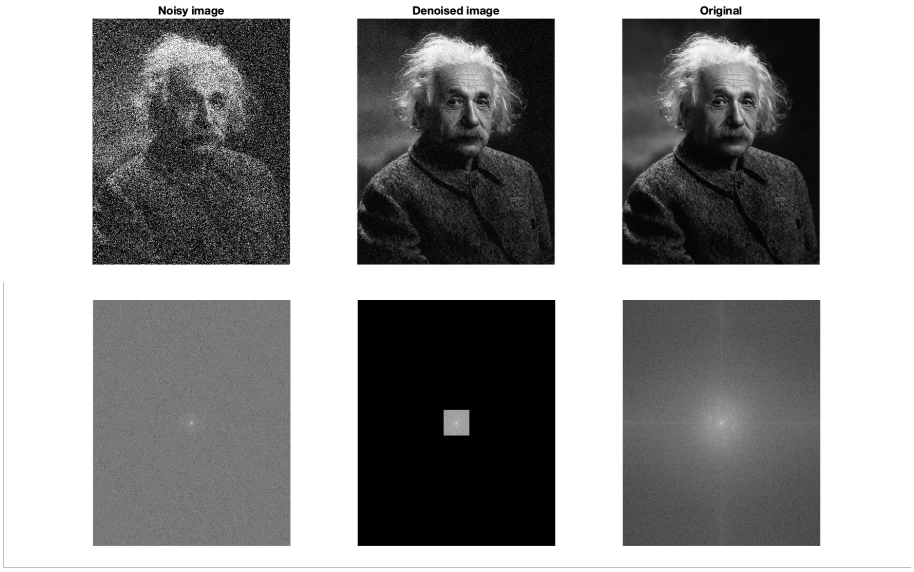

Figure 4.7: Noisy Einstein picture denoised with FFT filtering

The second example that is shown here, is a common denoising problem. The picture contains high frequent noise. In figure 4.8 the original noisy picture is shown in the upper left subplot. Its Fourier coefficients on a log scale are shown below. There can be noticed all the small peaks in the high frequent areas, which we can't see in 4.7 on Einsteins Fourier coefficients. This might indicate, that these are the frequencies causing the noise. In the first attempt of denoising, there is used again the simple filter only passing the low frequent coefficients. The filter and the denoised image are depicted on the right side. There can be seen a drastic reduction of the noise, however also there is a great loss of detail in the picture.

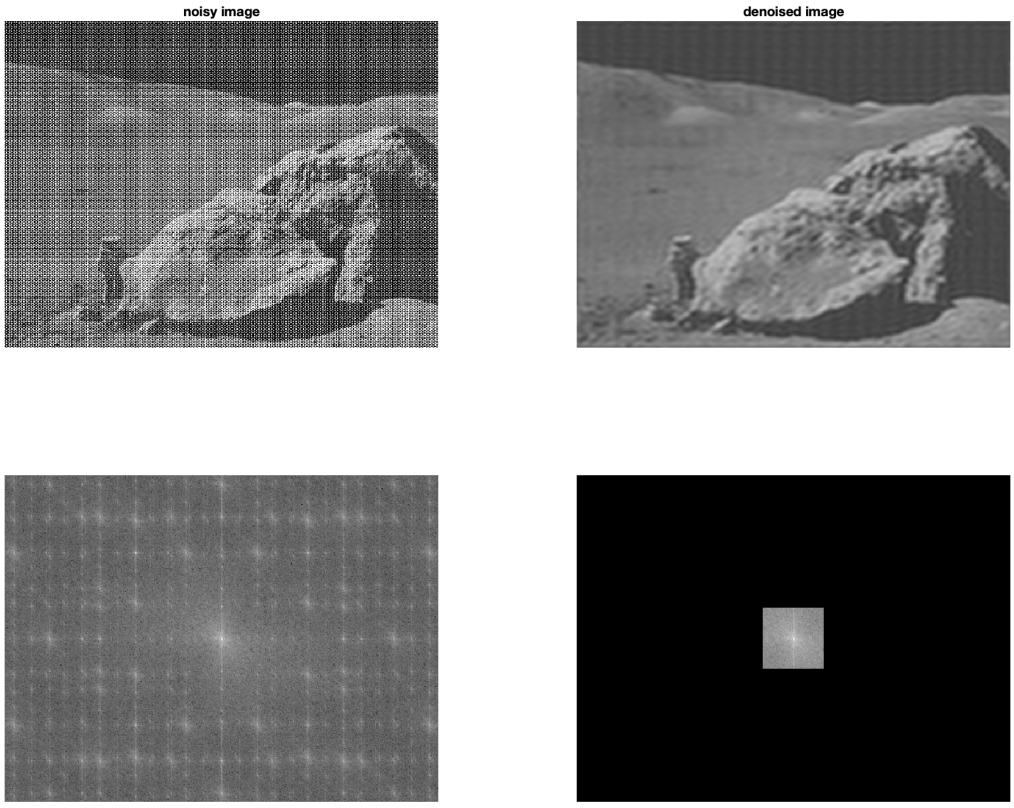

Figure 4.8: Low frequency filter denoising

In order to increase the detailed information contained by the denoised picture, there will be applied a more complex filter. This filter is consisting of three individual filters. The first filter is again the squared filter passing all the low main frequencies as in the previous attempt. Further there is an additional filter which is eliminating all high frequencies in a frame around the inner square. This middle layer is necessary, as here still important information is stored, such that the Fourier coefficients are lager than at the edges of the spectrum. From the rest of the coefficients also all high peaks are eliminated, however the threshold is set lower than on the middle layer, as the mean value of the coefficients is lower. These filters can be seen in figure 4.9. In the left subplot the coefficients are plotted as a surface with the filtering layers visualised. Right the filtered coefficients are again shown in a log scale. The eliminated peaks can clearly be identified as black holes.

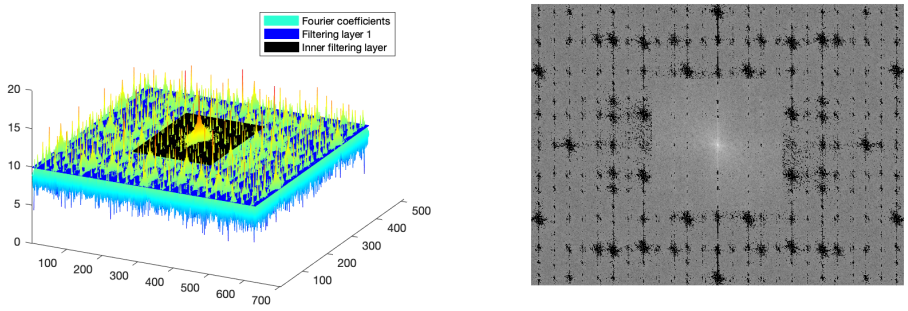

Figure 4.9: Filter applied for denoising high frequent noise

The resulting denoised image shown in figure 4.10 is clearly more detailed than in the first attempt in figure 4.8. This means, that in this case, the noise wasn't randomly distributed, such that it was possible to eliminate the specific noisy frequencies. This led to a smaller loss of details in the image.

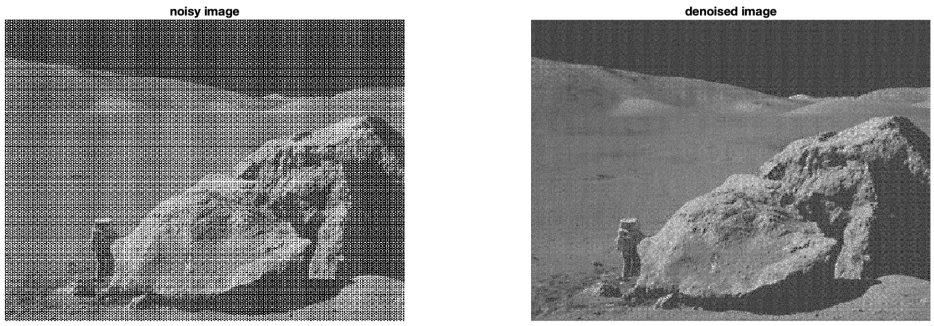

Figure 4.10: Layered filter denoising

4.2.4 Conclusion

The FFT can be used to find the dominant frequencies in a set of data. These dominant frequencies can be identified because of their peaks in the powerspectrum. With respect to images the main frequencies are found in the low frequency spectrum. In case peaks appear in the high frequent spectrum, there exist highly frequent noise, which can be locally filtered. Consequently there can be kept much detail of the picture. If the noise is randomly distributed, all higher frequencies contain noise, such that with the denoising also detail will be lost.

5 Prediction - Regression

5.1 Linear Regression

5.1.1 Theory

Given in a data matrix X, of which each row corresponds to some output b. In a two dimensional case X is a given vector with data, with the corresponding output b. In this case it can be visualised in a coordinate system with the data vector X on the x axis and the output on the y axis.
The aim of linear regression is to find a least square fit linear line to all datapoints. In order to solve this problem the so called 'pseudo inverse' is used. [3]

$$\text{Equation of straight line: } c_0 + c_1 \cdot X = b$$

$$\begin{pmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}$$

$$X^{-1}Xc = X^{-1}b \Rightarrow c = X^{-1}b$$

$$\text{SVD of X: } X = U\Sigma V^T$$

$$c = (U\Sigma V^T)^{-1}b = V\Sigma^{-1}U^T b$$

The 'pseudo inverse' is using the so called 'economy SVD', which uses a special attribute of the matrix product to simplify the matrix multiplication. The Σ matrix can be reduced to a diagonal squared matrix by eliminating the block of zeros. Consequently also the corresponding columns or rows of U or V have to be eliminated. This operation doesn't change anything on the matrix product and enables us to take the inverse of the matrix Σ. Also the numerical error is decreased by using the simplified pseudo inverse, rather than determining the Inverse of a huge data matrix. Following there can be determined the following term for the least square fit line.

$$b = X \cdot c$$

This equation can now be used to predict the outcome b for data X in a linear model.
In figure 5.1 the linear regression in a single variate case is shown. The datapoints were computed with a linear function, to which there was added normally distributed white noise. The linear regression plotted in red is nearly identical to the true slope (blue, dashed line).

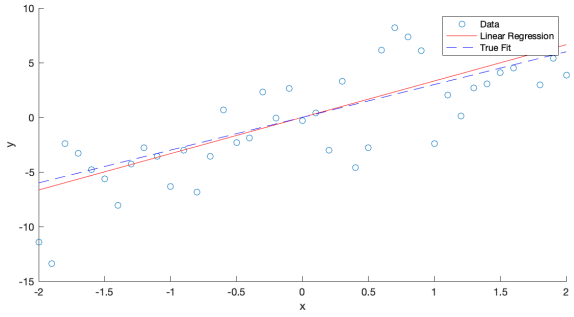

Figure 5.1: Single variate linear regression

5.1.2 Linear Regression Prediction on Linear Dataset

The dataset which is used in this example contains information about car models and their fuel efficiency. This datasets contains 398 instances in a time span from 1970 to 1982. Each line of the datamatrix contains the following information.

1. mpg: miles per galon fuel efficiency
2. cylinders: absolut number
3. displacement: engine displacement in cubic meters
4. horsepower
5. weight
6. acceleration
7. model year
8. origin: (1:America, 2:European, 3: Japanese)

The aim of the following linear regression is to predict the information about the fuel efficiency contained in the first column. Therefore regression constants c_i have to be found, such that the following equation is satisfied with a minimal error e.

$$b = c_0 + c_1x_1 + c_2x_2 + c_3x_3 + c_4x_4 + c_5x_5 + c_6x_6 + c_7x_7 + e$$

Therefore again the pseudo inverse is used as already explained in 5.1.1. In the multivariate case, the regression constants are given by a vector.

$$X = U\Sigma V^T \Rightarrow X^{-1} = (U\Sigma V^T)^{-1} = V\Sigma^{-1}U^T$$

$$c = V\Sigma^{-1}U^Tb = V \begin{pmatrix} \sigma_0 & & \\ & \ddots & \\ & & \sigma_7 \end{pmatrix}^{-1} U^Tb$$

Estimate of $b = X \cdot c$

In the following two regressions the importance of the correct choice for the trainings dataset will be shown. With the trainings set, the regression constants, will be determined. For the prediction a completely new part of the dataset is used to verify the regression quality.

In the first case, the first 250 rows are taken for the trainings dataset. As the data is sorted by time, the newer car models aren't represented in the trainings dataset. This may lead to a poor performance, in the prediction of the test dataset, containing these newer car models.
In figure 5.2 there are shown three plots. The first and second plot contain the prediction of the train dataset (in red) and the true fuel efficiencies (black). The two subplots on the right contain the prediction of the test dataset (in red) and the true values (in black). For better visualisation the data got sorted by ascending true fuel efficiency in the lower plots.

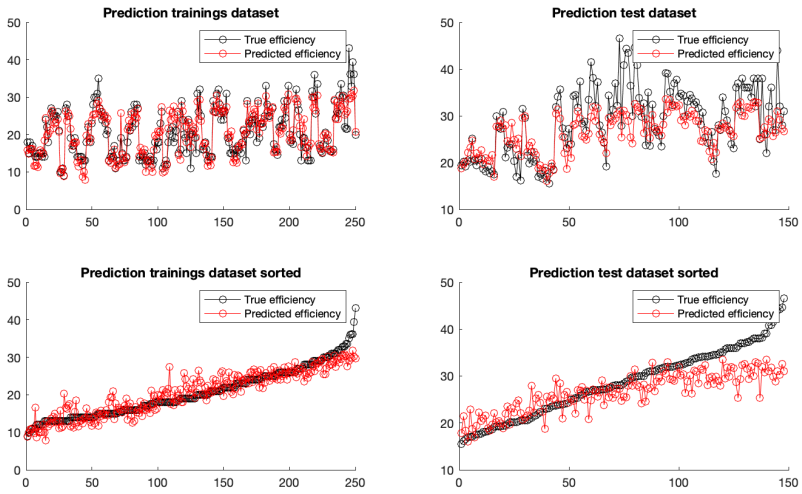

Figure 5.2: Multivariate Linear Regression on car fuel efficiency with badly chosen trainings data

There can be observed, that the regression model perfectly fits the training data (older car models), however it performs horrible in the test dataset.
This already is an indicator, that the trainings dataset was chosen wrong. The missing representation of the newer car models may lead to worse predictions.
In the second case there will be used an evenly distributed trainings data set which contains every second row of the dataset. The same plots as for the first case are shown in figure 5.3

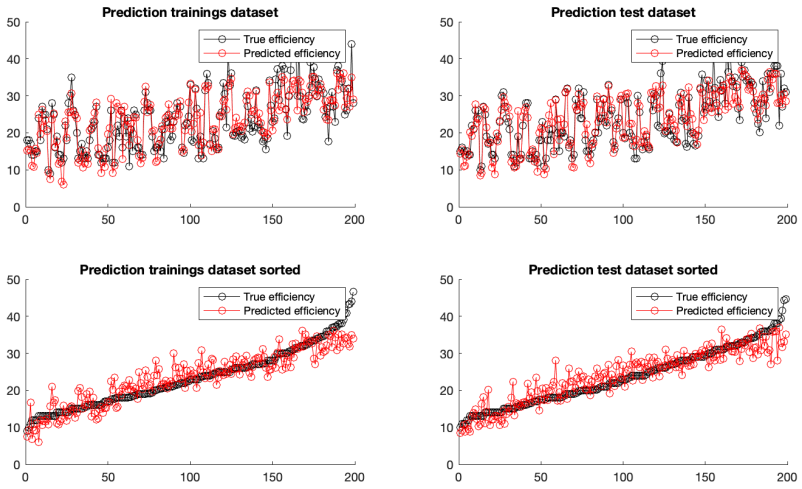

Figure 5.3: Multivariate Linear Regression on car fuel efficiency with better chosen trainings data

As a result of the different choice in the trainings dataset, a better regression in the test dataset is reached. Consequently this is the proof, that the missing representation of the newer car models led to the worse prediction in the first case.

Another rather interesting question is how the individual factors (columns) contribute to the fuel efficiency. Therefore the values of the regression constants can be analysed. This however doesn't work with the normal dataset, as the different variables are not expressed in the same unit of measurement. Thus the size of the regression constants depend on the units of measurement. Therefore it is crucial to outweigh this dependence before modelling the linear regression. This is done by expressing the factors in a standardised form. This standardised form is computed by dividing the deviation of the individual factor to the mean of the factor, through the standard deviation. Such that the equation to determine the standardised factors is given by:

$$\tilde{f}_i = \frac{f_i - \sum_{i=1}^{N} f_i \frac{1}{N}}{\sqrt{\frac{1}{N-1}\sum_{i=1}^{N} |f_i - \frac{1}{N}\sum_{i=1}^{N} f_i|^2}} = \frac{f_i - mean(f)}{std(f)}$$

In figure 5.4 the correlations of the individual factors to the fuel efficiency are shown.

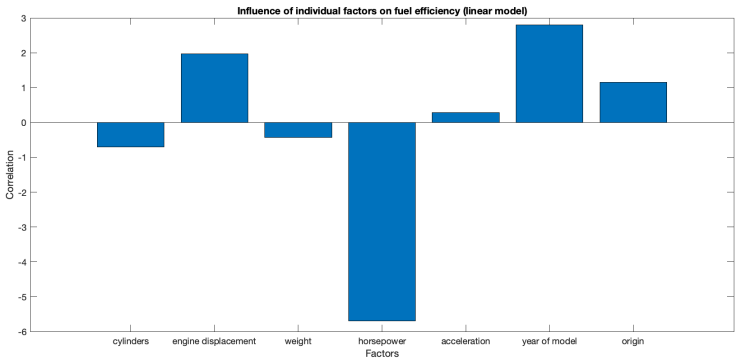

Figure 5.4: Correlation of individual factors to fuel efficiency

Negative values can be interpreted as negative contribution, as well as the absolute value as the importance of the factor in the linear model. Consequently we can also tell that it has been crucial to modify the training data after the first prediction, as the year of the car model is the second most important factor. The most contributing factor is the horsepower, which is actually negatively contributing. This means that cars with more horsepower have a worse fuel efficiency (in the linear model).

5.1.3 Linear Regression on Nonlinear Problems with Linearization

The previous data set had good linear correlations, such that it was possible to approximate and predict with an ordinary linear regression. However not all datasets are having this linearity, such that there has to be done some more analysis before prediction.
The following example is a dataset containing altitude, temperature, humidity and pressure of the earths standard atmosphere.
Target is to predict the air pressure from the humidity, altitude and temperature data. First this will be done by a classic linear regression as in the previous example. The outcome is shown in figure 5.5.

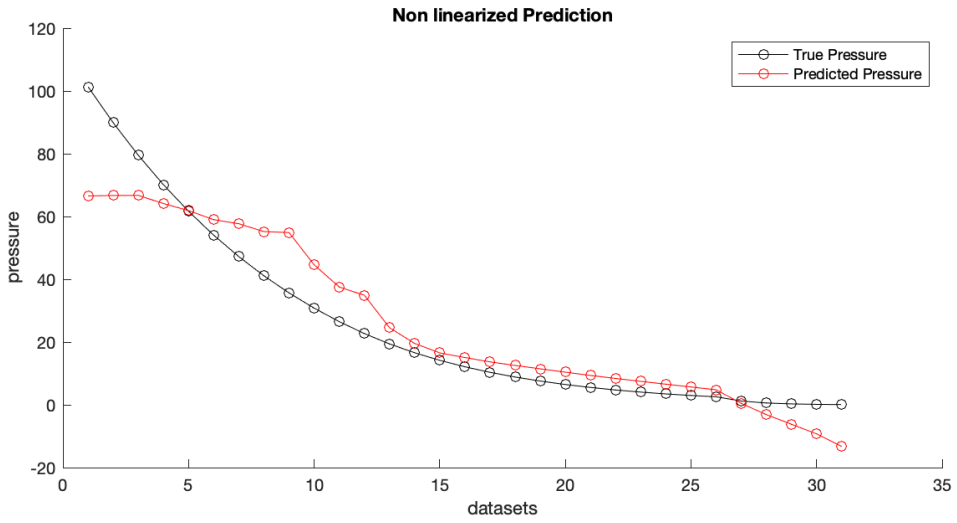

Figure 5.5: Non linearized normal linear regression

It is obvious that this is a very poor prediction, such that it is necessary to determine the correlation between every single variable and the pressure. These correlations are shown in figure 5.6

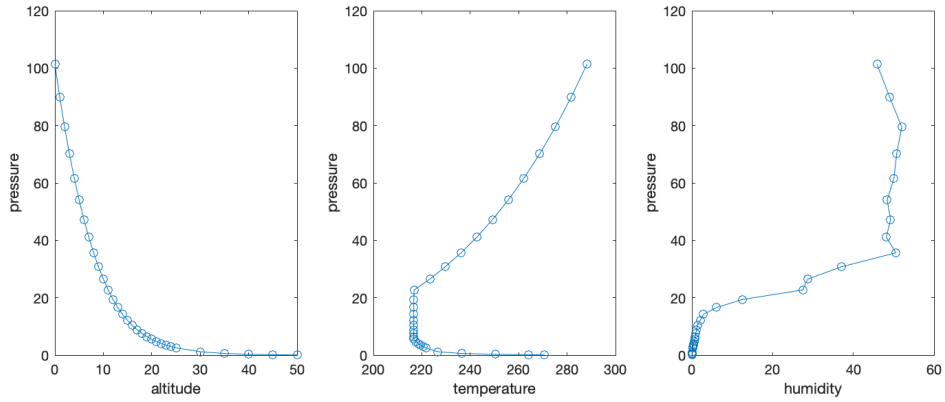

Figure 5.6: Correlations of individual variables to the atmospheric pressure.

From this plot there can immediately be seen an exponential correlation between altitude and pressure. This can't be expressed in a simple linear expression, such that most analysts would suggest to use a non linear prediction model. These are difficult to implement and would go too far for this estimate.
Therefore it is possible to linearise the problem. This is done by taking the logarithmic height. Consequently the exponential relation appears to be a linear problem, perfect for linear regression. In other non linear correlations this can be adapted and used as well. Fore example in quadratic or cubic relations. This also leads to the next section 5.2 on polynomial regression. The prediction of the linearised system is given in figure 5.7.

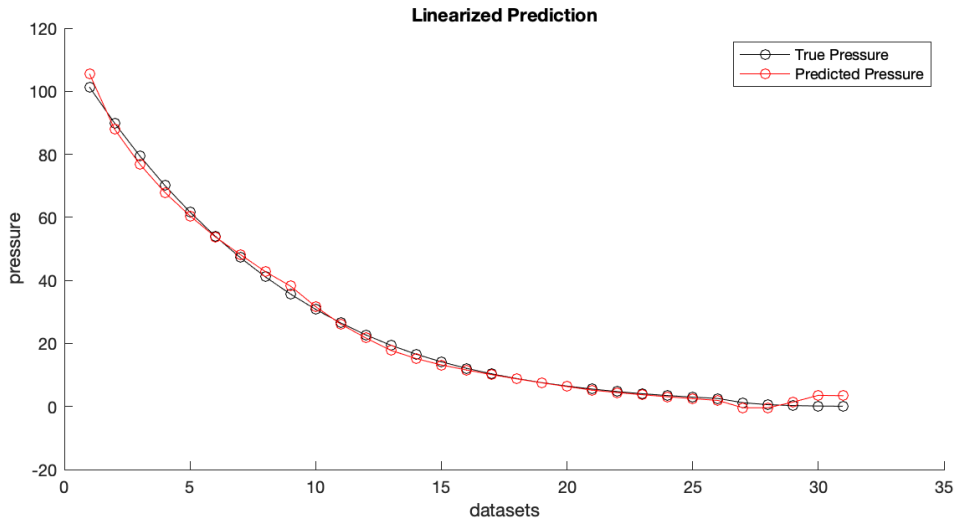

Figure 5.7: Linear Regression model on linearised system.

5.1.4 Conclusion

The linear regression is a basic regression method for finding a least square minimised straight line for some data. This can be performed with single variate or multi variate problems. In most cases it is used for prediction, such that the regression coefficients are determined with the help of a trainings dataset, before predicting the real data. This is one of the easiest forms of supervised learning. Most commonly it is used in linearly correlated cases, however also simple non linear systems can be linearised.

5.2 Polynomial Regression

5.2.1 Theory

In case of more complex relations, the linear regression tends to under-fit the data. Therefore the complexity of the model has to be increased. This can be achieved with polynomial regression.
Polynomial regression is still considered as a linear model, however has the nature of a quadratic, cubic ... higher order function. These higher order models are gained by adding higher powers of the features to the problem.

$$b = c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n$$

The following example will show a single variate problem, with an under-fitting problem. This is solved by polynomial regression. In this case there were used order $2, 3, 10$ polynomials. The regressions are shown in figure 5.8.

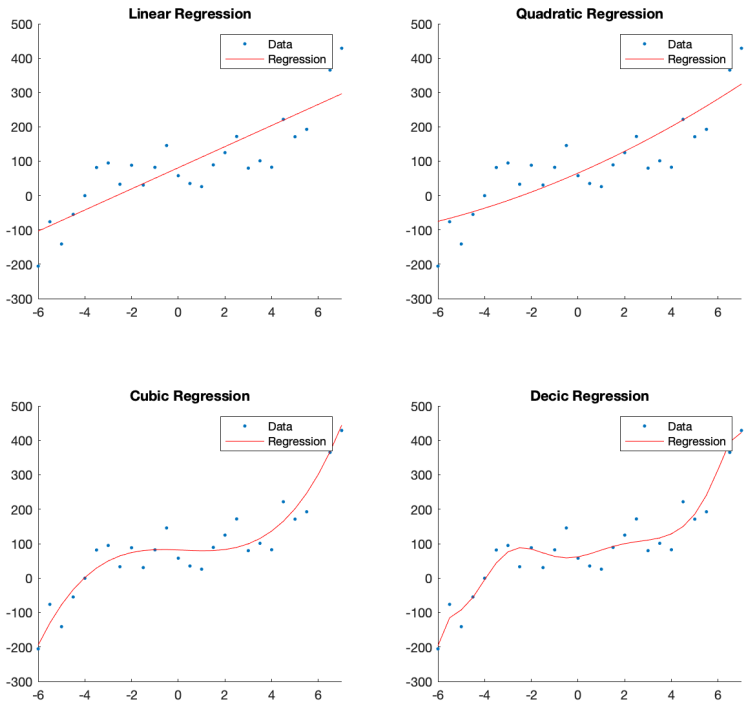

Figure 5.8: Single variate polynomial regression

There can clearly be seen an improvement from the linear to the quadratic and from the quadratic to the cubic regression. Even if the decic regression might capture it best, it overfits the problems. Consequently it won't be useable in a predictive task. For prediction the cubic regression might work best. This shows the act of balance between over-fitting and under-fitting.
The polynomial regression can also be expanded for multi variate problems, as there are simply added higher order terms of each variable.

5.2.2 Polynomial Regression on Yacht Hydrodynamics

The following dataset contains data about the hydrodynamic of yachts. These measurements were taken in 308 individual experiments of 22 different hull forms. The aim of this abstract, is to predict the residual resistance from the geometry of the ship. The residual resistance is the energy loss caused by vessel and viscous pressure resistance.
The aim is to determine this factor for 8 of the 22 boat hull forms. Therefore the trainings dataset is reduced to 14 different hulls.

1. **Linear Regression:**
   In this first regression model there is used the common linear regression of form:

$$b = c_0 + c_1 \cdot x_1 + c_2 \cdot x_2 + \cdots + c_7 \cdot x_7$$
$$b = \begin{pmatrix} \vdots & \vdots \\ 1 & X \\ \vdots & \vdots \end{pmatrix} \cdot c$$

   The regression constants $c_0, \cdots, c_7$ are determined with the pseudo inverse again.

2. **Quadratic Regression:**
   In the second regression model there is used a quadratic regression of form:

$$b = c_0 + c_1 \cdot x_1 + \cdots + c_7 \cdot x_7 + c_8 \cdot x_1^2 \cdots + c_{14} \cdot x_7^2$$
$$b = \begin{pmatrix} \vdots & \vdots & \vdots \\ 1 & X & X^2 \\ \vdots & \vdots & \vdots \end{pmatrix} \cdot c$$

   Also in this. case the constants were again determined with the pseudo inverse.

3. **Cubic Regression:**
   In the third regression model there is used a cubic regression of form:

$$b = c_0 + c_1 \cdot x_1 + \cdots + c_7 \cdot x_7 + c_8 \cdot c_1^2 \cdots + c_{14} \cdot c_7^2 + c_{15} \cdot x_1^3 + c_{21} \cdot x_7^3$$
$$b = \begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ 1 & X & X^2 & X^3 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix} \cdot c$$

   Also in this. case the constants were again determined with the pseudo inverse. However therefore the matrix product was calculated with the pinv() command, for reduction of accuracy, through numerical error.

In figure 5.9 the predictions of all three models can be compared. On the left hand side is the linear model , in the middle the quadratic model and on the right hand side the cubic model. The predictions of all models are plotted in order (upper plot) and sorted by the true target value (lower subplot).

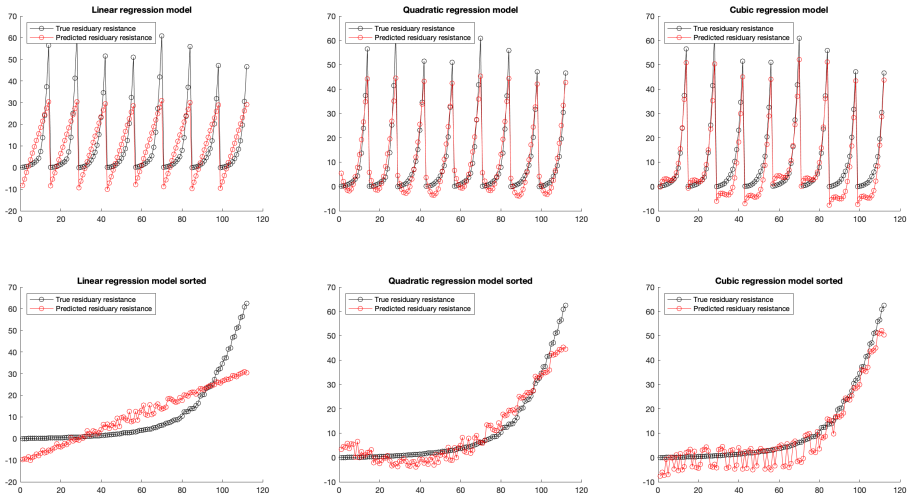

Figure 5.9: Linear and quadratic regression on yacht hydrodynamics

Validating all three models, there has to be said, that there is a significant increase of accuracy from the linear to the quadratic model. However from the quadratic to the cubic model there can't be observed such a growth in accuracy. Instead it tends to show signs of overfitting in some parts of the prediction. Form this cross validation, there can now be chosen the best performing model. In this case the choice would be the quadratic model.

5.2.3 Hurricane Track-forecasting with polynomial regression

Introduction and Data

The following abstract is based on a model used by the NHC (National Hurricane Center) to predict Hurricane tracks with the help of persistence. The forecasting method is described in [4]. The data given for prediction is a set of historical storm tracks. This is in this case the HUDRAT2 best track dataset. In order to achieve better performance, all storms, that were observed less than five days are eliminated from the dataset.

With the statistical CLIPER method, which will be interpreted and reconstructed in this abstract, are performed predictions in timespan from 12 to 72 hours. For simplification, this will be reduced to a 6, 12, 18 and 24 hour prediction.

Basic and adapted Predictors

For all instances the following basic predictors are available.

Predictor	Abbreviation
Initial longitude	X_0
Initial latitude	Y_0
Initial E. to W. component	U_0
E. to W. component 12 hrs. ago	U_{-12}
Initial S. to N. component	V_0
S. to N. component 12 hrs. ago	V_{-12}
Sustained Maximal Windspeed	W
Calendar day	D

In the paper by Neumann, these basic predictors were modified, for zonal and meridional prediction. In the following table these are listed.

Nr.	Predictors Meridional	Predictors Zonal
1	Y_0	X_0
2	V_0	U_0
3	V_{-12}	U_{-12}
4	$V_0(V_{-12})^2$	$Y_0 - 24$
5	$(W - 71)V_{-1}$	V_0
6	$V_0(W - 71)$	$V_0^2 U_{-12}$
7	$V_0^2 U_{-12}$	$(Y_0 - 24)V_0 U_{-12}$
8	$(Y - 24)^2 V_0$	$X_0 - 68$
9	$(D - 248)^2 V_{-12}$	
10	$V_0(D - 248)^2$	
11	$(Y - 24)^2(D - 248)$	
12	$(W - 71)(D - 248)V_{-12}$	
13	U_0	
14	$(D - 248)^2$	

Further in the paper [4] there is descried how the zonal and meridional predictions are calculated. This is a linear regression. In the following formula there is only shown a prediction of one specific timespan as for example 12 hour.

$$dX = c_0 + \sum_{i=1}^{8} c_i P_i$$

$$dY = c_0 + \sum_{i=1}^{14} c_i P_i$$

The performance of the self coded prediction in this linear type performs worse than the original method. This may be a result from using a different dataset or a least square regression, which is much more sensitive to outliers than the L1 regression.
In the following plot 5.10 there are shown the zonal and meridional predictions for forecasting periods of 6,12,18 and 24 hours in red and the true values in black. These were sorted, in order to have a better visualisation.

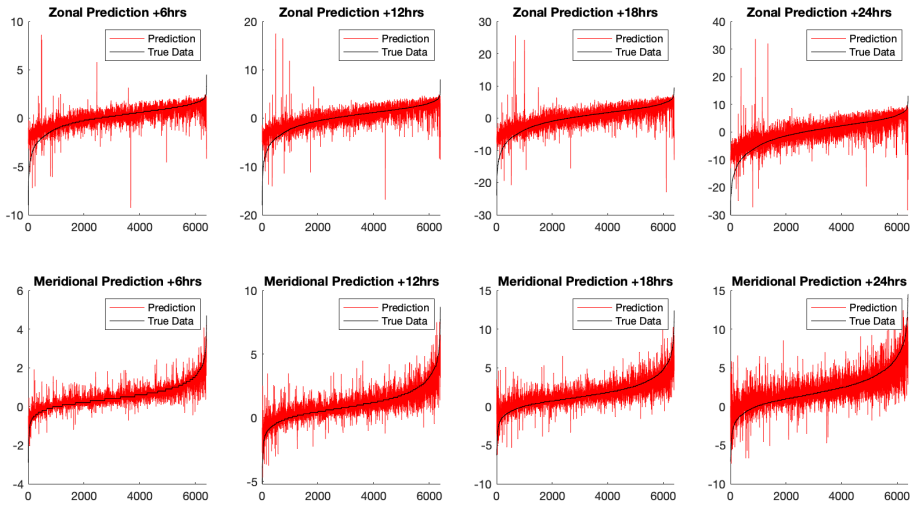

Figure 5.10: Linear CLIPER model predictions

The error of the computed zonal and meridional predictions is calculated as a difference from the true values. The total error is calculated as follows:

$$E = \sqrt{E_m^2 + E_z^2}$$

The errors of the linear model for the specific forecasting timespans are given in figure 5.11.

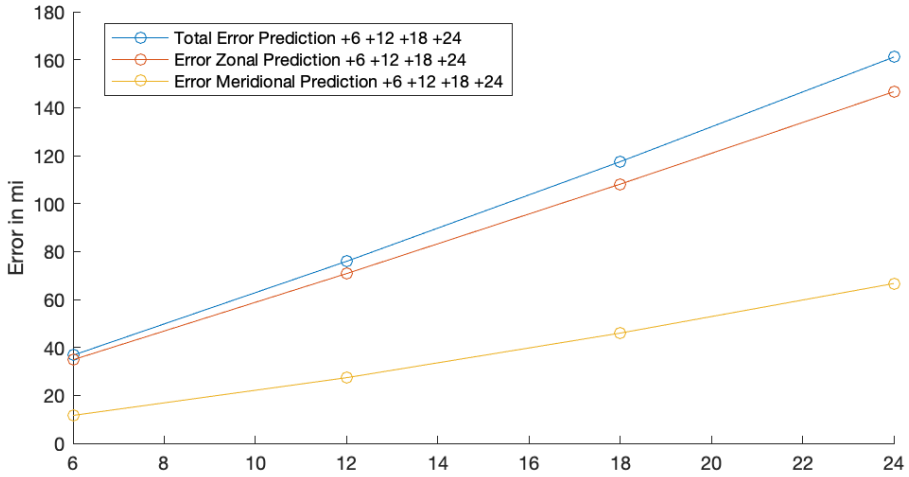

Figure 5.11: Error of the linear CLIPER predictions in nautical miles

Further the predictions of twenty arbitrarily chosen events are determined and visualised in figure 5.12. The map is reduced to the Atlantic basin, as the dataset only contains Atlantic Hurricane Data. The true tracks, given by the best track dataset, are marked in black and the predicted Tracks are plotted red. The first track point is the initial position.

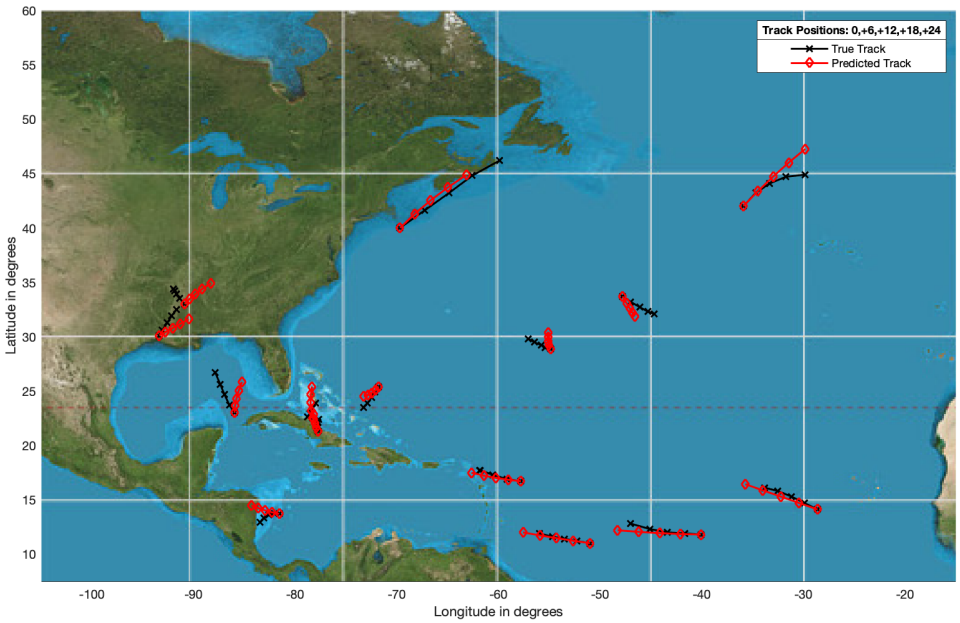

Figure 5.12: Linear CLIPER model prediction of 10 arbitrary events

Optimization with cubic regression

All further ideas are adaptations to the CLIPER model, which are based on the knowledge from the previous sections about polynomial regression. The idea is to optimise the regression by adding higher order terms of the chosen predictors. In this case this will be second and third order terms. This leads to the following regression equations.

$$dX = c_0 + \sum_{i=1}^{8} \sum_{j=1}^{3} c_{i,j} P_i^j$$

$$dY = c_0 + \sum_{i=1}^{14} \sum_{j=1}^{3} c_{i,j} P_i^j$$

This adaptation results in a large number of additional predictors, which will contribute to a more accurate prediction of this highly non linear problem.
In this specific case the data matrix was adapted, such that:

$$\tilde{X} = \begin{pmatrix} \vdots & \vdots & \vdots & \vdots \\ 1 & X & X^2 & X^3 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

The zonal and meridional predictions of the adapted model are shown in figure 5.13. The predictions are again sorted according to the true values, which are plotted in black. There can be seen a significant increase of accuracy, in comparison to the linear regression.

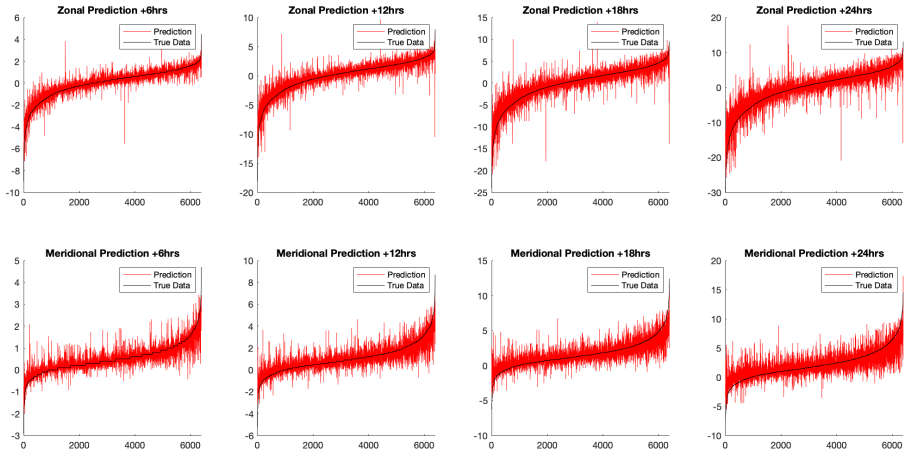

Figure 5.13: Polynomial CLIPER prediction

Also the analysis of the prediction error shown in figure 5.14 shows a significant increase of accuracy. The total error for the 24 hour predictions decreases to about 120 nautical miles and for the 12 hour to about 50 nautical miles. This also is the official mean performance of CLIPER in the timespan from 1983 to 1987 [5].

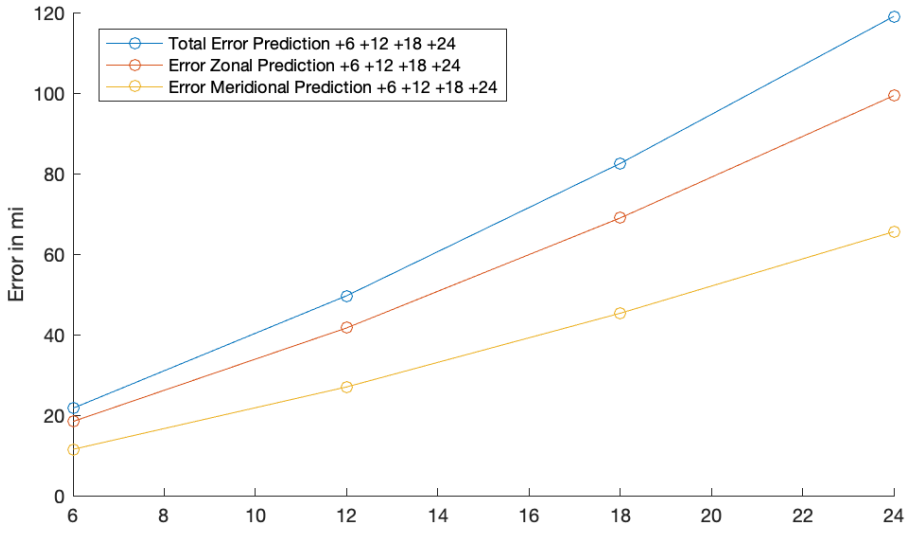

Figure 5.14: Error of the linear CLIPER predictions in nautical miles

The next figure 5.15 is showing the predicted tracks of the optimised cubic regression model. For comparability the same events as in figure 5.12 were taken. Also in those arbitrary chosen cases, there can be observed slight improvements.

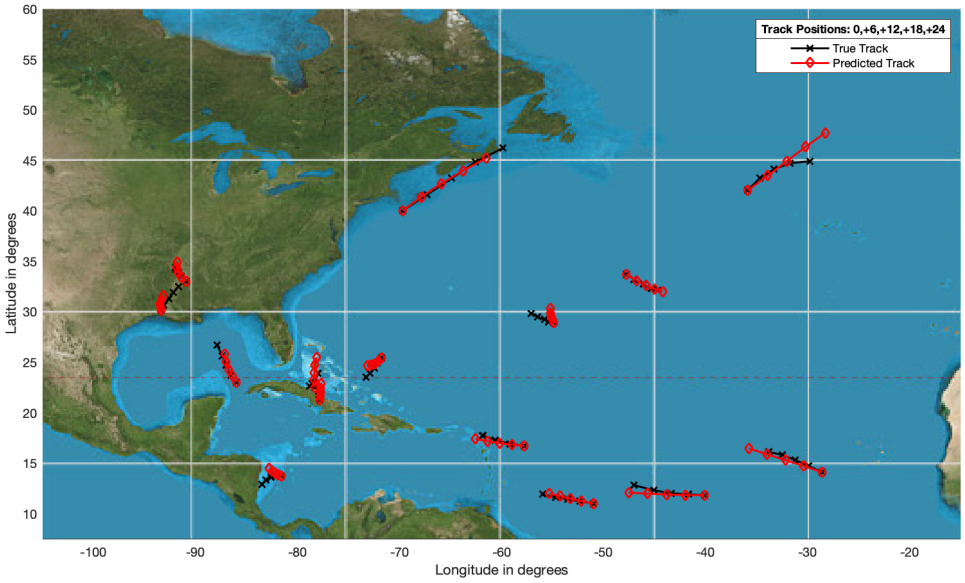

Figure 5.15: Polynomial CLIPER model prediction of the 10 events from figure 5.12

5.2.4 Conclusion

From the improvements achieved by using polynomial regression in the two previous sections there can be concluded, that highly non linear problems are better solved with cross validated polynomial regressions. This leads to a larger number of predictors and regression constants, which contribute differently to the target value.
The cross validation is important, as even if the regression fits the training data very good, it might overfit in prediction. Therefore different polynomial models should ideally be compared before prediction.

5.3 Compressed Sensing

5.3.1 Theory

This section is closely related to the compression chapter 3 in which we have seen, that all natural images and signals are highly compressible. This means, that there exists a sparse representation of the data in some arbitrary domain (sparse means a great part of the data being redundant). As for example in figure 3.9 we have seen, that most Fourier coefficients of the Einstein picture were zero. This leads to the fundamental question, if it is even necessary to have the full original data to reconstruct the high resolution data.
From the Nyquist-Shannon sampling theorem described in [6] there always was the constrain for minimal sampling a sampling frequency minimal strictly greater than the maximal frequency. This however leads to a huge number of samples, even though the signal only consists of few overlapping frequencies. The Nyquist-Shannon sampling theorem actually holds true in very frequent data, in which a lot of different frequencies contribute to a signal. However in most of those cases, these aren't natural signals, as they aren't very compressible at all. In the following figure 5.16 there are plotted a reconstructed signal from samples with $f_{sample} > 2 \cdot f_{max}$ in the upper plot, from which the exact signal can be reconstructed. In the lower subplot there were taken uniformly distributed samples violating the sampling theorem. Form these samples there can't be reconstructed the signal.

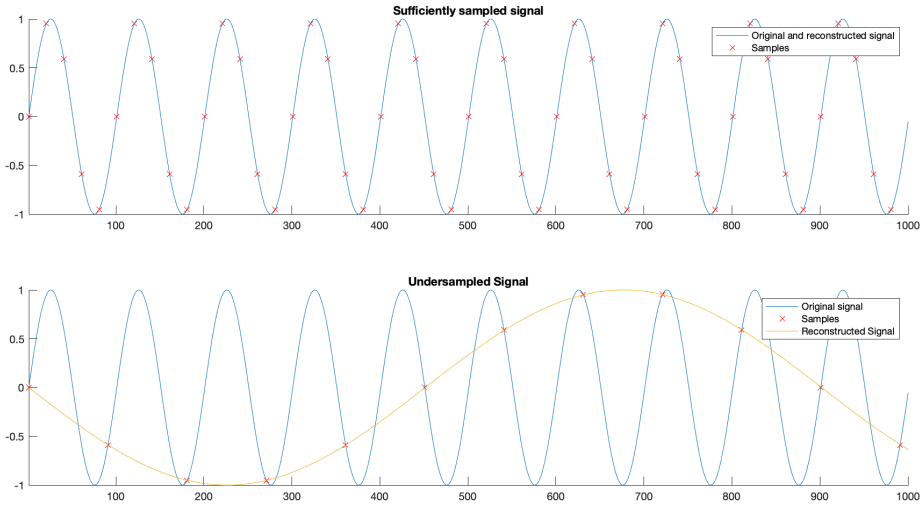

Figure 5.16: Nyquist-Shannon sampling theorem

From this theorem there would be no chance sampling less frequent than $2 \cdot f_{max}$. This however is a very strict limitation. In order to beat the Nyquist-Shannon sampling theorem there is used compressed sensing. There are two requirements to be able to reconstruct a signal from fewer samples than allowed from the theorem. [8]

1. Original data has to have a sparse representation in some domain. (Given for most natural signal)

2. Random sampling (not evenly spaced)

If both those requirements are satisfied, it is possible to solve a highly underdetermined system. The randomly sampled measurements are written as a vector b. The time correlation between the random measurements and the high resolutional data is written in a matrix form (R). Each row corresponds to a specific sample of the downsampled data and each column is a sample point in the high resolutional data. (For example if our first downsampled measurement is taken at sample point 30 in the high resolutional data, the element $S[1, 30] = 1$.) Additionally we have an orthogonal basis Φ of the domain, in which the data has a sparse representation. In those examples there will be used the Fourier domain. Multiplying both matrices leads to the matrix Ψ of the underdetermined system. The last step is to solve the following undetermined system.

$$b = R \cdot \Phi \cdot x$$
$$b = \Psi \cdot x$$

Until now these problems have always been solved by the pseudo inverse in this paper. However this won't give a sufficient solution, as there are infinitely many solutions and the sparsest possible solution x is desired. This problem can't be solved by the pseudo inverse, as in least square regression all factors contribute, such that there will be no zero entries. The problem of finding the sparsest possible solution is a very difficult task and can be transformed into a convex problem. A sparse vector has got a minimal L_1 norm, which is computed as $||x||_1 = \sum_{i=1}^{n} ||x_i||$. This leads to the following minimisation problem:

$$x_{\text{sparse}} = \min_x ||x||_1 + \lambda||\Psi \cdot x - b||_2 \quad \lambda \in \mathbb{R}^+$$
$$x_{\text{sparse}} = \min_x ||x||_1 \text{ s.t. } ||\Psi \cdot x - b||_2 < \alpha \quad \alpha \in \mathbb{R}^+$$

In figure 5.17 this minimising problem is visualised. From left to right there are shown the downsampled measurement vector b, the random measurement time correlation matrix R, the orthogonal basis Φ of the sparse domain and two possible solutions x for the undetermined system. The left vector x is sparse and was determined by solving the described convex minimisation problem with the L_1 norm. The second solution is not a sparse vector, which resulted from the pseudo inverse solution. In this case the solution x represents the Fourier coefficients of the signal. Reconstruction of the full signal is consequently done by applying the inverse FFT.

Remark: In case of the Fourier domain the Φ matrix is the complex conjugate of the discrete Fourier matrix. This can be explained with the algorithm of the inverse Fourier transformation:

$$v = \overline{\frac{1}{n} F_n \hat{c}}$$

Adapted to the compressive sensing notation this means that:

$$b = R \cdot \overline{\frac{1}{n} F_n} \cdot x$$

In the sparse solution x of the L_1 norm optimization, there can be identified the two frequencies, which contribute to the signal. This is the desired sparse solution as the signal which was used in this example exactly consisted of two frequencies with 2 and 5 Hz.

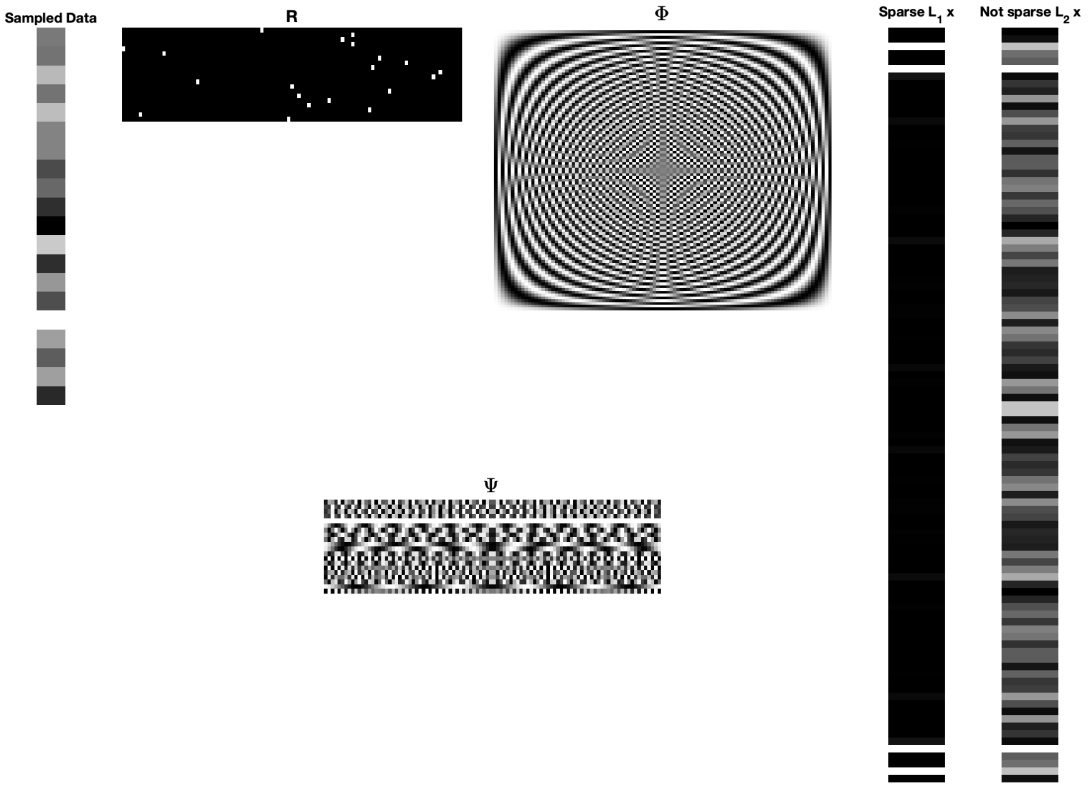

Figure 5.17: Compressed Sensing Theory

5.3.2 Signal Reconstruction with Compressed Sensing

In this section the object of interest will be a signal with a length of one second of form:

$$f(t) = sin(2\pi \cdot 10Hz \cdot t) + sin(2\pi \cdot 70Hz \cdot t) + sin(2\pi \cdot 160Hz \cdot t) + cos(2\pi \cdot 120Hz \cdot t)$$

The original high resolutional signal is sampled at 1000 measurements per second. The signal and its powerspectrum of frequencies, gained by the Fourier transformation are shown in figure 5.18. This is already known form section 3.2.3. The most dominant frequencies are represented by the peaks. In the plot the frequencies are scaled into Hz, such that the frequencies from the formula above can be identified as the x-coordinates of the peaks.
Let's assume that there wasn't captured the full signal, as the measurement process is really expensive. The aim is to measure randomly distributed samples, in order to reconstruct the full signal with compressed sensing. There was chosen to only take 50 measurements instead of the 1000, such that only 5% of the data is known. These measurements are visualised with respect to the original signal in figure 5.19.

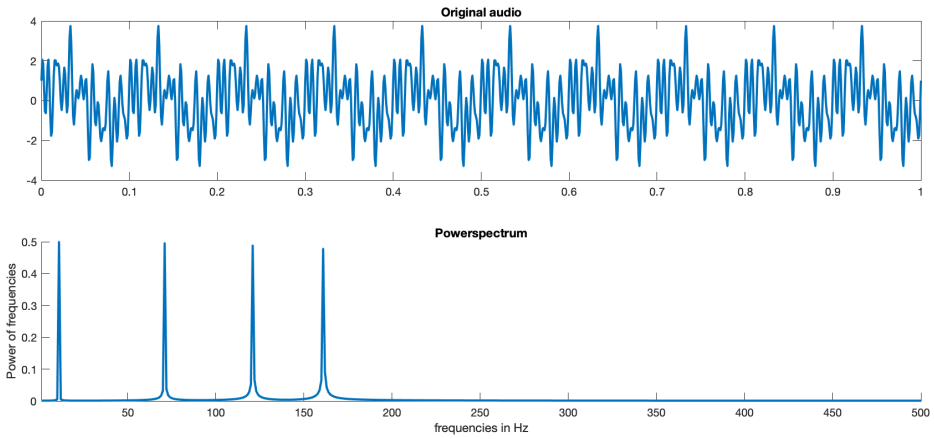

Figure 5.18: Original signal with powerspectrum of frequencies

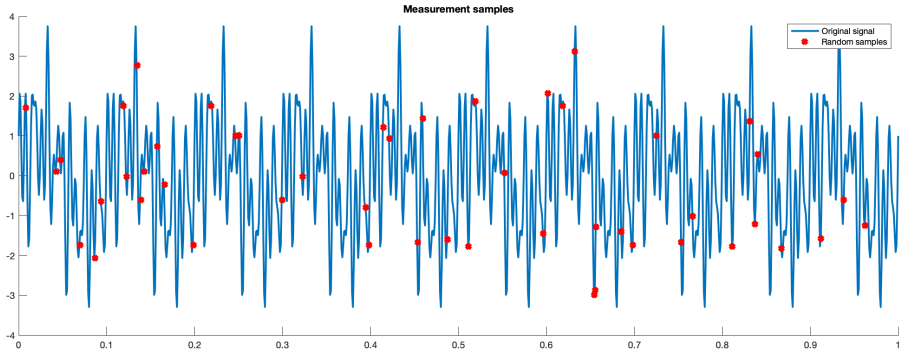

Figure 5.19: Measurement samples (in red)

In the following process of the compressed sensing, there is again used the Fourier domain. Φ is represented by $\frac{1}{1000}F_{1000}$. For demonstration, there are shown both solutions, the sparse L_1 norm solution of x and the least square solution. For both solutions, the gained powerspectrum ($\frac{x \cdot \bar{x}}{1000}$) and the reconstructed signals (by applying the inverse FFT), are shown in figure 5.20. For the sparse solution the reconstruction is nearly identical to the original signal. The least square solution only captures the measurement samples, however doesn't represent the original signal at all.
Also it is possible to identify the peak frequencies from the power spectrum of the sparse solution. Even though they aren't exactly identical to the original powerspectrum, they do capture the most important information.

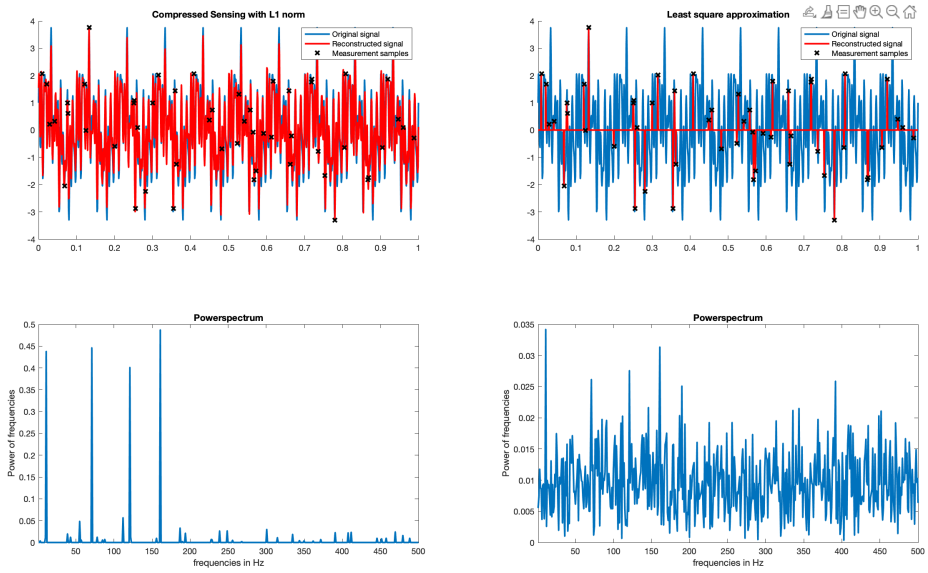

Figure 5.20: Reconstruction of signal with sparse and least-square solution.

This method with only tiny weaknesses can even be optimised. The amplitude of the peaks is a bit different from the original, however the sparsity pattern is clearly visible. This can be used to filter out the important frequencies and perform a least square regression. The pseudo inverse is less expensive and with evenly weighted frequencies it performs better than the L_1 optimisation. The resulting signal and powerspectrum are plotted in the following figure 5.21. It is nearly an identical reconstruction possible.

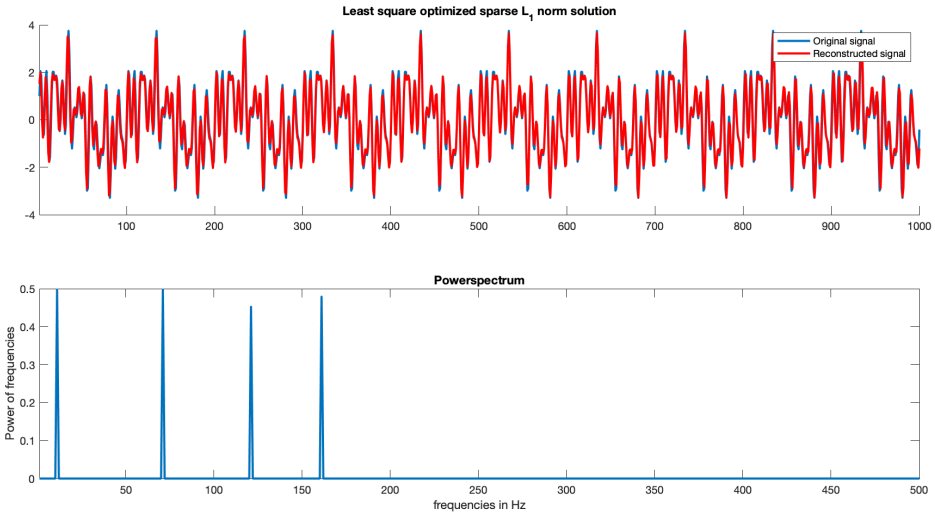

Figure 5.21: Least square optimised sparse L_1 solution

6 Classification

6.1 Classification with Principal Component Analysis (PCA)

6.1.1 Theory

As we have already mastered the prediction of continuous numerical data, it is now time to predict absolut values. This is done in this first chapter, by analysing the principal components. The statistical use of the SVD with the PCA is explained in the following paragraph. The notation of the PCA is adapted to the following Code examples.

1. First there will be crated a mean data matrix M of our given matrix X

$$X = \begin{pmatrix} | & | & | \\ x_1 & \cdots & x_n \\ | & | & | \end{pmatrix}$$

$$\tilde{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \text{Average subject/ column}$$

$$M = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix} \cdot (- \tilde{x} -)$$

2. The unique information of every instance x_i is gained by subtracting the mean matrix M from the data matrix X.

$$B = X - M$$

3. From here there are two possibilities, either the PCA can be computed with eigenvectors of the so called Covariance matrix C. In this step there is the first big difference from the usual notation of the PCA. As normally the data is stored in the rows of the data matrix X, the covariance matrix is usually denoted as written below in brackets. In this case the data is stored in the columns and therefor the covariance is calculated as written in the first equation.

$$C = BB^T = U\Sigma\Sigma U^T = U\Lambda U^T$$
$$(C = B^T B = V\Sigma\Sigma V^T = V\Lambda V^T)$$

   The other possibility is computing the SVD automatically gaining both eigen basis U and V.

$$B = U\Sigma V^T$$

4. The principal components are the first columns of U. However also from the matrix V these principal components can be reconstructed by claculatig $B \cdot V \cdot \Sigma^{-1}$.

5. It is also of interest to know how much variance of the data is captured from the first k principal components.

$$\text{Reached Variance } = \frac{\sum_{i=1}^{k} \sigma^2}{\sum_{i=1}^{n} \sigma^2}$$

6.1.2 PCA Face Classification Algorithm

The aim of this section is, to develop an algorithm for face recognition. It should recognise a person from a totally new picture, taken under different (some in really bad) lighting conditions. The data set used here is the Yale Dataset B. It contains pictures of 37 individuals in each 60 lighting positions. The following figure 6.1 contains the frontally lighted faces of the first 36 individuals, as well as the different lighting positions demonstrated on person one.

Figure 6.1: Yale Data Base B

In the next step there are extracted two arbitrary pictures of each person from the training data, as the test pictures for the prediction should be new to the algorithm.
Finally the different steps of the PCA explained in 6.1.1 can be executed.

1. The data matrix X is now consisting of the 58 pictures of each individual reshaped into a column vector. In this first step the average face is calculated by taking the mean of all columns. This average face M, shown in figure 6.2 is now subtracted from X, such that B is obtained. An interesting observation is, that the average face looks like a young child's face. This is the origin of development, all humans have in common and therefore is represented in the average face.

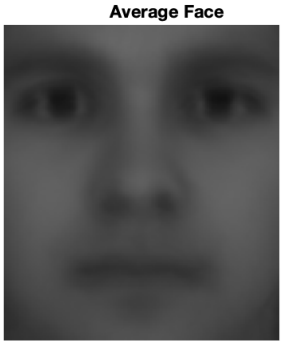

Figure 6.2: Average face

2. The SVD is calculated in order to determine the principal components (in this case also called eigenfaces), which are contained in the first. columns of U. These are identical to the ones reconstructed from V through $B \cdot V \cdot \Sigma^{-1}$. The first 25 eigenfaces are shown in figure 6.3. An important observation is, that still very general information is stored in the first few eigenfaces, as these are very close to the average face. Therefore there can be concluded, that for the recognition process in all further steps, the principal components from the fifth eigenface upwards are of interest.

Figure 6.3: Principal components (eigenfaces)

3. These Principal components span an orthonormal vector-space, onto which the original pictures (from the 'pixel space') can be projected. This is just done by applying the dot product.

$$P_{x_i} = U^T \cdot x_i$$

For visualisation there were only taken three columns of U, in this case the fifth, sixth and seventh principal component. The projections can be plotted in a three dimensional euclidean coordinate system, with the axis representing the different principal components. In the following figure 6.4 this is done with all lighting pictures (contained in the trainings database) for two individuals. The projections of the individuals are marked in different colours. Additional the pictures are plotted right and left of the projections. It is very remarkable how well the two individuals can be identified from the projection onto only three principal components.

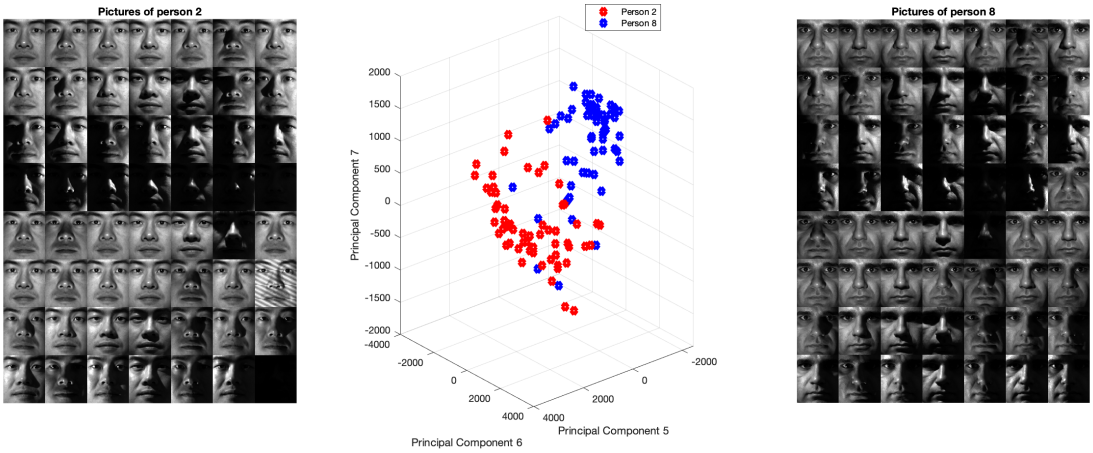

Figure 6.4: Projection onto the eigenface space.

4. Further there is generated the mean face of each person, from the 58 different lighting positions in the trainings data. These pictures give a detailed picture with very high structural information, because of the shadows of the different lightings. These mean faces are given in figure 6.5

Figure 6.5: Mean face of every individual from 58 different lightings

5. Further these mean faces are getting projected onto specific Principal components. For the choice of the principal components it is crucial to look for a good separation of each individual. This leads us to the task to find these principal components, for which the projections are widely and evenly distributed. From the both lower subplots in figure 6.6 there can be concluded, that the first three principal components aren't that suitable for recognition, as it still is a very. dense cloud, whereas the principal components five, six and seven are showing widely distributed projections. This also confirms the theory, that still fairly general information is stored in the first principal components. In the two upper subplots of the figure, there are shown the singular values, as well as the variance captured by the number of singular values. There can be seen a fast increase of captured variance in the beginning.

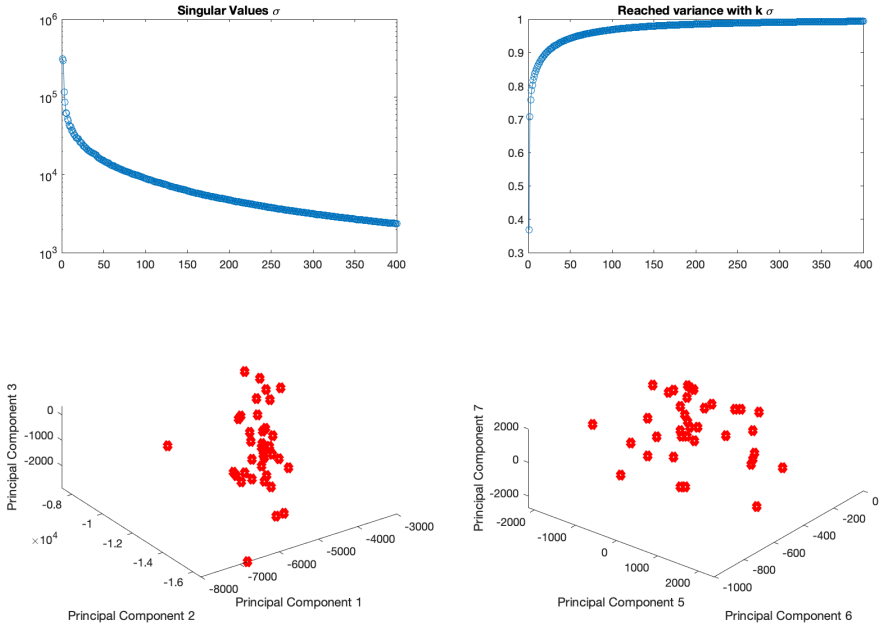

Figure 6.6: Projections and Principal Component variance

6. Last step is now the actual recognition process. As the first principal components result in a densely structured cloud of projections, there are chosen the principal components fife to 25 for the prediction process.
   The testimages, which are totally new to the algorithm are now projected onto the eigenface space of these specifically chosen principal components and compared with the projections of the meanfaces onto the same space. The closest one is chosen, by looking for the minimal difference between the projections of the meanfaces and the testface. In the following figure 6.7 the testimage, the predicted meanface and the projections onto the fifth, sixth and seventh principal component are shown. The corresponding person numbers are written above the graphic. Due to very bad lighting conditions in some of these pictures the algorithm predicts in average about 70% correct.

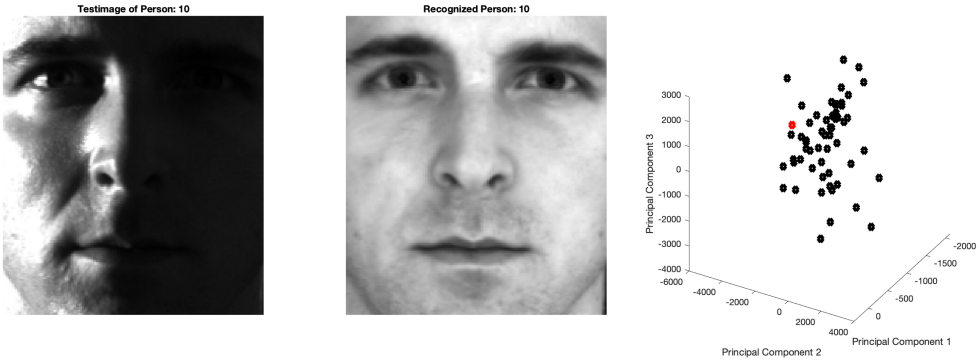

Figure 6.7: Prediction with PCA

6.2 Classification with Sparse Representation

6.2.1 Theory

This section is closely related to 5.3. In the chapter of compressed sensing there was used the sparsity of natural signals in domains as the Fourier domain, in order to get a sparse solution x for a highly underdetermined system. Additionally we were able to reconstruct a signal from randomly distributed samples.
In this section there will be used, that data will also have a sparse solution in a overcomplete library of training data. The overcomplete library will be denoted by Ψ in the following sections, such that there has to be found the sparsest solution for the following undetermined system.

$$b = \Psi \cdot x$$

This can again be written as a convex problem of form:

$$x_{\text{sparse}} = \min_x ||x||_1 \text{ s.t } ||\Psi x - b||_2 < \alpha \quad \alpha \in \mathbb{R}^+$$
$$x_{\text{sparse}} = \min_x ||\Psi x - b||_2 + \lambda ||x||_1 \quad \lambda \in \mathbb{R}^+$$

The solution x represents a linear combination of training data from the overcomplete library in order to reconstruct the test data as close as possible. With a least square solution, this would lead to a solution, with every column of the library contributing. The L_1 optimisation solution however improves the model by reducing the number of strongly contributing data from the library.
This is especially helpful if it is a classification task, in which a test data should be matched to the correct subject in the library.
In comparison to the principal component analysis, it is crucial to have a large enough library of the subjects. If this library however captures a sufficient amount of information the spares representation outperforms the PCA method from 6.1.1. This is not always possible as not always such good library is available. [8]

6.2.2 Sparse Representation Face Classification Algorithm

There is again used the Yale Face Dataset as an in the previous chapter. There are separated two images of each person for validation of the recognition process.
As the system needs to be highly underdetermined for the minimisation problem, the images have to be downsampled. In this case the original size is 192×168 and they were downsampled to 15×13. For the resizing process the lanczos filter is used, as it performs well on small images and reduces ringing in the signal.
The library Ψ consists of all training images downsampled with lanczos as column vectors. This leads to a highly underdetermined system with 195 equations for 2146 unknowns. In figure 6.8 this process is visualised. There can actually be seen the correct peak in the sparse vector x, that represents the test image, even though glasses were added. [8]

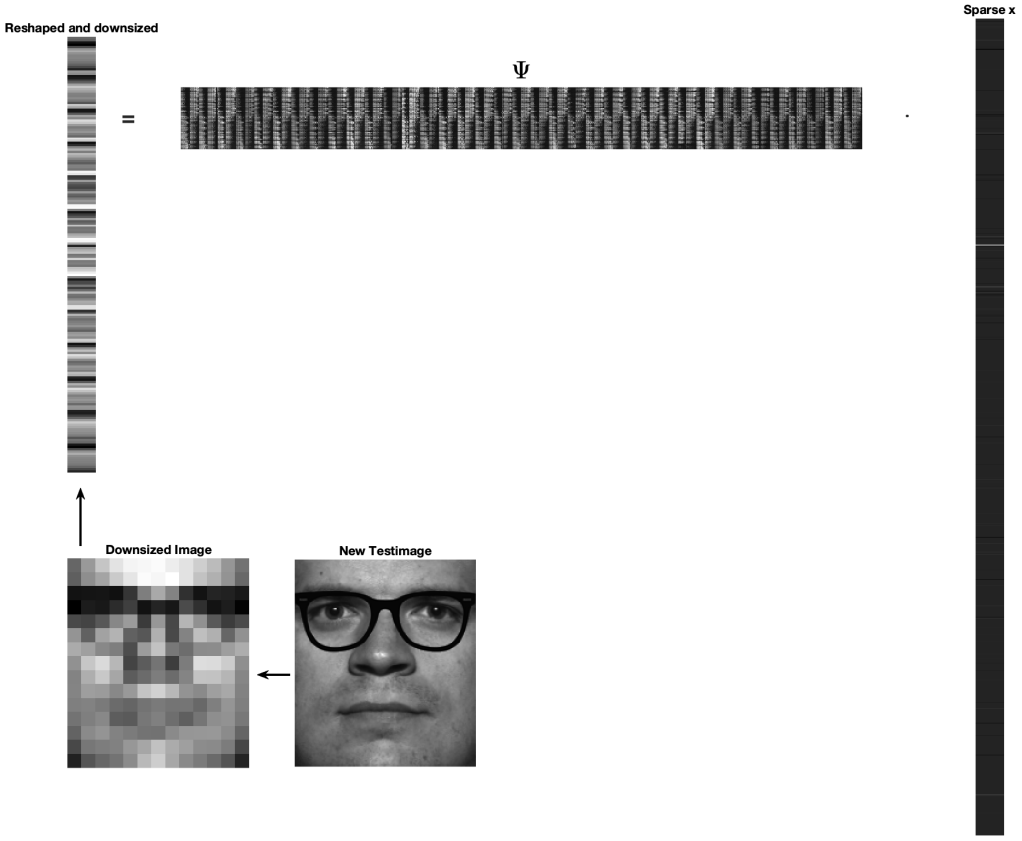

Figure 6.8: Sparse representation classification algorithm

In the next figure 6.9 there are shown the least square solution, which was computed by taking the pseudo inverse and the L_1 optimised solution. The least square doesn't lead to a sparse solution as all trainings faces will contribute.
From the coefficients in x the cropped faces can be reconstructed by multiplying the originally scaled training pictures with x. Then also the error of the reconstruction can be determined, by subtracting the actual test face from the reconstruction.

In the following example (figure 6.9) we are able to directly see the difference between the leats square (pseudo inverse) solution and the L_1 optimisation. The sparsity of the optimised x can clearly be identified in the bar plot. With this increased sparsity, also the reconstructed image has a lot clearer features, such that also visually the right person can be identified better on the right side.

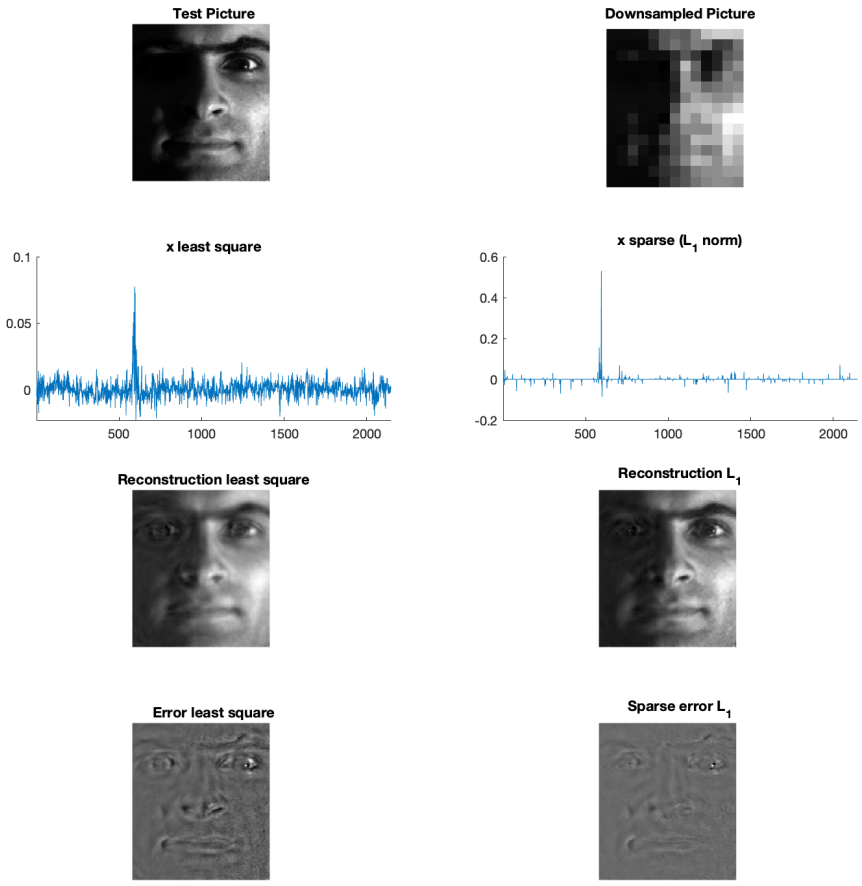

Figure 6.9: Sparse representation classification comparison least square and L_1 optimisation

Yet there hasn't been any policy how to choose the identified person from the determined x. This may seem trivial with the reconstructed images, however in order to automate such process there has to be developed an idea for this. Some may say, that it is sufficient to just take the person with the largest coefficient in the sparse x. However with bad lighting condition this may lead to some false classifications. An alternative way is, to sum up the absolute values of the coefficients corresponding to each person and find the maximum of these. If two peaks exist, but there is still more low representation in the images of the correct person, this will also be taken into account by this second strategy. This is shown in figure 6.10.

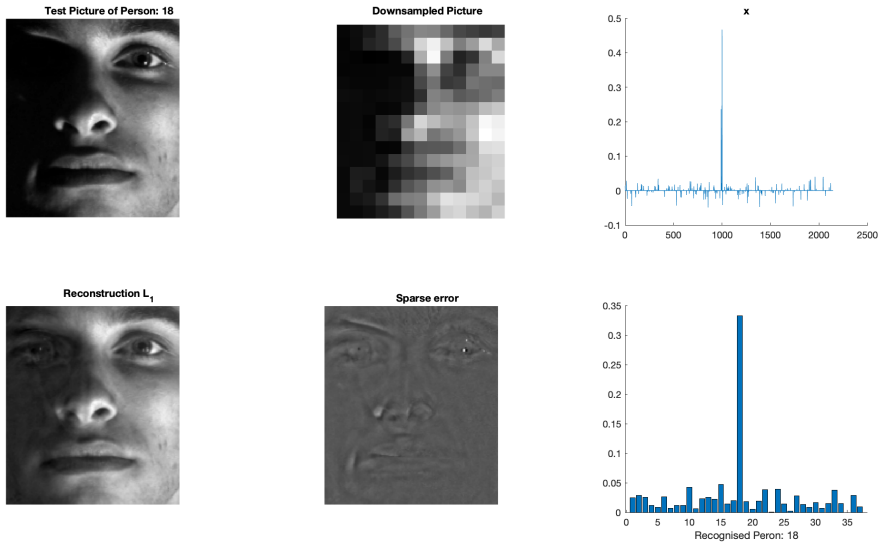

Figure 6.10: Identification with summed coefficients of sparse x vector

Robustness of sparse solutions

As in sparse solutions of x only few terms contribute to the reconstruction, it is possible to identify faces from highly corrupted data. This means, that the algorithm still identifies the person even though his outer appearance changed (e.g. a beard or glasses). In the following example fake glasses were added to one of the test pictures. The same identification process as described above is run on this highly corrupted data. The results can be seen in figure 6.11

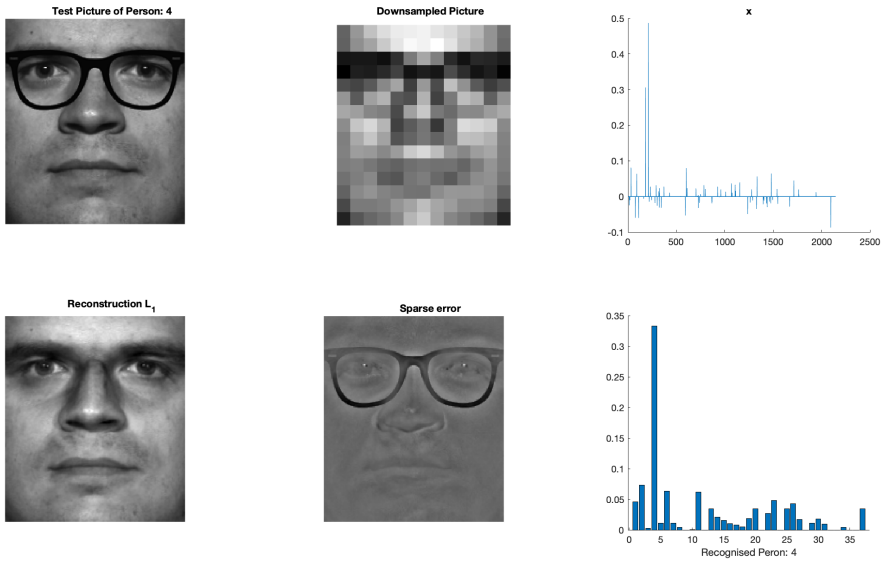

Figure 6.11: Identification of highly corrupted image

Iterative solution
In this paragraph there is a simple self developed iterative solution for such a minimisation problem. This doesn't lead to the optimal solution, however performs unexpectedly well. Therefore the minimisation can be modified such that:

$$x_{\text{sparse}} = \min_x ||x||_1 \text{ s.t } ||\Psi x - b||_2 < \alpha \quad \alpha \in \mathbb{R}^+$$
$$\Rightarrow x_{\text{'sparse'}} = \text{find } x \text{ s.t. } ||\Psi x - b||_2 < \alpha \wedge ||x||_1 < \beta \quad \alpha, \beta \in \mathbb{R}^+$$

Therefore each iteration the two least contributing factors are set to zero and the pseudo inverse is recalculated with the left faces (coefficients greater than 0). This reduces the number of coefficients and leads to some kind of 'sparse' solution x. This however leads to large amplitudes with different signs for closely related faces, which however don't represent the person. The results of this simplified version is shown in figure 6.12. The same corrupted image is taken for comparison.

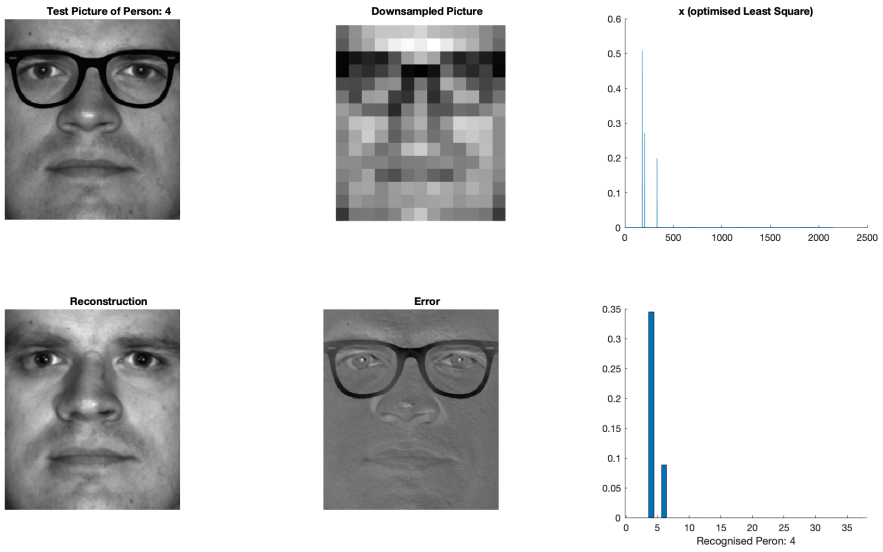

Figure 6.12: Iterative solution for simplified minimisation problem

6.2.3 Conclusion

The sparse representation classification is a reliable algorithm if a large enough library of the subjects is available. In comparison to the PCA classification, which had a classification reliability of about 70% the spars representation performs better with 95% reliability. This example with the Yale Face Dataset offered a large library. Probably in a case of very few images per person the PCA algorithm would take advantage.

7 Physical Models and ROMs

7.1 ROMs Proper Orthogonal Decomposition

7.1.1 Theory

ROMs are so called **R**educed **O**rder **M**odels. These are simplifications of high dimensional dynamical (physical) systems. There can be extracted dominant effects and behaviours, such that the complex system can be projected into a low order model.
In this section there will especially be tackled the **POD P**roper **O**rthogonal **D**ecomposition, which is based on the SVD (explained in 2.5). There will be tackled systems of the following form.

$$f(x,t)$$

These are for example fluid flow models. The data which is used in this section, are measurements m dependent on location x and time t. Such that these can be arranged in a matrix.

$$M = \begin{pmatrix} m(x_1,t_1) & m(x_1,t_2) & \cdots & m(x_1,t_n) \\ m(x_2,t_1) & m(x_2,t_2) & \cdots & m(x_2,t_n) \\ \vdots & \vdots & \cdots & \vdots \\ m(x_m,t_1) & m(x_m,t_2) & \cdots & m(x_m,t_n) \end{pmatrix}$$

This is a structuring with coherent structure of space x and time t. Two dimensional subjects can be reordered and reshaped, as long as it is persistent for all time modes.
The main idea of the Proper Orthogonal Decomposition is to decompose the complex function related to both space and time into deterministic spatial functions $\Phi(x)$ and time coefficients $c(t)$. Such that the problem can be written as.

$$f(x,t) = \sum_{k=1}^{n} c_k(t) \cdot \Phi_k(x)$$

Therefore the SVD can be used as it performs exactly this decomposition in the following way. Therefore the measurement matrix M of the described structure is decomposed. [7]

$$M(x,t) = \begin{pmatrix} \\ \Phi(x) \\ \\ \end{pmatrix} \cdot \begin{pmatrix} \\ \Sigma \\ \\ \end{pmatrix} \cdot \begin{pmatrix} \\ c(t) \\ \\ \end{pmatrix}$$

By taking the economy SVD there is again lost no information and it will get computationally faster. This is a very useful decomposition, as there exist now instead of one term being dependent on both space and time, two terms only being dependent on one of both. Additionally the correlation between the is linear. The non linearity of the system is represented now in both terms only.

7.1.2 POD on Fluid Flow Past a Cylinder

In this section a benchmark problem in fluid dynamics is tackled. It is the flow past a cylinder. In this case the vorticity is the measurement of interest depending on time and space.

$$v(x,t)$$

The measurements are visualised in a grayscale image in the following figure 7.2. The data matrix

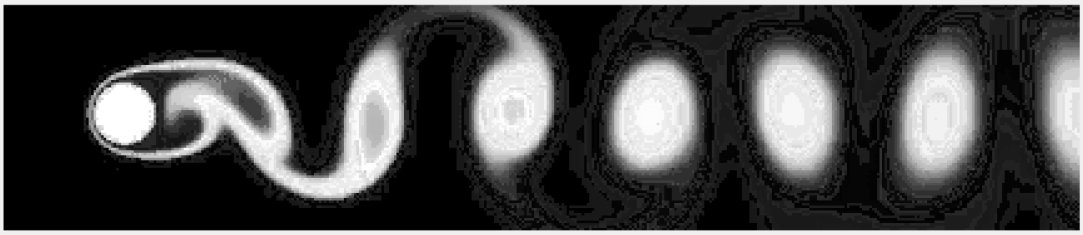

Figure 7.1: Vorticity of flow past a cylinder

with the measurements is constructed as described in the introductory to ROMs. Further the economy SVD is taken, such that the different spatial POD modes $\Phi(x)$ are stored in the columns of U and the corresponding time coefficients $c(t)$ in the rows of V. For a better understanding of the correlations coming up next, there are first plotted the singular values and the variance captured by k singular values.

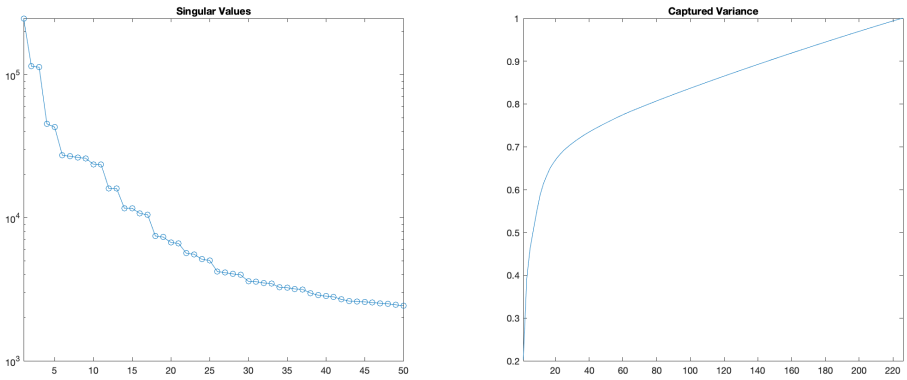

Figure 7.2: Singular values and captured variance

There can be identified one dominating singular value, followed by value pairs. These pairs are mode pairs that are closely related.
Next the different mode pairs (spatial POD modes) are visualised in figure 7.3. The first one is the mean flow corresponding to the most dominant singular value, followed by the mode pairs. These spatial mode pairs are closely related in structure.

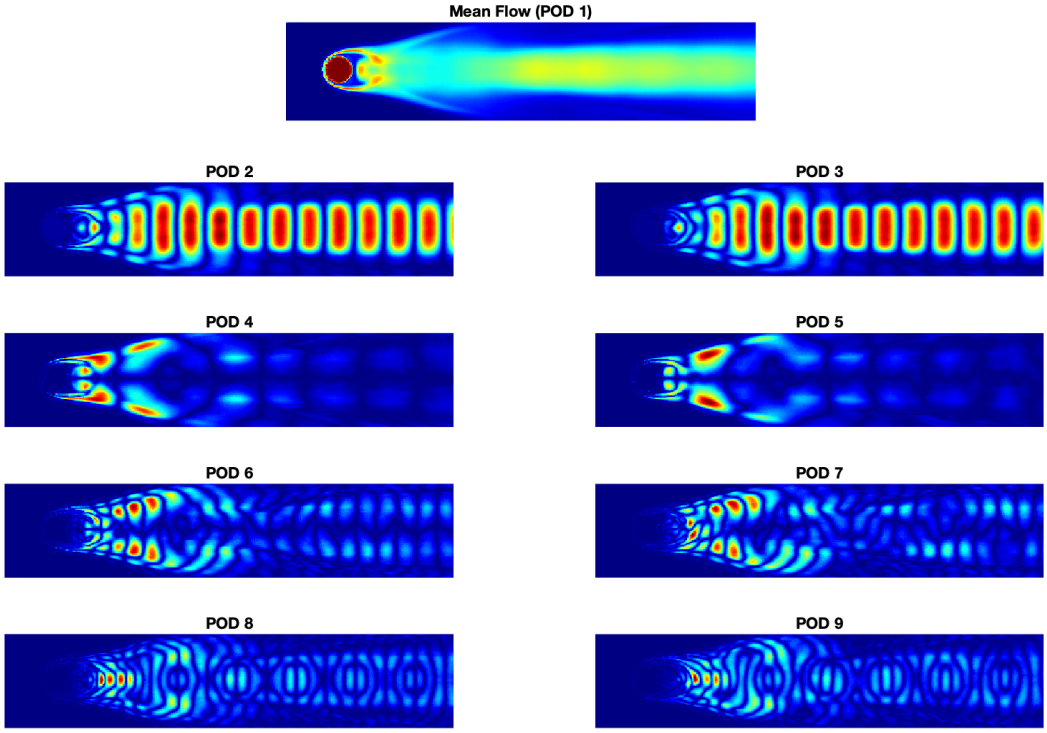

Figure 7.3: POD Modes

These spatial modes contribute to the vorticity flow weighted by factors dependent on time. Such that the vorticity flow can be written as a product of the spatial modes Φ and the time coefficients c.

$$v(x,t) = \Phi(x) \cdot c(t)$$

The time coefficients are stored in the V matrix. In figure 7.4 the first nine modes are visualised, with the corresponding coefficients.
There can be observed that the coefficients of each mode are a periodic. This is explained with the repeating vorticity structure. There also exists a correlation structure in the coefficients considering the mode pairs. These are nearly identical oscillations, which are phase shifted. The face shift can be identified as $\frac{\pi}{2}$.
These periodic oscillation are highly non linear. With the POD it was however possible to isolate them and have a linear correlation between these non linear terms.
These oscillation terms allow us to predict future vorticity flow states. Also based on this periodicity of the example, there can be used compressed sensing in order to be able to reduce the number of measurements, that are necessary to take for construction of the flow field. This will be explained in detail in the next section 7.1.3.

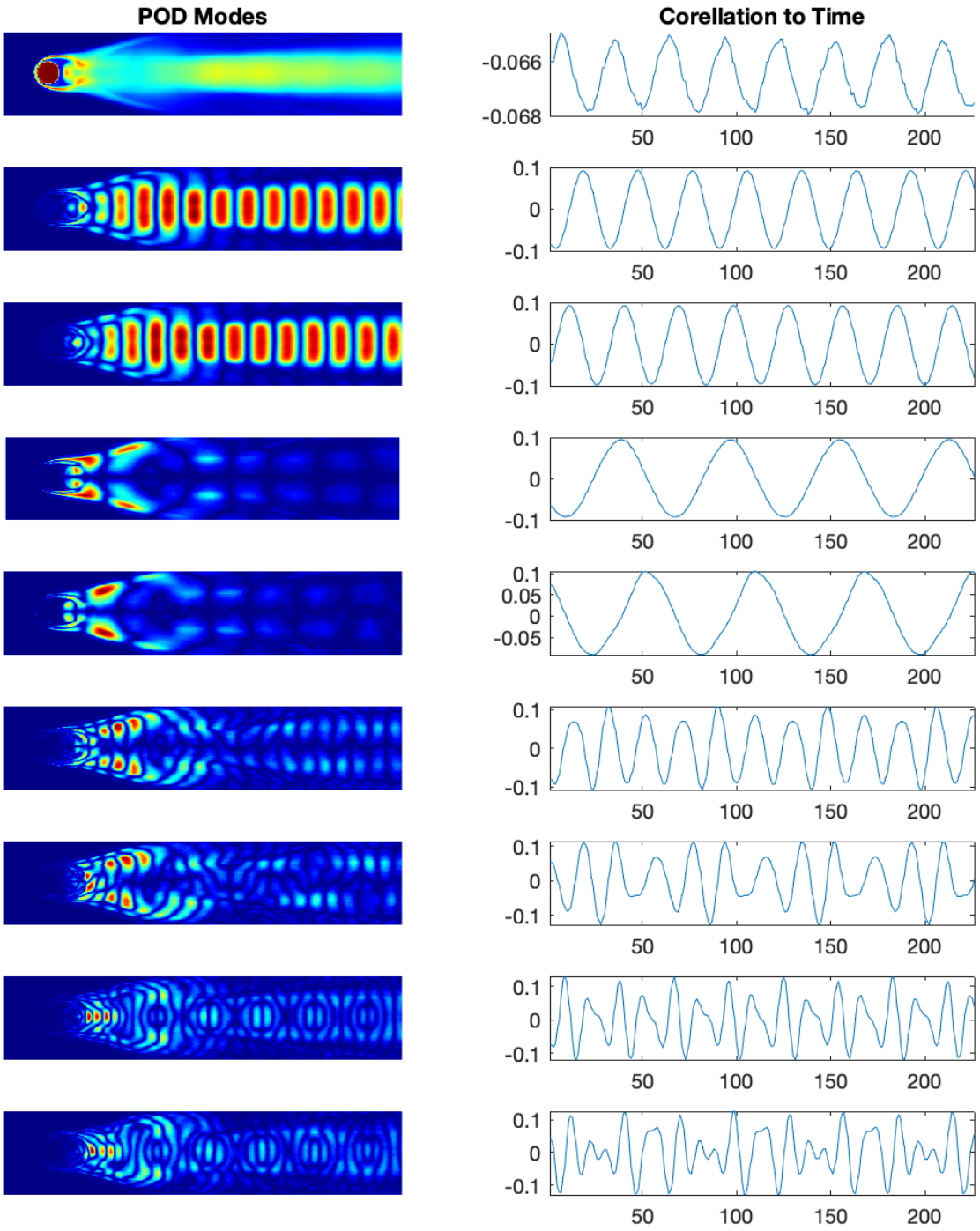

Figure 7.4: POD Modes with coefficients

7.1.3 POD & Compressed Sensing Flow Past a Cylinder

In the previous section the data matrix with the vorticity measurements dependent on time and space was huge. In a realistic environment, as in most experiments, there can't be taken such a huge amount of measurements. Therefore this section will try to give a simplified example for a concrete usage of compressed sensing, explained in **??**, in combination with the POD. Therefore the same data as in the previous section is used, with the main difference, that there were taken fewer measurements temporally. Also an incomplete spatial measurement can be reconstructed with compressed sensing. In this specific example, the measurements will be reduced from 226 to 40.
The measurement times have to be randomly sampled, to enable the usage of compressed sensing. However with increasing randomness of the chosen times, the accuracy of the POD modes decreases. For this tradeoff problem, there was the developed following solution in this example. The measurement times are evenly spaced over the whole observation time, with a slight random offset.
Additionally there has to be taken into account, that this is a time evolving system, consequently the measurements again have to be stored in the right time order in the reduced data matrix X.
Further there is taken the economy SVD of this reduced data matrix. The first modes of the reduced system are shown in the following figure 7.5.

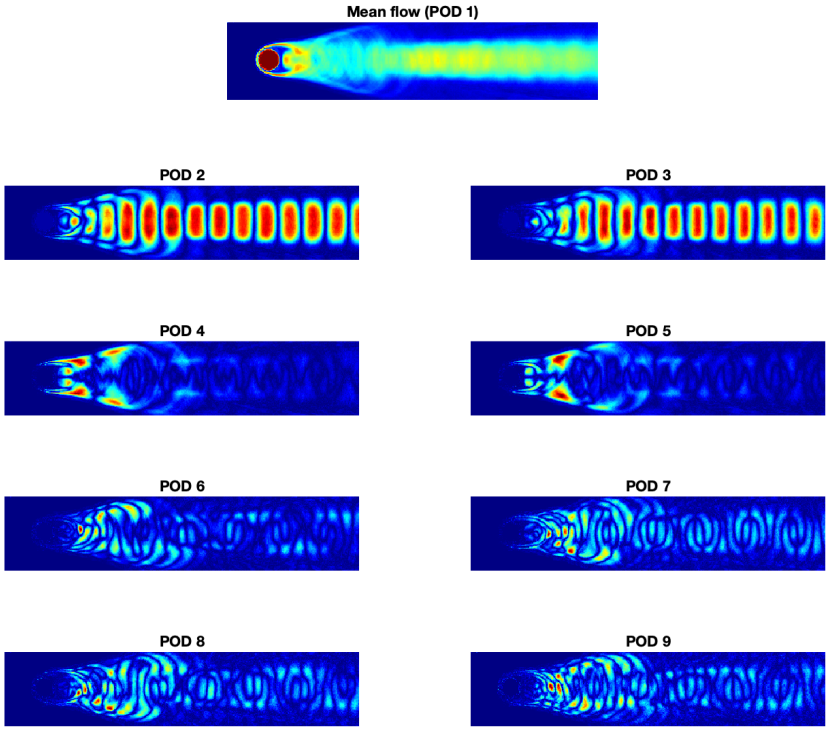

Figure 7.5: POD Modes of the reduced data

The next step is to analyse the coefficients dependent of the time, as there were only taken 40 measurements, this will also only lead to 40 time coefficients. The coefficients to the corresponding modes are given in figure 7.6. These however don't show any good correlations yet.

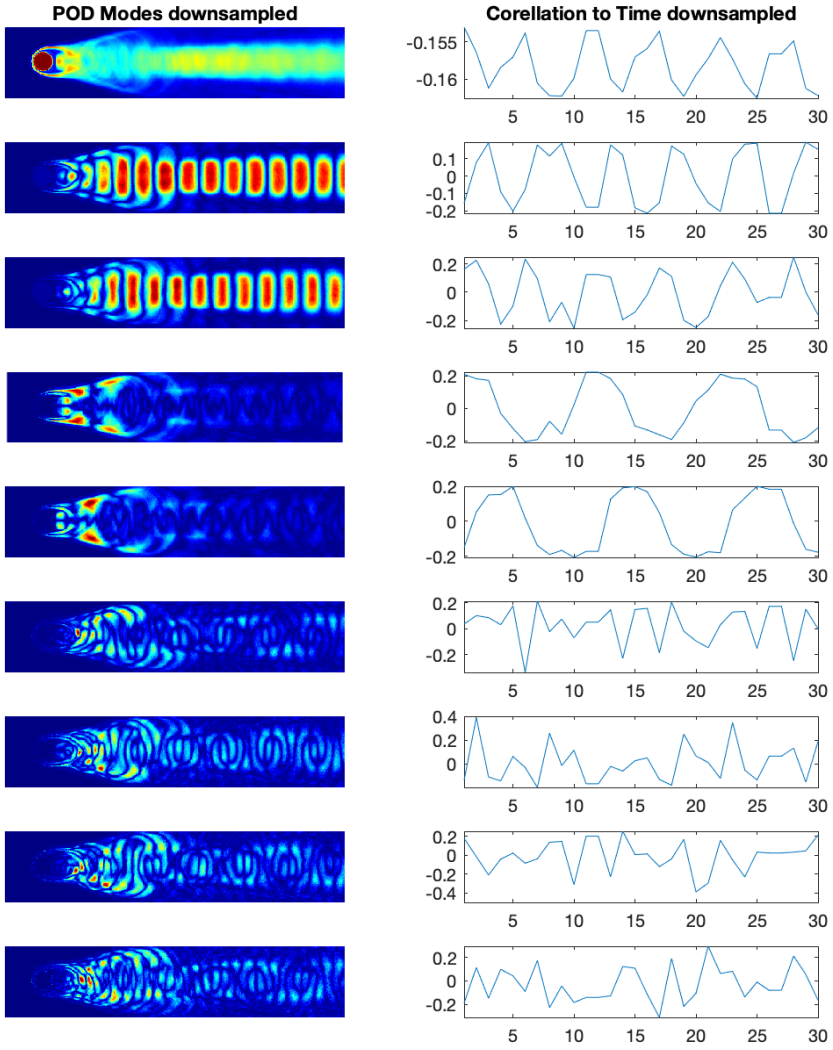

Figure 7.6: POD Modes and correlating coefficients

In order to recreate the full number of samples, there will be used compressed sensing. The domain is chosen to be the Fourier space, as periodic functions will have an extremely sparse representation. Therefore the orthogonal basis $\Phi = \frac{1}{226} F_{226}$ is the Fourier matrix of size 226.
The following plots in figure 7.7 show the fully reconstructed sample length of the coefficients.

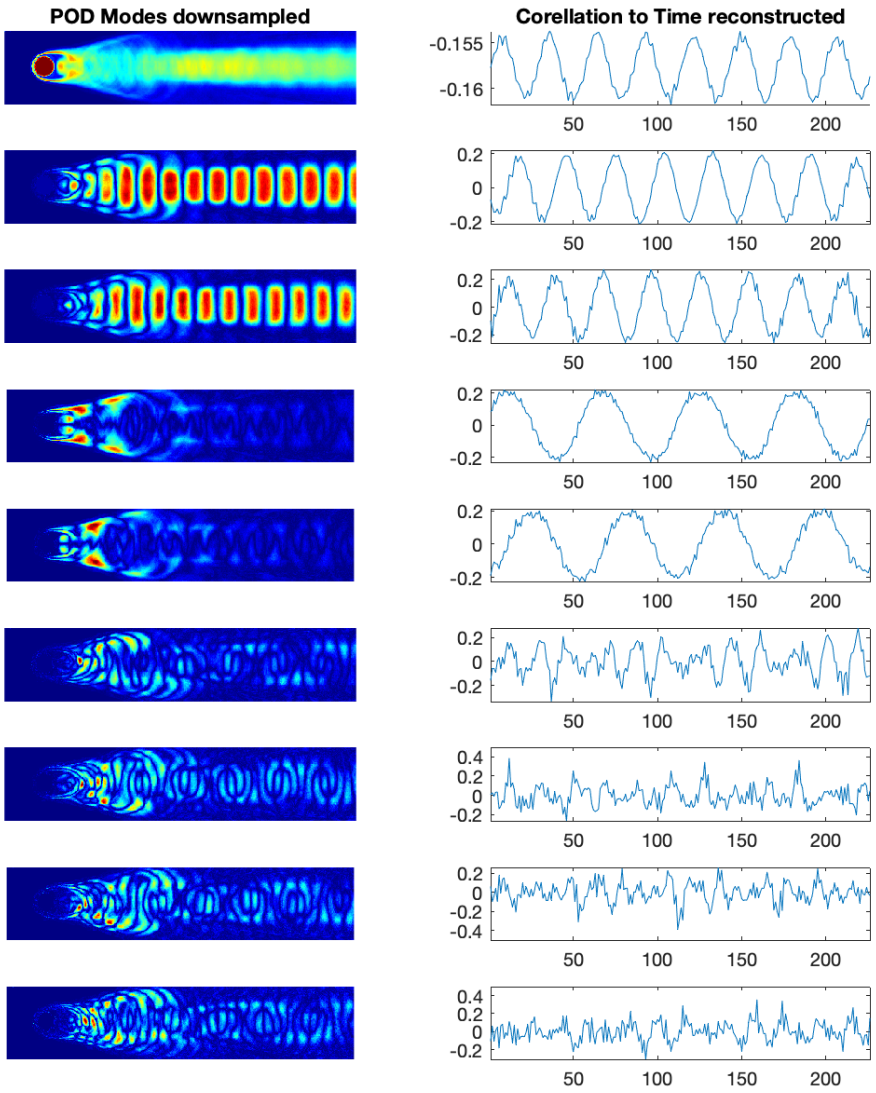

Figure 7.7: Reconstructed coefficients over whole sample time

These periodic signals however are still very spiky sucht that it will create a very noisy reconstruction. For visualisation the first frame of the reconstructed flow with a strong ringing effect was chosen. It is shown in figure 7.8. This means, that some filtering is required in order to denoise the coefficients. This is done in this case by transforming the signal once again into the Fourier domain and elimination the high frequent noise by eliminating all eytremely high frequencies and frequencies below a certain threshold. This leads to the filtered and denoised coefficients shown in figure 7.9.

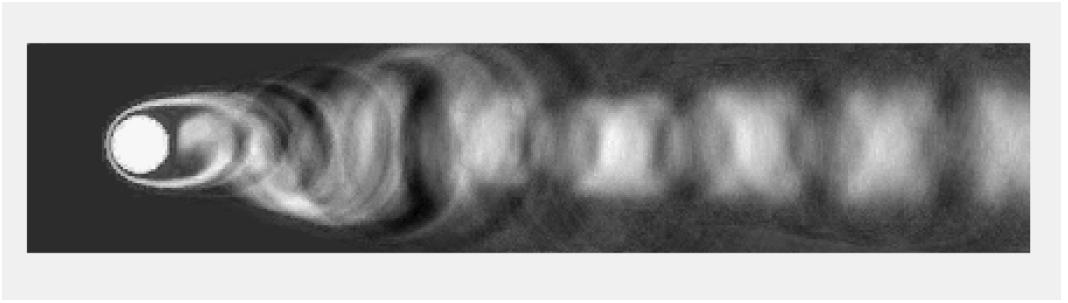

Figure 7.8: Reconstructed Flow with ringing effect

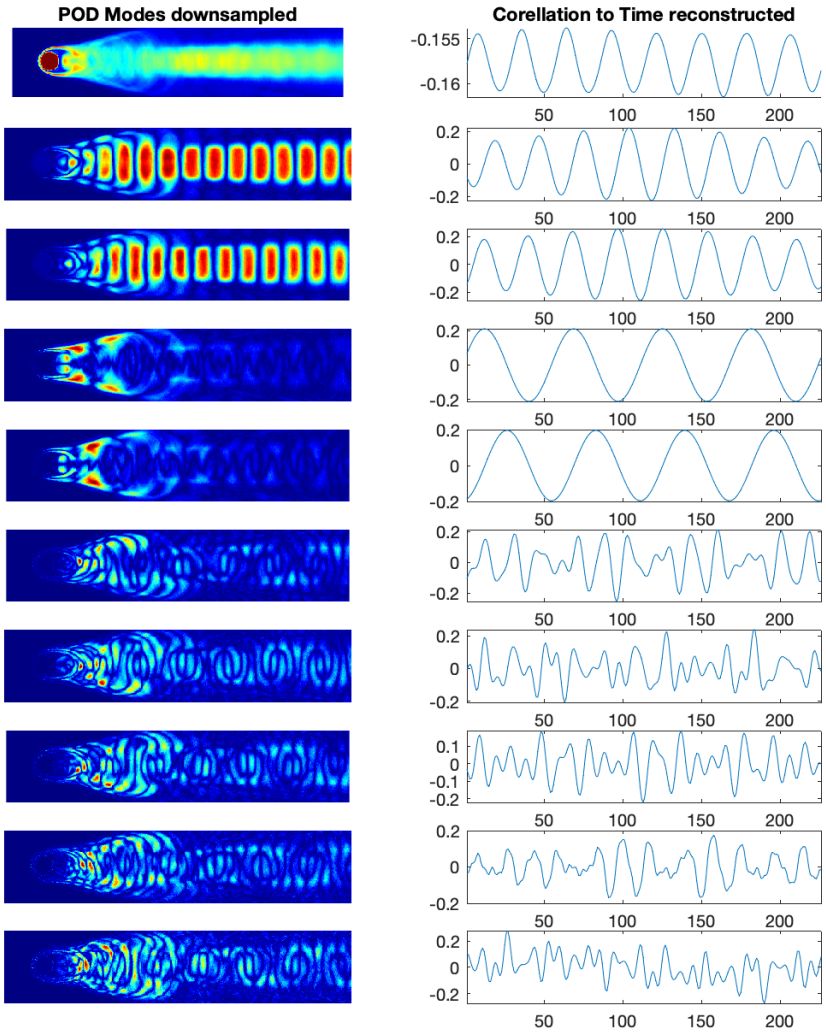

Figure 7.9: Denoised coefficients to corresponding POD modes

There can be seen a dramatic improvement of the denoising. This will hopefully work out and reduce the am´mount of ringing in the reconstructed vorticity flow. In figure 7.10 the same frame as in figure 7.8 was chosen in order to compare both. This definitely shows a weaker ringing effect and a clearer structure of the original flow field. For comparison the original flow field was depicted in 7.2.

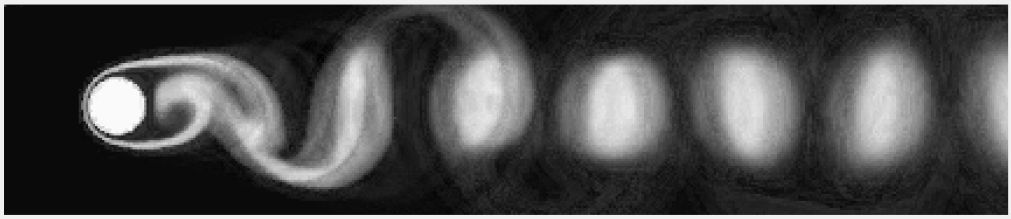

Figure 7.10: Reconstruction with cleaned coefficients

7.1.4 POD on noisy data

Gaussian White Noise
In previous sections the proper orthogonal decomposition was performed on a full dimensional flow field and downsampled measurements. In this section there will be shown, how POD models perform on noisy data. Therefore again the flow field past the cylinder is taking as the objective. In figure 7.11, the first timeframe of the noisy measurements is shown.

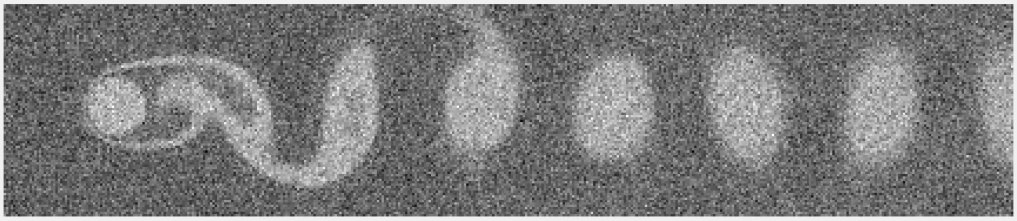

Figure 7.11: First timeframe of noisy measurements

Even though this data contains very much gaussian white noise, there is a low rank structure beneath. The task is to find and extract this low rank structure. This can be done in different ways.
As the SVD already provides a low rank approximation by extracting the first few singular values, the SVD itself may already reveal this low rank structure.

$$X = U \cdot \Sigma \cdot V^T$$

For verification of this assumption the following plot shows the singular values and the captured variance. There can be seen, that the singular values decrease rapidly until approximately rank 16. In these first modes, the low rank structure is captured. The variance which is captured is slowly increasing in the higher ranks, as the random noisy is high dimensional. Figure 7.12 contains the singular values and variance. Further the first 9 POD modes are shown in figure 7.13.

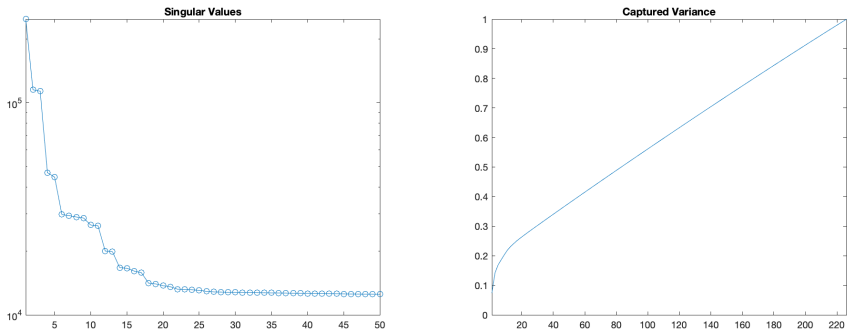

Figure 7.12: Singular values and captured variance of POD modes

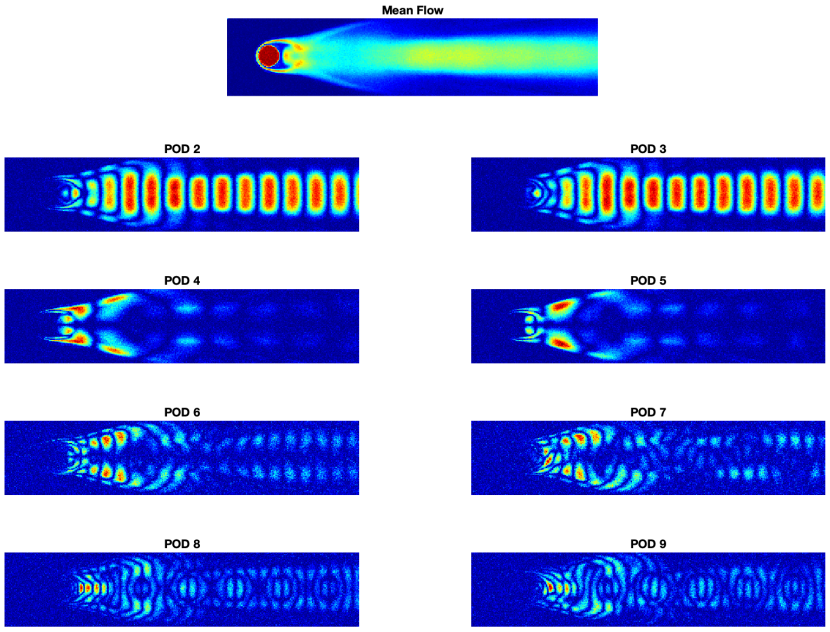

Figure 7.13: POD modes recovered from noisy measurement data

Even though the POD modes seem clean on the first glance, there can still be seen white noise in the low rank subspace.
The next figure 7.14 shows the effect of the noise on the time coefficients. The greatest consequence can be observed on the time coefficients of the mean flow.
The disturbance in the time data is limited, as the noise is constant and with equal intensity over the whole time interval. In case the noise contamination would differ, there was also a greater impact on the time coefficients.

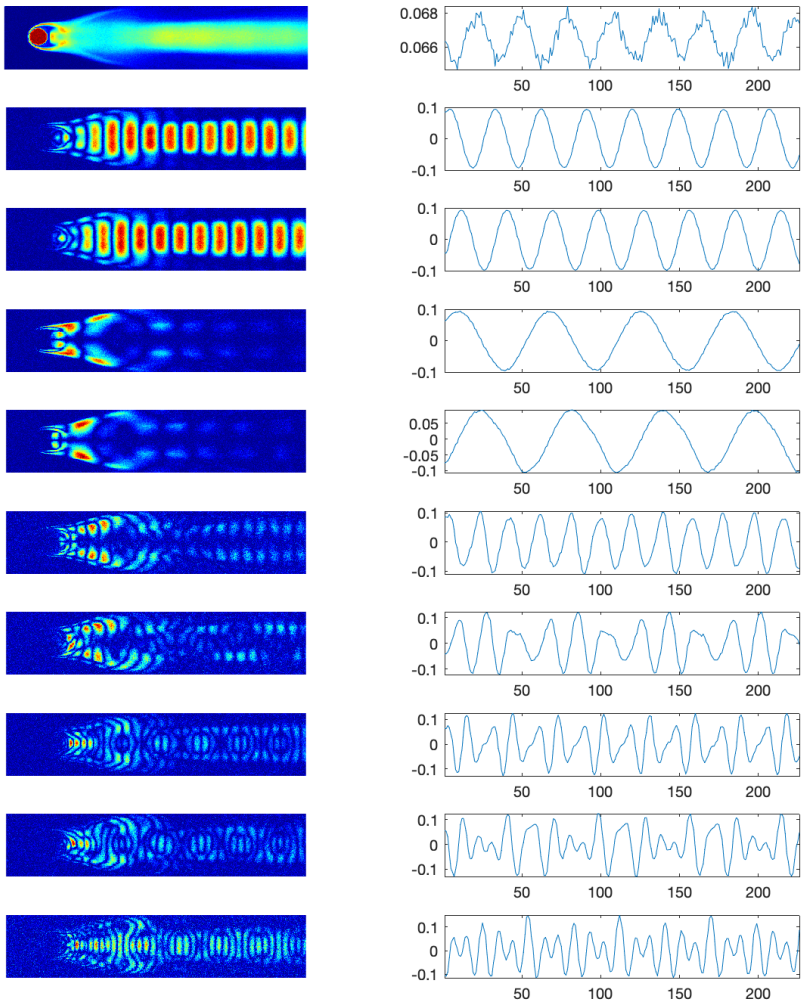

Figure 7.14: POD modes and time coefficients recovered from noisy measurement data

The reconstruction of the fluid flow (figure 7.15) from the low rank structure in the POD subspace is done as described. In this example, the first 20 POD modes were chosen for reconstruction.

$$\tilde{X} = U_r \cdot \Sigma_r \cdot V_r^T$$

$$\tilde{X} = \begin{pmatrix} u_{1,1} & \cdots & u_{1,20} \\ \vdots & \vdots & \vdots \\ u_{n,1} & \cdots & u_{n,20} \end{pmatrix} \cdot diag(\sigma_1, \cdots, \sigma_{20}) \cdot \begin{pmatrix} v_{1,1} & \cdots & v_{1,20} \\ \vdots & \vdots & \vdots \\ v_{n,1} & \cdots & v_{n,20} \end{pmatrix}^T$$

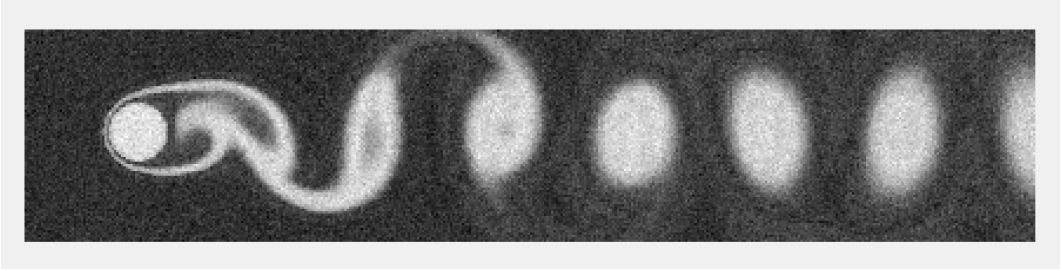

Figure 7.15: Reconstructed fluid flow

This first timeframe of the reconstructed flow shows a dramatic improvement in terms of noise contamination.

Salt and Pepper Noise

In the example shown above the data was contaminated with gaussian white noise. The low rank structure was easily extracted by applying the singular value decomposition.
In the following case one third of the data-points is corrupted with salt and pepper noise. The following figure 7.16 shows the corrupt measurements of the flow field.

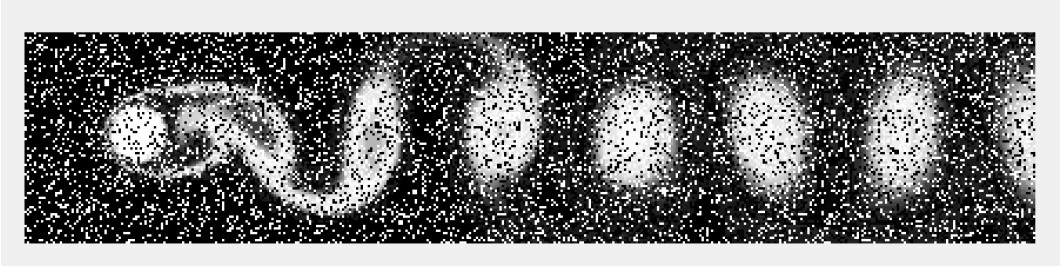

Figure 7.16: Cylinder flow with salt and pepper noise

As in the previous example again first the singular value decomposition is applied. This should ideally recover the low rank structure behind the noise. The singular values and the captured variances are plotted in the the following figure 7.17.

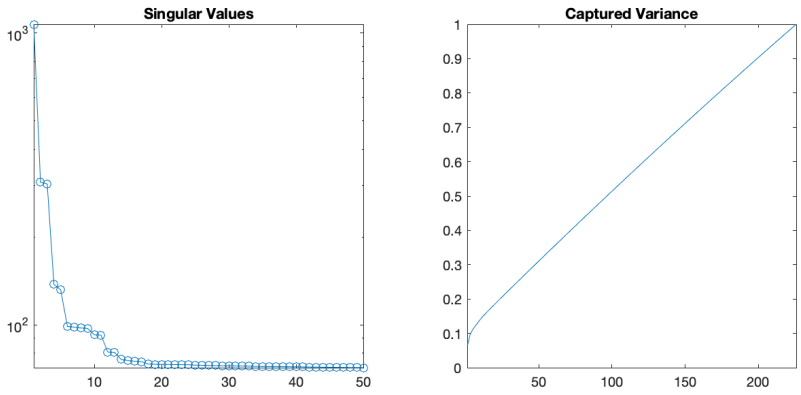

Figure 7.17: Singular values and captured variance of salt and pepper contaminated fluid flow

The singular values decay fast in the beginning. After the twentieth value they all stay on a relatively high level, such that also the captured variance only increases slowly. The first nine corresponding POD modes are shown in figure 7.18

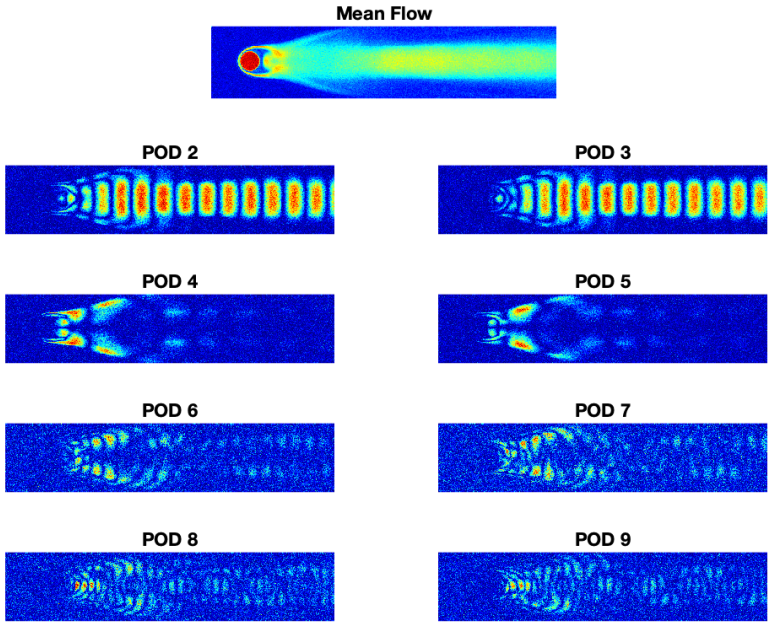

Figure 7.18: First nine POD modes of salt and pepper contaminated flow data

Compared to the POD modes of the clean data shown in figure 7.3, there can clearly be seen a loss in quality. The time correlation factors (in U) aren't dramatically affected, as the noise again is constant over time, such that the noisy is only found in the spatial content (in V).
From the first low rank structured POD modes the fluid flow can be reconstructed with less noise. The first time frame of the reconstruction has got some noise reduction (figure 7.19). The reconstruction in this case isn't as good as in the previous example, as the noise has more correlation and therefore is contained in the principal modes.

Figure 7.19: Reconstruction from first 20 POD modes

Therefore another method has to be introduced in-order to gain better results.

This is the so called robust principal component analysis. [8] The reason for the corrupt reconstruction is, that the SVD method is also based on least square, which means that heavy outliers, as salt and pepper noise also cause bad performance of the PCA. Therefore the only solution is, to denoise the data before taking the svd. In order to denoise the data, the noisy measurements M have to be split into a low rank structure dominated matrix L and a sparse noisy matrix S.

$$M = L + S$$

This is a highly non convex optimisation problem, which has to be formulated in a simplified way in order to be solved.

$$\min_{L,S} ||L||_* + ||S||_1 \quad subject\,to\ L + S = M$$

In this case $||L||_*$ is the nuclear norm, which is defined as the sum of the singular values of the matrix $\sum_i \sigma_i$.
In this case the problem can be identified similar to a low rank matrix completion problem (e.g. Netflix Problem). [9] In minimisation problems which include the nuclear norm, it is necessary to define a soft-thresholding operator $D_\tau(M)$. The singular value decomposition of the matrix M is defined as:

$$M = U\Sigma V$$

According to this definition, $D_\tau(M)$ is defined as:

$$D_\tau(M) = Udiag(\{\sigma_i - \tau\}_+)V$$

In the Code this is split into the two functions *soft_thresh(... shrink( )...)*.
After initialising the operator, a so called Singular Value Threshold Algorithm SVT can be implemented. This is an iterative algorithm, which repeats the following step until some stopping criterion is reached.

$$Y_0 = 0$$

$$\begin{cases} L_k = D_\tau(Y_{k-1}) \\ S_k = D_\tau(X - L_{k-1}) \\ Y_k = Y_{k-1} + \mu(M - L_k - S_k) \end{cases}$$

The coefficients μ and τ can be seen as tuning parameters, which can be changed arbitrary with an effect on performance. In the following figure 7.20, the denoised low rank data and the sparse noise can be seen.

In the next step it is now possible to perform the SVD on the low rank structured flow, in order to get the denoised POD modes. The singular values of the low rank data and the variance captured by these singular values is shown in figure 7.21.
A drastic improvement can be seen. The singular values decrease continuously and the variance is increasing in logarithmic structure.

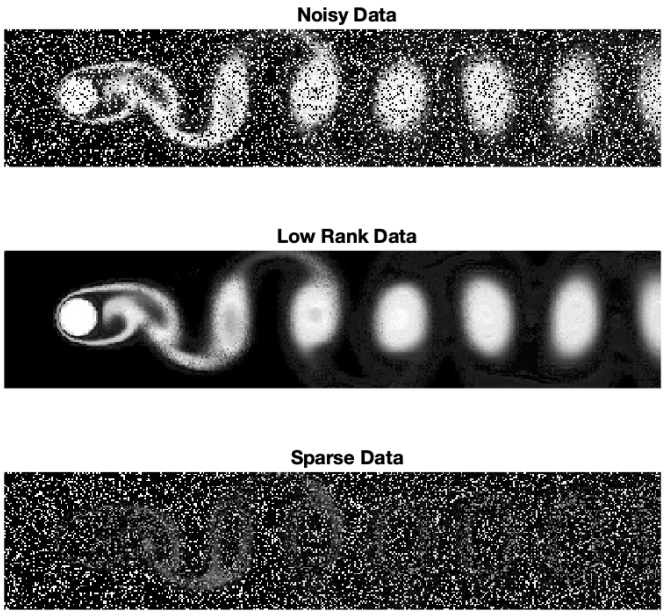

Figure 7.20: Low rank structure and sparse noise

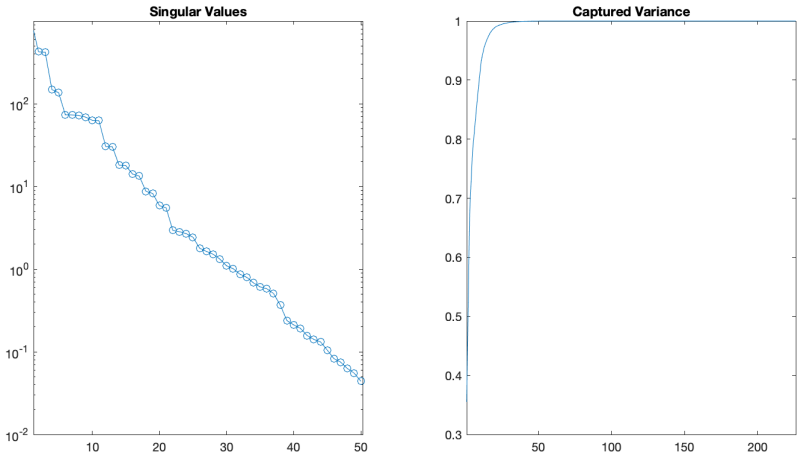

Figure 7.21: Singular values and variance of low rank reconstruction of the noisy data

Further the POD modes are analysed. The first nine are shown in figure 7.22. These seem a lot cleaner in structure. Also it is now possible to use more POD modes for reconstruction, as the noise already was canceled out before, such that a higher accuracy can be achieved.

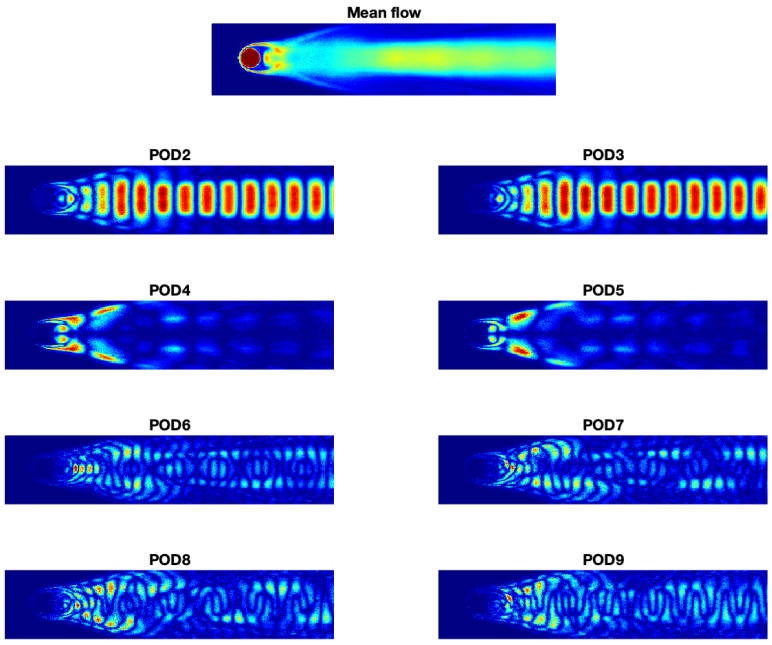

Figure 7.22: POD modes of low rank reconstruction of the noisy data

With these clean POD modes, it is possible to reconstruct the fluid flow with a denoised reduced order model. The following figure 7.23 shows the first timeframe of the reconstructed flow. For. the reconstruction there were used the first 20 POD modes.

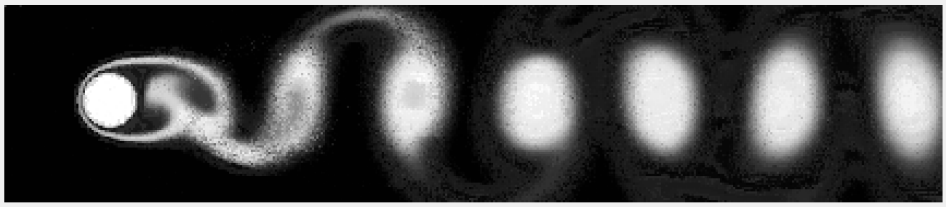

Figure 7.23: Reconstructed fluid flow with first 20 POD modes of low rank data

In comparison to the first reconstruction from figure 7.19 did the second outperform it by far. This does especially work with salt and pepper noise, as the problem can be written similar to a low rank matrix completion problem.

7.1.5 Conclusion

PODs and ROMs enable us to model high dimensional problems in a low rank subspace, which captures most of the variance and details. Recurrent flow fields for example can be perfectly captured in a low rank structure with recurrently dominant POD modes. These POD models in case of low rank structures can also be gained with missing data, as the full resolution can be reconstructed with compressed sensing. In terms of noisy data, the POD already filters a great amount of noise. This especially works with low rank systems. In order to enhance this noise reduction in the POD models, the measurements can be denoised before by applying the SVT (singular value thresholding algorithm).

7.2 System Modeling

7.2.1 Theory

Real world systems may often be complicated and therefore need to be simplified before translating into a mathematical model. In this section we will have a look at the modelling of a pandemic. Therefore several different models have been developed. The simplest model is a *SIR* model. In this basic model the population is subdivided into three groups. In more advanced models there exist even more categories. The groups behaviour is described with ODEs, such that we otain a system of ODEs for the pandemic behaviour of the whole population.

$$\begin{aligned} y_1' &= f_1(y_1, y_2, \ldots, y_n, t) \\ y_2' &= f_2(y_1, y_2, \ldots, y_n, t) \\ &\vdots \\ y_n' &= f_n(y_1, y_2, \ldots, y_n, t) \end{aligned}$$

This may in some linear case be solved analytically, however a more practical solution are numerical simulations. In the following chapter the *Forward Euler* is considered with a reasonably small step with.

$$\vec{y}(t_{i+1}) = \vec{y}(t) + \triangle t \vec{f}(y_1, y_2, \ldots, y_n, t)$$

This is a relatively simple method to numerically simulate ODEs with small effort.

7.2.2 SIR Model for Pandemic Simulation

A SIR model divides the population in the following three groups:

- (S) Susceptible Population:
  Unprotected population, that can be infected with the virus.
- (I) Infected:
  The population which carries the various and can spread it.
- (R) Removed:
  This category includes all recovered people and also all casualties.

In the first simple model we make the drastic assumption, that a recovered person can never be infected again, such that immunity lasts for the whole pandemic after one infection. The model is depicted in the following diagram.

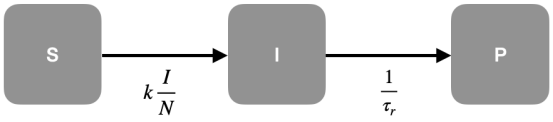

Figure 7.24: SID Model

In a simulation with an overall population of 1000 and an initial number of 20 infected we obtain the

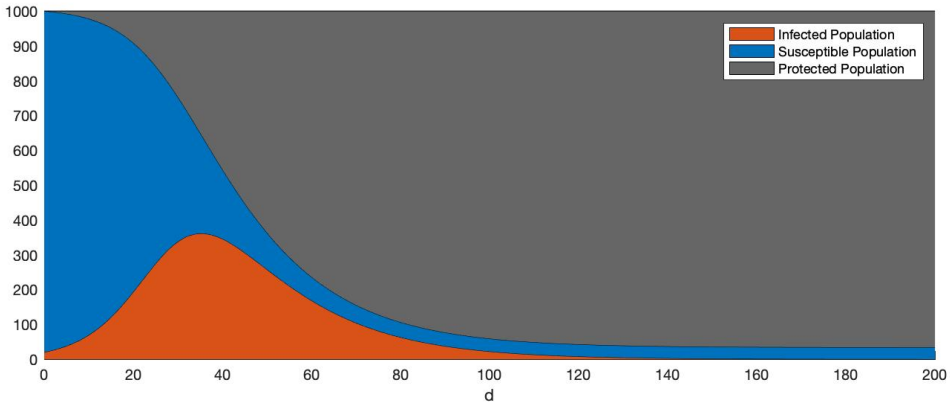

Figure 7.25: SID Model simulation

following model. There can be nicely observed the exponential ascent until a maximum is reached and the infections decay because of the high number of removed individuals.

For a more contagious virus the simulation has a steeper rise. But this simple model can't model multi-wave phenomena.

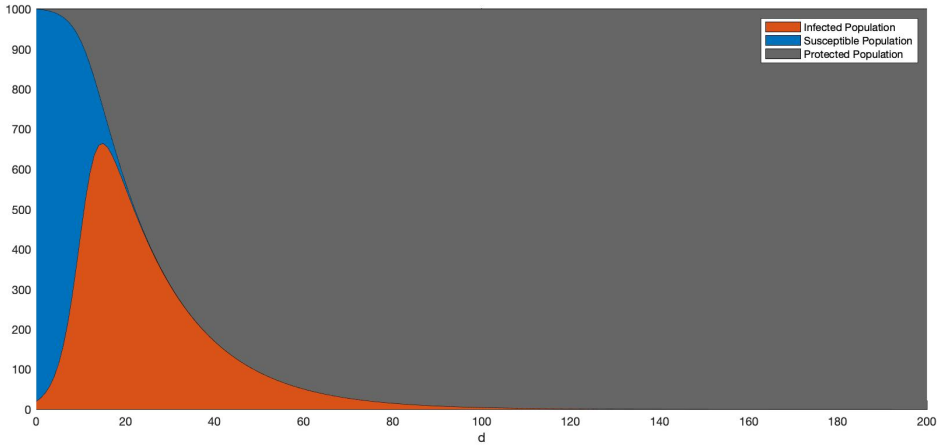

Figure 7.26: SID Model simulation

7.2.3 PSEIRD Model for Pandemic Simulation

A more sophisticated model is the PSEIRD model, which involves more groups and relations. In the following table all modelled groups and parameters are listed. [10] In this section the aim is to develop a model for the Pandemic in Portugal from the 26th February 2020 until May 2021. The mathematical model of the population can be described with a system of differential equations.

Table 1: Modelled Groups

Variable	Definition
P	Protected population
S	Susceptible population
E	Infected population in incubation period
I	Infected population responsible for spreading
I_d	Infected and identified population (positively tested)
R	Recovered and vaccinated population
D	Total casualties

Table 2: Modelling Parameters

Parameter	Definition
α	Protection factor $S \to P$
k_{PS}	Transition rate $P \to S$
k	Infection rate $S \to E$
τ_i	Incubation time $E \to I$
τ_{Id}	Transition time $I \to I_d$
d	Fraction of detected I
l	Mortality
τ_R	Recovery time
τ_D	Death time
k_{RP}	Reinfection Rate
v	Daily vaccinated people

$$\frac{dP}{dt} = k_{RP}R - k_{PS}P - v + \alpha(1 - \frac{I}{N})I_d S$$
$$\frac{dS}{dt} = k_{PS}P - k\frac{I}{N}S - \alpha(1 - \frac{I}{N})I_d S$$
$$\frac{dE}{dt} = k\frac{I}{N}S - \frac{1}{t_I}E$$
$$\frac{dI}{dt} = \frac{1}{t_I}E - (\frac{1-d}{t_R} + \frac{d}{t_{Id}})I$$
$$\frac{dI_d}{dt} = \frac{d}{t_{Id}}I - (\frac{1-l}{t_R} + \frac{l}{t_D})I_d$$
$$\frac{dR}{dt} = \frac{1}{t_R}((1-l)I_d + (1-d)I) + v - k_{RP}R$$
$$\frac{dD}{dt} = \frac{l}{t_D}I_d$$

In order to fit the model to real world data one uses the cumulative number of detected cases N_{TId}. It is given by the following equation.

$$\frac{dN_{TId}}{dt} = \frac{d}{t_{Id}}I$$

As well one can calculate the daily detected cases N_{Id}:

$$\frac{dN_{Id}}{dt} = \frac{d}{t_{Id}t_I}E - (\frac{1-d}{t_R} + \frac{d}{t_{Id}})N_{Id}$$

The complete system (except of auxiliary groups) is visualised in the following figure.

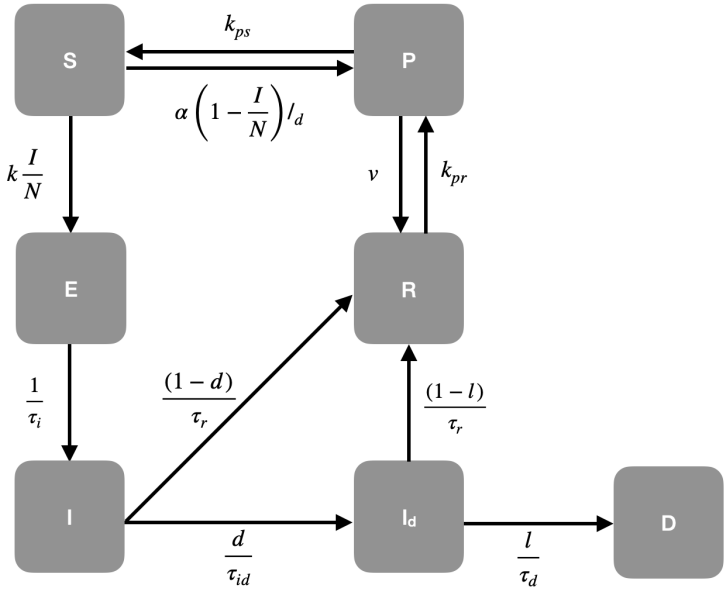

Figure 7.27: PSEIRD Model

In the following simulation initial conditions $S_0 = 10.2 \cdot 10^6, I_0 = 3, t_0 = 56$ are assumed. Additional the parameters are chosen to be:

Table 3: Simulation Parameters

Parameter	Value
α	$1.3 \cdot 10^{-3}$
k_{PS}	$\|_{91}2.7 \cdot 10^{-2}\|_{132}5.5 \cdot 10^{-2}\|_{244}8.3 \cdot 10^{-2}\|_{279}1.6\|_{359}27.8\|_{384}1.8 \cdot 10^{-2}$
k	$0.9\|_{247}0.3$
τ_i	3.5
τ_{Id}	6.25
d	0.25
l	0.02
τ_R	14
τ_D	$5.5\|_{133}16\|_{344}9.6\|_{389}50$
k_{RP}	$1 \cdot 10^{-2}$
v	0

Determining the parameters for the fitting data is a very complicated task and can be looked up in the original paper [10]. The simulation and official data is depicted in figure 7.28.
As we can see the model is a very good representation of the actual pandemic behaviour. The daily

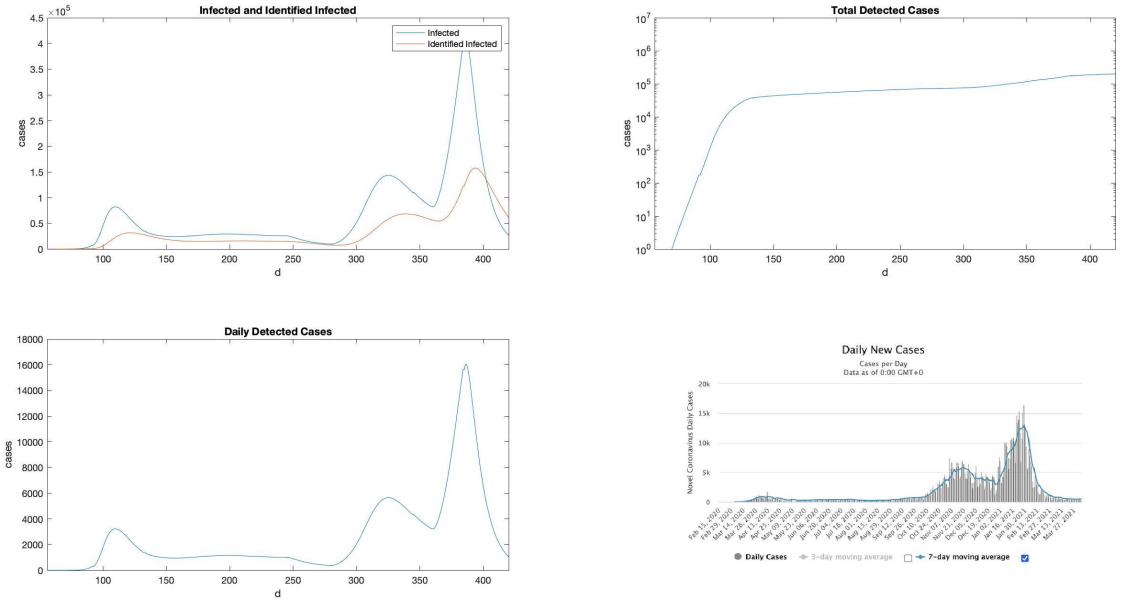

Figure 7.28: PSEIRD Model simulation (top left) infected and identified infected population (top right) total detected cases (bottom left) daily detected cases (bottom right) true data

detected cases almost correspond perfectly with the real world data. Such models can be used to predict and control pandemic behaviour.
In the following the model is used to predict the pandemic behaviour during the Easter holidays. Four different scenarios are therefor taken into account.

- Schools closed and easter holiday leakage lasting 25 days.
- Schools closed and easter holiday leakage lasting 50 days.
- Schools closed no easter leakage lasting 25 days.
- Schools closed no easter leakage lasting 50 days.

These four scenarios were simulated with similar contact behaviour as Christmas and give the following development 7.29.

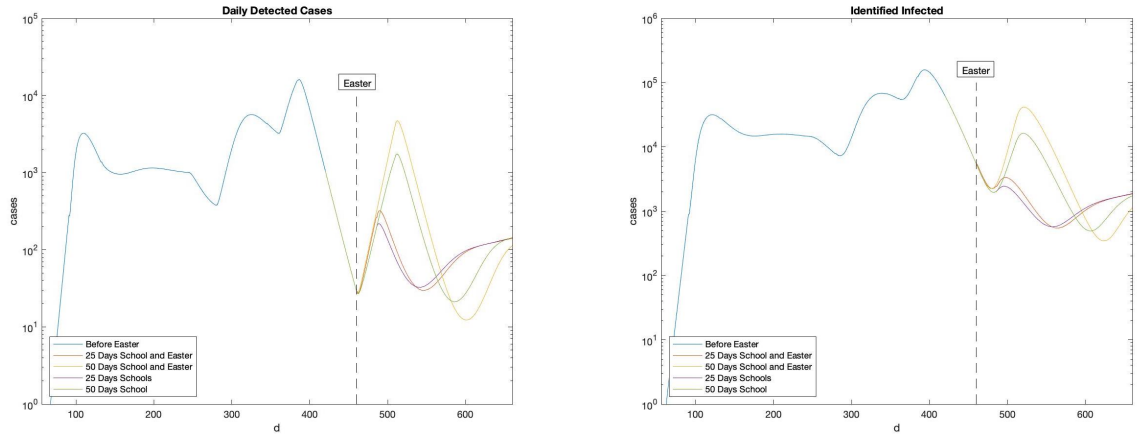

Figure 7.29: PSEIRD Model simulation for easter holidays (left) daily detected cases (right) detected infected cases

7.3 Sparse Identification of Nonlinear Dynamics

7.3.1 Theory

This section is about discovering dynamical system models from data. One computational automated discovery method is the SINDy algorithm developed Steve L. Brunton and J. Nathan Kutz in [8]. The algorithm uses that many dynamical systems of form:

$$\frac{d}{dt}y = f(y)$$

have a possible sparse right-hand function with few active terms from the set of all right-hand terms. Therefore we try to approximate f with a linear system of a large library Ψ of likely right-hand side terms and a coefficient vector ξ.

$$f(x) \approx \Psi(x)\xi$$

As $f(x)$ is most likely to have a sparse representation in Ψ, the coefficient vector ξ also has to be sparse. This means that ξ is desired to have as few active terms as possible. In order to find this sparse ξ we use similar to 6.2 sparse regression. In order to find the governing dynamical equations from our temporal data y, we first approximate the derivative $\dot{y}$ with finite difference schemes or Fourier.

$$y = \begin{pmatrix} | & | & | & | \\ y(t_1) & y(t_2) & \dots & y(t_n) \\ | & | & | & | \end{pmatrix} \quad \dot{y} = \begin{pmatrix} | & | & | & | \\ \dot{y}(t_1) & \dot{y}(t_2) & \dots & \dot{y}(t_n) \\ | & | & | & | \end{pmatrix}$$

Next a library Ψ is built with candidate nonlinear functions of y.

$$\Psi(y) = \begin{pmatrix} | & | & | & | & | & | & | & | & | \\ 1 & y & y^2 & \dots & y^n & \dots & \sin(y) & \cos(y) & \dots \\ | & | & | & | & | & | & | & | & | \end{pmatrix}$$

y^n denotes all polynomial combinations of y. The set of candidate functions can be chosen from intuition and physical knowledge. The dynamical system can now be written as a system of ordinary differential equations.

$$\dot{y} = \Psi(y)\xi$$

Such a dynamical model is identified by using a convex L_1 regularised sparse regression as denoted below:

$$\xi = \min_{\xi} ||\dot{y} - \Psi\xi||_2 + \lambda||\xi||_1$$

This may be solved with a convex optimisation solver. From the sparse coefficient vector ξ we can now read an interpretable dynamical model. This method has got the huge advantage of interpretability over neural networks, which are considered as black-boxes.

7.3.2 Lorenz Attractor Sparse Identification

In this section the nonlinear dynamics of the chaotic Lorenz attractor will be identified with the help of the SINDy algorithm. The governing equations are given as:

$$\begin{aligned}\dot{x} &= \sigma(y - x)\\ \dot{y} &= x(\rho - z) - y\\ \dot{z} &= xy - \beta z\end{aligned}$$

The aim is now to discover these equations from the fully simulated data x, y, z. For the data shown in the graph below 7.30 , typical values $\sigma = 10, \rho = 28, \beta = \frac{8}{3}$ are chosen.

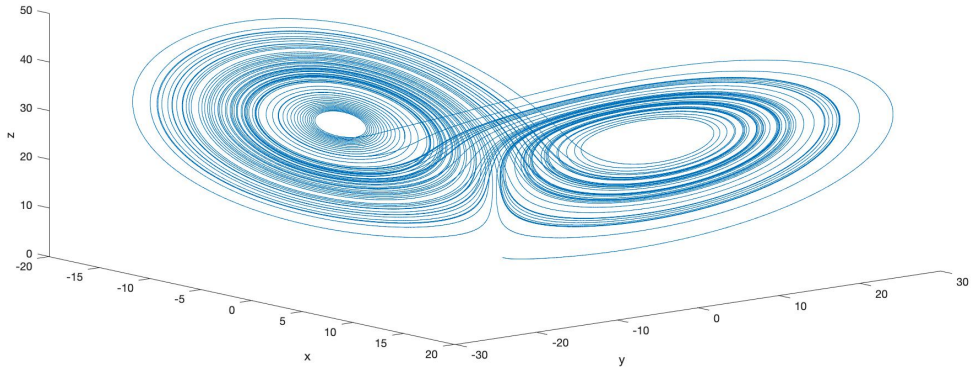

Figure 7.30: Lorenz attractor simulation (initial conditions $(0, 1, 1.05)$, time span $(0, 80)$)

From the data x, y, z the derivatives $\dot{x}, \dot{y}, \dot{z}$ are computed with finite difference schemes. The library in this example contains the following candidate terms:

$$\Psi = \begin{pmatrix} | & | & | & | & | & | & | & | & | \\ x & y & z & x^2 & y^2 & z^2 & xy & xz & yz \\ | & | & | & | & | & | & | & | & | \end{pmatrix}$$

Consequently the sparse regression we have to solve has got the following form:

$$\dot{X} = \Psi(X)\Xi$$

$$\begin{pmatrix} | & | & | \\ \dot{x} & \dot{y} & \dot{z} \\ | & | & | \end{pmatrix} = \begin{pmatrix} | & | & | & | & | & | & | & | & | \\ x & y & z & x^2 & y^2 & z^2 & xy & xz & yz \\ | & | & | & | & | & | & | & | & | \end{pmatrix} \begin{pmatrix} | & | & | \\ \xi_x & \xi_y & \xi_z \\ | & | & | \end{pmatrix}$$

The library was chosen arbitrary and could also contain other terms, however the relevant are included in this case. Solving the optimisation problem

$$\Xi = \min_{\xi} ||\dot{X} - \Psi\Xi||_2 + \lambda||\Xi||_1$$

with the cvx solver, we obtain the following sparse coefficients ξ_x, ξ_y, ξ_z.

	xi_x	xi_y	xi_z
x	-10	28	-1.9511e-05
y	10	-0.99996	1.331e-05
z	4.9629e-07	-4.2126e-07	-2.6666
xx	3.7975e-07	5.5362e-08	1.3865e-05
yy	2.9479e-08	-3.4585e-07	4.0407e-06
zz	-4.0937e-08	-7.1987e-09	-1.3338e-06
xy	-3.1456e-07	4.3069e-07	0.99998
xz	2.026e-06	-1	5.2337e-07
yz	-1.595e-06	-6.285e-07	-4.1398e-07

Figure 7.31: Identified sparse coefficient vectors Ξ

From the dominating coefficients in this table the governing equation can be read. The accuracy of such models increases with the precision of the data X and a decreasing time-step size. The model from the table is now simulated and compared to the Lorenz attractor for different initial values.

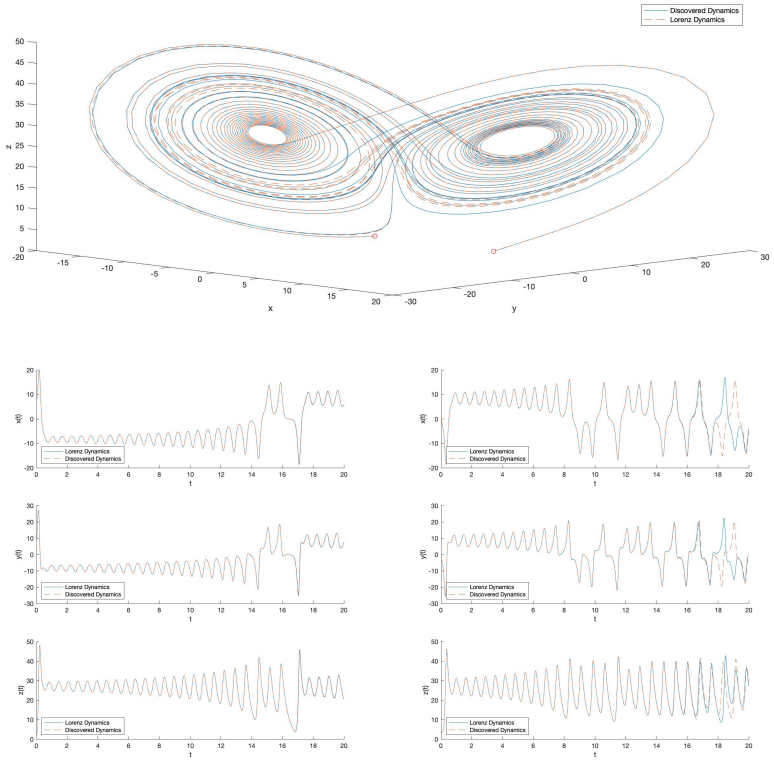

Figure 7.32: Simulation of discovered model and comparison to true Lorenz attractor

References

[1] Christian Karpfinger, Hellmut Stachel. *Lineare Algebra*. Springer Spektrum, 2020.

[2] Christian Karpfinger. *Höhere Mathematik in Rezepten*. Springer Spektrum, 2015.

[3] Gilbert Strang *Linear Algebra*. Springer, 2003.

[4] Charles J. Neumann, *NOAA Technical Memorandum NWS SR-62* [*An Alternate to the HURRAN - Tropical Cyclone Forecast System*.] NOAA, 1972.

[5] Charles J. Neumann, *The National Hurricane Center NHC83 Model* NHC, 1988.

[6] Emiel Por, Maaike van Kooten, Vanja Sarkovic *Nyquist-Shannon sampling theorem* 2019.

[7] Julien Weiss *A tutorial on the Proper Orthogonal Decomposition* TU-Berlin, 2019.

[8] Steven L. Brunton, J. Nathan Kutz *Data-Driven Science and Engineering* Cambridge University Press, 2019.

[9] Marie Michenková *Numerical Algorithms for low-rank matrix completion problems* Swiss Federal Institute of Technology Zurich, 2011.

[10] Maria Jardim Beira, Pedro José Sebastião *A differential equations model-fitting analysis of COVID-19 epidemiological data to explain multi-wave dynamics* Nature Scientific Reports