Discrete Basis Function Methods for the Solution of Inverse Problems in Mechanical Measurements

Sabrina Pretzler

Discrete Basis Function Methods for the Solution of Inverse Problems in Mechanical Measurements

supervised by

O.Univ.-Prof. Dipl.-Ing. Dr.techn. Paul O'Leary

and

Dipl.-Ing. Dr.mont Matthew Harker

Chair of Automation
University of Leoben
Austria

Eidesstattliche Erklärung

Danksagung

Hiermit möchte ich mich bei einer Reihe von Menschen bedanken, ohne die die Verfassung meiner Diplomarbeit in dieser Art und Weise nicht möglich gewesen wäre. Der größte Dank gilt meinen Eltern, Monika und Manfred. Sie haben mich nicht nur finanziell sondern auch durch ihre fachliche sowie menschliche Erfahrung unterstützt. Durch ihr Vertrauen in mein Können war eine weitgehend sorgenfreie Ausbildung erst möglich.

Ein großes Dankeschön gilt Paul, Doris, Matthew, Christoph und den anderen Mitarbeitern am Lehrstuhl für Automation, welche mir das Verfassen der Diplomarbeit erleichtert haben und bei organisatorischen Erledigungen stets zur Seite standen. Weiters möchte ich mich bei der Firma Geodata GmbH für die kooperative Zusammenarbeit, das zur Verfügung gestellte Material sowie für die mir entgegengebrachte Wertschätzung bedanken.

Selbstverständlich möchte ich auch allen Studienkollegen und Freunden für die schöne Studienzeit in Leoben bedanken, welche mein Leben bestimmt in vielerlei hinsicht geprägt hat. Großer Dank gebührt auch meinen Schwestern Nadja, Lisa und Fanni, welche mich teils fachlich, teils Menschlich während meinem Studium begleiteten. Das erfolgreiche Abschließen dieser Arbeit wurde vor allem durch Kevin ermöglicht, der mir den nötigen Beistand entgegen brachte und mir in jeder schwierigen Situation Rückhalt gab.

Abstract

This thesis presents a new approach to curve reconstruction from over constrained gradients. This type of problem arises when measuring deformation of structures using inclinometers. The new methods investigated are based on discrete orthonormal polynomials and a method of synthesizing constrained basis functions, whereby the constrained basis functions span the complete space of all possible solutions. Furthermore, they are ordered in increasing mode number, which supports a simple solution for spectral regularization.

Two new methods are derived for the reconstruction of curves from gradients. The first reconstruction method uses admissible functions for regularization, the second method is of variational nature. Monte Carlo simulations are presented which verify the principle of the numerical approach. Additionally a real inclinometer measurement system for the measurement of a deflected beam was implemented and an independent optical system was constructed for measurement validation. The real measurements confirmed the correctness of the new approach. Furthermore, they revealed issues which are relevant for future research, i.e., placing constraints on the interpolating functions and not on the reconstructed points.

Index Terms

curve reconstruction from gradients, discrete orthonormal polynomials, admissible functions, inclinometers, inverse boundary value problem

Kurzfassung

Diese Diplomarbeit präsentiert einen neuen Ansatz zur Kurvenrekonstruktion von überbestimmten Gradienten. Diese Aufgabestellung entsteht bei der Verwendung von Inklinometern (Neigungssensoren) für die Messung der Durchbiegung von Strukturen. Die untersuchten Methoden basieren auf diskreten, orthonormalen Polynomen und auf einer Methode zum Generieren von Ansatz Funktionen unter Randwertbedingungen. Die Ansatzfunktionen für Randwertbedingungen spannen den Raum aller möglichen Lösungen für die Durchbiegung auf. Zusätzlich sind die Ansatzfunktionen sortiert nach aufsteigendem Polynomgrad, wodurch die einfache Anwendung spektraler Regularisierung ermöglicht wird.

Zwei neue Methoden zur Kurvenrekonstruktion wurden hergeleitet. Die erste Rekonstruktionsmethode verwendet gültige Funktionen für Regularisierung, die zweite Methode basiert auf Variationsrechnung. Zur Verifizierung der numerisch hergeleiteten Ansätze wird eine Monte Carlo Simulation verwendet. Zusätzlich wurde ein praktischer Prüfaufbau erstellt. Dieser dient als echtes Inklinometermesssystem zur Messung der Durchbiegungen eines Balkens. Ein unabhängiges, optisches Messsystem wurde zur Messverifizierung verwendet. Die echten Messungen bestätigen die Richtigkeit der neuen Ansätze. Darüberhinaus zeigten sich während der Messdatenauswertung relevante Themen für weitere Forschung, wie die Verwendung der Randwertbedingungen an interpolierenden Funktionen und nicht an den rekonstruierten Punkten.

Schlagwörter

Kurvenrekonstruktion von Gradienten, Diskrete Orthogonale Polynome, Gültige Funktionen, Inklinometer, Inverse Randwertproblem

0.1 Summary

This diploma thesis is dealing with curve reconstruction from gradients, e.g. inclinometer measurements, which is essentially an inverse problem. Inclinometers are instruments used to measure gradients, commonly known as slope or tilt, of an object with respect to gravity. They have a wide field of application, e.g. civil engineering, mechanical engineering, aircraft industry, geoscience; they are used to monitor ground subsidence [22] and deformation of structures [18],[35].

The reconstruction of a curve from gradients is fundamentally an inverse problem. If, in addition, there are constraints placed on the curve, e.g. in the case of a cantilever, the measurement is an inverse boundary value problem. This diploma thesis presents new approaches for curve reconstruction from gradients allowing for regularization, least-squares fittings, with orientational and positional constraints. The methods presented deliver numerical solutions for Inverse Boundary Value Problems and Inverse Initial Value Problems. The numerical integration of an over-constrained system is possible and easy to implement. The solution to the inverse problem is a simple matrix multiplication; the exact number of computations required is known a-priori, which makes this method suitable for real-time computations.

In this thesis two different solution approaches are derived and verified; one reconstruction solution uses admissible functions for the boundary value problem; the second solution is of variational nature. Both solutions are formulated for the deformation of both continuous structures and piece wise linear approximation of structures. The proposed methods were verified using numeric simulations in MATLAB, which were performed for both continuous and non-continuous rods.

A test apparatus has been designed and implemented with the aim of determining the viability of the newly proposed methods in conjunction with the use of commercially available sensors. The test system consists of a deformable beam, whereby different constraints can be applied. In addition, an optical reference measurement was developed, which enables the comparison of the reconstructed curve to the optically measured curve. A series of n inclinometers is mounted along the beam, which provides a means of measuring the local gradient. The inclinometer data is acquired using a Digital Bus. The measured data is stored in an SQL Database. The computational reconstruction using the inclinometer data measured in the laboratory is performed in MATLAB, whereby the Data is directly retrieved from the SQL Database. The optical reference measurement is also performed using MATLAB and evaluated using Image Processing and Polynomial fitting with Basisfunctions. The measurement and the reconstructions are compared in order to verify the methods introduced in this thesis. The discrete solutions to curve reconstruction from gradients presented in this thesis have been verified by both numerical simulations and physical experiments. The methods are viable for real-time measurements of structure with constraints.

Contents

Chapter 1

Introduction

Monitoring structures and deformations is becoming fundamental in many parts of engineering. Typical elements for such purposes are theodolites and optical laser rangefinders. These instruments need permanent view of the object, which diminishes its field of application. For example in monitoring train tracks, bridges or heavy machinery under grounds, as well as in the monitoring of ground subsidence, a perfect view is not always given. This, and additionally the reduction of costs of inclinometers as well as their constantly improving accuracy, is why scientists revert to inclination measurements for this field of application.

The inclination displayed by inclination measuring instruments and sensors is based on the gravitation and has been used already in ancient times for the reconstruction of buildings or navigation at seas. The simplest and best known inclination measurement is the water level. Due to higher accuracy, better resolution, reliability and the possibility of application in automatic systems, the use of electronic inclination measurements, be it of inductive or capacitive nature, is becoming more popular [16]. Nowadays inclination measuring instruments are used for

- measuring an inclination,
- measuring the straightness of a line, e.g. of a machine guide way,
- measuring the flatness of a surface and
- long-term monitoring of objects.

Typical examples of the use of inclinometers in mechanical engineering is the detection of slanting in construction machines, the levelling in agricultural and commercial vehicle technology as well as the tracking of the position of the sun for solar panels. Furthermore they find use in monitoring of deformation of structures [18] and in wind energy turbines for both, monitoring [15] and alignment adaptation [2] purposes. Inclinometer measuring systems are indispensable in geotechnical monitoring of earth subsidence [10].

The essential difficulty in inclinometer measurement evaluation lies in the numerical integration, in the reconstruction of an element's deflection. Even more so, when boundary conditions and other constraints are given for the reconstructed curve. This type of problem falls into the general class of curve (or surface) reconstruction from gradients (or gradient fields), with the possible fulfilment of combined Dirichlet and Neumann boundary conditions.

Basically, there are two general cases of curve reconstruction from gradients:

1. reconstructions without constraints;
2. reconstructions with positional (Dirichlet) or derivative (Neumann) constraints.

In geoscience there are many applications for both types of inclinometer measurements. Landslide investigation and monitoring slope stability [22] for example, are measurements which can be linked to type one. In order to monitor landslides, inclinometer probes are inserted in hollow casings using guide wheels to ensure correct alignment (see Figure 1.1). The hollow casings are placed deep enough into the ground, where they perform the same movement as their surroundings. When monitoring ground slippage there are no constraints to be placed onto the curve, as there are no fixed points along the measurement baseline. The movement of the earth is detected by the change of the inclinometer values. An additional measurement has to be performed in order to account for the constant of integration.

When monitoring rigid structures, positional and/or derivative constraints are given and need to be taken into account. In this case the reconstruction of the curve is characteristic of type two. For example, the deflection of a simple cantilever is subject to both positional, $\boldsymbol{y}(0) = 0$, and derivative constraints $\dot{y}(0) = 0$, $\ddot{y}(0) = 0$. Furthermore, constraints may be homogeneous, as in the case of the cantilever, or non homogeneous. In this case the inclinometers are mounted onto the structure where they measure the gradient (local slope) of the structure at the position where it is mounted. Reconstructing the change of the inclinometer values delivers a curve showing the change of the structure. An example of an application of the reconstruction of rigid structures is the monitoring of heavy machinery. Due to the machine's physical properties, the positional, first and second derivative constraints are known and can be placed on the structure.

Much research has been done in the field of numeric curve reconstruction [30, 6]. Burdet and Zanella [6], for example, tested two methods: direct polynomial approximation functioned but was prone to large errors since it did not permit a least squares approach; they then resorted to using a linear combination of possible deflection curves which were pre computed for the structure. This approach gave superior results. There are two dangers associated with their approach:

1. The set of possible deflection curves used are not complete, i.e. they can not model all possible deformations.
2. The curves used are all bending modes of the structure. Consequently, the method provides no possibility of detecting an unexpected deformation, i.e. a deformation resulting from damage to the structure and its constraining elements.

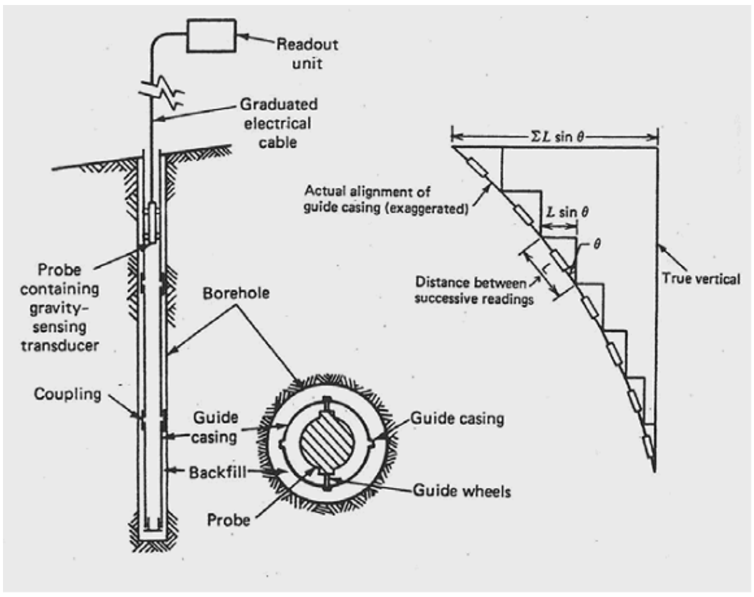

Readout unit
Graduated electrical cable
Probe containing gravity-sensing transducer
Borehole
Coupling
Guide casing
Backfill
Probe
Guide casing
Guide wheels
ΣL sin θ
Actual alignment of guide casing (exaggerated)
L sin θ
θ
L
Distance between successive readings
True vertical

Figure 1.1: Principle of inclinometer probe operations, after [10].

Xingmin et al. [18] implements a least squares approximation with Lagrange interpolating polynomials for the inclinometer values and perform polynomial integration to obtain the deflection curve, but without regularization. This method can neither accommodate orientational constraints, nor implement regularization.

A simply supported beam was considered by Van Cranenbroeck [35]. He computed the explicit polynomial curves associated with the beam and the associated derivative curves. The inclinometer values are used to determine the coefficients of the derivative curve and the original curve is deducted implicitly; the method cannot implement least-squares fitting or regularization. Furthermore it uses the Euler-Bernoulli-Equation for the reconstruction. This accounts for a restriction to Euler-Bernoulli solutions. Therefore the reconstruction is applicable only for known deformations and loads.

A more general problem was reported by Golser [11], where inclinometers were used to measure rail track subsidence. This application has both orientational and positional constraints on the solution. He proposed a local approximation technique, similar to a Savitzky-Golay approach [33], whereby three inclinometer values were used to determine the solution at the center point of the three measurements. This solution does not implement general constraints and does not ensure a global least square minimum fit. In addition, it is limited to local second order curves.

The fundamental underlying problem in the measurement of the deformation of structures using inclinometers is the reconstruction of a curve from its local derivatives in the presence of constraints. The inclinometers measure the local gradients of the deformed curve, where the fixed points in the structure correspond to boundary values. It is an iterative method and does not provide for regularization. Additionally, since the structures will in general have multiple fixed points, i.e., multiple boundary values, the Runga-Kutta method must be extended with the method of Shooting [32]. Shooting has the problem, in particular in the presence of noise, that the multiple solutions may not be mutually convergent. The simple Shooting method fails completely under such conditions. It then needs to be extended in some manner so as to accommodate arbitration between the different solutions. Usually some form of averaging is used to obtain the ”best” solution [32].

In 2005 an ASTM standard for monitoring ground movement [19] was established. It provides a detailed information on the installation of the test apparatus, as well as on the procedure of the measurement and the calculations necessary for the reconstruction of the unconstrained system. As the standard is written for monitoring ground movement, which is a reconstruction of Type 1, the article does not cover constraints which may or may not need to be placed onto a structure/ system.

This thesis presents two new procedures for the reconstruction of deflection curves of arbitrary structures from inclinometer measurement data. The main focus lies in defining and evaluating several new solutions for curve reconstruction, in both variational and gradient space; both, constraints and regularization are supported. The solution is numerically efficient and by definition, suitable for real-time computations. Two new solutions to curve reconstruction are derived. The constraints associated with specific structures are utilized for regularization of the solution. The methods are applicable for uniformly, as well as non uniformly spaced nodes. The different solutions are numerically tested using synthetic data. In addition, a laboratory measurement set-up modelling the deformation of a beam was designed and implemented. This enables physical verification of the results. Inclinometers are mounted onto the metal sheet and deliver the values used for the reconstruction.

Chapter 2

Framework

2.1 Boundary Value Problems and Initial Value Problems

Modern computer driven sensor systems are spatially and temporally discrete. The general monitoring structure is of continuous nature. This requires a discrete solution to systems of equations which represent a continuous system. In the case of inclinometers the first derivative of position with respect to space, i.e. the slope, is measured. If the structure is additionally subject to constraints, e.g., immovable pillars etc., the measurement problem becomes an inverse boundary value problem. An algebraic solution of such problems requires three fundamental elements:

1. Continuous basis functions which are sampled at discrete points form discrete orthonormal basis function sets. Such basis functions enable the correct calculation of properties of continuous systems from discrete observations;

2. A method of generating admissible functions, which fulfil the constraints of the boundary or initial value problem;

3. Linear differential operators of sufficient accuracy to enable the direct algebraic solution of differential equations.

2.2 Continuous and discrete basis functions

Considerable research has been performed on discrete polynomials and their synthesis [24, 37, 38, 36, 17, 39, 40, 1]. The research was primarily in conjunction with the computation of moments for image processing. Gram [13] introduced the concept of continuous polynomials which form unitary discrete polynomial basis when sampled at discrete points. He also showed their application to least squares approximation of continuous polynomials from a finite set of discrete observations. The polynomials were synthesized via a three term relationship in a process which is now called a Gram-Schmidt orthogonalization[1]. The recurrence relationship for the Gram polynomials is,

$$\boldsymbol{g_n}(x) = 2\ \alpha_{n-1}\ x\ \boldsymbol{g}_{n-1}(x) - \frac{\alpha_{n-1}}{\alpha_{n-2}}\ \boldsymbol{g}_{n-2}(x), \tag{2.1}$$

whereby,

$$\alpha_{n-1} = \frac{m}{n} \left(\frac{n^2 - 1/2}{m^2 - n^2} \right)^{1/2} \tag{2.2}$$

and

$$\boldsymbol{g}_0(x) = 1, \quad \boldsymbol{g}_{-1}(x) = 0 \quad \text{and} \quad \alpha_{-1} = 1, \tag{2.3}$$

the computation is performed for equidistant discrete nodes x_k lying on the real axis,

$$x = -1 + \frac{(2k-1)}{m}, \quad 1 \leq k \leq m, \tag{2.4}$$

where m represents the number of rods; note these points do not span the full range $[-1, 1]$. The basis functions are scaled by $\sqrt{m}$ yielding a unitary basis set. Indeed this procedure generates orthonormal polynomials; however, the process is unstable with respect to numerical roundoff errors, so that previously a complete set of discrete polynomials with small enough errors could not be synthesized for a large number of nodes and/or for polynomials of higher degree [28]. The first solution for the generation of virtually perfect polynomial basis sets of arbitrary size can be associated with Mukundan [23]; however, the errors in this algorithm— although very small — are concentrated at lower degree polynomials.

2.3 A numerically stable recurrence relationship

All polynomials, both complex and real[2] can be synthesized from a functional analysis point of view using a three term recurrence [9]. A new numerically stable two step recurrence procedure [27] was introduced in 2008. This procedure enables the synthesis of orthonormal basis functions from arbitrary nodes located within the unit circle in the complex plane. Furthermore, the procedure is stable with respect to numerical errors, e.g., a Gram polynomial of degree $d = 1000$ can be generated and the maximum error corresponds to the numerical resolution of the data type being used for the computations — in the case of MATLAB the maximum error corresponds to the spacing of floating point numbers, i.e., $eps = 2.22E - 16$.

The two step recurrence procedure is formulated as a vector-matrix algebraic computation. The following nomenclature is used: $\boldsymbol{g_n}$ is a vector containing the Gram polynomials of degree n computed at all nodes; $\boldsymbol{P_n}$ is a matrix formed by concatenating the first n basis functions, i.e., $\boldsymbol{P_n} \triangleq [\boldsymbol{p_1}, \ldots, \boldsymbol{p_n}]$; the symbol $\circ$ represents the Hadamard product[3].

[1]"The method of generating n orthogonal vectors from n linearly independent vectors is called the *Gram−Schmidt* process"[7]

[2]Some examples of polynomials generated from specific nodes are: the Fourier basis can be synthesized from nodes which are evenly spaced along the unit circle in the complex plane; the discrete cosine basis functions are produced when the Tchebychev nodes, and the Gram polynomials are generated from nodes evenly spaced on the real axis.

[3]The Hadamard product of the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, i.e., $\boldsymbol{c} = \boldsymbol{a} \circ \boldsymbol{b}$ is implemented in MATLAB as: $c = a. * b$.

Step 1: An initial estimate, i.e., for the n^{th} polynomial is generated using,

$$\boldsymbol{p_n} = \alpha_n \, \boldsymbol{x} \circ \boldsymbol{p}_{n-1} + \beta_n \, \boldsymbol{p}_{n-2}. \tag{2.5}$$

This basis function is subject to numerical errors which, if not eliminated, accumulate with further computations of the recurrence relationship.

Step 2: The unwanted correlations are eliminated using a complete re-orthogonalization,

$$\begin{aligned} \boldsymbol{p_n} &= \boldsymbol{p_n} - \boldsymbol{P}_{n-1} \left(\boldsymbol{P}_{n-1}^{\mathrm{T}} \, \boldsymbol{p_n} \right), & (2.6) \\ &= \left(\boldsymbol{I} - \boldsymbol{P}_{n-1} \, \boldsymbol{P}_{n-1}^{\mathrm{T}} \right) \cdot \boldsymbol{p_n}, & (2.7) \end{aligned}$$

The components $\boldsymbol{P}_{n-1} \, \boldsymbol{P}_{n-1}^{\mathrm{T}}$ is a projection onto the basis functions $\boldsymbol{P}_{n-1}$; $\boldsymbol{I} - \boldsymbol{P}_{n-1} \, \boldsymbol{P}_{n-1}^{\mathrm{T}}$ is the projection onto the orthogonal complement. The correct computational sequence is given in Equation 2.6. Finally the vector is renormalized to ensure a unitary basis,

$$\boldsymbol{p_n} = \frac{\boldsymbol{p_n}}{\|\boldsymbol{p_n}\|_2}. \tag{2.8}$$

Basis function sets synthesized in this manner fulfil the condition,

$$\boldsymbol{P_n}^{\mathrm{T}} \, \boldsymbol{P_n} = \boldsymbol{I_n}. \tag{2.9}$$

2.3.1 Runge Phenomenon

The Runge phenomenon is well known; it describes the tendency of interpolation polynomials to exhibit excessive oscillations at the edges of an interval. The effect can be demonstrated using the Runge function,

$$f(x) = \frac{1}{1 - 25x^2}, \tag{2.10}$$

defined on the interval $[-1 \leq x \leq 1]$. It is easily verified that irregularly spaced nodes are required if stable interpolation results are to be achieved.

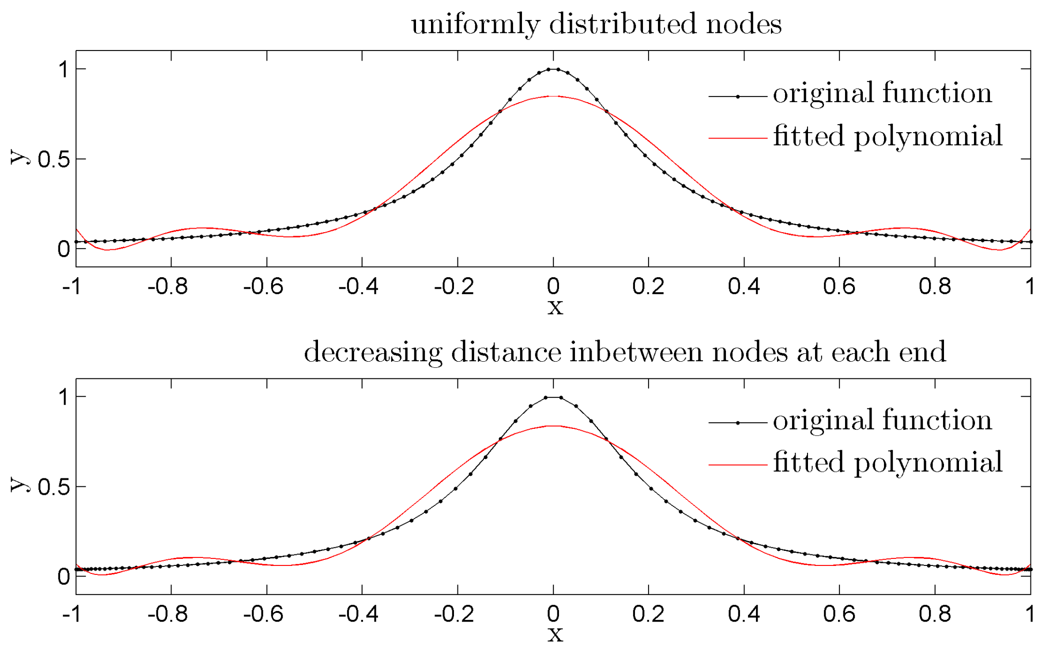


Figure 2.1: This Figure shows the Runge function, in black. The upper graph shows the Runge function generated on 100 evenly spaced points x. The graph below shows the Runge function generated on 100 unevenly distributed points x. Here the cosine function was used to generate the unevenly distributed points, whose density becomes greater as the function approaches -1 and 1. It is clearly to be seen, that the ends of the bottom graph have lower tendency to oscillations, which corresponds to an optimal nodes placement for this polynomial fit.

2.3.2 Optimization of Covariance Propagation

The derivation of the equations for covariance propagation is given in Section 2.10. At this stage the effect of node placement on covariance propagation is shown. To demonstrate this effect, the case of monitoring the deformation of a cantilever using inclinometers[4] is considered. The monitoring of structures using inclinometers is an established technique [18], a good overview can be found in [22]. The present standards, e.g. [19] do not deal with the solution of such measurements when a system is constrained. A framework for the solution of such problems [29] was proposed in 2008. However, none of these publications consider the use of optimal sensor placement with the aim of minimizing the variance of the measurement result.

The first three constrained polynomial basis functions which fulfil the constraints associated with a simple cantilever are shown in Figure 2.2. These polynomials were generated by applying constraints to a complete set of Gram polynomials. They are unitary and admissible functions for the solution of the inverse boundary problem of determining the bending of a cantilever from inclinometer measurements.

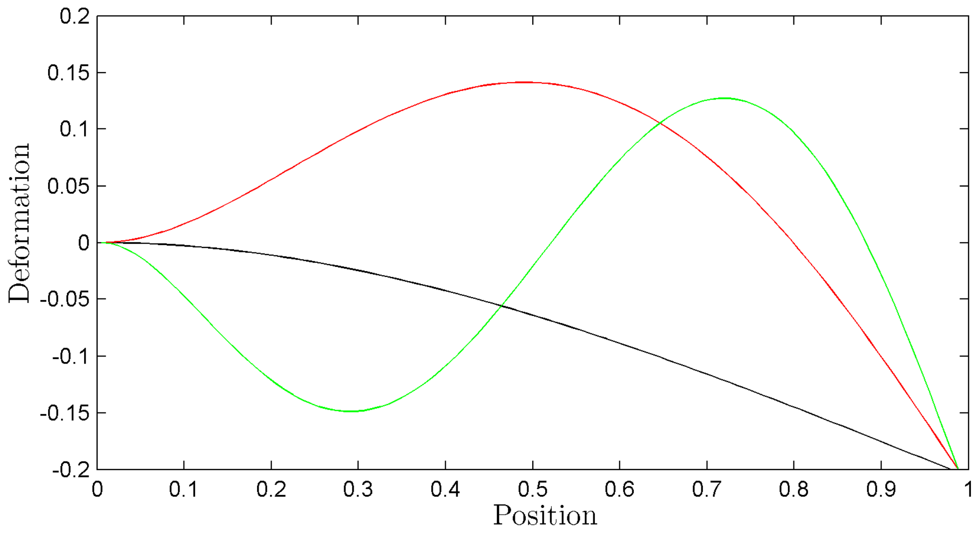


Figure 2.2: The first three constrained polynomial basis functions which fulfil the constraints associated with a simple cantilever. These admissible functions fulfil the boundary conditions $\boldsymbol{y}(0) = 0$, $\dot{\boldsymbol{y}}(0) = 0$ and $\ddot{\boldsymbol{y}}(0)$.

The variance of the solution along the length of the beam depends on the placement of the sensors, which corresponds to the placement of the nodes. This means that with the same sensors and noise levels, a measurement with better confidence interval can be achieved using optimal placement of the sensors and the basis functions computed for the corresponding nodes. The variances of the reconstructions of the first mode of the cantilever with uniformly spaced sensors and with sensors placed at points corresponding to the Tchebychev points are shown in Figure 2.3.

[4]A cantilever has been chosen for this example to be in contrast to the oscillatory behaviour shown in the previous point. The cantilever measurement problem is dealt with in more detail later in this paper.

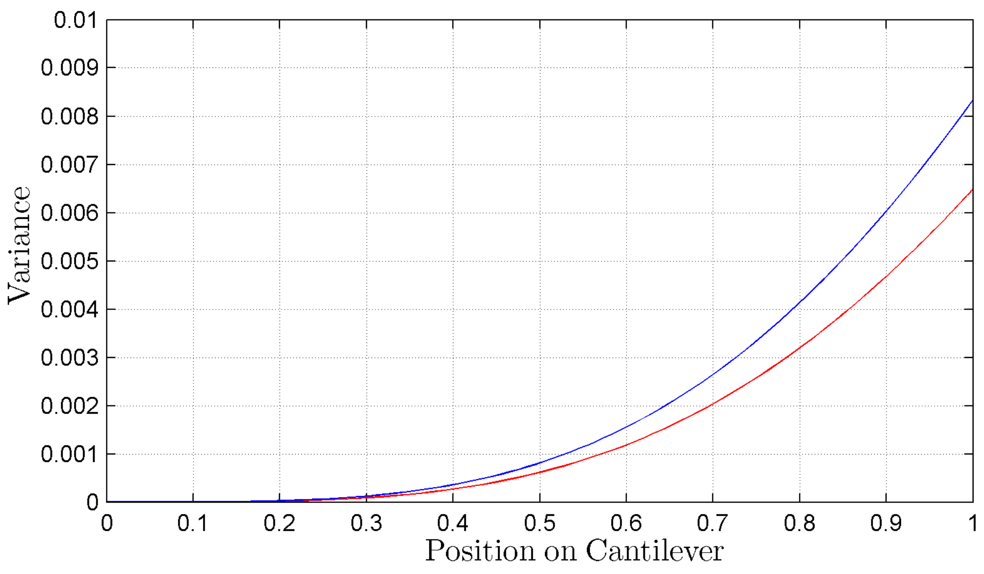


Figure 2.3: Variance of the cantilever deformation reconstructed from inclinometer measurements, assuming independent and identically distributed noise for the sensor perturbations. The reconstruction is performed using constrained basis functions, red: with uniformly spaced sensors (nodes); blue: with the same number of sensors but with a continually closer spacing approaching the end of the cantilever.

2.3.3 Basis Function Design

The ability of the new synthesis procedure to generate discrete orthonormal basis functions from arbitrary nodes enables the design of customized basis functions for specific applications. In this manner the performance of the solution to inverse problems can be optimized with respect to noise propagation and numerical efficiency. This is of particular interest when signals, which are not well described by either the Fourier transform, Discrete Cosine transform (DCT) or Gram basis functions, need to be analysed. One example for such signals is the Taylor function $x^{(-\frac{1}{2})}$. When modelling the Taylor function with one of the listed basis functions above, the model will not be satisfying, as problems such as the *Runge phenomenon* will occur. This is when custom basis functions come in handy. Consider, for example, the custom basis functions shown in Figure 2.4. They have been designed for the detection of a specific time limited signal emanating from an acoustic resonator mounted in the mould of an injection moulding machine. The basis functions model the characteristic envelope of the oscillations and the characteristic frequencies.

Knowing this characteristic envelope of the oscillation, we can generate a complex set of nodes which fits our example (see Figure 2.5). The placement of the nodes on the complex plane was performed using polar coordinates. The radius determines the magnitude envelope for the set of complex basis functions and the angular coordinate determines the instantaneous phase. The resulting complex discrete basis functions are in perfect quadrature, this eliminates the need for an all-pass filter [21] in the implementation of an optimal signal detector and it

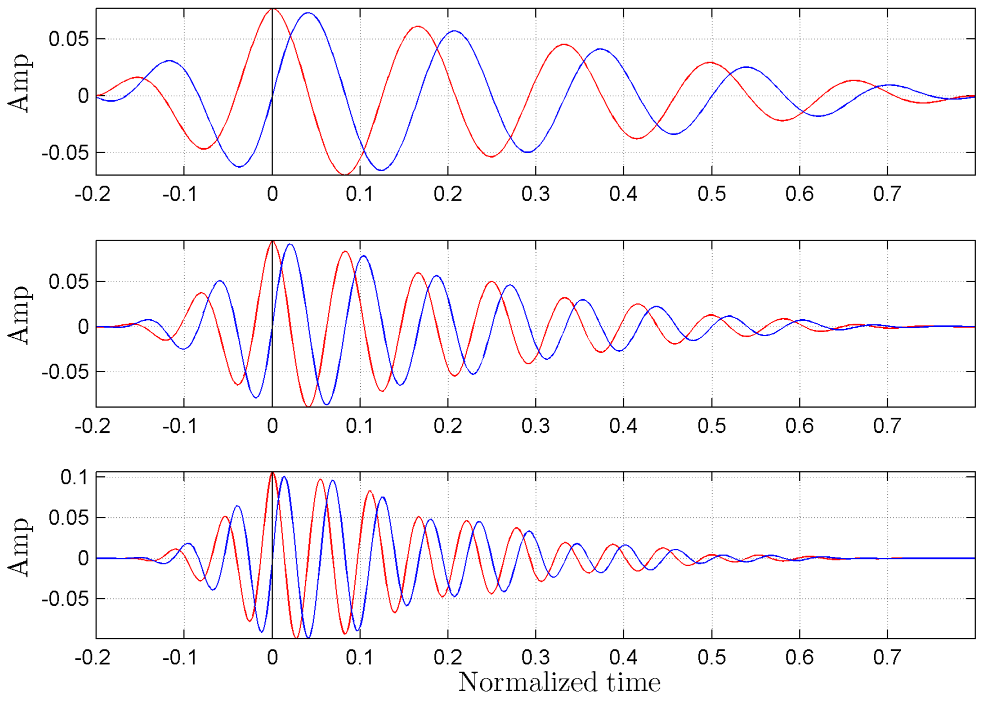


Figure 2.4: The real (red) and imaginary (blue) components of the first three complex basis functions. The real and imaginary components are in quadrature to each other and bounded by one and the same envelope. This makes the functions locally shift invariant. These functions were designed specifically to detect a transient acoustic event in the monitoring of an injection-moulding machine.

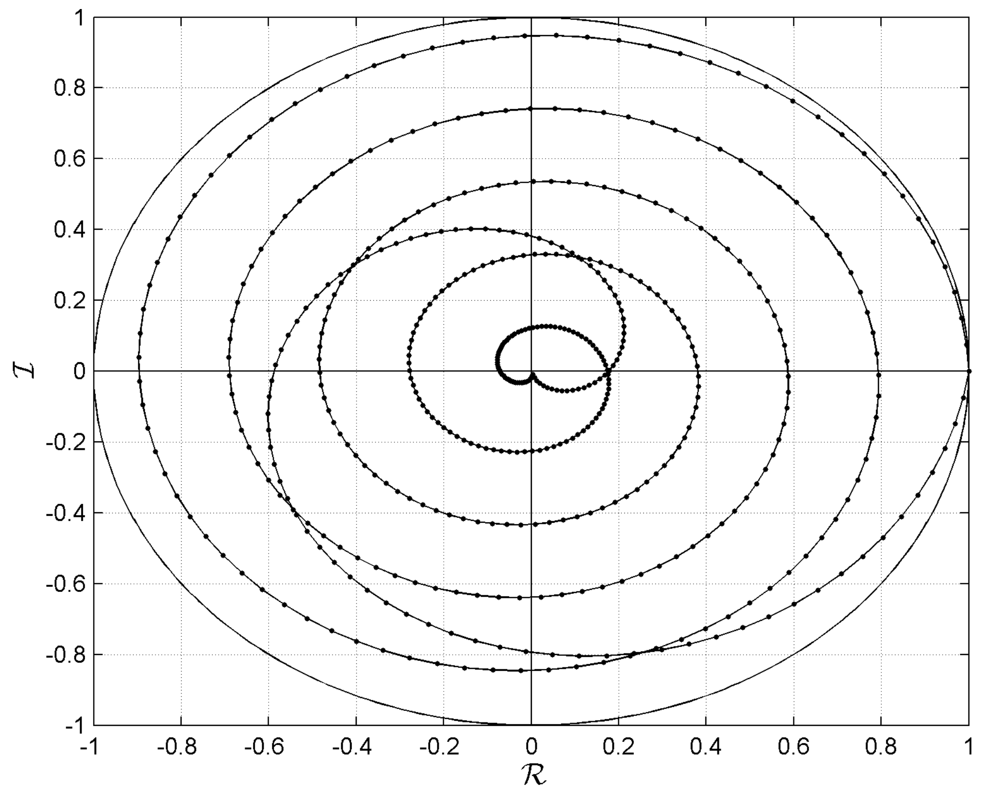


Figure 2.5: The required placement of the nodes on the complex plane for the generation of the basis functions shown in Figure 2.4.

ensures that the detection process is locally shift invariant.

2.4 Synthesizing derivatives of discrete basis functions

To solve differential equations and inverse problems it is necessary to have both the basis functions and their derivatives available. The generation of the derivative of the basis functions via derivatives of the recurrence relationships is known for the Legendre and Tchebychev polynomials [20]. The procedure is extended here to the generalized recurrence relationship,

$$\boldsymbol{p_n} = \alpha \left(\boldsymbol{p}_{n-1} \circ \boldsymbol{x}\right) + \boldsymbol{P}_{n-1}\boldsymbol{\beta}. \tag{2.11}$$

Taking the derivative with respect to x yields,

$$\dot{\boldsymbol{p_n}} = \alpha \left(\dot{\boldsymbol{p}}_{n-1} \circ \boldsymbol{x} + \boldsymbol{p}_{n-1} \circ \dot{\boldsymbol{x}}\right) + \dot{\boldsymbol{P}}_{n-1}\boldsymbol{\beta}. \tag{2.12}$$

The values of α and $\boldsymbol{\beta}$ are already computed during the synthesis of the basis functions. Consequently, the derivatives of the basis functions $\dot{\boldsymbol{P}}$ can be computed simultaneously with basis functions $\boldsymbol{p}$ with marginal additional numerical effort. This procedure is independent of the placement of the nodes.

2.5 Differentiating Matrices $\boldsymbol{D}$

A differentiating matrix $\boldsymbol{D}$ which computes the numerical derivatives of a vector for each entry, i.e., $\dot{\boldsymbol{y}} = \boldsymbol{D}\,\boldsymbol{y}$, is required if linear differential operators are to be formulated in an algebraic method. The generation of $\boldsymbol{D}$, with sufficient accuracy, is the central issue in finding good algebraic solutions to differential equations and inverse boundary value problems. It is an issue which is not given sufficient attention in present literature. The quality of the solution stands and falls with the quality of $\boldsymbol{D}$.

There are numerous problems associated with the usual three point estimates, as embodied by the *gradient* function in MATLAB[5]: The estimate is only of degree two accurate which is insufficient if we are solving problems whose solutions are of higher degree. Furthermore, the band-diagonal matrix is only of degree one at the ends of the support. This is particularly damaging if constraints are to be placed on the boundaries of the solution.

Given a complete set of basis functions $\boldsymbol{P}$ and their derivatives $\dot{\boldsymbol{P}}$, they are related by differentiating matrix $\boldsymbol{D}$ in the following manner [29],

$$\dot{\boldsymbol{P}} = \boldsymbol{D}\,\boldsymbol{P}. \tag{2.13}$$

Now defining the regularized differentiating matrix $\boldsymbol{D}_r$,

$$\boldsymbol{D}_r \triangleq \dot{\boldsymbol{P}}\,\boldsymbol{P}^{\mathrm{T}} = \boldsymbol{D}\,\boldsymbol{P}\,\boldsymbol{P}^{\mathrm{T}}. \tag{2.14}$$

If the matrix $\boldsymbol{P}$ is complete, i.e., $\boldsymbol{P}\,\boldsymbol{P}^{\mathrm{T}} = \boldsymbol{I}$ then $\boldsymbol{D}_r = \boldsymbol{D}$.

If for example $\boldsymbol{P}$ is a truncated basis set, then $\boldsymbol{D}_r$ is a regularizing differential estimator, a projection of $\boldsymbol{D}$ onto the span of the basis functions contained in $\boldsymbol{P}$.

A differentiating matrix should be rank-1 deficient, i.e., it should have a single null vector - the constant vector. However, the condition number of a differentiating matrix increases with the degree of the polynomial being considered, this is a fundamental property of such matrices. At some point the matrix starts to have additional null spaces which are associated with numerical limitations, this effect is shown in Figure 2.6. Synthesizing $\boldsymbol{D}$ using Equation 2.14 is stable using computation in double format up to degree $d = 33$. The additional null spaces associated with the numerical resolution are a problem when solving inverse problems. Consider solving the simple integration problem, e.g., integrating over a number of inclinometer measurements $\dot{\boldsymbol{y}}_{\boldsymbol{m}}$ to obtain a deflection,

$$\boldsymbol{D}\,\boldsymbol{y} = \dot{\boldsymbol{y}}_{\boldsymbol{m}}. \tag{2.15}$$

Solving for $\boldsymbol{y}$ yields,

$$\boldsymbol{y} = \boldsymbol{D}^{+}\,\dot{\boldsymbol{y}}_{\boldsymbol{m}} + \mathrm{null}\,\{\boldsymbol{D}\}\,\boldsymbol{\alpha}. \tag{2.16}$$

where, $\boldsymbol{D}^{+}$ is the Moore Penrose pseudo inverse and $\mathrm{null}\,\{\boldsymbol{D}\}$ is the null space of $\boldsymbol{D}$ respectively. If numerical null spaces are present in addition to the constant vector, they will lead to incorrect solutions of the simple task of performing numerical integration and the numerical approach will be absolutely unsuitable for more complicated inverse problems.

[5]The differentiating matrix implicitly used within MATLAB can be obtained for an n point vector with the following command *[D, Gx] = gradient(eye(n));*

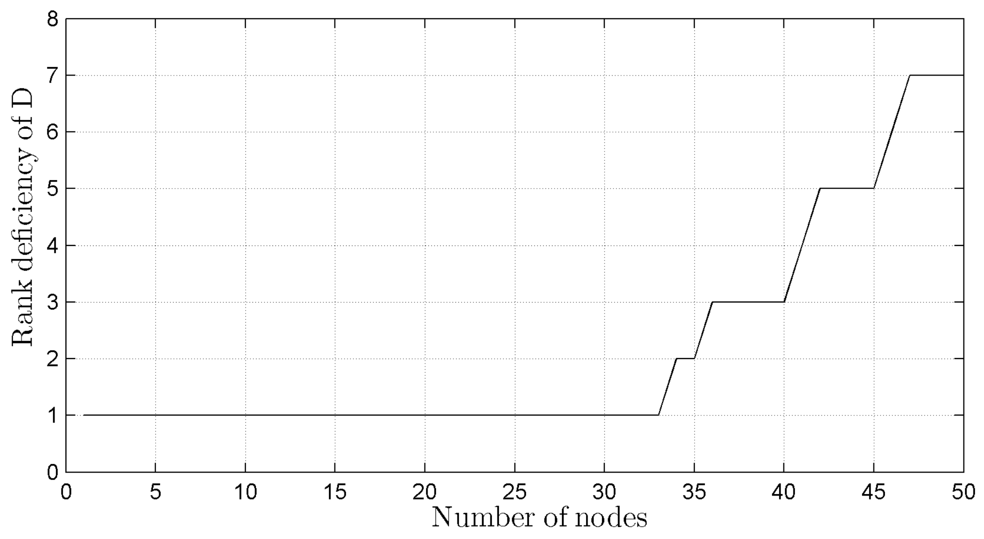


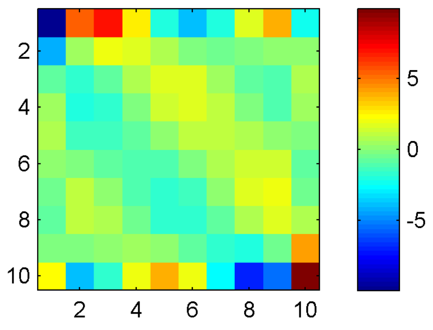
Figure 2.6: Rank deficiency of the differentiating matrix $\boldsymbol{D}$ as a function of the number of nodes when $\boldsymbol{D}$ is synthesized using Equation 2.14. The matrix $\boldsymbol{D}$ is rank-1 deficient, as it should be, up to 33 nodes.

To alleviate this problem, a local differentiating matrix is used when a large number of points, e.g. $n > 20$, is required. A generalized algorithm to generate local differentiating operators, which also works for irregularly placed nodes, were formulated. This requires different estimators at each point along the computation. However, it is important to note that, since the nodes do not change, these operators can be computed a-priori and do not need to be generated at run-time; for this reason a more general approach, which enables the use of varying basis functions for the differential operator instead of using the same Basis function for the entire differential operator, was considered superior at the cost of numerical efficiency.

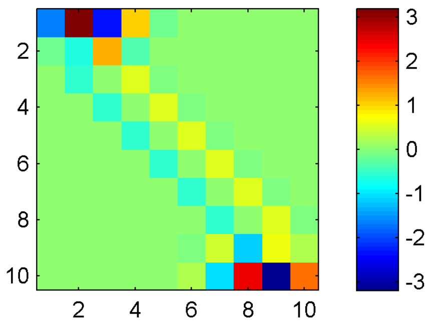
The differentiating matrices used in this thesis are local operators with a predefined support length l_s and a number of basis functions n_b. This enables the generation of both regularizing and non-regularizing operators. Only odd support lengths are supported, since for even l_s the derivative is valid at interstitial points. The matrix $\boldsymbol{D}$ for an n point computation has three implicit partitions:

1. the top partition of the matrix is calculated using the end point equations for the beginning of the support for $n_s = (l_s - 1)/2$ points,
2. the middle part is a band diagonal matrix, with n_b points, where $n_b = (2n_s+1)/(n-n_s)$, for which a center point computation is required, and
3. the last part is, again, calculated using an end point equation for the end of the support for n_s points.

The structure of the three portions of $\boldsymbol{D}$ for the example of $l_s = 5$ and $n = 10$ is shown in Equation 2.17. All computations of the local derivative are of length l_s, ensuring a constant



(a) Global differentiating matrix



(b) Local differentiating matrix

Figure 2.7: On the left hand side a differentiating matrix $\boldsymbol{D}$ for $n = 10$ points is computed for the positive right half set of Tchebychev points. On the right hand side, a local differentiating matrix $\boldsymbol{D}$ with $l_s = 5$ for $n = 10$ points is computed for the positive right half set of Tchebychev points.

approximation degree $d = l_s - 1$.

$$\boldsymbol{D}_{5,10} = \left[\begin{array}{cccccccccc}
+ & + & + & + & + & 0 & 0 & 0 & 0 & 0 \\
+ & + & + & + & + & 0 & 0 & 0 & 0 & 0 \\
\hline
+ & + & + & + & + & 0 & 0 & 0 & 0 & 0 \\
0 & + & + & + & + & + & 0 & 0 & 0 & 0 \\
0 & 0 & + & + & + & + & + & 0 & 0 & 0 \\
0 & 0 & 0 & + & + & + & + & + & 0 & 0 \\
0 & 0 & 0 & 0 & + & + & + & + & + & 0 \\
0 & 0 & 0 & 0 & 0 & + & + & + & + & + \\
\hline
0 & 0 & 0 & 0 & 0 & + & + & + & + & + \\
0 & 0 & 0 & 0 & 0 & + & + & + & + & +
\end{array}\right] \tag{2.17}$$

Each of the three portions is generated by segmenting a local portion $\boldsymbol{x}_l$ of length l_s from $\boldsymbol{x}$, the vector of nodes, and generating the local differentiating matrix $\boldsymbol{D}_l(\boldsymbol{x}_l)$

$$\boldsymbol{D}_l(\boldsymbol{x}_l) = \dot{\boldsymbol{P}}(\boldsymbol{x}_l)\,\boldsymbol{P}(\boldsymbol{x}_l) \tag{2.18}$$

for $(\boldsymbol{x}_l)$. At the start the top n_s rows of $\boldsymbol{D}_l$ are used to form the top portion of $\boldsymbol{D}$. In the band diagonal portion a $\boldsymbol{D}_l$ is generated at each point and the center row of $\boldsymbol{D}_l$ is transferred to the appropriate position in $\boldsymbol{D}$ and at the bottom, the last n_s rows of $\boldsymbol{D}_l$ are used.

A local differentiation matrix for $n = 10$ generated for the positive right half Tchebychev points with $l_s = 5$ is shown in Figure 2.7a. The condition number of the local differentiating matrix is two orders magnitude lower than for the global solution. The operator will be precise for functions up to degree $d = 4$; e.g., the deflection of a cantilever with uniform load is a quartic equation, so this operator would be sufficient.

2.6 Generating Admissible Functions

Admissible functions are functions which fulfil the constraints associated with a boundary- or initial-value problem. A homogeneous constraint requires the solution or some property of the solution, e.g., a derivative, to be zero at a specific location. The approach taken here is to define a set of constrained orthonormal basis functions $\boldsymbol{P_c}$ which span the complete space of all possible solutions which fulfil the constraints. Consequently, any admissible function $\boldsymbol{y}_a$ is a linear combination of the basis functions contained within $\boldsymbol{P_c}$,

$$\boldsymbol{y}_a = \boldsymbol{P_c}\,\boldsymbol{s} \tag{2.19}$$

where, $\boldsymbol{s}$ is the vector of coefficients; more formally $\boldsymbol{s}$ is the spectrum of $\boldsymbol{y}_a$ with respect to $\boldsymbol{P_c}$.

The constraints associated with initial- and boundary-value problems can all be formulated as linear operators. Once again, using the constraints for the cantilever for demonstration purposes: the first constraint $\boldsymbol{y}(0) = 0$ requires the first value of y to be zero, formulated as

$$\boldsymbol{y}_a^{\mathrm{T}}\,\boldsymbol{c_1} = 0 \tag{2.20}$$

with $\boldsymbol{c_1} = [1, 0, \ldots 0]^T$; the second constraint $\dot{\boldsymbol{y}}(0) = 0$ requires the first derivative at the beginning of the beam to be zero. Using $\boldsymbol{D}$, as derived in the previous section, the present constraint formulated as,

$$\boldsymbol{y}_a^{\mathrm{T}}\,\boldsymbol{D}^{\mathrm{T}}(1,:) = 0 \tag{2.21}$$

and consequently $\boldsymbol{c_2} = \boldsymbol{D}(1,:)^T$. Similarly, the third constraint, $\ddot{y}(0) = 0$, can be defined $\boldsymbol{c_3} = \boldsymbol{D}^2(end,:)^T$. All three constraints are concatenated to form a constraint matrix $\boldsymbol{C} = [\boldsymbol{c_1}, \boldsymbol{c_2}, \boldsymbol{c_3}]$. If any admissible function is to be generated, all the constrained basis functions must fulfil the constraints, i.e.,

$$\boldsymbol{P_c}^{\mathrm{T}}\,\boldsymbol{C} = \boldsymbol{0} \tag{2.22}$$

The task now is to generate a set of orthonormal constrained basis functions $\boldsymbol{P_c}$ from a complete orthonormal set of basis functions $\boldsymbol{P}$ such that the conditions $\boldsymbol{P_c}^{\mathrm{T}}\,\boldsymbol{C} = (\boldsymbol{P}\,\boldsymbol{X})^T\,\boldsymbol{C} = \boldsymbol{X}^{\mathrm{T}}\,\boldsymbol{P}^{\mathrm{T}}\,\boldsymbol{C} = \boldsymbol{0}$, $\boldsymbol{P_c}^{\mathrm{T}}\,\boldsymbol{P_c} = \boldsymbol{I}$ are fulfilled and the lower triangular portion of $\boldsymbol{X}$ is zero. If each admissible function fulfils the constraints, then $\boldsymbol{P}^{\mathrm{T}}\,\boldsymbol{C}$ spans the null space of $\boldsymbol{X}^{\mathrm{T}}$. Applying the QR-decomposition on the Matrix $\boldsymbol{P}^{\mathrm{T}}\,\boldsymbol{C}$ delivers the matrices $\boldsymbol{Q_1}$ and $\boldsymbol{R_1}$:

$$\boldsymbol{Q_1}\,\boldsymbol{R_1} = \boldsymbol{P}^{\mathrm{T}}\,\boldsymbol{C}. \tag{2.23}$$

The matrix $\boldsymbol{Q_1}$ can be split into two partitions, where p is the rank of C:

1. $\boldsymbol{Q}_{1s} = \boldsymbol{Q_1}\,(:, 1:p)$ containing the span of $\boldsymbol{P}^{\mathrm{T}}\,\boldsymbol{C}$
2. $\boldsymbol{Q}_{1n} = \boldsymbol{Q_1}\,(:, p+1:end)$, containing the null space of $\boldsymbol{P}^{\mathrm{T}}\,\boldsymbol{C}$

$\boldsymbol{R_1}$ is an upper triangular matrix, with zero entries in the bottom rows from $p+1$ until the end. The matrix $\boldsymbol{Q}_{1n}$ already contains the vectors of the null space of $\boldsymbol{P}^{\mathrm{T}}\boldsymbol{C}$, yet there is still need for further ordering of the matrix. This is done by using an RQ-decomposition:

$$\boldsymbol{R_2}\,\boldsymbol{Q_2} = \boldsymbol{Q}_{1n}. \tag{2.24}$$

$\boldsymbol{R_2}$ is the matrix $\boldsymbol{X}$ which is used for the computation of $\boldsymbol{P_c}$:

$$\boldsymbol{P_c} = \boldsymbol{P}\,\boldsymbol{X} \tag{2.25}$$

```
[Q, R] = qr( P' * C );
%
p = rank( C );
%
Q2 = Q(:,p+1:end);
%
[X,Q3]=rq( Q2 );
%
Pc = P * X;
%
```

Source Code 2.1: Code for the synthesis of the constrained basis functions.

The MATLAB code for the implementation of the constrained basis functions is presented in Source Code 2.1. This formulation is completely generic and valid for any boundary value problem. It delivers a complete set of admissible functions which span the space of all solutions to the boundary conditions, i.e., all possible admissible functions are linear combinations of $\boldsymbol{P_c}$.

The matrix $\boldsymbol{C}$ defines the null-space of the constrained basis functions $\boldsymbol{P_c}$. However, it does not uniquely define the basis functions. Given n points and $n_c = \operatorname{rank}\{C\}$ constraints, there is a solution space of dimensionality $n_s = n - n_c$. There are multiple different sets of orthonormal vectors which form a vector basis set for this space. The basis functions $\boldsymbol{P}$, from which $\boldsymbol{P_c} = \boldsymbol{P}\,\boldsymbol{X}$ are derived, provide the structure for the vector basis set. This is demonstrated using the example of a doubly held beam with the constraints $\boldsymbol{y}(0) = 0$, $\dot{\boldsymbol{y}}(0) = 0$, $\boldsymbol{y}(1) = 0$ and $\dot{\boldsymbol{y}}(1) = 0$. In Figure 2.8 two sets of constrained basis conditions are shown: the first derived from the discrete cosine basis functions and the second from the Gram polynomials. The two sets of basis functions are clearly different.

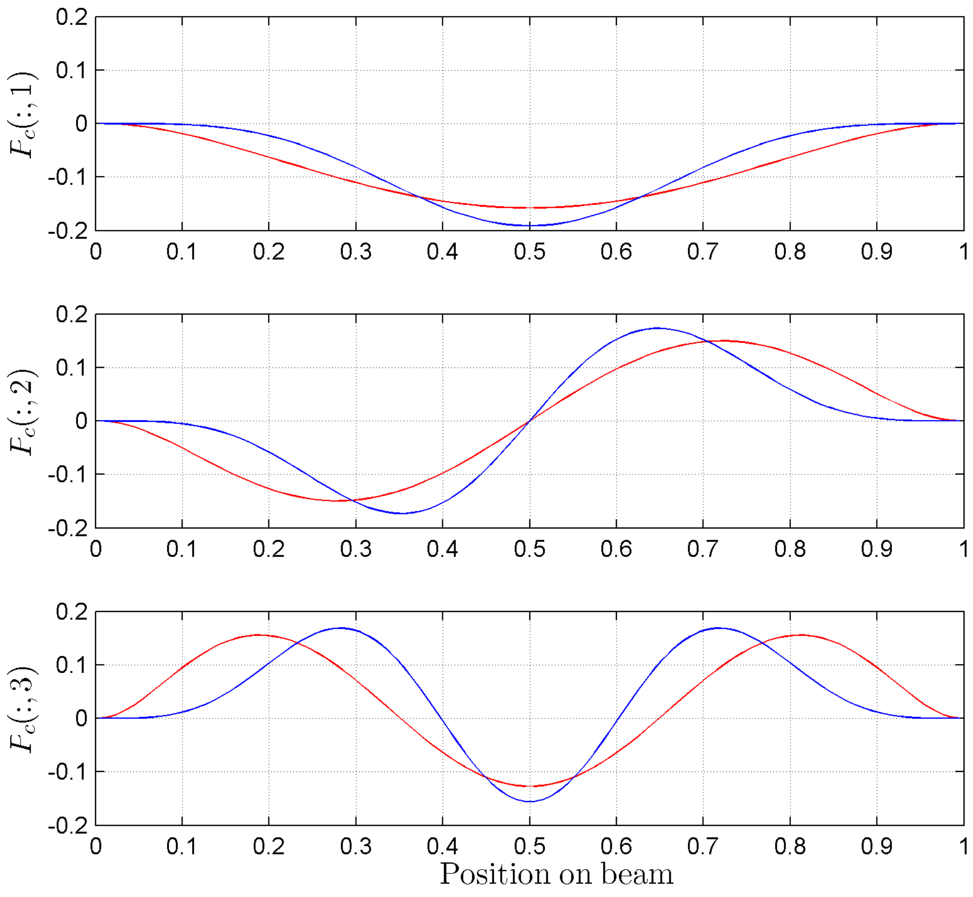


Figure 2.8: The first three constrained basis functions of $\boldsymbol{P_c}$ for a doubly held beam with the constraints, $\boldsymbol{y}(0) = 0$, $\dot{\boldsymbol{y}}(0) = 0$, $y(1) = 0$ and $\dot{\boldsymbol{y}}(1) = 0$. The basis functions in red are derived from the discrete cosine basis functions, in blue the basis functions from the Gram polynomials are shown.

2.7 Boundary Value Problems

A boundary value problem is characterized by a differential equation and one or more boundary conditions. For example, the computation of the deflection of a doubly held beam is such a problem: the Euler-Bernoulli differential equation describes the bending,

$$\frac{\mathrm{d}^2}{\mathrm{d}x^2}\left(EI\frac{\mathrm{d}^2 y(x)}{\mathrm{d}x^2}\right) - q(x) = 0, \tag{2.26}$$

where $y(x)$ is the deflection of the beam and $q(x)$ is the distributed load. The constraints on the ends are: $\boldsymbol{y}(0) = 0$, $\dot{\boldsymbol{y}}(0) = 0$, $\boldsymbol{y}(1) = 0$ and $\dot{\boldsymbol{y}}(1) = 0$. Consider the specific case when,

$$\frac{\mathrm{d}^4 y(x)}{\mathrm{d}x^4} - \lambda\, y(x) = 0. \tag{2.27}$$

The discrete equivalent of which is

$$\boldsymbol{D}^4\,\boldsymbol{y} - \lambda\,\boldsymbol{y} = 0. \tag{2.28}$$

This is, by definition, an eigenvector problem. The eigenvectors of $\boldsymbol{D}^4$ are the bending modes, which unfortunately cannot be computed directly. As is also the case with Sturm-Liouville type problems, where the direct computation delivers unstable results[6]. However, the concept of constrained basis functions can be applied to this problem and yield stable results. The solution to the problem is a linear combination of a set of constrained basis functions $\boldsymbol{y} = \boldsymbol{P_c}\,\boldsymbol{s}$, where $\boldsymbol{s}$ is the spectrum with respect to $\boldsymbol{P_c}$. Substituting into Equation 2.28 yields,

$$\left\{\boldsymbol{D}^4\,\boldsymbol{P_c} - \lambda\,\boldsymbol{P_c}\right\}\boldsymbol{s} = 0. \tag{2.29}$$

Multiplying by $\boldsymbol{P_c}^{\mathrm{T}}$ yields,

$$\left\{\boldsymbol{P_c}^{\mathrm{T}}\,\boldsymbol{D}^4\,\boldsymbol{P_c} - \lambda\,\boldsymbol{I}\right\}\boldsymbol{s} = 0. \tag{2.30}$$

Since $\boldsymbol{P_c}$ is orthonormal, the eigenvalues of $\boldsymbol{P_c}^{\mathrm{T}}\,\boldsymbol{D}^4\,\boldsymbol{P_c}$ are the same as for $\boldsymbol{D}^4$. Given $\boldsymbol{s}$ an eigenvector of $\boldsymbol{P_c}^{\mathrm{T}}\,\boldsymbol{D}^4\,\boldsymbol{P_c}$, the eigenvectors $\boldsymbol{y}_e$ of $\boldsymbol{D}^4$ can be computed as,

$$\boldsymbol{y}_e = \boldsymbol{P_c}\,\boldsymbol{s}. \tag{2.31}$$

That is, $\boldsymbol{s}$ is the spectrum of the eigen-functions with respect to the constrained basis functions. In Section 2.6 it was shown that the constrained basis functions are not unique. Consequently it is now necessary to determine the best set of constrained basis functions for a specific boundary value problem. The spectra of the eigenfunctions of the above example are given in Tables 2.1 and 2.2. From this data it is clear that fewer constrained Gram

[6]This is easily verified by generating the matrix $\boldsymbol{D}$ in MATLAB and computing directly the eigenvalues and eigenvectors of $\boldsymbol{D}^4$.

	Modes		
	1	2	3
1	−0.99989	0	−0.014991
2	0	0.9981	0
3	−0.01507	0	0.99075
4	0	0.061566	0
5	−0.00043537	0	0.1343
6	0	−0.0034509	0
7	0	0	−0.012428
8	0	0	0
9	0	0	−0.00068759
10	0	0	0

Table 2.1: The first 10 coefficients of the spectrum of the first 3 eigenfunctions of Equation 2.29 with respect to the constrained Gram polynomials.

	Modes		
	1	2	3
1	0.97419	0	0.22546
2	0	−0.93801	0
3	−0.20827	0	0.87975
4	0	−0.30903	0
5	−0.07647	0	0.36243
6	0	−0.13385	0
7	−0.035054	0	0.1746
8	0	−0.067349	0
9	−0.01823	0	0.093476
10	0	−0.037233	0

Table 2.2: The first 10 coefficients of the spectrum of the first 3 eigenfunctions of Equation 2.29 with respect to the constrained discrete cosine functions.

polynomials are required to describe the bending modes than for the constrained discrete cosine basis. This will enable a higher degree of regularization without loss of accuracy. Consequently, for this example, the constrained Gram polynomials are the basis of choice.

2.8 Inverse Boundary Value Problems

Inverse boundary value problems occur when a measurement is performed on the behaviour of an inverse system described by a boundary value problem. Given the measurements, one wishes to determine their causes or some other properties of the system. Consider the monitoring of constrained structures using strings of inclinometers; the deflection $\boldsymbol{y}(x)$ of the structure from the measured local gradients $\dot{\boldsymbol{y}}_{\boldsymbol{m}}(x)$ is to be computed, while fulfilling the boundary conditions. One proceeds to the solution as follows.

2.8.1 Determine the constrained basis functions

The methods presented in Sections 2.6 and 2.7 are used to determine the most suitable set of constrained basis functions $\boldsymbol{P_c}$ for the boundary value problem being considered.

2.8.2 Inverse problem as a least squares problem

The measurement problem is now formulated as a least squares vector problem. The solution is formulated as a linear combination of the constrained basis functions, $\boldsymbol{y} = \boldsymbol{P_c}\,\boldsymbol{s}$ and the relationship between the desired result $\boldsymbol{y}$ and the vector of measurement values $\dot{\boldsymbol{y}}_{\boldsymbol{m}}$ is formulated,

$$\begin{aligned} \dot{\boldsymbol{y}}_{\boldsymbol{m}} &= \boldsymbol{D}\,\boldsymbol{y}, && (2.32)\\ &= \boldsymbol{D}\,\boldsymbol{P_c}\,\boldsymbol{s}. && (2.33) \end{aligned}$$

Solving for $\boldsymbol{s}$,

$$\boldsymbol{s} = \{\boldsymbol{D}\,\boldsymbol{P_c}\}^+\ \dot{\boldsymbol{y}}_{\boldsymbol{m}} \tag{2.34}$$

and evaluating $\boldsymbol{y}$ yields a solution to the boundary value problem,

$$\boldsymbol{y} = \boldsymbol{P_c}\ \{\boldsymbol{D}\,\boldsymbol{P_c}\}^+\ \dot{\boldsymbol{y}}_{\boldsymbol{m}}. \tag{2.35}$$

Now defining $\boldsymbol{L} \triangleq \boldsymbol{P_c}\ \{\boldsymbol{D}\,\boldsymbol{P_c}\}^+$ yields,

$$\boldsymbol{y} = \boldsymbol{L}\,\dot{\boldsymbol{y}}_{\boldsymbol{m}}. \tag{2.36}$$

This is a very significant result. It states that the solution to the inverse problem of performing measurements on a system described by a boundary value problem can be solved as a linear matrix operator $\boldsymbol{L}$. This operator is invariant given a specific measurement problem and can be computed a-priori, so that at run-time it is only necessary to perform one matrix multiplication. The worst case number of FLOPS required to perform this computation for n sensors is,

$$W(n) = (2n - 1)\ n. \tag{2.37}$$

For example, given a string of $n = 31$ sensors $W(n) = 1891$ FLOPS are required to solve the inverse problem. This is a very modest number of operations, but more importantly it is constant and known a-priory. This makes this method suitable for real-time embedded computations.

2.9 Regularization

"In general terms, *regularization* is the approximation of an ill-posed problem by a family of neighbouring well-posed problems." [8]. In any real world system uncertainties are present and need to be taken into account. When modelling real world applications it is necessary to find a good approximation to a vector $\boldsymbol{x}$ satisfying an approximation equation

$$\boldsymbol{A}\,\boldsymbol{x} \approx \boldsymbol{y}, \tag{2.38}$$

where $\boldsymbol{y}$ is the measurement result perturbed by noise. $\boldsymbol{A}$ may be a discrete approximation, or a matrix of basis functions at given points. This kind of problem arises in many fields of science and the vector $\boldsymbol{x}$ has a solution of the form

$$\boldsymbol{x} = \boldsymbol{A}^{+}\,\boldsymbol{y}, \tag{2.39}$$

where $\boldsymbol{A}^{+}$ is the Moore Penrose Pseudo Inverse of $\boldsymbol{A}$. It ensures a solution which is of minimum norm. However, it is only one of many different solutions. The function given in Equation (2.39) is a solution using the *least squares method*, which is a so called *regularization techniques*. *Regularization techniques* are used to obtain meaningful solution estimates for such problems [25]. This form of regularization is a very basic method and a standard approach to the approximation solution of overdetermined systems. It means that the overall solution minimizes the sum of squared residuals, where a residual is the difference between an observed and the fitted value:

$$\underset{y}{min}\, E = ||\boldsymbol{A}\,\boldsymbol{x} - \boldsymbol{y}||^2, \tag{2.40}$$

where $||\cdot||$ is the Euclidean norm. Another very typical method of regularization is the *Tikhonov regularization*, after *Andrey Nikolayevich Tikhonov*. In order to be able to give preference to a certain solution with desirable properties, the regularization term is included in the minimization:

$$E = ||\boldsymbol{A}\,\boldsymbol{x} - \boldsymbol{y}||^2 + ||\alpha\,\boldsymbol{\Gamma}\,\boldsymbol{x}||^2, \tag{2.41}$$

where $\boldsymbol{\Gamma}$ is the *Tikhonov matrix* that can be chosen to suit the current problem. In most cases, though, it's an identity matrix. The solution to this regularization can be written out as seen in Equation (2.42).

$$\boldsymbol{x} = (\boldsymbol{A}^{\mathrm{T}}\,\boldsymbol{A} + \alpha\,\boldsymbol{\Gamma}^{\mathrm{T}}\,\boldsymbol{\Gamma})^{-1}\,\boldsymbol{A}^{\mathrm{T}}\,\boldsymbol{y} \tag{2.42}$$

These are the two most important kinds of *regularization*. There are many methods which can be used to implement *regularization*. In this thesis we limit *regularization* to the application of the least squares differences method as well as the use of admissible functions. To do so, our desired solution $\boldsymbol{y}$ is split into a homogeneous and a particular solution; the particular solution containing information of constraints and the homogeneous solution consisting of a matrix of basis functions and its spectrum (see Equation (2.55)).

2.10 Covariance propagation for linear operators

The covariance Λ_y of a vector $\boldsymbol{y}$ is defined as,

$$\Lambda_y \triangleq \{\boldsymbol{y} - E(\boldsymbol{y})\} \{\boldsymbol{y} - E(\boldsymbol{y})\}^T, \tag{2.43}$$

where $E(\boldsymbol{y})$ is the expected value of $\boldsymbol{y}$. If $\boldsymbol{y} = \boldsymbol{L}\,\boldsymbol{x}$, then,

$$\begin{aligned} \Lambda_y &= \{\boldsymbol{L}\,\boldsymbol{x} - E(\boldsymbol{L}\,\boldsymbol{x})\} \{\boldsymbol{x} - E(\boldsymbol{L}\,\boldsymbol{x})\}^T, & (2.44)\\ &= \boldsymbol{L}\,\{\boldsymbol{x} - E(\boldsymbol{x})\} \{\boldsymbol{x} - E(\boldsymbol{x})\}^T\,\boldsymbol{L}^{\mathrm{T}} & (2.45)\\ &= \boldsymbol{L}\,\Lambda_x\,\boldsymbol{L}^{\mathrm{T}}, & (2.46) \end{aligned}$$

where Λ_x is the covariance of the vector $\boldsymbol{x}$. Consequently, the covariance propagation can be computed for all the solutions presented in this paper, because they are all implemented as linear operators.

2.11 Mathematical Framework to the Solution of a Boundary Value Problem

The problem encountered is the availability of the inclinometer measurement and the wish to find the original curve's deflection knowing the position, as well as other boundary values, at some points along the curve. This means, a special class of inverse boundary value problems is treated. The system being monitored, typically a structure, is modelled by a boundary value problem,

$$\boldsymbol{L}\,\boldsymbol{y} - \boldsymbol{\lambda}\,\boldsymbol{y} = 0 \quad \text{given} \quad \boldsymbol{C}\,\boldsymbol{y} = d.$$

Whereby, $\boldsymbol{L}$ is the linear differential operator for the system, $\boldsymbol{\lambda}$ represents the eigenvector of $\boldsymbol{y}$ and $\boldsymbol{C}$ is the constraint matrix of the form

$$\boldsymbol{C} = [\boldsymbol{c_1}, \boldsymbol{c_2}, \boldsymbol{c_3}, ..., \boldsymbol{c_n}], \tag{2.47}$$

$\boldsymbol{c_i}$ representing a vector of constraints. The vector $\boldsymbol{d}$ represents the boundary values, i.e.

$$\boldsymbol{c_i}^{\mathrm{T}}\,\boldsymbol{y} = \boldsymbol{d_i}. \tag{2.48}$$

The system considered in this thesis is being monitored by a vector of n sensors. The i^{th} sensor is at the position x_i, and yields the measurement value $\dot{y}_{mi}$, which is perturbed. The concatenation of the measurement values $\dot{y}_{mi}$ forms the measurement vector $\dot{\boldsymbol{y}}_{\boldsymbol{m}}{}^{\mathrm{T}} = [\dot{y}_{m1}, \dot{y}_{m2}, ..., \dot{y}_{mn}]$ and the position vector $\boldsymbol{x}^{\mathrm{T}} = [x_1, x_2, ..., x_n]$. The relationship between the measurement vector $\dot{\boldsymbol{y}}_{\boldsymbol{m}}$ and the dependent variable $\boldsymbol{y}$ is also governed by a differential equation

$$\boldsymbol{L}_{\boldsymbol{m}}\,\boldsymbol{y} - (\dot{\boldsymbol{y}}_{\boldsymbol{m}} + \boldsymbol{\epsilon}) = 0, \tag{2.49}$$

where $\boldsymbol{\epsilon}$ represents the perturbation obtained due to noise. Measurements with inclinometers are a special case of this type of problem, where $\boldsymbol{L}_{\boldsymbol{m}} = \boldsymbol{D}$, the differential matrix. Therefore,

$$\boldsymbol{D}\,\boldsymbol{y} - (\dot{\boldsymbol{y}}_{\boldsymbol{m}} + \boldsymbol{\epsilon}) = 0. \tag{2.50}$$

The task now is to determine the optimal $\boldsymbol{y}$. As will be seen, there are a number of figures of merit with respect to which optimality can be determined. The simplest solution to the measurement problem is to solve the differential equation

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \dot{\boldsymbol{y}}_{\boldsymbol{m}} \tag{2.51}$$

Consequently,

$$\boldsymbol{y}(x) = \int \dot{\boldsymbol{y}}_{\boldsymbol{m}}\,\mathrm{d}x + c_0, \tag{2.52}$$

c_0 being the constant of integration. As we have a discrete number of sensors, a discrete solution to integration must be formed. The problem can be formulated as a least squares problem:

$$E = \overset{min}{\underset{y}{}}\,||\boldsymbol{D}\,\boldsymbol{y} - (\dot{\boldsymbol{y}}_{\boldsymbol{m}} + \boldsymbol{\epsilon})||_2^2. \tag{2.53}$$

The solution to this problem is well known [12]:

$$\boldsymbol{y} = \boldsymbol{D}^{+}(\dot{\boldsymbol{y}}_{\boldsymbol{m}} + \boldsymbol{\epsilon}) + null\{\boldsymbol{D}\}\,\boldsymbol{\alpha}. \tag{2.54}$$

For the specific problem at hand, it is known that $null\{\boldsymbol{D}\} = 0$, which is why this term is neglected in the calculations in this thesis. In order to simplify calculations, the perturbation $\boldsymbol{\epsilon}$ is left aside as a vector for itself. Instead it is assumed that the vector of the inclinometer measurement $\dot{\boldsymbol{y}}_{\boldsymbol{m}}$, is, as any real world measurement, perturbed.

This method takes no advantage of the information contained in the equation for the boundary value problem, which could be used for regularization. It may be used as a base for comparison of other solutions. In order to include perturbation and add regularization to the problem, we use admissible functions. The admissible functions of the BVP used in our measurements are defined as a combination of the particular and the homogeneous solution, $\boldsymbol{y_c}$ and $\boldsymbol{y_h}$ respectively, i.e.,

$$\boldsymbol{y} = \boldsymbol{y_c} + \boldsymbol{y_h}. \tag{2.55}$$

Here $\boldsymbol{y}_c$ represents a truly arbitrary function fulfilling the given boundary conditions, here: $\boldsymbol{y_1}$ and $\boldsymbol{y_n}$. As all the non significant entries vanish during the matrix multiplication, the vector is freely selectable, with the only exception of the entries at the position of the known boundary values. Yet it seems easiest and most lucid to use a vector full of zeros, except for the entries at both, the beginning. and the endpoints and with the information of these values available, the vector $\boldsymbol{y_c}$ can be predefined as

$$\boldsymbol{y_c} = \begin{bmatrix} y_1 \\ 0 \\ \vdots \\ 0 \\ y_n \end{bmatrix}. \tag{2.56}$$

By splitting up the vector $\boldsymbol{y}$ into a particular and a homogeneous part, we can add regularization to our problem, which is obtained using the homogeneous solution $\boldsymbol{y_h}$. $\boldsymbol{y_h}$ fulfils the homogeneous boundary conditions and is described by the basis functions $\boldsymbol{P_c}$ and the spectrum $\boldsymbol{s}$. Now we can rewrite Equation (2.55):

$$\boldsymbol{y} = \boldsymbol{y_c} + \boldsymbol{P_c}\, \boldsymbol{s}. \tag{2.57}$$

In this thesis, I present two ways of solving this problem. The derivation of the solution, as well as their performance in reconstruction is presented below.

2.12 Deriving Operators for Reconstruction

While working on this thesis, two solution approaches to reconstruction were derived and tested. Measurements performed using inclinometers deliver perturbed values for the spatial gradient at specific points $\boldsymbol{x_i}$. Many structures can be modelled using these measurements and curve reconstruction. In the special case of a cantilever, the deflection can be modelled by a fourth order polynomial, as defined by the *Euler-Bernoulli-Equation*.

The first approach taken for the solution of the problem was the reconstruction using error minimization in the gradient space of the curve, as both, the measurement done by the inclinometers as well as the deflection of the beam lie in the gradient space.

This does not accompany a mean error in the space of the curve itself, which brings us to the second solution. The second solution is a variational solution minimizing the error in the space of the curve.

The function $\boldsymbol{y}$ used for demonstration of the reconstruction in this section is a function of arbitrary values between zero and one (see Figure (2.9)). It is assumed that the first and the last values of y are known and can be used for reconstruction purposes. This means that the constraints placed on each reconstruction for the current problem are positional constraints. The particular solution $\boldsymbol{y_c}$ is defined as explained in Equation (2.56). In the following examples, only positional constraints at the beginning and the end point were defined. Hence, the constraint matrix $\boldsymbol{C}^{\mathrm{T}}$ is defined as follows:

$$\boldsymbol{C}^{\mathrm{T}} = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}; \tag{2.58}$$

The constraints matrix is a combination of all constraint vectors. The constraint vectors contain information about the position of the constraint as well as of the nature of the constraint itself. It is necessary to note that the constraints placed onto the reconstruction can be of varying nature. It can be a positional constraint, which is a constraint in the space of the curve, as well as an inclination constraint, which is then in the space of the first derivative. A constraint can be defined in any space desired. In Table 2.2 the MATLAB source code of a constraint matrix with positional constraints on the end points, as well as a constraint in the first derivative on the first point is listed. Furthermore the code for the implementation of the particular solution is given.

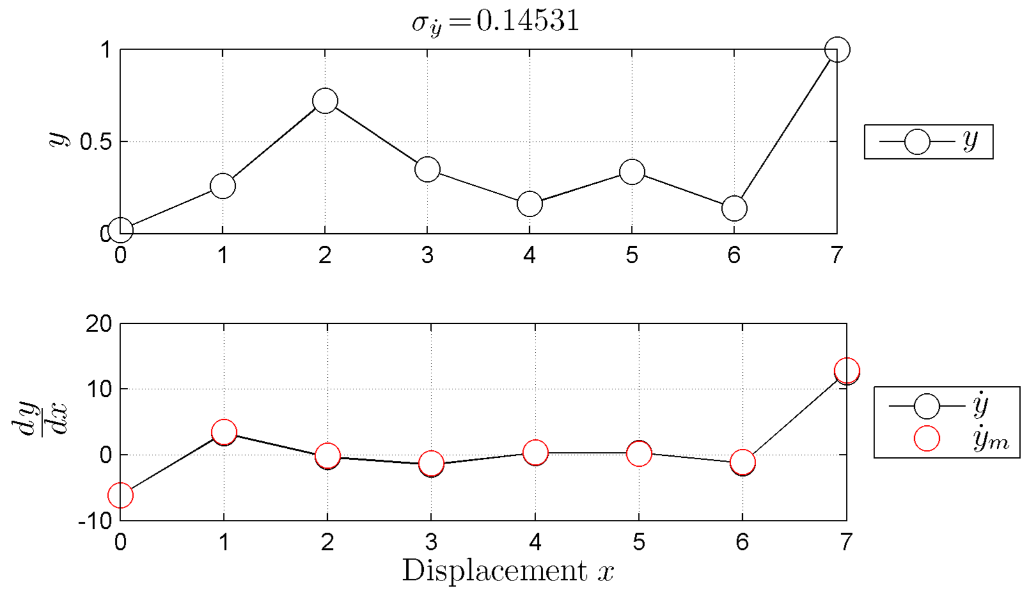


Figure 2.9: The simulated curves $\boldsymbol{y}$, $\dot{\boldsymbol{y}}$ and $\dot{\boldsymbol{y}}_{\boldsymbol{m}}$. The values for $\boldsymbol{y}$ are simulated using the 'rand' function in MATLAB. The values for the vector $\dot{\boldsymbol{y}}$ are retrieved by applying the differential operator $\boldsymbol{D}$ onto the vector $\boldsymbol{y}$. $\dot{\boldsymbol{y}}_{\boldsymbol{m}}$ is created by adding random, normally distributed noise of the magnitude of 4% of $\dot{\boldsymbol{y}}$ to itself.

```
%
%Define the constraint vectors
c1 = zeros (noPts , 1);
c1 (end) = 1;
c2 = Dc(1,:);
%
%Put the two vectors into a matrix
C = [c1, c2];
%
%define the particular solution
yc = zeros (noPts, 1);
yc (1) = y (1);
yc (end) = y (end);
%
```

Source Code 2.2: Code for the synthesis of the constraints.

2.12.1 Solution 1 - solving using admissible functions

In this section the solution of the equation using admissible functions for the BVP is investigated. This approach yields a regularization based on spectral regularization.

The functional used in Solution 1 for error minimization lies in the gradient space, i.e. in the space of the deflection of the curve. The problem is defined by a discrete number of points, as well as a discrete function. Writing out the discrete formulation for the error functional yields,

$$E = ||\dot{\boldsymbol{y}}_{\boldsymbol{m}} - \boldsymbol{D}\ (\boldsymbol{y_c} + \boldsymbol{P_c}\, \boldsymbol{s_1})\, ||_2^2. \tag{2.59}$$

$\dot{\boldsymbol{y}}_{\boldsymbol{m}}$ represents the inclinometer data from the measurement, $\boldsymbol{D}$ is the differentiating matrix, $\boldsymbol{y_c}$ is the constraint vector, which, in the special case of the cantilever with constraints at the both ends of the beam, contains information only in the first and the last entry. $\boldsymbol{P_c s_1}$ represents the desired function y, $\boldsymbol{P_c}$ being the admissible constrained basis and $\boldsymbol{s_1}$ the unknown vector. $\boldsymbol{s_1}$ is the only unknown in this equation. In order to find a constant vector $\boldsymbol{s_1}$ minimizing the error functional above, we derive it with respect to $\boldsymbol{s_1}$ and set it equal to zero. This yields,

$$\boldsymbol{s_1} = (\boldsymbol{D}\,\boldsymbol{P_c})^+\ (\dot{\boldsymbol{y}}_{\boldsymbol{m}} - \boldsymbol{D}\,\boldsymbol{y_c}) + null\{\boldsymbol{D}\,\boldsymbol{P_c}\,\boldsymbol{\alpha}\}. \tag{2.60}$$

The vector $\boldsymbol{s_1}$ is computed using Equation 2.60 and then back substituted into Equation (2.57):

$$\boldsymbol{y_{r1}} = \boldsymbol{y_c} + \boldsymbol{P_c}\,\boldsymbol{s_1}, \tag{2.61}$$

The results of the reconstruction with Solution 1 using the derivative values, representing the measured inclinometer data from Figure 2.9 is presented in Figure 2.10. It can be seen that the reconstruction for this arbitrary curve is valid. The mean error of the reconstruction is only 4% for this example.

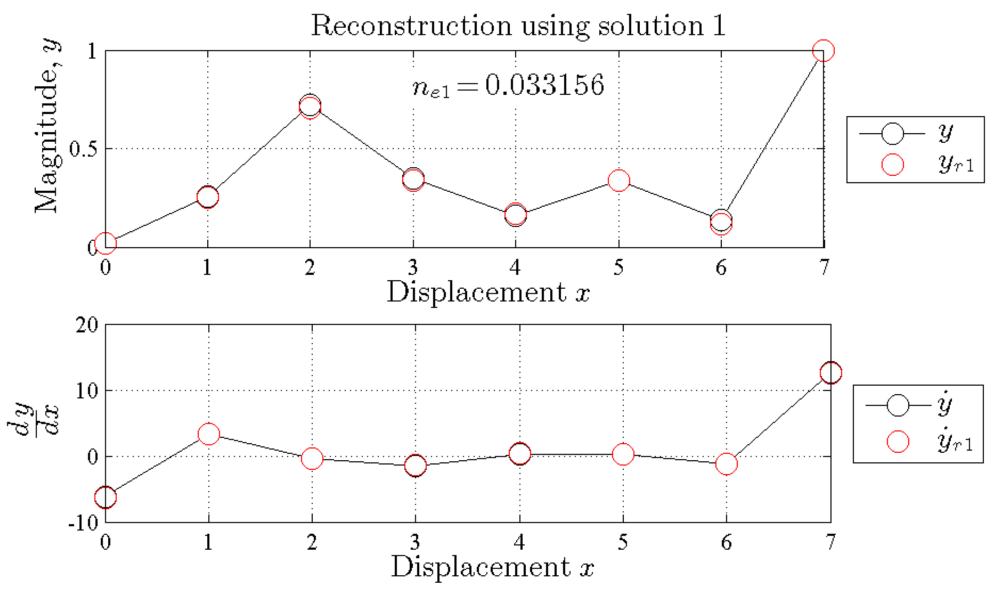


Figure 2.10: Reconstruction of curve $\boldsymbol{y}$ using Solution 1. $\boldsymbol{y}$ is the arbitrary function which is to be reconstructed of its calculated derivative $\dot{\boldsymbol{y}}$ including noise, as displayed in Figure 2.9. $\boldsymbol{y_{r1}}$ is the reconstruction of the noisy function $\dot{\boldsymbol{y}}_{\boldsymbol{m}}$ and $\dot{\boldsymbol{y_{r1}}}$ is the newly calculated derivative of the reconstructed curve $\boldsymbol{y_{r1}}$. n_{e1} is the mean error of the reconstruction, compared to the original function without noise.

2.12.2 Variational Solutions

The solution presented to Equation (2.59) minimizes the difference between the square of the computed derivative and the measured gradient, i.e. the solution lies in the gradient space. The properties of the least squares (LS) algebraic solution will ensure that the mean error in the gradient is zero. However, this does not imply that the mean error in the curve is zero. Consequently, the solution for the curve could be biased. The aim now is to investigate an alternative solution which perform the LS approximation in the space of the curves.

There is a further issue which may now need to be considered; there are two common methods of applying inclinometers:

1. the inclinometers are mounted directly onto a structure which is deformed. In this case the measurement is truly a gradient measurement;
2. the inclinometers are mounted onto a chain of rods. In this case the measurement is a peace wise linear approximation to the gradient (first derivative). Such solutions are commonly used for monitoring ground movement in bore holes [22].

Derivation of the Variational Solution

It is important to take the different differentiating matrices for each of these cases into account. Let us define:

$\boldsymbol{D}$:	generic differentiating matrix, i.e. a discrete approximation of a differential operator
$\boldsymbol{D_c}$:	continuous differential operator (local gradient)
$\boldsymbol{D_d}$:	piece wise linear differential operator (linear approximation)

It is now also necessary to determine a method of computing a discrete indefinite integral as required to solve the simplest differential equation,

$$\frac{\mathrm{d}}{\mathrm{d}x} y(x) = m, \tag{2.62}$$

m being the measured data. Integrating yields,

$$y(x) = \int \{m\} \, \mathrm{d}x + c_0, \tag{2.63}$$

c_0 being the constant of integration. The error can be formulated as a discrete LS functional,

$$E = \min_y ||\boldsymbol{D}\, \boldsymbol{y} - \dot{\boldsymbol{y}}_{\boldsymbol{m}}||_2^2, \tag{2.64}$$

where $\dot{\boldsymbol{y}}_{\boldsymbol{m}}$ is the discrete equivalent of m. The solution to the above equation is known to be

$$\boldsymbol{y} = \boldsymbol{D}^+ \, \dot{\boldsymbol{y}}_{\boldsymbol{m}} + \mathrm{null}\{\boldsymbol{D}\}\, \alpha, \tag{2.65}$$

$\mathrm{null}\{\boldsymbol{D}\}\, \alpha$ being the discrete constant vector of integration. Now formulating the problem of reconstructing the curve from measured local gradients. The measured curve can be reconstructed as:

$$y_m(x) = \int \{m\} \, \mathrm{d}x + c_0. \tag{2.66}$$

Note, this is an indefinite integral. The difference to the continuous curve is then

$$e(x) = y(x) - y_m(x) \tag{2.67}$$

The functional which now describes the total error is,

$$E(x) = \int_a^b \{e^2\}\, \mathrm{d}x + c_1. \tag{2.68}$$

Back substituting in e from Equation (2.67) yields,

$$= \int_a^b \left[\int m\, \mathrm{d}x + c_0 - y(x) \right]^2 \mathrm{d}x + c_1. \tag{2.69}$$

This is the equation which needs to be approximated by a discrete equivalent. The discrete equivalent $\boldsymbol{e}$ for e is:

$$\boldsymbol{e} = \boldsymbol{y} - \left(\boldsymbol{D}^{+}\, \dot{\boldsymbol{y}}_{\boldsymbol{m}} + \mathrm{null}\{\boldsymbol{D}\}\, \alpha \right). \tag{2.70}$$

Furthermore,

$$\boldsymbol{e}^2 = \boldsymbol{e} \circ \boldsymbol{e}, \tag{2.71}$$

the symbol $\circ$ denotes the Hadamard Product. It is now necessary to find a discrete equivalent to the integral over $\boldsymbol{e}^2$. Given the integrator matrix $\boldsymbol{A_c} = \boldsymbol{D_c}^{+}$, we may define the integral over the complete curve as,

$$E = \left[\boldsymbol{A_c}(end, :) - \boldsymbol{A_c}(1, :) \right] \boldsymbol{e}^2. \tag{2.72}$$

Defining

$$\boldsymbol{a} \triangleq \left[\boldsymbol{A_c}(end, :) - \boldsymbol{A_c}(1, :) \right]^T \tag{2.73}$$

yields,

$$E = \boldsymbol{a}^{\mathrm{T}} \boldsymbol{e} \circ \boldsymbol{e}. \tag{2.74}$$

Unfortunately, the Hadamard operator is a non-analytic function so that differential calculus is not directly applicable for finding an optimal solution. However, Equation (2.74) can be reformulated as

$$E = \boldsymbol{e}^{\mathrm{T}} \boldsymbol{A}\, \boldsymbol{e}, \tag{2.75}$$

whereby $\boldsymbol{A} = diag(\boldsymbol{a})$. This is a weighted least squares problem and true for any of the differential matrices $\boldsymbol{D}$, $\boldsymbol{D_c}$ or $\boldsymbol{D_d}$ presented above. Further on in this thesis, the calculations given above will be applied to measurements performed on a laboratory set-up. The laboratory consists of a beam which will simulate various deflections. These deflections will be measured both by inclinometers and, for verification purposes, by the Light Sectioning Method. This will give us the opportunity to validate the various solutions presented here. However, in the laboratory set-up, only a gradient measurement was performed. The piece wise linear approximation will not be dealt with in the practical part. As we will not apply the piece wise linear approximation to the practical set-up, the solutions will be derived for the gradient measurement.

Variational solution with spectral regularization

The operator described in this section will perform the reconstruction using minimization in the space of the curve $\boldsymbol{y}$, in order for the mean error to be zero in the space of the curve instead of in the space of the gradient. The unknown of the function,

$$\boldsymbol{y_{r2a}} = \boldsymbol{y_c} + \boldsymbol{P_c}\, \boldsymbol{s_{2a}}, \tag{2.76}$$

to which we seek a solution, is the vector $\boldsymbol{s_{2a}}$. In equation (2.65) $\boldsymbol{y}$ was defined as $\boldsymbol{y} = \boldsymbol{D}^+\, \boldsymbol{m} + null\{\boldsymbol{D}\}\, \alpha$. For the calculation in this section we will define $\boldsymbol{m}$ to be the measured data $\dot{\boldsymbol{y}}_{\boldsymbol{m}}$. $null\{\boldsymbol{D}\}\, \alpha$ is $\boldsymbol{1}\, \boldsymbol{\beta_2}$ and $\boldsymbol{D}$ is replaced by the newly defined continuous differential operator $\boldsymbol{D_c}$. This brings us to the equation

$$\boldsymbol{y} = \boldsymbol{D_c}^+ \dot{\boldsymbol{y}}_{\boldsymbol{m}} + \boldsymbol{1}\, \boldsymbol{\beta_2} \tag{2.77}$$

To minimize the error of the difference between $\boldsymbol{y}$ and $\boldsymbol{y_{r2a}}$ we define the following functional E:

$$E = ||\left(\boldsymbol{D_c}^+ \dot{\boldsymbol{y}}_{\boldsymbol{m}} + \boldsymbol{1}\boldsymbol{\beta_2}\right) - (\boldsymbol{y_c} + \boldsymbol{P_c}\boldsymbol{s}_2)||_2^2, \tag{2.78}$$

where $\boldsymbol{D_c}^+$ is the Moore Penrose pseudo-inverse of the differentiating matrix $\boldsymbol{D_c}$. To find $\boldsymbol{s_2}$ minimizing the error functional at hand, the error functional needs to be derived with respect to the unknowns in this equation. There are two unknowns in the equation, which is why the solution is not as trivial as in the previous section. Two different approaches of deriving the two functionals are going to be dealt with in detail:

1. Solution 2a, using bloc matrices for the derivation;
2. Solution 2b, deriving the functional by either unknown.

Solution 2a - Bloc Matrices Now we use the knowledge of solving an equation of the form

$$E = \|\left(\boldsymbol{A} - \boldsymbol{B}\,\boldsymbol{\gamma}\right)\|_2^2. \tag{2.79}$$

$\boldsymbol{\gamma}$ minimizing the error of Equation (2.79) is known to be $\boldsymbol{\gamma} = \boldsymbol{B}^+\,\boldsymbol{A}$. The aim in this section is to apply this knowledge to the functional from Equation (2.78). As there is not only one, but two unknowns, both are combined in a partitioned vector $\boldsymbol{\gamma_{2a}}$:

$$\boldsymbol{\gamma_{2a}} = \begin{bmatrix} -\boldsymbol{\beta_{2a}} \\ \boldsymbol{s_{2a}} \end{bmatrix} \tag{2.80}$$

Now we rewrite the functional of Equation (2.78), where B is defined as $\boldsymbol{B} = \begin{bmatrix} \mathbf{1} & \boldsymbol{P_c} \end{bmatrix}$:

$$E = \|\left(\left(\boldsymbol{D_c}^+\dot{\boldsymbol{y}}_{\boldsymbol{m}} - \boldsymbol{y_c}\right) - \boldsymbol{B}\boldsymbol{\gamma_{2a}}\right)\|_2^2, \tag{2.81}$$

On the left hand side of the functional $(\boldsymbol{D_c}^+\dot{\boldsymbol{y}}_{\boldsymbol{m}} - \boldsymbol{y_c})$ we have known matrices, on the right hand side we have a known matrix $\boldsymbol{B}$ multiplied by an unknown constant vector $\boldsymbol{\gamma_{2a}}$, for which we seek a solution. The solution to such a problem is known to be:

$$\boldsymbol{\gamma_{2a}} = \begin{bmatrix} \mathbf{1} & \boldsymbol{P_c} \end{bmatrix}^+ \left(\boldsymbol{D_c}^+\,\dot{\boldsymbol{y}}_{\boldsymbol{m}} - \boldsymbol{y_c}\right). \tag{2.82}$$

As β_{2a} is a constant, and therefore only the first entry of the vector $\boldsymbol{\gamma_{2a}}$, $\boldsymbol{s_{2a}}$ can be computed as

$$\boldsymbol{s_{2a}} = \boldsymbol{\gamma_{2a}}\left(2 : end\right). \tag{2.83}$$

Now the reconstruction can be calculated according to Equation (2.76).

Solution 2b - Seperate Derivation The solution presented in Section 2.12.2 is a valid solution for the reconstruction problem at hand. In order to make the calculation more efficient, a different solution approach was considered. The functional is

$$E = \|\left(\boldsymbol{D_c}^+\dot{\boldsymbol{y}}_{\boldsymbol{m}} + \mathbf{1}\boldsymbol{\beta_{2b}}\right) - \left(\boldsymbol{y_c} + \boldsymbol{P_c}\boldsymbol{s_{2b}}\right)\|_2^2. \tag{2.84}$$

At first the function's derivative is computed with respect to $\boldsymbol{s_{2b}}$ and $\boldsymbol{\beta_{2b}}$ and set equal to zero, which leads to Equations (2.85) and (2.86), respectively:

$$\boldsymbol{\beta_{2b}} = \frac{1}{n}\mathbf{1}^{\mathrm{T}}\left(\boldsymbol{y_c} + \boldsymbol{P_c}\boldsymbol{s_{2b}} - \boldsymbol{D_c}^+\dot{\boldsymbol{y}}_{\boldsymbol{m}}\right) \tag{2.85}$$

$$\boldsymbol{s_{2b}} = \boldsymbol{P_c}^T\left(\boldsymbol{D_c}^+\dot{\boldsymbol{y}}_{\boldsymbol{m}} - \boldsymbol{y_c} + \mathbf{1}\boldsymbol{\beta_{2b}}\right). \tag{2.86}$$

Substituting (2.86) into Equation (2.85) leads to (2.87) and (2.88), where c represents the number of constraints and n the number of points. $\boldsymbol{I}$ stands for the identity matrix, $\mathbf{1}$ for a row vector of ones.

$$\boldsymbol{\beta_{2b}} = \frac{1}{c}\mathbf{1}\left(\boldsymbol{P_c}\boldsymbol{P_c}^T\left(\boldsymbol{D_c}^{+}\dot{\boldsymbol{y}}_{\boldsymbol{m}} - \boldsymbol{y_c}\right) + \boldsymbol{y_c} - \boldsymbol{D_c}^{+}\dot{\boldsymbol{y}}_{\boldsymbol{m}}\right) \tag{2.87}$$

$$\boldsymbol{s_{2b}} = \left(\boldsymbol{I} - \frac{1}{n}\boldsymbol{P_c}^{\mathrm{T}}\mathbf{1}\mathbf{1}^{\mathrm{T}}\boldsymbol{P_c}\right)^{-1}\boldsymbol{P_c}^{\mathrm{T}}\left[\left(\boldsymbol{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^{\mathrm{T}}\right)\left(\boldsymbol{D_c}^{+}\dot{\boldsymbol{y}}_{\boldsymbol{m}} - \boldsymbol{y_c}\right)\right]. \tag{2.88}$$

Defining intermediate variables, such that

$$\boldsymbol{s_{2b}} = \boldsymbol{M}^{-1}\boldsymbol{P_c}^{\mathrm{T}}\boldsymbol{K}, \tag{2.89}$$

where

$$\begin{array}{rcl} \boldsymbol{M} & \triangleq & \boldsymbol{I} - \frac{1}{n}\boldsymbol{P_c}^T\mathbf{1}\mathbf{1}^{\mathrm{T}}\boldsymbol{P_c} \\ \boldsymbol{K} & \triangleq & \left(\boldsymbol{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^{\mathrm{T}}\right)\left(\boldsymbol{D_c}^{+}\dot{\boldsymbol{y}}_{\boldsymbol{m}} - \boldsymbol{y_c}\right). \end{array}$$

Now considering the inverse term of $\boldsymbol{M}$; the inverse of a function of the form

$$\boldsymbol{A} = \boldsymbol{B} - \boldsymbol{u}\,\boldsymbol{v}^{\mathrm{T}}, \tag{2.90}$$

is known to be:

$$\boldsymbol{A}^{-1} = \boldsymbol{B}^{-1} - \frac{\boldsymbol{B}^{-1}\boldsymbol{u}\,\boldsymbol{v}^{\mathrm{T}}\boldsymbol{B}^{-1}}{1 + \boldsymbol{v}^{\mathrm{T}}\boldsymbol{B}^{-1}\boldsymbol{u}}. \tag{2.91}$$

This knowledge can be applied to find the inverse of the matrix $\boldsymbol{M}$. Let us define

$$\begin{array}{rcl} \boldsymbol{A} & \triangleq & \boldsymbol{M} \\ \boldsymbol{B} & \triangleq & \boldsymbol{I} \\ \boldsymbol{B}^{-1} & \triangleq & \boldsymbol{I} \\ \boldsymbol{u} & \triangleq & -\frac{1}{n}\,\boldsymbol{P_c}^T\,\mathbf{1} \\ \boldsymbol{v}^{\mathrm{T}} & \triangleq & \mathbf{1}^{\mathrm{T}}\boldsymbol{P_c}. \end{array}$$

Back substituting these terms into Equation (2.91) and defining $\boldsymbol{w} \triangleq \boldsymbol{P_c}^T\,\mathbf{1}$, $\boldsymbol{M}^{-1}$ can now be expressed as

$$\boldsymbol{M}^{-1} = \boldsymbol{I} - \frac{\frac{1}{n}(\boldsymbol{I}(-\boldsymbol{w})\boldsymbol{w}^{\mathrm{T}}\boldsymbol{I})}{1 - \frac{1}{n}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{I}\boldsymbol{w}}. \tag{2.92}$$

Simplifying and using the information that $n - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} = c$ yields,

$$\boldsymbol{M}^{-1} = \boldsymbol{I} + \frac{\boldsymbol{P_c}^T \mathbf{1}\mathbf{1}^{\mathrm{T}} \boldsymbol{P_c}}{c}. \tag{2.93}$$

Now the term $\boldsymbol{K}$ is dealt with in more detail. Multiplying out the terms in the brackets yields,

$$\boldsymbol{K} = \left[\boldsymbol{D_c}^+ \dot{\boldsymbol{y}}_{\boldsymbol{m}} - \frac{1}{n} \mathbf{1}\mathbf{1}^{\mathrm{T}} \boldsymbol{D_c}^+ \dot{\boldsymbol{y}}_{\boldsymbol{m}} - \boldsymbol{y_c} + \frac{1}{n} \mathbf{1}\mathbf{1}^{\mathrm{T}} \boldsymbol{y_c} \right]. \tag{2.94}$$

$\mathbf{1}\mathbf{1}^{\mathrm{T}}$ is an nxn matrix of ones, where n is the number of points. The computation $\frac{1}{n}\mathbf{1}\mathbf{1}^{\mathrm{T}} \boldsymbol{D_c}^+$ corresponds to calculating the mean of the columns of $\boldsymbol{D_c}^T$. However, the columns of $\boldsymbol{D_c}^T$ are mean free. Consequently,

$$\frac{1}{n} \mathbf{1}\mathbf{1}^{\mathrm{T}} \boldsymbol{D_c}^+ = \mathbf{0}, \tag{2.95}$$

and with this

$$\frac{1}{n} \mathbf{1}\mathbf{1}^{\mathrm{T}} \boldsymbol{D_c}^+ \dot{\boldsymbol{y}}_{\boldsymbol{m}} = 0. \tag{2.96}$$

The multiplication of a vector of ones, $\mathbf{1}$, and the vector $\boldsymbol{y_c}$, divided by the number of elements n delivers the mean value, i.e., $\bar{\boldsymbol{y_c}}$. This means

$$\bar{\boldsymbol{y_c}} = \frac{1}{n} \mathbf{1}^{\mathrm{T}} \boldsymbol{y_c}. \tag{2.97}$$

Based on latter information, Equation (2.88) can be reformulated as follows:

$$\boldsymbol{s_{2b}} = \left(\boldsymbol{I} + \frac{\boldsymbol{P_c}^T \mathbf{1}\mathbf{1}^{\mathrm{T}} \boldsymbol{P_c}}{c} \right) \boldsymbol{P_c}^T \left(\boldsymbol{D_c}^+ \dot{\boldsymbol{y}}_{\boldsymbol{m}} - \boldsymbol{y_c} + \mathbf{1} \bar{\boldsymbol{y_c}} \right) \tag{2.98}$$

Having found the new $\boldsymbol{s_{2b}}$, we can calculate the reconstruction as follows:

$$\boldsymbol{y} = \boldsymbol{y_c} + \boldsymbol{P_c} \left(\boldsymbol{I} + \frac{\boldsymbol{P_c}^T \mathbf{1}\mathbf{1}^{\mathrm{T}} \boldsymbol{P_c}}{c} \right) \boldsymbol{P_c}^T \left(\boldsymbol{D_c}^+ \dot{\boldsymbol{y}}_{\boldsymbol{m}} - \boldsymbol{y_c} + \mathbf{1} \bar{\boldsymbol{y_c}} \right) \tag{2.99}$$

$\boldsymbol{P_c} \boldsymbol{P_c}^{\mathrm{T}} \boldsymbol{y_c}$ delivers a matrix full of zeros, consequently,

$$\boldsymbol{P_c} \left(\boldsymbol{I} + \frac{\boldsymbol{P_c}^T \mathbf{1}\mathbf{1}^{\mathrm{T}} \boldsymbol{P_c}}{c} \right) \boldsymbol{P_c}^T \boldsymbol{y_c} = 0. \tag{2.100}$$

Now we can write the final formulation of the reconstruction $\boldsymbol{s_{2b}}$ as follows,

$$\boldsymbol{s_{2b}} = \left(\boldsymbol{I} + \frac{\boldsymbol{P_c}^T \mathbf{1}\mathbf{1}^{\mathrm{T}} \boldsymbol{P_c}}{c} \right) \boldsymbol{P_c}^T \left(\boldsymbol{D}^+ \dot{\boldsymbol{y}}_{\boldsymbol{m}} + \mathbf{1} \bar{\boldsymbol{y_c}} \right). \tag{2.101}$$

Solution 2a and 2b in comparison As already stated above, Solutions 2a and 2b are principally the same. However, due to the computation of irrelevant terms, and therefore unnecessary computational effort in Solution 2a, it was necessary to derive Solution 2b. In Figure 2.11 you will find the reconstruction using Solution 2 and each solution approach. It demonstrates that Solution 2a and Solution 2b are in fact the same in terms of reconstruction ability. The only difference in the two solutions lies in their numerical efficiency, which is better for Solution 2b. This is the reason why $\boldsymbol{s_{2b}}$ from Equation (2.101) will be used for further reconstruction calculations for Solution 2.

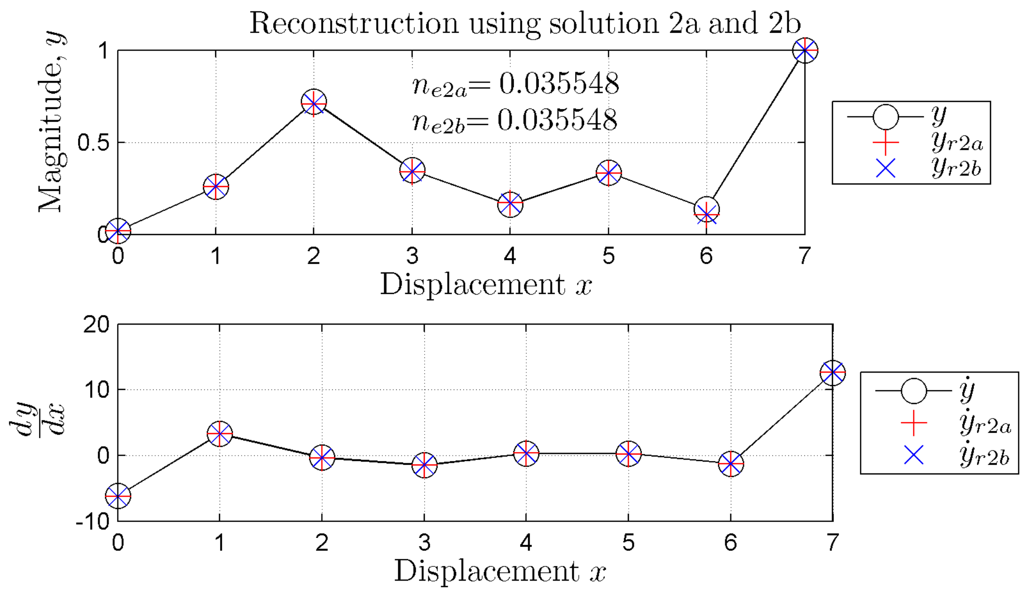


Figure 2.11: Reconstruction of $\boldsymbol{y}$ using Solution 2a and 2b. In the first plot the original function $\boldsymbol{y}$ and the reconstructions of its derivative $\boldsymbol{y_{r2a}}$ and $\boldsymbol{y_{r2b}}$ using $\boldsymbol{s_{2a}}$ and $\boldsymbol{s_{2b}}$ from Equation (2.83) and (2.101), respectively, are shown. One can see that both solutions deliver the exact same reconstruction, as well as the same mean error. In the second plot the derivative of the vector $\boldsymbol{y}$, $\dot{\boldsymbol{y}}$, which was used for reconstruction, is displayed. Furthermore the derivatives of the reconstructed curves were calculated according to the function $\dot{\boldsymbol{y}}_{r2a,b} = \boldsymbol{D}\,\boldsymbol{y}_{r2a,b}$, and plotted onto the graph.

Chapter 3

Numerical Simulation

3.1 The solutions in comparison

The examples for the reconstruction given in Section 2.12.1 and 2.12.2 have an arbitrary, non-continuous function as its original function. This function was chosen only for demonstration purposes. For further verification and comparison of the alternative solutions, the reconstructions using a polynomial function, as well as the result of a Monte-Carlo simulation for the reconstruction of polynomials and an arbitrary function will be discussed in detail in this section.

3.2 Reconstruction of the arbitrary function y

The most generic way of examining the reconstruction operators is to use a completely arbitrary curve y, as can be seen in Figure 3.1. To do so, it was necessary to generate a random vector $\boldsymbol{y}$ on 7 points, representing an arbitrary curve. Then the derivative was taken using the differential operator $\boldsymbol{D_c}$ from Equation (2.18), which delivers the derivative, $\dot{\boldsymbol{y}}$, of the curve. Any real world measurement is perturbed in some way, so it is assumed that the present measurement is biased by some kind of noise as well. This is why noise of two percent of the range of $\dot{\boldsymbol{y}}$ is added to the derivative. It has to be said, that when computing the derivative of a vector like $\boldsymbol{y}$, it was observed that it has very large values at both end points. This is why for the computation of σ these values are left aside when computing the range of the derivative, otherwise the noise generated would be too large for the simulation of a measurement. Adding the generated noise to the derivative $\dot{\boldsymbol{y}}$ provides the simulated real life measurement $\dot{\boldsymbol{y}}_{\boldsymbol{m}}$, see Figure 3.1. The MATLAB code for generating such a function is given in Table 3.1.

```matlab
%
% define the number of points desired
numPts = 7;
%
% compute the vector x
x = linspace(0,numPts, numPts+1);
%
% compute the arbitrary vector y and scale it
y = randn(numPts+1,1);
y = y/range(y);
y = y- min(y);
%
% compute the Gram Polynomials
[G,dG] = gramPolyDiff(numPts+1,1);
%
% compute the Differentiating matrix Dc
Dc = dG * pinv(G);
%
% compute the simulated measured derivatives
dy = Dc * y;
%
% compute sigma for noise generation
sigma = 0.02 * norm( dy );
%
%add noise to the derivative:
dym = dy + sigma * randn( numPts+1, 1);
%
```

Source Code 3.1: Code for the synthesis of the constraints.

For the current reconstruction problem, positional constraints were placed onto the beginning and endpoint of the vector. To implement the particular solution, the position at which the constraints were placed needs to be known. Since the function which is to be reconstructed is in fact known, we know the positional constraints and with that the particular solution $\boldsymbol{y_c}$ can be defined according to Source Code 2.2.

The reconstruction operators are applied to the perturbed inclinometer measurement, of which the results are given in Figure 3.2. The mean error of each reconstruction is given in this figure. It can be seen that Operator 1 reconstructs better than Operator 2. The mean error of Operator 1 is 0.1098, while the reconstruction using Operator 2 deviates from the original function by a mean error of 0.14936.

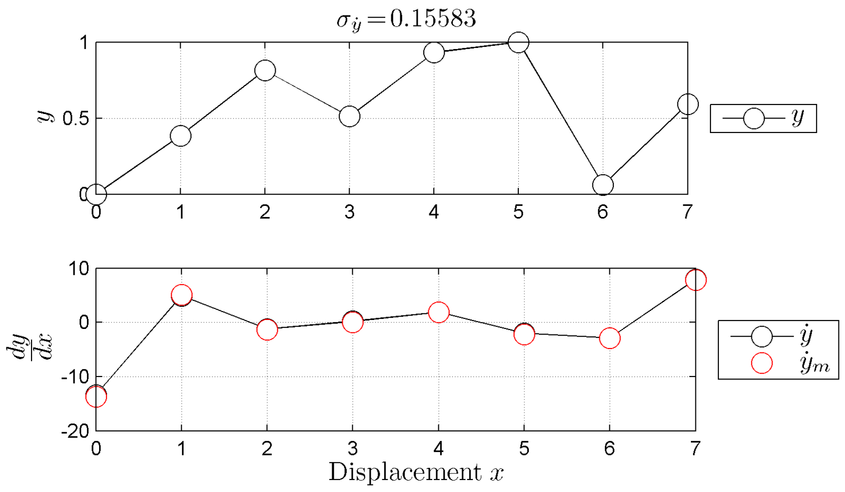


Figure 3.1: Here the function $\boldsymbol{y}$ used for reconstruction is given. The curve was generated on 8 points (0 to 7) using the *randn*-function in MATLAB. $\dot{\boldsymbol{y}}_{\boldsymbol{m}}$, the simulated measured derivative is plotted ontop of the simulated derivative without noise, vector $\dot{\boldsymbol{y}}$ is shown in red in the bottom subplot.

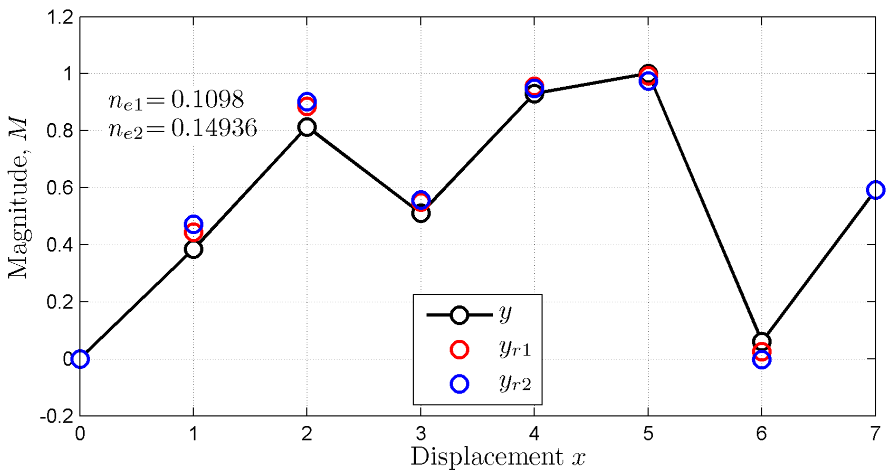


Figure 3.2: In this image one can see the reconstructed points $\boldsymbol{y_{r1,2}}$, in red and blue respectively, plotted onto the original function $\boldsymbol{y}$. The mean error of the first reconstruction is $n_{e1} = 0.1098$, the one of the second reconstruction is $n_{e2} = 0.14936$. This mean error is always related to the performance of the reconstructions, as well to the noise generated when calculating $\dot{\boldsymbol{y}}_{\boldsymbol{m}}$. The noise was generated with $\sigma = 0.15583$, so the mean errors n_{e1} and n_{e2} are in fact plausible results.

In Figure 3.3 the error of each reconstruction, as well as the error of their derivatives is presented. The course of the error is similar for both reconstructions, yet they differ in terms of magnitude. This is caused by the distribution of the noise onto the function, which means that the reconstruction is proportional to the measured data. The error of the beginning and end point of the reconstructed functions $\boldsymbol{y_{r1,2}}$ is zero, which makes perfect sense because that was our known boundary condition for the reconstruction. The derivatives of the reconstructions have large errors on both edges, since the generic curve is not polynomial and the implied nature of the matrix differential operator is polynomial. The first method of reconstruction represents the original function best, i.e. is able to handle noise better than the second operator.

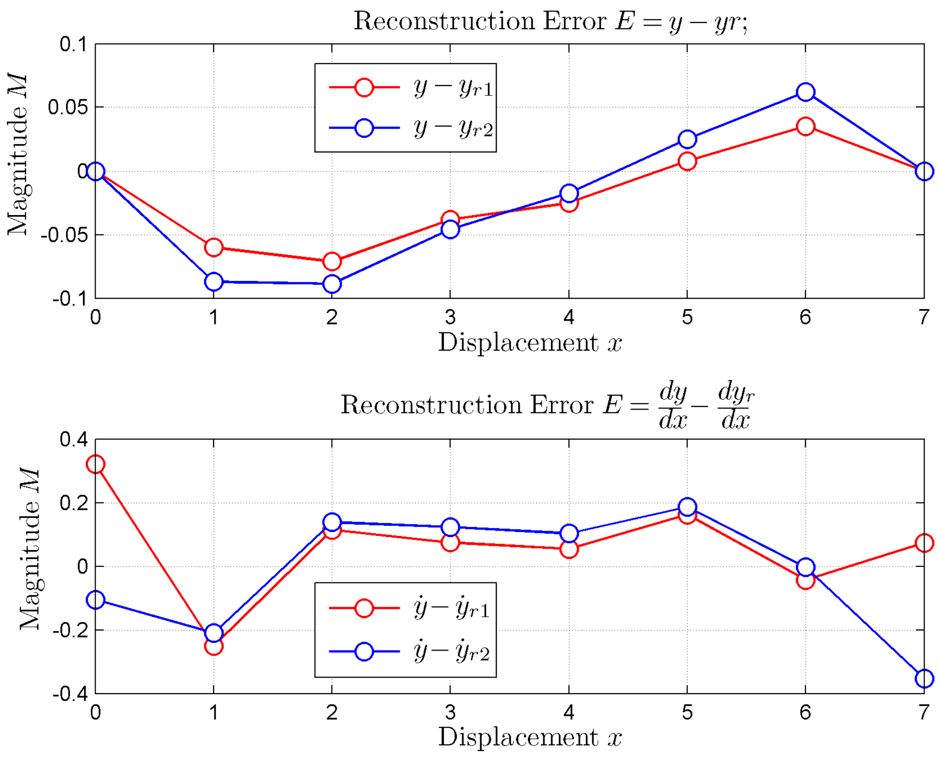


Figure 3.3: In the upper plot, the error of the reconstruction in the space of the curve can be observed. In the lower plot the reconstruction error of the derivatives can be seen. The greatest error in the derivative space is at the Displacement $x = 0$ and $x = 7$. This is caused by the differential operator $\boldsymbol{D}$. Due to the implied polynomial nature of the matrix differential operator the derivative at the edges of a function is especially difficult to compute.

3.3 Polynomial Reconstruction

In the real application we are examining the deflection of structures, e.g., beams etc. These are governed by the *Euler-Bernoulli* differential equation; the solutions which are known to be polynomial in nature. Hence, deflection curves and their derivatives are synthesized from polynomials.

The first step taken for the simulation was generating a polynomial $\boldsymbol{y}$ of degree 4 on seven points from constrained polynomials i.e. $\boldsymbol{y} = \boldsymbol{B_c \alpha}$, representing a deflection according to *Euler-Bernoulli*. Then the steps explained in the previous sections are followed:

1. calculate derivative using operator $\boldsymbol{D_c}$ from Equation (2.18);
2. generate noise of two percent of the range of $\dot{\boldsymbol{y}}$;
3. add noise to $\dot{\boldsymbol{y}}$, denoted as $\dot{\boldsymbol{y}}_{\boldsymbol{m}}$;
4. define boundary conditions

For the problem at hand, the beginning and the end points were chosen as fixed points and are therefore used as boundary conditions in the space of the curve. These same boundary conditions were used for each reconstruction. The reconstruction operators are applied to the perturbed inclinometer measurement, of which the results are given in Figure 3.5 and 3.6. As foreseen, the mean error of each reconstruction is of the same magnitude as σ used for the noise simulation. In fact, $\sigma = 0.078842$, where $n_{e1} = 0.023277$ and $n_{e2} = 0.052757$, which means that during reconstruction the effective noise is diminished.

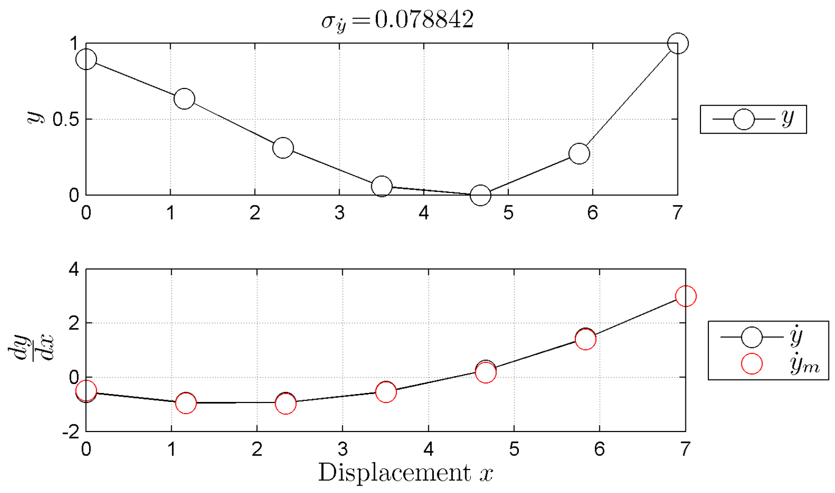


Figure 3.4: Polynomial $\boldsymbol{y}$ and its derivative $\dot{\boldsymbol{y}}$, in the bottom subplot, calculated using the differentiating matrix $\boldsymbol{D}$. On top of $\dot{\boldsymbol{y}}$ the vector $\dot{\boldsymbol{y}}_{\boldsymbol{m}}$, representing the measured data, is displayed in red.

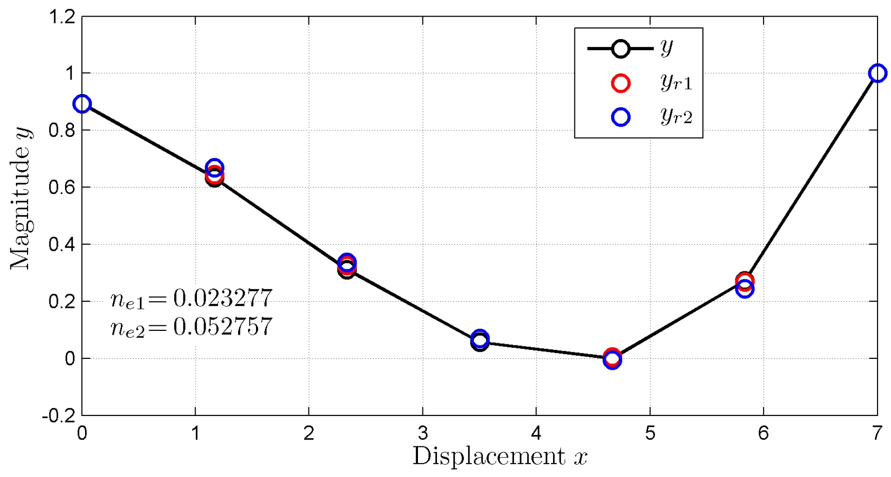


Figure 3.5: The original polynomial $\boldsymbol{y}$ and its reconstructions $\boldsymbol{y_{r1}}$ and $\boldsymbol{y_{r2}}$ in red and blue, respectively.

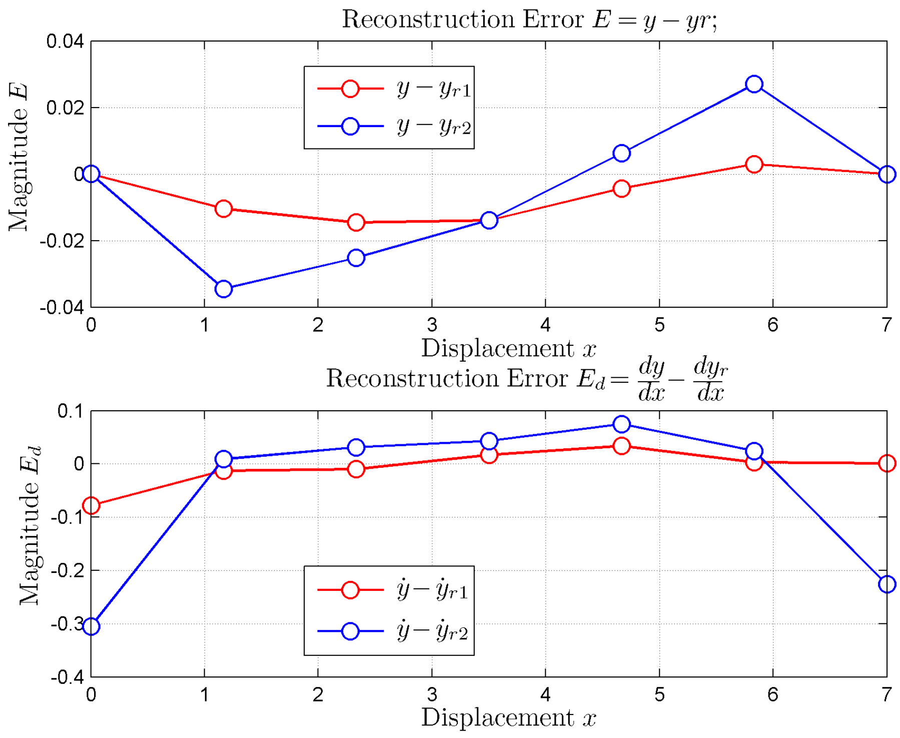


Figure 3.6: In this figure the reconstruction errors are plotted. In the upper plot, the reconstruction error E of the reconstructions in the line space can be observed. Here it can be seen that the respective errors of each function are, at a maximum, close to two percent, which is very small considering that the simulated original data had an error of four percent. In the lower subplot the errors in the derivative space are plotted.

In Figure 3.6 the error of each reconstruction, as well as the error of their derivatives is presented. It can be seen that the error distribution is proportional for each reconstruction. However, for Reconstruction 1 it is much lower than for Reconstruction 2. Yet, one reconstruction is not enough to verify the operators. To be able to make a statistical statement, a Monte-Carlo simulation is performed in Section 3.5.

3.4 Statistical evaluation

The results of the Monte-Carlo simulation are used for statistical evaluation. The reconstructions are going to be compared in various ways. The first part is to evaluate the reconstruction of each reconstruction point along the curve. This will give information of whether or not there is a systematic error to be encountered. The reconstruction errors of each point are going to be compared in terms of their

1. mean / expectation value (α_1),
2. standard deviation (α_2),
3. skewness (α_3),
4. kurtosis (α_4).

The second step is the evaluation of the entire reconstruction for each simulation. For that the Euclidean Norm of each reconstruction is taken and afterwards compared in terms of its

1. mean value ($\bar{x}$),
2. mode value (x_m),
3. median value ($x_{0.5}$).

This gives information of the general reconstruction ability of each reconstruction operator.

Mean / Expectation Value (α_1, $\bar{x}$):

"The *mean* or *expectation value* of a random variable is the sum of all possible values x_i of x multiplied by their corresponding probabilities"[5]. It is mathematically defined as

$$\alpha_1 = \bar{x} = \sum_{i=1}^{n} x_i P(x_i). \tag{3.1}$$

Knowing the mean of the error distribution of a reconstruction operator one can make a statement on its performance as well as on the error likely to be encountered. The lower the mean value of the error, the better the result of the reconstruction.

Standard Deviation (α_2)

The standard deviation is defined as ”The positive square root of the variance ... It is a measure of the average deviation of the measurement x from the expected value”.[5]. The standard deviation has the definition:

$$\alpha_2 = \sqrt{\mathrm{var}(x)} \tag{3.2}$$

It defines the likelihood of the error, calculated by the mean, to occur. The standard deviation has the same dimension as x and is therefore identified with the error of the measurement:

$$\alpha_2 = \delta x. \tag{3.3}$$

Skewness (α_3)

”The third moment about the mean is sometimes called *skewness* ... It is positive (negative) if the distribution is skew to the right (left) of the mean. For symmetric distributions the skewness vanishes” [5]. In the case of a mean error greater than zero, a positive skewness means, that the decline on the side to zero is sharper than the decline on the right side of the mean. The skewness of a normal distribution is zero. It is a measure of asymmetry of the distribution function. (Also see Figure 3.7).

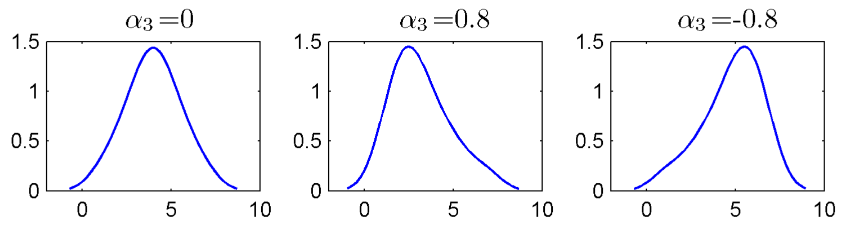


Figure 3.7: In this figure an example for data with normal distribution, data with positive skew and data with negative skew is given.

Kurtosis (α_3)

”The kurtosis α_4 describes how strongly a bell-shaped distribution is curved and thus how steep the peak is”[3]. Whereby a normal distribution has a kurtosis value equal to three. A value greater than three has a steeper peak and thence fat tails. A kurtosis value smaller than three results in a flat distribution and tails which are quite thin, see Figure 3.8.

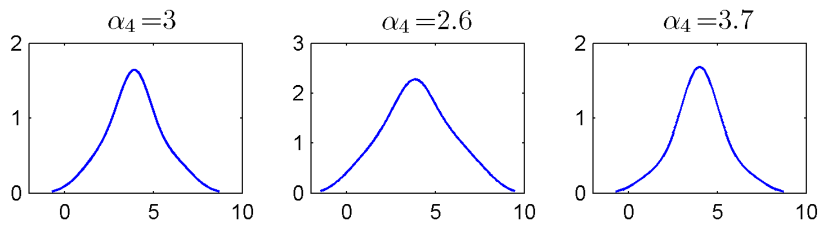


Figure 3.8: In the first subplot of this figure (from left to right) one can see a data distribution with normal distribution, i.e. with kurtosis = 3. In the second subplot a data distribution with kurtosis = 2.6, i.e. kurtosis smaller than 3 is shown. On the right a distribution with kurtosis = 3.7, i.e. kurtosis greater than 3 is given.

Euclidean Norm

"The *Euclidean norm* or the *absolute value* of a vector is

$$|a| = ||a||_2 = a = \sqrt{a^T a} = \sqrt{\sum_j a_j^{2}}\text{"} \tag{3.4}$$

[5]. The norm is a function that assigns a positive length or size to each vector in a vector space. In a 2-dimensional space where vectors are usually drawn as arrows, the Euclidean norm assigns the length of the arrow to the vector. By taking the norm of each reconstruction one can make a statement on the average level of achievement, i.e. the average error of reconstruction.

Mode Value (x_m)

"The *mode* x_m (or *most probable value*) of a distribution is defined as that value of the random variable that corresponds to the highest probability:

$$P\left(x = x_m\right) = max. \tag{3.5}$$

[5]. It is hence the value with the highest occurrence. The *mode* is not necessarily a unique number, however in terms of the computations done in this thesis it is, because the distribution of the present example is *unimodal. Unimodal* is a distribution with only one local maximum.

Median Value ($x_{0.5}$)

"The *median* $x_{0.5}$ of a distribution is defined as that value of the random variable for which the distribution function equals 1/2:

$$F(x_{0.5}) = P(x < x_{0.5}) = 0.5\text{"} \tag{3.6}$$

[5]. In other words, the median value is the value below which 50% of the values fall. The median is resistant towards outliers, which makes it a good dispersion measure and is therefore a good alternative to the mean value. The *median value* of our reconstruction errors is considered a 'good' result, when it is close to zero.

3.5 Monte-Carlo Simulation

The reconstructions presented up until now were simulated using functions without varying noise. In order to be able to make statistical statements on the performance of the reconstruction functions on noise behaviour, a Monte-Carlo simulation [1] is performed. Two different curves were used for implementation: for the first simulation a curve $\boldsymbol{y}$ with 8 arbitrary values in the range from zero to one, for $x = 0...7$, is used (see Figure 3.1). The constraints placed on the reconstruction are positional constraints on the end points; the constraints were $\boldsymbol{y}(0) = 0$ and $y(end) = 0.5924$. The second Monte-Carlo simulation is done using a polynomial of degree 4, evaluated on 8 ($x = 0...7$) points with the constraints $\boldsymbol{y}(0) = 0.9994$ and $y(end) = 0$, whereby $\boldsymbol{y}(0)$ represents the first, $y(end)$ the last entry in the $\boldsymbol{y}$-vector (see Figure 3.4). Both simulations use the data of 10.000 reconstructions with varying noise.

3.5.1 Monte-Carlo Simulation - arbitrary curve $\boldsymbol{y}$

The first Monte-Carlo simulation was performed using the arbitrary curve $\boldsymbol{y}$ and its derivative $\dot{\boldsymbol{y}}$ see Section 3.2, Figure 3.1. It should be noted that the Gaussian noise is added to the derivative of the curve, i.e. $\dot{\boldsymbol{y}}$, and not to $\boldsymbol{y}$. This enables the evaluation of the behaviour of the reconstructions with respect to noise.

The results of the MC simulation for the arbitrary curve reconstruction are shown in Figure 3.9. For the simulation 10000 iterations with random noise having a standard deviation of $\sigma_{ma} = 0.1558$ was added to the simulated inclinometer measurement[2]. The mean reconstruction of all simulations and the standard deviation of each node is given. The distribution of the standard deviation is shown in Figure 3.10. For the statistic evaluation of both operators, a histogram of the norm of the reconstruction error is given in Figure 3.12[3]. The

[1]"The use of random numbers in computer programs is often called *the Monte Carlo method"* [5]. Here the *Monte Carlo* Simulation is referring to a method of reconstructing a function various times in order to be able to make a statistical statement.

[2]The MATLAB function *randn* was used to generate the vectors of random noise

[3]The curve fitted to the histogram is generated using the *fitdist* and *gevpdf* function provided by MATLAB

code for fitting such a distribution function is given in Source Code 3.2. The most significant results are:

1. both reconstruction methods are mean free, i.e. both reconstruct the expected value of the curve correctly;
2. the statistical properties of reconstruction method 1 (a discrete *Rayleigh Ritz* method) are constantly marginally better than for method 2 (a variational solution), see Table 3.1 for a summary of the descriptive statistics.

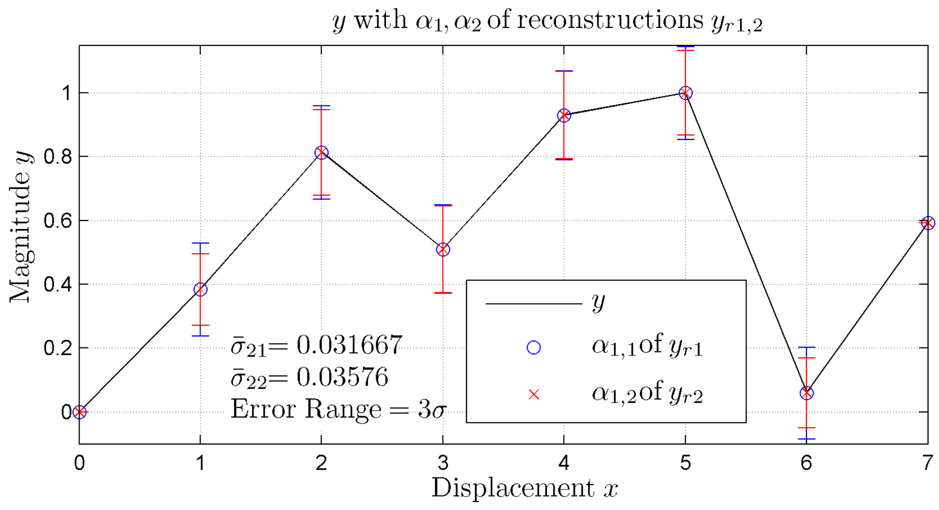


Figure 3.9: Results of the Monte-Carlo simulation with the standard deviation of $\boldsymbol{y_{r1}}$ and $\boldsymbol{y_{r2}}$, i.e. α_{21} and α_{22}, respectively. 10000 iterations were performed. The mean reconstructions $\boldsymbol{\alpha_{11}}$ and $\boldsymbol{\alpha_{12}}$ are given in blue (**o**) and red (**x**), respectively. The standard deviations $\boldsymbol{\alpha_{21}}$ (blue) and $\boldsymbol{y}\alpha_{22}$) (red) are plotted in bars, multiplied by a factor 3. The original function $\boldsymbol{y}$ is given in black. The constraints placed on the reconstructions are $\boldsymbol{y}(0) = 0$ and $\boldsymbol{y}(end) = 0.5924$.

```
%fit distribution to the histogram of the error vector erV using 'gev' -
          generalized extreme value
fit = fitdist(hist(erV),'gev');
%
%extract parameters (param(1) = k; param(2) = sigma, param(3) = mu)
for i = 1:3
    param(i) = fit.Param(i);
end
%
%generate y
y = gevpdf(x,param(1),param(2),param(3));
```

Source Code 3.2: Code for synthesis of the distribution fit for the histogram.

Reconstructions		1	2
mean error	α_1	0.0000	0.0000
standard deviation	α_2	0.0317	0.0358
skewness	α_3	0.0047	0.0088
kurtosis	α_4	2.9722	2.9809
mean norm error	$\bar{x}$	0.0940	0.1065
mode norm value	x_m	0.0735	0.0866
median norm value	$x_{0.5}$	0.0863	0.0994

Table 3.1: Descriptive statistics of Monte-Carlo simulation of an arbitrary curve $\boldsymbol{y}$. 10000 iterations were performed. The statistic data of the reconstruction of each point is given. The average of the mean error at each point is zero for each reconstruction operator. The values of the mean, standard deviation, skewness and kurtosis were computed using the error vector of each reconstruction. The mean, mode and median values were computed using the distribution of the Euclidean norm of each reconstruction. Reconstruction Method 1 is superior in terms of standard deviation of the error vector as well as in respect to mean, mode and median values of the norm of the error vector. Method 2 shows better performance in terms of kurtosis and skewness. Method 1 delivers marginally better results for the reconstruction of an arbitrary function $\boldsymbol{y}$.

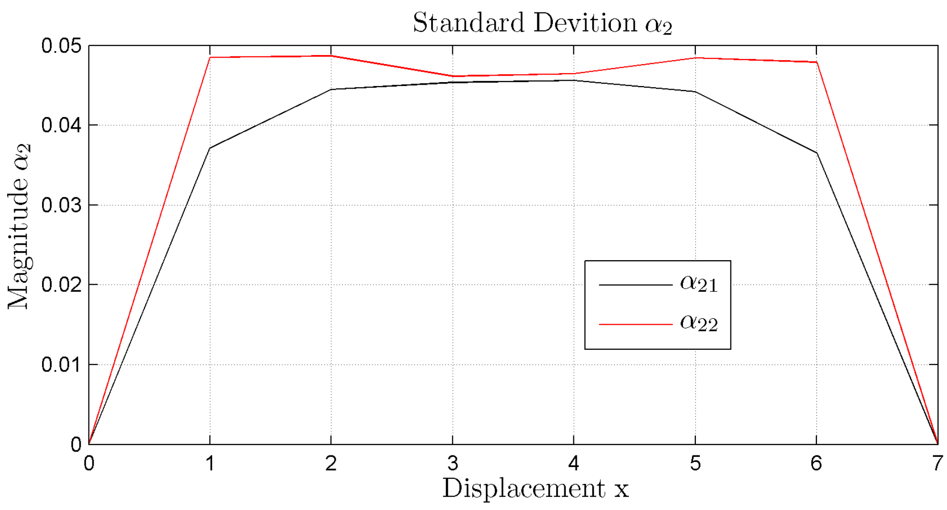


Figure 3.10: Standard deviation of the reconstruction error of each node. The first reconstruction has a maximum at $x = 3.5$. The second reconstruction looks like the inversion of the first one and has a minimum at the same position. The average standard deviation of all points is $\alpha_{2,1} = 0.0317$ for Reconstruction 1 and $\alpha_{2,2} = 0.0358$ for Reconstruction 2. The standard deviation at the end points is equal to zero, which follows from the constraints placed on the reconstruction.

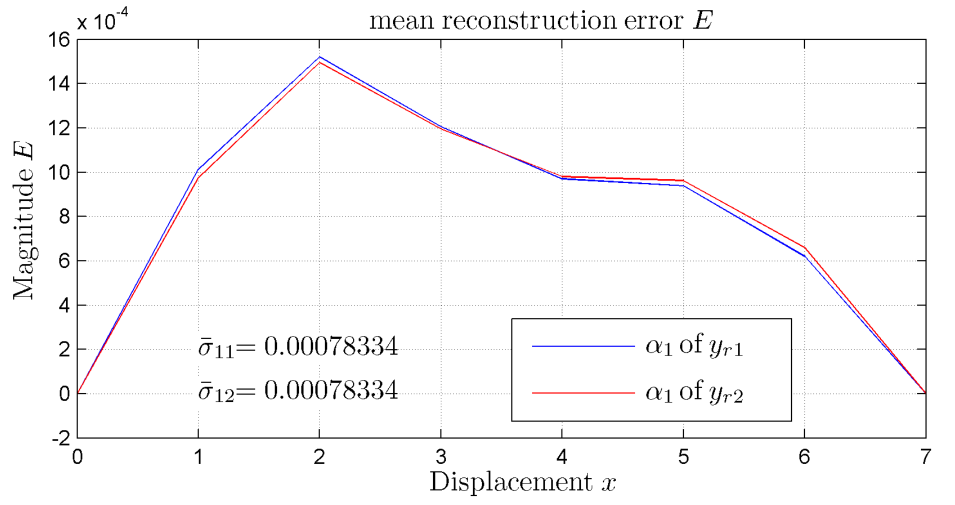


Figure 3.11: The mean of the reconstruction error of the arbitrary curve $\boldsymbol{y}$ at each node. Although the reconstructions themselves are mean free, the reconstruction of each point is not. However, it is only of Magnitude 10^{-4}, and therefore neglectably small.

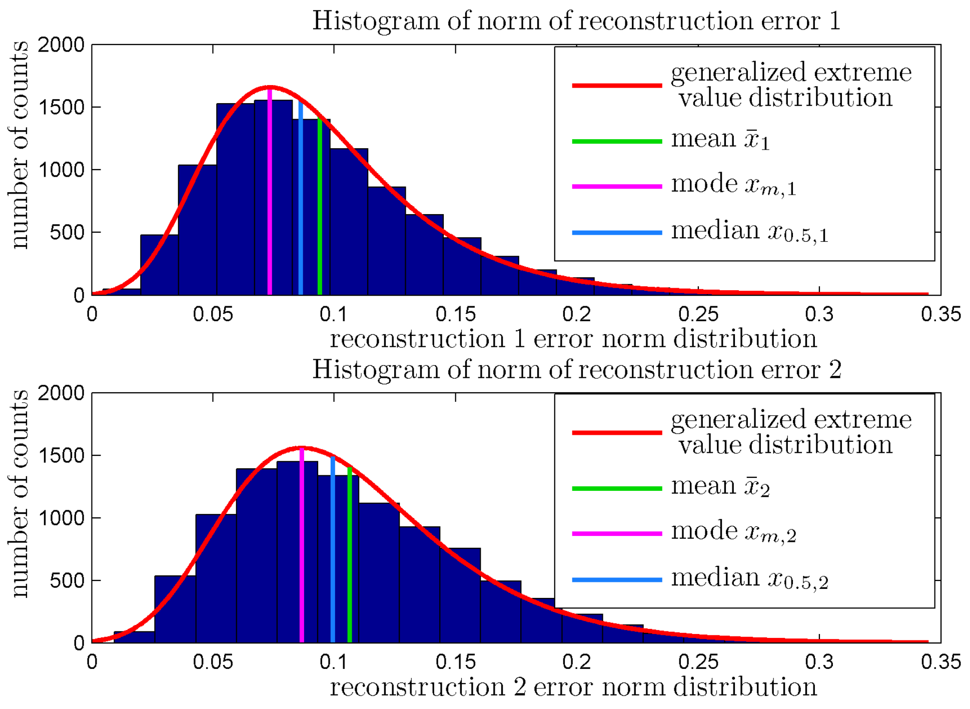


Figure 3.12: Distribution of the norm of the reconstruction error of the arbitrary curve $\boldsymbol{y}$ for Reconstruction 1 and 2 in the top and bottom plot, respectively. The curve fitted to the histogram is generated using the *fitdist* and *gevpdf* function provided by MATLAB. The mean of the norm of the reconstruction error $(\bar{x}_{1,2})$, the mode of the norm of the error $(x_{m,1,2})$ as well as the median of the norm of the error $(x_{0.5,1,2})$ are displayed. For the values see Table 3.1. Method 1 delivers lower values for: mean, mode and median value.

3.5.2 Monte-Carlo Simulation - polynomial $\boldsymbol{y}$

In this section the Monte-Carlo simulation is performed using the reconstruction of a polynomial $\boldsymbol{y}$. The polynomial used is shown in Figure 3.13. It is a polynomial of degree 4 and was chosen to show the behaviour of curve reconstruction on a function which represents the deflection of a cantilever supported at one end. For each simulation noise was added to the derivative. The noise was generated using the *randn* function provided by MATLAB multiplied by a factor σ_m, whereby $\sigma_{mp} = 0.0096$ for the polynomial at hand. In Figure 3.15 the mean reconstruction at each node of Reconstruction 1 and 2 is plotted onto the original function $\boldsymbol{y}$. The mean error of each reconstruction point is smaller for the polynomial function; it is only of magnitude 10^{-5} (see Figure 3.17). Unlike the mean reconstruction error, the standard deviation for polynomials follows the same systematic as the standard deviation for the arbitrary curve presented in the previous section (Figure 3.16). The norm of the reconstruction error shows better results for the polynomial reconstruction. This has two reasons:

1. the range of the derivative of the arbitrary curve is much higher and thus σ_a computed is of greater magnitude than σ_p of the polynomial curve;

2. the basis functions used for the computation of the derivative matrix $\boldsymbol{D_c}$ as well as for the integrator matrix $\boldsymbol{A_g}$ were Polynomial basis functions.

It is obvious that Polynomial basis functions for the computation of these two matrices deliver better results for a polynomial than for an arbitrary function, not necessarily being of polynomial nature. However, if the nature of the structure observed is unknown, or not polynomial, one is advised to use different basis functions or a different way of computing the derivative matrix $\boldsymbol{D_c}$ as well as the integrator matrix $\boldsymbol{A_g}$, as our reconstruction methods are valid for any basis functions desired.

In general it can be said, that Reconstruction Method 1 has marginally, however constantly, better results than Reconstruction Method 2, no matter if the original function be of arbitrary or polynomial nature; see Tables 3.2 for the descriptive statistics.

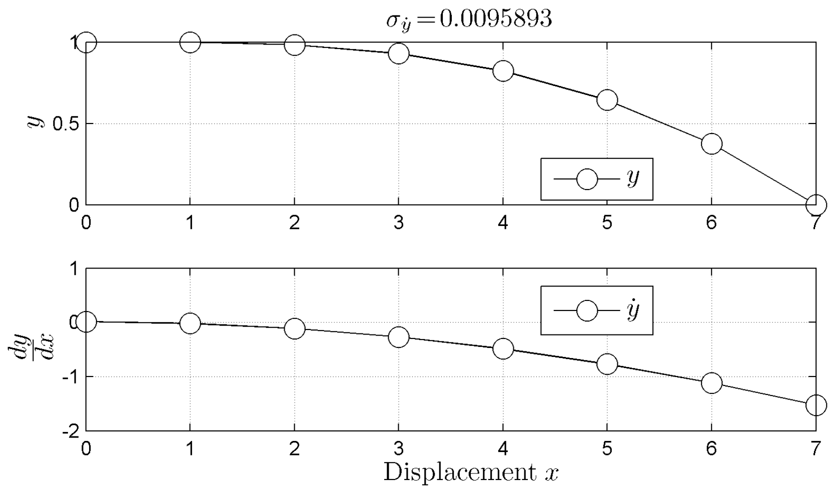


Figure 3.13: Polynomial $\boldsymbol{y}$ and its derivative $\dot{\boldsymbol{y}}$, in the bottom subplot, calculated using the differentiating matrix $\boldsymbol{D}$.

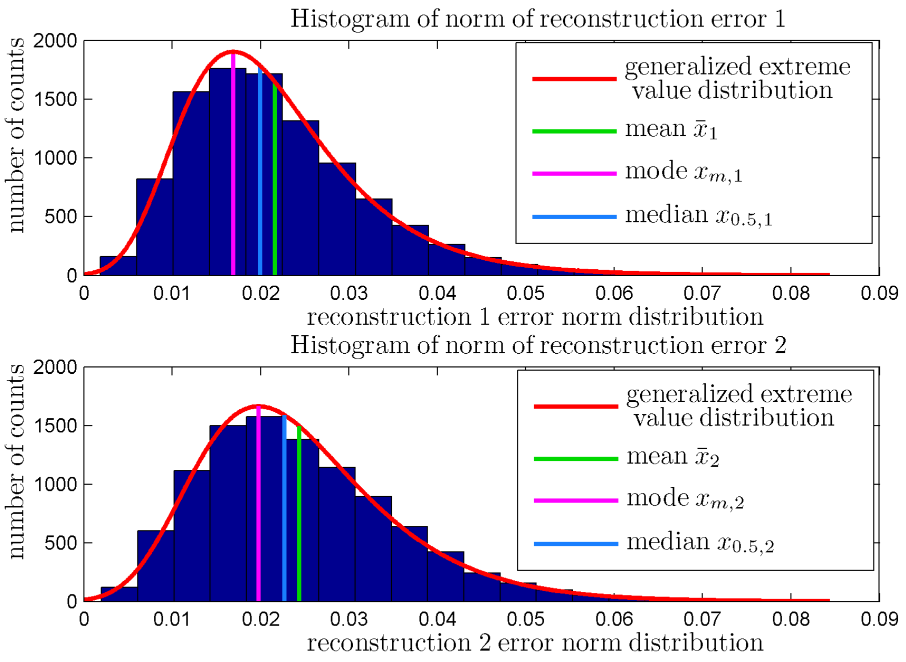


Figure 3.14: Distribution of the norm of the reconstruction error of the arbitrary curve $\boldsymbol{y}$ for Reconstruction 1 and 2 in the top and bottom subplot, respectively. The curve fitted to the histogram is generated using the *fitdist* and *gevpdf* function provided by MATLAB. The mean of the norm of the reconstruction error ($\bar{x}_{1,2}$), the mode of the norm of the error ($x_{m,1,2}$) as well as the median of the norm of the error ($x_{0.5,1,2}$) are displayed. For the values see Table 3.2. Method one delivers lower, and thence better results for each, mean, mode and median value.

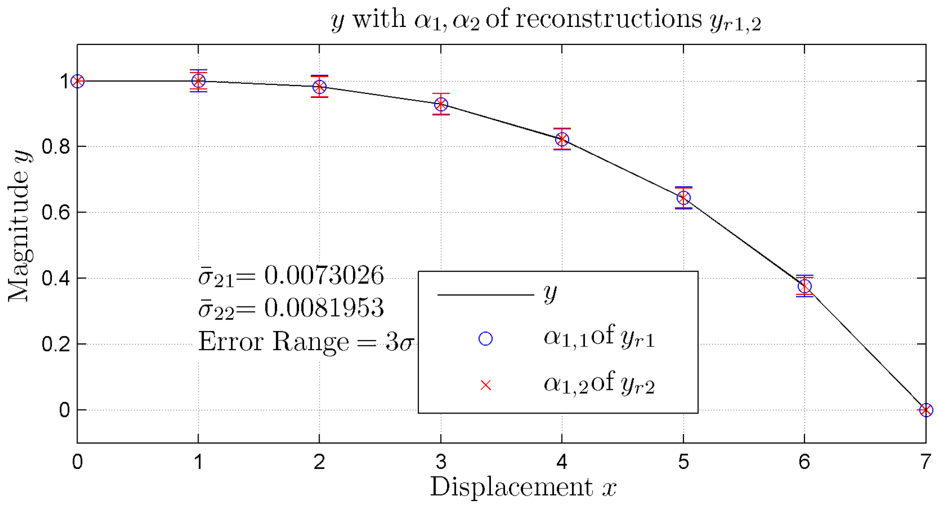


Figure 3.15: The result of the Monte-Carlo simulation of the reconstruction of the original curve $\boldsymbol{y}$, displayed in black. The mean reconstruction error of each node of Reconstructions 1 and 2 is plotted in blue (**o**), and red (**x**), respectively. The standard deviation of each reconstruction point is given in blue (Operator 1) and red (Operator 2) bars, multiplied by a factor three.

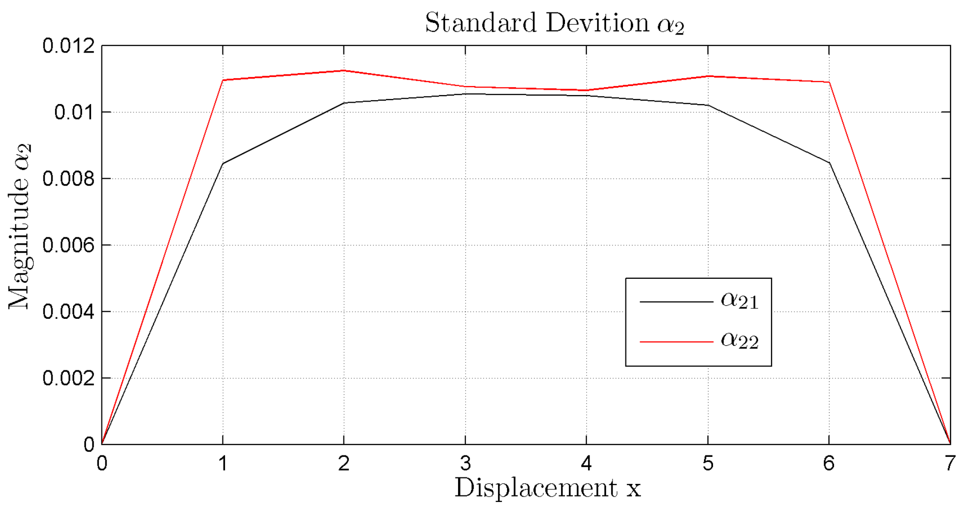


Figure 3.16: In this Figure the standard deviation of the reconstruction error of each node is given for Reconstruction 1 (blue) and 2 (red).

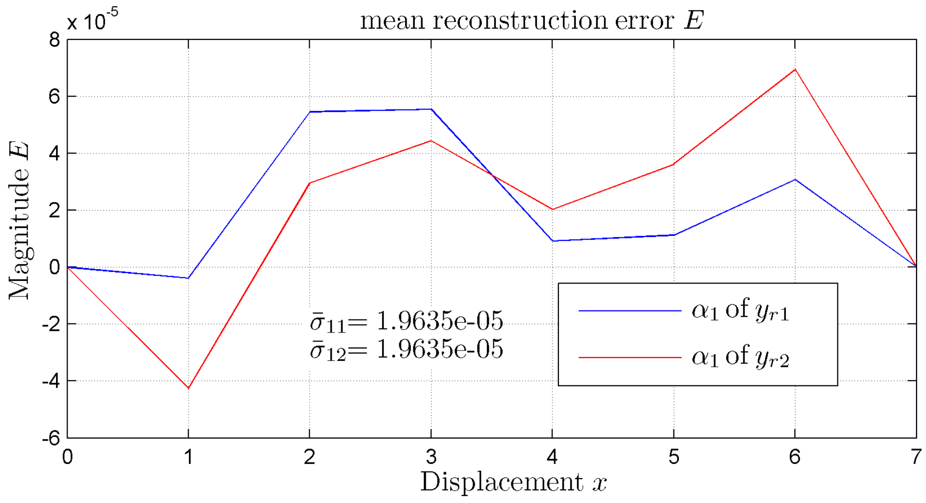


Figure 3.17: The mean reconstruction error of the polynomial $\boldsymbol{y}$ at each node. The mean reconstruction error is of Magnitude 10^{-5}, i.e. the reconstructions are mean free and are therefore valid.

Reconstructions		1	2
mean error	α_1	0.0000	0.0000
standard deviation	α_2	0.0073	0.0082
skewness	α_3	0.0099	-0.0085
kurtosis	α_4	3.0398	3.0373
mean norm error	$\bar{x}$	0.0217	0.0244
mode norm value	x_m	0.0169	0.0198
median norm value	$x_{0.5}$	0.0199	0.00227

Table 3.2: Result of the Monte-Carlo simulation of the reconstructions of the Polynomial function $\boldsymbol{y}$. 10000 iterations were performed. The statistic data of the reconstruction of each point is given. The average of the mean error at each point is zero for each reconstruction method. The values of the mean, standard deviation, skewness and kurtosis were computed using the error vector of each reconstruction. The values mean, mode and median were computed using the distribution of the Euclidean norm of each reconstruction. Reconstruction Method 1 is superior in every respect, with the only exception of the kurtosis value. The statistical values presented confirm that Method 1 has better performance on curve reconstruction of a polynomial of degree 4.

Chapter 4

Experimental Verification

For experimental verification of the reconstruction methods an assembly of a structure for deflection measurement was implemented. This measurement set-up includes three independent measurements of the deflection: the inclinometer measurement, a traversing laser time-of-flight sensor and a traversing plane of light sensor. This arrangement enables an objective verification of the reconstruction methods. In this set-up all the variables occurring in measurements using inclinometers, such as numerical errors, difficulty of zero-measurements, data transfer, etc., are taken into account.

4.1 Laboratory Set-Up

The following requirements were placed on the physical assembly:

1. easily measure different deformations,
2. mount inclinometers onto the structure,
3. realize a reference measurement.

For the representation of the beam which is deflected, a metal sheet (200cmx15cmx2cm) was chosen. This had the following reasons:

1. low bending stiffness;
2. easy to restrain;
3. ease of mounting elements onto it.

Two additional aluminium U-beams were mounted onto the structure to ensure an optimal bending as well as the application of the *Light Sectioning Method* (LSM). This assembly was mounted onto an aluminium profile on one end; the other end can either hang loosely or can be supported by two bearings which can be placed at any position along the beam. 14 inclinometers were mounted onto the metal sheet at approximately equidistant spacing. These inclinometers measure the gradient at each respective point, delivering the gradient measurement data $\dot{\boldsymbol{y}}_{\boldsymbol{m}}$.

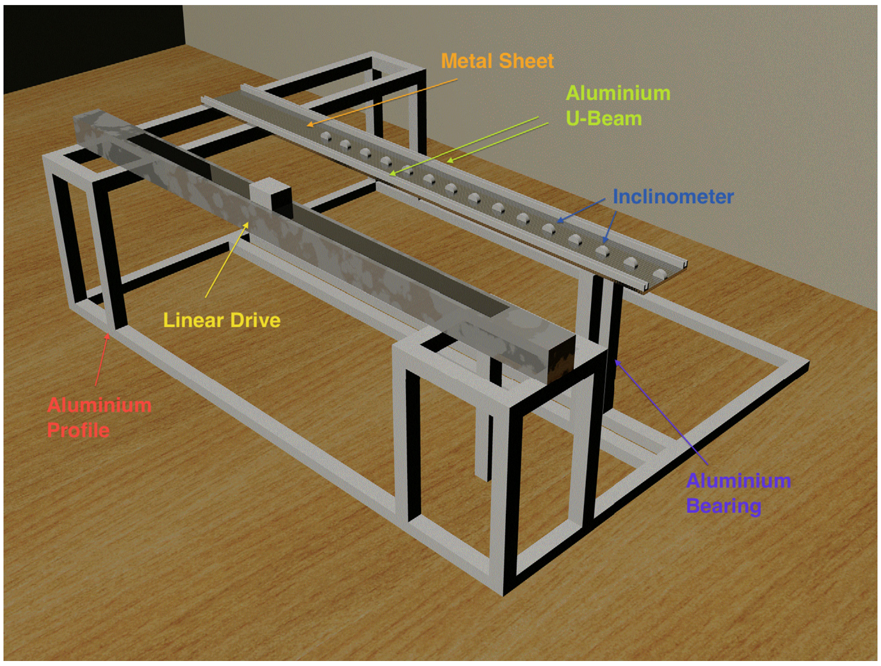


Figure 4.1: 3D-model of assembly. To the left the linear drive is illustrated. With the help of the linear drive the reference measurements LSM/ *Time of Flight Measurement* (TFM) are performed. In the middle one can see the metal sheet with the two aluminium U-beams on each side. The half-cylinder elements represent the inclinometers which are mounted onto the metal sheet at equidistant spacing. The metal sheet is supported with a bearing on one end and fixed onto the aluminium profile on the other end. The support can be shifted along the metal sheet at any desired position.

The inclinometer data is read out by the so-called *DAMOS SQL* Program provided by Geo-Data. It reads the data from the inclinometers and stores it into an SQL Database. To read the Data saved in the SQL Database, a connection between MATLAB and the SQL Database had to be established. For that, and ODBC data source had to be created. Windows has an *ODBC Data Source Administrator* with which it is possible to establish such a connection. The Remote Control for TCP/IP and the Named Pipes in the Microsoft SQL Server Surface Area Configuration had to be enabled. The TCP/IP Connection Protocol had to be enabled in the SQL Server Configuration. Further more the mode of *Dynamic Ports* was switched to *Static Ports*. One specific port was chosen for this communication and added to all Anti-Virus programs, yet the *Anti-Hacker Module* of *Kaspersky Anti Virus* had to be shut down completely. The port number also had to be entered in the client configuration in the Microsoft interface for establishing an ODBC data source.

or can be supported by two bearings which can be placed at any position along the beam. 14 inclinometers were mounted onto the metal sheet at approximately equidistant spacing. These inclinometers measure the gradient at each respective point, delivering the gradient measurement data $\dot{\boldsymbol{y}}_{\boldsymbol{m}}$.

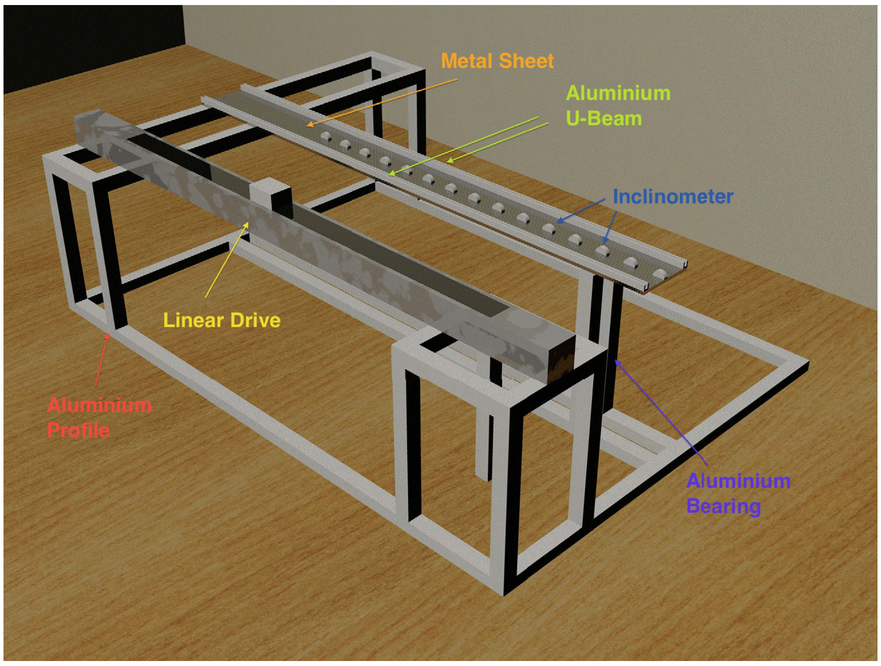


Figure 4.1: 3D-model of assembly. To the left the linear drive is illustrated. With the help of the linear drive the reference measurements LSM/ *Time of Flight Measurement* (TFM) are performed. In the middle one can see the metal sheet with the two aluminium U-beams on each side. The half-cylinder elements represent the inclinometers which are mounted onto the metal sheet at equidistant spacing. The metal sheet is supported with a bearing on one end and fixed onto the aluminium profile on the other end. The support can be shifted along the metal sheet at any desired position.

The inclinometer data is read out by the so-called *DAMOS SQL* Program provided by Geo-Data. It reads the data from the inclinometers and stores it into an SQL Database. To read the Data saved in the SQL Database, a connection between MATLAB and the SQL Database had to be established. For that, and ODBC data source had to be created. Windows has an *ODBC Data Source Administrator* with which it is possible to establish such a connection. The Remote Control for TCP/IP and the Named Pipes in the Microsoft SQL Server Surface Area Configuration had to be enabled. The TCP/IP Connection Protocol had to be enabled in the SQL Server Configuration. Further more the mode of *Dynamic Ports* was switched to *Static Ports*. One specific port was chosen for this communication and added to all Anti-Virus programs, yet the *Anti-Hacker Module* of *Kaspersky Anti Virus* had to be shut down completely. The port number also had to be entered in the client configuration in the Microsoft interface for establishing an ODBC data source.

Two measurement methods were chosen for the verification of the measurements and computations performed by inclinometers and reconstruction methods, respectively:

1. (Light) Time of Flight Measurement (TFM),
2. Light Sectioning Method (LSM).

Both measurement methods require the use of a linear traversing stage. Therefore a linear drive is mounted parallel to the deflection beam onto the aluminium profile. The linear drive is controlled by an OPC interface, which is controlled by MATLAB. See Figure 4.2 for a picture of the assembly and Figure 4.1 for a 3D-drawing of the assembly. For the TFM the sensor is constantly displaced along the beam, i.e. the sensor is moving while the measurement is performed. For the LSM the camera and laser are stopped at various positions along the beam, because the image acquisition is time delayed and one would not be able to determine at which position the image was taken, i.e. with which x-coordinate the computed y-coordinate corresponds.

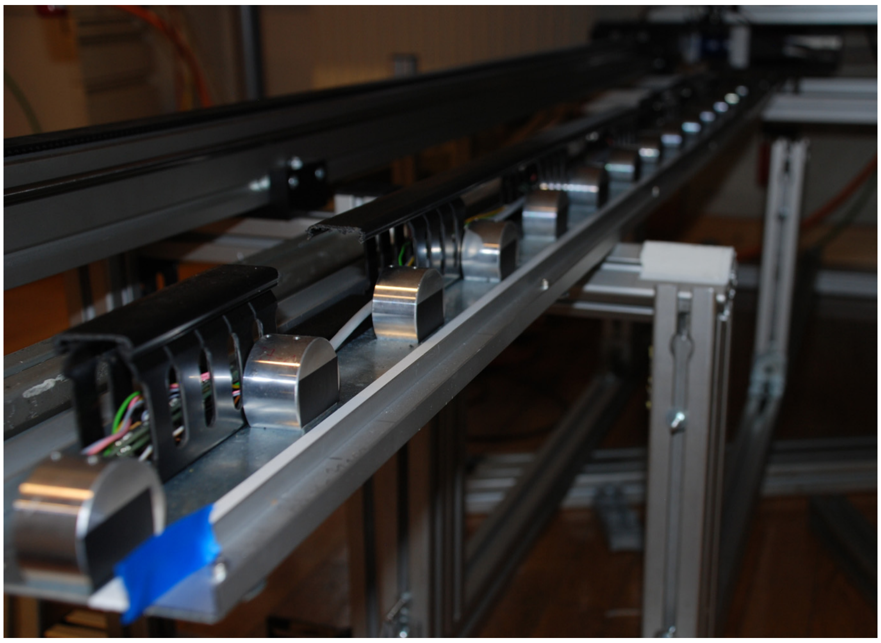


Figure 4.2: This image shows the metal sheet on which the inclinometers are mounted onto. To the left one can see the linear drive, also mounted onto the aluminium profile.

4.2 (Light) Time of Flight Measurement

The first reference measurement chosen was the time of flight measurement. For this measurement method a *Light Travel Time Sensor* (see Table 4.2 for specific information of the sensor) is used.

Name	Light Travel Time Sensor
Original Name	Lichtlaufzeitsensor
Product Number	X1TA100QXT3
Distance	0.1 ... 10.2m $\pm 1mm$
Structural Shape	81x55x30mm
Light Type	Laser (red)

Table 4.1: Detailed information of light travel time sensor.

This type of sensor detects the distance between an object and itself. It consists of a light source emitting light pulses at a constant rate, and a light detector, detecting the delay of the emitted light. The measurement uses the fact, that the distance of an object is proportional to the reflection time by the speed of light:

$$d = 3 \cdot 10^8 \, t \qquad [m], \tag{4.1}$$

whereby d is the distance between the sensor and the object; t is the time covered to travel from the sensor to the object and back. The sensor is connected to the OPC, which is connected to MATLAB, and the distance is observed at a constant rate of time while the sensor is moved by the linear drive (see Figure 4.3 for a better overview).

One of the problems encountered with this sensor was the accuracy. The sensor is able to measure distances up to 10m. The accuracy of a measurement lying in a distance range of 10m is $\pm 1mm$. For this kind of measurement this is a very satisfying result. The measurement at hand, though, lies in distances up to 30mm, sometimes varying only in millimetres. Although the output signals can be adapted for this distance, the accuracy cannot. This means that deflections varying only by a few millimetres cannot be detected with sufficient accuracy. Because of this, the LSM was chosen as an alternative reference measurement.

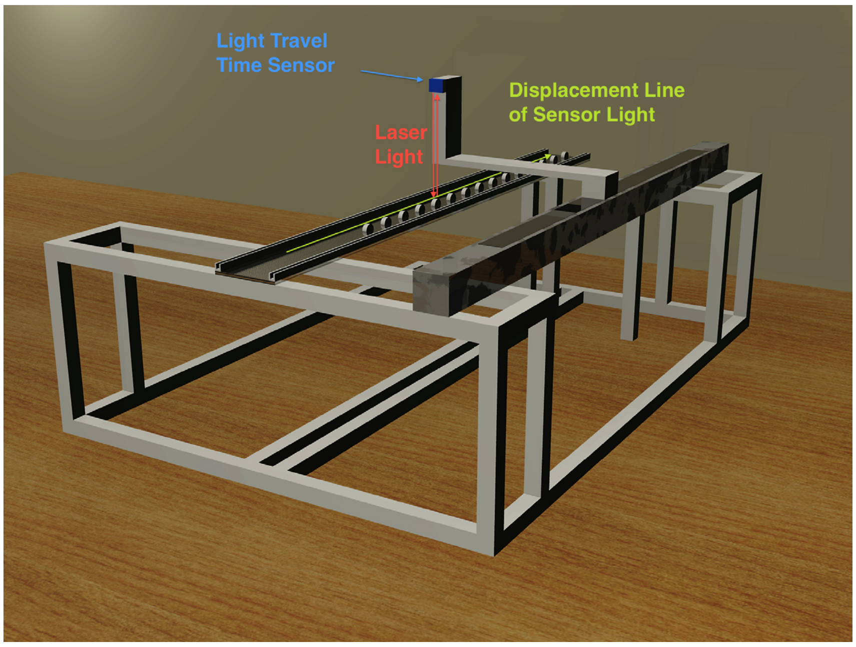


Figure 4.3: 3D Model of the assembly for the TFM set-up. The sensor is displayed as a blue box. It was necessary to mount the sensor at a certain distance from the object to be measured. The red line depicts the laser light emitted and observed by the sensor. During the measurement the sensor is displaced by the linear drive and the distance between the sensor and the metal sheet is measured at a constant rate. The green arrow shows the position and direction of the measurement points.

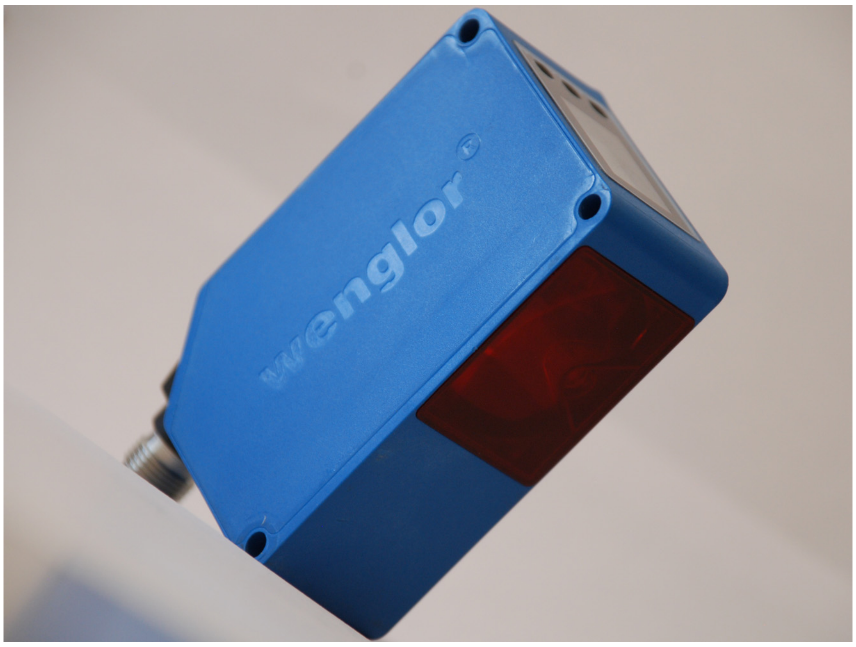


Figure 4.4: *Light Travel Time Sensor* X1TA100QXT3 from *Wenglor*.

4.3 Light Sectioning Method

4.3.1 Set-Up

The *Light Sectioning Method* (LSM) is an image processing method. The LSM can be adapted for a 2D or 3D application. It is a kind of structured lighting, which implies that a great deal of information is known about the light source as well as of the object [14]. In the current task the position of the laser line, the object to be measured as well as its position is known and can be adapted for application of the LSM, which allows us to use this method for measuring the deflection of the metal sheet. The light source is a laser (see Table 4.3.1 for detailed information of the laser) which projects a plane of light. This plane of light results in a line once projected onto the plane, i.e. the measurement object. The camera (see Table 4.3.1 for specific data) is set up in a manner that it can view the projected laser line on the object, see Figure 4.6 for the schematic alignment and Figure for the 3D model of the assembly for the LSM.

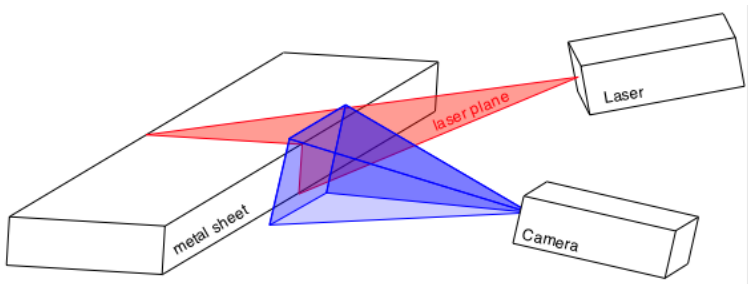


Figure 4.5: Schematic set-up of the LSM. Here the plane of light produced by the laser (red) is projected onto the thin side of the metal sheet; this results in a line once projected onto the metal sheet. The camera is mounted in a way that it can view this projected line perfectly. The camera's range of vision is displayed in blue.

Name	LASIRIS SNF Laser Class II
Weigth	65g
Diode power	1 to 200 mW
Wavelength	635 to 1550 nm
Intensity Distribution	Uniform (non-Gaussian) lengthwise
Input Voltage	4.8 - 6.5 V_{DC}

Table 4.2: Detailed information of laser LASIRIS SNF.

Geometrically the image observed by the camera is a perspective image. The camera is positioned so that the optical axis is normal to the surface being observed. For this reason the projective distortion can be neglected. To ensure a more precise image, an interference filter matched to the wavelength of the laser is used for the image acquisition. For each point at which an image is to be taken, the linear drive stops for a specific period of time for the camera to take a picture of the laser line. The idea behind taking the images along

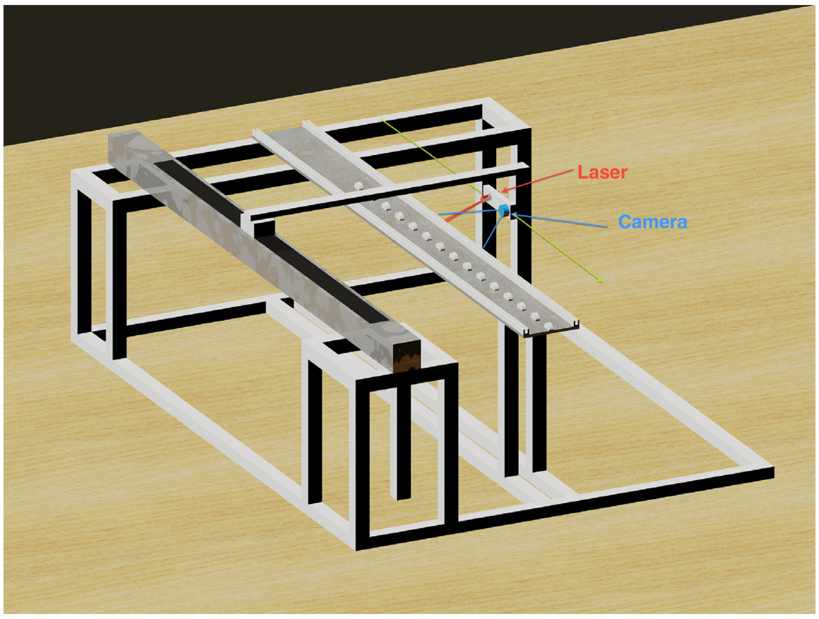


Figure 4.6: The 3D Model of the LSM. The laser and the camera are mounted onto the linear drive which is itself mounted onto the aluminium profile, parallel to the metal sheet. The line projected by the laser, as well as the laser itself is given in red. The camera is displayed in blue. The green arrow shows the direction and the positional displacement of the camera-laser assembly.

Name	Logitech QuickCamPro 9000
Device Type	Web camera
Gross Sensor Resolution	2MP
Camera Type	Color
Focus Adjustment	Auto

Table 4.3: Detailed information of laser LASIRIS SNF.

the beam is that for each image taken, the camera position only moves in one (defined as the x-direction) , however it does not move in the other (y-direction). This means that, once a specific conversion factor was defined, the top of the laser line in each image corresponds with the aluminium U-beam's top at the respective x-coordinate; which is the deflection of the entire assembly. This means it is necessary to find the top of the laser line in each image to be able to find the beam's deflection. All the images taken are stored in a MATLAB structure for further processing. The entire image processing performed can be described as

$$A^* \xrightarrow{\text{classification}} A \xrightarrow{\text{normalization}} N \xrightarrow{\text{binarization}} B \xrightarrow{\text{segmentation}} S \xrightarrow{\text{rotation}} R \xrightarrow{\text{center determination}} y_{pixels}, \tag{4.2}$$

and is explained in detail in the following Section.

4.3.2 Image Processing

To actually use the data acquired by the camera, information needs to be preprocessed and extracted from each image. To do so, various image processing steps are applied. The process itself is of hierarchical nature; a workspace which contains a number of *image objects* $A^* = \{\mathrm{R}_1, ..., \mathrm{R}_m\}$ is defined. Then specific necessary image processing methods are applied to each one of the images. The processes described in this section are used for computations for the LSM described in Section 4.3.1. Firstly, the images are classified for their relevance, delivering the set of the *regions of interest* $A = \{\mathrm{R}_1, ..., \mathrm{R}_n\}$, whereby $A \in A^*$ and $n \leq m$, where n and m are the number of containing elements. The second step is the *normalization* of the image. This is a method to distribute the brightness of the image over the whole available range of pixel intensities. After the contrast was adapted, a threshold is computed and applied to binarize the image. This is done to facilitate and speed up further computations. For noise suppression a Morphological Operator is used. The center of gravity is computed to find a point in the image, which can be used for curve fitting. The entire process used in the LSM can be described as

$$A^* \xrightarrow{\text{classification}} A \xrightarrow{\text{normalization}} N \xrightarrow{\text{binarization}} B \xrightarrow{\text{segmentation}} S \xrightarrow{\text{center determination}} y_{pixels} \tag{4.3}$$

Region of Interest Classification

The *Region of Interest* (ROI) of an image is a certain element $A = \{\mathrm{R}_1, ..., \mathrm{R}_n\}$ of an image $A^* = \{\mathrm{R}_1, ..., \mathrm{R}_m\}$, which contains the information used for computation purposes, whereby

$$A \in A^* \qquad and \qquad n \leq m, \tag{4.4}$$

where n and m are the number of containing elements. This procedure can be described:

$$A^* = \{\mathrm{R}_1, ..., \mathrm{R}_m\} \xrightarrow{\text{classification}} A = \{\mathrm{R}_1, ..., \mathrm{R}_n\}. \tag{4.5}$$

There are various ways of defining a ROI. It can be found automatically by some kind of geometric or color attribute using image segmentation, or, by hand defining specific coordinates, using a-priori knowledge of the image. The data in x-direction (y-direction) is stored in columns (rows); see Figure 4.7 for further explanations. When classifying the image by hand the following need to be defined:

1. the minimum x_{min}, y_{min} and
2. the maximum x_{max}, y_{max} of the ROI.

Once these boundaries are defined, the image can be cropped to the desired size. An example for this step is given in Figure 4.8. In Figure 4.8.a the original image is shown. In Figure 4.8.b one can see the classified image - the ROI. The MATLAB implementation for this step is given in Source Code 4.1.

```
%Define maxima and minima in pixels
xmin = 700;
xmax = 760;
ymin = 210;
ymax = 450;
%
for i = 1 : noimages
    %define current image as a matrix on its own
    image_current = img{i};
    %crop current image and store into a new structure
    img_crop {i} = image_current(ymin:ymax,xmin:xmax);
end
```

Source Code 4.1: Code for the image cropping.

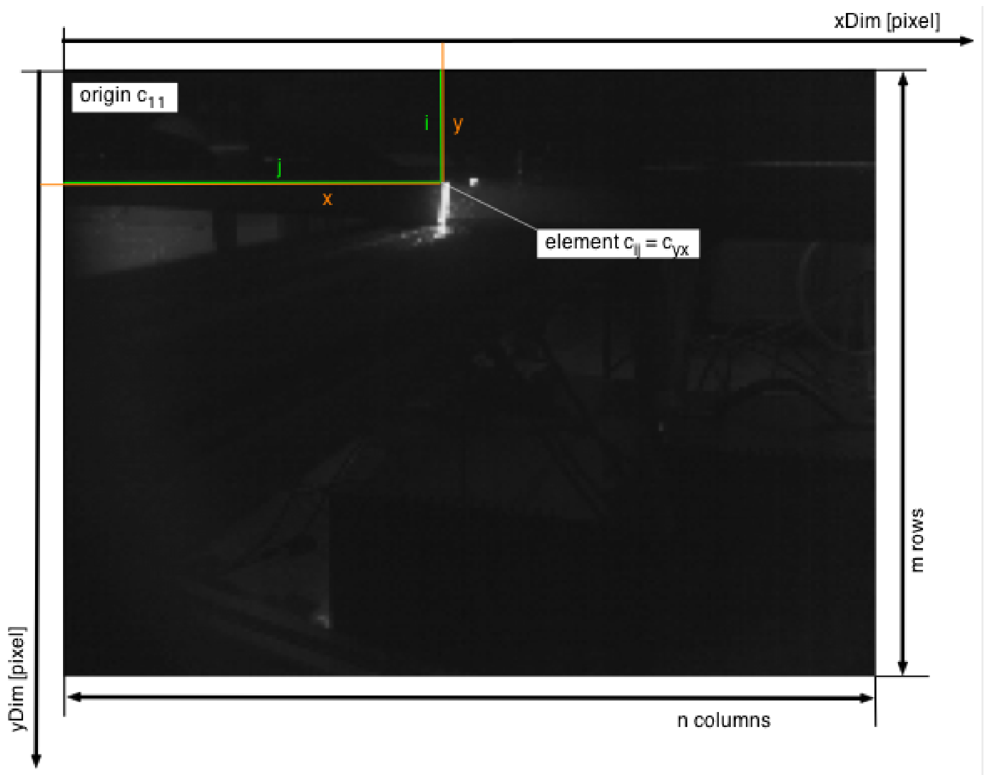


Figure 4.7: Each matrix has m rows, representing the y-dimension, and n columns, representing the x-dimension. The corrrepsonding indecis are i and j. Each element, i.e. each pixel, has a defined position at row i and column j as well as an 8 bit brightness ranging from zero to 255. The origin of the image, $c_{1,1}$ is in the upper left corner.

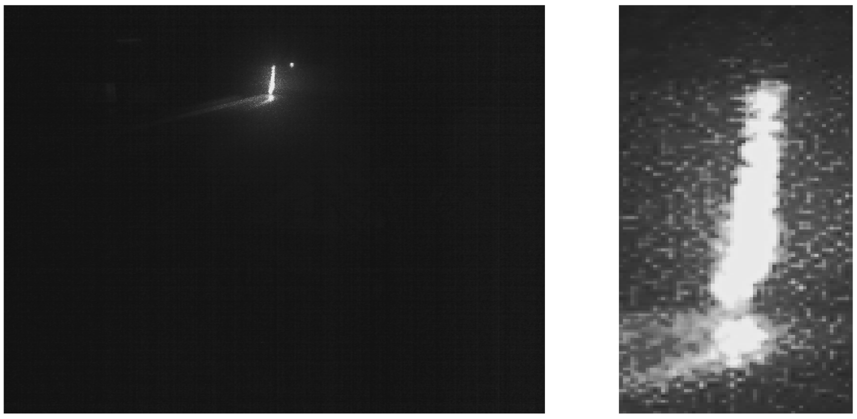


Figure 4.8: Example for the Classification Process. In a) image A^* is given. The image A, after classificaiton, is given in b).

Normalization

Normalization is a simple and efficient way to distribute the brightness over the whole available range of pixel intensities. The normalization yields the normalized image N:

The image A has an 8-bit encoding, i.e. normally the value of the pixels reach from zero to 255. The problem is that due to lighting, not the entire range is utilized. The goal of normalization is to have values between zero and one, where one represents the greatest intensity of the original image and zero the lowest. Thus, the whole spectrum between zero and one is used. The computation of the normalization is represented by the equation

$$c_{i,j}^{(out)} = \frac{d_{max} - d_{min}}{g_{max} - g_{min}} \left(c_{i,j}^{(in)} - g_{min} \right) + d_{min}. \tag{4.6}$$

The MATLAB implementation is given in Source Code 4.3. In Figure 4.9 an example of this operation is given. It can be seen that the contrast was increased on the normalized image in Figure 4.9. After the image was normalized, the next step is the definition of a threshold value, to be able to divide the image into two sections - the white section being the section carrying important information, the black section being the background only.

```
%normalize the image
imgN = (dMax - dMin) / (gMax - gMin) * (G - gMin) + dMin;
```

Source Code 4.2: Code for image normalization.

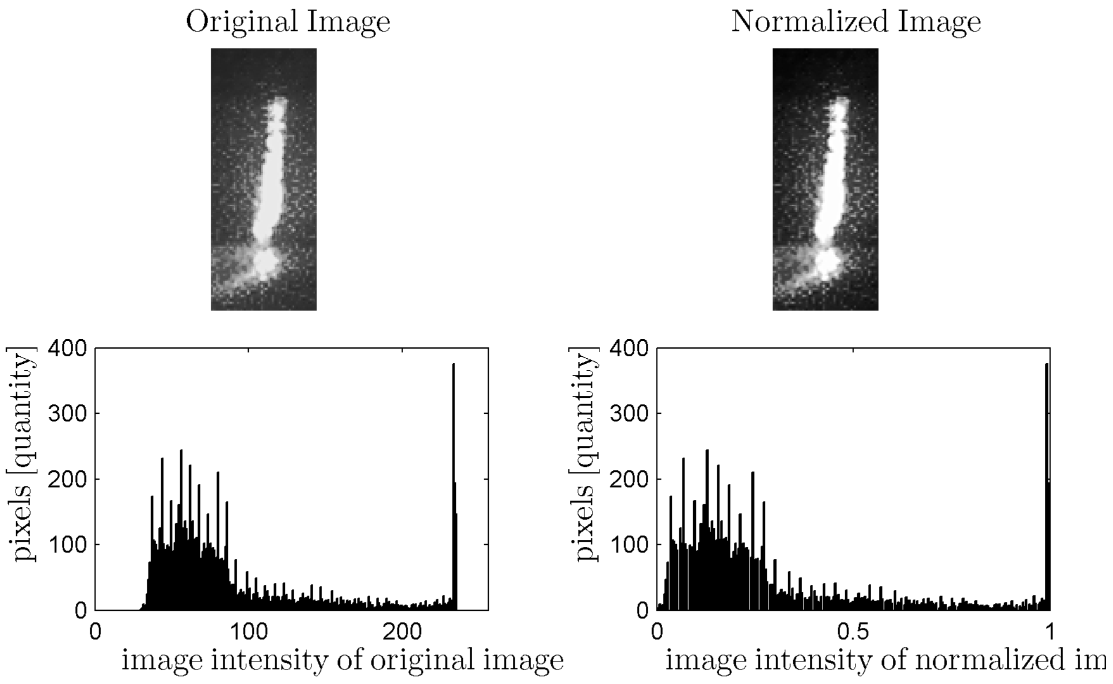


Figure 4.9: This example shows the original image to the left and the normalized image to the right. The original image A had lower contrast, hardly any values close to zero or close to 255. The normalized image has its data distributed between zero and one, with the smallest value being zero, the highest being 1. The histogram was computed using 250 evenly distributed bins.

Gray-Level Segmentation

Methods which are based on the determination of a threshold are used for *gray-level segmentation* or binary segmentation. Usually a threshold value is defined, which is used to binarize the image, generating a *bilevel* image. "A bilevel image should contain all of the essential information concerning the number, position, and shape of objects while containing a lot less information." [31] The process can be described as

$$N \xrightarrow{\text{binarization}} B. \tag{4.7}$$

This basic principle is described as

$$c_{i,j}^{(out)} = \begin{cases} 1 & \text{if } c_{i,j}^{(in)} \geq t \\ 0 & \text{else} \end{cases}$$

where $c_{i,j}^{in}$ is the image pixel at the ith column (x-direction) and jth row (y-direction). If this certain pixel value is higher than the value t - the threshold value - then the pixel value is changed to 1 (white). Is the input pixel lower than t, then the pixel value is changed to zero (black). This leads to a binary or monochrome image; an image consisting only of the values zero and one. While this can be a disadvantage because lots of information is lost, it can be very advantageous because the image is much easier to interpret. It can be used for binary morphology, construction of a mask or simply to identify specific objects in an image [26]. The task is to find an appropriate threshold value t, which delivers satisfactory results. There are various ways of finding a threshold value:

1. Self-defined Threshold, i.e., by hand,
2. Brightness Class Average,
3. Gaussian Curve Intersection.[1]

A *Self-defined Threshold* is the use of a threshold defined to fit a desired solution. "If some property of an image after segmentation is known a priori, the task of threshold selection is simplified, since the threshold is chosen to ensure that this property is satisfied." [34].

The threshold value is adapted iteratively until it fits a wanted solution. For the application at hand, a threshold value of $t_1 = 0.9$ was defined. The threshold value is displayed in cyan in Figure 4.10 and the image binarized with this value is shown in Figure 4.11.a. While this is an easy solution, it is not suitable for automation processes.

Brightness Class Average is the easiest algorithm to determine a threshold. It assumes that the histogram of the image is of bimodal nature; this means that the histogram features two clearly developed peaks. The histogram of the ROI shown in Figure 4.8 is given in Figure 4.10. The two peaks are abbreviated with I_{p1}, representing the darker pixels with a mean value close to zero, and with I_{p2}, representing the higher values, i.e. the brighter pixels in the image. The computation for the threshold is the average of the two peak values:

$$t = \frac{I_{p1} + I_{p2}}{2}. \tag{4.8}$$

Creating a threshold value t by using Option 2 is simple, however it does not fit my application as the image of the ROI has lots of scattered light, which needs to be removed. If the threshold is defined by the mean of the maximum (one) and the minimum (zero), too much information is removed (see Figure 4.11.b).

Gaussian Curve Intersection models the bimodal histogram as the sum of two Gaussian distributions. This is reasonable, since the data of each class should be normally distributed. the curves can be described as

$$y_1(x) = I_{p1}\mathrm{e}^{\{\frac{x-I_{m1}}{\sigma_1}\}^2} \quad \text{and} \quad y_2(x) = I_{p2}\mathrm{e}^{\{\frac{x-I_{m2}}{\sigma_2}\}^2},$$

$I_{m1,2}$ being the maxima of the curve. The task now is to find the intersection point of these two curves. The intensity at which the two curves intersect is then the *Gauss threshold* t_g. The Gaussian curves fitted to the histogram of the ROI, as well as all the thresholds computed are shown in Figure 4.10. In Figure 4.11.c the ROI binarized using t_g is given.

[1]There are further methods to binarize an image, this however exceeds the scope of this thesis.

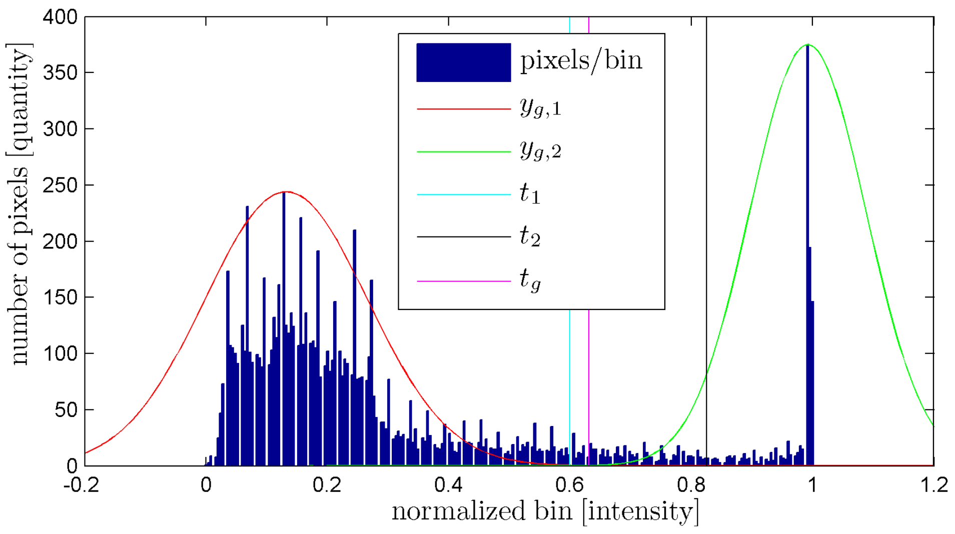
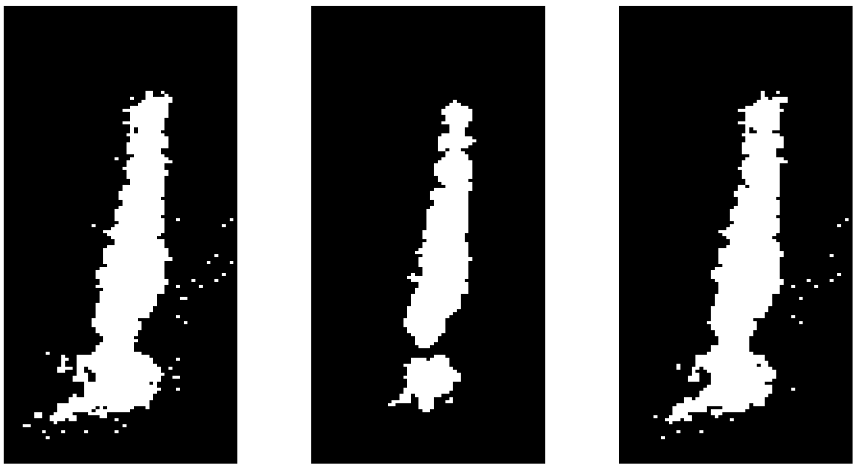


Figure 4.10: Histogram of the image N from Figure 4.8.b, computed using 250 bins, displayed in blue bars. There are two peaks to be observed. One at $I_{p1} = 0.13$ and one at $I_{p2} = 0.99$. This means the histogram is bi-modal. All threshold values computed are displayed. Threshold 1 is the *Self-defined Threshold*, displayed in cyan; Threshold 2 uses the *Brightness Class Average*, displayed in black; Threshold 3 is the threshold computed using the *Gaussian Curve Intersection*, given in magenta. This threshold was computed using the intersection of $y_{g,1}$ and $y_{g,2}$. $y_{g,1}$ and $y_{g,2}$ are normal distribution fits using the local mean, maximum and standard deviaiton values.

Once the threshold value is defined, the images are binarized using this threshold value, i.e. they are converted into an image consisting only of black and white pixels. In MATLAB this is implemented by using the *im2bw*-function. See Source Code 4.3 for the MATLAB implementation of applying a threshold.

```
%apply threshold tg
img_tg = im2bw(imgN,tg);
```

Source Code 4.3: Code for thresholding.



a) ROI binarized with Self-defined Threshold t_1=0.6

b) ROI binarized with Brightness Class Average t_2=0.82533

c) ROI binarized with Gaussian Curve Intersecti t_g=0.63131

Figure 4.11: ROI after binarization. Image a) uses Threshold-Method 1, i.e. the *Self-determined* threshold t_1 for binarization. The image T obtained by using *Brightness Class Average* method is displayed in b). In c) the image T computed using Threshold-Method 3, i.e. the *Gaussian Curve Intersection*.

Morphological Operators

Morphological Operators can thin, thicken, find boundaries etc. They can be used to find contours of an object in an image [34] or to extract more information of the objects in the image. Most morphological operators require two elements:

1. Structuring Element,
2. Object of Interest.

The structuring element defines the actual size and form of the operator. The process can be described as

$$B \xrightarrow[Structuring\,Element]{\text{segmentation}} S. \tag{4.9}$$

The object of interest is usually the image. The two basic types of morphological operators are *erosion* and *dilation*, whereby *erosion* is used for the reduction of an image, *dilation* for the enhancement of the object. For the current problem, the morphological operator *erode* is used to remove the noise in the image and therefore to enhance the ability to find the top point of the *laser line* (LL). The difficulty here lies in finding the structuring element which removes as much data as necessary, without throwing away too much information. The structuring element chosen for the current problem is a $3x3$ vector of ones. The MATLAB code for the *erosion* of the image previously produced by application of the *Gauss* threshold is given in Source Code 4.4. The image before and after *erosion* is given in Figure **??** and 4.12.

```
SE = ones(3);
img_SE = imerode(img_tg,SE);
```

Source Code 4.4: Code for erosion.

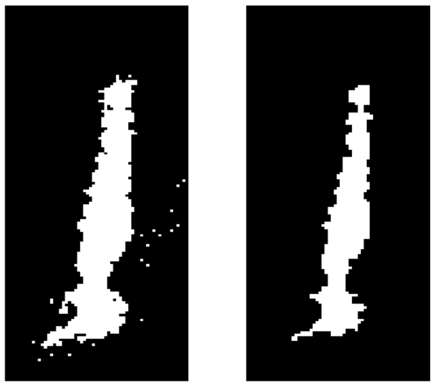


Figure 4.12: Segmentation Example; in a) one can see the original image before segmentation. In b) Image S, ROI after applying the morphological operator *erosion* using a 3x3 matrix full of ones as structuring element, is displayed.

Center Determination

To be able to fit a curve using a number of images taken at specific points $\boldsymbol{x}$, one needs to define a specific point in the image which is taken as a reference point for the curve computation. The process here can be described as

$$S \xrightarrow{\text{centerdetermination}} \boldsymbol{y_{pixels,COG}}. \tag{4.10}$$

There are various ways of finding such a point, as there are numerous different properties encountered in images. One very common approach is using the first moment, commonly called the *Center of Gravity* (COG) of the image. The "Centroid of an area is a point in a plane area such that the moment of area about any axis through that point is zero" [4]. The COG, also referred to as the *centroid*, of a plane is the arithmetic mean position of all the points in the shape. Hence, the coordinates $\boldsymbol{p}$ can easily be computed for any given area. The center of gravity for a binary image is computed as

$$\boldsymbol{p} = [x_c, y_c]^T = \left[\frac{\sum_{y,x \in R} (x \cdot c_{yx})}{\sum_{y,x \in R} c_{yx}}, \frac{\sum_{y,x \in R} (y \cdot c_{yx})}{\sum_{y,x \in R} c_{yx}} \right]^T, \tag{4.11}$$

where c_{yx} is the intensity of the pixel at the corresponding y- and x-position. Before this computation can be performed, the image is rotated by 5 degrees using the *imrotate* function, see Source Code 4.5 for the MATLAB implementation. This is done because the LL is not completely straight and thus the computation of the top of the LL using the COG would be incorrect.

```
degree = 5;
img_rot = imrotate(img_SE,degree);
```

Source Code 4.5: Code for rotation of the image.

In the case of a binary image the intensity of each pixel taken into account is one, which makes the determination of the COG a simple mean computation. The line in red shows the column of the COG. The code for the computation of the COG in the y- and x- direction is given in Source Code 4.6. In Figure 4.13 the COG was determined for the image given.

```
% find the position of all non-zero entries in image_rot
[YCoord, XCoord] = find(image_rot);
%find the mean of all non-zero entries
COGx = mean(XCoord);
COGy = mean(YCoord);
```

Source Code 4.6: Code for synthesis of COG.

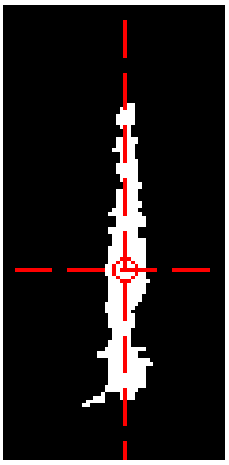


Figure 4.13: The COG in the x (columns)- and y (rows)-direction.

4.3.3 Line Fitting

The vector $\boldsymbol{y_{pixels,COG}}$ consists of the position of the COG in each image. With this the position of the first non-zero entry in the corresponding row-vector (y-dimension) is determined for each image. This is done using the *find* command in MATLAB, see Source Code 4.7.

```
top_img = find(im(:,COGx)>0,1,'first');
```

Source Code 4.7: Code for synthesis of the position of the top of the LL.

This process is performed on all the images recorded during the measurement process. The result is stored in a vector $\boldsymbol{y_{pos}}$ which already represents the deflection of the beam in pixels. It is known that the position of the top of the LL in the image corresponds with the real top of the LL on the aluminium U-beam. Therefore the change of this specific point in the y direction must cohere with the metal sheets deflection. The scaling factor from pixel to millimetres is computed as

$$c_{conv} = \frac{l_{LL,lab}}{l_{LL,img}} \qquad \left[\frac{\mathrm{mm}}{\mathrm{p}}\right], \tag{4.12}$$

where $l_{LL,lab}$ is the length of the LL measured in the laboratory, in millimetres, and $l_{LL,img}$ is the length of the LL in pixels. Knowing this conversion vector, the deflection in millimetres can be computed as

$$\boldsymbol{y_{mm}} = \boldsymbol{y_{pos}}\, c_{conv} \qquad [\mathrm{mm}] \tag{4.13}$$

To start at the position $\boldsymbol{y}(0) = 0$, the following computation is performed:

$$\boldsymbol{y_{def}} = \boldsymbol{y_{mm}} - \boldsymbol{y_{mm}}(0) \qquad [\mathrm{mm}]. \tag{4.14}$$

4.3.4 Experiments

For the experimental verification, 6 measurements were performed, whereby 2 of these 6 measurements serve as a Zero- or Reference Measurement. Two different deflections were chosen for the other 4 measurements. The support used for the current measurements was placed at $x = 1198$ $[mm]$, whereby $x = 0$ $[mm]$ is the position at which the metal sheet is restrained onto the aluminium profile.

While putting these measurements into practice, the following difficulties had to be solved:

A major difficulty encountered during calibration of the three measurement systems to a single coordinate system was that the beam has a significant deflection under its own weight before any additional weights were added. Consequently, a true zero measurement for the inclinometers where the beam is not subject to any deflection was not possible. The minimal deflection was considered to be the zero measurement and deflections, inclinations, were measured relative to this position.

For the LSM, the camera, as well as the laser, can be adapted for each measurement to fit best the current case, i.e. the scale of deflection. For example, for large deflections, the angle at which the beam reflects the LL changes in respect to the camera as the deflection grows. As a result, the camera may not detect the LL very well and the measurement might be useless. This is why the camera is put into a different position from when the deflection is greater. However, after adapting the camera, the ROI has to be newly defined. When defining the ROI one must not neglect the fact that reflections can occur during measuring. For example, the reflections generated by the inclinometers mounted onto the metal sheet, as well as the reflections of the metal sheet and of the bearings, disturb the data and produce noise in the measurement data.

Measurement 1

To generate the deflections easily, a weight of 1.1 kg was fastened onto the metal sheet. It was placed on two different locations along the beam, Position 1 and 2.

Reference Measurement The first measurement performed is the Reference Measurement. This measurement is performed for calibration purposes concerning inclinometers. Additionally, it is useful to use a zero measurement to ensure the consistency of both measurement systems used. Once the metal sheet was put into place, the inclinometer values are set to zero using the *Auto Zero*-button of the DAMOS-SQL program. In Figure 4.14 the deflection chosen for the Reference Measurement is given. This deflection is measured using the LSM.

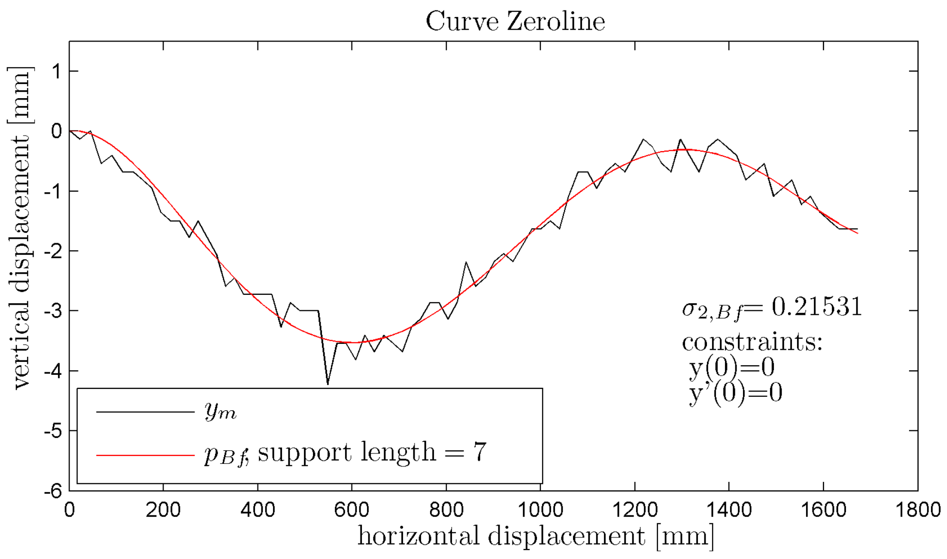


Figure 4.14: In black the data received using the LSM is given. A polynomial was fitted to this data using Polynomial basis functions and the constraints $\boldsymbol{y}(0) = 0\,[mm]$ and $\dot{\boldsymbol{y}}(0) = 0\,[mm]$. The error of this curve with respect to the orginal data shows a standard deviation of $0.215\,[mm]$.

Measurement Weight at Position 1: To change the deflection quickly and without having to move around the metal sheet too much, a weight was placed onto the metal sheet at $x = 1672\,[mm]$, which is at the end of the metal sheet. The deflection can be observed in Figure 4.15. To attune both measurement systems, the curve obtained by the Reference Measurement is subtracted from the curve of the current deflection. The deflection computed in this manner represents the deflection which is described by the inclinometer values for the current position of the weight (see Figure 4.16 for this newly computed deflection). For further computations only the fitted polynomial curves of the respective measurements are used.

The inclinometer data is imported using the SQL-Database via MATLAB. The required information is extracted from the matrix delivered by the SQL-Database and put into a vector $\dot{\boldsymbol{y}}_{\boldsymbol{m}}$. Then Reconstructions 1 and 2 are applied using the constraints $\boldsymbol{y}(0) = 0\,[mm]$ and $\boldsymbol{y}(1198) = 0\,[mm]$. The inclinometers are not placed onto the beam at position zero, because it is assumed that the inclination is zero there. However, when placing constraints onto the basis functions for reconstruction, the constraint for the deflection at position zero has to be placed there. This is why an additional, artificial inclination of value zero is included in the measurement vector at position zero. This way the constraint of $\boldsymbol{y}(0) = 0\,[mm]$ is placed at the correct position. The constraint of $\dot{\boldsymbol{y}}(0) = 0\,[mm]$ was not placed onto the reconstruction. This is done because the distance between the artificial inclinometer value at $x = 0\,[mm]$ and the first measured inclinometer value at $x = 136\,[mm]$ is too great. If the constraint of $\dot{y}(0) = 0\,[mm]$ were applied, the second point at $x = 136\,[mm]$ would be forced to be at position zero as well, which is incorrect. The results of these reconstructions are plotted on top the curve given in Figure 4.16. See Figure 4.17 for the reconstructions.

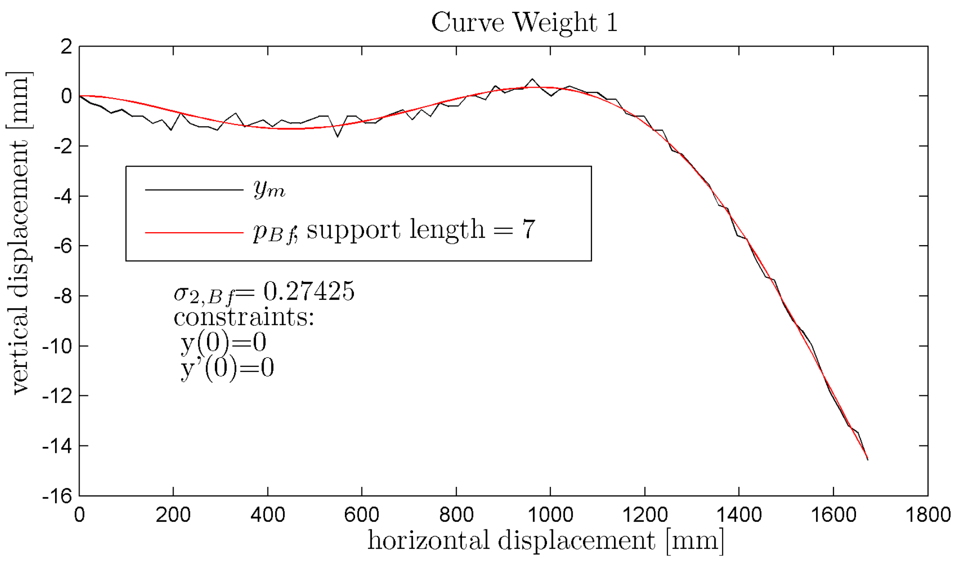


Figure 4.15: In black the data received using the LSM is given. To obtain this deflection a weight was placed at $x = 1672\,[mm]$. A polynomial was fitted to this data using Polynomial basis functions and the constraints $\boldsymbol{y}(0) = 0\,[mm]$ and $\dot{\boldsymbol{y}}(0) = 0\,[mm]$. The error of this curve with respect to the data received from the camera shows a standard deviation of $0.274\,[mm]$. The entire deflection has a range of about $15\,[mm]$, which means the fit is about 1.8% accurate.

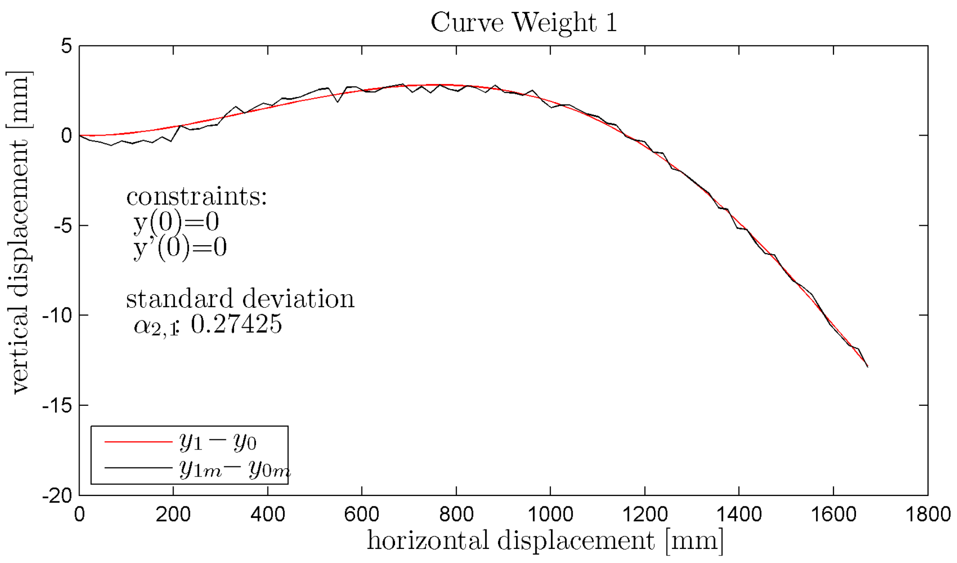


Figure 4.16: The deflection given in red is the calibrated curve, whereby the weight of $1.1kg$ is positioned at the end of the beam. The inclination at each particular point should be the data measured by the respective inclinometer. The standard deviation of the fitted curve amounts $0.274\,[mm]$.

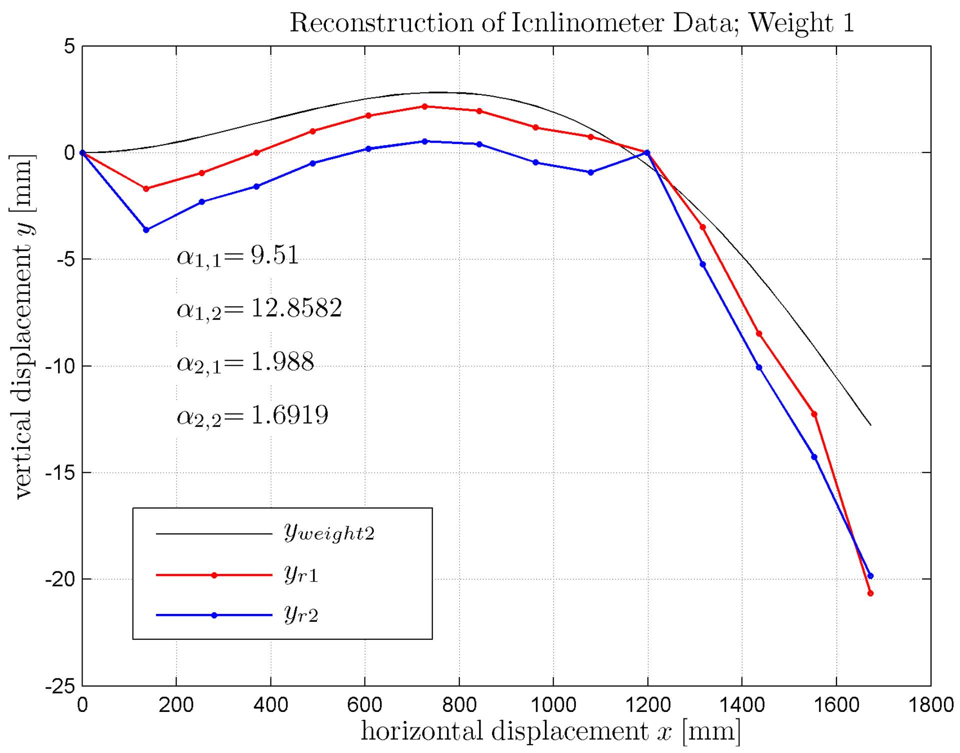


Figure 4.17: In black, one can observe the curve obtained by the LSM for the current deflection. Reconstruction 1 and 2 are given in red and blue, respectively. The norm as well as the standard deviation of the error vector of Reconstruction 1 is much higher for this deflection.

The reconstructed curve follows the same form as the curve computed using the LSM, though with what seems an offset of about 2 [mm]. At position $x = 1198\,[mm]$ the reconstructed curve deviates from this pattern. This is because a positional constraint of $y = 0\,[mm]$ at $x = 1198\,[mm]$ is put onto the reconstruction. This forces the reconstructed curve to be zero at this exact point, which it does. Additionally to an offset, it can be seen that at both end points the reconstruction values show a great error. This error occurred because the number of inclinometers used is not sufficient for this kind of deflection. The deformations presented here have a range of only $2\,[cm]$, whereby the beam itself has a length of about $1.7\,[m]$. This demands a very high accuracy of the inclinometers as well as the LSM. To remedy the offset as well as the great error at both end points one can either use more inclinometers for monitoring of the beam or place the constraints on the interpolating function and not on the reconstructed points.

Measurement Weight at Position 2: The same weight used in the previous section was used at position $x = 739\,[mm]$, resulting in a new deflection. This deflection is given in Figure 4.18. The curve generated this way has a range of about $7\,[mm]$. This is much smaller than the curve from the previous section. The calibrated curve is given in Figure 4.19. The range of the calibrated curve is only $4\,[mm]$.

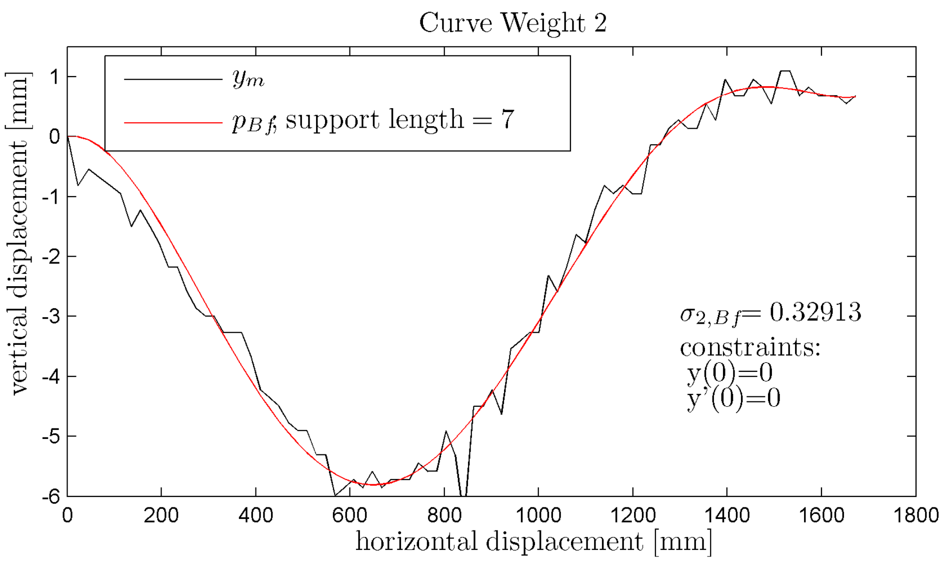


Figure 4.18: In black the data received using the LSM is given. A polynomial was fitted to this data using Polynomial basis functions and the constraints $\boldsymbol{y}(0) = 0\,[mm]$ and $\dot{\boldsymbol{y}}(0) = 0\,[mm]$. The error of this curve with respect to the data received from the camera shows a standard deviation of $0.329\,[mm]$. The entire deflection has a range of about $6.5\,[mm]$, which means the fit is about 4.4% accurate. The standard deviation for the fit through the measured data is $\sigma_{2,Bf} = 0.329\,[mm]$.

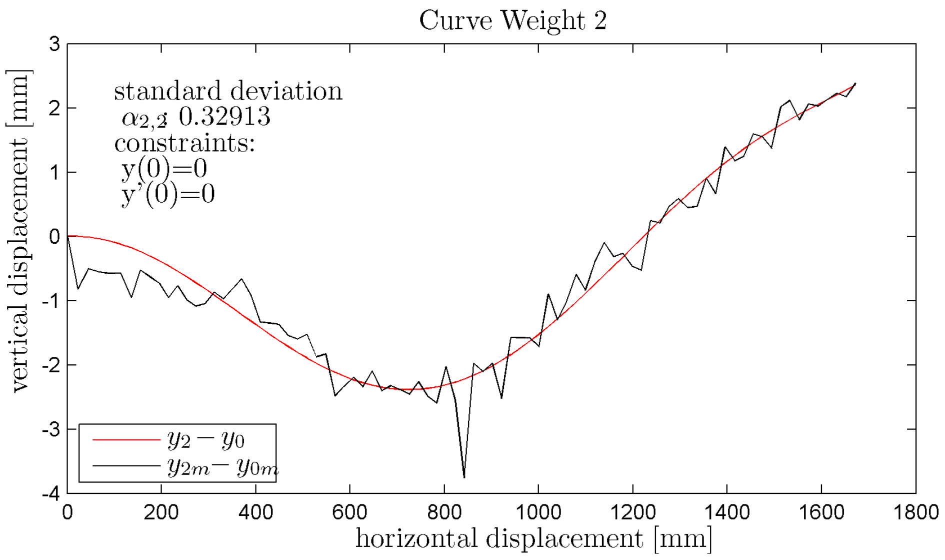


Figure 4.19: In black one can observe the data of the subtraction. In red the polynomial computed using both previously fitted polynomials is given. The range of the curve is only about $4.5\,[mm]$.

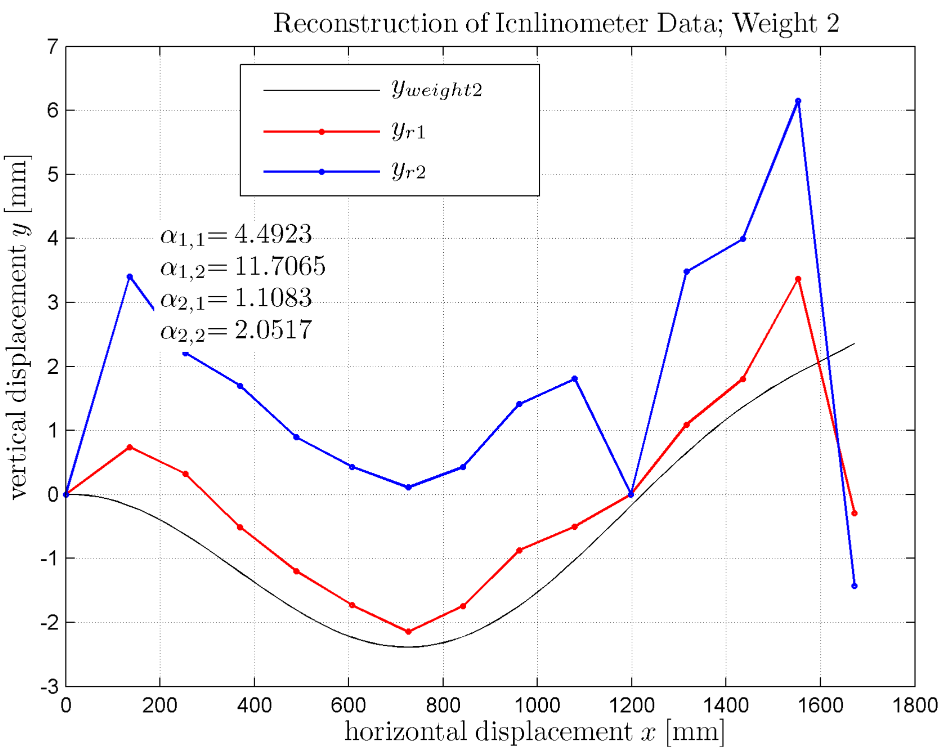


Figure 4.20: In black the curve obtained by the LSM for the current deflection is given. Reconstruction 1 and 2 are given in red and blue, respectively. The constraints $\boldsymbol{y}(0) = 0\,[mm]$ and $\boldsymbol{y}(1198) = 0\,[mm]$ were placed onto the reconstructions. Both reconstructions describe the curve's shape very well but they seem to be shifted. Great errors can be seen at either end of the curve. Furthermore a clear outlier can be found at $x = 1198\,[mm]$ which is the location of the support, and the location where the positional constraint of zero is placed onto the curve. The errors observed in this measurement are very similar to the ones from the first measurement. There seems to be some kind of systematic error. The errors can be diminished by appling one of the two solutions given in Section 4.3.4. For this deflection the norm as well as the standard deviation of the error vector of Reconstruction 1 is lower than Reconstruction 2.

Measurement 2

Reference Measurement: In Figure 4.21 the deflection of the Reference Measurement obtained using the LSM is given.

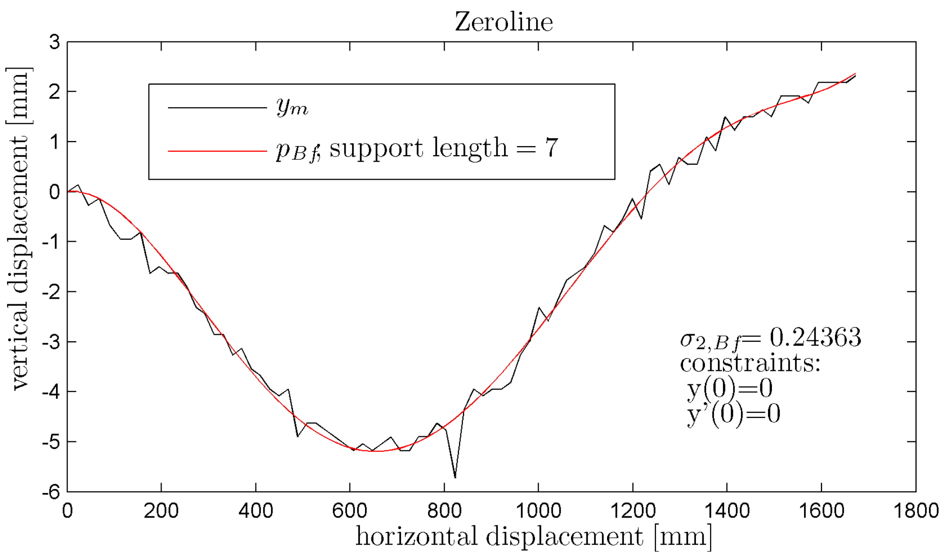


Figure 4.21: In black one can see the data received using the LSM. A polynomial was fitted to this data using polynomial basis functions and the constraints $\boldsymbol{y}(0) = 0\,[mm]$ and $\dot{\boldsymbol{y}}(0) = 0\,[mm]$. The error of this curve with respect to the orginal data shows a standard deviation of $0.244\,[mm]$.

Measurement Weight at Position 1: The weight was placed onto the metal sheet at $x = 1672\,[mm]$, which is the end of the metal sheet. The deflection is given in Figure 4.22. The calibrated curve is given in Figure 4.23. The reconstructed curve using the inclinometer data as well as the Reconstruction Methods 1 and 2 are given in Figure 4.24.

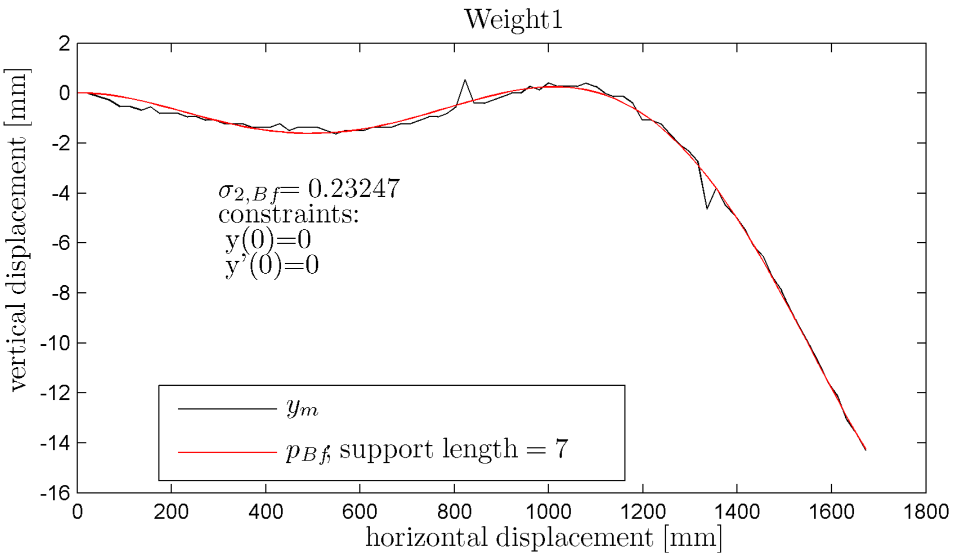


Figure 4.22: The data received using the LSM is given in black. A polynomial was fitted to this data using polynomial basis functions and the constraints $\boldsymbol{y}(0) = 0\,[mm]$ and $\dot{\boldsymbol{y}}(0) = 0\,[mm]$. The error of this curve with respect to the data received from the camera shows a standard deviation of $0.233\,[mm]$.

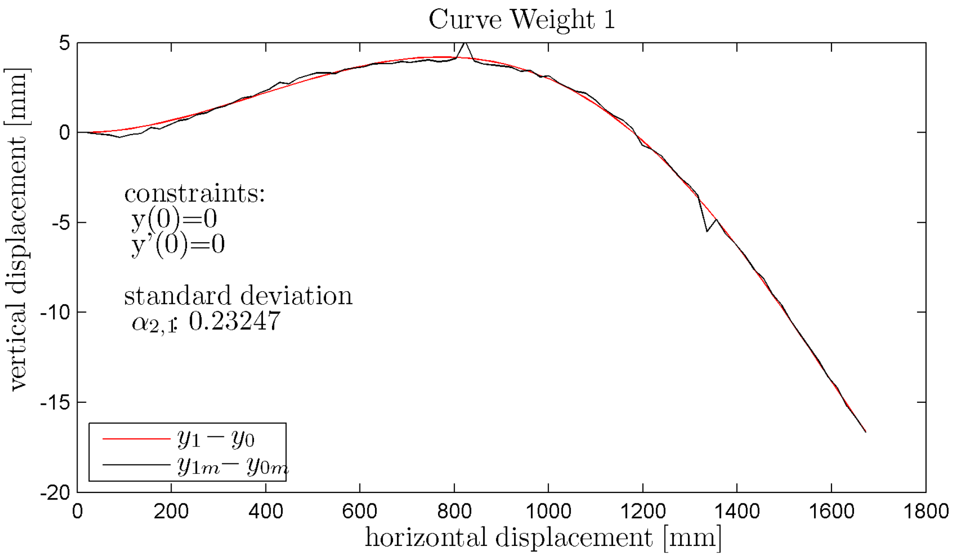


Figure 4.23: In black the data of the subtraction is given. The polynomial computed using both previously fitted polynomials is given in red.

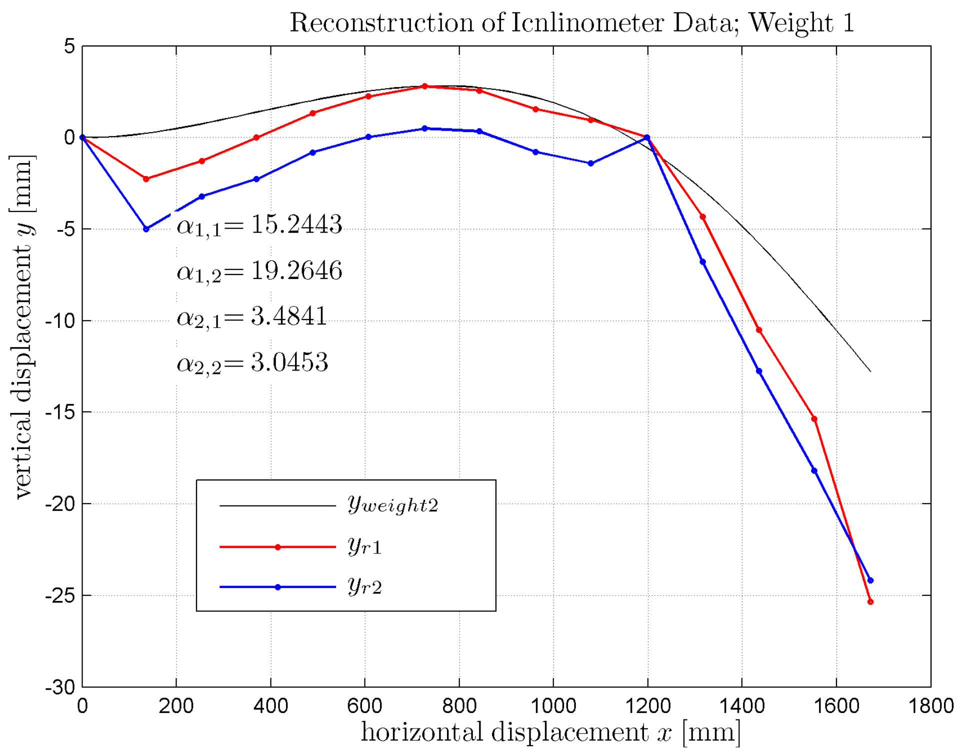


Figure 4.24: In black one can see the curve obtained by the LSM for the current deflection. Reconstruction 1 and 2 are given in red and blue, respectively. Both reconstruction methods used the constraints $\boldsymbol{y}(0) = 0\,[mm]$ and $\boldsymbol{y}(1198) = 0\,[mm]$. It can be seen that these constraints are perfectly fulfilled. There is an outlier at $x = 1189\,[mm]$. The end point of the reconstruction shows the greatest error. Both reconstructions follow the same pattern as the reconstructions from the previous sections.

Measurement Weight at Position 2: The same weight used in the previous section was used at position $x = 739\,[mm]$, resulting in the deflection given in Figure 4.25. The calibrated curve is given in Figure 4.26.

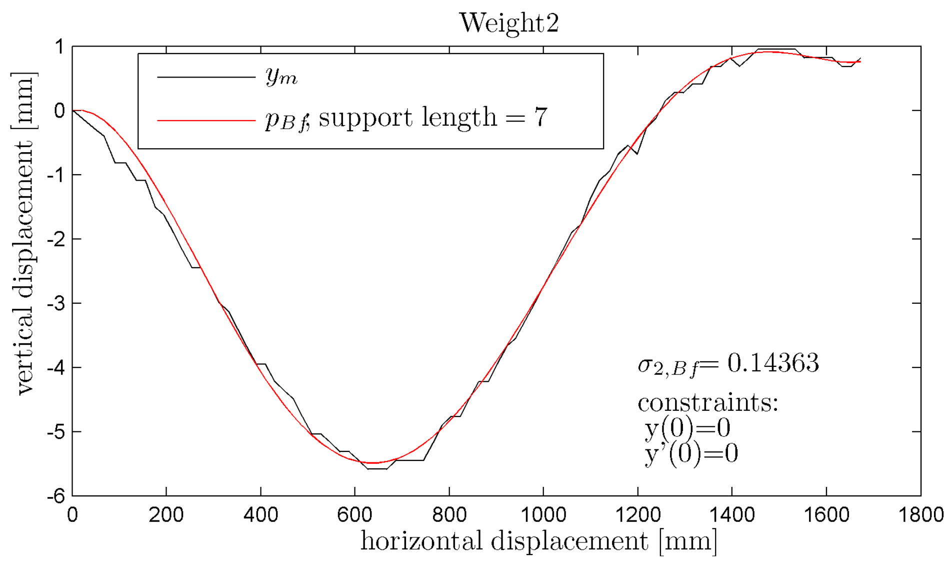


Figure 4.25: The data received using the LSM is shown in black. A polynomial was fitted to this data using Polynomial basis functions and the constraints $\boldsymbol{y}(0) = 0\,[mm]$ and $\dot{\boldsymbol{y}}(0) = 0\,[mm]$. The error of this curve with respect to the data received from the camera shows a standard deviation of $0.144\,[mm]$. The entire deflection has a range of about $6.4\,[mm]$, which means the fit is about 2.25% accurate.

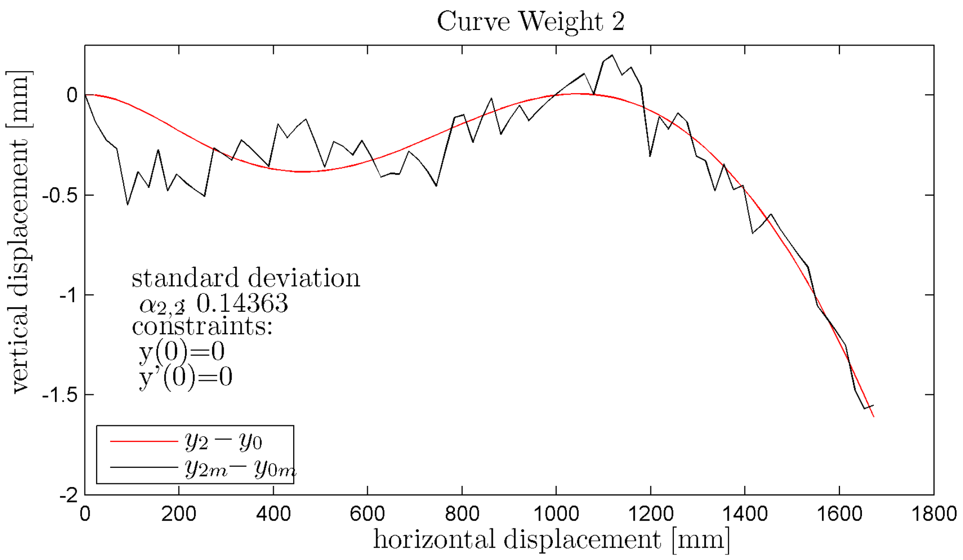


Figure 4.26: In black the data of the subtraction is given. In red the polynomial computed using both previously fitted polynomials is given. The standard deviation of the error of the fitted polynomial is $0.144\,[mm]$. The range of this curve is only $1.5\,[mm]$.

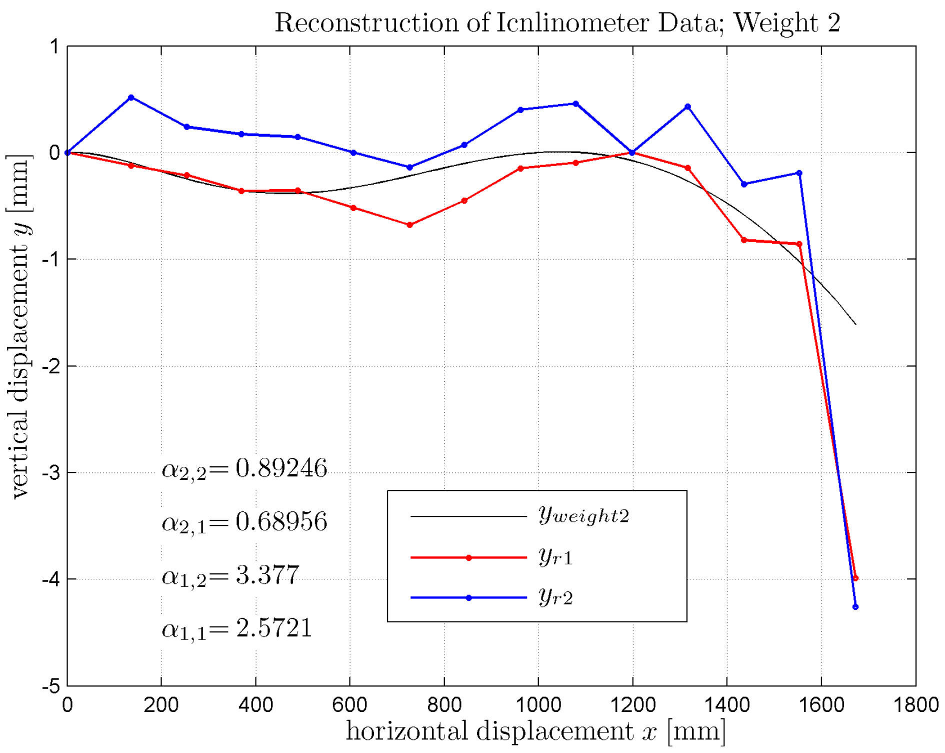


Figure 4.27: In black the curve obtained by the LSM for the current deflection is given. Reconstruction 1 and 2 are given in red and blue, respectively. The constraints placed on this curve are $\boldsymbol{y}(0) = 0\,[mm]$ and $\boldsymbol{y}(1198) = 0\,[mm]$. These two constraints are fulfilled by both reconstruction methods. The constraint of $\dot{\boldsymbol{y}}(0) = 0\,[mm]$ was not used of reconstruction because the distance between the first and second point is too great. If this constraint were placed on the first point at $x = 0\,[mm]$ the second point at which the inclination is measured would be forced to be zero as well, falsifying the curve computed. Great errors can be seen at the end of the curve. The reconstruction computed using Reconstruction Method 1 is very close to the curve's real deformation. A great deviation can be observed at $x = 726\,[mm]$ and at the end point. The second reconstruction method shows a shift in the vertical direction.

Experimental Interpretation

The errors observed in the measurements performed show a systematic error for the end points. Even though the inclination at $x = 0\,[mm]$ should be zero, the curve shows an inclination at exactly this point. The error at the end point is inferior. The reason for this error at the endpoints is the lacking number of inclinometers necessary to investigate the deformations of 2 metres. To avoid this error, one can

1. use more inclinometers for monitoring of the beam;
2. place the constraints on the interpolating function and not on the reconstructed points.

The second error, observed in most measurements is the shift in the vertical direction - positive or negative. This systematic shift probably is connected to the error at the end points and can be prevented following Proposals 1 and 2 above. Part of the errors observed are due to sufficient impreciseness from the inclinometers. The inclinometers tend to drift slightly while turned on.

Despite these errors encountered, the reconstructions of the curve are very close to the real deformation. Especially Reconstruction Method 1 delivered very good reconstructions for all measurements. Reconstruction Method 1 systematically reconstructs better than Reconstruction Method 2.

Chapter 5

Conclusion

This thesis has presented new approaches to the reconstruction of curves from over constrained gradients. This corresponds to a special case of an inverse boundary value problem. The methods reconstruct both, Inverse Boundary Value Problems and Inverse Initial Value Problems. Additionally, the reconstruction is a simple matrix multiplication, whereby the computation of the Matrix $\boldsymbol{M}$ can be done a priory which makes the presented methods suitable for real-time computations.

The proposed methods were verified using both Monte-Carlo simulations and with real inclinometer measurements on a deflected beam. Two independent reference measurement systems were implemented as a means of performing an objective verification. The experiments covered by this study showed that the implemented methods work under laboratory conditions and it's possible to detect the original curve's deflection using inclinometer values (or some other kind of gradient value) and the presented reconstruction operators.

The methods not only deliver a solution to the inverse problem but also deliver an estimate for the upper-bound on the uncertainty of the reconstruction. During the final testing of the system it was observed that placing the derivative constraints on the reconstruction points leads to an error at the beginning and ends of the reconstruction. Placing these constrains on the interpolating function would alleviate this problem.

There is further and future work necessary on optimal node placement i.e. sensor locations and interpolating constraints.

List of Figures

List of Tables

List of Source Codes

Bibliography

[1] J. Baik, T. Kriecherbauer, K.D.T.R. McLaughlin, and P.D. Miller. *Discrete Orthogonal Polynomials. (AM-164): Asymptotics and Applications (AM-164)*. Annals of Mathematics Studies. Princeton University Press, 2007.

[2] Paul Berberian. Wind turbine alignment faq. *Alignment Supplies, Inc.*, 2012.

[3] Michael R. Berthold, Christian Borgelt, Frank Hppner, and Frank Klawonn. *Guide to Intelligent Data Analysis: How to Intelligently Make Sense of Real Data*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[4] S.S. Bhavikatti. *Problems & Solutions In Engineering Mechanics*. New Age International (P) Limited, 2006.

[5] S. Brandt. *Data Analysis: Statistical and Computational Methods for Scientists and Engineers*. Springer, 1999.

[6] O. Burdet and L. Zanella. Automatic Monitoring of the Riddes Bridges using Electronic Inclinometers. In *IABMAS, First International Conference on Bridge Maintenance, Safety and Management*, 2002.

[7] A.S. Deif. *Advanced Matrix Theory for Scientists and Engineers, 2nd Edition*. Abacus books on mathematics. Abacus Press, 1991.

[8] H.W. Engl, M. Hanke, and A. Neubauer. *Regularization of Inverse Problems*. Mathematics and Its Applications Series. Kluwer, 2000.

[9] Walter Gautschi. *Orthogonal Polynomials, Computation and Approximation*. Oxford University Press, Oxford, 2004.

[10] Green GE and Mikkelsen PE. Measurement of ground movement with inclinometers. *unknown*, 1986.

[11] J. Golser. Fallbeispiel zur Bauwerksüberwachung mittels online Neigungssensoren. In *25. Cristian Veder Kolloquium*, 2010.

[12] Gene H. Golub and Charles F. Van Loan. *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences)(3rd Edition)*. The Johns Hopkins University Press, 3rd edition, 1996.

[13] J.P. Gram. Ueber die entwicklung realler funktionen in reihen mittelst der methode der kleinsten quadrate. *Journal fuer die reine und angewandte Mathematik*, page 150..157, 1883.

[14] Dr.mont Matthew Harker. Script to measurements of 3d objects, March 2011. Measurement Systems.

[15] Christian Hesse, Rainer Heer, Sebastian Horst, and Hans Neuner. A concept for monitoring wind energy turbines with geodetic techniques. *FIG Symposium*, 2006.

[16] H. Hinnen, Dr. M. Gassner, M. Jaray, and E. Mueller. *Compendium, The secrets of inclination measurement*. Wyler AG, Switzerland, 2012.

[17] Khalid Hosny. Exact Legendre moment computation for gray level images. *Pattern Recognition*, doi:10.1016/j.patcog.2007.04.014, 2007.

[18] Xingmin Hou, Xueshan Yang, and Qiao Huang. Using inclinometers to measure bridge deflection. *Journal of Bridge Engineering*, 10(5):564–569, 2005.

[19] ASTM International. Standard test method for monitoring ground movement using probe-type inclinometers. *ASTM International*, 2005.

[20] David A. Kopriva. *Implementing Spectral Methods for Partial Differential Equations: Algorithms for Scientists and Engineers*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[21] M. Lang. All pass filter design and applications. *In Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing, ICASSP*, 46:2505–2514, 1995.

[22] George Machan and Victoria Gene Bennett. Use of inclinometers for geotechnical instrumentation on transportation projects. *Transportation Research E-Circular*, E-C129, 2008.

[23] R. Mukundan. Some computational aspects of discrete orthogonal moments. *IEEE Transactions on Image Processing*, 13(8):1055–1059, 2004.

[24] R. Mukundan, S. Ong, and P. Lee. Image analysis by Tchebichef moments. *IEEE Transactions on Image Processing*, 10(9):1357–1363, 2001.

[25] Arnold Neumaier. Solving ill-conditioned and singular linear systems: A tutorial on regularization. *SIAM Review*, 40, 1998.

[26] Paul O'Leary. Script to digital image processing, October 2003.

[27] Paul O'Leary and Matthew Harker. An algebraic framework for discrete basis functions in computer vision. In *IEEE Indian Conference on Computer Vision, Graphics and Image Processing*, Bhubaneswar, Dec, 2008.

[28] Paul O'Leary and Matthew Harker. Discrete polynomial moments and savitzky-golay smoothing. *Waset Special Journal*, 72, 2010, pp. 439-443, 2010.

[29] Paul O'Leary and Matthew Harker. 'a framework for the evaluation of inclinometer data in the measurement of structures. *Procedings of the IEEE*, 61, no 5, pp 1237-1251, 2012.

[30] Paul O'Leary, Matthew Harker, and J. Golser. Direct discrete variational solutions to curve reconstructions from derivatives and their application to track subsidence measurements. *Instrumentation and Measurement Technology Conference (I2MTC), 2011 IEEE*, 2011.

[31] J. R. Parker. *Algorithms for Image Processing and Computer Vision.* John Wiley & Sons, Inc., 1st edition, 1996.

[32] S.M. Roberts and J.S. Shipman. *Two-point boundary value problems: shooting methods.* Modern analytic and computational methods in science and mathematics. American Elsevier Pub. Co., 1972.

[33] A. Savitzky and M. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical Chemistry*, 36 (8):1627–1639, 1964.

[34] Milan Sonka, Vaclav Hlavac, and Roger Boyle. *Image Processing, Analysis, and Machine Vision.* Thomson-Engineering, 2007.

[35] Joel Van Cranenbroeck. Continuous beam deflection monitoring using precise inclinometers. In *FIG Working Week 2007*, Hong Kong, SAR, 13..17 May, 2007.

[36] G.Y. Yang, H.Z. Shu, G.N. C. Han, and L.M. Luo. Efficient Legendre moment computation for grey level images. *Pattern Recognition*, 39:74–80, 2006.

[37] Pew-Thian Yap and Paramesran Raveendren. Image analysis by Krawtchouk moments. *IEEE Transactions on Image Processing*, 12(11):1367–1377, 2003.

[38] Pew-Thian Yap and Paramesran Raveendren. An efficient method for the computation of Legendre moments. *IEEE Transactions on Pattern Analysis and Maschine Intelligence*, 27(12):1996–2002, 2005.

[39] Hongquing Zhu, Huazhong Shu, Jun Liang, Limin Luo, and Jean-Louis Coatrieus. Image analysis by discrete orthogonal Racah moments. *Signal Processing*, 87:687–708, 2007.

[40] Hongquing Zhu, Huazhong Shu, Jian Zhou, Limin Luo, and Jean-Louis Coatrieus. Image analysis by discrete orthogonal Racah moments. *Pattern Recongition Letters*, doi:10.1016/j.patrec.2007.04.013, 2007.

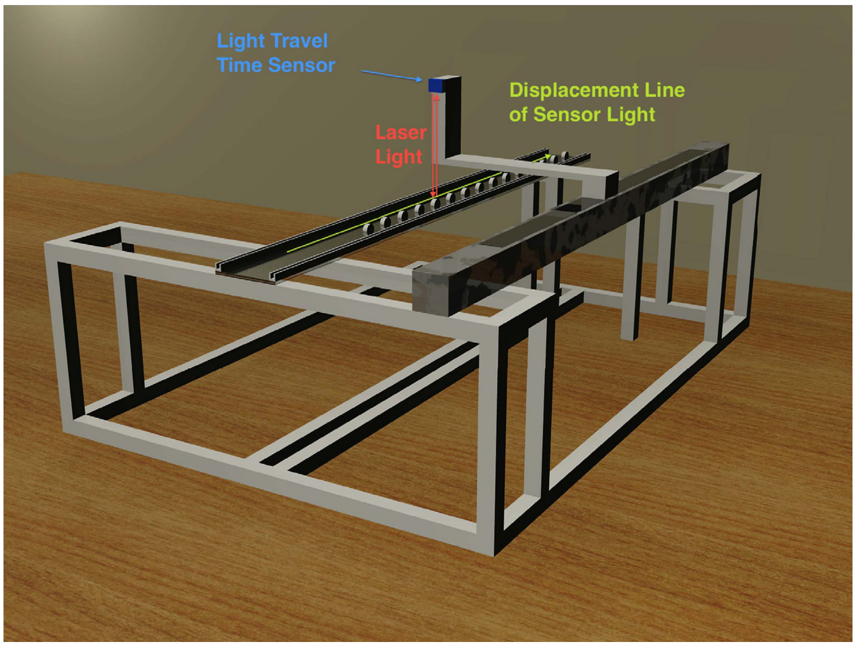


Figure 4.3: 3D Model of the assembly for the TFM set-up. The sensor is displayed as a blue box. It was necessary to mount the sensor at a certain distance from the object to be measured. The red line depicts the laser light emitted and observed by the sensor. During the measurement the sensor is displaced by the linear drive and the distance between the sensor and the metal sheet is measured at a constant rate. The green arrow shows the position and direction of the measurement points.

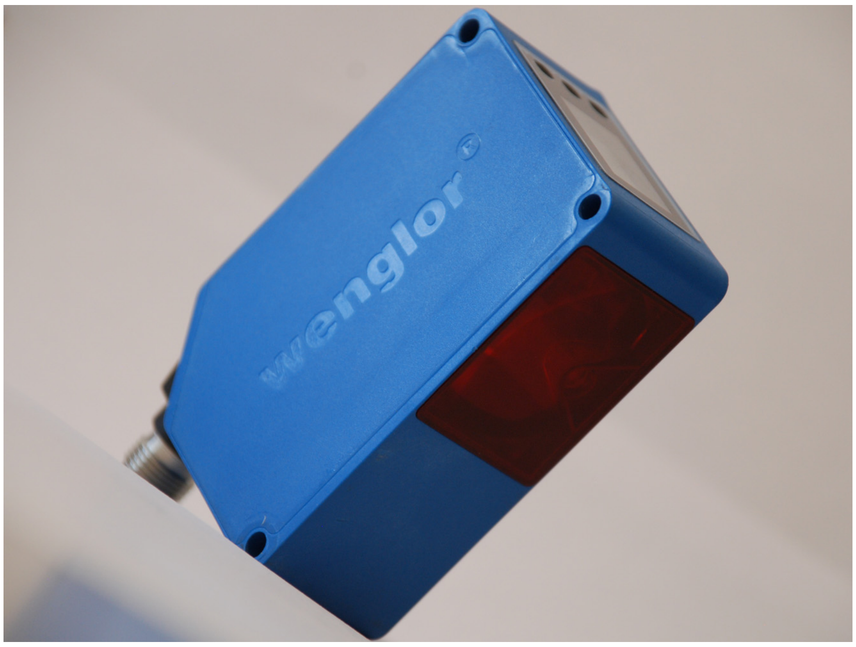


Figure 4.4: *Light Travel Time Sensor* X1TA100QXT3 from *Wenglor*.

4.3 Light Sectioning Method

4.3.1 Set-Up

The *Light Sectioning Method* (LSM) is an image processing method. The LSM can be adapted for a 2D or 3D application. It is a kind of structured lighting, which implies that a great deal of information is known about the light source as well as of the object [14]. In the current task the position of the laser line, the object to be measured as well as its position is known and can be adapted for application of the LSM, which allows us to use this method for measuring the deflection of the metal sheet. The light source is a laser (see Table 4.3.1 for detailed information of the laser) which projects a plane of light. This plane of light results in a line once projected onto the plane, i.e. the measurement object. The camera (see Table 4.3.1 for specific data) is set up in a manner that it can view the projected laser line on the object, see Figure 4.6 for the schematic alignment and Figure for the 3D model of the assembly for the LSM.

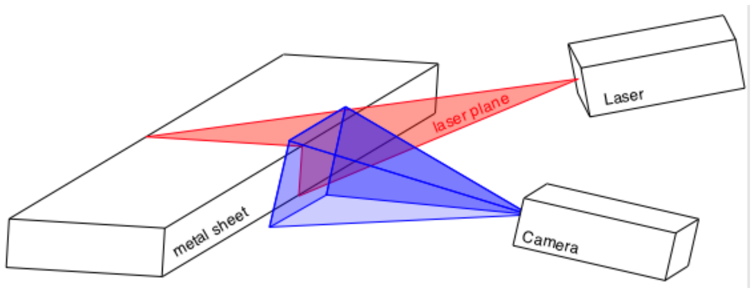


Figure 4.5: Schematic set-up of the LSM. Here the plane of light produced by the laser (red) is projected onto the thin side of the metal sheet; this results in a line once projected onto the metal sheet. The camera is mounted in a way that it can view this projected line perfectly. The camera's range of vision is displayed in blue.

Name	LASIRIS SNF Laser Class II
Weigth	65g
Diode power	1 to 200 mW
Wavelength	635 to 1550 nm
Intensity Distribution	Uniform (non-Gaussian) lengthwise
Input Voltage	4.8 - 6.5 V_{DC}

Table 4.2: Detailed information of laser LASIRIS SNF.

Geometrically the image observed by the camera is a perspective image. The camera is positioned so that the optical axis is normal to the surface being observed. For this reason the projective distortion can be neglected. To ensure a more precise image, an interference filter matched to the wavelength of the laser is used for the image acquisition. For each point at which an image is to be taken, the linear drive stops for a specific period of time for the camera to take a picture of the laser line. The idea behind taking the images along one can see the classified image - the ROI. The MATLAB implementation for this step is given in Source Code 4.1.

```
%Define maxima and minima in pixels
xmin = 700;
xmax = 760;
ymin = 210;
ymax = 450;
%
for i = 1 : noimages
    %define current image as a matrix on its own
    image_current = img{i};
    %crop current image and store into a new structure
    img_crop {i} = image_current(ymin:ymax,xmin:xmax);
end
```

Source Code 4.1: Code for the image cropping.

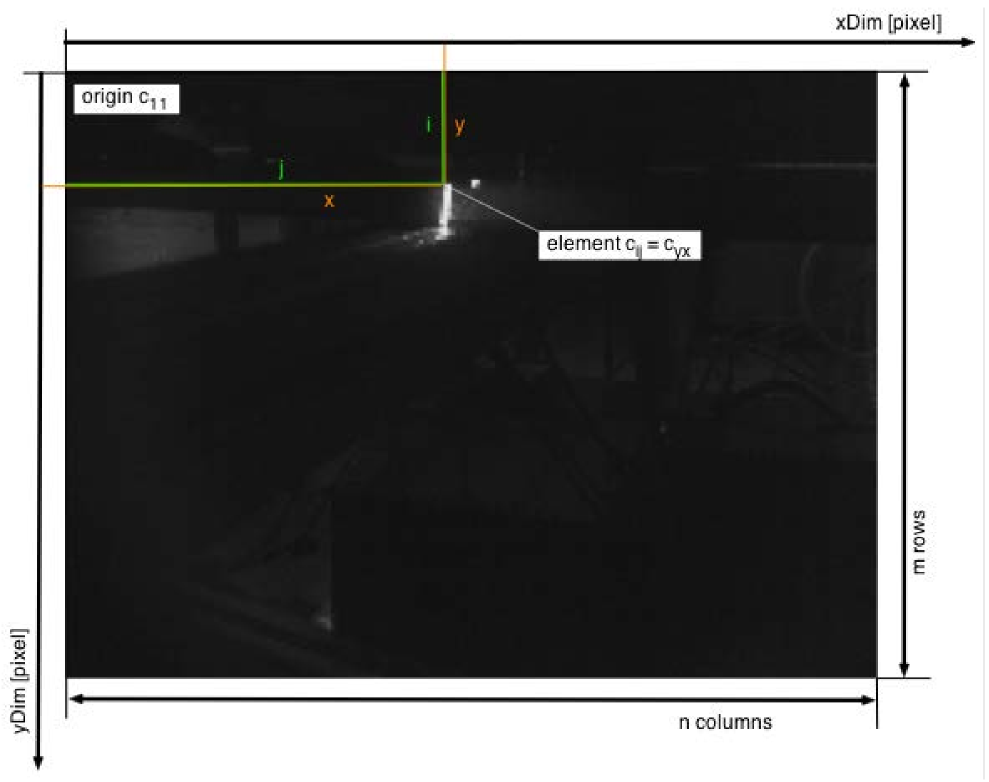


Figure 4.7: Each matrix has m rows, representing the y-dimension, and n columns, representing the x-dimension. The corrrepsonding indecis are i and j. Each element, i.e. each pixel, has a defined position at row i and column j as well as an 8 bit brightness ranging from zero to 255. The origin of the image, $c_{1,1}$ is in the upper left corner.