CHALMERS
UNIVERSITY OF TECHNOLOGY

Database for training predictive AI-based assessment algorithms in Structural Health Monitoring

Master"s thesis in the Master"s Programme
Structural Engineering and Building Technology

PÄR NÄSSLANDER

Department of Architecture and Civil Engineering
Division of Structural Engineering
Concrete Structures
CHALMERS UNIVERSITY OF TECHNOLOGY
Master"s Thesis
Gothenburg, Sweden 2021

MASTER”S THESIS

Predictive database for training AI algorithms in Structural Health Monitoring

Master”s Thesis in the Master”s Programme Structural Engineering and Building Technology

PÄR NÄSSLANDER

Department of Architecture and Civil Engineering
Division of Structural Engineering
Concrete Structures
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2021

Predictive database for training AI algorithms in Structural Health Monitoring

Master"s Thesis in the Master"s Programme
Structural Engineering and Building Technology

PÄR NÄSSLANDER

Examensarbete ACEX30-YY-NN
Institutionen för arkitektur och samhällsbyggnadsteknik
Chalmers tekniska högskola, 2021

Department of Architecture and Civil Engineering
Division of Structural Engineering
Concrete Structures
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

Predictive database for training AI algorithms in Structural Health Monitoring
Master"s Thesis in the Master"s Programme
Structural Engineering and Building Technology
PÄR NÄSSLANDER

Department of Architecture and Civil Engineering
Division of Structural Engineering
Concrete Structures
Chalmers University of Technology

ABSTRACT

Structural health monitoring (SHM) is a useful technique for ensuring integrity and safety of a structure. Although it is still under development the concept of SHM is to detect possible errors or weaknesses in an early stage and address suitable maintenance or strengthening options. According to Berrocal et al. (2021) a new method in the field of SHM has been established recently by measuring the strain using distributed optical fiber sensors (DOFS). Furthermore, implementing DOFS in reinforced concrete proves to be an accurate and reliable tool for measuring crack widths and deflection.

A promising possibility with DOFS is to make automated assessments and predictions of various structural members by combining DOFS with artificial intelligence (AI). The learning capability of AI would be based on a training database comprising the strain data from DOFS and the crack information. Apparently, the economic and environmental cost to produce many different scenarios in a laboratory would be very high. One alternative is to simulate the various scenarios in finite element analysis (FEA) models obtaining artificial strain profiles together with the crack locations. However, this method requires good compatibility between the strain from FE models and DOFS which is not the case. Therefore, postprocessing the artificial strains can hopefully make the two become closely related to each other. If this becomes reality, a future AI training database can be built by collecting strains from FE models with different geometries, material properties, loading conditions and support setups.

This thesis aims to examine the problem stated above, i.e., if strains from FE models can, trough manipulation, become comparable to the strains in DOFS. Attempting to answer this question the work was divided in two parts, the calibration procedure, and the strain postprocessing. In the first part, the ambition was to create a digital copy with similar structural behavior as three beams from an experiment made earlier. As a result, a major calibration process took place comparing different random field methods using the JCSS Probabilistic Model Code in concrete. In addition, more detailed comparisons were made between different bond slip characteristics. In the second part, the processing methods were implemented with the intention of synchronising the artificial strains from FE model with the strains from DOFS.

Although the FE model had success of demonstrating identical behavior as the tested beams it was hard reproduce the shape from the DOFS. The strains from the DOFS illustrate a rather detailed strain profile, but without the capability of reflecting all the cracks. Under these circumstances it is difficult to reach accordance between the two type of strains. The results indicate that a thinner type of DOFS would perhaps give more satisfying results. This is however left for future research.

Key words: Structural health monitoring, distributed optical fiber sensors, finite element analysis, artificial intelligence

Contents

Preface

The work of the thesis took place between January and September 2021 and is a part of an ongoing research project concerning Structural Health Monitoring. Because of current restrictions most of the work was done at home although some experimental work was carried out at the Department of Structural Engineering, Concrete Structures at Chalmers University of Technology.

The thesis was made by Pär Násslander with Ignasi Fernandez as supervisor and examinator. I especially want to thank Ignasi for showing great patience and assisting me with valuable information and useful feedback. I also want to address my gratitude to Carlos G. Berrocal for his support during the laboratory work.

Göteborg March 2021

Pär Nässlander

1 Introduction

1.1 Background

Structural Health Monitoring (SHM) is a promising technique for observing the structural performance and assessing the undergoing process of aging and deterioration in reinforced concrete. Monitoring the structural behavior in reinforced concrete can detect errors or weaknesses in and early stage. This allows to address suitable maintenance or strengthening options before possible damage take place. Consequently, it can change today's method of making inspections between certain time intervals, to a method using more directed actions on specific crucial areas recognized by the SHM.

According to Berrocal et al. (2021) a SHM system based on optical fiber sensors has played an important role during the last decades. Furthermore, a new method of damage detection in reinforced concrete has been established not long ago, using distributed optical fiber sensors (DOFS). The DOFS shows high potential due to their lightweight, small size and high resistance to the environment, but also due to the new implementation of unprecedented spatial resolution (Berrocal et al. 2021).

Performing assessment and predictions of constantly incoming data from DOFS in various members of a structure is obviously demanding. One future opportunity to tackle this problem is by implementing artificial intelligence (AI). Founded on a strain/crack database from numerous loads and supports setups, geometries and material characteristics, the AI algorithm would be able to determine crack locations and crack widths by reading the strains from DOFS.

The learning capability of AI is depended on the training database containing strain data from DOFS together with crack details from each specific case. Apparently, the economic and environmental cost to produce the different scenarios in a laboratory would be very high. One possibility is to simulate the various scenarios in FE models obtaining artificial strain data and current crack information. This alternative can provide a huge amount of training data in a relatively short time. However, because of several reasons, the strains in DOFS and FE models are different and needs to be considered. Nonetheless, if the creation of a trustworthy training database is possible, predictive assessment of concrete structures driven by AI algorithms can become the new reality.

1.2 Aim and objectives

The aim of the thesis is to evaluate the possibility of creating a reliable training database established on the results from FE models. More precisely, examine and validate if the strain in the reinforcement from a FE model can become comparable to the strains in the DOFS.

Throughout this work, a close link between experimental test results and results from FEA were constructed. The thesis was categorized into the following specific objectives:

- Collecting data from earlier experiment through digital image correlation (DIC) and DOFS.
- Creating a well calibrated FE model based on the gathered data from DIC.

- Utilizing the calibrated FE model to extract strain data from the reinforcement.
- Postprocessing the artificial strain data with the purpose of imitating the strain profiles from the DOFS.
- Investigating the compatibility between the processed strain profiles and the strain profiles from the DOFS.

1.3 Limitations

All the beams were tested under service condition only. The parameters used in the FE model during the calibration process were limited to the JCSS Probabilistic Model Code in concrete and the bond slip properties in the reinforcement.

1.4 Methodology

Results from DIC and DOFS were obtained from an experiment made earlier. These test results were analyzed and organized.

A literature study was done about the theory of FE modelling in DIANA FEA. Parallel to the literature study a model was constructed in DIANA FEA based on the same material properties, geometry, loading and support setup as in the experiment. The ambition was to create a digital copy with similar structural behavior as the beams from the experiment. Because of that, a major calibration process took place comparing different random field methods in concrete using JCSS Probabilistic Model Code. Also, comparisons were made between different bond slip characteristics.

After obtaining a well calibrated FE model, processing methods were implemented with the intention of matching the artificial strains from FE model with the strains from DOFS. Finally, the compatibility between the two strain types was analyzed and evaluated.

The methodology consists of two main targets:

- Calibration procedure of FE model with the purpose of replicating the structural behavior in the beams from the experiment.
- Imitating the shape of the strain profiles from DOFS by postprocessing strain data from the calibrated FE model.

2 A brief overview on SHM

2.1 Structural health monitoring

Throughout history people have performed SHM, e.g., visual and audial inspection on objects in order to evaluate its condition. Today, almost all major industries use some kind of SHM for optimization purposes. Although the discipline of structural engineering has a reputation of being less prone to implement new innovations, much development has been made on the subject during the last two decades, partly due to major research in optical fiber sensing technology. Various countries around the world have applied optical fiber sensors (OFS) in civil engineering structures for detecting and measuring physical quantities such as strain, crack and displacement. Already today, OFS plays an important role in health monitoring of dams, bridges and other important civil structures.

More recently, attention has also been drawn to artificial intelligence (AI) owing to the ability to learn highly abstract features from raw data to fulfill recognition, classification and prediction tasks. These characteristics can become an essential tool in SHM providing alerts and predictions of the structural integrity by reading the stream of data coming from DOFS.

2.1.1 Optical fiber sensors

In comparison to other existing assessment methods as vision, vibration/impact, conductivity or radar testing, OFS are able to evaluate the behavior of structures more deeply and uninterrupted. Moreover, they possess advantages as: high precision, small size, low weight, corrosion resistance and anti-electromagnetic interference. It enables OFS to be placed into very tight areas of structural elements, endure chemically aggressive environments, survive lightning strikes, and in addition, form sensor chains using a single fiber.

Due to the progressing development, the variation of OFS is wide and are used in many fields and applications. A standard optic fiber sensing system is mainly composed of light source, receiver, optical fiber, modulator element, detector, and signal processing unit (Han & Su, 2014). Light from a light source is transmitted through an optical fiber to modulation area where light interacts with external factors and change its optical property. Next, light is transmitted back to detector and analytical unit for processing of physical quantity based on changes in light wave parameters, see Figure 2.1.

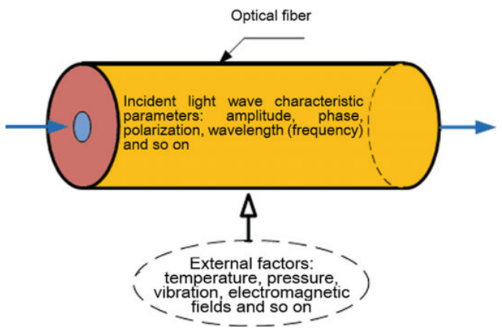

Optical fiber
Incident light wave characteristic parameters: amplitude, phase, polarization, wavelength (frequency) and so on
External factors: temperature, pressure, vibration, electromagnetic fields and so on

Figure 2.1 Basic working principle of optical fiber sensor (Du et al., 2020).

OFS are composed of fiber core and cladding as shown in Figure 2.2. The cladding can be covered with an external coating to provide protection against surrounding environment. Fiber cores are usually made from silica glass or polymer material and can work as both sensing unit and signal transmission. As illustrated in **Error! Reference source not found.**, the cladding seals the optical wave by reflection at the interface between core and cladding.

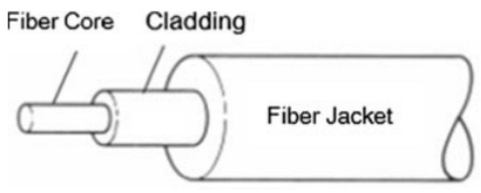


Figure 2.2 Optic fiber sensor (Du et al., 2020).

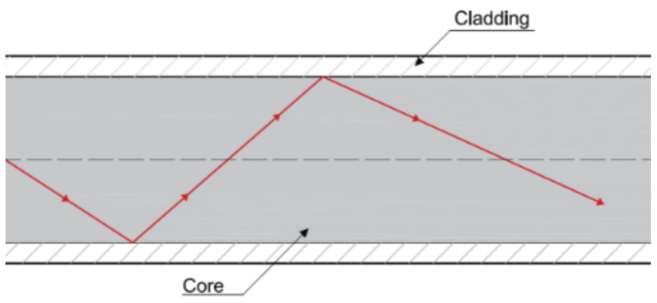


Figure 2.3 Light transmission and reflection in an optical fiber (Barrias et al., 2016).

DOFS can be categorized into different classification standards depending on which property to be considered. A common classification is illustrated in Figure 2.4 and consist of three groups: interferometric sensors, grating-based sensors and distributed sensors (Barrias et al., 2016). The first two categories have been widely researched and used in civil engineering monitoring applications and in contrast to DOFS they both have limitations regarding number of measuring points along the optical fiber. DOFS work as one entire sensing and transmission unit where the sensing points are distributed continuously along the optical fiber. It enables to capture phenomena that are distributed continuously in 3D space, e.g., temperature- or stress field. The recent advancements in DOFS have expanded its sensing range with a spatial resolution from 1 mm up to hundreds of kilometers (Barrias, Casas, & Villalba, 2016).

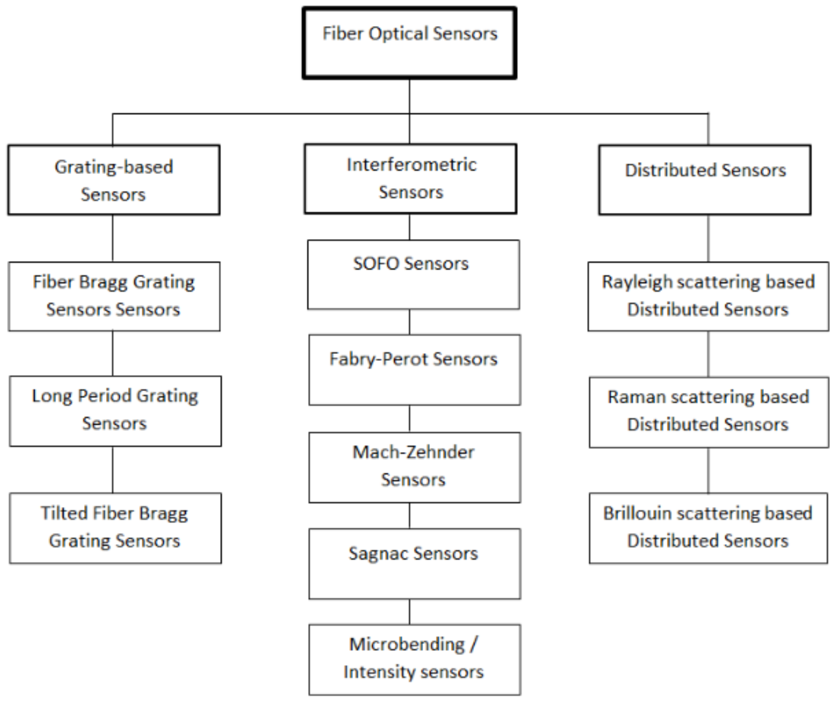

Fiber Optical Sensors
Grating-based Sensors
Interferometric Sensors
Distributed Sensors
Fiber Bragg Grating Sensors Sensors
Long Period Grating Sensors
Tilted Fiber Bragg Grating Sensors
SOFO Sensors
Fabry-Perot Sensors
Mach-Zehnder Sensors
Sagnac Sensors
Microbending / Intensity sensors
Rayleigh scattering based Distributed Sensors
Raman scattering based Distributed Sensors
Brillouin scattering based Distributed Sensors

Figure 2.4 Common classification system of optical fiber sensors (Barrias et al., 2016)

The working principle of DOFS is based on reflection and interference of light. When changes in strain or temperature in a material is transferred to DOFS the scattered signal within the fiber is modulated by the physical parameters. The variation of the modulated signal is then quantified through light backscattering (Du, Sun, Li, & Zhang, 2020). There are three different light backscattering processes: Raman, Brillouin and Rayleigh scattering.

According to Barrias et al. (2016) Brillouin scattering DOFS are currently the most studied and applied measuring tool for SHM in civil structures. Their measurement range is very suitable for large structures as tunnels, dams, bridges, pipelines etc. Raman scattering is mainly sensitive to temperature and is therefore not well suited for structures (Berrocal et al., 2021). Instead it has been used in other areas, e.g., detection of water leakage in dikes. For high spatial resolution with millimeter scale Rayleigh scattering shows high potential. Even though a limiting sensing range of about 70 m the Rayleigh scattering outperform Brillouin or Raman system when it comes to spacial resolution (Barrias, Casas, & Villalba, 2016). Moreover, several recent studies have applied Rayleigh-based DOFS to measure strain, crack widths and vertical deflecion with good results.

2.1.2 Artificial intelligence

Although AI is not a in the scope of the thesis, AI together with DOFS will most probably play an important role in future SHM. For that reason, brief information about AI and deep learning (DL) is presented along with current research and applications.

AI is a field of computer science aiming machines to demonstrate similar intelligence in solving problems as human beings. Machine learning (ML), which is a subcategory of AI, see Figure 2.5, has the ability to iteratively learn to tackle problems from experience without being explicitly programmed to do so. ML enables computers to learn hidden patterns in collected data and performs well for solving regression and classification.

Handling and using big streams of raw data in real-time is complex and has occupied scientists for several years. In the arena of SHM, the challenge is the constant monitoring of structural systems to detect, localize and assess irregularities. DL, a subbranch of ML is one of the most promising tools for managing such tasks. It has recently dominated the research field and it appears to be the most capable method for autonomously processing and analyzing large amount of information (Jin, Ye, & Yun, 2019). Furthermore, DL can learn abstract features among collected data to perform classification tasks or future predictions, e.g., image and audio recognition, traffic prediction, self-driving cars, spam email detection etc. As a result, DL-based methods have becoming to play an important role in the field of SHM.

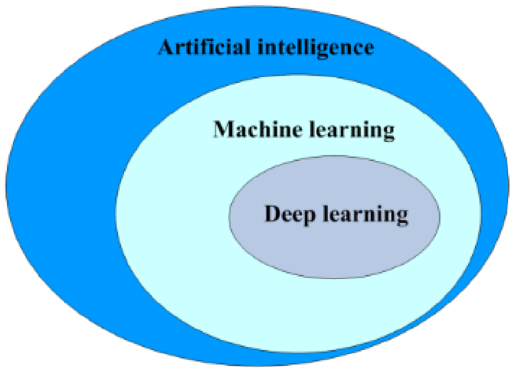


Figure 2.5 Categorization of AI (Jin et al., 2019).

Several studies have focused on vision-based or vibration-based monitoring together with DL. According to Catbas & Dong (2021), DL methods based on observations from computer vision is an interesting technique because of the long-distance and low-cost application with minimum intrusion to the daily operation of the structure. Vibration-based detection utilizes the vibration response of the monitored structure in combination with DL to identify and validate structural damage. Abdeljaber et al. (2020) claims that the implementation of DL algorithms into vibration-based methods have resulted in better performance and accuracy in SHM.

The advancing sensing technology together with increasing computing power, fast developing communication technologies, and progressing DL algorithms makes the interaction between DL methods and DOFS very interesting. The usage of DOFS together with DL, a study made by Ansari et al. (2020), proved that a DL method is capable of distinguishing microcracks from the environmental noise when measuring the strain distribution in DOFS mounted on a steel beam. Another research in the same field from Karypidis (2019) constructed a DL method which successfully quantified the damage in reinforced concrete beams by reading the strain distribution in DOFS. In this case the damage assessment was performed by introducing different strain thresholds along the sensors. The DL method could then classify and predict the amount of damage in a beam by reading the strains continuously.

Although research has been made on DL-based SHM the fully potential is most likely not revealed, especially the interaction between DOFS and DL algorithms. A future scenario would enable DL to automatically receive, process and analyze the constant flow of input data from DOFS located in various members in a structure, and providing notifications of ongoing abnormalities or structural damage. As a primary step in this future scenario, this work aims to investigate the possibility of creating an artificial strain-database which possess similar strain properties as from DOFS. This potential database can in future research serve as a training tool for DL systems to localize and measure cracks and making reliable assessments and predictions.

3 Finite element modelling

This chapter is aimed to describe the finite element modelling and the material property of concrete implementing the JCSS Probabilistic Model Code.

A finite element model is a digital copy of a physical structure with a number of assumptions, generalizations, and idealizations. First, simplifications and assumptions are made from the physical structure. Then, discretizing the digital model is done into a finite element model applying proper material models, boundary conditions, loading conditions, etc. The analysis method of reinforced concrete (RC) involves therefore many choices, e.g., the type of finite elements, choice of material models, structural solution methods etc.

The material models for concrete are complex due to the existence of non-linear stress-strain relations, multi-axial stress conditions, anisotropic stiffness, progressive cracking, and time dependant behaviour as creep and shrinkage. RC becomes even more complicated with the action of reinforcement and the stress transfer between the concrete and reinforcement, i.e., the bonding.

For practical reasons, a finite element model considers concrete as a homogeneous material neglecting the small effects of aggregates, pores, or cement particles. Also, it considers concrete as an isotropic material until cracking occurs. In contrast to steel, the failure modes of concrete are different whether the stress is mainly compressive or mainly tensile. The different material models used for concrete are based on various theories describing the different phases of the material behaviour. First, linear-elastic response is preferable during the initial phase. Later, when the concrete cracks or becomes close to compression failure, non-linear methods are decisive for how well the analysis reflect the response of the real structure. Commonly, non-linear solution methods are performed by plasticity models or crack models based on fracture mechanics (Plos, 2000). Both use incremental iterative methods to solve the non-linear finite element equations.

3.1 Compressive behaviour of concrete

Concrete failure is caused by the growth of microcracks in both tension and compression. Unlike in tension, microcracks that forms under compression are well distributed until failure and can be seen as a homogeneous and isotropic material (Plos, 2000). This enables a constitutive model based on continuum mechanics to reflect the stress and strain relationship. Both elastic and plastic behaviour is normally applied and various material models can be chosen based on the different types of stress-strain diagrams, e.g., the constant compression curve as seen in Figure 3.1, the linear curve or the Thorenfeldt curve.

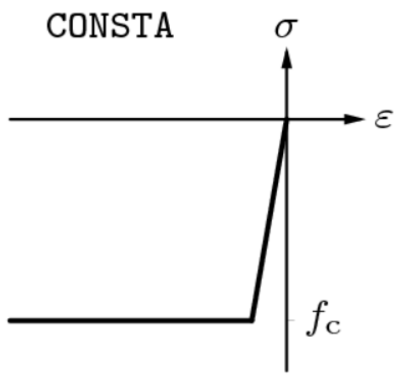


Figure 3.1 Constant compression function used in DIANA FEA (DIANA FEA BV, 2017).

3.2 Tensile behaviour of concrete

The microcracks in tension localizes into narrow zones where they eventually transform into real cracks. Discontinuities like cracks makes it problematic to use continuum mechanics to describe the cracking behaviour. Instead, fracture mechanics is often used when formulating constitutive relations in terms of stress and crack opening (Plos, 2000).

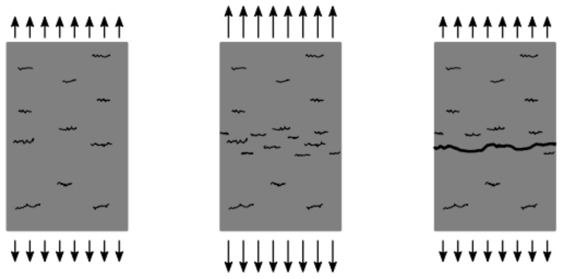


Figure 3.2 Uniaxial tensile crack formation until failure.

Most concrete FE models are using the smeared crack approach since real crack predictions for RC are more advanced (van den Bos & Garofano, 2016). With this approach cracked regions can take place over several integration points and multiple elements. This means that the material does not have any imperfections and the result is presented over a cracked region where crack localisation on element level cannot be found.

Instead of cracks smeared out over a specific area the stochastic models are more complex and implement variational material properties in a more realistic behaviour. With random fields, the material strength parameters are randomly distributed over the integration points. This means a clear difference in crack predictions compared to the smeared crack approach. According to van den Bos & Garofano (2016) random fields show a much better prediction of localising cracks compared to smeared crack approach.

The tensile behaviour of concrete is typically modelled using one of the two following methods: the total strain-based rotating crack model or the fixed crack model. The fixed crack model assumes that the axes of cracks remain fixed once crack axes are defined, whereas in rotating crack model the direction of cracks rotates continuously depending on the changes in the axes of principle strains when cracks are initiated.

With the fixed crack model stress-locking phenomena is present since stresses rotate after crack formation which can result in an overestimation of the failure load (Rijkswaterstaat, 2016).

The material model in tension includes softening functions which is based on the fracture energy. Common functions are, e.g., linear softening curve illustrated in Figure 3.3, exponential softening curve, and Hordijk nonlinear softening curve.

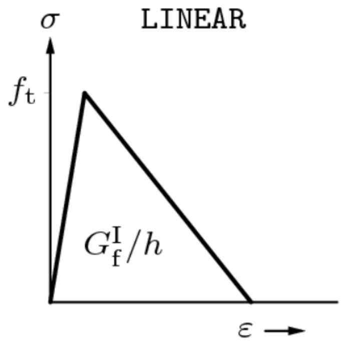


Figure 3.3 Multi-linear uniaxial stress-strain diagram in tension with parameters: tensile strength (f_t), fracture energy (G_f) and length (h) (DIANA FEA BV, 2017).

3.3 Behaviour of reinforcement steel

The material model for reinforcing steel is more simple using a linear elastic model until reaching yield stress, and thereafter, using a plastic material model. In RC modelling the bond behaviour is simplified and described by a bond-slip model which relates bond stresses to the slip of reinforcement.

3.4 JCSS Probabilistic Model Code

The Joint Committee on Structural Safety (JCSS) has developed a model code for probabilistic design, called JCSS Probabilistic Model Code. The software DIANA FEA has realized this model code for concrete, making it possible to apply varying strength by generating a random field. More precisely, the strength parameters are set the same as for a homogeneous material but with additional parameters defining the random field method. The random field inputs influence the random distribution of the shifting strength based on material correlation (van den Bos & Garofano, 2016).

Figures 3.4 and 3.5 demonstrate the difference in tensile strength implementing the JCSS Probabilistic Model Code and a conventional material model code, e.g. the Eurocode. In Figure 3.5 the tensile strength is constant and is defined by the strength parameter. Contrary, the tensile strength in Figure 3.4 is defined by a compressive strength parameter (basic compressive strength), but in addition, a specific random field model is chosen. This model normally localizes one larger crack in a member caused by the imperfection of the material whereas a standard method, e.g. the smeared crack approach, distributes the cracks over the entire body. Because of this feature, it is proven that the JCSS Probabilistic Model Code in DIANA predict and localize cracks more accurate in comparison to the constant smeared crack approach (van der Have, 2015).

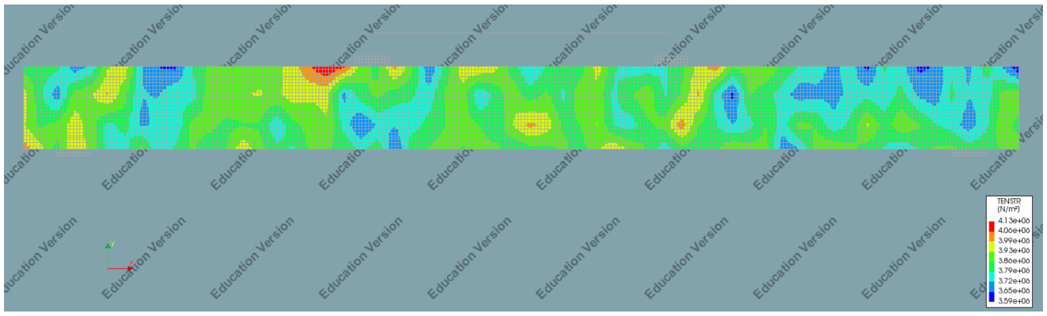


Figure 3.4 *The varying tensile strength in concrete beam simulated in DIANA using a random field with a correlation length of 0.05 m.*

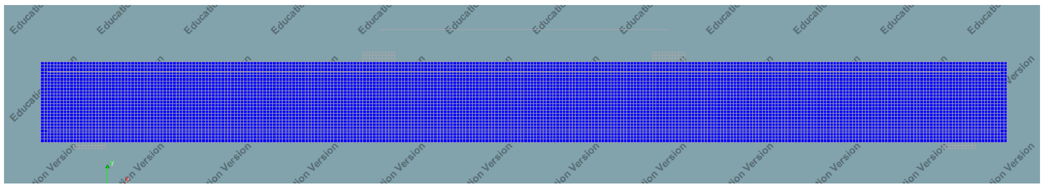


Figure 3.5 *The homogenous tensile strength in concrete beam using a conventional material model code.*

4 SHM experiment

This chapter aims to present general details about the SHM experiment which was published in the article: "Assessment and visualization of performance indicators of reinforced concrete beams by distributed optical fiber sensing", written by Berrocal et al. (2021). The experiment took place at Chalmers University and is a part of a project called: "Sensor-driven Cloud-based Strategies for Infrastructure Management". Moreover, the experiment was focused on assessment and visualization of reinforced concrete beams integrated with DOFS. In addition to DOFS, the beams were monitored by DIC to evaluate the compatibility between the fiber signals and the visual behavior of the beam. Information from the experiment regarding geometry, reinforcement/DOFS layout, load setup and material properties was taken from the article whereas the experimental results, i.e., the output data from DOFS and DIC, was exclusively obtained from one of the authors, Ignasi Fernandez. The experiment involves three identical reinforced concrete beams with the same geometry, material, and load/support setup.

4.1 Geometry, layout and loading condition

Berrocal et al. (2021) used a beam length of 3 m with a rectangular cross section of 200 mm x 250 mm, see Figure 4.1. Additionally, the beams were mounted with two upper and three lower longitudinal reinforcement bars with a diameter of 10 mm and 16 mm, respectively. The longitudinal bars were enclosed by 8 mm vertical stirrups between supports and point loads preventing shear cracks to take place.

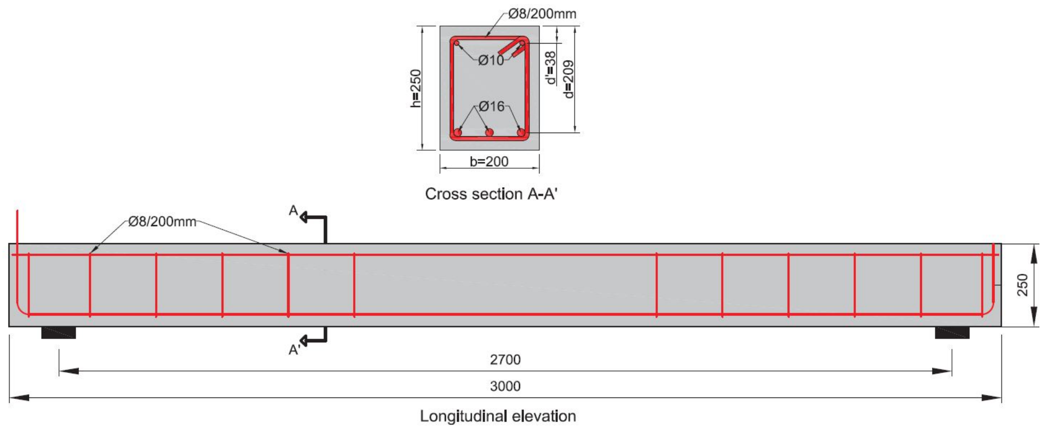


Figure 4.1 Geometry and distribution of reinforcement (Berrocal et al., 2021).

According to Berrocal et al. (2021) the beams were subjected to a four-point bending with two supports located 150 mm from each side of beam, and two point-loads located at 900 mm from each support, see Figure 4.2. This loading setup created a constant moment span between the point loads, and as a result, all crack observations were made in this section. Furthermore, displacement-controlled loading was applied with a displacement rate of 0.5 mm/min. Also, two loading cycles of 60 kN each were performed in the experiment but only the first cycle was considered in this report.

4.2 DOFS

The installation configuration of DOFS is displayed in Figure 4.2 & 4.3. A single robust DOFS was mounted with electric tape between the supports in five different positions along the beam. Two above the outer longitudinal tensile rebars, one under the longitudinal compressive rebar on the upper right side, one at mid-height bridging the stirrups, and one at the bottom resting on the formwork. The researchers did also mount a thin DOFS (bar2) for the purpose of comparing the strain sensitivity between thin and robust DOFS. This is however not covered in this thesis.

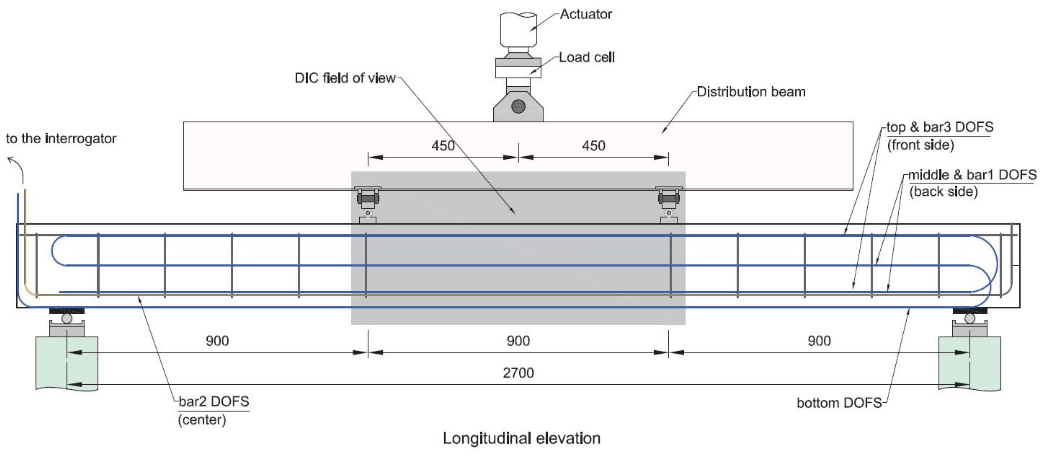
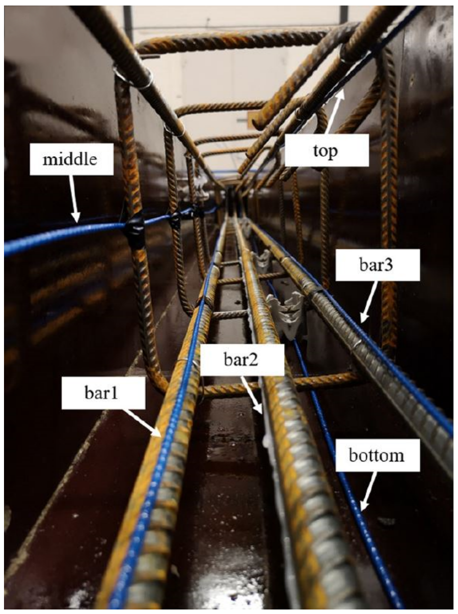


Figure 4.2 Loading setup, reinforcement layout and sensor deployment (Berrocal et al., 2021).



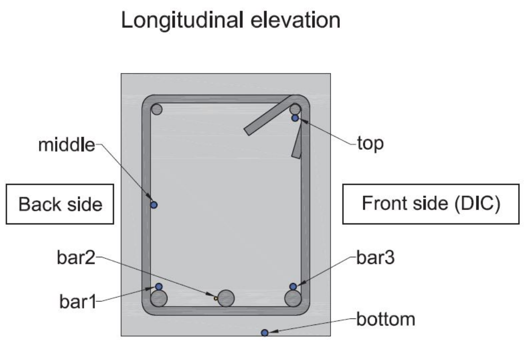


Figure 4.3 Reinforcement- and DOFS distribution layout (Berrocal et al., 2021).

The five different sensing locations in the beams provided good monitoring capability. By applying the sensors on different heights, it could confirm the conventional beam theory where the maximum strain was measured in the lower DOFS and decreases proportionally with the height until it becomes negative for the

DOFS located in the compressive zone, see Figure 4.4. Moreover, Figure 4.4 illustrates different strain variations along the span where every peak indicates the crack position. The figure also demonstrates the crack propagation on different heights of the beam during increasing load levels.

Furthermore, different height-locations of the DOFS made it possible for the researchers to calculate the deflection based on the Euler-Bernoulli beam theory by measuring the different strain variations.

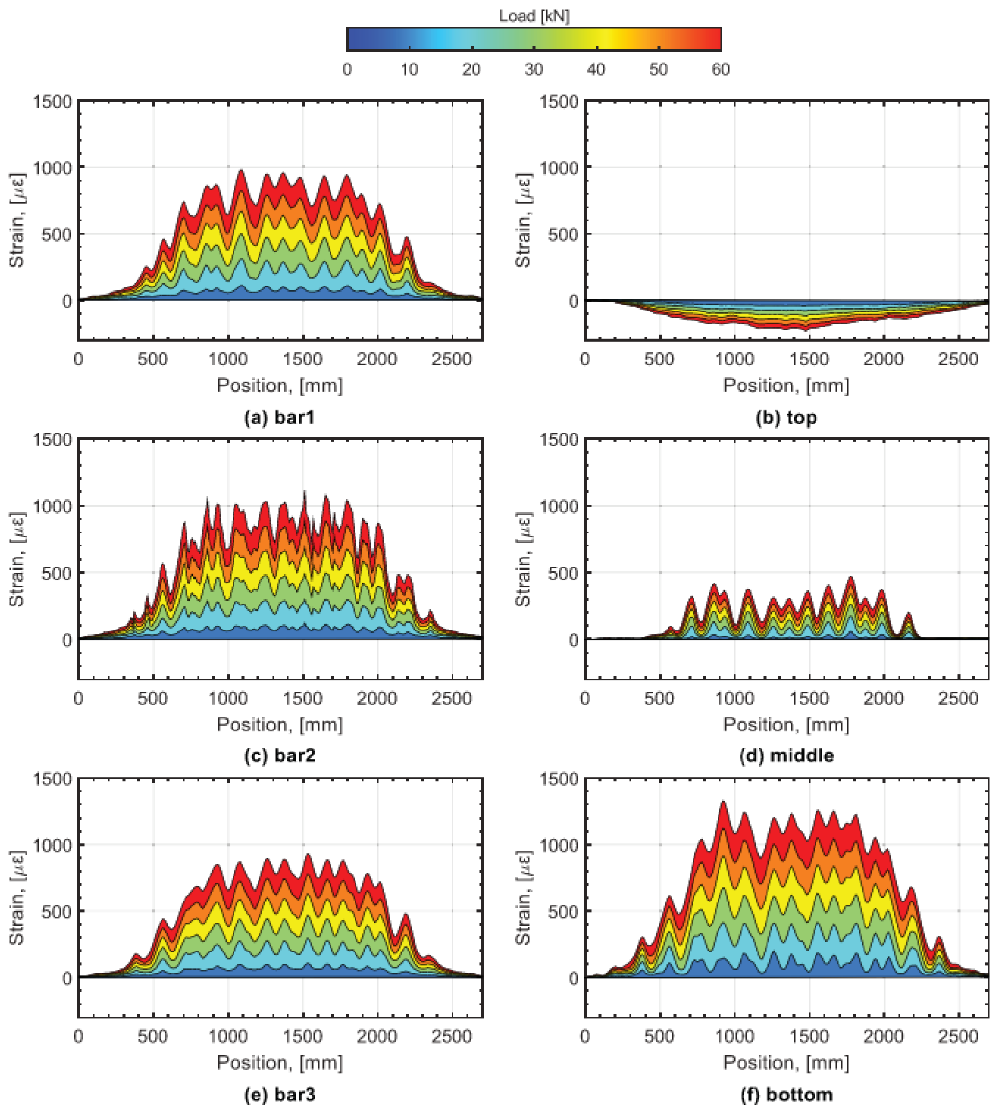


Figure 4.4 *Strain profiles for increasing load levels from the different DOFS in one of the beams (Barrias, Casas, & Villalba, 2016).*

Having DOFS mounted on each side of the beam could confirm that the cracks did not always propagate perpendicular to the main axis of the beam since the measurements on the two sides indicated different crack positions.

In the experiment, DIC was monitoring only one side of the beams and was aimed to measure the deformation and surface strains between the two loading points. These measurements were then used by the researchers to verify and evaluate the DOFS regarding crack location, crack width and beam deflection.

Although there were five robust sensor deployments in the experiment, only sensor "bar3" on the front side was of interest in this thesis since it was the only sensor that was situated on a reinforcement bar and on the same side as the DIC. Having data from both DIC and DOFS made it possible to first create an FE model reflecting the cracks and deflection from the DIC. Then, processing the strain profile in the reinforcement bar from FEM-design with the purpose of reflecting the strain profile from the DOFS.

Figure 4.5 demonstrate the strain profiles from the experiment for the DOFS "bar3" together with DIC. The figure illustrates that the strain profiles from DOFS can identify cracks, i.e., the distinct strain peaks from the DIC. However, when a secondary crack grows close to an existing one, it can lead to a convoluted strain peak which prevents the distinction of two cracks. According to Berrocal et al. (2021), this can be avoided by considering the strain history using a lower load level.

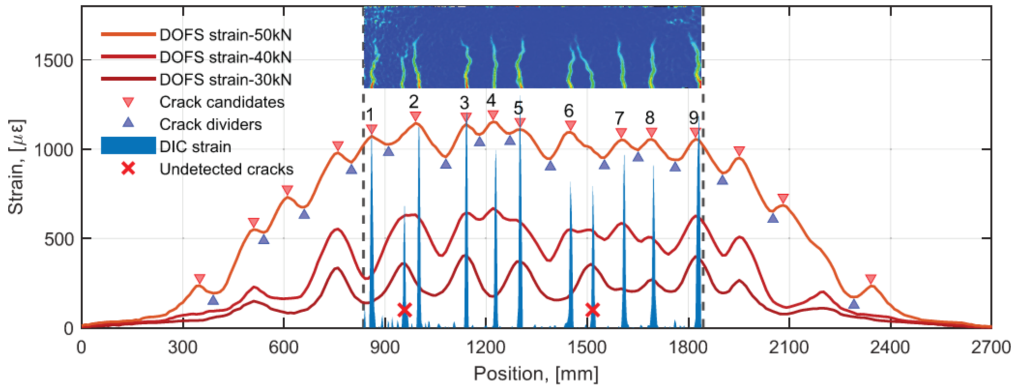


Figure 4.5 Localization of cracks through DOFS and verified with strain profile captured from DIC. (Barrias, Casas, & Villalba, 2016).

4.3 Material properties

One of the authors of the article Ignasi Fernandez, provided material characteristics of the concrete beams which are summarized in Table 4.1. The concrete was tested using three different samples with a diameter and height of 100 mm and 200 mm respectively.

According to Berrocal et al. (2021) the reinforcement used in the experiment was normal ductility carbon-steel with a nominal yield strength of 500 MPa and a Young's modulus of 200 GPa. Furthermore, the type of DOFS used were BRUsens V9 from Solifos featuring an inner steel tube and an external cladding. This type of shape is categorized as "robust DOFS".

Table 4.1 Concrete properties.

Beam	f_{cm} [MPa]	E_{cm} [GPa]	G_f $[N/m]$
1	79.5	32.7	124.2
2	62.5	36.1	149.2
3	72.4	31.0	129.5
Mean	71.5	33.3	134
Standard dev.	8.5	2.6	10.7

5 Model Calibration and Data Processing

The beginning of this chapter is to inform the reader about the FE model and the calibration process, i.e., constructing a FE model with similar structural behavior as the three beams from the experiment. The ending part of this chapter focus on describing the strain postprocessing methods and suggest ideas on how to evaluate similarities/differences between the processed artificial strains from FE model and the strains from DOFS.

The purpose of implementing finite element modelling was to reproduce the structural behavior of the three beams from the experiment, more precisely, the deflection, crack pattern and crack width. The software used throughout the thesis was DIANA FEA BV, release 10.4. The reason of choosing the DIANA was because of its prominence of analyzing reinforced concrete members, not only considering the anisotropic behavior in stress but also the effect of heterogeneity in concrete by applying the JCSS Probabilistic Model Code.

After the reproduction was made the next step was to extract strain data from the bottom reinforcement in the FE model. Because of several reasons, which are further described in Section 5.3, the strain in the reinforcement bar from the FE model is different compared to the strains from the DOFS. Therefore, the strain in the reinforcement bar was postprocessed in MATLAB with the intention of recreating a similar strain profile as in the DOFS.

If the two strain curves become closely related to each other, in the sense of shape and amplitude, the processed strain curve together with the crack information can act as reference for future AI-training. Potentially, AI can become capable of reading strains from other DOFS mounted in future objects and predict cracks based on the strain-crack database collected from artificial FE models.

5.1 Constructing FE model

As demonstrated in Figure 5.1, the geometry, distribution of reinforcement, loading and support setup described in Chapter 4, were replicated in FE model. That means a pinned support placed on the supported steel plate on the left side and a roller support on the right. The simulated load in DIANA was displacement controlled, which was similar to the loading setup described in Section 4.1, but instead of 0.5 mm/min a 0.1 mm/load-step was set until the maximum load of 60 kN was reached. Likewise, same concrete material properties from Table 4.1 were applied in the JCSS Probabilistic Model Code except from the Young's modulus in which the model code is based on the input value of the compressive strength of concrete (basic compressive strength f_{co}) according to equation (5.1). Worth to mention is that the basic compressive strength in DIANA is based on a cylinder of 300x150 mm while the strength from the experiment is tested on a cylinder of 200x100 mm. However, the difference was assumed to be negligible. Moreover, the Young's modulus and Piosson's ratio in the material property of the reinforcement was set to 200 GPa and 0.3 respectively.

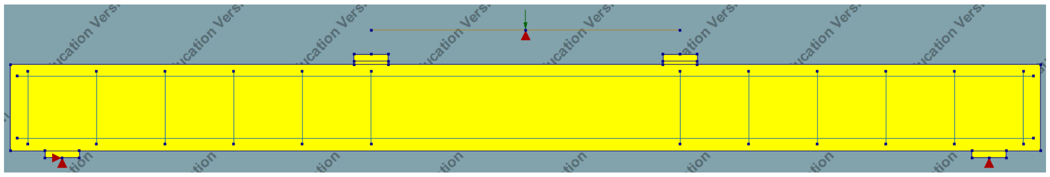


Figure 5.1 Replicated FE model based on the tested beams from experiment.

$$E_c = 10500 * \alpha({f_{co}}^{0.96})^{\frac{1}{3}} * \left(\frac{1}{1+\beta}\right) \tag{5.1}$$

Due to the interest of in-plane stresses the model was created in two dimensions with hexagonal/quadrilateral mesh elements using a size of 10 mm (7826 nodes). The performed analysis was non-linear with Newton Raphson iteration method. The convergence of solution was established on energy, displacement, and force. Other input parameters not mentioned were set as default, see Appendix A for more details on model setup in DIANA.

The prioritized outputs from DIANA were horizontal strains, horizontal and vertical displacements, and reaction forces. All results from Diana were first exported in tabular format, then imported in MATLAB for further data processing.

For higher efficiency, mostly all work in DIANA was executed by commands written in python. Working with Python has been very timesaving when creating models, but also giving the user a better control and overview of given input data. Besides, running many analyses after each other (44 times) during the calibration process, the use of Python was inevitable.

5.2 Model calibration

5.2.1 Selecting random field method in concrete

As we know, concrete is unpredictable due to its heterogeneity and makes it impossible to predict crack locations since every sample behaves differently. This was also the case when comparing the three beams from the experiment. Two of them showed similar deflection and crack widths, whereas the third beam demonstrated a remarkably lower deflection and smaller crack widths.

The effect of heterogeneity is captured in DIANA using the JCSS Probabilistic model code and, because of that, comparing test result with DIANA model was rather inconsistent. Creating FE models with constantly changing results, caused by the random field, made it often hard to compare and understand the impact of specific parameters. Therefore, several repeated analyses had to be made before a general structural behavior could be determined. Another issue regarding the JCSS Probabilistic Model Code was the disability of replicating results. For example, it prevented the user from extracting additional information from a specific analysis if it was already performed.

The most important factors for comparing different random fields with test results from the experiment was the crack amount and maximum displacement. Since the software was incapable of exporting crack locations and crack widths into tabular format, an approximated method to detect cracks was executed in MATLAB by identifying horizontal displacements and notifying when a certain threshold value was reached.

Priority was made on comparing a high quantity of different random field methods and choosing the best fitted model in relation to the tested beams. Nonetheless, the importance of assigning variables on theoretical basis was not to be forgotten. Therefore, a brief explanation of the different chosen input parameters and their characteristics are given below.

The JCSS Probabilistic Model Code in DIANA operate internally with the "total strain rotating crack model" with a linear curve in tension and a constant curve in compression (DIANA FEA BV, 2017). Furthermore, the Young's modulus E_c, tensile strength f_{ct} and compressive strength f_c are all based on the input parameter "basic compressive strength" f_{co}. In the different random field models several parameters were set as constants, whereas others were varying. Figure 5.2 illustrates possible combinations of variables that was used in the FE model.

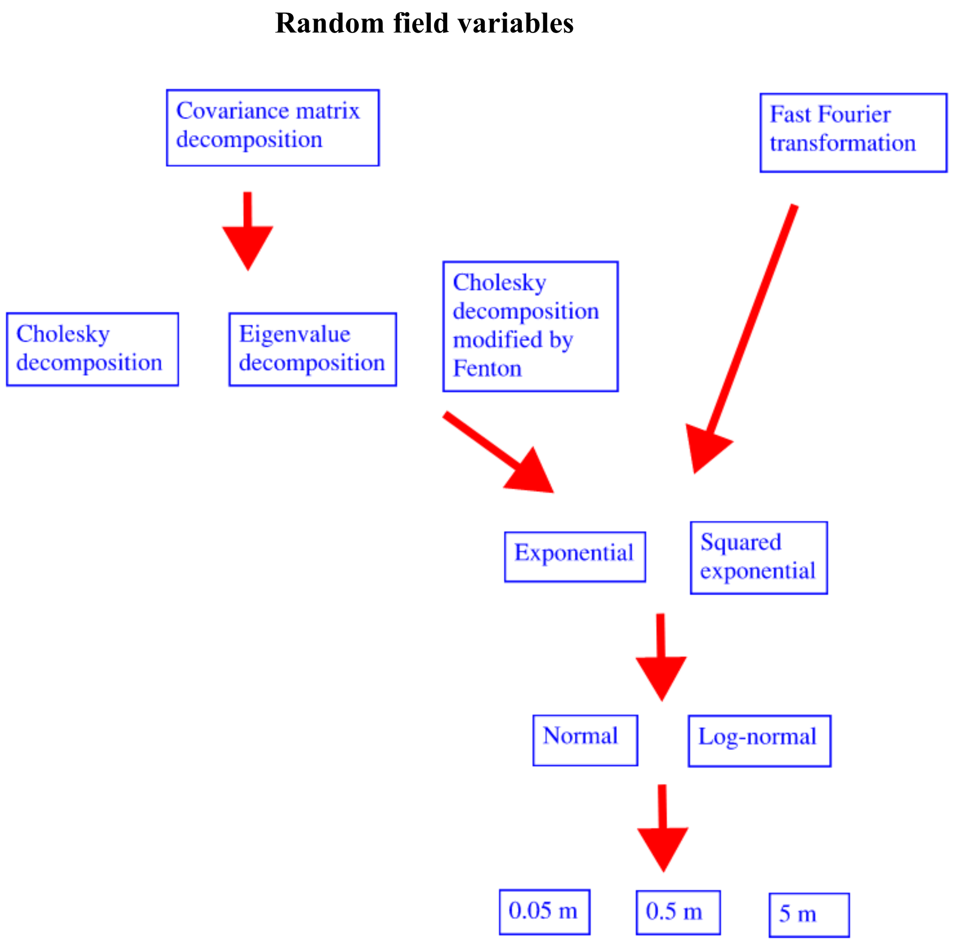


Figure 5.2 Different combinations of the (Joint Committee on Structural Safety, 2020)random field definitions used in FE model.

There are three random field generators, the Covariance Matrix Decomposition (CMD) method, Fast Fourier Transformation (FFT) method and Local Average Subdivision (LAS) method. According to van der Have (2015) the CMD is the most common method and suits small scale structures as for example a beam. This is because the computation time strongly increases when many nodes are applied, for example in a large model. Instead, van der Have points out the FFT as a more efficient and more accurate method when dealing with large scale structures with many numbers of nodes. Nevertheless, the CMD method can be defined in all directions whereas the FFT is limited to the xy-plane. The third method, i.e. the LAS method, was not included in the thesis because of several reasons. According to van der Have (2015), the method is more complex for the user and harder to implement. Also, it is proven to be the least accurate method.

The decomposition of the CMD is defined by three options, i.e. the Cholesky decomposition, eigenvalue decomposition or Cholesky decomposition modified by Fenton. All three methods were considered in the FE model.

According to van den Bos & Garofano (2016), every node in the random field mesh generates random variables which are correlated to each other. How strong the correlation is from a certain distance is determined by a correlation function. It consists of an exponential or squared exponential function including a threshold value and a correlation length. Both the exponential and squared exponential functions were applied to the FE model since they are often used for engineering practices according to van der Have (2015).

The correlation length indicates how much the curvature fluctuates along the elements. Figure 5.3 illustrate the curvature of the compressive strength with correlation lengths of 50 cm and 200 cm along a 500 cm long concrete beam. As stated by van der Have (2015), the correlation length varies from 0.5 to 5 m in most cases. The FE model applied lengths of 0.5 and 5 m, but also 0.05 m because of the dense mesh used in the random field. To demonstrate how the correlation length affect the tensile strength in the FE model a comparison was made between 0.05 m from figure 3.4 above and 0.5 m in Figure 5.4. As seen in Figure 3.4 the predicted strength seems to shift more frequent in a realistic manner compared to Figure 5.4.

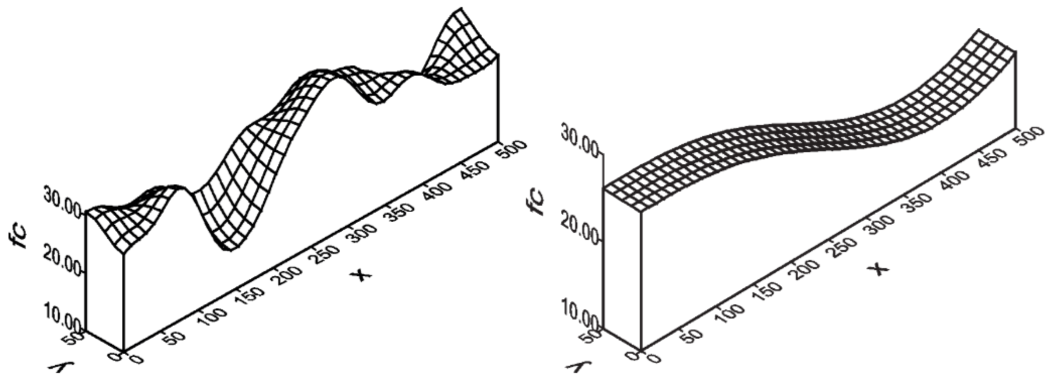


Figure 5.3 The varying compressive strength in a concrete beam using random fields with a correlation lengths of 50 cm versus 200 cm along a 500 cm long beam (van der Have, 2015).

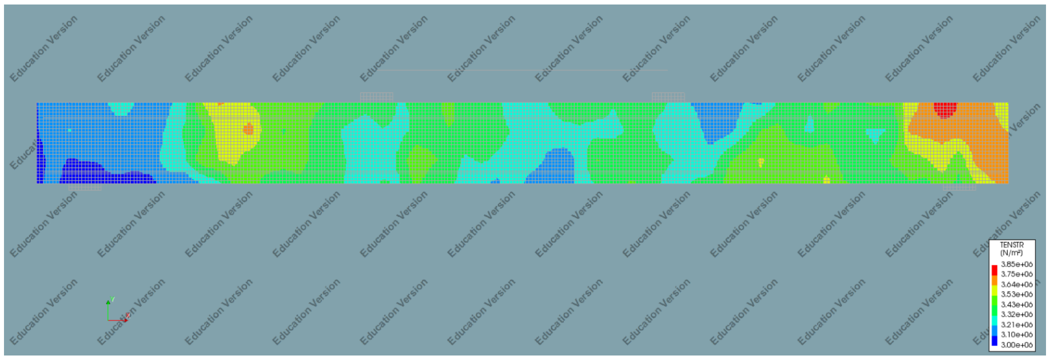


Figure 5.4 A random field of the tensile strength with a correlation length of 0.5 m.

When defining the distribution type in the random field two options are offered, the normal and log-normal distribution type. According to van der Have (2015) the two distribution types are commonly used and were therefore included in the FE model.

To limit the amount of random field combinations some of the parameters were set to constant values. Amongst them was the threshold value which determine the minimum correlation between two locations (DIANA FEA BV, 2017). The threshold value was set to 0.5 according to recommendations from the JCSS Probabilistic Model Code (2020). Other parameters in the random field generator that were assigned by constant values concerned the mesh size. The variables nx, ny and nz are the number of grid lines in the global x, y and z direction for the CMD method (DIANA FEA BV, 2015). With an estimated 2D random field mesh size of 50 x 50 mm the variables nx, ny and nz were defined by the values 60, 5 and 1. According to DIANA FEA BV (2015) the number of grid lines in the Fast Fourier Transformation method are determined by the exponents mx and my, see equation 5.1 and 5.2, and were set to 4 and 0 respectively.

$$N_x = 2^{m_x} \tag{5.1}$$

$$N_y = 2^{m_y} \tag{5.2}$$

5.2.2 Bond-slip

After selecting a suitable random field model described in previous section, the next step was to configure the bond-slip behavior. The bonding strength influence the concentration of cracks and the crack width, and consequently, it can facilitate a better accordance between the FE model and the test results. In contrast to the calibration procedure of the random field model in concrete, adjusting the bond-slip parameter served more as a fine-tuning instrument. Although there are many different input parameters which affect the performance of the bond-slip interface, the major part concerns the maximum and ultimate shear capacity in ultimate limit state (DIANA FEA BV, 2019). According to DIANA FEA BV (2021), the α factor is the exponent of the shear stress function that occur from the slip effect before the maximum stress capacity is reached. Since the tested beams were loaded under service state conditions only, the α factor was the only parameter affecting the bonding strength.

Instead of identifying cracks by reaching a specific limit in lateral displacement described in previous section, the bond-slip method studied the strain peaks at different concrete levels in a more detailed and trustful way. However, this technique is not applicable when looping many different models repeatedly but suits well when analyzing a few as in this case, see Appendix C for the entire procedure written in MATLAB.

5.3 Postprocessing strain data

The output strain data from the lower longitudinal reinforcement in FE model requires processing methods to become comparable to the strains from the DOFS. The reason is that the optical fiber does not fully reflect the strains in the reinforcement since it has a protective outer sheet which smooth out the signals. Also, the DOFS can never be situated inside the reinforcement which one would desire, instead, the fiber needs to be located between the reinforcement and the concrete. This effect leads to disturbance in the strains of the DOFS caused by the surrounding concrete.

Several methods to smooth the artificial strain profile was performed in MATLAB. Especially the functions “sgolayfilt” and “filter” became of interest. Changing systematically the parameters of the functions gave many different types of curves. First, the different curves were visually evaluated based on the smoothness and number of peaks. Then, the best-fitted curves were, together with the strain profile from the DOFS, further assessed by a quantitative study considering the width, prominence, and peak height. The results were presented in tables added with additional information as mean prominence and mean width.

6 Results

6.1 Model calibration

6.1.1 Selecting random field method in concrete

As described in Chapter 5, particularly Section 5.1, the simulated analyses were automatically repeated and the results were extracted to an Excel-file which is presented in Table 6.1. The upper part of the table presents the DIC results from the experiment. Below, it presents results from different combinations of random field methods simulated in DIANA. Although the settlements of supports underneath the steel plate were not considered in Diana, the impact from the settlements in the tested beams was relatively small and could therefore be neglected (Fernandez, 2021).

Table 6.1 Results from different concrete material models simulated in DIANA.

DIC	**Displ. (mm)**	**Crack nr**	**Crack width (10^{-2} mm)**								
Beam 2	6,4	9	78	112	111	98	132	96	78	108	124
Beam 3	6,6	9	141	137	95	77	126	116	150	127	90
Beam 4	5,3	9	124	100	39	105	94	39	107	101	85
Mean	6,1	9									
DIANA											
COVARI_CHOLES_EXPONE_NORMAL_005	5,3	5	0	139	0	131	114	131	0	124	
COVARI_CHOLES_EXPONE_NORMAL_05	5,3	7	119	0	125	116	57	113	100	119	
COVARI_CHOLES_EXPONE_NORMAL_5	5,6	6	113	0	105	118	114	98	62		
COVARI_CHOLES_EXPONE_LOGNOR_005	5,5	7	97	116	64	121	0	128	111	106	
COVARI_CHOLES_EXPONE_LOGNOR_05	5,1	6	123	0	134	0	130	117	66	125	
COVARI_CHOLES_EXPONE_LOGNOR_5	5,2	7	116	125	115	117	125	64	124		
COVARI_CHOLES_SQTYPE_NORMAL_005	5,6	7	139	104	110	28	122	109	104		
COVARI_CHOLES_SQTYPE_LOGNOR_005	5,5	6	106	68	120	127	125	130	0		
COVARI_EIGEN_EXPONE_NORMAL_005	5,6	7	114	124	94	120	118	124	114		
COVARI_EIGEN_EXPONE_NORMAL_05	5,5	6	0	126	114	62	123	115	0	117	
COVARI_EIGEN_EXPONE_NORMAL_5	5,1	5	123	0	134	0	126	0	133	0	123
COVARI_EIGEN_EXPONE_LOGNOR_005	5,6	7	121	119	115	124	122	63	115		
COVARI_EIGEN_EXPONE_LOGNOR_05	5,4	7	113	0	126	0	128	64	125	79	114
COVARI_EIGEN_EXPONE_LOGNOR_5	5,3	7	119	52	133	114	126	119	120		
COVARI_EIGEN_SQTYPE_NORMAL_005	5,3	7	116	17	97	124	62	125	126	0	126
COVARI_EIGEN_SQTYPE_NORMAL_05	5,4	7	119	0	112	62	96	93	68	118	
COVARI_EIGEN_SQTYPE_NORMAL_5	5,2	6	115	99	71	0	71	8	107	117	
COVARI_EIGEN_SQTYPE_LOGNOR_005	5,3	5	126	0	138	0	132	0	127	108	
COVARI_EIGEN_SQTYPE_LOGNOR_05	5,4	6	112	102	70	101	61	0	117		
COVARI_EIGEN_SQTYPE_LOGNOR_5	4,8	6	122	74	26	34	0	74	120		
COVARI_FENTON_EXPONE_NORMAL_005	5,7	7	118	107	113	115	120	119	116		
COVARI_FENTON_EXPONE_NORMAL_05	5,7	7	114	19	111	108	120	0	124	63	101
COVARI_FENTON_EXPONE_NORMAL_5	5,6	8	109	110	114	81	27	113	64	103	
COVARI_FENTON_EXPONE_LOGNOR_005	5,5	6	106	117	119	86	132	131			
COVARI_FENTON_EXPONE_LOGNOR_05	5,5	4	128	121	0	123	113				
COVARI_FENTON_EXPONE_LOGNOR_5	5,4	6	113	0	118	118	25	116	72	113	
COVARI_FENTON_SQTYPE_NORMAL_005	5,6	6	122	100	125	111	118	111			
COVARI_FENTON_SQTYPE_NORMAL_05	5,2	6	126	0	108	0	121	111	67	115	
COVARI_FENTON_SQTYPE_NORMAL_5	5,1	6	120	115	67	132	0	131	0	122	
COVARI_FENTON_SQTYPE_LOGNOR_005	5,4	5	127	0	132	0	97	118	116		
COVARI_FENTON_SQTYPE_LOGNOR_05	5,3	7	118	116	91	26	119	0	76	116	
COVARI_FENTON_SQTYPE_LOGNOR_5	5,3	6	116	64	118	0	106	0	117	115	
FFOURT_EXPONE_NORMAL_005	5,0	6	123	0	126	68	65	126	0	121	
FFOURT_EXPONE_NORMAL_05	5,3	6	122	125	104	107	125	122			
FFOURT_EXPONE_NORMAL_5	5,1	6	121	120	69	12	119	120			
FFOURT_EXPONE_LOGNOR_005	4,9	6	123	89	36	36	86	123			
FFOURT_EXPONE_LOGNOR_05	5,2	6	118	104	63	21	23	117			
FFOURT_EXPONE_LOGNOR_5	4,8	6	122	126	26	26	127	121			
FFOURT_SQTYPE_NORMAL_005	4,8	6	121	72	27	27	76	120			
FFOURT_SQTYPE_NORMAL_05	4,9	6	123	127	37	41	128	122			
FFOURT_SQTYPE_NORMAL_5	5,1	6	121	119	64	68	118	120			
FFOURT_SQTYPE_LOGNOR_005	5,0	6	121	102	30	30	104	120			
FFOURT_SQTYPE_LOGNOR_05	5,2	6	120	110	65	66	111	118			
FFOURT_SQTYPE_LOGNOR_5	5,0	8	119	104	27	28	27	27	106	118	
Max	5,7	8									
Mean	5,3	6,3									

In Table 6.1, the reader can surprisingly notice that the maximum displacement from DIANA is lower than the mean displacement from the DIC. Another interesting observation is the number of cracks. All three beams each indicate nine cracks whereas the mean value of the different random field combinations is not more than 6.3 cracks.

Comparing DIANA with the tested beams from the experiment assumes that the geometry, reinforcement distribution, loading and support setup, and material properties are identical. This is however difficult to prove. During consultation with Fernandez (2021) after the first results were obtained, possible reasons were discussed regarding the difference in stiffness between the model and the beams. According to Fernandez (2021) a beam scanning was performed to verify the location of the reinforcing bars. The outcome showed a 14 mm lower position of the top bars, in other words the distance from the upper surface to the top reinforcement was changed from a theoretical distance of 33 mm to the measured value of 47 mm. Going through more details resulted in additional findings. The applied concrete strength in DIANA was based on a strength test made 6 months and 2 weeks after the casting date. It turned out that the experiment took place several months before the actual strength test. Accordingly, a new compressive strength was estimated to 55 MPa from previous value of 71.5 MPa. Moreover, Fernandez (2021) recommended adjusting the modulus of elasticity from 200 GPa to 195 GPa.

It is evident that the new values decrease the stiffness of the model, and because of that, a new set of analyses was performed, see results in Table 6.2. Even though the mean displacement between DIC and DIANA became almost indistinguishable with the new results, the number of cracks in the model was still notably low. This could be caused by the inconsistent behavior in the random field or/and in the concrete beams. Another reason could be the fact that the fracture energy in the model was set too high since the fracture energy was never changed although the compressive strength in the concrete was decreased. Nevertheless, in the next step of the calibration process, the number of cracks could be adjusted by changing the bond-slip parameter, see next section. For that reason, the general outcome of the different random field methods in the second run proved to be close enough.

Table 6.2 Results from concrete material models using revised geometry and material properties.

DIC	Displ. (mm)	Crack nr	Crack width (10^{-2} mm)									
Beam 2	6,4	9	78	112	111	98	132	96	78	108	124	
Beam 3	6,6	9	141	137	95	77	126	116	150	127	90	
Beam 4	5,3	9	100	39	105	94	39	107	101	85	56	
Mean	6,1	9										
DIANA												
COVARI_CHOLES_EXPONE_NORMAL_005	5,9	7	107	123	27	135	118	64	120			
COVARI_CHOLES_EXPONE_NORMAL_05	6,1	8	109	0	126	94	112	62	117	26	119	
COVARI_CHOLES_EXPONE_NORMAL_5	5,8	8	119	0	123	109	75	114	116	25	114	
COVARI_CHOLES_EXPONE_LOGNOR_005	6,2	9	110	73	116	76	52	70	91	116	87	
COVARI_CHOLES_EXPONE_LOGNOR_05	5,9	8	114	111	108	60	121	126	109	113		
COVARI_CHOLES_EXPONE_LOGNOR_5	6,3	8	92	35	107	97	74	101	77	63	8	
COVARI_CHOLES_SQTYPE_NORMAL_005	6,0	8	100	96	133	122	100	117	65	98		
COVARI_CHOLES_SQTYPE_LOGNOR_005	6,2	9	96	66	0	122	64	102	91	90	72	102
COVARI_EIGEN_EXPONE_NORMAL_005	6,2	8	55	111	102	116	85	74	26	41	18	
COVARI_EIGEN_EXPONE_NORMAL_05	5,9	7	111	69	128	0	129	118	125	27		
COVARI_EIGEN_EXPONE_NORMAL_5	5,8	6	0	127	113	65	112	120	118			
COVARI_EIGEN_EXPONE_LOGNOR_005	6,1	8	114	85	38	0	124	111	59	120	110	
COVARI_EIGEN_EXPONE_LOGNOR_05	6,0	8	92	86	65	111	75	128	119	0	114	
COVARI_EIGEN_EXPONE_LOGNOR_5	5,9	8	116	98	97	89	118	26	125	115		
COVARI_EIGEN_SQTYPE_NORMAL_005	6,0	6	0	119	90	113	120	56	121	0	9	
COVARI_EIGEN_SQTYPE_NORMAL_05	6,0	8	110	26	115	108	80	97	74	109		
COVARI_EIGEN_SQTYPE_NORMAL_5	5,8	6	106	107	71	69	104	106				
COVARI_EIGEN_SQTYPE_LOGNOR_005	6,0	6	0	127	122	126	80	110	86			
COVARI_EIGEN_SQTYPE_LOGNOR_05	5,9	7	111	101	69	89	67	61	19	0	108	
COVARI_EIGEN_SQTYPE_LOGNOR_5	5,9	8	110	109	64	97	85	30	22	110		
COVARI_FENTON_EXPONE_NORMAL_005	6,0	7	114	0	127	71	126	116	114	128		
COVARI_FENTON_EXPONE_NORMAL_05	6,0	7	65	119	69	105	77	119	56			
COVARI_FENTON_EXPONE_NORMAL_5	6,1	7	104	67	115	105	100	104	91			
COVARI_FENTON_EXPONE_LOGNOR_005	6,1	7	94	101	125	110	103	71	95	13		
COVARI_FENTON_EXPONE_LOGNOR_05	5,8	6	92	124	0	135	70	131	72			
COVARI_FENTON_EXPONE_LOGNOR_5	5,8	8	115	0	120	65	114	98	25	127	0	115
COVARI_FENTON_SQTYPE_NORMAL_005	6,0	8	100	125	103	98	108	130	25	115		
COVARI_FENTON_SQTYPE_NORMAL_05	5,9	6	109	59	113	65	20	66	0	110		
COVARI_FENTON_SQTYPE_NORMAL_5	5,9	7	109	110	91	72	34	20	61	104		
COVARI_FENTON_SQTYPE_LOGNOR_005	6,2	7	113	77	109	123	64	108	115			
COVARI_FENTON_SQTYPE_LOGNOR_05	5,9	7	121	115	98	107	103	112	108			
COVARI_FENTON_SQTYPE_LOGNOR_5	6,0	7	108	75	99	59	96	110	107			
FFOURT_EXPONE_NORMAL_005	5,7	6	111	109	69	69	21	109				
FFOURT_EXPONE_NORMAL_05	5,4	6	123	119	71	74	120	122				
FFOURT_EXPONE_NORMAL_5	5,7	6	101	77	33	33	17	100				
FFOURT_EXPONE_LOGNOR_005	5,5	6	112	76	32	33	69	110				
FFOURT_EXPONE_LOGNOR_05	5,6	7	115	114	74	27	16	115	113			
FFOURT_EXPONE_LOGNOR_5	5,7	6	114	102	68	69	105	112				
FFOURT_SQTYPE_NORMAL_005	5,6	6	115	106	70	71	110	113				
FFOURT_SQTYPE_NORMAL_05	5,8	7	109	109	70	30	70	21	107			
FFOURT_SQTYPE_NORMAL_5	5,8	6	109	107	70	71	22	107				
FFOURT_SQTYPE_LOGNOR_005	5,7	6	113	104	69	69	107	112				
FFOURT_SQTYPE_LOGNOR_05	5,9	6	104	104	70	25	25	102				
FFOURT_SQTYPE_LOGNOR_5	5,9	6	105	98	70	71	99	103				
Max	6,3	9										
Mean	5,9	7,0										

The reader can notice some missing rows in the beginning of Table 6.2. It was caused by combining Covariance Matrix Decomposition, Cholesky decomposition and squared exponential function together with a correlation length of more than 0.05 m. For these specific cases the analyses in DIANA were interrupted since the ratio between the correlation length and the element size was set too high.

For further processing one specific random field method from Table 6.2 had to be chosen. Obviously, should the specific combination be in in accordance with the theoretical aspects described in Chapter 3, but more essentially, possess great similarities with the DIC regarding the structural behavior especially when it comes to displacement and number of cracks. The selected random field model consisted of the Covariance Matrix Decomposition method combined with Cholesky decomposition and squared exponential function. Furthermore, the model used a log-normal distribution type together with a correlation length of 0.05 m, see the eighth row in DIANA table in Table 6.2. The selected random field method corresponds well to the theory explained in Chapter 3, Section 2, regarding the advantage of using a correlation length of 0.05 m. In addition, the model presents a displacement of 6.2 mm together with 9 cracks which is very close to the DIC with a mean displacement of 6.1 and a mean crack number of 9.

6.1.2 Bond-slip

As mentioned before, a fast and efficient way to detect cracks was used in previous section at the expense of accuracy. This section is aimed to analyze the crack pattern more profoundly and implement changes in the bond-slip parameter if necessary.

The earlier approximated method to detect cracks was executed in MATLAB and was based on identifying horizontal displacements and notifying when a certain threshold value was reached. The more detailed approach treated in this section was also performed in MATLAB but instead of studying the horizontal displacement, focus was put on horizontal strains along the artificial beam.

As we know concrete can resist a certain amount of tensile stress until the capacity limit is reached. After that, cracks take place. How to decide if a crack is formed by analyzing strain values is not always that simple, especially in the initial phase of a crack. To ensure a crack took place, only strains in the concrete were validated because of a more clear and visible strain profile in contrast to the strains in the reinforcement. In addition, the strains were measured at two different depths, at 5 mm and 36 mm from the bottom surface, the latter is at the level of the bottom reinforcement. Furthermore, five different load steps were examined to follow a detailed crack progression.

Crack detection by analyzing strains was based on finding peaks above a certain threshold value, i.e., the strain limit in concrete when a crack take place. Normally, the maximum tensile strain for concrete is between 0.00015 and 0.00025. In DIANA however, the maximum tensile strain was set by own personal experience and depends on where the strain is measured. In this case, a maximum strain of 0.0008 at 5 mm from bottom surface and a maximum strain of 0.002 at 36 mm seemed to be reasonable although this was 2.5 to 10 times higher than the recommendations.

When running the analyses in DIANA, the selected random field method from previous section was always kept the same. As mentioned before, the only variable used during the bond-slip calibration was the "exponent α" factor which affect the bond-slip interface and thereby the number of cracks. In the first session, consisting of four analyses, exponent α was not changed from previous random field calibration and was therefore set by default to 0.4. Surprisingly, this session showed a stiffer behavior with less deflection and cracks compared to the results from the same model in previous section. An average displacement of 5.9 mm and 7 cracks compared to a displacement of 6.2 mm and 9 cracks. One reason of a changing number of cracks is the new sophisticated method of calculating cracks which can sometimes differ from the "former" method used in previous section, see Section 5.2.1 & 5.2.2 for more information about the methods. Another important aspect of varying results in general is the effect of heterogeneity by implementing the JCSS Probabilistic Model Code which makes it impossible to replicate the structural behavior although the exact same input values are used.

The next session of four analyses was performed with the purpose of increasing the number of cracks slightly, in other words, the exponent α was changed from 0.4 to

0.3. This resulted in an average increase of cracks, more precisely 9 cracks in average, and an unchangeable displacement of 5.9 mm. Figure 6.1 display one of the analyses exposed to the fifth load step, i.e. the final load of 60 kN. The figure consists of two concrete strain profiles at the depth of 5 and 36 mm together with the corresponding crack limit. The dashed vertical lines indicate the constant moment span which is the limited area where all activities were observed. Furthermore, the figure presents the calculated number of peaks using the former method in previous section together with the new method by applying “findpeaks” command in MATLAB. This command is governed by a specific threshold level, in this case the crack limit, and the criteria of a minimum distance of 30 mm between the peaks.

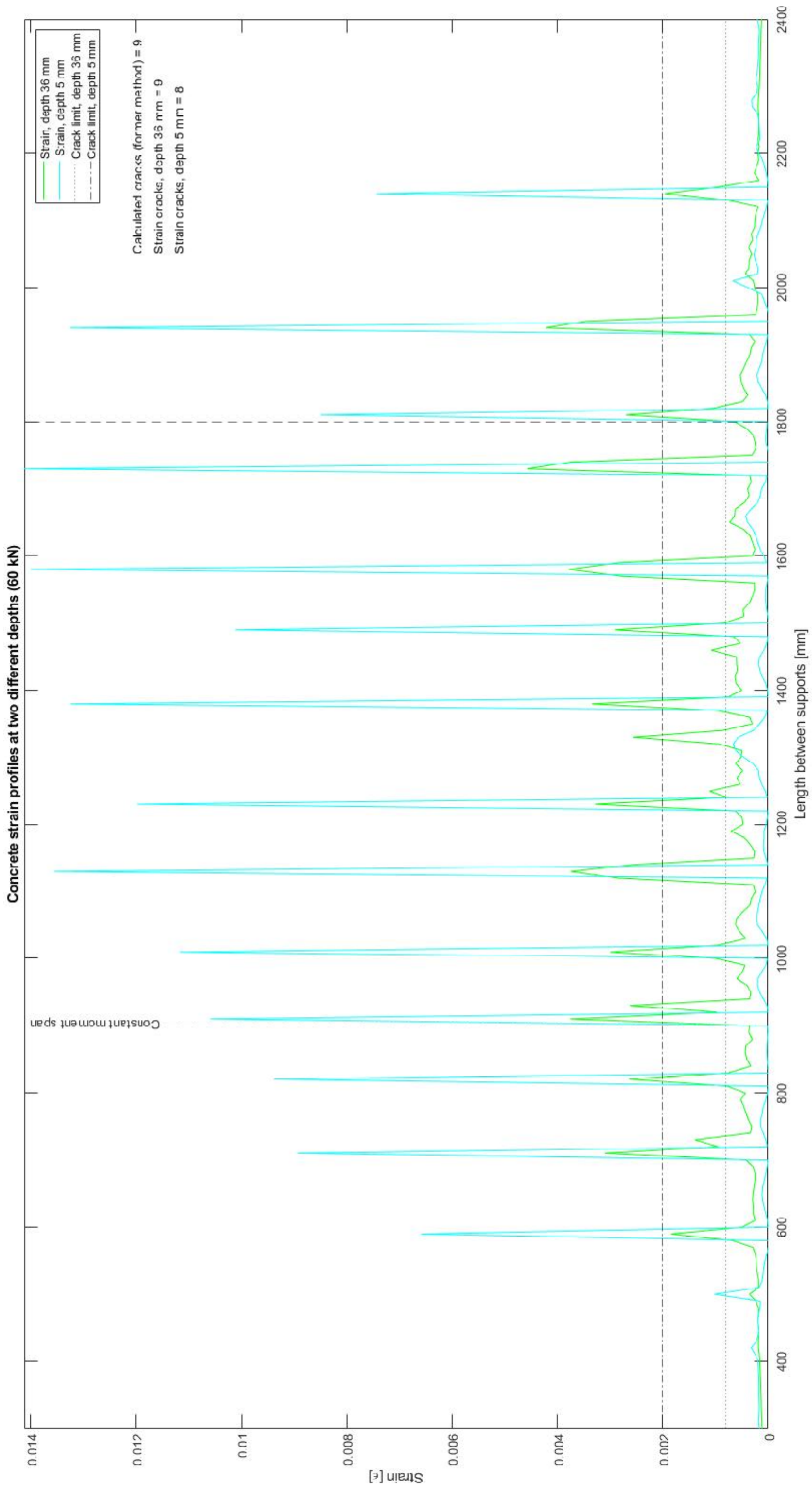


Figure 6.1 Concrete strain profiles from DIANA model at 5 mm and 36 mm from bottom surface.

One interesting observation in Figure 6.1 is the area in the middle part of the beam. At three occasions the strain indicates a crack at the level of reinforcement (depth of 36 mm) but not at the bottom surface (depth of 5 mm). To get a better understanding, Figure 6.2 presents an overlap of the strain profiles plotted in figure 6.1 together with the visual crack pattern in DIANA, both from the same analysis. Apparently, the simulated beam reveals identical crack pattern as the tensile strain profiles. The difference in crack readings at the three locations occurred when the cracks started from one location and transfered towards the center of beam when reaching the reinforcement.

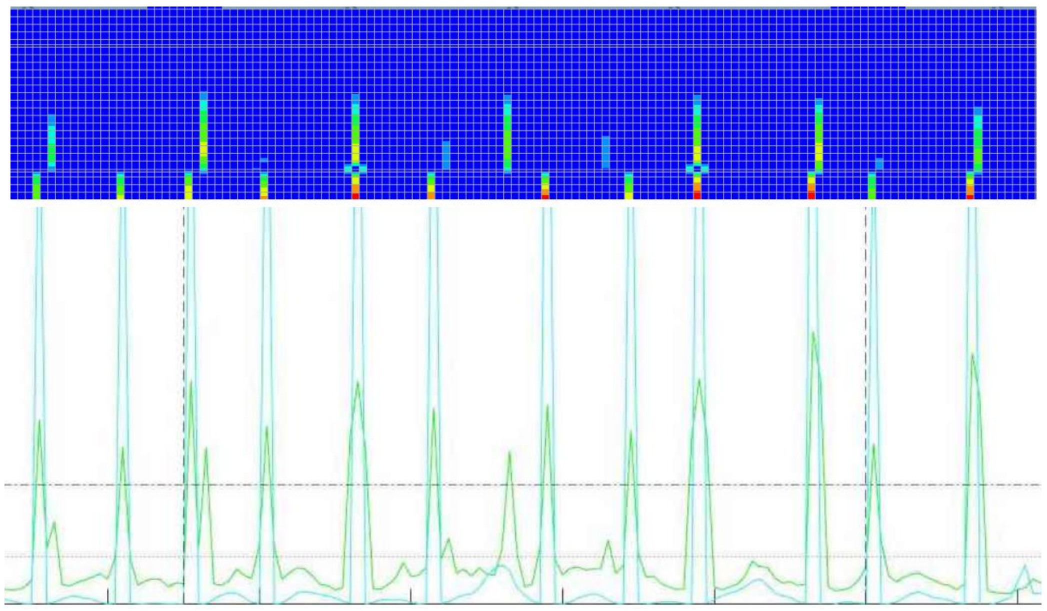


Figure 6.2 Simulated crack pattern from DIANA model to evaluate the obtained strain profiles.

The outcome of changing the "exponent α" from default value of 0.4 to 0.3 seemed promising. Even though the same number of cracks was developed in the model and experiment, more study was made on comparing the strains between the two. Figure 6.3 present the strains in concrete at 36 mm depth (location of the bottom reinforcement) in the three beams from the experiment together with the concrete strains from the DIANA model.

DIC is a great technique to capture vertical displacements or detecting pronounced strain differences. However, reading strains on a high detail level is not very applicable according to Fernandez (2021). Figure 6.3 illustrate this problem where the strains from FE model are more pronounced compared to the DIC. In addition, DIC illustrate almost no existing strain between the peaks since the values are too small. Although it is hard to verify the actual strains, the strains between the cracks in the DIANA model are probably closer to the "real" behavior by the fact of an existing tensile stress throughout the bottom part of the beam.

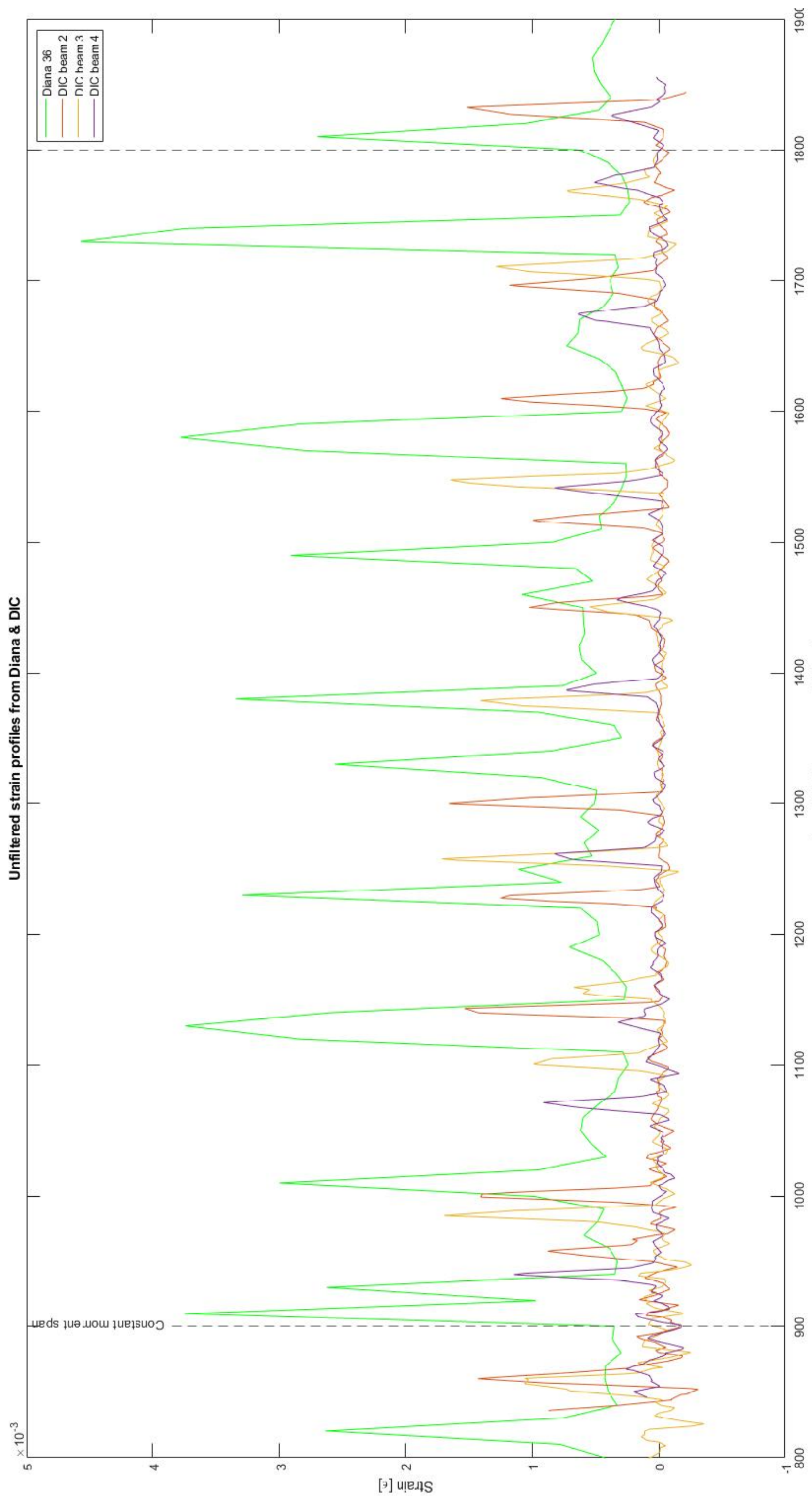


Figure 6.3 Concrete strain profiles from DIANA model and DIC.

To make the strains more comparable the entire strain interval from DIANA model was lowered by 0.00055. By doing so, the strain valleys from the model were set close to zero, similar to the strains from the DIC. In addition, a filter was implemented to exclude low strain values between the peaks, see Figure 6.4.

Even after the strain reduction and the added filter, Figure 6.4 still indicate a huge difference between the peaks from DIANA model and DIC. The average strain peak from the four strain curves is presented in Table 6.3.

Despite the low detail level of the DIC, figure 6.4 illustrate that the peak distance and the number of peaks between the model and DIC correspond well to each other. For that reason, the value of 0.3 for "exponent α" appeared to be a good estimation.

Table 6.3 Average strain of peaks.

Source	**Average peak strain**
DIANA model	0.0029
DIC beam 2	0.0012
DIC beam 3	0.0011
DIC beam 4	0.0007

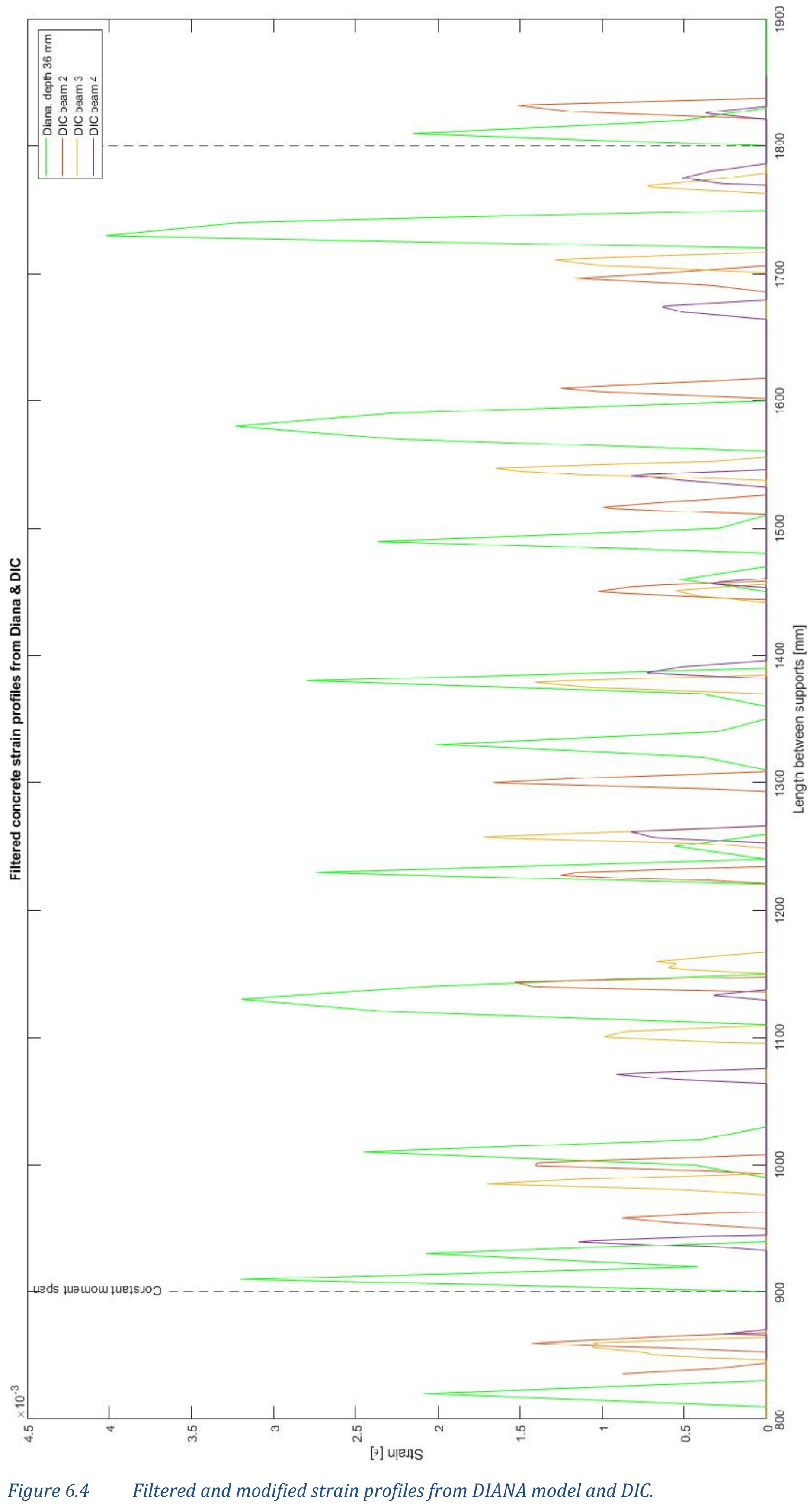


Figure 6.4 Filtered and modified strain profiles from DIANA model and DIC.

6.2 Postprocessing strain data

The constructed FE model was proved to have similar structural performance as the tested beams from the experiment. This created the possibility to investigate if the artificial strain in the reinforcement is comparable to the strains from the DOFS. If they become comparable, strain peaks (cracks) from DOFS can be identified by using the artificial strain as a reference,

Primary, it was of interest to see how the strains in the reinforcement and in the DOFS behaves in relation to the strains in the concrete at the same depth. Figure 6.5 illustrate a close relationship between the artificial strain in the concrete and reinforcement. The total number of nine peaks in the reinforcement correspond well to the number of peaks in the concrete (neglecting peaks closer than 30 mm). Figures 6.6 to 6.8 illustrate the relationship between the DIC in the concrete and the DOFS. In contrast to Figure 6.5 the curves from the DOFS are smoother and do not capture all the peaks from the DIC. This is probably caused by the protective outer sheet that is mounted around the optical fiber.

Not only is the strain from the reinforcement in Figure 6.5 more detailed but also the fact that the difference in amplitude between the peaks and valleys is higher. Figure 6.9 demonstrate the dissimilarities by gathering the strains from the DOFS together with the artificial strain from the reinforcement.

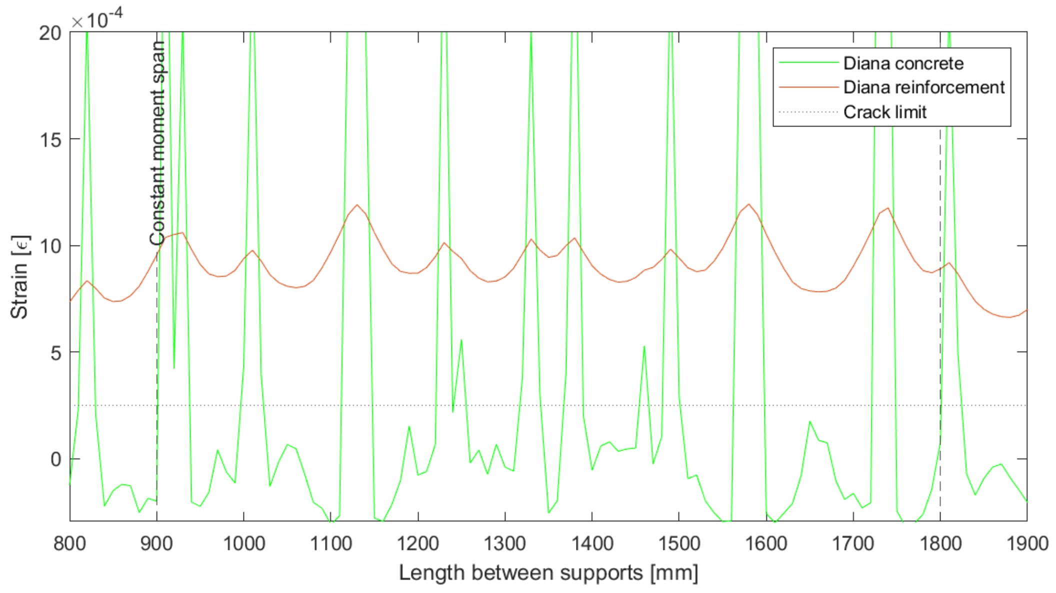


Figure 6.5 Strain from concrete vs strain from reinforcement in DIANA model.

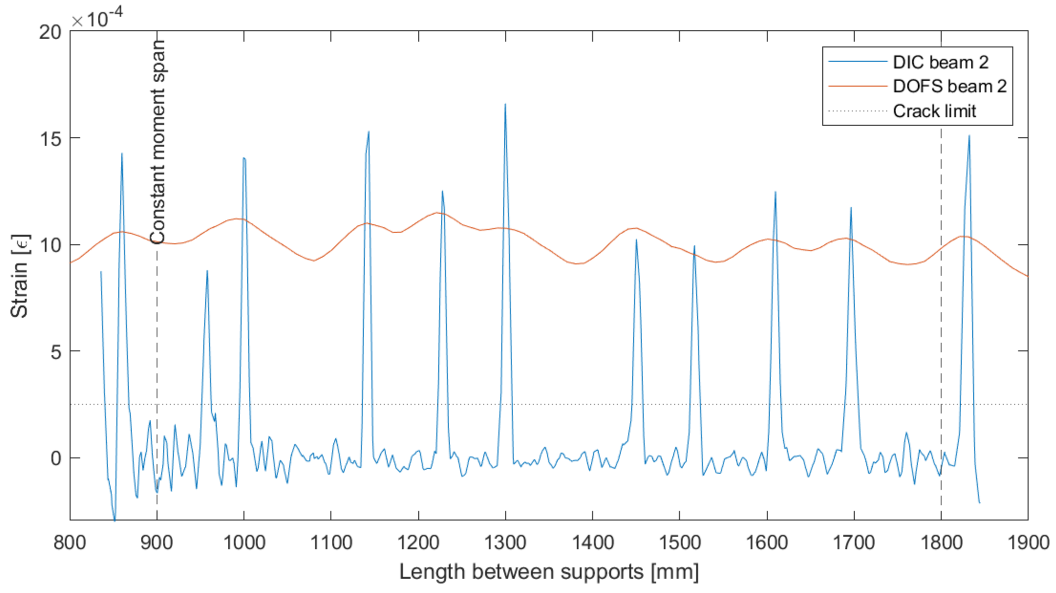


Figure 6.6 Strain from DIC vs strain from DOFS, beam 2.

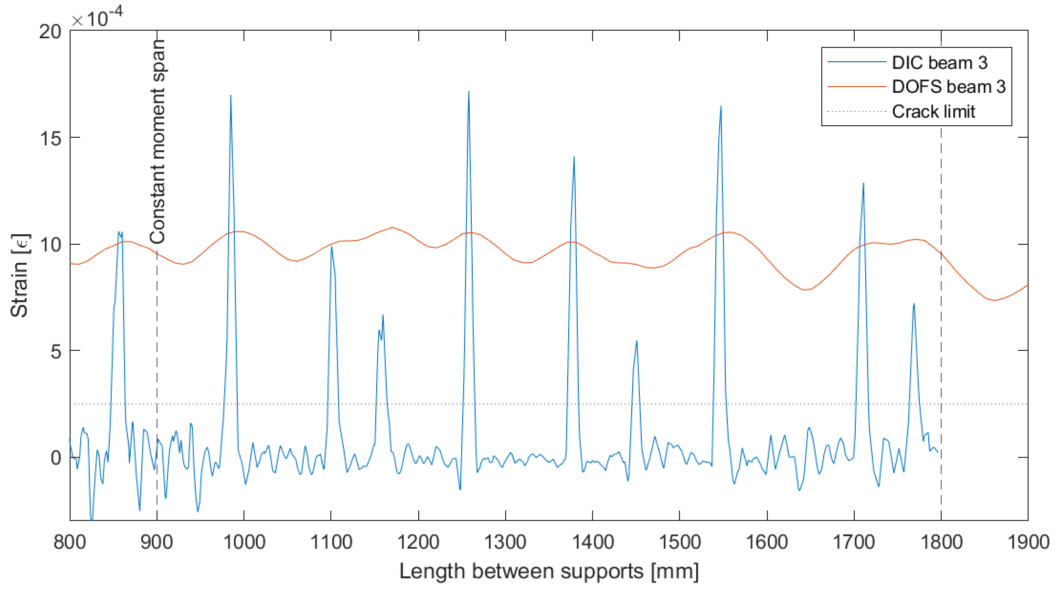


Figure 6.7 Strain from DIC vs strain from DOFS, beam 3.

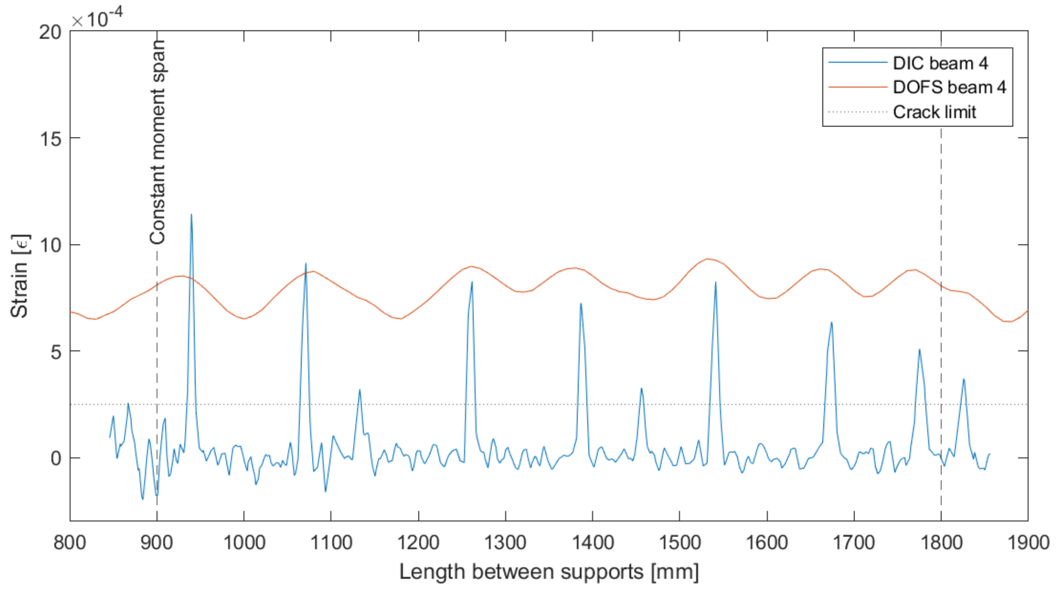


Figure 6.8 Strain from DIC vs strain from DOFS, beam 4.

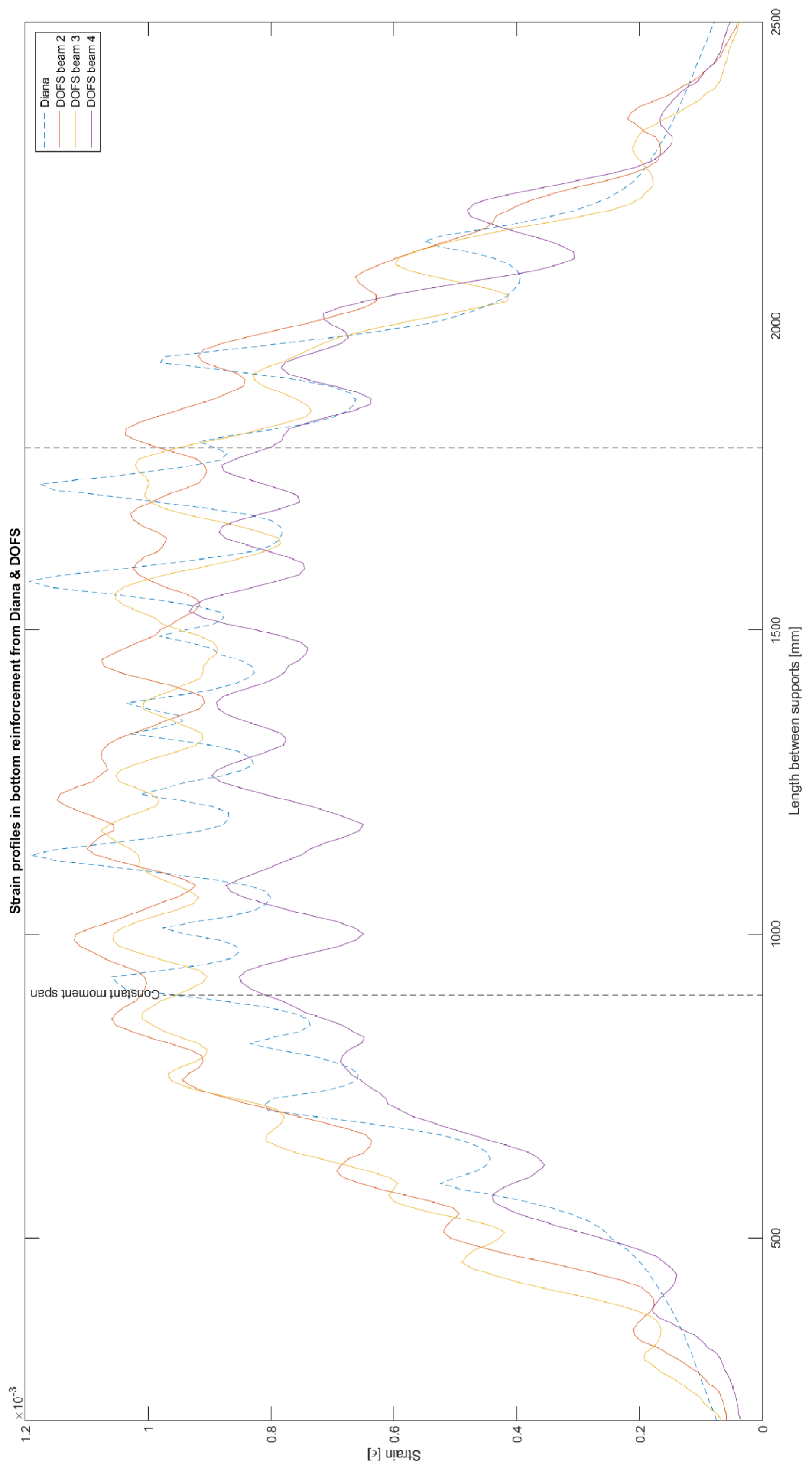


Figure 6.9 Strain profiles from DOFS and from the reinforcement in DIANA model.

To investigate the possibility of predicting cracks using DOFS in future elements, it was first necessary to study the possibility of predicting cracks in the existing DOFS. The artificial strain needed to be adopted to the shape of the strains coming from the DOFS. However, the problem was that the DOFS were not sensitive enough to reflect all the cracks, see Figure 6.6 to 6.8. That means that even though the shape of the artificial strain would become more or less identical to the strains from the DOFS, the predicted amount of cracks would not cover all the cracks that took place in the tested beams. Nonetheless, the work continued attempting to process the artificial strain into a similar shape as the strains from the DOFS.

To simplify the processing procedure only two of the three strain profiles from the DOFS were included. The reason was that the third profile, i.e. beam 4, differed too much from the others, see Figure 6.9. With the purpose of modifying the strain profile in an efficient and suitable way several filter commands in MATLAB were investigated, especially the commands "sgolayfilt" and "filter". "Sgolayfilt" presented more satisfying results and was therefore chosen for further work. The two different input values in "sgolayfilt" consisted of degree and framelength. In the interest of comparing many different input values a loop consisted of 25 combinations was analyzed. 9 of these combinations displayed interesting curves which in turn was categorized into 3 groups depended on their shape, see Figure 6.10. As seen in the figure, the 3 groups are categorized by color. In addition, the two strain profiles from the DOFS are added to make the comparison more clear.

As the reader can observe, it is complicated to favor one group over the other since they all show different qualities. The filtered strains in the color cyan are the smoothest curves with less variation. These curves have more rounded peaks and do not reflect all the peaks from the original strain profile which is also the case in the DOFS. In contrast, the filtered strains in black indicate more variation and a higher resolution. These characteristics are observed in the DOFS as well, but on the other hand, it captures all the peaks which is not the case with the DOFS.

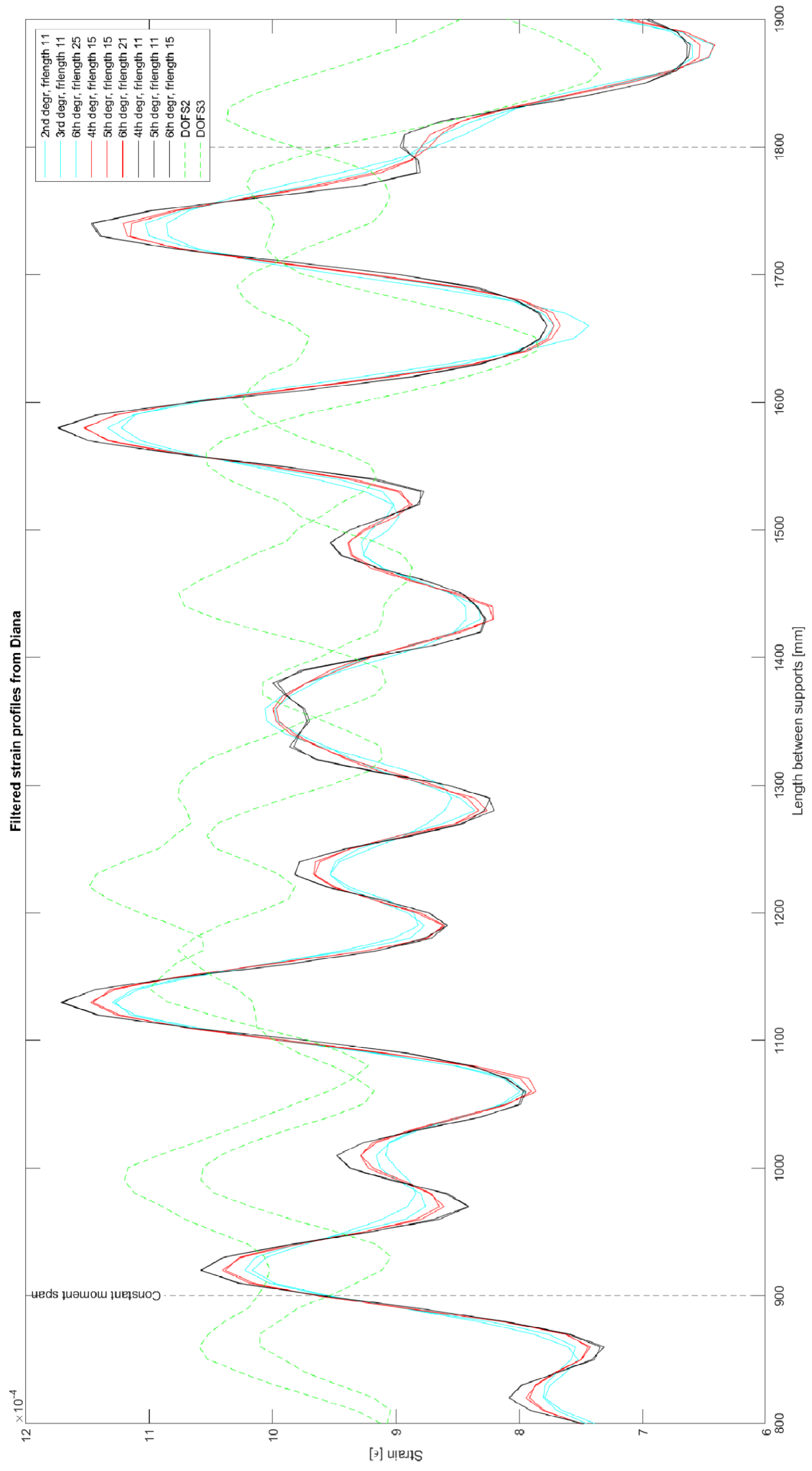


Figure 6.10 Filtered strain profiles from DIANA together with two strain profiles from the DOFS.

Since it is hard to decide by observation which group of curves is more similar to the DOFS, a quantitative study was added considering the width and prominence. The results were obtained from the two DOFS together with the filtered strains from the two groups which differed the most, i.e. the group in color cyan and color black. The results are presented in Table 6.4 and 6.5, and Figures 6.11-6.14.

The quantitative study indicates what was mentioned before, the challenge of selecting a specific filter that cover the entire behavior of the DOFS. The strain profile needs to be detailed enough to detect the minor variances in the slope, i.e. when minor cracks appear between larger and more prominent cracks. Contrary, the strain profile should imitate the smooth shape of the DOFS which include small slopes, rounded peaks, and low variation in amplitude.

Table 6.4 Quantitative comparison of strain profiles from MATLAB filters and DOFS.

Filter: 2nd degree, framelength 11									
Peak location (mm)	920	1010	1130	1230	1360	1490	1580	1740	
Peak height (10^{-4})	10	9.2	11	9.5	10	9.3	11	11	
Prominence (10^{-4})	0.66	0.40	3.0	0.76	1.6	0.26	3.3	2.2	
Width (mm)	30	35	65	45	88	31	70	57	
Filter: 4th degree, framelength 11									
Peak location (mm)	920	1010	1130	1230	1330	1380	1490	1580	1740
Peak height (10^{-4})	11	9.5	12	10	10	10	10	12	11
Prominence (10^{-4})	0.99	1.1	3.5	1.2	0.16	1.7	0.73	3.8	2.6
Width (mm)	29	38	56	42	14	91	36	60	47
DOFS, beam 2									
Peak location (mm)	990	1140	1220	1290	1450	1600	1690		
Peak height (10^{-4})	11	11	11	11	11	10	10		
Prominence (10^{-4})	1.2	0.44	2.3	0.11	1.7	0.55	1.1		
Width (mm)	64	32	210	23	73	43	69		
DOFS, beam 3									
Peak location (mm)	990	1170	1260	1370	1560	1720	1770		
Peak height (10^{-4})	11	11	11	10	11	10	10		
Prominence (10^{-4})	1.4	1.7	0.71	0.97	1.7	0.074	0.37		
Width (mm)	69	110	41	50	83	14	34		

Table 6.5 Comparison of mean prominence and mean width.

Strain profile	**2nd degree, framelength 11**	**4th degree, framelength 11**	**DOFS, beam 2**	**DOFS, beam 3**
Mean prominence (10^{-4})	1.53	1.75	1.05	0.99
Mean width (mm)	52.7	45.8	73.4	57.9

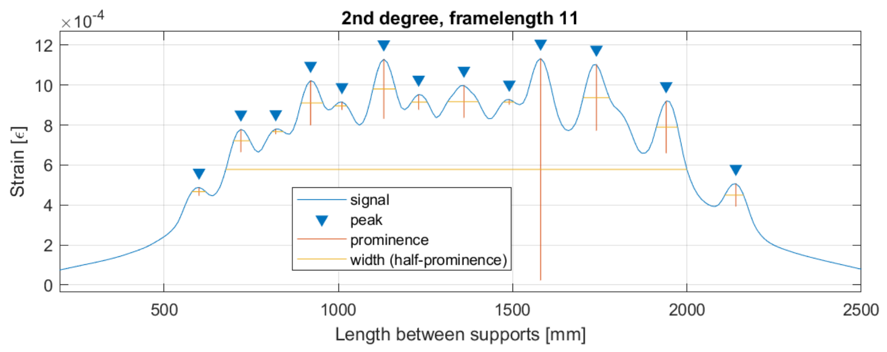


Figure 6.11 Filtered strain profile: Illustration of peaks and their prominence and width.

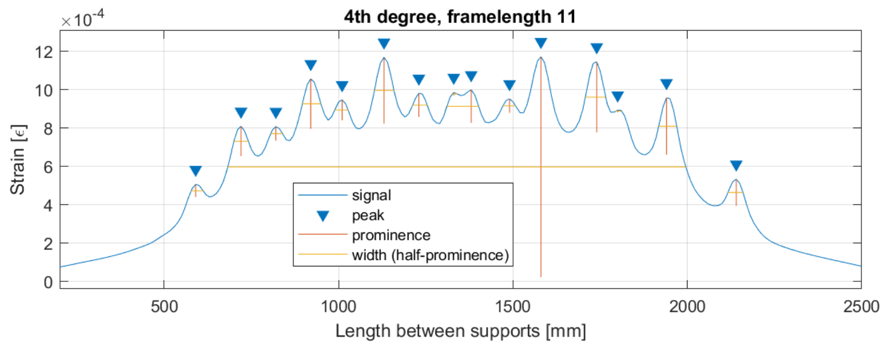


Figure 6.12 Filtered strain profile: Illustration of peaks and their prominence and width.

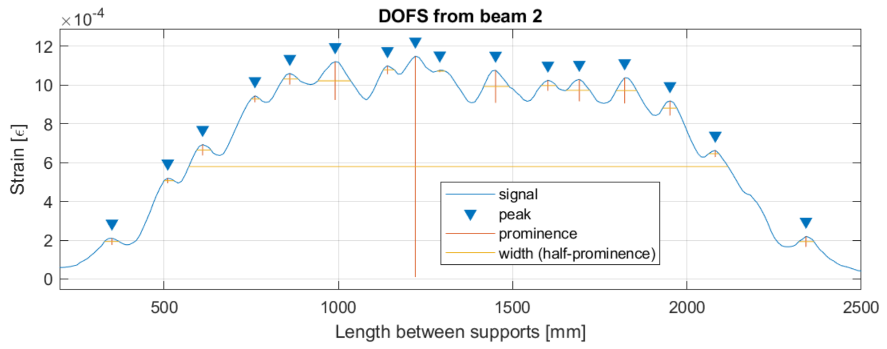


Figure 6.13 Strain from DOFS in beam 2: Illustration of peaks and their prominence and width.

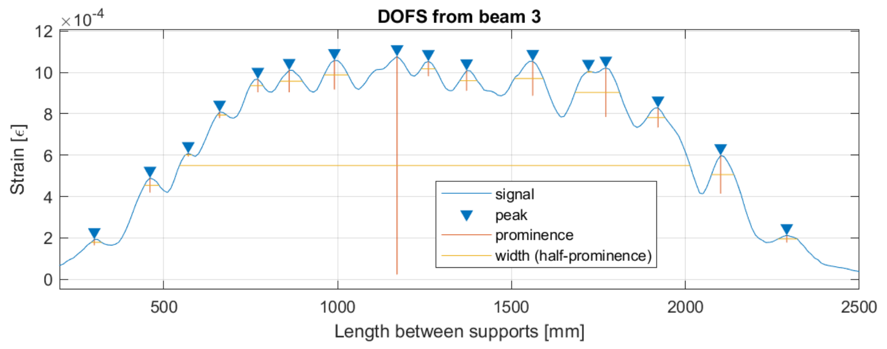


Figure 6.14 Strain from DOFS in beam 3: Illustration of peaks and their prominence and width.

7 Conclusion and further research work

There is a great potential of creating an AI training database by implementing FEA. To generate a database with pure experimental data would be very time consuming and expensive. Also, the inconsistent behavior of concrete may need many test results before a general failure mode can be revealed. Although these problems are avoided when creating an artificial database the method generate another concern, which is the compatibility issue.

As the results from Chapter 6 indicate, it is problematic to make the artificial strain in the reinforcement comparable to the strains in the DOFS. The main reason is probably due to the DOFS itself since some of the cracks are not detected. In the report of Berrocal et al. (2021), a comparison was made between thin DOFS and the DOFS used in this project which is categorized as robust, see Figure 7.1 & 7.2. As Figure 7.2 illustrate, the thin DOFS give rise to strain reading anomalies and are able to exclude meaningful data. On the other hand, the chance of missing out cracks seems to be very small.

Without knowing the technical details, it would be of interest to study other types of DOFS which are more sensitive than the robust fibers, or on the contrary, less responsive than the thin fibers. Maybe then, the peaks can become distinct enough without the effect of inaccurate readings. If that would be the case, further research can take on the challenge of constructing a future AI training database by collecting processed artificial strain data and their crack pattern from numerous FE models.

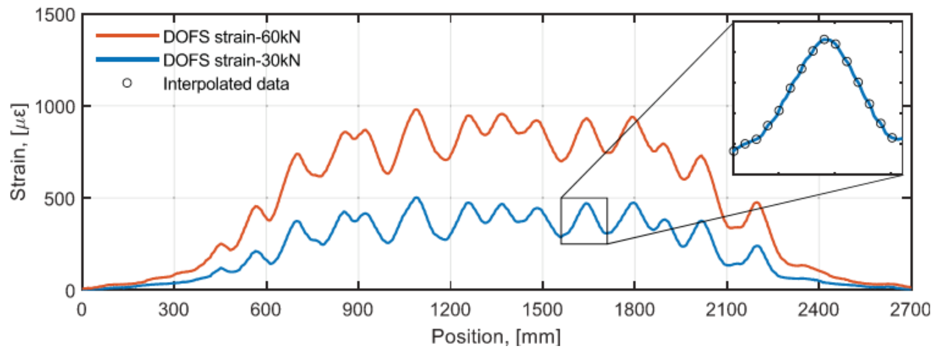


Figure 7.1 *Two strain profiles from the experiment obtained at different load levels using robust DOFS (Berrocal et al. 2021)*

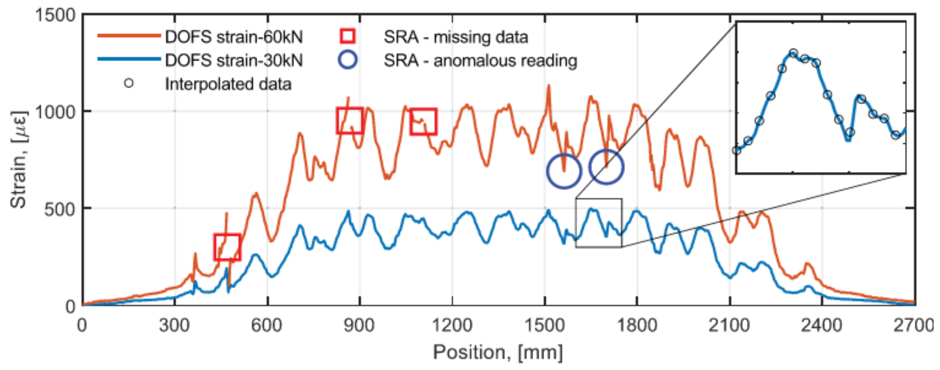


Figure 7.2 *Two strain profiles from the experiment obtained at different load levels using robust DOFS (Berrocal et al. 2021).*

8 References

Abdeljaber, O., Avci, O., Gabbouj, M., Hussein, M., Inman, D., & Kiranyaz, S. (2020). *A review of vibration-based damage detection in civil structures: From traditional methods to Machine Leraning and Deap Learning applications.* Elsevier.

Ansari, F., Song, Q., Tang, G., & Zhang, C. (2020). *Deep learning method for detection of structural microcracks by brillouin scattering based distributed optical fiber sensors.* IOP Publishing.

Barrias, A., Casas, J. R., & Villalba, S. (2016). *A Review of Distributed Optical Fiber Sensors for Civil Engineering Applications.* Catalonia: MDPI.

Berrocal, C. G., Casas, J. R., Fernandez, I., Francesco Bado, M., & Rempling, R. (2021). *Assessment and visualization of performance indicators of reinforced concrete beams by distributed optical fibre sensing.* Sage Publications.

Catbas, F. N., & Dong, C.-Z. (2021). *A review of computer vision-based structural health monitoring at local and global levels.* Sage.

DIANA FEA BV. (2015). *DIANA-10.0*. Retrieved from User's manual: https://dianafea.com/manuals/d100/MatLib/node188.html

DIANA FEA BV. (2017). *DIANA-10.1*. Retrieved from User's Manual: https://dianafea.com/manuals/d101/RelNot/node27.html

DIANA FEA BV. (2019, 08 20). *Bondslip models.* Retrieved from https://dianafea.com/system/files/bondslipModels.pdf

Du, Y., Sun, B., Li, J., & Zhang, W. (2020). *Optical Fiber Sensing and Structural Health Monitoring Technology.* Wuhan: Springer.

Fernandez, I. (2021, Februari 17). Associate Professor. (P. Nässlander, Interviewer)

Han, J. P., & Su, Y. H. (2014). Structural Health Monitoring of Civil Infrastructure Using Optical Fiber Sensing Technology: A Comprehensive Review. *Hindawi Publishing Corporation*, 11.

Jin, T., Ye, X., & Yun, C. (2019). *A review on deep learning-based structural health monitoring of civil infrastructures.* Researchgate.

Joint Committee on Structural Safety. (2020, 10 10). *JCSS.* Retrieved from Probabilistiv model code, Part III: https://www.jcss-lc.org/publications/jcsspmc/part_iii.pdf

Karypidis, D. (2019). *Structural Health Monitoring of Concrete Elements Using Deep Machine Learning.* Gothenburg: Chalmers University of Technology.

Plos, M. (2000). *Finite element analyses of reinforced concrete structures.* Göteborg: Chalmers University of Technology.

Rijkswaterstaat. (2016). *Guidelines for Nonlinear Finite Element Ananlysis of Concrete Structures.* Ministry of Infrastructure and the Environment.

van den Bos, A., & Garofano, A. (2016). Crack Predictions Using Random FIelds. *ResearcgGate*, 11.

van der Have, R. (2015). *Random Fields for Non-Linear Finite Element Analysis of Reinforced Concrete.* Delft: Delft University of Technology.

APPENDIX A
Python script for constructing FE model and running analyses

```
import math as np


L = 3
h = 0.25
j = 0.8
D = 0.016
Dp = 0.010

b = h*j;

name = 'beam_'

nbars = 3
nbarsp = 2
nstirrups = np.ceil(0.3*L/0.2)+1

As = 0.000197*nbars         # np.pi*D**2/4*nbars = 0.000201
Asp = np.pi*Dp**2/4*nbarsp
Ass = np.pi*8**2/4*2

us = np.pi*D*nbars
usp = np.pi*Dp*nbarsp
uss = np.pi*8*2

rho = As/(b*(h/2))

alf = 6.8
d = h-0.028-D/2
dp = 0.047                  # 0.028+Dp/2=0.033
A = b/2
B = (alf-1)*Asp + alf*As
C = -((alf-1)*Asp*dp + alf*As*d)

x = (-B + np.sqrt(B**2-4*A*C))/(2*A)
Ifis = b*x**3/3 + (alf-1)*Asp*(x-dp)**2 + alf*As*(d-x)**2

Mu = 550000000*As*0.9*d
Fu = Mu/(0.3*L)
Delta_max = 0.6*Fu*0.3*L*(3*(0.9*L)**2 - 4*(0.3*L)**2)/(24*33000000000*Ifis)


newProject( "Z:\Beam\Beam_L" + str(L) +"/" + name, 100 )
setModelAnalysisAspects( [ "STRUCT" ] )
setModelDimension( "2D" )
setDefaultMeshOrder( "LINEAR" )
setDefaultMesherType( "HEXQUAD" )
createSheet( "Concrete", [[ 0, 0, 0 ],[ L, 0, 0 ],[ L, h, 0 ],[ 0, h, 0 ]] )
createSheet( "Left_support_plate", [[ 0.05*L-0.05, -0.02, 0 ],[ 0.05*L+0.05, -0.02, 0
],[ 0.05*L+0.05, 0, 0 ],[ 0.05*L-0.05, 0, 0 ]] )
createSheet( "Right_support_plate", [[ 0.95*L-0.05, -0.02, 0 ],[ 0.95*L+0.05, -0.02, 0
],[ 0.95*L+0.05, 0, 0 ],[ 0.95*L-0.05, 0, 0 ]] )
createSheet( "Left_wooden_sheet", [[ 0.35*L-0.05, h, 0 ],[ 0.35*L+0.05, h, 0 ],[
0.35*L+0.05, h+0.01, 0 ],[ 0.35*L-0.05, h+0.01, 0 ]] )
createSheet( "Right_wooden_sheet", [[ 0.65*L-0.05, h, 0 ],[ 0.65*L+0.05, h, 0 ],[
0.65*L+0.05, h+0.01, 0 ],[ 0.65*L-0.05, h+0.01, 0 ]] )
createSheet( "Left_load_plate", [[ 0.35*L-0.05, h+0.01, 0 ],[ 0.35*L+0.05, h+0.01, 0
],[ 0.35*L+0.05, h+0.03, 0 ],[ 0.35*L-0.05, h+0.03, 0 ]] )
createSheet( "Right_load_plate", [[ 0.65*L-0.05, h+0.01, 0 ],[ 0.65*L+0.05, h+0.01, 0
],[ 0.65*L+0.05, h+0.03, 0 ],[ 0.65*L-0.05, h+0.03, 0 ]] )
createLine( "Line 1", [ 0.35*L, h+0.1, 0 ], [ 0.65*L, h+0.1, 0 ] )
createPointBody( "point 1", [ L/2, 2*h, 0 ] )
projection( "SHAPEEDGE", "Line 1", [[ L/2, h+0.1, 0 ]], [ "point 1" ], [ 0, -1, 0 ],
True )
removeShape( [ "point 1" ] )
createPointBody( "point 2", [ 0.05*L, -0.2, 0 ] )
createPointBody( "point 3", [ 0.95*L, -0.2, 0 ] )
createPointBody( "point 4", [ 0.35*L, h*2, 0 ] )
createPointBody( "point 5", [ 0.65*L, h*2, 0 ] )
projection( "SHAPEEDGE", "Left_support_plate", [[ 0.05*L, -0.02, 0 ]], [ "point 2" ],
[ 0, 1, 0 ], True )
removeShape( [ "point 2" ] )
projection( "SHAPEEDGE", "Right_support_plate", [[ 0.95*L, -0.02, 0 ]], [ "point 3" ],
[ 0, 1, 0 ], True )
removeShape( [ "point 3" ] )
projection( "SHAPEEDGE", "Left_load_plate", [[ 0.35*L, h+0.03, 0 ]], [ "point 4" ], [
0, -1, 0 ], True )
```

```
removeShape( [ "point 4" ] )
projection( "SHAPEEDGE", "Right_load_plate", [[ 0.65*L, h+0.03, 0 ]], [ "point 5" ], [
0, -1, 0 ], True )
removeShape( [ "point 5" ] )
createLine( "Bottom_reinf", [ 0.02, 0.028+D/2, 0 ], [ L-0.02, 0.028+D/2, 0 ] )
createLine( "Top_reinf", [ 0.02, h-dp, 0 ], [ L-0.02, h-dp, 0 ] )       # h-0.028-Dp/2
for i in range(1,int(nstirrups)+1):
    createLine( "Stirrup_" + str(i), [ 0.05*L+0.3*L-(i-1)*0.2, 0.02, 0 ], [
0.05*L+0.3*L-(i-1)*0.2, h-0.02, 0 ] )
    createLine( "Stirrup_" + str(2*nstirrups-(i-1)), [ 0.95*L-0.3*L+(i-1)*0.2, 0.02, 0
], [ 0.95*L-0.3*L+(i-1)*0.2, h-0.02, 0 ] )


####################  Add supports  ####################
addSet( "GEOMETRYSUPPORTSET", "Supports" )
createPointSupport( "Support_left", "Supports" )
setParameter( "GEOMETRYSUPPORT", "Support_left", "AXES", [ 1, 2 ] )
setParameter( "GEOMETRYSUPPORT", "Support_left", "TRANSL", [ 1, 1, 0 ] )
setParameter( "GEOMETRYSUPPORT", "Support_left", "ROTATI", [ 0, 0, 0 ] )
attach( "GEOMETRYSUPPORT", "Support_left", "Left_support_plate", [[ 0.05*L, -0.02, 0
]] )
createPointSupport( "Support_right", "Supports" )
setParameter( "GEOMETRYSUPPORT", "Support_right", "AXES", [ 1, 2 ] )
setParameter( "GEOMETRYSUPPORT", "Support_right", "TRANSL", [ 0, 1, 0 ] )
setParameter( "GEOMETRYSUPPORT", "Support_right", "ROTATI", [ 0, 0, 0 ] )
attach( "GEOMETRYSUPPORT", "Support_right", "Right_support_plate", [[ 0.95*L, -0.02, 0
]] )
addSet( "GEOMETRYSUPPORTSET", "Load_plate" )
createPointSupport( "Support_1", "Load_plate" )
setParameter( "GEOMETRYSUPPORT", "Support_1", "AXES", [ 1, 2 ] )
setParameter( "GEOMETRYSUPPORT", "Support_1", "TRANSL", [ 0, 1, 0 ] )
setParameter( "GEOMETRYSUPPORT", "Support_1", "ROTATI", [ 0, 0, 0 ] )
attach( "GEOMETRYSUPPORT", "Support_1", "Line 1", [[ 0.5*L, h+0.1, 0 ]] )

####################  Add tyings  ####################
addSet( "GEOMETRYTYINGSET", "Support_plates" )
createPointTying( "Left_tying", "Support_plates" )
setParameter( "GEOMETRYTYING", "Left_tying", "AXES", [ 1, 2 ] )
setParameter( "GEOMETRYTYING", "Left_tying", "TRANSL", [ 0, 1, 0 ] )
setParameter( "GEOMETRYTYING", "Left_tying", "ROTATI", [ 0, 0, 0 ] )
attachTo( "GEOMETRYTYING", "Left_tying", "SLAVE", "Left_load_plate", [[ 0.35*L,
h+0.03, 0 ]] )
attachTo( "GEOMETRYTYING", "Left_tying", "MASTER", "Line 1", [[ 0.35*L, h+0.1, 0 ]] )
createPointTying( "Right_tying", "Support_plates" )
setParameter( "GEOMETRYTYING", "Right_tying", "AXES", [ 1, 2 ] )
setParameter( "GEOMETRYTYING", "Right_tying", "TRANSL", [ 0, 1, 0 ] )
setParameter( "GEOMETRYTYING", "Right_tying", "ROTATI", [ 0, 0, 0 ] )
attachTo( "GEOMETRYTYING", "Right_tying", "SLAVE", "Right_load_plate", [[ 0.65*L,
h+0.03, 0 ]] )
attachTo( "GEOMETRYTYING", "Right_tying", "MASTER", "Line 1", [[ 0.65*L, h+0.1, 0 ]] )

####################   Material Definitions  ###################
## Concrete Material
addMaterial( "Concrete", "CONCDC", "JCSSPR", [ "JCSSRF" ] )
setParameter( "MATERIAL", "Concrete", "JCSSMC/JCSSG1/JCSSGR", "USER" )
setParameter( "MATERIAL", "Concrete", "JCSSMC/JCSS15/BASCST", 55000000 )    # 71500000
setParameter( "MATERIAL", "Concrete", "JCSSMC/JCSS15/STDDEV", 6500000 )     # 8500000
setParameter( "MATERIAL", "Concrete", "JCSSMC/POISON", 0.15 )
setParameter( "MATERIAL", "Concrete", "JCSSMC/GF1", 134 )
setParameter( "MATERIAL", "Concrete", "JCSSMC/DENSIT", 2400 )

setParameter( "MATERIAL", "Concrete", "JCSSRF/RFMETH", "COVARI" )
setParameter( "MATERIAL", "Concrete", "JCSSRF/COVARI/NX", np.ceil(L/0.05) )
setParameter( "MATERIAL", "Concrete", "JCSSRF/COVARI/NY", np.ceil(h/0.05) )
setParameter( "MATERIAL", "Concrete", "JCSSRF/COVARI/NZ", 1 )

setParameter( "MATERIAL", "Concrete", "JCSSRF/RFMETH", "FFOURT" )
setParameter( "MATERIAL", "Concrete", "JCSSRF/FFOURT/MX", 4 )
setParameter( "MATERIAL", "Concrete", "JCSSRF/FFOURT/MY", 0 )

setParameter( "MATERIAL", "Concrete", "JCSSRF/RFMETH", "LAS" )
setParameter( "MATERIAL", "Concrete", "JCSSRF/LAS/NDIV", 3 )
setParameter( "MATERIAL", "Concrete", "JCSSRF/LAS/KX", 4 )
setParameter( MATERIAL, "Concrete", "JCSSRF/LAS/KY", 1 )

setParameter( "MATERIAL", "Concrete", "JCSSRF/THRESH", 0.5 )
```

```
addGeometry( "Beam", "SHEET", "MEMBRA", [] )
setParameter( "GEOMET", "Beam", "THICK", b )
setElementClassType( "SHAPE", [ "Concrete" ], "MEMBRA" )
assignMaterial( "Concrete", "SHAPE", [ "Concrete" ] )
assignGeometry( "Beam", "SHAPE", [ "Concrete" ] )

## Reinforcement steel Material
addMaterial( "ReinforcementSteel", "REINFO", "REBOND", [] )
setParameter( "MATERIAL", "ReinforcementSteel", "REBARS/ELASTI/YOUNG", 1.95e+11 )
# 2e+11
setParameter( "MATERIAL", "ReinforcementSteel", "REBARS/POISON/POISON", 0.3 )
setParameter( "MATERIAL", "ReinforcementSteel", "REBARS/MASS/DENSIT", 7850 )
setParameter( "MATERIAL", "ReinforcementSteel", "RESLIP/DSNY", 1e+14 )
setParameter( "MATERIAL", "ReinforcementSteel", "RESLIP/DSSX", 1e+11 )
setParameter( "MATERIAL", "ReinforcementSteel", "RESLIP/SHFTYP", "BONDS6" )
setParameter( "MATERIAL", "ReinforcementSteel", "RESLIP/BONDS6/SLPVAL", [ 20000000,
8000000, 1e-05, 0.001, 0.002, 0.008, 0.4 ] )

addGeometry( "Bottom_reinf", "RELINE", "REBAR", [] )
setParameter( "GEOMET", "Bottom_reinf", "REITYP", "REITRU" )
setParameter( "GEOMET", "Bottom_reinf", "REITRU/CROSSE", As )
setParameter( "GEOMET", "Bottom_reinf", "REITRU/PERIME", us )
addElementData( "Bottom_reinf" )
setShapeType( "REINFORCEMENTSHAPE", "Bottom_reinf" )
setReinforcementType( "REINFORCEMENTSHAPE", "Bottom_reinf", "TRUSS_BOND_SLIP" )
assignMaterial( "ReinforcementSteel", "REINFORCEMENTSHAPE", [ "Bottom_reinf" ] )
assignGeometry( "Bottom_reinf", "REINFORCEMENTSHAPE", [ "Bottom_reinf" ] )
assignElementData( "Bottom_reinf", "REINFORCEMENTSHAPE", [ "Bottom_reinf" ] )
setParameter( "DATA", "Bottom_reinf", "INTERF", "TRUSS" )

addGeometry( "Top_reinf", "RELINE", "REBAR", [] )
setParameter( "GEOMET", "Top_reinf", "REITYP", "REITRU" )
setParameter( "GEOMET", "Top_reinf", "REITRU/CROSSE", Asp )
setParameter( "GEOMET", "Top_reinf", "REITRU/PERIME", usp )
addElementData( "Top_reinf" )
setShapeType( "REINFORCEMENTSHAPE", "Top_reinf" )
setReinforcementType( "REINFORCEMENTSHAPE", "Top_reinf", "TRUSS_BOND_SLIP" )
assignMaterial( "ReinforcementSteel", "REINFORCEMENTSHAPE", [ "Top_reinf" ] )
assignGeometry( "Top_reinf", "REINFORCEMENTSHAPE", [ "Top_reinf" ] )
assignElementData( "Top_reinf", "REINFORCEMENTSHAPE", [ "Top_reinf" ] )
setParameter( "DATA", "Top_reinf", "INTERF", "TRUSS" )

addGeometry( "Stirrups", "RELINE", "REBAR", [] )
setParameter( "GEOMET", "Stirrups", "REITYP", "REITRU" )
setParameter( "GEOMET", "Stirrups", "REITRU/CROSSE", Ass )
setParameter( "GEOMET", "Stirrups", "REITRU/PERIME", uss )
addElementData( "Stirrups" )

for i in range(1,int(nstirrups)*2+1):
    setShapeType( "REINFORCEMENTSHAPE", "Stirrup_" + str(i) )
    setReinforcementType( "REINFORCEMENTSHAPE", "Stirrup_" + str(i), "TRUSS_BOND_SLIP"
)
    assignMaterial( "ReinforcementSteel", "REINFORCEMENTSHAPE", [ "Stirrup_" + str(i)
] )
    assignGeometry( "Stirrups", "REINFORCEMENTSHAPE", [ "Stirrup_" + str(i) ] )
    assignElementData( "Stirrups", "REINFORCEMENTSHAPE", [ "Stirrup_" + str(i) ] )

setParameter( "DATA", "Stirrups", "INTERF", "TRUSS" )

## Steel plates
addMaterial( "Steel", "MCSTEL", "ISOTRO", [] )
setParameter( "MATERIAL", "Steel", "LINEAR/ELASTI/YOUNG", 2e+11 )
setParameter( "MATERIAL", "Steel", "LINEAR/ELASTI/POISON", 0.3 )
setParameter( "MATERIAL", "Steel", "LINEAR/MASS/DENSIT", 7850 )
addGeometry( "Plate thickness", "SHEET", "MEMBRA", [] )
setParameter( "GEOMET", "Plate thickness", "THICK", b )
setElementClassType( "SHAPE", [ "Left_load_plate", "Left_support_plate",
"Right_load_plate", "Right_support_plate" ], "MEMBRA" )
assignMaterial( "Steel", "SHAPE", [ "Left_load_plate", "Left_support_plate",
"Right_load_plate", "Right_support_plate" ] )
assignGeometry( "Plate thickness", "SHAPE", [ "Left_load_plate", "Left_support_plate",
"Right_load_plate", "Right_support_plate" ] )

## Wooden sheets
addMaterial( "Wood", "MCSTEL", "ISOTRO", [] )
setParameter( "MATERIAL", "Wood", "LINEAR/ELASTI/YOUNG", 2e+10 )
setParameter( "MATERIAL", "Wood", "LINEAR/ELASTI/POISON", 0.3 )
setParameter( "MATERIAL", "Wood", "LINEAR/MASS/DENSIT", 900 )
setElementClassType( "SHAPE", [ "Left_wooden_sheet", "Right_wooden_sheet" ], "MEMBRA"
)
assignMaterial( "Wood", "SHAPE", [ "Left_wooden_sheet", "Right_wooden_sheet" ] )
assignGeometry( "Plate thickness", "SHAPE", [ "Left_wooden_sheet",
"Right_wooden_sheet" ] )

## Dummy beam
addMaterial( "Dummy Material", "MCSTEL", "ISOTRO", [] )
setParameter( "MATERIAL", "Dummy Material", "LINEAR/ELASTI/YOUNG", 1e+12 )
setParameter( "MATERIAL", "Dummy Material", "LINEAR/ELASTI/POISON", 0.2 )
setParameter( "MATERIAL", "Dummy Material", "LINEAR/MASS/DENSIT", 1 )
addGeometry( "Dummy Geometry", "LINE", "CLS2B2", [] )
setParameter( "GEOMET", "Dummy Geometry", "SHAPE/RECTAN", [ 0.5, 0.2 ] )
setElementClassType( "SHAPE", [ "Line 1" ], "CLS2B2" )
assignMaterial( "Dummy Material", "SHAPE", [ "Line 1" ] )
assignGeometry( "Dummy Geometry", "SHAPE", [ "Line 1" ] )

####################   Load Definitions  ###########################
addSet( "GEOMETRYLOADSET", "Displacement" )
createPointLoad( "Disp", "Displacement" )
setParameter( "GEOMETRYLOAD", "Disp", "LODTYP", "DEFORM" )
setParameter( "GEOMETRYLOAD", "Disp", "DEFORM/SUPP", "Support_1" )
setParameter( "GEOMETRYLOAD", "Disp", "DEFORM/TR/VALUE", -0.001 )
setParameter( "GEOMETRYLOAD", "Disp", "DEFORM/TR/DIRECT", 2 )
attach( "GEOMETRYLOAD", "Disp", "Line 1", [[ L/2, h+0.1, 0 ]] )

####################   Meshing  ###########################
elementSetString = ["Concrete", "Top_reinf", "Left_support_plate", "Bottom_reinf",
"Right_support_plate", "Right_load_plate", "Line 1", "Right_wooden_sheet",
"Left_wooden_sheet", "Left_load_plate"]
for i in range(1,int(nstirrups)*2+1):
    elementSetString.append('Stirrup_'+str(i))
setElementSize( elementSetString, 0.01, -1, True )
setMesherType( elementSetString, "HEXQUAD" )

generateMesh( [] )

####################   Analysis  ###########################
n_steps = 60
addAnalysis( "Analysis1" )
addAnalysisCommand( "Analysis1", "NONLIN", "Structural nonlinear" )

setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"EXECUT(1)/LOAD/STEPS/EXPLIC/SIZES", "0.1(" + str(n_steps) + ")" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"EXECUT(1)/ITERAT/MAXITE", 350 )
addAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"EXECUT(1)/ITERAT/CONVER/ENERGY" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"EXECUT(1)/ITERAT/CONVER/ENERGY", True )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"EXECUT(1)/ITERAT/CONVER/DISPLA", True )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"EXECUT(1)/ITERAT/CONVER/FORCE", True )

dispNode = findNearestNodes([[L/2,h-0.01,0]])
# setAnalysisCommandDetail( "Analysis1", "Structural nonlinear", "OUTPUT(1)/FILE",
name + "_MaxDisp" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear", "OUTPUT(1)/DEVICE",
"TABULA" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(1)/LAYOUT/LINPAG", 1000000 )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(1)/LAYOUT/COLLIN", 1000000 )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(1)/SELECT/MODSEL", "USER" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(1)/SELECT/NODES(1)/RNGNRS", dispNode )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(1)/SELECT/ELEMEN(1)/RNGNRS", "NONE"  )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(1)/SELECT/REINFO(1)/RNGNRS", "NONE" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear", "OUTPUT(1)/SELTYP",
"USER" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(1)/USER/DISPLA(1)/TOTAL/TRANSL/GLOBAL/COMP", [ "Y" ] )
```

```
renameAnalysisCommandDetail( "Analysis1", "Structural nonlinear", "OUTPUT(1)",
"MaxDisp" )


nodeList = []
for i in range(0,L*200+1):
    nodeList.append([i/200,0.028,0])

NodeRange = findNearestNodes(nodeList)

# setAnalysisCommandDetail( "Analysis1", "Structural nonlinear", "OUTPUT(2)/FILE",
name + "_dispXX" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear", "OUTPUT(2)/DEVICE",
"TABULA" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(2)/LAYOUT/LINPAG", 1000000 )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(2)/LAYOUT/COLLIN", 1000000 )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(2)/SELECT/MODSEL", "USER" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(2)/SELECT/NODES(1)/RNGNRS", NodeRange )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(2)/SELECT/ELEMEN(1)/RNGNRS", "NONE")
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(2)/SELECT/REINFO(1)/RNGNRS", "NONE" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear", "OUTPUT(2)/SELTYP",
"USER" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(2)/USER/DISPLA(1)/TOTAL/TRANSL/GLOBAL/COMP", [ "X" ] )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(2)/USER/DISPLA(1)/TOTAL/TRANSL/GLOBAL/COORDI", True )
renameAnalysisCommandDetail( "Analysis1", "Structural nonlinear", "OUTPUT(2)",
"dispXX" )


# setAnalysisCommandDetail( "Analysis1", "Structural nonlinear", "OUTPUT(3)/FILE",
name + "_Bot_reinf" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear", "OUTPUT(3)/DEVICE",
"TABULA" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(3)/LAYOUT/LINPAG", 1000000 )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(3)/LAYOUT/COLLIN", 1000000 )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(3)/SELECT/MODSEL", "USER" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(3)/SELECT/NODES(1)/RNGNRS", "NONE" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(3)/SELECT/ELEMEN(1)/RNGNRS", "NONE" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(3)/SELECT/REINFO(1)/RNGNRS", "\"Bottom_reinf\"" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear", "OUTPUT(3)/SELTYP",
"USER" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(3)/USER/STRAIN(1)/TOTAL/GREEN/GLOBAL/COMP", [ "XX" ] )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(3)/USER/STRAIN(1)/TOTAL/GREEN/GLOBAL/COORDI", True )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(3)/USER/STRAIN(1)/TOTAL/GREEN/GLOBAL/LOCATI", "CENTER" )
renameAnalysisCommandDetail( "Analysis1", "Structural nonlinear", "OUTPUT(3)",
"Bot_reinf" )

reacNode=  findNearestNodes([[0.05*L,-0.03,0]])
# setAnalysisCommandDetail( "Analysis1", "Structural nonlinear", "OUTPUT(4)/FILE",
name + "_Load" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear", "OUTPUT(4)/DEVICE",
"TABULA" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(4)/LAYOUT/LINPAG", 1000000 )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(4)/LAYOUT/COLLIN", 1000000 )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(4)/SELECT/MODSEL", "USER" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(4)/SELECT/NODES(1)/RNGNRS", reacNode )
```

```
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(4)/SELECT/ELEMEN(1)/RNGNRS", "NONE"  )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(4)/SELECT/REINFO(1)/RNGNRS", "NONE" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear", "OUTPUT(4)/SELTYP",
"USER" )
setAnalysisCommandDetail( "Analysis1", "Structural nonlinear",
"OUTPUT(4)/USER/FORCE(5)/REACTI/TRANSL/GLOBAL/COMP", [ "Y" ] )
renameAnalysisCommandDetail( "Analysis1", "Structural nonlinear", "OUTPUT(4)", "Load"
)



###################   Looping analysis using different material models
############################

RFMETH = ["COVARI", "FFOURT"]
DECMTH = ["CHOLES", "EIGEN", "FENTON"]
COVTYP = ["EXPONE", "SQTYPE"]
DISTYP = ["NORMAL", "LOGNOR"]
CORLEN = [0.05, 0.5, 5]

oldname = "Analysis1"
for j in RFMETH:
    if j == "COVARI":
        for k in DECMTH:
            for l in COVTYP:
                for m in DISTYP:
                    for n in CORLEN:

                        ####################   Material Definitions
###################
                        ## Excluding high correlation-length (CORLEN) for specific
material combinations
                        if k == "CHOLES" and l == "SQTYPE" and n != 0.05:
                            break
                        ## Concrete Material
                        setParameter( "MATERIAL", "Concrete", "JCSSRF/RFMETH", j )
                        setParameter( "MATERIAL", "Concrete", "JCSSRF/COVARI/DECMTH",
k )
                        setParameter( "MATERIAL", "Concrete", "JCSSRF/COVTYP", l )
                        setParameter( "MATERIAL", "Concrete", "JCSSRF/DISTYP", m )
                        setParameter( "MATERIAL", "Concrete", "JCSSRF/CORLEN", n )

                        ## Analysis
                        name = j + "_" + k + "_" + l + "_" + m + "_" + str(n)
                        renameAnalysis( oldname, name )
                        setAnalysisCommandDetail( name, "Structural nonlinear",
"OUTPUT(1)/FILE", name + "_MaxDisp" )
                        setAnalysisCommandDetail( name, "Structural nonlinear",
"OUTPUT(2)/FILE", name + "_dispXX" )
                        setAnalysisCommandDetail( name, "Structural nonlinear",
"OUTPUT(3)/FILE", name + "_Bot_reinf" )
                        setAnalysisCommandDetail( name, "Structural nonlinear",
"OUTPUT(4)/FILE", name + "_Load" )

                        runSolver( [ name ] )
                        oldname = name

    else:
        for l in COVTYP:
            for m in DISTYP:
                for n in CORLEN:

                    ####################   Material Definitions  ###################
                    ## Concrete Material
                    setParameter( "MATERIAL", "Concrete", "JCSSRF/RFMETH", j )
                    setParameter( "MATERIAL", "Concrete", "JCSSRF/COVTYP", l )
                    setParameter( "MATERIAL", "Concrete", "JCSSRF/DISTYP", m )
                    setParameter( "MATERIAL", "Concrete", "JCSSRF/CORLEN", n )

                    ## Analysis
                    name = j + "_" + l + "_" + m + "_" + str(n)
                    renameAnalysis( oldname, name )
                    setAnalysisCommandDetail( name, "Structural nonlinear",
"OUTPUT(1)/FILE", name + "_MaxDisp" )
```

```
                    setAnalysisCommandDetail( name, "Structural nonlinear",
"OUTPUT(2)/FILE", name + "_dispXX" )
                    setAnalysisCommandDetail( name, "Structural nonlinear",
"OUTPUT(3)/FILE", name + "_Bot_reinf" )
                    setAnalysisCommandDetail( name, "Structural nonlinear",
"OUTPUT(4)/FILE", name + "_Load" )

                    runSolver( [ name ] )
                    oldname = name
```

APPENDIX B

Arrangement of data in Matlab

```
clear all
close all

addpath('../')

L = 3;
H = 0.25;
B = 0.2;
D = 16/1000;

%%%%%%%%%%%% Assemble data from Diana %%%%%%%%%%%%
RFMETH = ["COVARI", "FFOURT"];
DECMTH = ["CHOLES", "EIGEN", "FENTON"];
COVTYP = ["EXPONE", "SQTYPE"];
DISTYP = ["NORMAL", "LOGNOR"];
CORLEN = [0.05, 0.5, 5];

i=0;
for j = 1:length(RFMETH)
    if RFMETH(j)== "COVARI"
        for k = 1:length(DECMTH)
            for l = 1:length(COVTYP)
                for m = 1:length(DISTYP)
                    for n = 1:length(CORLEN)
                        if DECMTH(k) == "CHOLES" && COVTYP(l) == "SQTYPE" && CORLEN(n)
~= 0.05
                            break
                        end
                        beamNumber = RFMETH(j) + "_" + DECMTH(k) + "_" + COVTYP(l) +
"_" + DISTYP(m) + "_" + string(CORLEN(n));
                        beamNumber = char(beamNumber);
                        beam_case_int = a_read_tables(beamNumber,L,H,B,D);
                        beamNumber = regexprep(beamNumber,'[.]','');
                        beam_full.(beamNumber)=beam_case_int;

                        %%% Extract data to Excel
                        [~,ind] = min(abs(beam_case_int.load-60/2));
                        max_defl = beam_case_int.max_deflection(ind);
                        nr_cracks = beam_case_int.number_of_cracks_btwload(ind);
                        crack_width = beam_case_int.crack_width_btwload(ind,:)*1e6;

                        i = i+1;
                        filename = 'a_model_combination.xlsx';
                        sheet = 1;
                        xlswrite(filename,{beamNumber},sheet,['A' int2str(7+i)])
                        xlswrite(filename,max_defl,sheet,['B' int2str(7+i)])
                        xlswrite(filename,nr_cracks,sheet,['C' int2str(7+i)])
                        xlswrite(filename,crack_width,sheet,['D' int2str(7+i)])

                    end
                end
            end
        end
    else
        for l = 1:length(COVTYP)
            for m = 1:length(DISTYP)
                for n = 1:length(CORLEN)
                    beamNumber = RFMETH(j) + "_" + COVTYP(l) + "_" + DISTYP(m) + "_" +
string(CORLEN(n));
                    beamNumber = char(beamNumber);
                    beam_case_int = a_read_tables(beamNumber,L,H,B,D);
                    beamNumber = regexprep(beamNumber,'[.]','');
                    beam_full.(beamNumber)=beam_case_int;

                    %%% Extract data to Excel
                    [~,ind] = min(abs(beam_case_int.load-60/2));
                    max_defl = beam_case_int.max_deflection(ind);
                    nr_cracks = beam_case_int.number_of_cracks_btwload(ind);
                    crack_width = beam_case_int.crack_width_btwload(ind,:)*1e6;

                    i = i+1;
                    filename = 'a_model_combination.xlsx';
                    xlswrite(filename,{beamNumber},sheet,['A' int2str(7+i)])
                    xlswrite(filename,max_defl,sheet,['B' int2str(7+i)])
                    xlswrite(filename,nr_cracks,sheet,['C' int2str(7+i)])
                    xlswrite(filename,crack_width,sheet,['D' int2str(7+i)])
                end
            end
        end
    end
end

save(['a_model_combination'],'beam_full');

%%%%%%%%%%%% Extract data from DIC to Excel %%%%%%%%%%%%
load('Z:\beam\DICbeam2.mat');
load('Z:\beam\DICbeam3.mat');
load('Z:\beam\DICbeam4.mat');

%%% Find max load from 1st loading cycle
[~,ind_2] = findpeaks(beam2.load,'MinPeakHeight',55);
ind_2 = ind_2(1);
[~,ind_3] = findpeaks(beam3.load,'MinPeakHeight',55);
ind_3 = ind_3(1);
[~,ind_4] = findpeaks(beam4.load,'MinPeakHeight',55);
ind_4 = ind_4(1);

max_defl_2 = beam2.def(ind_2);
max_defl_3 = beam3.def(ind_3);
max_defl_4 = beam4.def(ind_4);

%%% Find nr of cracks & crack width
% THRESHOLD (0.25) IS DERIVED FROM BOND SLIP TUNING ----------------------------
[~,crackpos_2] =
findpeaks(beam2.eps,beam2.pos,'MinPeakHeight',0.25,'MinPeakDistance',30);
ind_crackpos_2 = find(crackpos_2>900 & crackpos_2<1800);
crackpos_btwload_2 = crackpos_2(ind_crackpos_2);
nr_cracks_2 = length(crackpos_btwload_2);
crack_width_2 = beam2.cracks(ind_2,ind_crackpos_2)*1e3;

[~,crackpos_3] =
findpeaks(beam3.eps,beam3.pos,'MinPeakHeight',0.25,'MinPeakDistance',30);
ind_crackpos_3 = find(crackpos_3>900 & crackpos_3<1800);
crackpos_btwload_3 = crackpos_3(ind_crackpos_3);
nr_cracks_3 = length(crackpos_btwload_3);
crack_width_3 = beam3.cracks(ind_3,:)*1e3;  % Only 9 cracks from data while 10 from
findpeaks.

beam4.pos(1)=0; % Error in first indices
[~,crackpos_4] =
findpeaks(beam4.eps,beam4.pos,'MinPeakHeight',0.25,'MinPeakDistance',30);
ind_crackpos_4 = find(crackpos_4>900 & crackpos_4<1800);
crackpos_btwload_4 = crackpos_4(ind_crackpos_4);
nr_cracks_4 = length(crackpos_btwload_4);
crack_width_4 = beam4.cracks(ind_4,ind_crackpos_4)*1e3;

%%% Write to Excel
xlswrite(filename,max_defl_2,sheet,'B2')
xlswrite(filename,max_defl_3,sheet,'B3')
xlswrite(filename,max_defl_4,sheet,'B4')

xlswrite(filename,nr_cracks_2,sheet,'C2')
xlswrite(filename,nr_cracks_3,sheet,'C3')
xlswrite(filename,nr_cracks_4,sheet,'C4')

xlswrite(filename,crack_width_2,sheet,'D2')
xlswrite(filename,crack_width_3,sheet,'D3')
xlswrite(filename,crack_width_4,sheet,'D4')
```

```matlab
function [beam_case_int] = a_read_tables(beamNumber,L,H,B,D)


%%% Reading bottom strains
fid=fopen([beamNumber '_Bot_reinf.tb']);
j=0;
while ~feof(fid)

    x=sscanf(fgetl(fid),'%g');
    if isempty(x)
        i=0;
    else
        if i==0
            j=j+1;
        end
        i=i+1;
        bar1(j,i)=x(3,1);
        if j==1
            pos(1,i)=x(4,1)*1000;
        end
    end
end
[pos,ind_pos] = unique(pos);
bar1 = bar1(:,ind_pos);
[~,crackpos] = findpeaks(bar1(end-1,:),pos,'MinPeakDistance',20);

fid=fclose(fid);

%%% Reading Cracks
fid=fopen([beamNumber '_dispXX.tb']);
j=0;

line = 1;
while ~feof(fid)

    x=sscanf(fgetl(fid),'%g');
    if isempty(x)
        i=0;
    else
        if i==0
            j=j+1;
        end
        i=i+1;

        dxx(j,i)=x(2,1);
        if j == 1
            nodePos(i)=x(3,1)*1000;
        end

    end
end
[x,ind] = sort(nodePos);
dx = dxx(:,ind);

for w = 1:length(crackpos)
    [~,idx1] = min(abs(x - (crackpos(w)-20)));
    [~,idx2] = min(abs(x - (crackpos(w)+20)));
    cracks(:,w) = dx(:,idx2)- dx(:,idx1);
end

is_cracked = zeros(size(cracks));

for i = 1:length(crackpos)
    z = find(cracks(:,i)>=2.5e-5,1,'first');
    cracks(1:z-1,i) = 0;
    is_cracked(z:end,i) = 1;
end

fid=fclose(fid);


%%% read max disp
fid=fopen([beamNumber '_MaxDisp.tb']);
i=0;
while ~feof(fid)

    x=sscanf(fgetl(fid),'%g');
    if isempty(x)
    else
        i=i+1;
        max_deflection(i,1)=x(2,1)*(-1000);
    end
end
fid=fclose(fid);


%%% read load
fid=fopen([beamNumber '_Load.tb']);
i=0;
while ~feof(fid)
    x=sscanf(fgetl(fid),'%g');
    if isempty(x)
    else
        i=i+1;
        load(i,1)=x(2,1)/1000;
    end
end

fid=fclose(fid);


%%% assemble data
ind_ini = find(pos<0.05*L*1000,1,'last');
ind_fin = find(pos>0.95*L*1000,1,'first');

nsteps = size(bar1,1)-1;

time = linspace(1,nsteps,nsteps);
timex20 = linspace(1,nsteps,20*nsteps);

beam_case_int.pos = repmat(pos(ind_ini:ind_fin)-pos(ind_ini),nsteps*20,1);
beam_case_int.bar1 = (pchip(time,(bar1(1:end-1,ind_ini:ind_fin))',timex20))';
beam_case_int.max_deflection = (pchip(time,(max_deflection(1:end-1))',timex20))';
is_cracked_int = (pchip(time,(is_cracked(1:end-1,:))',timex20))';
is_cracked_int(is_cracked_int<1) = 0;
beam_case_int.is_cracked = is_cracked_int;
beam_case_int.number_of_cracks = sum(is_cracked_int,2);
beam_case_int.crack_positions = repmat(crackpos-pos(ind_ini),nsteps*20,1);
beam_case_int.crack_width = (pchip(time,(cracks(1:end-1,:))',timex20))';
beam_case_int.load = (pchip(time,(load(1:end-1))',timex20))';

%%% Find number of cracks between point loads
ind_btwload = find(1050<=crackpos & crackpos<=1950);
is_cracked_btwload_int = is_cracked_int(:,ind_btwload);
beam_case_int.is_cracked_btwload=is_cracked_btwload_int;
beam_case_int.number_of_cracks_btwload = sum(is_cracked_btwload_int,2);
beam_case_int.crack_width_btwload = beam_case_int.crack_width(:,ind_btwload);
```

APPENDIX C
Data postprocessing in Matlab

```
clc
clear all
close all

addpath('../')

L = 3;
H = 0.25;
B = 0.2;
D = 16/1000;

%%% Assemble data from selected model combination
beamNumber = ['beam_'];
[beam_case, beam_case_int] = read_tables(beamNumber,L,H,B,D);


%%% Find maximum allowable strain in concrete
% First try Diana
f_co = 71.5;            % Basic compressive strength
alpha = 0.8;
beta = 0.7;
f_c = alpha*f_co^0.96;
f_ct(1) = 0.3*f_c^(2/3)*1e6;
E_c(1) = 10500*f_c^(1/3)*(1/(1+beta))*1e6;
% Second try, Diana
f_c = f_co;
f_ct(2) = 0.3*f_c^(2/3)*1e6;
E_c(2) = 10500*f_c^(1/3)*(1/(1+beta))*1e6;

% Eurocode f_ctk95 = 1.3*f_ct
f_ctk95 = 1.3*f_ct;
% Tested strength converted in table from EN
f_ctk95(3) = 5.8e6;
E_c(3) = 40e9;
% Tested E modulus
E_c(4) = 33.3e9;

% Tensile strength in concrete
eps_ct95 = f_ct(2)/E_c(1);      % Theoretical value
% From test results.
threshold_05 = 20e-4;
threshold_36 = 8e-4;

%%%%%%%%%%%%%%%%  Crack detection based on data from Diana %%%%%%%%%%%%%%%
loads = [45 47.5 50 52.5 60];
for i = 1:length(loads)
    [~,ind_load(i)] = min(abs(beam_case_int.load-loads(i)/2));

    %%% Finding cracks calculating lateral displacement in Matlab
    nrCracks_disp(i) = beam_case_int.number_of_cracks_btwload(ind_load(i));


    %%% Finding cracks using strain profile from Diana
    ind_ElBtwl = find(900 <= beam_case_int.elpos(1,:) & beam_case_int.elpos(1,:) <=
1800);
    [~,crackpos.(['eps36_' int2str(i)])] =
findpeaks(beam_case_int.eps_36(ind_load(i),ind_ElBtwl),beam_case_int.elpos(1,ind_ElBtw
l),'MinPeakHeight',threshold_36,'MinPeakDistance',30);
    [Peaks_eps05,crackpos.(['eps05_' int2str(i)])] =
findpeaks(beam_case_int.eps_05(ind_load(i),ind_ElBtwl),beam_case_int.elpos(1,ind_ElBtw
l),'MinPeakHeight',threshold_05,'MinPeakDistance',30);

    nrCracks_eps36(i) = length(crackpos.(['eps36_' int2str(i)]));
    nrCracks_eps05(i) = length(crackpos.(['eps05_' int2str(i)]));


    %%% Plotting
    elpos = beam_case_int.elpos(1,:);
    eps36(i,:) = beam_case_int.eps_36(ind_load(i),:);
    eps05(i,:) = beam_case_int.eps_05(ind_load(i),:);

    figure(i)
    plot(elpos,eps36(i,:),'g',elpos,eps05(i,:),'c')
    xlim([300 2400])
    ylim([0 inf]);

    yline(threshold_36,':');
    yline(threshold_05,'-.');
    xl = xline(900,'--',{'Constant moment span'});
    xl.LabelHorizontalAlignment = 'center';
    xline(1800,'--');
    legend('Strain, depth 36 mm',' Strain, depth 5 mm','Crack limit, depth 36 mm',
'Crack limit, depth 5 mm')
    title(['Concrete strain profiles at two different depths (' char(string(loads(i)))
' kN)'])
    xlabel('Length between supports [mm]')
    ylabel('Strain [\epsilon]')

    ytext = 0.85*max(Peaks_eps05);
    text(2050,ytext,['Calculated cracks (former method) = '
int2str(nrCracks_disp(i))])
    text(2050,ytext-ytext/30,['Strain cracks, depth 36 mm = '
int2str(nrCracks_eps36(i))])
    text(2050,ytext-ytext/15,['Strain cracks, depth 5 mm = '
int2str(nrCracks_eps05(i))])
    set(gcf, 'WindowState', 'maximized');

end

%%% Check that max deflection is similar as Diana resutls
maxDefl_bondSlip = beam_case_int.max_deflection(ind_load(end))
maxDefl_modelComb = 6.2
nrCracks_modelComb = 9


%%
close all
%%%%%%%%%%%%%%% Strain profile in concrete: Diana vs DIC %%%%%%%%%%%%%%%
%%% Import DIC data
load('Z:\beam\DICbeam2.mat');
load('Z:\beam\DICbeam3.mat');
load('Z:\beam\DICbeam4.mat');
x2 = beam2.pos;
x3 = beam3.pos;
x4 = beam4.pos;
y2 = beam2.eps*1e-3;     % CONVERTED TO PROPER UNIT (MIKROMETER)
y3 = beam3.eps*1e-3;
y4 = beam4.eps*1e-3;
% Data error
x4(1) = 0;

%-----------------------------------------------------------------------
%%% Modify eps36 to match DIC (valleys = 0)
eps36Mod = eps36(end,:)-(5.5e-4);
threshold_36Mod = threshold_36-(5.5e-4);      % treshold = 2.5e-4 IS ALSO APPLIED IN
MODEL COMBINATION
%-----------------------------------------------------------------------


%%% Plot unfiltered strain profiles
figure(1);
plot(elpos,eps36(end,:),'g',x2,y2,x3,y3,x4,y4)
xl = xline(900,'--',{'Constant moment span'});
xl.LabelHorizontalAlignment = 'center';
xline(1800,'--');
xlim([800 1900]);
legend('Diana 36', 'DIC beam 2', 'DIC beam 3', 'DIC beam 4')
title(['Unfiltered strain profiles from Diana & DIC'])
xlabel('Length between supports [mm]')
ylabel('Strain [\epsilon]')
set(gcf, 'WindowState', 'maximized');

%%% Filter strain curves fom Dian
eps36ModFilt = eps36Mod;
eps36ModFilt(eps36ModFilt<threshold_36Mod) = 0;

y2Filt = y2;
y2Filt(y2Filt<threshold_36Mod) = 0;

y3Filt = y3;
y3Filt(y3Filt<threshold_36Mod) = 0;

y4Filt = y4;
y4Filt(y4Filt<threshold_36Mod) = 0;
```

```
%%% Compare nr of cracks and peak heights between Diana and DIC
ind_ElBtwl = find(900 <= elpos & elpos <= 1800);
[Peaks_eps36Mod,~] =
findpeaks(eps36ModFilt(ind_ElBtwl),elpos(ind_ElBtwl),'MinPeakDistance',30);

ind_btwlDIC2 = find(900 <= x2 & x2 <= 1800);
[Peaks_beam2,~] =
findpeaks(y2Filt(ind_btwlDIC2),x2(ind_btwlDIC2),'MinPeakDistance',30);

ind_btwlDIC3 = find(900 <= x3 & x3 <= 1800);
[Peaks_beam3,~] =
findpeaks(y3Filt(ind_btwlDIC3),x3(ind_btwlDIC3),'MinPeakDistance',30);

ind_btwlDIC4 = find(900 <= x4 & x4 <= 1800);
[Peaks_beam4,~] =
findpeaks(y4Filt(ind_btwlDIC4),x4(ind_btwlDIC4),'MinPeakDistance',30);

nrCracks_eps36Mod = length(Peaks_eps36Mod);
nrCracks_beam2 = length(Peaks_beam2);
nrCracks_beam3 = length(Peaks_beam3);
nrCracks_beam4 = length(Peaks_beam4);
meanPeak_eps36Mod = round(mean(Peaks_eps36Mod),4);
meanPeak_beam2 = round(mean(Peaks_beam2),4);
meanPeak_beam3 = round(mean(Peaks_beam3),4);
meanPeak_beam4 = round(mean(Peaks_beam4),4);

%%% Plot filtered strain profile
figure(2);
plot(elpos,eps36ModFilt,'g',x2,y2Filt,x3,y3Filt,x4,y4Filt)
xl = xline(900,'--',{'Constant moment span'});
xl.LabelHorizontalAlignment = 'center';
xline(1800,'--');
xlim([800 1900]);
% ylim([0 18e-3]);
legend('Diana, depth 36 mm', 'DIC beam 2', 'DIC beam 3', 'DIC beam 4')
title(['Filtered concrete strain profiles from Diana & DIC'])
xlabel('Length between supports [mm]')
ylabel('Strain [\epsilon]')

% ytext = max(max(Peaks_beam3),max(eps36ModFilt));
% text(1570,ytext,['Diana: mean strain = ' num2str(meanPeak_eps36Mod)]);
% text(1570,ytext-ytext/30,['Beam2: mean strain = ' num2str(meanPeak_beam2)]);
% text(1570,ytext-ytext/15,['Beam3: mean strain = ' num2str(meanPeak_beam3)]);
% text(1570,ytext-ytext/10,['Beam4: mean strain = ' num2str(meanPeak_beam4)]);
set(gcf, 'WindowState', 'maximized');


%%
close all
%%%%%%%%%%%%%%%%%%%%% Compare strain profile between Diana and DOFS at level of
reinforcement %%%%%%%%%%%%%%%%%%%%%%
%%% DIANA
nodepos = beam_case_int.pos(1,:);
epsReinf = beam_case_int.bar1(ind_load(end),:);

%%% DOFS
load('Z:\beam\beam2_filtered_data.mat');
xDof2 = x;
bar3_2 = bar3_ip*1e-6;
maxValue = max(bar3_2,[],'all');
[~,colInd] = find(bar3_2 == maxValue);
threshold = 0.8*maxValue;
[~,IndMaxLoad2] =
findpeaks(bar3_2(:,colInd),'MinPeakHeight',threshold,'MinPeakDistance',length(bar3_2(:
,1))/4);
IndMaxLoad2(2) = [];

load('Z:\beam\beam3_filtered_data.mat');
xDof3 = x;
bar3_3 = bar3_ip*1e-6;
maxValue = max(bar3_3,[],'all');
[~,colInd] = find(bar3_3 == maxValue);
threshold = 0.8*maxValue;
[~,IndMaxLoad3] =
findpeaks(bar3_3(:,colInd),'MinPeakHeight',threshold,'MinPeakDistance',length(bar3_3(:
,1))/4);
```

```
IndMaxLoad3(2) = [];

load('Z:\beam\beam4_filtered_data.mat');
xDof4 = x;
bar3_4 = bar3_ip*1e-6;
maxValue = max(bar3_4,[],'all');
[~,colInd] = find(bar3_4 == maxValue);
threshold = 0.8*maxValue;
[~,IndMaxLoad4] =
findpeaks(bar3_4(:,colInd),'MinPeakHeight',threshold,'MinPeakDistance',length(bar3_4(:
,1))/4);
IndMaxLoad4(2) = [];

%%% Plot
figure(1);
subplot(2,2,1);
plot(elpos,eps36Mod,'g',nodepos,epsReinf);
yline(threshold_36Mod,':');
xl = xline(900,'--',{'Constant moment span'});
xl.LabelHorizontalAlignment = 'center';
xline(1800,'--');
xlim([800 1900]);
ylim([-3e-4 2e-3]);
legend('Diana concrete', 'Diana reinforcement', 'Crack limit')
% title(['Strain profiles from Diana & DOFS at bottom reinforcent'])
xlabel('Length between supports [mm]')
ylabel('Strain [\epsilon]')
set(gcf, 'WindowState', 'maximized');

subplot(2,2,2);
plot(x2,y2,xDof2,bar3_2(IndMaxLoad2,:));
yline(threshold_36Mod,':');
xl = xline(900,'--',{'Constant moment span'});
xl.LabelHorizontalAlignment = 'center';
xline(1800,'--');
xlim([800 1900]);
ylim([-3e-4 2e-3]);
legend('DIC beam 2', 'DOFS beam 2', 'Crack limit')
% title(['Strain profiles from Diana & DOFS at bottom reinforcent'])
xlabel('Length between supports [mm]')
ylabel('Strain [\epsilon]')

subplot(2,2,3);
plot(x3,y3,xDof3,bar3_3(IndMaxLoad3,:));
yline(threshold_36Mod,':');
xl = xline(900,'--',{'Constant moment span'});
xl.LabelHorizontalAlignment = 'center';
xline(1800,'--');
xlim([800 1900]);
ylim([-3e-4 2e-3]);
legend('DIC beam 3', 'DOFS beam 3', 'Crack limit')
% title(['Strain profiles from Diana & DOFS at bottom reinforcent'])
xlabel('Length between supports [mm]')
ylabel('Strain [\epsilon]')

subplot(2,2,4);
plot(x4,y4,xDof4,bar3_4(IndMaxLoad4,:));
yline(threshold_36Mod,':');
xl = xline(900,'--',{'Constant moment span'});
xl.LabelHorizontalAlignment = 'center';
xline(1800,'--');
xlim([800 1900]);
ylim([-3e-4 2e-3]);
legend('DIC beam 4', 'DOFS beam 4', 'Crack limit')
% title(['Strain profiles from Diana & DOFS at bottom reinforcent'])
xlabel('Length between supports [mm]')
ylabel('Strain [\epsilon]')

figure(2)
plot(nodepos,epsReinf,'--
',xDof2,bar3_2(IndMaxLoad2,:),xDof3,bar3_3(IndMaxLoad3,:),xDof4,bar3_4(IndMaxLoad4,:))
;
xl = xline(900,'--',{'Constant moment span'});
xl.LabelHorizontalAlignment = 'center';
xline(1800,'--');
xlim([200 2500]);
% ylim([0 18e-3]);
```

```
legend('Diana', 'DOFS beam 2', 'DOFS beam 3', 'DOFS beam 4')
title(['Strain profiles in bottom reinforcement from Diana & DOFS'])
xlabel('Length between supports [mm]')
ylabel('Strain [\epsilon]')
set(gcf, 'WindowState', 'maximized');



% ---------------------------------------------------------------------------------------
----------------
%% Filter the strain profile in reinforcement to match DOFS
% % Filter command
% windowSize = [5 7 9 11 13];
% a = 1;
% figure(1)
% plot(nodepos,epsReinf,'LineWidth',3)
% hold on
%
% indEndPeak = find(nodepos>1560 & nodepos<1700);
% [~,posEndPeak] = findpeaks(epsReinf(indEndPeak),nodepos(indEndPeak));
% for l = 1:length(windowSize)
%     b = (1/windowSize(l))*ones(1,windowSize(l));
%     filt(l,:) = filter(b,a,epsReinf);
%     [~,xEndPeak] = findpeaks(filt(l,indEndPeak),nodepos(indEndPeak));
%     delta(l) = abs(xEndPeak-posEndPeak);
%     plot(nodepos-delta(l),filt(l,:))
%
% end
% xl = xline(900,'--',{'Constant moment span'});
% xl.LabelHorizontalAlignment = 'center';
% xline(1800,'--');
% xlim([800 1900]);
% % ylim([0 18e-3]);
% legend('epsReinf', '1', '2', '3', '4', '5')
% title(['Comparing and finding suitable DOFS strain curve'])
% xlabel('Length between supports [mm]')
% ylabel('Strain [\epsilon]')
% set(gcf, 'WindowState', 'maximized');
% hold off

%% Sgolayfilt
close all
% Run SGolay with different variables to find similar DOFS curve shapes
deg = [2 3 4 5 6];
fl =[11 15 21 25 31];

i = 1;
for j = 1:length(deg)
    for k = 1:length(fl)
        sgolay = sgolayfilt(epsReinf,deg(j),fl(k));
        sg.(char(["deg" + string(deg(j)) + "_fl" + string(fl(k))])) = sgolay;
        TitleName = char(["Degree = " + string(deg(j)) + ", Framelength = " +
string(fl(k))]);
        figure(i)
        i = i+1;
        plot(nodepos,sgolay)
        hold on
        plot(nodepos,epsReinf,'r--','LineWidth',0.2)

plot(xDof2,bar3_2(IndMaxLoad2,:),'k',xDof3,bar3_3(IndMaxLoad3,:),'k','LineWidth',1);
        xl = xline(900,'--',{'Constant moment span'});
        xl.LabelHorizontalAlignment = 'center';
        xline(1800,'--');
        xlim([800 1900]);
        legendName = char(["Deg = " + string(deg(j)) + ", Frl = " + string(fl(k))]);
        legend(legendName,'epsReinf','DOFS2','DOFS3')
        title(['Comparing and finding suitable DOFS strain curve'])
        xlabel('Length between supports [mm]')
        ylabel('Strain [\epsilon]')
        set(gcf, 'WindowState', 'maximized');
        hold off


    end
end
```

```
%% Analyse selected (SGolay) strain profiles and DOFS curves
close all
%%% Selected curves from SGolay filter that match DOFS:
% sg.deg2_fl11
% sg.deg3_fl11
% sg.deg6_fl25

% sg.deg4_fl15
% sg.deg5_fl15
% sg.deg6_fl21

% sg.deg4_fl11
% sg.deg5_fl11
% sg.deg6_fl15

%%% Selected curves can be grouped by shape in 3 categories
figure(1)
% Curve 1, 2 & 9
plot(nodepos,sg.deg2_fl11,'c',nodepos,sg.deg3_fl11,'c',nodepos,sg.deg6_fl25,'c')
hold on

% Curve 4, 6 & 8
plot(nodepos,sg.deg4_fl15,'r',nodepos,sg.deg5_fl15,'r',nodepos,sg.deg6_fl21,'r')

% Curve 3, 5 & 7
plot(nodepos,sg.deg4_fl11,'k',nodepos,sg.deg5_fl11,'k',nodepos,sg.deg6_fl15,'k')
plot(xDof2,bar3_2(IndMaxLoad2,:),'g--',xDof3,bar3_3(IndMaxLoad3,:),'g--
','LineWidth',0.5);
xl = xline(900,'--',{'Constant moment span'});
xl.LabelHorizontalAlignment = 'center';
xline(1800,'--');
xlim([800 1900]);
% ylim([0 18e-3]);
title(['Filtered strain profiles from Diana'])
legend('2nd degr, frlength 11', '3rd degr, frlength 11', '6th degr, frlength 25',...
    '4th degr, frlength 15', '5th degr, frlength 15', '6th degr, frlength 21', ...
    '4th degr, frlength 11', '5th degr, frlength 11', '6th degr, frlength 15',...
    'DOFS2','DOFS3')
xlabel('Length between supports [mm]')
ylabel('Strain [\epsilon]')
set(gcf, 'WindowState', 'maximized');
hold off

%%% Compare properties with DOFS
figure(2)
subplot(3,2,1)
findpeaks(sg.deg2_fl11,nodepos,'Annotate','extents')
xlim([200 2500]);
% ylim([7e-4 12e-4]);
% legend('1','2','3','4','5','DOFS beam 2', 'DOFS beam 3')
title('2nd degree, framelength 11')
xlabel('Length between supports [mm]')
ylabel('Strain [\epsilon]')
set(gcf, 'WindowState', 'maximized');


subplot(3,2,3)
findpeaks(sg.deg4_fl15,nodepos,'Annotate','extents')
xlim([200 2500]);
% ylim([7e-4 12e-4]);
% legend('1','2','3','4','5','DOFS beam 2', 'DOFS beam 3')
title('4th degree, framelength 15')
xlabel('Length between supports [mm]')
ylabel('Strain [\epsilon]')

subplot(3,2,5)
findpeaks(sg.deg4_fl11,nodepos,'Annotate','extents')
xlim([200 2500]);
% ylim([7e-4 12e-4]);
% legend('1','2','3','4','5','DOFS beam 2', 'DOFS beam 3')
title('4th degree, framelength 11')
xlabel('Length between supports [mm]')
ylabel('Strain [\epsilon]')

subplot(3,2,2)
findpeaks(bar3_2(IndMaxLoad2,:),xDof2,'Annotate','extents')
```

```
xlim([200 2500]);
% ylim([7e-4 12e-4]);
% legend('1','2','3','4','5','DOFS beam 2', 'DOFS beam 3')
title('DOFS from beam 2')
xlabel('Length between supports [mm]')
ylabel('Strain [\epsilon]')
set(gcf, 'WindowState', 'maximized');

subplot(3,2,4)
findpeaks(bar3_3(IndMaxLoad3,:),xDof3,'Annotate','extents')
xlim([200 2500]);
% ylim([7e-4 12e-4]);
% legend('1','2','3','4','5','DOFS beam 2', 'DOFS beam 3')
title('DOFS from beam 3')
xlabel('Length between supports [mm]')
ylabel('Strain [\epsilon]')
set(gcf, 'WindowState', 'maximized');


ind_nodeBtwl = find(900 <= beam_case_int.pos(1,:) & beam_case_int.pos(1,:) <= 1800);
[pks2_11,locs2_11,w2_11,p2_11] =
findpeaks(sg.deg2_fl11(ind_nodeBtwl),nodepos(ind_nodeBtwl));
[pks4_15,locs4_15,w4_15,p4_15] =
findpeaks(sg.deg4_fl15(ind_nodeBtwl),nodepos(ind_nodeBtwl));
[pks4_11,locs4_11,w4_11,p4_11] =
findpeaks(sg.deg4_fl11(ind_nodeBtwl),nodepos(ind_nodeBtwl));

pks2_11=pks2_11*10000;
pks4_11=pks4_11*10000;
p2_11=p2_11*10000;
p4_11=p4_11*10000;
p2_11_mean=mean(p2_11);
p4_11_mean=mean(p4_11);
w2_11_mean=mean(w2_11);
w4_11_mean=mean(w4_11);

ind_btwlDof2 = find(900 <= xDof2 & xDof2 <= 1800);
ind_btwlDof3 = find(900 <= xDof3 & xDof3 <= 1800);

[pks_Dof2,locs_Dof2,w_Dof2,p_Dof2] =
findpeaks(bar3_2(IndMaxLoad2,ind_btwlDof2),xDof2(ind_btwlDof2));
[pks_Dof3,locs_Dof3,w_Dof3,p_Dof3] =
findpeaks(bar3_3(IndMaxLoad3,ind_btwlDof3),xDof3(ind_btwlDof3));

p_Dof2=p_Dof2*10000;
p_Dof3=p_Dof3*10000;
p_Dof2_mean=mean(p_Dof2);
p_Dof3_mean=mean(p_Dof3);
w_Dof2_mean=mean(w_Dof2);
w_Dof3_mean=mean(w_Dof3);

%% Final plot
close all
subplot(3,1,1)
plot(nodepos,sg.deg2_fl11,xDof2,bar3_2(IndMaxLoad2,:),xDof3,bar3_3(IndMaxLoad3,:));
xl = xline(900,'--',{'Constant moment span'});
xl.LabelHorizontalAlignment = 'center';
xline(1800,'--');
xlim([800 1900]);
% ylim([0 18e-3]);
legend('Diana', 'DOFS beam 2', 'DOFS beam 3', 'DOFS beam 4')
title(['Strain profiles in bottom reinforcement from Diana & DOFS'])
xlabel('Length between supports [mm]')
ylabel('Strain [\epsilon]')
set(gcf, 'WindowState', 'maximized');

subplot(3,1,2)
plot(nodepos,sg.deg4_fl15,xDof2,bar3_2(IndMaxLoad2,:),xDof3,bar3_3(IndMaxLoad3,:));
xl = xline(900,'--',{'Constant moment span'});
xl.LabelHorizontalAlignment = 'center';
xline(1800,'--');
xlim([800 1900]);
% ylim([0 18e-3]);
legend('Diana', 'DOFS beam 2', 'DOFS beam 3', 'DOFS beam 4')
title(['Strain profiles in bottom reinforcement from Diana & DOFS'])
xlabel('Length between supports [mm]')
ylabel('Strain [\epsilon]')
```

```
set(gcf, 'WindowState', 'maximized');

subplot(3,1,3)
plot(nodepos,sg.deg4_fl11,xDof2,bar3_2(IndMaxLoad2,:),xDof3,bar3_3(IndMaxLoad3,:));
xl = xline(900,'--',{'Constant moment span'});
xl.LabelHorizontalAlignment = 'center';
xline(1800,'--');
xlim([800 1900]);
% ylim([0 18e-3]);
legend('Diana', 'DOFS beam 2', 'DOFS beam 3', 'DOFS beam 4')
title(['Strain profiles in bottom reinforcement from Diana & DOFS'])
xlabel('Length between supports [mm]')
ylabel('Strain [\epsilon]')
set(gcf, 'WindowState', 'maximized');
```

```
function [beam_case, beam_case_int] = read_tables(beamNumber,L,H,B,D)


%%% Reading concrete strain at reinforcement bar1
fid=fopen([beamNumber  '_epsXX_reinf.tb']);
j=0;
while ~feof(fid)

    x=sscanf(fgetl(fid),'%g');
    if isempty(x)
        i=0;
    else
        if i==0
            j=j+1;
        end
        i=i+1;
        eps_36(j,i)=x(2,1);
        if j==1
            elpos(1,i)=x(3,1)*1000;
        end
    end
end
[elpos,el_ind_pos] = unique(elpos);
eps_36 = eps_36(:,el_ind_pos);

fid=fclose(fid);


%%% Reading concrete strain at bottom edge of beam
fid=fopen([beamNumber  '_epsXX_bot.tb']);
j=0;
while ~feof(fid)

    x=sscanf(fgetl(fid),'%g');
    if isempty(x)
        i=0;
    else
        if i==0
            j=j+1;
        end
        i=i+1;
        eps_05(j,i)=x(2,1);
        if j==1
            el_pos_05(1,i)=x(3,1)*1000;
        end
    end
end
[el_pos_05,el_ind_pos_05] = unique(el_pos_05);
eps_05 = eps_05(:,el_ind_pos_05);

fid=fclose(fid);


%%% Reading bottom strains
fid=fopen([beamNumber  '_Bot_reinf.tb']);
j=0;
while ~feof(fid)

    x=sscanf(fgetl(fid),'%g');
    if isempty(x)
        i=0;
    else
        if i==0
            j=j+1;
        end
        i=i+1;
        bar1(j,i)=x(3,1);
        if j==1
            pos(1,i)=x(4,1)*1000;
        end
    end
end
[pos,ind_pos] = unique(pos);
bar1 = bar1(:,ind_pos);
[~,crackpos] = findpeaks(bar1(end-1,:),pos,'MinPeakDistance',30);

fid=fclose(fid);
```

```
%%% Reading Cracks
fid=fopen([beamNumber  '_dispXX.tb']);
j=0;

line = 1;
while ~feof(fid)

    x=sscanf(fgetl(fid),'%g');
    if isempty(x)
        i=0;
    else
        if i==0
            j=j+1;
        end
        i=i+1;

        dxx(j,i)=x(2,1);
        if j == 1
            nodePos(i)=x(3,1)*1000;
        end

    end
end
[x,ind] = sort(nodePos);
dx = dxx(:,ind);

for w = 1:length(crackpos)
    [~,idx1] = min(abs(x - (crackpos(w)-20)));
    [~,idx2] = min(abs(x - (crackpos(w)+20)));
    cracks(:,w) = dx(:,idx2)- dx(:,idx1);
end

is_cracked = zeros(size(cracks));

for i = 1:length(crackpos)
    z = find(cracks(:,i)>=2.5e-5,1,'first'); % f/E= 4/32 twice as mutch as crack
strain
    cracks(1:z-1,i) = 0;
    is_cracked(z:end,i) = 1;
end

fid=fclose(fid);


%%% read max disp
fid=fopen([beamNumber  '_MaxDisp.tb']);
i=0;
while ~feof(fid)

    x=sscanf(fgetl(fid),'%g');
    if isempty(x)
    else
        i=i+1;
        max_deflection(i,1)=x(2,1)*(-1000);
    end
end
fid=fclose(fid);

%%% read displacement
fid=fopen([beamNumber  '_Disp.tb']);
j=0;
while ~feof(fid)

    x=sscanf(fgetl(fid),'%g');
    if isempty(x)
        i=0;
    else
        if i==0
            j=j+1;
        end
        i=i+1;
        deflection(j,i)=x(2,1)*(-1000);
        if j==1
            pos_defl(1,i)=x(3,1)*1000;
        end
    end
```

```
end

[pos_defl,ind_pos_defl] = unique(pos_defl);
deflection = deflection(:,ind_pos_defl);

fid=fclose(fid);

%%% read load
fid=fopen([beamNumber  '_Load.tb']);
i=0;
while ~feof(fid)
    x=sscanf(fgetl(fid),'%g');
    if isempty(x)
    else
        i=i+1;
        load(i,1)=x(2,1)/1000;
    end
end

fid=fclose(fid);


%%% assemble data
ind_ini = find(pos<0.05*L*1000,1,'last');
ind_fin = find(pos>0.95*L*1000,1,'first');
ind_el_ini = find(elpos<0.05*L*1000,1,'last');
ind_el_fin = find(elpos>0.95*L*1000,1,'first');

nsteps = size(bar1,1)-1;

time = linspace(1,nsteps,nsteps);
timex20 = linspace(1,nsteps,20*nsteps);

beam_case.pos = repmat(pos(ind_ini:ind_fin)-pos(ind_ini),nsteps,1);
beam_case.elpos = repmat(elpos(ind_el_ini:ind_el_fin)-elpos(ind_el_ini),nsteps,1);

beam_case.bar1 = bar1(1:end-1,ind_ini:ind_fin);
beam_case.eps_36 = eps_36(1:end-1,ind_el_ini:ind_el_fin);
beam_case.eps_05 = eps_05(1:end-1,ind_el_ini:ind_el_fin);

beam_case.max_deflection = max_deflection(1:end-1);
beam_case.number_of_cracks = sum(is_cracked(1:end-1,:),2);
beam_case.is_cracked = is_cracked(1:end-1,:);
beam_case.crack_positions = repmat(crackpos-pos(ind_ini),nsteps,1);
beam_case.crack_width = cracks(1:end-1,:);
beam_case.load = load;

beam_case_int.pos = repmat(pos(ind_ini:ind_fin)-pos(ind_ini),nsteps*20,1);
beam_case_int.elpos = repmat(elpos(ind_el_ini:ind_el_fin)-
elpos(ind_el_ini),nsteps*20,1);

beam_case_int.bar1 = (pchip(time,(bar1(1:end-1,ind_ini:ind_fin))',timex20))';
beam_case_int.eps_36 = (pchip(time,(eps_36(1:end-
1,ind_el_ini:ind_el_fin))',timex20))';
beam_case_int.eps_05 = (pchip(time,(eps_05(1:end-
1,ind_el_ini:ind_el_fin))',timex20))';

beam_case_int.max_deflection = (pchip(time,(max_deflection(1:end-1))',timex20))';
is_cracked_int = (pchip(time,(is_cracked(1:end-1,:))',timex20))';
is_cracked_int(is_cracked_int<1) = 0;
beam_case_int.is_cracked = is_cracked_int;
beam_case_int.number_of_cracks = sum(is_cracked_int,2);
beam_case_int.crack_positions = repmat(crackpos-pos(ind_ini),nsteps*20,1);
beam_case_int.crack_width = (pchip(time,(cracks(1:end-1,:))',timex20))';
beam_case_int.pos_defl = repmat(pos_defl(ind_ini:ind_fin)-
pos_defl(ind_ini),nsteps*20,1);
beam_case_int.deflection = (pchip(time,(deflection(1:end-
1,ind_ini:ind_fin))',timex20))';
beam_case_int.load = (pchip(time,(load(1:end-1))',timex20))';

%%% Find number of cracks between point loads
ind_btwload = find(1050<=crackpos & crackpos<=1950);
is_cracked_btwload_int = is_cracked_int(:,ind_btwload);
beam_case_int.is_cracked_btwload=is_cracked_btwload_int;
beam_case_int.number_of_cracks_btwload = sum(is_cracked_btwload_int,2);
beam_case_int.crack_width_btwload = beam_case_int.crack_width(:,ind_btwload);
```