Gloria Wolkerstorfer, BSc

# Deep Uncertainty Quantification of Arterial Wall Simulations with Neural Networks

**MASTER'S THESIS**

to achieve the university degree of

Master of Science

Master's degree programme:
Physics

submitted to

**Graz University of Technology**

**Supervisor**

Univ.-Prof. Dipl.-Phys. Dr. Wolfgang von der Linden
Institute of Theoretical and Computational Physics
Co-Supervisor
Dipl.- Ing. Sascha Ranftl
Institute of Theoretical and Computational Physics

Graz, April 2021

## AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

19. 04. 2021 Wolkerstorfer Gloria

Date, Signature

## Abstract

The quest of solving high dimensional partial differential equations and propagating uncertainties of their intrinsic parameters within feasible computational time is a known problem in data analysis. This thesis investigates a surrogate model in order to estimate uncertainties of partial differential equations which stem from a system describing aortic wall properties to help modelling of Aortic Dissection.

The aim of this work was twofold. Firstly, Gaussian process sampling techniques were investigated and adapted to model a stochastic, heterogeneous and spatially correlated degradation of aortic tissue, due to the accumulation of glycosaminoglycans. Therefore, random field realizations adhering to the Beta distribution were generated over a non-equidistant grid, such that the degradation field obeys physical and physiological constraints.

These random fields then served as the input for an uni-axial tensile test of aortic tissue with a nearly-incompressible, hyperelastic constitutive law. The output was the left Cauchy-Green tensor and was provided by Rolf-Pissarczyk with finite element software application FEAP [1]. Investigations considered a neo-Hookean model as well as the recently proposed Rolf-Pissarczyk-Holzapfel model [2].

Secondly, a surrogate model was trained to map the input random fields, describing the degradation parameter, onto the output of the finite element calculation representing the results of the tensile test. The surrogate, a Convolutional Neural Network more specifically a Bayesian Deep Convolutional Encoder-Decoder as proposed in [3], was then used to propagate the uncertainties through the model since a brute force calculation would have been computationally infeasible because of complexity of the model.

The main pillars of this work include i) finding an efficient method for generating random fields at non-equidistant points, ii) accurately predict the left Cauchy-Green stresses in tensile direction as well as iii) the propagation of uncertainties through the model.

## Kurzfassung

Das Lösen von hochdimensionalen partiellen Differentialgleichungen und die damit verbundene Quantifizierung der Unsicherheiten der zugrundeliegenden Parameter ist ein bekanntes Problem in der Datenanalyse.

In dieser Arbeit wird ein Surrogat Modell aufgestellt, welches die Unsicherheiten eines Aortenwanddegradationsparameters untersucht, dessen partielle Differenzialgleichungen aus der Analyse von Aortenwand Abrissen stammen.

Die zwei Hauptziele der Arbeit waren Folgende: Erstens, Findung einer geeigneten Technik zur Simulierung von stochastischen Prozessen, welche den Degradationsparameter der Aortenwand als Folge von Ansammlung an Glykosaminoglykane als heterogen und räumlich korreliert darstellen. Dafür wurden Zufallsfelder (engl. = random fields) nach einer Beta-Verteilung auf einem nicht äquidistanten Gitter simuliert, welche als Input zu einer Finiten Elemente Rechnung dienten. Dem physiologischen Modell der Finiten Elemente Rechnung liegt ein Zugtest zugrunde, welcher ein hyperelatisches und beinahe inkompressibles Gewebe annimmt. Das Ergebnis dieses Zugtests liefert den linken Cauchy-Green Tensor in Zugrichtung, dessen Grundlagen auf einem kürzlich publizierten Modell von Rolf-Pissarczyk-Holzapfel basieren. Die Berechnung des Zugtests wurde ebenfalls vom Autor Rolf-Pissarczyk mit Hilfe der Software FEAP durchgeführt.

Das zweite Hauptziel, nach erfolgreicher Simulation der Zufallsfelder, war es ein Surrogat Modell durch eines Neuronalen Netzes zu trainieren, welches den Input, die Zufallsfelder, auf den Output des Zugtests lernt. Mit Hilfe eines „Bayesian Autoecoders“ konnte dann die Quantifizierung der Unischerheiten durchgeführt werden, welche durch „brute-force“ Analyse der Finiten Elemente Simulation auf Grund der Rechenzeit nicht möglich gewesen wäre.

# Contents

# Acknowledgements

This page is fully dedicated to all my friends and supporters and even if you, the reader, are not mentioned here, even by reading this, I want to say thank you for caring. I want to start this off with a quote, which reads: *'One man's voice is another man's noise'* and today I would like to interpret it as follows, namely I am extremely grateful to be surrounded by so, so many people with clear minds and strong voices.

First and foremost, the greatest thanks go to the best supervisors Wolfgang von der Linden and Sascha Ranftl, who I both really appreciate for their great knowledge, their mentoring, both academically and personally, and in addition I really acknowledge their guidance through my Bachelor and Master studies over the past almost three years. Moreover, I will always remember our monday morning meetings on a regular 'bayesis' with a smile, because it was always a pleasure to start the week like this. In addition, a strong thanks also goes to PhD candidates Malte Rolf-Pissarczyk and Antonio Pepe, not only for providing the evaluated data, but also for the fruitful discussions and their collaborative mindset.

Second, the biggest personal thanks goes to my family, especially my mom and dad, my sister Iris and my aunt Marianne who are the biggest supporters in my life, who always have an open ear for me and on whom I can always count. I cannot thank you enough for supporting me over the past 24 years and for enabling me to study physics.

Next comes, my chosen family, Hannah, Gerry, Robert, Felix + C. You are the best. In terms of friendship you are more than just friends. I don't want to miss any second we have spent together or any memory we created during the past 5 years and I want to express my deepest gratitude towards each and every one of you for being part of the greatest study group.

The greatest shoutout in terms of studying goes to Kevin-Alexander Blasl. Thank you Alex, you were my partner in crime, the best study buddy I could have imagined and I want you to know that I could not have achieved and learned far as much as I did without you! You are not only a true friend, but you are also my personal gauge boson, which interaction I highly valuate. If I was to assign you to a particle, you would be a photon, because you enlightened every study day and any lecture. Thank you for everything!

Last but not least, thank you mon cher Edouard. Thank you for being the biggest motivation and inspiration in my life, for bearing me everyday and for laughing with me even about the silliest (math) jokes.

# Glossary

## Probability Theory

**univariate process** - process dependent on one variable

**bivariate process** - process dependent on two variables

**multivariate process** - process dependent on multiple variables

**univariate vector process** - process dependent on one variable but with different variances within variable dimension

**multivariate vector process** - process dependent on multiple variables and with different variances within those variable dimensions

**lag-vector** - vector used for describing time/space steps in stochastic process

**auto-covariance** - covariance of stochastic process with itself

**cross-correlation function** - off diagonal elements in correlation function

**correlation coefficient** - normalized covariance which shows to magnitude of linear relation

**ACF** - Auto-Correlation function, the auto-correlation describes the correlation of a stochastic process with itself

**PSDF** - Power Spectral Density function, or simply PSD

**SRM** - Spectral Representation method, also called FFT-method or sum of cosines method

**Stochastic Partial Differential Equation method (SPDE)** - grid-independent method to generate Gaussian random fields by solving a PDE

**GRF** - Gaussian Random Field

**GMRF** - Gaussian Markov Random Field is a method to approximate Gaussian random fields by imposing Markov property to GRF

**KLE** - Karhunen–Loève expansion is a factorisation method to generate Gaussian random fields

**Ornstein-Uhlenbeck process** - Gaussian stochastic process with an exponential kernel in one-dimension

**homoscedastic** a sequence of random variables which have the same, finite variance. Complementary to a sequence being heteroscedastic.

**weakly homogeneous** in this context used to describe a random field with only the first two moments known. Moreover, the mean needs to be constant and the covariance only dependent on the lag-vector

**stationary process** or homogeneous process, does not vary its mean and variance over time/spatial dimension, i.e. joint probability is invariant to shifts

**isotropy** invariant under rotation, in this case, the isotropic covariance function only depends on the lag-vector, but not on the direction

**ergodic process** the joint probability distribution is completely determined by one (sufficiently large) realization

**Ergodicity** a (large enough) collection of samples will converge to the true distribution

## Machine Learning

**NN** - Neural Network

**CNN** - Convolutional Neural Network

**SVGD** - Stein Variational Gradient Descent

**Encoder** - compresses data e.g. images, via selection or extraction of features

**Decoder** - decompresses data in order to retrieve input dimensions, i.e. of input image

**KL** - Kullback Leibler divergence is a measure of difference between two probability distributions. It is used in Machine Learning to learn data distributions instead of e.g. regression functions.

**Variational Autoencoder** - or Bayesian Autoencoder, is a latent variable model which 'learns' a probability distribution, from which one can sample from. Thus, it is a generative model. The probability distribution is learned via Kullback Leibler divergence, which is different to other regression models, which only learn a (hypercomplex) function. VAEs are used for surrogate modelling i.e. in uncertainty quantification.

**verbose** - optional parameter, if true, network status is printed during training

**SVD** - Singular Value Decomposition - factorization method to decompose a $m \times n$ matrix into a $m \times n$ orthogonal, $n \times n$ diagonal and another $n \times m$ orthogonal part

**PCA** - Principal Component Analysis: tool to re-base a set according to descending variance, similar to SVD

**active subspace** - is called the span of particular directions in the input parameter space. Perturbation of the inputs along these active directions changes the output more, on average, than perturbing the inputs orthogonal to the active directions. By focusing on the model's response along active directions and ignoring the relatively inactive directions, we reduce the dimension for parameter studies, that are essential to engineering tasks such as design and uncertainty quantification [4]. Examples can be found online at [5].

**ADAM** - Adaptive Moments method inferred from physical idea, to optimize loss function by tracking previous descents and adapting the learning rate accordingly

**dropout** - regularization method used to prevent overfitting by ignoring some layers of the network during training

**batch normalization or Batch Norm** - regularization method used to prevent overfitting by normalizing each batch individually during training

**pooling layers** - compress information of data during training, by either averaging (average-pooling) over a certain window or taking a window's maximum value (max-pooling)

**fully connected vs. sparsely connected** - in a fully connected network, each neuron within a layer is connected to all neighbouring neurons of another layer; sparsely connected means the opposite, that not all neurons are connected within two layers.

**feature map/ activation map** - gives the output activation of a given filter, i.e. produces a high value at a given location, if the feature represented in the convolutional filter is present at that location of the input.

**latent space** - vector space where features to map onto lie

**universal approximation theorem** - theorem which says that a neural network is capable to learn any degree complex model for infinite width

**Cosine Annealing** - a cosine for the learning rate annealing function is used

**Warm Restarts** - the learning rate is restarted every now and then (e.g. re-raised back up)

**nugget effect** - represents short scale randomness or noise in a random and spatially correlated variable [6].

## Biomechanical Engineering

**FEM** - Finite Elements Method are grid based methods to evaluate PDEs at grid point locations

**GIP** - Gaussian Integration Points are called the grid points at which the FEM solver evaluates the input, in this work Gauss-Kronrod quadtrature is used.

**surrogate model** - substitute function which is cheaper to evaluate

**nH** - neo Hookean model is a nonlinear stress-strain model of hyperelastic materials undergoing deformations

**deformation gradient** - often denoted by $\mathbf{F}$, is the deformation gradient which defines the local deformation.

**constitutive model** - idealized model to approximate observed physical behaviour of an ideal material, i.e. behaviour under stress or strain

**hyperelastic material** - also referred to as Green-elastic material, postulates the existence of a Helmholtz free energy function $\Psi$, which is defined per unit reference volume rather than per unit mass. If it solely depends on the deformation gradient $\mathbf{F}$ it is called strain energy function or stored energy function [7].

**heterogeneous material** - depends on local position in medium.

**homogeneous material** - does not depend on local position in medium.

**isotropic material** - a material is said to be isotropic, if the values of the strain energy function $\Psi(\mathbf{F})$ and $\Psi(\mathbf{F}^*)$ are the same for all orthogonal tensors relative to the reference configuration. In other words, if translation or rotation of the system leads to the same strain-energy function [7].

**First Piola-Kirchhoff stress tensor** - often denoted as $\mathbf{P}$, is a second order tensor, and the derivative of the scalar valued homogeneous strain energy function with respect to the tensor variable $\mathbf{F}$.

**reference/initial frame** - before deformation

**current frame** - after deformation

**uniaxial extension test** - tensile test with the same extension along the tensile direction

**Rolf-Pissarczyk-Holzapfel model** - constitutive model of incompressible aortic tissue with directional elastic and collagen fibers to analyse a tissue degradation parameter, as proposed in [2].

**curse of dimensionality** - calculation gets computationally infeasible when moving to higher complexity models. Thus, surrogate modelling for i.e. uncertainty quantification becomes necessary.

**PAV** Principal Absolute Value - sum of the squared Cauchy-Stress-tensor elements for each location.

**experimental uncertainty** - uncertainties due to observation errors.

**epistemic uncertainty** - systematic uncertainties arising within the modelling process, e.g. due to lack of knowledge or limited access to data.

**aleatoric uncertainty** - statistical uncertainty, repetition of the same event might give slightly distributed results due to lack of perfectly precise measurements. Difference to epistemic uncertainty is the awareness of performing an imperfect measurement.

**algorithmic uncertainty** - uncertainty due to numerical errors and finite computer precision.

**parameter uncertainty** - quantities which 'true' values are not known but considered in computations. They arise due to e.g. non-representative sampling or too little data.

**structural uncertainty** - uncertainties due to approximating reality/natural laws by equations.

# 1 Introduction

Aortic Dissection (AD), see Fig. 1, is usually initiated by a small tear at the innermost layer of the aorta, which then gradually propagates within the aortic layers leading to a so-called false lumen. The presence of a false lumen changes the local hemodynamics in the aorta, and consequently, causes tissue remodeling (degradation) and thrombus formation and growth. The annual occurrence of AD is 3–6 cases per 100,000 population, but the mortality rate during the first 24 hours can be high, if it is left untreated. Surgical repair of the aorta and a placement of a synthetic graft are needed for ascending aortic dissection and for certain descending aortic dissections. Usually, endovascular stent grafts are used for certain patients, especially when dissection involves the descending thoracic aorta. One fifth of patients die before even reaching the hospital, and up to one third die of operative or perioperative complications [8].

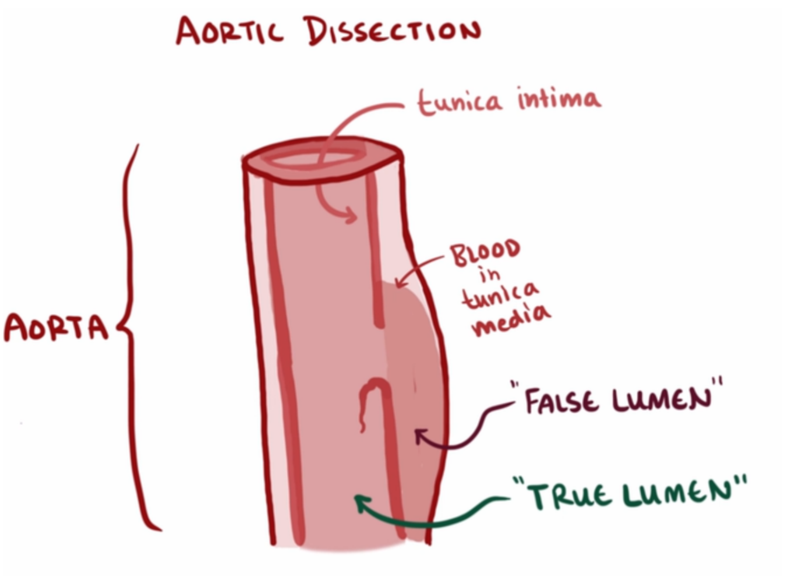


Figure 1: Sketch of Aortic Dissection. The Aorta is one of the main blood vessels in the body, consisting of three layers, the (tunica) intima, (tunica) media and (tunica) adventitia. During an Aortic Dissection a tear within the most inner layer occurs and leads to a 'False Lumen', beside the main blood vessel, indicated as 'True Lumen'. This image was taken from a video about AD in [8].

Moreover, diagnosis methods of AD such as MRI or CT are expensive. Therefore, simulations and models for non-invasive diagnosis methods are needed, which enable to recognize cheap, fast and yet, very accurately if the patient undergoes an Aortic Dissection. For the simulation of human tissue, computationally expensive simulations as e.g. proposed in [2], are needed in order to calculate specific properties, e.g. Stress, Strain, at certain locations. These calculations aim to predict where AD is most likely to occur and should describe aortic wall properties precisely within some range of uncertainty. In this case, the model used incorporates a degradation of the aortic wall, which can be characterized by a degradation parameter that lies between zero an one, ranging from healthy to degraded aortic walls respectively.

Numerical methods for in-silico analysis, like Finite Element Methods, work in principle, yet, they are unsuitable for performing Uncertainty Quantification. Hence, Neural Networks which are known to be capable of dealing with loads of data are used to learn a surrogate function, which can be then used to predict (values of) stress tensors at unseen samples. Those comparably 'cheap' to evaluate predictions can be used to perform uncertainty quantification, e.g. by utilizing Convolutional Neural Networks, which have shown promising results when applied to image-to-image regression including Variational Autoencoders [9, 10] and Generative Adversarial models [11, 12, 13].

In this work, the input is a degradation parameter describing the degradation of the aortic wall. This parameter is a random variable or rather Random field as is depends on the position, see Chapter 3. Gaussian random fields are widely used for modeling stochastic processes with applications in sampling groundwater resources [14], soil analysis [15], or, investigations of the cosmic microwave background, [16, 17, 18, 19].

The main advantage of Gaussian random fields is that they inherit Gaussian properties. Integration, differentiation as well as Fast Fourier transformation, yield again a Gaussian. In addition, one of the most important property is that any Gaussian can be fully described by its first and second moment, the mean and covariance. A summary of probability theory is outlined in Section 2. Nevertheless, Gaussian nature is not universal, which means that fitting a Gaussian process through any data may not yield meaningful results. Moreover, there is a strong correlation between the values at neighboring sites. The degradation parameter, by definition, can take on the values between 0 and 1. Due to lack of detailed knowledge, a uniform distribution is assumed, resulting in a correlated uniform random field. Therefore, sampling is not as simple as one might think at first glance. It is common to first sample Gaussian random fields and then map them onto the desired distribution, either through an analytic relation, or via iterative methods, see Section 3.4.

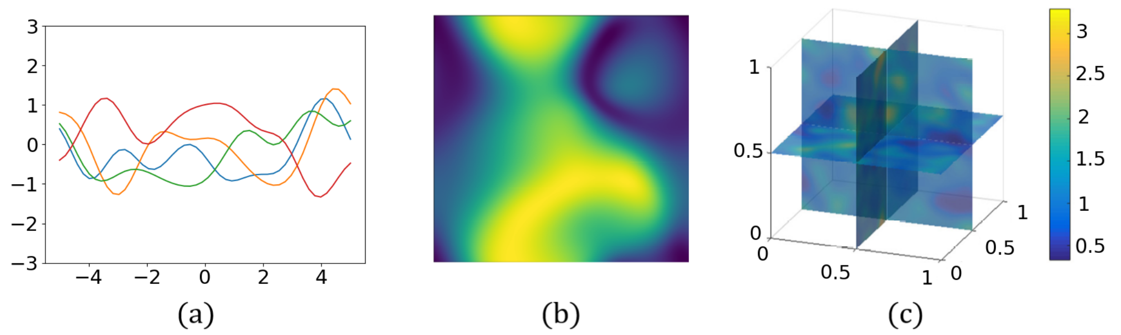


Figure 2: Curse of dimensionality. This figure shows samples drawn from a Gaussian random distribution in (a) one dimension (b) two dimensions and (c) three dimensions. Image (c) was taken from Noethinger 2018 [20]. All samples include spatial correlation to their neighbouring points, which requires the inversion of the covariance matrix. Factorisation methods like Cholesky decomposition become infeasible, since complexity grows with $N^d$ with $N$ being the number of points and $d$ the number of dimensions. Moreover, mapping Gaussian onto non-Gaussian random fields makes calculations even more demanding.

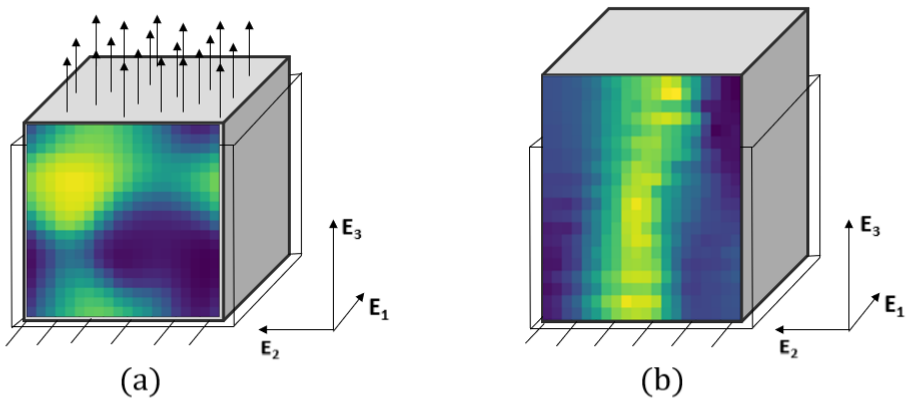


Figure 3: Uniaxial Tensile Test. The random field input is taken into the FEM solver and an uniaxial tensile test is performed on the heterogeneous tissue. In subfigure (a) the two dimensional random field is outlined, which is stacked as layers to a 3-dimensional input. Subfigure (b) shows the Cauchy-Stress-Tensor in tensile direction $E_3$.

In this thesis, Gaussian random fields, as shown in Fig. 2, were generated in Python via Spectral Methods, since it turned out to be the most efficient generating method for this application. More details can be found in Chapter 3.3.3. Other common sampling techniques of random fields, such as the Stochastic Partial Differential Equation approach, are discussed Chapter 3.3.4. In this work, the distribution of the degradation parameter was simulated as Gaussian and then mapped onto a Beta distribution, with special case of a Uniform distribution, because of the prior constraint that the degradation is bounded between $[0, 1]$. Other distributions to map analytically to would be e.g. the Gamma or Lognormal distribution, which are outlined in Section 3.5 and their generating code will be made available on `github/wolke26`.

After generating the 2D data, two identical layers of those fields were stacked behind each other, such that a 3-dimensional 'aortic-wall' was obtained, as shown in Fig. 3. This assumption of two identical layers is justified by the sliminess of the tissue. The high resolution images were down-sampled to a lower resolution, by taking only the values at the later Gaussian integration points of the Finite Element simulation. Moreover, the heterogeneous block was assumed as a hyperelastic material and the stress/strain tensors for the Rolf-Pissarczyk-Holzapfel model were computed with `FEAP`. The model consisted of collagen and elastic fibres, which were perpendicular and parallel to the blood flow. Two models used for the analysis were i) the neo-Hookean and ii) the Rolf-Pissarczyk-Holzapfel model which are explained in full detail in Section 4.

In order to perform uncertainty quantification of the computationally expensive model, a surrogate model was trained to map the input random fields onto the FEM solution output. This surrogate mapping was performed by a convolutional neural network, more specifically, a Bayesian-Encoder-Decoder, which benefits from including Kullback Leibler divergence (KL) in the loss function. Thus, the surrogate learns a probability distribution, which enables to propagate uncertainties through the network. More details on the convolutional neural networks, the architecture and further references can be found in Chapter 5.1.

Thus, after training the network once, one can readily use the surrogate model to predict more stress-tensor data, in order to perform uncertainty quantification, as outlined in Fig. 4.

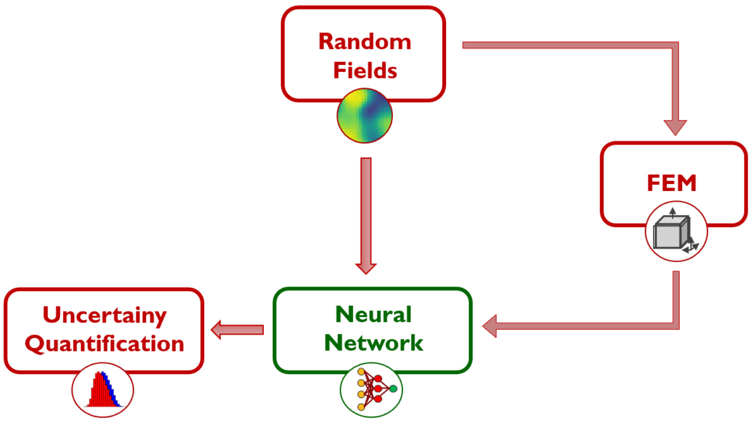


Figure 4: Uncertainty Quantification (UQ) of Finite Element Method (FEM) via surrogate model. Random fields are generated and used as an input to Finite Element simulations, which are too costly to perform UQ with. The mapping of input to output of the simulation is learned by a neural network, with the aim that its surrogate can be used for uncertainty quantification.

In this work, a total of 10.000 input-output sample sets were available and were split in the following way: 1) Training data (4200), 2) Test data (800), and 3) Validation data (5000). The allocation was taken such that enough data was left for comparing the surrogate model to unseen data. The network performance is outlined in Section 6.1. By setting up the surrogate model a worthwhile gain of CPU run time for future simulations can be obtained, with the downside of introducing further errors in the model. A more detailed discussion about considerations regarding uncertainties is given in Section 5.2.3.
For implementing the Bayesian-Encoder-Decoder, an architecture similar to Zhu and Zabaras 2018 [3] was used, with minor modifications as discussed in Chapter 5.2. The implementation was done in Python, the network itself was implemented with `Torch` [21]. A good introduction to the use of `PyTorch`is given by Stefan Otte's lecture, [22] with respective examples outlined on his Github page, see `github/sotte` [23].

# 2 Probability Theory

This Section gives an overview of Bayesian Probability Theory, which was relevant for this work and concisely summarizes [24]. For a broader approach the reader is referred to standard literature, such as [25] and a rather entertaining introduction is outlined in [26].

## 2.1 Fundamentals of Bayesian Probability Theory

In general, if an experiment is performed, all possible outcomes can be summarized in a certain event space, called $\Omega$. The probability of a certain event $A$ happening, can be written as $p(A)$, for all possible events in the event space. Moreover, the probability of $p(A) \geq 0$ is non negative and the total sum of all events holds $p(\Omega) = 1$. The probability of two mutually exclusive events $A$ and $B$ happening can then be given by their union

$$p(A \cup B) = p(A) + p(B). \tag{1}$$

In addition to the sum rule and normalisation, one finds the product rule as

$$p(A, B \mid \mathcal{I}) = p(A \mid B, \mathcal{I})\, p(B \mid \mathcal{I}) \tag{2}$$

$$p(A \mid \mathcal{I}) = p(A, B \mid \mathcal{I}) + p(A, \neg B \mid \mathcal{I}) \tag{3}$$

with $\neg B$ as the complement of event B. Those relations are called Kolmogorov axioms and lead to the most important rule of conditional probabilities, namely Bayes Theorem

$$p(A \mid B, \mathcal{I}) = \frac{p(B \mid A, \mathcal{I})\, p(A \mid \mathcal{I})}{p(B \mid \mathcal{I})}, \tag{4}$$

which was first introduced by Thomas Bayes, but only mentioned 2 years after his death in a letter conversation by Richard Price in 1793 [27]. The importance of this deceptively simple formula can hardly be overstated as it is the basis for countless innovations in the last decades. It is also pivotal for the understanding of the concepts as presented in this work [24].

## 2.2 Random Variables

There are two types of random variables, discrete and continuous ones. A discrete random variable is defined as a real valued function, which can at most take countably infinite numbers of values, whereas continuous variables can hold uncountably infinite numbers of values. The most common way to characterize a random variable is via its probability mass function, when it is discrete, and via its probability density function, if it is continuous. The probability density function (PDF) of a random variable $X$ is $p(x)$ is given by a non negative function, $p(x) \geq 0$, which gives the probability that a continuous variable $X$ lies in the interval $[a, b]$ such that

$$P(X \in (a.b)) = \int_a^b p(x) \ \mathrm{d}x \ . \tag{5}$$

In order to constitute a valid PDF, the function $p(x)$, also needs to be normalized, hence

$$\int_{-\infty}^{\infty} p(x)\, \mathrm{d}x = 1\ . \tag{6}$$

Moreover, a probability distribution can be introduced by using the CDF, or cumulative distribution function. The cumulative distribution function is defined by the probability that a random variable $X$ is smaller than or equal to $x$,

$$F(x) = P(X \leq x, |\ \mathcal{I}) = \begin{cases} \sum_{k\ \leq\ x}\ p(k) & \text{if X is discrete,} \\ \int_{-\infty}^{x} p(x')dx' & \text{if X is continuous} \end{cases} \tag{7}$$

i.e. it provides a probability that a random variable $X$ falls within the interval $(-\infty, x)$, [28, 25]. In addition, the following requirements have to be fulfilled:

- $0 \leq F(x) \leq 1, \forall\ x$
- $F_X$ is monotonic increasing for all $x < y$, $x \in X$ and $y \in X$ with $F(x) < F(y)$
- if $X$ is discrete $F(x)$ is a piecewise constant function of $x$
- if $X$ is continuous $F(x)$ is a continuous function of $x$

### 2.2.1 Transformation of Random Variables

Based on having a random variable $X$, another one, $Y$ can be obtained via a transformation. This transformation affects the infinitesimal volume $\mathrm{d}V_X \to \mathrm{d}V_Y$ which includes the relation

$$p_X(x)\ \mathrm{d}V_X = p_Y(y)\, \mathrm{d}V_Y\ , \tag{8}$$

which holds, because both cases contain the same infinitesimal probability mass. Therefore, one can write

$$p_Y(y) = p_X(x) \left| \frac{\partial X_i}{\partial Y_j} \right| , \tag{9}$$

whereby the change in volume is given by the Jacobian determinant

$$|J| = \left| \frac{\partial X_i}{\partial Y_j} \right| = \begin{bmatrix} \frac{\partial X_1}{\partial Y_1} & \frac{\partial X_1}{\partial Y_2} & \cdots & \frac{\partial X_1}{\partial Y_n} \\ \frac{\partial X_2}{\partial Y_1} & \frac{\partial X_2}{\partial Y_2} & \cdots & \frac{\partial X_2}{\partial Y_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial X_n}{\partial Y_1} & \frac{\partial X_n}{\partial Y_2} & \cdots & \frac{\partial X_n}{\partial Y_n} \end{bmatrix} . \tag{10}$$

### 2.2.2 Moments of Distributions

Probability distributions can be characterized in different ways. The most common way for discrete models is provided by the Probability mass function (PMF), which assigns a probability to each value of the discrete random variable, and the Probability density function (PDF), holds analogously for the continuous case. Another approach, which is sometimes more convenient to use, is based on moments of those functions. Moments are the means of powers of random variables. The first and second moment, mean $\mu$ and variance $\sigma^2$, are written as

$$\mathbb{E}[X] := \langle X \rangle := \int_{-\infty}^{\infty} x\, p(x)\; \mathrm{d}x \tag{11}$$

$$\mathbb{V}[X] := \langle \big(X - \langle X \rangle\big)^2 \rangle := \int (x-\mu)^2 p(x)\; \mathrm{d}x. \tag{12}$$

Moreover, the $n$-th moment is written as

$$\mathbb{E}[(X - \mathbb{E}[X]^n)] := \int_{-\infty}^{\infty} (x-\mu)^n\; f(x)\;\; \mathrm{d}x. \tag{13}$$

For the Gaussian distribution the first two moments are enough to fully describe the distribution, which makes them so powerful. For other distributions the knowledge of all n-th order densities have to be known, however there is often not sufficient data available. This property also influences the correlation function and thus, the spectral representation, which can then be generalized to the case of random fields, to give a caveat of the next Section. The first and second moments of a *homogeneous* random field are invariant with respect to a group operation (e.g. a linear shift) to their argument, defined on a class of commutative topological groups [29]. This holds for all homogeneous random fields, however for the proper description of most other distributions, the knowledge of all moments needs to be known. Thus, often a random field is said to be *weakly* homogeneous, to refer to that only the first two moment characteristics are known.

**Example: Gaussian PDF and CDF**

In case of a single real-valued random variable, the univariate Gaussian distribution is represented by the probability density function with mean $\mu$ and variance $\sigma^2$ as

$$p(x \mid \mu, \sigma) = \mathcal{N}(x \mid \mu, \sigma) = \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}}} \exp\left\{ -\frac{1}{2\sigma^2}(x-\mu)^2 \right\}. \tag{14}$$

The cumulative distribution of the centered Gaussian with $\sigma = 1$ can be obtained via the error function, reading

$$F(x) = \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right].$$

Fig. 5 shows the PDF (left) and CDF (right) of a Gaussian distribution.

## 2.3 Multivariate Random Variables

Since many applications require the consideration of multiple random variables simultaneously, the notions of PMF and PDF are extended to multiple random variables.

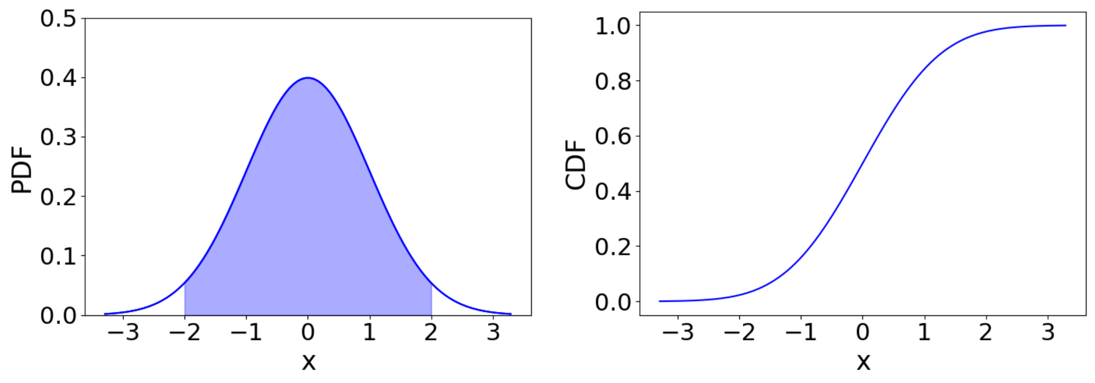


Figure 5: Standard univariate Gaussian distribution. Figure (a) shows the Probability density function with 95 % confidence region in light blue and figure (b) shows the Cumulative distribution function.

Therefore, the concepts of conditional and marginal probability distributions are introduced. Based on a collection of either discrete, or continuous, random variables the so-called joint probability mass function $P_X(x)$, or joint probability density $p_X(x)$ can be defined. In this Section the discussion is restricted to a bivariate process, thus, two continuous random variables $x_1 \in X_1$ and $x_2 \in X_2$, which are contained by the random vector $\vec{X} = [X_1, X_2]^T$. In statistics these variables could resemble independent distributions, such as people's weight, height, or else. The generalisation to multiple random variables is straightforward [24]. The joint probability density of $X_1$ and $X_2$ reads

$$p(x_1, x_2) = p(x_1 \mid x_2)\; p(x_2) = p(x_2 \mid x_1)\; p(x_1), \tag{15}$$

where $p(x_1 \mid x_2)$ and $p(x_2 \mid x_1)$ are called conditional distributions and $p(x_1)$, respectively $p(x_2)$, are the marginal distributions, which can be obtained by marginalization rule as

$$p(x_1) = \int_{-\infty}^{\infty} p(x_1, x_2)\;\, \mathrm{d}x_2. \tag{16}$$

If $X_1$ and $X_2$ are mutually dependent, the complete information of the system can be calculated using this equation. Another important quantity is the covariance function, also called kernel function, which describes the joint variability of two random variables

$$\operatorname{cov}(x_1, x_2) := k(x_1, x_2) = \mathbb{E}\big[(X_1 - \mathbb{E}[X_1])(X_2 - \mathbb{E}[X_2])\big] = \mathbb{E}[X_1 X_2] - \mathbb{E}[X_1]\mathbb{E}[X_2] \tag{17}$$

If the covariance has a positive sign, then increasing one variable will lead to an increase of the other one. In the opposite case, if the covariance has a negative sign, then a increase of one variable will correspond to a decrease of the other, see [24]. More information regarding the covariance function, including references, is outlined in Section 3.2.

If the correlation between two random variables is zero, they are called uncorrelated. One can formulate the normalized covariance, also called correlation coefficient, as

$$\rho(x_1, x_2) = \frac{\operatorname{cov}(x_1, x_2)}{\sqrt{\mathbb{V}[X_1]\mathbb{V}[X_2]}},$$
$$\text{which satisfies} \qquad -1 \leq \rho(x_1, x_2) \leq 1$$

For a multivariate random variable set in two dimensions, one defines the complete set as random vector $\vec{\boldsymbol{X}} = [\boldsymbol{X}_1, \boldsymbol{X}_2]^T$ with $\boldsymbol{X}_1 = [X_1^{(1)}, X_1^{(2)}]^T$ and $\boldsymbol{X}_2 = [X_2^{(1)}, X_2^{(2)}]^T$, whereby the lower index indicates the sample and the upper one the dimension within. Hence, the Gaussian PDF from Eq. (14) can be extended to the multivariate case with

$$\mathcal{N}(\boldsymbol{x}_1 \mid \boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11}) = \frac{1}{(2\pi)^{\frac{d}{2}}} \frac{1}{|\boldsymbol{\Sigma}_{11}|^{\frac{1}{2}}} \exp\Big\{ -\frac{1}{2}(\boldsymbol{x}_1 - \boldsymbol{\mu}_1)^T \boldsymbol{\Sigma}_{11}^{-1} (\boldsymbol{x}_1 - \boldsymbol{\mu}_1) \Big\}, \tag{18}$$

with $\boldsymbol{\mu}_1$ being the 2-dimensional mean vector, $\boldsymbol{\Sigma}_{11}$ the symmetric, positive definite $2 \times 2$ covariance matrix and $|\boldsymbol{\Sigma}_{11}|$ as its determinant. The covariance matrix can be calculated via the kernel function analogously to Eq. (17) with $\Delta\boldsymbol{X}_i = \boldsymbol{X}_i - \mathbb{E}[\boldsymbol{X}_i]$ yielding

$$\boldsymbol{\Sigma}_{11} = \mathbb{E}[\Delta\boldsymbol{X}_1 \Delta\boldsymbol{X}_1^T] = \mathbb{E}\left[\begin{bmatrix}\Delta X_1^1 \\ \Delta X_1^2\end{bmatrix} \begin{bmatrix}\Delta X_1^1 & \Delta X_1^2\end{bmatrix}\right] = \begin{bmatrix} k(x_1^1, x_1^1) & k(x_1^1, x_1^2) \\ k(x_1^1, x_1^2)^T & k(x_1^2, x_1^2)\end{bmatrix}. \tag{19}$$

For further details see [25]. Multivariate Gaussian distributions have the crucial property that both, marginal and conditional distributions of multivariate Gaussians yield again Gaussian distributions. This property is important when studying Gaussian processes, which will be introduced in the next Section. For the two multivariate random variables $\boldsymbol{x}_1 \in \boldsymbol{X}_1$ and $\boldsymbol{x}_2 \in \boldsymbol{X}_2$ their joint random vector is given by

$$p(\boldsymbol{x}_1, \boldsymbol{x}_2) \sim \mathcal{N}\left( \begin{bmatrix}\boldsymbol{x}_1 \\ \boldsymbol{x}_2\end{bmatrix} \middle| \begin{bmatrix}\boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2\end{bmatrix}, \begin{bmatrix}\boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{12}^T & \boldsymbol{\Sigma}_{22}\end{bmatrix} \right), \tag{20}$$

then the marginal distribution $p(\boldsymbol{x}_1)$ is a Gaussian distribution as well, and can be computed by integration over the volume

$$p(\boldsymbol{x}_1) = \int p(\boldsymbol{x}_1, \boldsymbol{x}_2)\, \mathrm{d}V_{\boldsymbol{x}_2} = \mathcal{N}\big(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11}\big). \tag{21}$$

The conditional distribution of $\mathbf{x}_1$ given $\mathbf{x}_2$ is also Gaussian

$$p(\boldsymbol{x}_1 \mid \boldsymbol{x}_2) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{X}_1|\boldsymbol{X}_2}, \boldsymbol{\Sigma}_{\boldsymbol{X}_1|\boldsymbol{X}_2}) \tag{22}$$

with

$$\boldsymbol{\mu}_{\boldsymbol{X}_1|\boldsymbol{X}_2} = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{11}\boldsymbol{\Sigma}_{22}^{-1}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2), \tag{23}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{X}_1|\boldsymbol{X}_2} = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}\ . \tag{24}$$

To illustrate these properties, Fig. 6 depicts an example of a bivariate Gaussian distribution with the associated marginal and conditional distributions. In addition, Fig. 6 also shows a positive covariance between $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$.

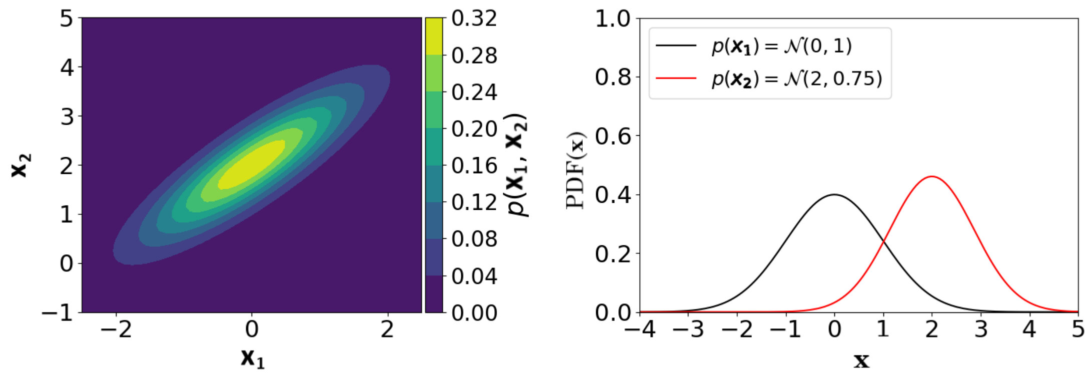


Figure 6: Multivariate Gaussian distributions. (a) Contours of a bivariate Gaussian distribution $p(\boldsymbol{x}_1, \boldsymbol{x}_2)$. (b) Marginal distribution of centered Gaussian $p(\boldsymbol{x}_1)$ with $\sigma^2 = 1$, shown in black and the marginal distribution $p(\boldsymbol{x}_2)$ shown in red.

## 3 Random Fields

This section outlines the main principles of random fields by starting off with the most simplest case, the Gaussian random process. In this chapter a new notation of indices is introduced. Main literature recommendations for an introduction to random fields include [30, 31, 32, 24, 25].

A *random field* or *stochastic process* is the extension of a multivariate random variable up to infinitely many dimensions, such that, given a probability space $(\Phi, \mathcal{F}, P)$ with $\mathcal{F}$ being a $\sigma$-algebra of subsets of $\Phi$ and $P$ is a countable additive, non-negative measure on $(\Phi, \mathcal{F})$ with total mass $P(\Omega) = 1$. Then, from a parameter set $X$, a random field is a finite and real valued measurable function $f(\boldsymbol{x}, \boldsymbol{\phi})$ with $\boldsymbol{\phi} \in \Phi$, for every fixed $\boldsymbol{x} \in X$. The synonyms *stochastic process* and *random field* are used, whereby some authors associate *stochastic processes* with *time*-dependent and *random fields* with *space*-dependent functions [24, 31].
Broadly speaking, the dimension of coordinates are usually within the range from one to four, but any $n > 0$ is possible. Random fields in two or more dimensions are encountered in a wider range of science, especially earth sciences, such as hydrology, agriculture, geology and climate modelling commonly use random fields, see [30, 33, 34, 16, 17]. In the following discussion, analogously to [24], the focus lies on univariate processes. According to [24], a stochastic process can be thought of as a function of two variables, an index parameter $\boldsymbol{x}$ and a probability parameter $\boldsymbol{\phi}$, which values range throughout the *event space* $\Phi$, also called *sample space*. For any fixed $\boldsymbol{\phi} \in \Phi$, the function $f(\boldsymbol{x}, \boldsymbol{\phi})$ is deterministic, and referred to as *sample path, sample function* or further, *realization* and for any fixed $\boldsymbol{x}$, $f(\boldsymbol{x}, \boldsymbol{\phi})$ becomes a random variable [24]. The collection of all possible realizations is called ensemble. The first order distribution called cumulative distribution function (CDF) of a stochastic process is defined as

$$F(f, \boldsymbol{x}) = P\left(f(\boldsymbol{x}) \leq f\right), \tag{25}$$

which directly leads to the first order density, or probability density function (PDF) reading

$$p(f, \boldsymbol{x}) = \frac{\partial F(f, \boldsymbol{x})}{\partial f}. \tag{26}$$

Here, the mean function reads

$$m(\boldsymbol{x}) = \mathbb{E}\left[f(\boldsymbol{x})\right] = \int\limits_{-\infty}^{\infty} f\ p(f, \boldsymbol{x})\ \mathrm{d}f$$

and similarly the variance function of the random process is defined as

$$\sigma^2(\boldsymbol{x}) = \mathbb{E}[f(\boldsymbol{x}) - m(\mathbf{x})]^2 = \mathbb{E}[f(\boldsymbol{x})^2] - \mathbb{E}[m(\boldsymbol{x})^2]. \tag{27}$$

Considering two random variables $f(\boldsymbol{x}_1)$ and $f(\boldsymbol{x}_2)$ with $\boldsymbol{x_1} \in X_1$ and $\boldsymbol{x_2} \in X_2$ the second order distribution can be written as

$$F(f_1, f_2, \boldsymbol{x_1}, \boldsymbol{x_2}) = P\Big(f(\boldsymbol{x_1}) \leq f_1\ ,\ f(\boldsymbol{x_2}) \leq f_2\Big) \tag{28}$$

with corresponding second order density

$$p(f_1, f_2, \boldsymbol{x}_1, \boldsymbol{x}_2) = \frac{\partial^2 F(f_1, f_2, \boldsymbol{x}_1, \boldsymbol{x}_2)}{\partial f_1 \partial f_2}. \tag{29}$$

From the second order density, Eq. (29), the (auto)-correlation function can be defined as the expectation of the joint moment with $i, j \in (1, 2)$

$$R(\boldsymbol{x_i}, \boldsymbol{x_j}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_i\ f_j\ p(f_i, f_j, \boldsymbol{x_i}, \boldsymbol{x_j})\ \mathrm{d}f_j \mathrm{d}f_j = \mathbb{E}[f_i\ f_j]. \tag{30}$$

Similar to before one defines the kernel or (auto)-covariance function as

$$k(\boldsymbol{x_i}, \boldsymbol{x_j}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} [f_i - m(\boldsymbol{x_i})]\ [f_j - m(\boldsymbol{x_j})]\ p(f_i, f_j, \boldsymbol{x_i}, \boldsymbol{x_j}) \mathrm{d}f_i \mathrm{d}f_j = \tag{31}$$

$$= \mathbb{E}\big[[f_i - m(\boldsymbol{x_i})][f_j - m(\boldsymbol{x_j})]\big]. \tag{32}$$

In case of e.g. a centered Gaussian (30) and (31) fall together. As before, the normalized covariance as correlation coefficient reads

$$\rho(\boldsymbol{x_i}, \boldsymbol{x_j}) = \frac{k(\boldsymbol{x_i}, \boldsymbol{x_j})}{\sqrt{k(\boldsymbol{x_i}, \boldsymbol{x_i})\ k(\boldsymbol{x_j}, \boldsymbol{x_j})}} = \frac{k(\boldsymbol{x_i}, \boldsymbol{x_j})}{\sqrt{\sigma^2(\boldsymbol{x_i})\sigma^2(\boldsymbol{x_j})}} \tag{33}$$

and shows the magnitude of strength of linear relation. In the univariate process this function becomes a scalar value, for multivariate processes it is a matrix, see Eq. (19).

## 3.1 Stationarity and Isotropy

In addition, if a stochastic process obeys both of the following conditions, it is said to be *weakly homogeneous*, or homogeneous in the weak sense. First, the mean function needs to be constant, i.e. not space/time-dependent,

$$\mathbb{E}[f(\boldsymbol{x}_1, \boldsymbol{x}_2)] = const < \infty, \forall\ \boldsymbol{x}_1 \in X_1 \text{ and } \boldsymbol{x}_2 \in X_2. \tag{34}$$

Secondly, the covariance function depends only on the difference between, and not on the absolute position of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, namely $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = k(\boldsymbol{x}_1 - \boldsymbol{x}_2) = k(\boldsymbol{\tau})$, with $\boldsymbol{\tau}$ called *lag-vector* or simply *lag*.

Another coining term synonymously used for homogeneous stochastic processes is **stationary**. This means, that a process does not vary in its stochastic dimension, i.e. time or space. In addition, if the regularity of the covariance function in a multi-dimensional processes is invariant under rotation of the coordinate system, the process is called **isotropic**, which is equivalent of saying that the covariance function only depends on the distance between two points but not on its direction, i.e.

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = k(||\boldsymbol{x}_1 - \boldsymbol{x}_2||) = k(||\boldsymbol{\tau}||). \tag{35}$$

A stochastic process is **ergodic** in the strict sense if the joint probability distribution is completely determined from one realization of the process alone. As with homogeneity, weaker criteria for ergodicity are often used. Roughly speaking, if a stochastic process is ergodic in the mean or correlation function, then the mean or the correlation function of the process can be computed from an average over the parameter space $X$. The formal requirements and conditions for ergodicity are omitted for the sake of brevity and the reader is referred to, e.g. [35, 36] and for a more thorough overview see [24] and references therein.

## 3.2 Covariance Functions and Correlation Length

After introducing the general concept of random fields, this subsection deals with the most general, and simplest case of stochastic processes, namely the Gaussian process. In general, a stochastic process is referred to as Gaussian if all joint probability distributions are Gaussian, i.e. a infinite collection of random variables, with any finite subset following a Gaussian distribution. Gaussian processes can be fully described by their mean $\boldsymbol{\mu}(\boldsymbol{x}_1, \boldsymbol{x}_2)$ and covariance $\boldsymbol{\Sigma}(\boldsymbol{x}_1, \boldsymbol{x}_2)$ reading

$$f(\boldsymbol{x}_1, \boldsymbol{x}_2) \sim \mathcal{GP}\big(\boldsymbol{\mu}(\boldsymbol{x}_1, \boldsymbol{x}_2), \boldsymbol{\Sigma}(\boldsymbol{x}_1, \boldsymbol{x}_2)\big) \ . \tag{36}$$

A detailed overview of kernels and their properties can be found in [37] and the video lecture [38]. In this work a zero-mean squared exponential covariance function was applied, which is a common choice [39]. Hence the covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{x}_1, \boldsymbol{x}_2)$ is a $n$ x $n$-matrix which holds the entries $\boldsymbol{\Sigma}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The kernel function of a centered Gaussian oughts to be of positive semidefinite form reading

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sigma_n^2 \exp\Big\{ - (\boldsymbol{x}_1 - \boldsymbol{x}_2)^T \boldsymbol{M} (\boldsymbol{x}_1 - \boldsymbol{x}_2) \Big\}. \tag{37}$$

with $\sigma_n^2$ as an amplitude magnification. Here $\boldsymbol{M}$ is a diagonal covariance matrix of a $d$-dimensional Gaussian with a **characteristic length** scale $\ell_m^2 > 0$ and $m = 1, 2, \dots, d$ yielding

$$\boldsymbol{M} = \begin{pmatrix} \ell_1^2 & & & \\ & \ell_2^2 & & \\ & & \ddots & \\ & & & \ell_\mathrm{m}^2 \end{pmatrix}. \tag{38}$$

The characteristic lengthscale describes the range of correlation between two data points. An outline of two different stochastic Gaussian processes with varying length scales is given in Fig. 7.

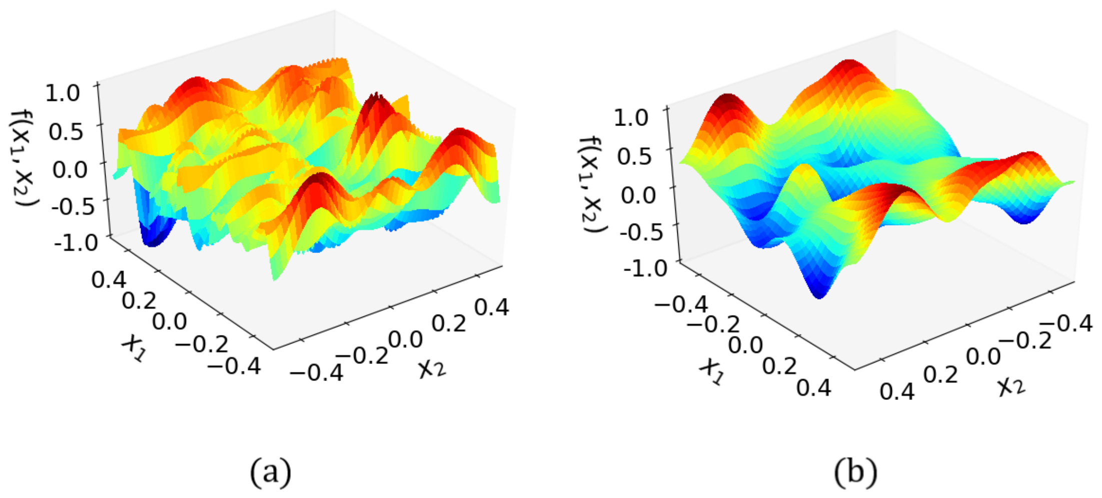


Figure 7: Realizations of two-dimensional Gaussian processes (a) Realization of anisotropic process (b) Realization of isotropic process, i.e. different characteristic length scales in $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ direction.

## 3.3 Random Process Sampling

After summarizing the probabilistic background and some useful definitions in Section 2, this subsection gives an overview of sampling methods used to generate stochastic process realizations. By today, there exists a zoo of different sampling methods for all kinds of stochastic processes. Even though most theoretical cornerstones have been proposed in the early 80s and 90s, the field of generating stochastic processes is still an active area of research. This is also because of growing computational power. In general, any of the existing sampling methods has their particular advantages and drawbacks. In essence, the choice usually boils down to a trade-off between speed and accuracy.

For this thesis, one of the simplest, fastest and computationally cheapest methods was used, namely the Spectral Representation method (SRM). SRM was first introduced in the early 90s by Shinozuka and Deodatis [40] and combines the advantages of Fast Fourier Transform with the disadvantage of generating samples only on evenly spaced grids. A more detailed description of the Spectral Representation method is given in Section 3.3.3.

In Section 3.3.4, a short summary of the Stochastic Partial Differential Method is outlined, which is a computationally more expensive method to generate random fields, but brings the benefit of generating samples on arbitrary grids. In addition, a numerical implementation of Gaussian processes, as well as non-Gaussian processes are considered in 3.3.3 and 3.5, respectively. Some code snippets are attached in the appendix. The complete code will be made available on `github/wolke26` [41].

### 3.3.1 Gaussian Process Sampling - A Primer

Random field generation schemes are essentially separated in two groups, depending whether they perform in space (direct methods) or wave-number (spectral methods). Direct methods generate realizations by filtering a white noise through the square root of the covariance matrix, whereby the simulation cost is essentially related to the computation of that square root of which the inverse needs to be calculated [42]. This method is called Cholesky factorization, see Section A.4 in [39], and comes with the drawback of scaling with $\mathcal{O}(N^3)$, where $N$ is the number of sampling points in d-dimensions. If however, the covariance matrix is sparse, or/and circulant, an optimized factorisation method, such as Turning bands, Toeplitz method [43], which can be exploited for faster algorithms, FFT - Moving Average [44], a method similar closely related to the standard spectral methods, which computes with $\mathcal{O}(N \log(N))$, see [45, 46]. In addition, direct methods use polynomial approximations of that square root, and obtain preconditioned iterative schemes that are interesting for sampling large dimensional random fields [47, 48]. In the paper of De Carvalho (2019) [42] the authors outline a great overview of the topic, including references to novel methods, in which e.g. it is proposed to dispatch the generation over smaller subdomains and to introduce statistical dependence between the random variables of the different subdomains [49]. However, it needs to be stated, that none of these models allows to decrease the complexity to $\mathcal{O}(N)$, which is necessary for favourable scaling over large clusters of processes [49].

On the other hand, one can simulate stochastic processes in the spectral domain, see

[50, 40, 51, 52] by using the byproduct of FFT to speed up the sampling process. There, the numerical cost is essentially that of computing the inverse Fourier transform, such that the complexity can be lowered to $\mathcal{O}(N\ \log(N))$. Yet, for every plus there is a minus, meaning that in this case spectral methods can quickly become memory expensive [24]. Furthermore, simulation of big domains often require cluster calculations, in which communication between processors can lead to efficiency loss. Thus, in [42] the authors introduce a scalable parallel scheme for sampling Gaussian random fields in order to overcome those limitations. They decompose the simulation domain into overlapping subdomains, each of which is assigned to a single processor.

### 3.3.2 Factorisation and the Curse of Dimensionality

The first method mentioned is the most common method to obtain a GP sample. It is a factorisation method, known as Cholesky decomposition. The Gaussian function over some points $\boldsymbol{x}$ reads

$$\mathcal{N}(f \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}} \frac{1}{|\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\boldsymbol{f}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{f}-\boldsymbol{\mu})\right\} \tag{39}$$

whereby the mean function $\boldsymbol{\mu} = \boldsymbol{\mu}(\boldsymbol{x}, \boldsymbol{\theta})$ and covariance function $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}(\boldsymbol{x}, \boldsymbol{\theta})$ are both dependent on some hyperparameters. For sampling by factorization the precision matrix $\mathbf{Q} = \boldsymbol{\Sigma}^{-1}$ needs to be calculated for every sample, e.g. by using Cholesky decomposition for which the square root $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^T$ is calculated. To obtain a Gaussian realization the result is multiplied on a set of standard normal variables $\boldsymbol{z} = \mathcal{N}(\mathbf{0}, \mathbf{I})$ and the mean function is added yielding

$$\boldsymbol{y}(\boldsymbol{x}) = \boldsymbol{\mu} + \mathbf{L}\boldsymbol{z}. \tag{40}$$

However, for most simulations in high dimensions Cholesky decomposition becomes computationally infeasible, because it scales with $N^3$, with $N$ being the number of sample points. This effect is also called *curse-of-dimensionality* and is outlined by Fig. 2.

Nevertheless, for low dimensional problems with relatively small covariance matrix, Cholesky decomposition works quite well.

### 3.3.3 Spectral Representation Method

This subchapter provides one of the most common sampling techniques of random fields being the spectral representation method (SRM), also known as spectral method and sum of cosines method. The SRM is one of the most common methods, because of its computational efficiency resulting from FFT. One of the main cornerstones to this method was introduced by Shinozuka and Deodatis in [50] and [40], who constructed it based on former work done by Rice (1954) [53]. Moreover, this method has been applied in many research areas, see Benowitz (2013, 2015) [54, 55], which also gives a conclusive introduction to this sampling method and thus, will be one of the main papers summarized in this Chapter.

However, despite its efficiency, SRM can only be applied for sampling on regular grids. Modifications for the Fast Fourier Transformation method on non-equidistant grids are possible, thus an ongoing field of research [56, 57].

First and foremost, SRM is based on two main concepts being **Ergodicity** and the **Wiener-Khintchine theorem**.

**Ergodicity** in the context of spectral representation method means that the collection of random samples, obtained by the SRM method, will converge to the true, in this case zero-mean Gaussian distribution.

In addition, **Wiener-Khintchine theorem** tells the relation between the auto correlation function (ACF) dependent on the lag-vector $\xi$ and the power spectral density (PSD), which reads

$$\mathcal{S}(\omega) = \frac{1}{(2\pi)} \int_{-\infty}^{\infty} \mathcal{R}(\xi) e^{-i\omega\xi} \, \mathrm{d}\omega \quad \text{and} \tag{41}$$

$$\mathcal{R}(\xi) = \int_{-\infty}^{\infty} \mathcal{S}(\omega) e^{i\omega\xi} \, \mathrm{d}\xi \tag{42}$$

meaning that, given either the PSD, or the ACF, the other can always be calculated by applying a Fourier transform. Thus, Wiener-Khintchine theorem is a special case of of the **cross-correlation theorem** [58]. For further references, see [59, 60, 61].

Let's define the random variable $\phi_n \in [0, 2\pi)$ as independent uniformly distributed random phase angles over the time domain t, such that the discretized frequency domain reads

$$\Delta\omega = \frac{\omega_u}{N} \tag{43}$$

$$\omega_n = n\Delta\omega \tag{44}$$

with $\omega_u$ being a cut-off frequency and $N$ being the number of realizations. Then, having defined the auto-correlation function and the spectral density function in Eq. (41) and Eq. (42) respectively, with $i = \sqrt{-1}$ being the imaginary unit, SRM enables to a sample stationary and Gaussian univariate processes $f(t)$ as

$$f(t) = \sqrt{2} \sum_{n=0}^{N-1} \left[ \sqrt{S(\omega_n)\Delta\omega} \cos\left(\omega_n t + \phi_n\right) \right]. \tag{45}$$

By help of Central Limit Theorem, $f(t)$ becomes asymptotically Gaussian as $N \to \infty$, however, in most cases $N$ actually does not need to be extremely large to reach sufficient accuracy [40, 24]. Moreover, Eq. (45) gives rise to another name of SRM, namely the *sum of cosines* method. In order to benefit from computational speed of FFT the sum of cosines can further be modified, see [50, 40], as

$$f(t) = \mathrm{Re}\left\{\sqrt{2}\sum_{n=0}^{N-1}\sqrt{S(\omega_n)\Delta\omega}e^{i\phi_n}e^{i\omega_n t}\right\}. \tag{46}$$

Here Re{.} denotes the real part. To compactify this equation one can introduce a vector **A** with components

$$A_n = 2\sqrt{S(\omega_n)\Delta\omega}e^{i\phi_n} \tag{47}$$

for which a stochastic sample dependent on $\phi_n$ can be generated as

$$f(t) = \mathrm{Re}\left\{\mathrm{FFT}(\mathbf{A})\right\}. \tag{48}$$

## Sampling of Gaussian Random Fields with SRM in two dimensions

In this Section the univariate two-dimensional case will be explained first, followed by numerical example analogous to Shinozuka (1996) [40]. In general, when sampling a stochastic field, either the auto-correlation function (ACF) or the power spectral density (PSD) needs to be known. In the context of engineering it is common to choose an ACF first. In astrophysics where one cannot really choose but has to fit the later, the PSD is considered. Here, an ACF is defined, such that one obtains a two-dimensional univariate (2D-1V) homogeneous and stationary, zero mean random field $f(x_1, x_2)$ with $x_i$ and $i = 1, 2$ being the dimensional axes. As mentioned in Chapter 3.1, the axes of the ACF in the stationary case only depend on their spatial separation $\xi_i$, *lag-vector*, and are time independent. In case of slowly changing systems, weak stationarity is often sufficient enough.
Then the (weakly) stationary ACF reads

$$\mathcal{R}(x_1, x_2) = \mathcal{E}[(f(x_1 + \xi_1, x_2 + \xi_2)f(x_1, x_2))] = \mathcal{R}(\xi_1, \xi_2). \tag{49}$$

The Wiener-Khintchine theorem for two dimensions states that the power spectral density and the auto-correlation function in Eq. (41) are related via Fourier transformation

$$\mathcal{S}(\omega_1, \omega_2) = \frac{1}{(2\pi)^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\mathcal{R}(\xi_1, \xi_2)e^{-i(\omega_1\xi_1+\omega_2\xi_2)}\mathrm{d}\omega_1\mathrm{d}\omega_2 \tag{50}$$

and

$$\mathcal{R}(\xi_1, \xi_2) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\mathcal{S}(\omega_1, \omega_2)e^{i(\omega_1\xi_1+\omega_2\xi_2)}\mathrm{d}\xi_1\mathrm{d}\xi_2 \tag{51}$$

with $\omega_i$ being the frequency vectors respectively. Moreover, the following properties are met by symmetry

$$\mathcal{S}(\omega_1, \omega_2) = \mathcal{S}(-\omega_1, -\omega_2)$$
$$\mathcal{R}(\xi_1, \xi_2) = \mathcal{R}(-\xi_1, -\xi_2)$$

and in case of quadrant symmetric ACF and PSD the following relations are valid too:

$$\mathcal{S}(\omega_1,\omega_2) = \mathcal{S}(\omega_1,-\omega_2) = \mathcal{S}(-\omega_1,\omega_2) = \mathcal{S}(-\omega_1,-\omega_2)$$
$$\mathcal{R}(\xi_1,\xi_2) = \mathcal{R}(\xi_1,-\xi_2) = \mathcal{R}(-\xi_1,\xi_2) = \mathcal{R}(-\xi_1,-\xi_2).$$

When sampling random fields, a distinction between the stochastic field and the simulated field must be made. The stochastic field is represented as an infinite sum of its elements, whereby the simulated field is regulated by its upper cut-off $\omega_u$. This cut-off is a value above which the PSD is assumed to be zero.
For the stochastic properties in the implementation this means that as $N_i \to \infty$ the ergodicity of the field is restored. In some cases like [55] values of $N_i = 16$ with $i = 1,2$ were sufficient enough. A detailed review and some proofs are given in [40]. Furthermore the following property can be proven, see [50]

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathcal{S}(\omega_1,\omega_2)\,\mathrm{d}\omega_1\,\mathrm{d}\omega_2 = \mathcal{R}(0,0) = \sigma^2. \tag{52}$$

For later clarity the variables $A_{n_1n_2}$ with $n_i = 0,1,2,\dots,N_i-1$ are introduced

$$A_{n_1,n_2} = \sqrt{2\,\mathcal{S}(\omega_{1_{n_1}},\omega_{2_{n_2}})\,\Delta\omega_1\,\Delta\omega_2} \tag{53}$$
$$\tilde{A}_{n_1,n_2} = \sqrt{2\,\mathcal{S}(\omega_{1_{n_1}},-\omega_{2_{n_2}})\,\Delta\omega_1\,\Delta\omega_2}. \tag{54}$$

The discretized frequency vectors $\omega_i$ of length $N_i - 1$ have to be chosen with sufficiently small, but finite steps $\Delta\omega_i$ reading

$$\Delta\omega_1 = \frac{\omega_{1u}}{N_1}, \quad \Delta\omega_2 = \frac{\omega_{2u}}{N_2}. \tag{55}$$

yielding the frequency vectors as

$$\omega_{1_{n_1}} = \Delta\omega_1 \cdot n_1 \quad \omega_{2_{n_2}} = \Delta\omega_2 \cdot n_2. \tag{56}$$

Here $\omega_{iu}$ is called cut-off wave number defining the limit above which the PSD is assumed to be of insignificant magnitude. A 2D-1V homogeneous random field can be calculated analogously to [40] p. 34, as

$$\begin{aligned} f(x_1,x_2) = \sqrt{2}\sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1}\Big[&A_{n_1,n_2}\cos\left(\omega_{1_{n_1}}x_1 + \omega_{2_{n_2}}x_2 + \Phi^{(1)}_{n_1n_2}\right)\\ &+\ \tilde{A}_{n_1,n_2}\cos\left(\omega_{1_{n_1}}x_1 + \omega_{2_{n_2}}x_2 + \Phi^{(2)}_{n_1n_2}\right)\Big]. \end{aligned} \tag{57}$$

The random phase angle $\Phi^{(i)}_{n_1n_2}$ takes values between $[0,2\pi)$ and in case of two dimensions $\Phi^{(1)}_{n_1n_2}$ and $\Phi^{(2)}_{n_1n_2}$ are two independent sets of random variables, which can be organized as a matrix with size $(N_1-1, N_2-1)$, consisting of independent elements. To obtain random field samples Eq. (57) can readily be used. However, for a large number of samples the sum of cosines slows down the calculation substantially. Shinozuka (1996) [40] and Brigham (1988) [62] suggested to modify the exponential terms in Eq. (57) to

enhance efficiency.
It should be said straight away that the use of FFT speeds up calculation to $\mathcal{O}(N\log(N))$ flops. Yet, this does not come for free. The two main restrictions to keep in mind are first, an equidistant grid is introduced, which may be disadvantageous, and second, that even if in two dimensions calculations work out nicely, the FFT approach can lead to memory issues as dimensions grow. Thus, Biehler (2016) [24] sticked to calculating the sums of cosines when sampling 3D processes instead.

The periodicity of $f(x_1, x_2)$ is given by the size of $L_{x_i}$ and in order to avoid aliasing one considers

$$\Delta x_1 \leq \frac{\pi}{\omega_{1u}} \quad \text{and} \quad \Delta x_2 \leq \frac{\pi}{\omega_{2u}} \tag{58}$$

with

$$M_1 \geq 2\ N_1 \quad \text{and} \quad M_2 \geq 2\ N_2. \tag{59}$$

The lengths of the dimensional axes are given by

$$\begin{aligned} L_{x_1} &= \frac{2\pi}{\Delta\omega_1} = \Delta x_1\ M_1 \\ L_{x_2} &= \frac{2\pi}{\Delta\omega_2} = \Delta x_2\ M_2 \end{aligned} \tag{60}$$

and in order to make full advantage of FFT it is recommended [40] to define $M_1$ and $M_2$ as powers of two.

In case where the FFT seems applicable Eq. (57) can be rewritten as

$$\begin{aligned} f(k_1\Delta x_1, k_2\Delta x_2) = Re\Bigg[\sum_{n_1=0}^{M_1-1}\sum_{n_2=0}^{M_2-1}\Big\{ & B_{n_1,n_2}\exp\Big[i(n_1\Delta\omega_{1_{\mathrm{n}_1}})(k_1x_1) + i(n_2\Delta\omega_{2_{\mathrm{n}_2}})(k_2x_2)\Big] \\ & + \tilde{B}_{n_1,n_2}\exp\Big[i(n_1\Delta\omega_{1_{\mathrm{n}_1}})(k_1x_1) - i(n_2\Delta\omega_{2_{\mathrm{n}_2}})(k_2x_2)\Big]\Big\}\Bigg] \\ k_1 = 0, 1, \ldots, M_1 - 1\,;\, k_2 = 0, 1, \ldots, M_2 - 1 & \end{aligned} \tag{61}$$

where $Re$ indicates the real part and $B_{n_1,n_2}$ and $\tilde{B}_{n_1,n_2}$ are defined using (53) and (54) as

$$\begin{aligned} B_{n_1,n_2} &= \sqrt{2}\ A_{n_1,n_2}\exp\left[i\Phi^{(1)}_{n_1n_2}\right] \\ n_1 &= 0, 1, \ldots M_1 - 1 \quad n_2 = 0, 1, \ldots M_2 - 1 \end{aligned} \tag{62}$$

$$\begin{aligned} \tilde{B}_{n_1,n_2} &= \sqrt{2}\ \tilde{A}_{n_1,n_2}\exp\left[i\Phi^{(2)}_{n_1n_2}\right] \\ n_1 &= 0, 1, \ldots M_1 - 1 \quad n_2 = 0, 1, \ldots M_2 - 1 \end{aligned} \tag{63}$$

using $\Phi^{(1,2)}_{n_1n_2}$ as the phase angles and $\Delta\omega_1 = \frac{\omega_{1u}}{N_1}$ $\Delta\omega_2 = \frac{\omega_{2u}}{N_2}$ as defined in (55).

The upper cut-off wave number defines the point from which the Power Spectral Density is assumed to be zero, such that $\omega_i$ lies in the region

$$-\omega_{1u} \leq \omega_1 \leq \omega_{1u} \quad \text{and} \quad -\omega_{2u} \leq \omega_2 \leq \omega_{2u}. \tag{64}$$

This leads to a simplification of Eq. (61) reading

$$\begin{aligned} f(k_1\Delta x_1, k_2\Delta x_2) = Re\Bigg[\sum_{n_1=0}^{M_1-1}\sum_{n_2=0}^{M_2-1}\Big\{ & B_{n_1,n_2}\exp\left[i\frac{2\pi n_1 k_1}{M_1} + i\frac{2\pi n_2 k_2}{M_2}\right] \\ & + \tilde{B}_{n_1,n_2}\exp\left[i\frac{2\pi n_1 k_1}{M_1} - i\frac{2\pi n_2 k_2}{M_2}\right]\Big\}\Bigg]. \\ k_1 = 0, 1, \ldots, M_1 - 1\,;\, k_2 = 0, 1, \ldots, M_2 - 1 & \end{aligned} \tag{65}$$

**Numerical Examples with Plots**

Let's consider a two-dimensional homogeneous stochastic field $f(x_1, x_2)$ with zero mean and an autocorrelation function $R(\xi_1, \xi_2)$ given as

$$R(\xi_1, \xi_2) = \sigma^2 \exp\left\{ -\left(\frac{\xi_1}{\ell_1}\right)^2 - \left(\frac{\xi_2}{\ell_2}\right)^2 \right\}. \tag{66}$$

Here $\sigma^2$ is the standard deviation of the random field acting as an amplitude term and $\ell_i$ is proportional to the correlation length. Then the corresponding power spectral density reads

$$S(\kappa_1, \kappa_2) = \sigma^2 \frac{\ell_1 \cdot \ell_2}{4\pi} \exp\left\{ -\left(\frac{\ell_1 \kappa_1}{2}\right)^2 - \left(\frac{\ell_2 \kappa_2}{2}\right)^2 \right\}, \tag{67}$$

which holds the relation in Eq. (52). A stochastic sample can be generated by summation of cosines or FFT. By using Eq. (65) three different sample fields are obtained with values listed in Table 1. The generating code can be found in the Appendix Section and online.

Table 1: Three cases of Gaussian Random field samples simulated via Spectral Representation Method.

| | **Case 1** | **Case 2** | **Case 3** |
|---|---|---|---|
| **property** | **value** | **value** | **value** |
| $\sigma$ | 1 | 1 | 1 |
| $N_1$ | 16 | 16 | 16 |
| $N_2$ | 16 | 16 | 16 |
| $M_1$ | 64 | 256 | 256 |
| $M_2$ | 64 | 256 | 64 |
| **property** | **value / [m]** | **value / [m]** | **value/ [m]** |
| $\ell_1$ | 1 | 4 | 4 |
| $\ell_2$ | 1 | 4 | 1 |
| $\kappa_{u_1}$ | 5 | 1.25 | 1.25 |
| $\kappa_{u_2}$ | 5 | 1.25 | 5 |
| $\Delta\kappa_1$ | 0.3125 | 0.0781 | 0.0781 |
| $\Delta\kappa_2$ | 0.3125 | 0.0781 | 0.3125 |
| $L_{x_1}$ | 20.1 | 80.4 | 80.4 |
| $L_{x_2}$ | 20.1 | 80.4 | 20.1 |
| $\Delta x_1$ | 0.628 | 2.51 | 2.51 |
| $\Delta x_2$ | 0.628 | 2.51 | 0.628 |

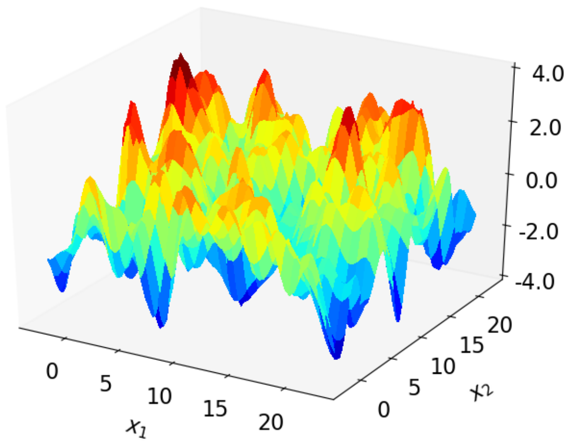


Figure 8: Case 1: Sample function of stochastic random field with correlation length scales $\ell_1 = \ell_2 = 1.0$.

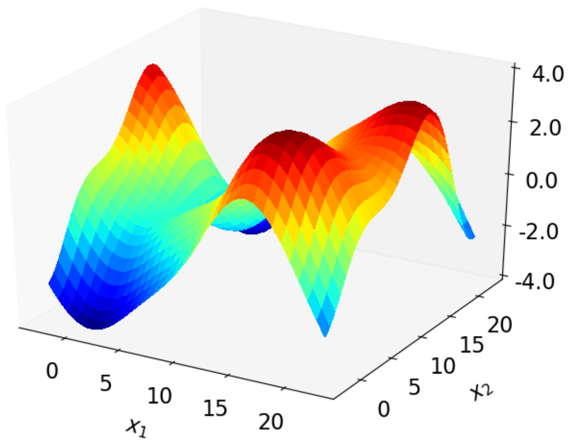


Figure 9: Case 2: Sample function of stochastic random field with correlation length scales $\ell_1 = \ell_2 = 4.0$.

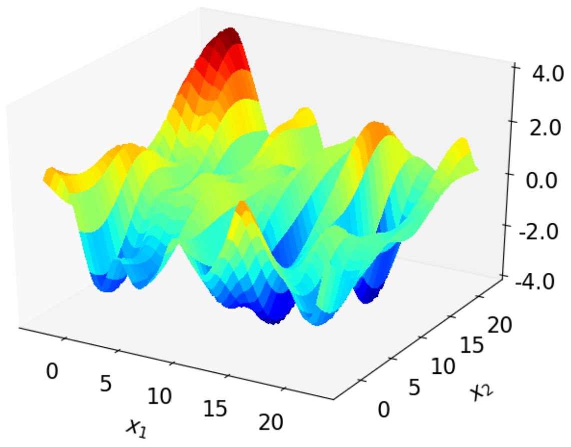


Figure 10: Case 3: Sample function of stochastic random field with correlation length scales $\ell_1 = 4.0$ and $\ell_2 = 1.0$.

### 3.3.4 Stochastic Partial Differential Equation Method (SPDE) of Gaussian Markov Random Fields

This Section introduces another way of sampling stochastic processes. Even though Spectral Methods were utilized to generate random fields for this thesis, the Stochastic Partial Differential Equation method remains an interesting field of research, thus, is shortly summarized to give an overview of alternative methods. The main sources summarized here are [63, 64, 65, 66, 67].

The Stochastic Partial Differential Equation method (SPDE) has the main benefit that contrary to SRM, also non-equidistant grids of random fields can be sampled. This comes with the drawback of being much slower since a PDE is solved at each grid point. In general, sampling Gaussian Markov Random Fields (GMRF) is another common approach to generate Gaussian random fields, alongside spectral methods, factorization and Karhunen-Loève. While factorisation methods suffer from the *big-N-problem*, GMRF overcome this by imposing a Markov property to the field, which means that the information of a point is only dependent on its most direct neighbours. This behaviour leads to an almost diagonal covariance matrix, which consequently holds a sparse precision matrix ($\mathbf{\Sigma} = \mathbf{Q}^{-1}$). The sparsity of the precision matrix $\mathbf{Q}$ makes computations much easier and faster.

The idea of GMRF is to approximate a Gaussian Random Field (GRF) by a Gaussian Markov Random Field. This is possible since GRFs can explicitly be constructed by using a certain stochastic partial differential equation (SPDE) which has GFs with Matérn covariance function as a solution, when driven by Gaussian white noise. The basis function representation, with piecewise linear basis functions and Gaussian weights with Markov dependence, is determined by a meshgrid (or triangulation) of the domain. In addition, the approximation with GMRF by imposing a Markov property only uses the square root of the time required by standard algorithms, e.g. factorisation, see [63].

As given in Section 3.3.2, factorisation methods require the calculation of the inverse covariance, the precision matrix $\mathbf{Q}$ at every iterations. Thus, computations scale with $\mathcal{O}(N^d)$ for $d$-dimensions.

The trick of GMRF now is to make this matrix sparse by imposing a Markov property, hence $\mathbf{Q}_{\text{GMRF}} \approx \mathbf{Q}_{\text{GRF}}$. Under mild conditions [63], the Cholesky factorisation $\mathbf{Q} = \mathbf{R}^T\mathbf{R}$ also becomes sparse and a realization of a field $\boldsymbol{y}$ can be obtained as

$$\boldsymbol{y} = \boldsymbol{\mu} + \mathbf{R}^{-1}\boldsymbol{z}, \qquad \boldsymbol{z} \in \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{68}$$

with $\boldsymbol{\mu}$ and $\mathbf{I}$ being the mean and unity respectively. The conditional expectation becomes

$$\boldsymbol{\mu}_{\mathbf{X}_1|\mathbf{X}_2} = \boldsymbol{\mu}_1 + \mathbf{R}_{11}^{-1}\Big((\mathbf{R}_{11}^{-1})^T\big(\mathbf{Q}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2)\big)\Big) \tag{69}$$

instead of

$$\boldsymbol{\mu}_{\mathbf{X}_1|\mathbf{X}_2} = \boldsymbol{\mu}_1 + \mathbf{\Sigma}_{12}\mathbf{\Sigma}_{22}^{-1}(\boldsymbol{x}_2 - \boldsymbol{\mu}_2). \tag{70}$$

For calculating the determinant one can use the relation

$$\log|\mathbf{R}| = \text{trace}\big(\log(\mathbf{R})\big)\ . \tag{71}$$

In contrast to Section 3.2, GMRF utilize the Matérn class as common choice for a covariance function, with $||\mathbf{d}||$ as the distance between two points. The kernel, or covariance function reads

$$\mathrm{cov}(||\mathbf{d}||) = \frac{\sigma^2}{2^{\nu-1}\Gamma(\nu)}(\boldsymbol{\eta}||\mathbf{d}||)^{\nu}\, \mathbf{K}_{\nu}(\boldsymbol{\eta}||\mathbf{d}||), \qquad \mathbf{d} \in \mathcal{R}^d \tag{72}$$

with $\mathbf{K}_{\nu}$ being the modified Bessel function of the second kind, $\boldsymbol{\eta} > 0$ ia a scale parameter, $\nu > 0$ the smoothness and the marginal variance is given with $\sigma^2 > 0$. Fields with Matérn covariances are the solution to Stochastic Partial Differential Equations, see Whittle (1954, 1963) [68, 69], hence

$$\left(\boldsymbol{\eta}^2 - \Delta\right)^{\alpha/2}\boldsymbol{y}(\boldsymbol{x}) = \mathcal{W}(\boldsymbol{x}) \tag{73}$$

with $\Delta = \sum_i \frac{\partial^2}{\partial \boldsymbol{x}_i{}^2}$, $\alpha = \nu +$ d/2 and $\sigma^2 = \frac{\Gamma(\nu)}{\Gamma(\alpha)\eta^{2\nu}(4\pi)^{d/2}}$. Moreover, $\mathcal{W}(\boldsymbol{x})$ is formally defined as Gaussian white noise. Thus, the Matérn class wave number spectrum is given by

$$\mathbf{R}(\mathbf{k}) \propto \frac{1}{(\boldsymbol{\eta}^2 + ||\mathbf{k}||^2)^{\alpha}}\,. \tag{74}$$

According to Rozanov (1977) [70] a stationary field is Markov if and only if the spectral density is a reciprocal of a polynomial. The idea is to construct a discrete approximation of the continuous field using basis functions $\{\psi_k\}$ and weights $\{w_k\}$,

$$\boldsymbol{y}(\boldsymbol{x}) = \sum_k \psi_k(\boldsymbol{x}) w_k \tag{75}$$

in order to find the distribution of the weights $w_k$ by solving $(\boldsymbol{\eta}^2 - \Delta)^{\alpha/2}\boldsymbol{y}(\boldsymbol{x}) = \mathcal{W}(\boldsymbol{x})$ and obtaining $\boldsymbol{y}(\boldsymbol{x})$ as a weak solution to the SPDE for each set of test functions $\{\psi_k\}$.

$$\Big[\langle \psi_{\mathbf{k}}, (\eta^2 - \Delta)^{\alpha/2}\boldsymbol{y}(\boldsymbol{x})\rangle\Big]_{\mathbf{k}=1,\dots,n} = [\langle \psi_{\mathbf{k}}, \mathcal{W}\rangle]_{\mathbf{k}} \tag{76}$$

Following [67] closely one can replace the solution $\boldsymbol{x}$ with its basis expansion to obtain

$$\Big[\langle \psi_i, (\eta^2 - \Delta)^{\alpha/2}\psi_j\rangle\Big]_{i,j}\mathbf{w} = [\langle \psi_{\mathbf{k}}, \mathcal{W}\rangle]_{\mathbf{k}}\,. \tag{77}$$

In case of $\alpha = 2$ and $\psi_i = \psi_j$ the result is called *Galerkin* solution, see [67] p. 13. A review by Lindgren (2011) [63] summarizes the benefits of GMRF in the following way:

- *no positive definite matrix constraint for the covariance matrix:* The SPDE method is independent of the direct construction of a positive definite matrix.
- *symmetry property:* Usually, the covariance matrix is restricted to a symmetry property, which can be dropped for SPDE.
- $\mathcal{S}^2$ *manifolds:* Multivariate GRFs can also be constructed on manifolds, e.g. a sphere.
- *Markov property:* The Markov property is often indispensable for model analysis using Markov Chain Monte Carlo techniques, hence GMRF simulations are fast. Rue et. al. [71] proposed an algorithm under the assumption of a $n_1 \times n_2, n_1 \leq n_2$ grid with $(2m+1) \times (2m+1)$ points a GMRF which reduces the simulation cost to $2n_1^2 n_2 m$ flops.

One might think that the Matérn covariance function is rather restrictive for statistical modelling, but it covers the most commonly used models in spatial statistics [63]. Also, Stein (1999) [72], p. 14 has a practical suggestion: "Use the Matérn model" [73]. For more information about the Matérn family, see [73] Section 2.6, also [74] and more recently [75].

For an extensive introduction, the interested reader is referred to [29, 63, 65, 76, 47]. Modern applications with outlined toy examples can be found in Staber (2018) [77], as well as a modified approach of a scalable parallel scheme to sample Gaussians over very large domains is presented in De Carvalho (2019) [42]. An excellent introduction to Gaussian Markov Random Fields (GMRF) is provided in the paper by Rue et. al. (2002) [71], as well as in the book from 2005 [64], which provides an application oriented approach to GMRFs with SPDE. For sampling RFs on complex geometries one is referred to a more recent paper by Pezzuto (2019) [78]. This Section also takes references from talks given by Lindström (2014) [67] and Lindgren (2015) [79]. Helpful lecture notes about SPDE and stochastic processes are also provided by Lindgren (2006) [32] as well as in the work done by Lang (2007) [80].

Some code snippets of how to generate random fields via SPDE method are attached online [41].

## 3.4 Non-Gaussian Process Theory and Gaussian Related Distributions

Even though Gaussian distributions clearly stand out with their effectiveness, flexibility and the fact that a lot of samples tend to a Gaussian distribution by means of the central limit theorem, yet nature still not always behaves in a Gaussian manner. In fact, many natural phenomena have strong non-Gaussian characteristics, like being heavy tailed or strictly bounded. Thus, an effective way to sample non-Gaussian random fields is needed and outlined in this Section. The interested reader is referred to the most cited reference in this field, namely Grigoriu (1995) [81], which gives a general overview of non-Gaussian translation process theory with the bonus of additional MATLAB supplements. Further references and numerical examples for non-Gaussian fields of analytic distributions, such as Beta or Log-normal random fields, are outlined in the next Section.

In general, over the years sampling methods have improved massively, not only from a computational point of view, also from a theoretical viewpoint. Since the early years the authors Grigoriu [82, 83, 84, 85], Yamazaki [86] and Popescu [87] proposed similar algorithms for generating a non-Gaussian, scalar (standard deviation $\sigma_G$) random field $H(\mathbf{x})$ with a prescribed correlation function $\rho_{\mathrm{H}}(\boldsymbol{\tau})$. In principle, there are two steps necessary: first, sample a zero-mean, scalar Gaussian random field $G(\mathbf{x})$, with a prefixed correlation structure $\rho_{\mathrm{G}}(\boldsymbol{\tau})$ and second, transfer $G(\mathbf{x}) \to H(\mathbf{x})$ according to

$$H(\mathbf{x}) = f\big[G(\mathbf{x})\big], \tag{78}$$

where $f[\cdot]$ is represented by a nonlinear function. If the mapping $f\,[G(\mathbf{x})]$ is represented by a nonlinear function, the process $H(\mathbf{x})$ is truly non-Gaussian.

More explicitly,

$$H(\mathbf{x}) = F^{-1}_{\text{Non-Gaussian}} \circ \Psi_{\text{Gaussian}}\big[G(\mathbf{x})\big] = F^{-1}_{\text{NG}}\Big\{\Psi\big[G(\mathbf{x})\big]\Big\}, \tag{79}$$

whereby $F^{-1}_{NG}$ stands for the inverse of the prescribed non-Gaussian cumulative distribution (CDF) and $\Psi(\cdot)$ represents the standard Gaussian CDF, see Fig. 5.

The transformation is called *memoryless*, if the value $H(\mathbf{x})$ at any arbitrary instant $\mathbf{x}$ only depends on the value of $G(\mathbf{x})$ at $\mathbf{x}$. A process $H(\mathbf{x})$ is said to be stationary in the strict sense, if $G(\mathbf{x})$ is stationary [81, 88]. Contrary, if the result of the transformation $H(\mathbf{x}) = f\big[G(\mathbf{x})\big]$ does not only depend on the current value of $\mathbf{x}$, but also depends on previous history, the process is denoted to have memory. Moreover, the transformation from Gaussian to non-Gaussian in a "forward" manner is always possible, if auto-correlation function (ACF) or power spectral density function (PSDF) of the Gaussian process are known [88]. Each of the two is good enough, because the counterpart can readily be calculated using Wiener-Khintchine theorem. The main goal of translation process theory is to calculate from a known ACF $R_G(\boldsymbol{\tau})$, of a zero-mean Gaussian process (with $\boldsymbol{\tau}$ being the distance/correlation of the input space dimension points), a specific probability distribution with desired power spectral density $S_{NG}(\boldsymbol{\omega})$. It can be proven, see Grigoriu 1995 [81], that the auto-correlation functions of the translation field and its underlying

Gaussian field are linked via the Rosenblatt transformation, see [88, 89]

$$R_{NG}(\tau) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F_{\mathrm{NG}}^{-1}\Big\{\Psi[G(\boldsymbol{x_1})]\Big\}F_{\mathrm{NG}}^{-1}\Big\{\Psi[G(\boldsymbol{x_2})]\Big\}\ \phi\{x_1, x_2; \rho_G(\boldsymbol{\tau})\}\ \mathrm{d}x_1\mathrm{d}x_2 \tag{80}$$

In this integral $\phi\{x_1, x_2; \rho_G(\boldsymbol{\tau})\}$ is the joint Gaussian probability density and $\rho_G(\tau)$ denotes the normalized correlation function defined as

$$\phi\{x_1, x_2; \rho_G(\boldsymbol{\tau})\} = \frac{1}{2\pi\sigma^2\sqrt{1-\rho_G{}^2(\boldsymbol{\tau})}} \exp\left(\frac{x_1^2 + x_2^2 - 2\rho_G(\boldsymbol{\tau})\ x_1\ x_2}{2\sigma^2\sqrt{1-\rho_G{}^2(\boldsymbol{\tau})}}\right), \tag{81}$$

and respectively

$$\rho_G(\boldsymbol{\tau}) = \frac{R_G(\boldsymbol{\tau})}{\sigma^2}. \tag{82}$$

Unfortunately, the direct inversion of this transformation is not always possible. This can be a problem if, i.e. in Astrophysics a stellar objects with a specific, yet unknown, target SDF is examined, which makes it difficult to sample random fields accordingly from it. Even though Wiener-Khintchine theorem gives an appropriate target $R_{NG}(\boldsymbol{\tau})$ it may not have an analytic CDF. Then, the underlying Gaussian ACF cannot be determined and the prescribed non-Gaussian $R_{NG}(\boldsymbol{\tau})$ and $F_{NG}$ are said to be "incompatible" [88]. One solution to this problem can be found, if the underlying distribution belongs to the family of fundamental distribution, i.e. Gamma-, Beta-, Uni-, Logn- distribution. Those are outlined in Section 3.5. However, there is yet another option, which is the iterative approximation outlined in Section 3.6.

## 3.5 Non-Gaussian Process Sampling Methods for Analytic Functions

Sampling non-Gaussian random fields, as outlined in equations (100) - (103), can be a daunting task, especially if the correlation functions $\rho_{\mathrm{G}}$ and $\rho_{\mathrm{H}}$ are not explicitly calculable. Fortunately, for some distributions an analytic relation exists, making them easy to sample from and thus, is the topic of this Chapter. A very nice introduction can be also be found in [90], whereas [91] summarizes the topic more concisely.

### 3.5.1 Gamma Distribution

Let $G_s(\mathbf{x})$ with $s = 1, 2, \dots, 2m$ be a collection of independent zero-mean Gaussian fields with the same covariance function $\rho_{\mathrm{G}}$. Then, Gamma random fields are calculated as

$$\mathcal{G}_m(\mathbf{x}) = \frac{1}{2}\sum_{s=1}^{2m} G_s^2(\mathbf{x}). \tag{83}$$

That is, because the corresponding one-dimensional marginal PDF is a Gamma distribution with $m$ degrees of freedom

$$f_{\mathcal{G}_m}(g) = \frac{1}{\Gamma(m)} g^{m-1} e^{-g/2}, \qquad g \geq 0, \tag{84}$$

where $\Gamma(\cdot)$ is the Gamma function [90]. In addition, the $k$-moments of the distribution are given by

$$\mathbf{E}\left[\mathcal{G}_m^k\right] = \frac{\Gamma(m+k)}{\Gamma(m)}, \qquad k > -m. \tag{85}$$

In particular, the mean and variance are

$$\mu_{\mathcal{G}_m} = \sigma^2_{\mathcal{G}_m} = m. \tag{86}$$

According to [90] the relation between $\rho_{\mathcal{G}_m}(\boldsymbol{\tau})$ and $\rho_{\mathrm{G}}(\boldsymbol{\tau})$ is independent of m and yields

$$\rho_{\mathcal{G}_m}(\boldsymbol{\tau}) = \rho_G^2(\boldsymbol{\tau}). \tag{87}$$

One remarkable characteristic of the Gamma distribution is, that it holds both, the Chi-squared as well as the Exponential distribution as special cases.

### 3.5.2 Beta Distribution

Having sampled two independent Gamma distributed fields, e.g. $\mathcal{G}_m(\mathbf{x})$ and $\mathcal{G}_n(\mathbf{x})$, characterized by the same correlation function $\rho_{\mathcal{G}}(\boldsymbol{\tau})$ it is possible to obtain a Beta distributed random field as

$$\mathcal{B}_{mn}(\mathbf{x}) = \frac{\mathcal{G}m(\mathbf{x})}{\mathcal{G}_m(\mathbf{x}) + \mathcal{G}_n(\mathbf{x})}. \tag{88}$$

Their one-dimensional marginal PDF is a Beta$(m, n)$ distribution

$$f_{\mathcal{B}_{mn}}(b) = \frac{1}{\mathcal{B}(m,n)} b^{m-1}(1-b)^{n-1}, \quad 0 \leq b \leq 1. \tag{89}$$

The respective moments of order $k$ are given as

$$E\left[\mathcal{B}^k\right] = \frac{\Gamma(m+k)\Gamma(m+n)}{\Gamma(m)\Gamma(m+n+k)}. \tag{90}$$

In particular, the mean and variance are

$$\mu_{\mathcal{B}_{mn}} = \frac{m}{m+n}, \quad \sigma^2_{\mathcal{B}_{mn}} = \frac{mn}{(m+n)^2(m+n+1)}. \tag{91}$$

According to [90] the relation between $\rho_{\mathcal{B}_{mn}}(\boldsymbol{\tau})$ and $\rho_{\mathrm{G}}(\boldsymbol{\tau})$ can be expressed as

$$\rho_{\mathcal{B}_{mn}}(\boldsymbol{\tau}) = 1 - S_{m+n}\left[\rho_{\mathrm{G}}(\boldsymbol{\tau})\right], \quad \text{with } n+m > 1, \tag{92}$$

whereas

$$S_q(\rho) = q\left(\frac{1-\rho}{-\rho}\right)^q \left[\log(1-\rho) - \sum_{i=1}^{q-1} \frac{1}{i}\left(\frac{-\rho}{1-\rho}\right)^i\right], \quad 0 \leq \rho \leq 1, q \in \{1, 2, \dots\}, \tag{93}$$

with bounding values at $S_q(0) = 1$ and $S_q(1) = 0$. Thus, the class of Beta fields is a useful way to describe strictly bounded random distributions. In addition, a special case of the Beta distribution is given at values $m = 1$ and $n = 1$ which corresponds to the Uniform distribution $\mathcal{B}_{11}(\mathbf{x})$. This relation is utilized within this work to generate the uniformly distributed random fields used as input to the FEM model.

### 3.5.3 Lognormal Distribution

Another quite interesting relation of an analytic non-Gaussian random fields is given by the Lognormal field. If $G(\mathbf{x})$ is a homogeneous, zero-mean, unit-variance Gaussian random field with correlation $\rho_{\mathrm{G}}$, one can define a Lognormal random field as

$$\mathcal{L}(\mathbf{x}) = e^{\mu + G(\mathbf{x})} \tag{94}$$

where $\mu$ and $\sigma > 0$ are two real parameters. Moreover, the field is characterized by the one-dimensional marginal PDF

$$f_{\mathcal{L}}(l) = \frac{1}{l\sigma\sqrt{2\pi}} e^{-(\ln(l)-\mu)^2/(2\sigma^2)}, \text{with } \ l > 0. \tag{95}$$

The $k$-th moments can be written as

$$E\left[\mathcal{L}^k\right] = e^{k\mu + k^2\sigma^2/2} \tag{96}$$

where the two first moments, mean and variance, yield

$$\mu_{\mathcal{L}} = e^{\mu + \sigma^2/2}, \tag{97}$$

$$\sigma^2_{\mathcal{L}} = e^{2\mu + \sigma^2/2}(e^{\sigma^2 - 1}). \tag{98}$$

The relationship between Lognormal and Gamma fields, $\rho_{\mathcal{L}}(\boldsymbol{\tau})$ and $\rho_{\mathrm{G}}(\boldsymbol{\tau})$, can be expressed as

$$\rho_{\mathcal{L}}(\boldsymbol{\tau}) = \frac{e^{\sigma^2 \rho_{\mathrm{G}}(\boldsymbol{\tau})} - 1}{e^{\sigma^2} - 1} \tag{99}$$

### 3.5.4 Results

Numerical examples from these random distributions are shown in Fig. 11 and Fig. 12. The samples were generated via Fast-Fourier-Transform method (Spectral method), see Chapter 3.3.3, in two spatial dimensions. Every point is spatially correlated to its neighbours, according to the correlation lengthscale. The samples of random fields are generated in the boundaries of $[0, 1]$, whereas yellow depicts higher values and blue accords to lower values, as indicated by the colorbar. In order to generate a Uniform random field, first, four Gaussian random fields were generated, yielding two Gamma distributed random fields, according to Eq. (83), which by definition (88) yielded an Uniform random field. This is a special case for the Beta distribution with parameters $m = n = 1$. The code used to generate these images is provided in the Appendix Section.

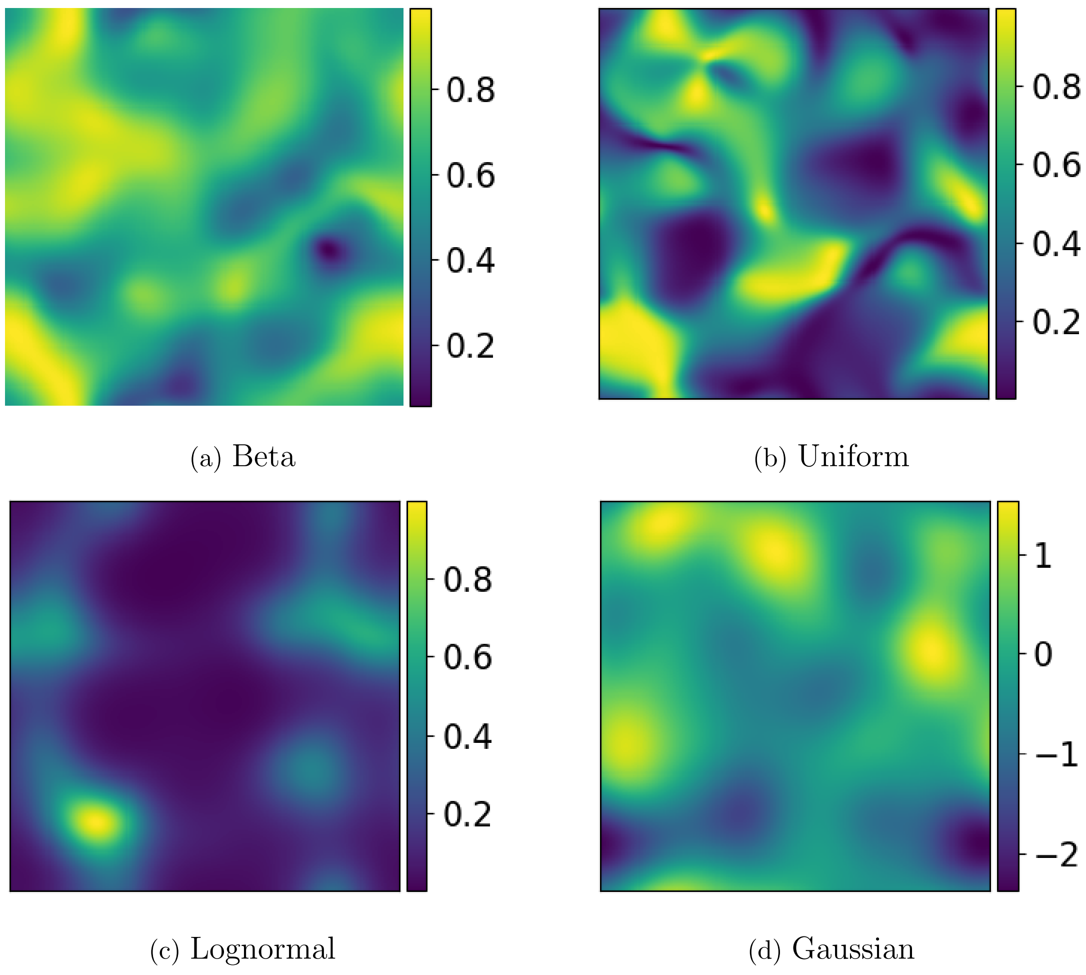
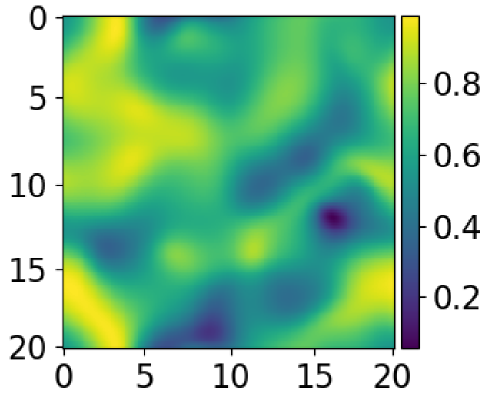
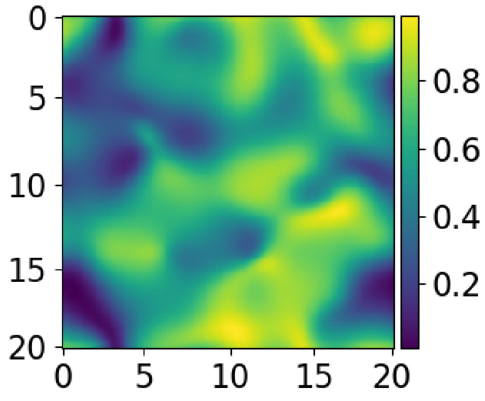


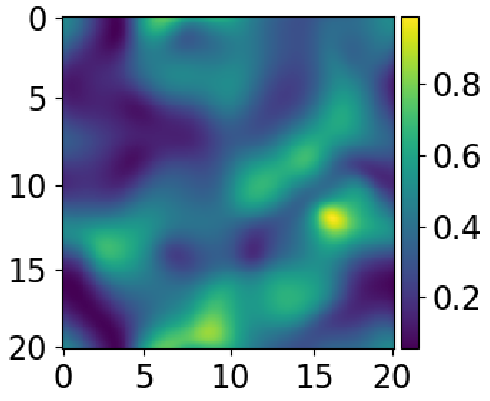
Figure 11: Samples of two-dimensional stochastic processes following a Beta distribution (upper left), Uniform distribution (upper right), a Lognormal distribution (lower left) and a Gaussian distribution (lower right). The bounds for the Beta distributed fields were within $[0, 1]$, whereas the Gaussian sample was sampled between $[-2, 2]$, both indicated by the colorbar.



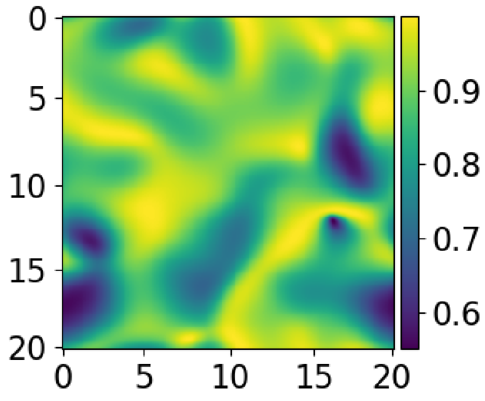
(a) Beta$(4, 2)$



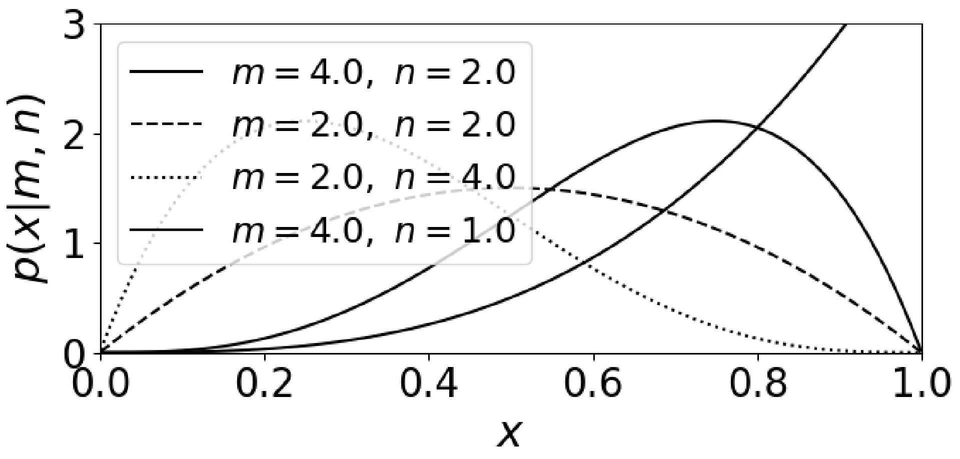
(b) Beta$(2, 2)$



(c) Beta$(2, 4)$



(d) Beta$(4, 1)$



(e) Comparison of Beta distributions

Figure 12: Samples of two-dimensional non-Gaussian processes following a Beta distribution with different parameters $m$ and $n$. At every point the fields correspond to samples drawn from a Beta-distribution, which is strictly bounded between $[0, 1]$. In addition, the random fields are spatially correlated, according to their correlation function. The colorbar indicates the value range of each field. The image at the bottom demonstrates the PDF in one dimension for different parameters $m$ and $n$ at one specific location in panels (a) – (d).

## 3.6 Other Approaches

Another way of sampling non-Gaussian random fields is given by an iterative solution, which was already introduced in the late 70's. One majorly important contribution of this approximation was done by Shields (2011) [88], who proposed an algorithm to approximate any PSDF, while making it unnecessary to sample a new Gaussian field at every iteration. This was common in previous methods [92, 86]. Since [88] was a milestone in this field, which lead to many subsequent publications, his proposed method is summarized briefly in this subchapter.

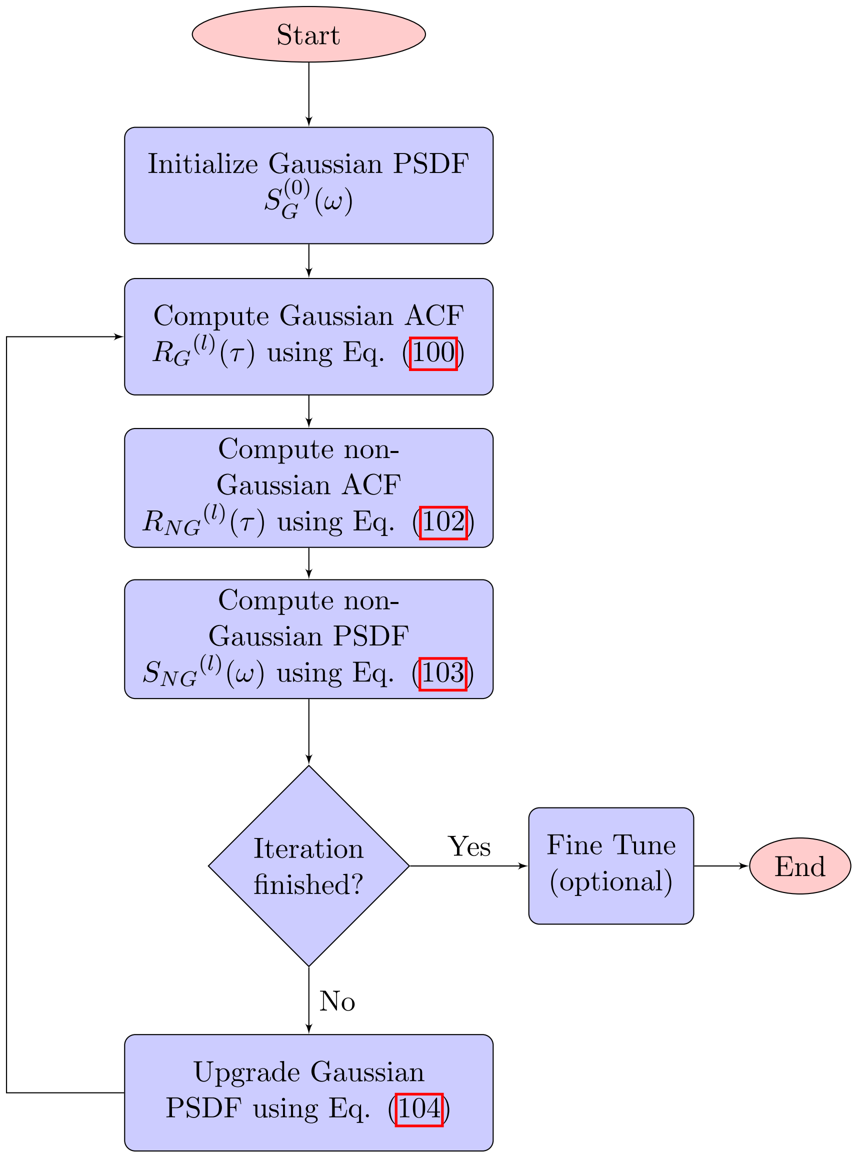

Start
Initialize Gaussian PSDF $S_G^{(0)}(\omega)$
Compute Gaussian ACF ${R_G}^{(l)}(\tau)$ using Eq. (100)
Compute non-Gaussian ACF ${R_{NG}}^{(l)}(\tau)$ using Eq. (102)
Compute non-Gaussian PSDF ${S_{NG}}^{(l)}(\omega)$ using Eq. (103)
Iteration finished?
Yes
Fine Tune (optional)
End
No
Upgrade Gaussian PSDF using Eq. (104)

Figure 13: Flowchart of proposed algorithm by Shields 2011 to approximate non-Gaussian random fields. For more details see Shields (2011).

To approximate any non-Gaussian random field one starts off with an initial guess for the power spectral density $S_G^{(0)}(\omega)$ at iteration step $l = 0$. The corresponding Gaussian ACF $R_G^{(0)}(\tau)$ can be readily computed by Wiener-Khintchine theorem e.g.

$$\mathcal{R}_G^{(l)}(\tau) = \int_{-\infty}^{\infty} \mathcal{S}_G^{(l)}(\omega) e^{i\omega\xi} \mathrm{d}\omega \tag{100}$$

where $i$ is the imaginary unit. The Gaussian correlation coefficient at iteration $l$ is computed as

$$\rho_G^{(l)}(\tau) = \frac{R_G^{(l)}(\tau)}{\sigma_G^2}. \tag{101}$$

where $\sigma_G^2$ is the scalar variance of the underlying Gaussian process. In the next step the non-Gaussian ACF $R_{NG}^{(l)}(\tau)$ is obtained via a non-linear mapping

$$R_{NG}^{(l)}(\tau) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F_{\mathrm{NG}}^{-1}\{\Psi[G(\mathbf{x_1})]\} \cdot F_{\mathrm{NG}}^{-1}\{\Psi[G(\mathbf{x_2})]\} \; \phi\{x_1, x_2; \rho_G^{(l)}(\tau)\} \mathrm{d}x_1 \mathrm{d}x_2 \; . \tag{102}$$

Finally the non-Gaussian PSDF at iteration $l$ is computed using the inverse Wiener-Khintchine theorem as follows

$$\mathcal{S}_{NG}^{(l)}(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \mathcal{R}_{NG}^{(l)}(\tau) e^{-i\tau\xi} \mathrm{d}\tau \; . \tag{103}$$

As a last step the Gaussian PSDF is upgraded via

$$\mathcal{S}_G^{(l+1)}(\omega) = \left[ \frac{\mathcal{S}_{NG}^T(\omega)}{\mathcal{S}_{NG}^{(l)}(\omega)} \right]^\beta \mathcal{S}_G^{(l)}(\omega) \tag{104}$$

The iterative scheme is looped until a certain convergence is reached, i.e. the relative difference between the non-Gaussian PSDF and the target non-Gaussian PSDF stabilize to a constant value

$$\epsilon^{(l+1)} = 100 \sqrt{\frac{\sum_{n=0}^{N-1} \left[ \mathcal{S}_{NG}^{(l+1)}(\omega_n) - \mathcal{S}_{NG}^T(\omega_n) \right]^2}{\sum_{n=0}^{N-1} \left[ \mathcal{S}_{NG}^{(T)}(\omega_n) \right]^2}}. \tag{105}$$

The exponent $\beta$ is chosen to optimize the convergence rate, common choices are within the range $1.3 \leq \beta \leq 1.5$. For further references and a numerical example of the iterative approximation of non-Gaussian stationary random processes, see [88]. A visualisation of the work flow is outlined in Fig. 13.

To sum up, the computational advantage of this algorithm stems from the fact, that the non-Gaussian PSDF can directly be computed from the Gaussian PSDF without generating any sample functions on the way. Even though those iterative sampling methods have gained extreme efficiency, the field still remains one of vivid research, see [93, 94, 89, 95, 96, 52]. This interest is also due to the wide range of applications ranging from simulation of wind force, turbulence of aircraft designs, climate models, calculating drag force on cylindrical elements or stress tensors in mechanical engineering, simulation of non-linear systems, and many more [81].

# 4 Finite Element Analysis

This Section gives a basic overview of the theoretical and computational methods used to generate the data for the surrogate learning. The main concepts of nonlinear solid mechanics on aortic tissue are summarized from [2]. In addition, a thorough introduction to nonlinear solid mechanics as well as theoretical background and can be found in [7], especially Chapters 2 and 6. After a brief introduction, a review of the paper written by Rolf-Pissarczyk et. al. [2] is given, in which the authors performed a uniaxial tensile test with a hyperelastic constitutive model. This model includes *collagen* fibers, *elastic* fibers and *ground substance* and is later used to generate the data for the surrogate learning in order to perform uncertainty quantification on.

The work of [2] is a cornerstone of this thesis, since the authors provide the computational framework, and more importantly, the FEM data used for the surrogate learning, which describes the properties of a hyperelastic material during an aortic dissection. The authors proposed a novel approach of finding a constitutive model, which consists of *ground substance*, *collagen* and *elastic fibers* and is able to capture degradation of inter-laminar elastic fibres during an aortic dissection. According to [2, 97] predominant reasons leading to AD are the degradation of elastic fibers, local accumulation of glycosaminoglycans (GAGs) and the loss and redifferentiation of smooth muscle cells. Moreover, they explain that those mechanisms are mostly found in the media, but can also involve adjoining layers and that most likely the pathological alteration of constituents can promote failure of the aortic wall.

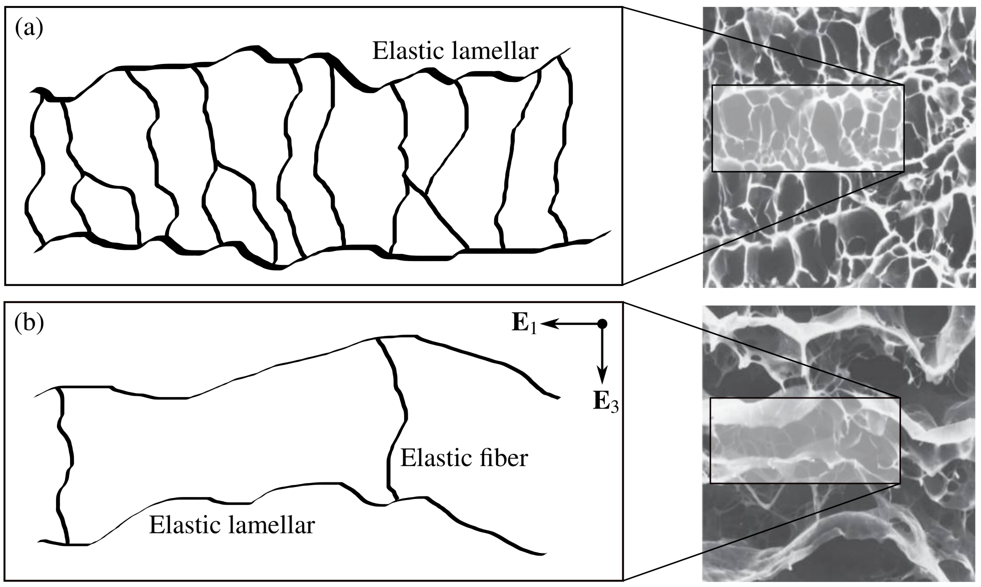


Figure 14: These 2D images illustrate a schematic representation of an elastic lamellar sheet within the human aortic media oriented in the circumferential direction $\mathbf{E}_1$. In (a) a healthy case with more interconnecting elastic fibers within an elastic lamellar sheet orientated radially in $\mathbf{E}_3$ is outlined building up the 3D architecture on the right handside, whereas (b) shows a diseased case with less elastic fibers in the elastic lamellar, also distributed radially in $\mathbf{E}_3$. These images are reprinted from Rolf-Pissarczyk et. al. [2].

Fig. 14 illustrated a two-dimensional schematic representation of an elastic lamellar sheet in circumferential direction $\mathbf{E}_1$ with radially distributed elastic fibers in $\mathbf{E}_3$ distinguishing two cases (a) the healthy and (b) the diseased lamellar unit of the medial layer in the aorta. Moreover, a general overview of previous attempts modelling AD is given in [2] and further references therein.

In the hyperelastic material model used in [2], elastic lamellae and inter-lamellar elastic fibres can be accounted for by a dispersion of elastic fibres, as postulated by Holzapfel et. al. [98], whereas inter-lamellar elastic fibres are assumed to be symmetrically dispersed in the lamellar unit of the media. In general, there are two main approaches to model fiber distributions, the generalized structural tensor and the angular integration, whereby a discrete version of the angular integration approach is also referred to as discrete fiber dispersion model, which was proposed by Li [99]. This model reduces computational costs by assuming the fibre dispersion to be a discrete sum of fibre contributions. This is done by discretizing a unit hemisphere, describing the dispersion of fibres by a finite number of elements, which determine the cost of computation [2]. In addition, a **degradation parameter** $\xi$ is introduced, which describes the disease-dependent degradation of elastic fibres. High values of the degradation parameter automatically exclude damaged or degraded elastic fibres in order to model the degradation of radially directed elastic fibres. The degradation initiates in the radial direction due to the highest occurring stretch. Given the random field input, the numerical analysis for generating the strain energy function including the Cauchy-Stress and elastic tensor elements was performed by Malte Rolf-Pissarczyk with the software `FEAP` [1].

## 4.1 Kinematics

As outlined in Holzapfel (2000) [7], Section 2, a **deformation map** $\boldsymbol{\chi}$ transfers between the initial (reference) $\mathbf{X}$ and the deformed (current) configuration $\mathbf{x} = \boldsymbol{\chi}(\mathbf{X})$. At every point the local deformation can be defined via the deformation gradient and is written as

$$\mathbf{F}(\mathbf{X}) = \frac{\partial \boldsymbol{\chi}(\mathbf{X})}{\partial \mathbf{X}}. \tag{106}$$

In the constitutive model [2], assume a strictly incompressible material, which constrains the deformation gradient to be positive, notated by the Jacobian $J = \det(\mathbf{F}) > 0$. Following the authors closely, the deformation gradient can be decomposed into a volumetric and an isochoric part, as $J^{1/3}\mathbf{I}$ and $J^{-1/3}\mathbf{F}$, respectively, with $\mathbf{I}$ being the second order unit tensor. The symmetric right Cauchy-Green tensor $\mathbf{C} = \mathbf{F}^T\mathbf{F}$ represents a measure of deformation in the reference configuration, whereas $\overline{\mathbf{C}} = \overline{\mathbf{F}}^{\mathrm{T}}\overline{\mathbf{F}}$ denotes a modification. In addition, the symmetric left Cauchy-Green tensor and its modified counterpart is provided by $\mathbf{b} = \mathbf{F}\mathbf{F}^{\mathrm{T}}$ and $\overline{\mathbf{b}} = \overline{\mathbf{F}}\,\overline{\mathbf{F}}^{\mathrm{T}}$. The first invariant can be further defined in the reference, as well in the current configuration, yielding

$$I_1 = \mathrm{tr}\,\mathbf{C} = \mathrm{tr}\,\mathbf{b}, \qquad \bar{I}_1 = \mathrm{tr}\,\overline{\mathbf{C}} = \mathrm{tr}\,\overline{\mathbf{b}} \tag{107}$$

and the fourth invariant $I_4$, describing the squared fibre stretch $\lambda^2$ in the direction of a vector $\mathbf{N}$ as

$$I_4 = \lambda^2 = \mathbf{C}\colon \mathbf{N}\otimes\mathbf{N} = \mathbf{n}\otimes\mathbf{n}, \qquad \bar{I}_4 = \overline{\lambda}^2 = \overline{\mathbf{C}}\colon \mathbf{N}\otimes\mathbf{N} = \overline{\mathbf{n}}\otimes\overline{\mathbf{n}}. \tag{108}$$

can be introduced likewise for both configurations using the dyadic product $\otimes$. Here, the overline indicator denotes the modified counterparts. In the reference configuration the direction of a fiber is denoted by the vector $\mathbf{N}$, consisting of polar angle $\Theta$ and azimuth angle $\Phi$ reading

$$\mathbf{N} = \sin\Theta\ \cos\Phi\ \mathbf{E}_1 + \sin\Theta\ \sin\Phi\ \mathbf{E}_2 + \cos\Theta\ \mathbf{E}_3, \tag{109}$$

where $\mathbf{E}_i$, $i = 1, 2, 3$, are the unit Cartesian basis vectors.

## 4.2 Strain-Energy Function

A hyperelastic material postulates a Helmholtz-free energy function $\Psi$, which is defined per unit volume, rather than per unit mass [7]. If this function solely depends on the strain tensor $\Psi = \Psi(\mathbf{F})$, then the Helmholtz-free energy is referred to as **strain energy function**, strain energy or stored energy. In this case the strain energy function represents the passive material behavior of the aortic wall, whereas the active material behaviour was neglected, see [2]. By an isothermic assumption this function can be decoupled for computational efficiency, thus

$$\Psi = \Psi_{\text{vol}} + \Psi_{\text{iso}}, \tag{110}$$

where $\Psi_{\text{vol}}$ and $\Psi_{\text{iso}}$ represent the purely volumetric part and the isochoric part of the deformation, respectively [7]. The volumetric is given as

$$\Psi_{\text{vol}} = \frac{K}{4}(J^2 - 1 - 2\ \ln(J)), \tag{111}$$

where $K$ is called bulk modulus and denotes a penalty parameter to enforce kinematic incompressibility. The isochoric part, which represents the respective constituents of the aortic wall, namely the ground substance $\Psi_{\text{g}}$, the collagen fibers $\Psi_{\text{c}}$ and the elastic fibers $\Psi_{\text{e}}$, is decomposed as

$$\Psi_{\text{iso}} = \Psi_{\text{g}} + \Psi_{\text{c}} + \Psi_{\text{e}}. \tag{112}$$

The ground substance is given by the (isotropic) neo-Hookean model, which only depends on the first invariant $\bar{I}_1$, i.e.

$$\Psi_{\text{g}}(\bar{I}_1) = \frac{\mu}{2}(\bar{I}_1 - 3), \tag{113}$$

where the constant $\mu > 0$ represents the shear modulus with the dimension of stress [2]. In addition, one can introduce a degradation parameter $\xi \in [0, 1]$ which describes the damage of elastic fibers as a result of the separation of the elastic lamellae, which is given as

$$\xi = \begin{cases} 0 & \text{healthy,} \\ 1 & \text{completely damaged.} \end{cases}$$

Moreover the authors introduce a degradation, or critical fiber angle $\Theta_\xi = \pi\ \xi/2$, to exclude elastic fibers from the total strain-energy function, i.e.

$$\Psi_{\text{e}_n} = \begin{cases} f_{\text{e}_n}(\lambda_{\text{e}_n}^2) & \text{if} \quad \Theta_n \geq \Theta_\xi \quad \text{and} \quad \lambda_{\text{e}_n}^2 \geq 1, \\ 0 & \text{else,} \end{cases} \tag{114}$$

with $f_{\mathrm{e}_n}$ representing the strain-energy function of $\lambda^2_{\mathrm{e}_n} = I_{4_n} = \mathbf{C} \colon \mathbf{N}_n \otimes \mathbf{N}_n$, which is defined as squared fiber stretch of elastic fibers. $\lambda^2_{\mathrm{c}_n}$ denotes the squared fiber stretch for collagen fibres respectively. Moreover, the subindex $n = \{0, \dots, m\}$ specifies the fiber number. Fig. 15 shows that damaged elastic fibres are excluded from the model, in case that a fibre angle $\Theta_n$ is smaller than a critical fiber angle $\Theta_\xi$ Fig. 15.

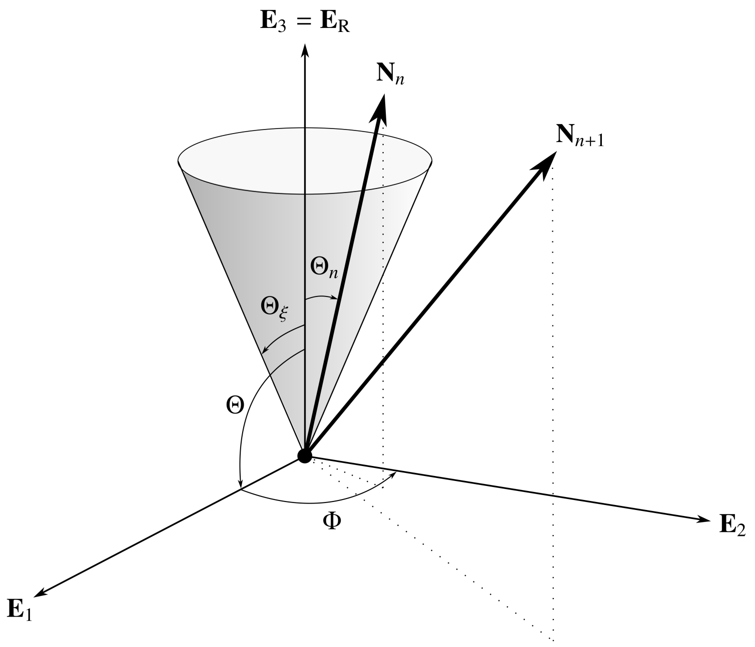


Figure 15: Critical fiber angle. This image illustrates elastic fibers distributed in and outside the cone, which is defined by an critical fiber angle $\Theta_\xi$. Any fiber $n$ exceeding the critical angle is excluded from the strain-energy function, e.g. $\mathbf{N}_{n+1}$, whereas elastic fibers inside the cone, such as $\mathbf{N}_n$, are included. The discrete fiber angle of fibers distributed inside the cone is given by $\Theta_n$. $\mathbf{E}_1, \mathbf{E}_2$ and $\mathbf{E}_3$ represent the unit Cartesian basis vectors. The indices describe the circumferential, the axial and the radial direction, respectively. This image was reprinted from [2].

Then, the isochoric part of the strain-energy function reads

$$\Psi_{\mathrm{iso}} = \Psi_{\mathrm{g}}(\bar{I}_1) + \sum_{n=1}^{m} \rho_{\mathrm{c}_n} \Psi_{\mathrm{c}_n}(\bar{\lambda}^2_{\mathrm{c}_n}) + \sum_{n=1}^{m} \rho_{\mathrm{e}_n} \Psi_{\mathrm{e}_n}(\bar{\lambda}^2_{\mathrm{e}_n}). \tag{115}$$

The total strain energy function Eq. (110) can now be inserted in the constitutive equation to calculate the Cauchy stress tensor

$$\boldsymbol{\sigma} = 2\rho\mathbf{b}\frac{\partial\Psi}{\partial\mathbf{b}} \tag{116}$$

with the left Cauchy-Green tensor $\mathbf{b} = \mathbf{F}\mathbf{F}^T$ being dependent on the deformation gradient $\mathbf{F}$ and the displacement vector $\mathbf{u}$ since $\mathbf{F} = 1 + \mathrm{Grad}(\mathbf{u})$ and $\rho$ being the fibre density. The boundary constraints follow from the equilibrium of the first Cauchy-Euler's law of motion

$$\mathrm{div}(\boldsymbol{\sigma}) + \rho\bar{\mathbf{b}} = \rho\ddot{\mathbf{u}} \tag{117}$$

with $\bar{\mathbf{b}}$ being the volume force vector. For further details see [7].

## 4.3 Uniaxial Tensile Test with the Rolf-Pissarczyk-Holzapfel Model

As presented in [2], an uniaxial extension test of an incompressible unit cube, with dimensions $1 \times 1 \times 1$ [mm$^3$], is performed and shown in Fig. 16. The unit cube is aligned by the unit Cartesian basis vectors $\mathbf{E}_1, \mathbf{E}_2$ and $\mathbf{E}_3$ and a uniform displacement along the top face was applied such that the loading direction coincides with the radial vector $\mathbf{E}_\mathrm{R} = \mathbf{E}_3$.

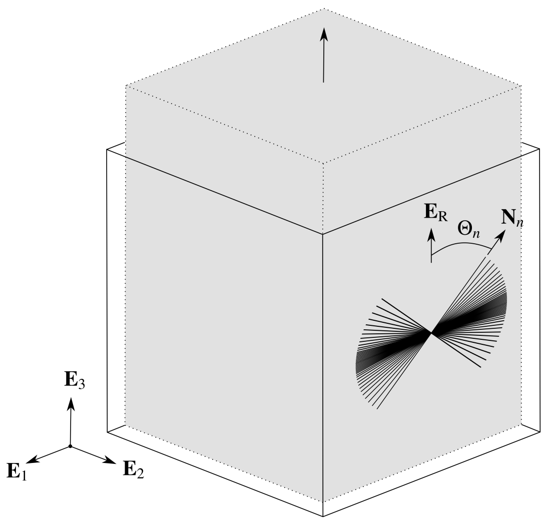


Figure 16: Uniaxial Extension. This image shows the reference and intermediate configurations of a unit cube under uniaxial extension in $\mathbf{E}_3$ direction. Hence, the radial vector $\mathbf{E}_\mathrm{R}$ is aligned with the $\mathbf{E}_3$-direction in the reference configuration. In this model a rotational symmetric dispersion of elastic fibers was investigated. The fiber dispersion within a cone, defined by the critical fiber angle $\Theta_n$, of arbitrary fibers such as $\mathbf{N}_n$, is outlined by the cross-section along the $(\mathbf{E}_1, \mathbf{E}_3)$-plane. This image was taken from the paper by Rolf-Pissarczyk et. al. [2].

The input to the Finite Element analysis were uniformly distributed random fields describing the degradation parameter, which were generated via the Spectral method, as outlined in Chapter 3.3.3. The total length size of the tissue field was $l_\mathrm{field} = 25.05$ [mm], thus the correlation length of the degradation parameter was chosen to be around one-third the length of the total field reading $\ell = 8.35$ [mm] and the simulated noise added in the random field sampling was chosen to be $\sigma^2 = 0.173$.

Therefore, a two-dimensional random field was simulated, then duplicated, and under the assumption of a very thin tissue those two layers were stacked behind each other to obtain a three-dimensional mesh for the unit cube as shown in Fig. 16. The assumption holds for very thin layers of tissues. In further investigations one could directly sample three-dimensional random fields, with the disadvantage of longer run time, however, this was beyond the scope of this work.

Moreover, the two-dimensional random field was sampled on an equidistant grid of size $2048 \times 2048$ and further downsampled to a $20 \times 20$ image, such that the evaluated grid points of the low-resolution image coincide with the non-equidistant Gaussian integration points (GIP) of the adaptive Gauss-Kronrod quadrature method, which was used in the Rolf-Pissarczyk-Holzapfel model. An illustration of those grid points is given in Fig. 17. In the image, the blue crosses denote the Gauss-Kronrod integration points, the two blue regions represent two unit cubes and the two dashed regions each define a unit cell. The distance between the GIP within one unit cube is always $p_{\text{cell inner}} = 2\sqrt{\frac{1}{3}}$, whereas the distance from one cell to another is given by $p_{\text{cell neighbour}} = 2(1 - \sqrt{\frac{1}{3}})$ leading to the non-equidistant grid. Each blue coordinate cross in Fig. 17 corresponds to a GIP, denoted by red dots in the three-dimensional mesh representation in Fig. 18. The coordinates $\mathbf{E}_1$, $\mathbf{E}_2$, $\mathbf{E}_3$ denote the coordinate axes, with $\mathbf{E}_1$ being the circumferential, $\mathbf{E}_2$ the axial and $\mathbf{E}_3$ being radial direction, therefore $\mathbf{E}_3$ is the tensile direction of the uniaxial extension.

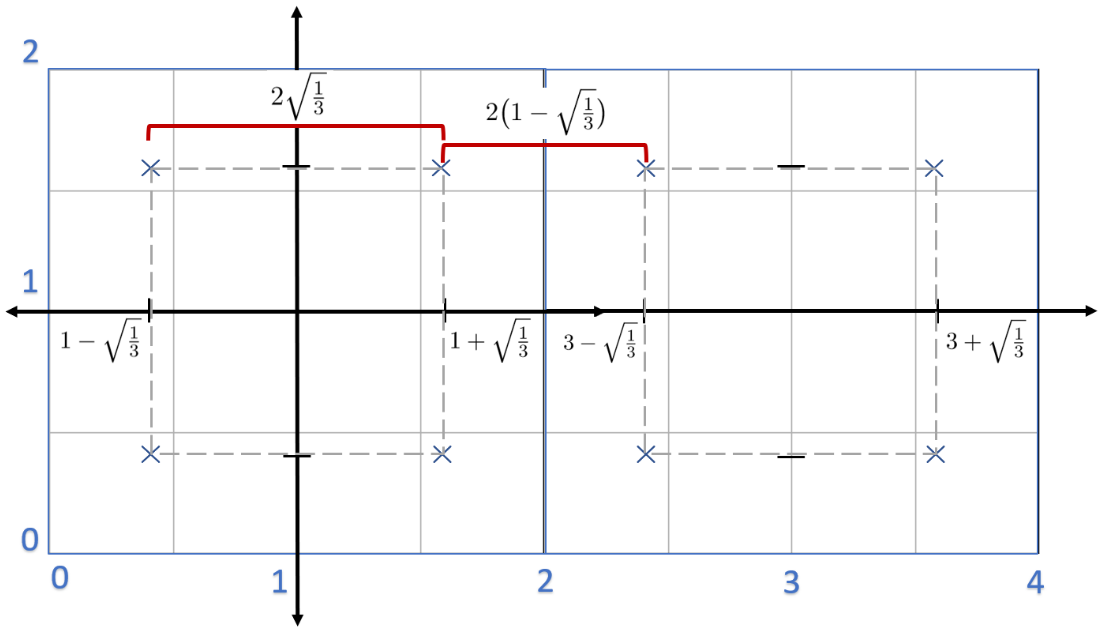


Figure 17: Sketch of Gaussian Integration read out along two axes.

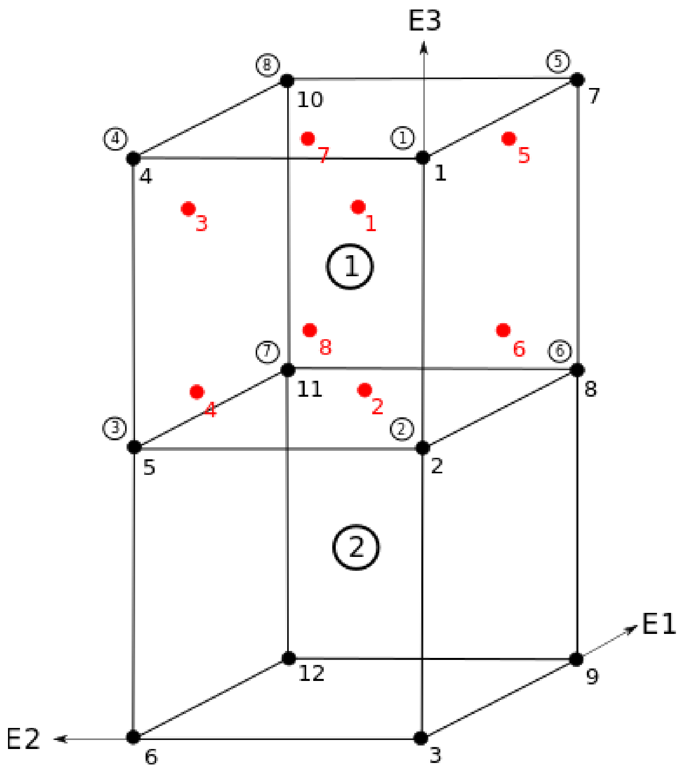


Figure 18: Three dimensional unit cube mesh. One unit cube of size $1 \times 1 \times 1$ [mm$^3$] holds eight Gaussian integration points (GIPs), indicated as red dots at which the random fields were sampled and used to calculate the solution to the Cauchy stress tensor. In particular, the tensile direction $\mathbf{E}_3$ was denoted as the QoI.

The reason why a higher resolution image was generated and then downsampled was because of the fact, that the Gaussian integration points of the FE calculation correspond to a non-equidistant grid, as shown below. Since Spectral methods, in a straightforward manner, are not applicable for non-equidistant points, a higher resolution image had to be generated. This was still faster than simulating the non-Gaussian field with SPDE or other investigated methods. Moreover, since the size of the high-resolution image is much larger than the actual grid, the error introduced by taking the nearest neighbouring points can be neglected. Further investigations could overcome this error by estimating the integration points via Bayes theorem, or by changing the grid of the FE analysis to be equidistant. For further references of the proposed model see [2].

# 5 Surrogate Model

## 5.1 An Introduction to Convolutional Neural Networks

**Convolutional Neural networks** (CNNs) are a family of deep neural networks which can extract spatial structure within data e.g. 1D time series or 2D images, which makes them an excellent candidate for image analysis.
Other than deep neural networks, which consists of fully connected layers, the layers in CNNs are sparsely connected and even share some parameters, which (usually) leads to a faster convergence by requiring less training data.
CNNs typically consist of three different components: **convolutional layers**, **pooling layers** and **fully-connected layers**. A convolutional layer consists of a so called **kernel**, sometimes also referred to as convolutional filter, which extracts local features, e.g. a person's nose, eyes, or mouth. Every learned feature is embedded in the model as a feature map. After training, the CNN will convolve these features over the tested image to see the global existence of local feature maps via an **activation vector**. In other words, the convolution of the image with the kernel results in a matrix, sometimes called **activation map**, which produces a high value (at a given location), if the feature represented in the convolutional filter is present at that location of the input. For a more detailed explanation, see the MIT video lecture provided at [100]. The pooling

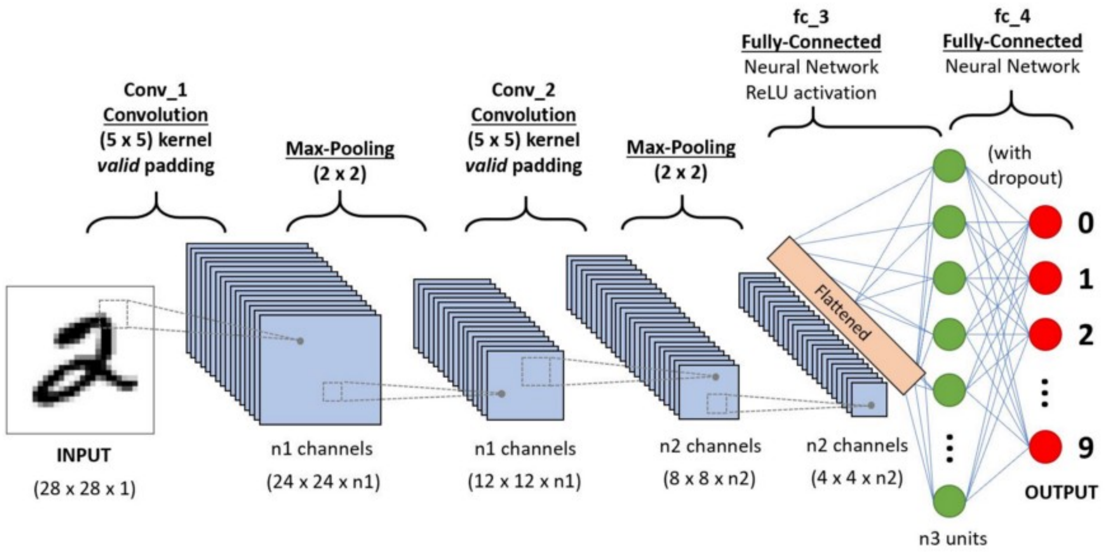


Figure 19: A classical Convolutional Neural Network consists of convolutional layers, pooling layers and fully connected layers. In this work however, the pooling layer was replaced by batch normalization, according to [101]. This image was taken from [102].

layer then extracts information, no matter of the location, hence it makes the network translational invariant. Pooling, sometimes also referred to as subsampling, is typically applied as either **max pooling** (most commonly used) or **average pooling**. As the name already indicates, max pooling takes the maximum value of the observed window, whereas average pooling takes the average value of the observed position of the kernel. The way a kernel is moved across the image is defined by the so called **stride**. A typical

value for stride is two, which means that the kernel is not moving one, but two pixels per time step along the image.
Finally, the third component of CNNs are the fully-connected layers, which produce different activation patterns, based on the set of activation feature maps. This means that neurons in the fully-connected layer will get activated for every component that is matched between the feature map and the input image. Thus, the number and variety of activated patterns gives plenty of options what an image can contain, but it is only the selection done by the fully connected layer in the end which concisely summarizes the extracted information of a given input. This information can then be classified by the output layer in the neural network in order to correctly classify the image. A brief example of a vanilla CNN classifying architecture, is shown in Fig. 19.

### 5.1.1 Training

When setting up a CNN, convolutional features, like eyes, nose, mouth, do not simply appear, but are the outcome of an optimization problem. The objective in Machine Learning is always to mimic features of some given data. This is obtained by minimizing a cost function (= finding the maximum a posteriori (MAP)), or in simple words, to learn a function which fits some form of data. Therefore, independent of the neural network type, the goal is always to reward correctly identified information and/or penalize unwanted behaviour in the model. The aim of training is to find the optimal **(hyper)parameters** like **weights** and **biases**, which minimize the cost function in a specific problem. Commonly used cost functions are e.g. the **root mean squared error (RMSE)** or **binary cross entropy loss (BCE)**.

### 5.1.2 Kullback-Leibler Divergence for Variational Formulation

Yet, there is another option for the loss function, namely the **Kullback-Leibler ($\mathcal{KL}$) divergence**. In [103] the author provides a simple introduction to the topic. Moreover, recommended sources include [25] and [26]. KL divergence has its origin in information theory where the primary goal is to quantify the amount of information in a dataset. The most important measure of information is entropy $H$ and its definition for a probability distribution $p(x)$ is denoted by

$$H = -\sum_i p(x_i) * \log\big(p(x_i)\big)$$

The key ingredient is borrowed from a frequently used method in probability theory, when highly complex distributions of data are approximated and replaced by simpler ones. Hence, $\mathcal{KL}$ divergence helps to measure how much information was kept, respectively lost, when making such approximations.

Frequently the goal is to obtain a posterior distribution $p(\theta \mid \mathcal{D})$, describing the data $\mathcal{D}$ by parameter set $\theta$. $\mathcal{KL}$ divergence $D_{\mathcal{KL}}$ can be defined by a slight modification of the above formula. One introduces a second distribution $q(\theta)$ which aims to approximate the posterior distribution via its logarithmic difference as

$$\begin{aligned} D_{KL} &= -\sum_i p(\theta_i \mid \mathcal{D}_i) \cdot \Big(\log\big(p(\theta_i \mid \mathcal{D}_i)\big) - \log\big(q(\theta_i)\big)\Big) \\ &= -\sum_i p(\theta_i \mid \mathcal{D}_i) \cdot \log\bigg(\frac{p(\theta_i \mid \mathcal{D}_i)}{q(\theta_i)}\bigg). \end{aligned}$$

In fact, $\mathcal{KL}$ divergence is the expectation $\mathbb{E}$ of the logarithmic difference between the probability of data in the original distribution compared to the approximating distribution. The normalization constant of the posterior can be dropped when $\mathcal{KL}$ is minimized, hence the unnormalized posterior is left in the equation as $\tilde{p}(\theta \mid \mathcal{D})$. Then, the expectation yields

$$D_{\mathcal{KL}} \coloneqq \mathbb{E}\Big[\log\big(q(\theta)\big) - \log\big(\tilde{p}(\theta \mid \mathcal{D})\big)\Big] \tag{118}$$

This equation is minimized by finding the optimal parameter $\theta^*$ yielding

$$q(\theta^*) \approx \tilde{p}(\theta \mid \mathcal{D}),$$

such that

$$q(\theta^*) = \arg\min \mathcal{KL}\Big(q(\theta)| \;\; p(\theta \mid \mathcal{D})\Big) = \arg\min \mathbb{E}\Big[\log\big(q(\theta)\big) - \log\big(\tilde{p}(\theta \mid \mathcal{D})\big)\Big]. \tag{119}$$

The combination of neural networks and $\mathcal{KL}$ divergence enables to learn complex approximate distributions of the data, by minimizing the information loss when approximating a distribution. A common application is a ”Variational Autoencoder” which learns the best way to approximate the information in a data set. Even more general is the area of Variational Bayesian Methods. Those methods come into play, if, for example, one wants to compute Monte Carlo simulations with intractable integrals which is a common technique in Bayesian inference. Those methods are often computationally too expensive, thus variational Bayesian methods can be used to partly replace such calculations [103]. A more detailed introduction to Variational Autoencoders is given in the next Section and their use in this work is outlined in Section 5.2.

### 5.1.3 Variational Autoencoder

In general, one has to distinguish between the terms **Encoder-Decoder** and **Variational Autoencoder**. An Encoder-Decoder is typically utilized wherever data needs to be compressed and decompressed. Usually, this occurs in two sorts of problems, namely image classification in ’Vanilla’ CNNs and image-to-image regression tasks. In this work a Variational Autoencoder is used, which is a slightly modified version of the first incorporating e.g. $\mathcal{KL}$ divergence. In this work, the terms Encoder-Decoder and Variational Autoencoder both encounter for the later one.

The two key words when dealing with **Variational Autoencoders**, short VAEs, or synonymously ’Bayesian Autoencoders’, are **Principle Component Analysis** (PCA) and **dimensional reduction**. Usually, the input data, which ought be learned by the neural network, is of high complexity. In order to reduce the data complexity to a 'neural network feasible format', the input needs to be compressed. Only then it can be fed through the neural network before being decompressed to a desireable output format again. The process of compression (e.g. via selection or extraction of features) is called **encoder**, whereas the reverse process of decompression is called **decoder**. The whole process of reducing the number of features to an encode space (also latent space) is understood as dimensional reduction. An encoder decoder architecture is considered good, if it keeps the maximum information when encoding, while showing minimal error when reconstructing the data in the decoder. Fig. 20 shows the principles of an Encoder-Decoder structure.

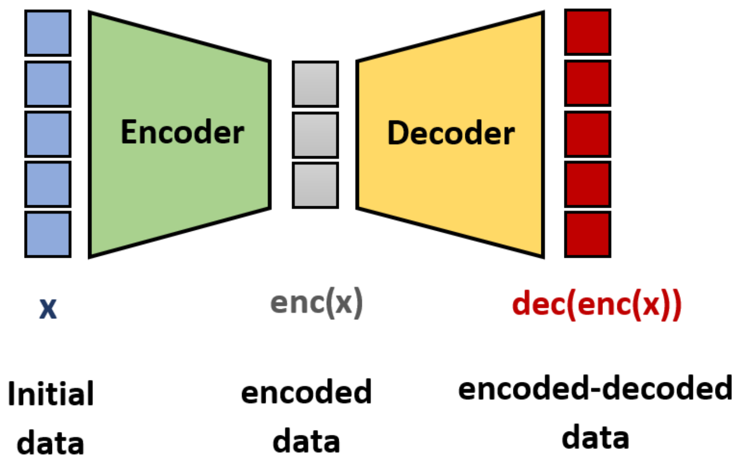


Figure 20: Principle of Encoder-Decoder structure. The initial data size is reduced when entering the encoder and its' shape is reconstructed when leaving the decoder, in order to map image input to output samples.

The second major ingredient of VAEs along dimensional reduction is Principle component analysis. PCA can be obtained by an eigendecomposition of the covariance matrix, or **single value decomposition** (SVD), in order to set up a linear projection of the data onto orthogonal subspaces. With PCA the axes of features in the data are transformed by a change basis to another linear orthogonal basis set. This basis represents the data along directions of maximal variance. Those axes are then sorted by their maximal variance. The main benefit of PCA is, that data can be stored much more efficiently, by using its statistics, without loosing any information. In addition, it is also common to apply dimensional reduction through PCA, namely by defining a cut off value, after which higher orders are neglected and hence, a certain percentage of variance is stored. A great introduction of Principle Component Analysis is given in [104].

## 5.2 Uncertainty Quantification with a Variational Autoencoder

The Encoder-Decoder structure used in this work heavily relies on previously presented efforts done by Zhu and Zabaras (2018) [3]. Thus, this Chapter can be seen as a brief summary of their paper. In general, a deep convolutional encoder-decoder network is of similar fashion to a deep learning image-to-image regression task. Therefore, the Bayesian approach to convolutional neural networks achieves state of the art performance in terms of prediction accuracy and uncertainty quantification in comparison to other approaches, like Gaussian processes [3]. Moreover, a variational gradient descent method [105], based on Stein's operator, i.e. **Stein variational gradient descent** (SVGD) was adapted to convolutional neural networks to perform Bayesian inference on millions of uncertain parameters. Furthermore, the encoder-decoder structure allows to extract multi-scale features and spatial correlations from the input, which are processed by the decoder in order to reconstruct the output. In [3] the model was tested to map a permeability input onto a flow/pressure output with stochastic dimensions up to 4225. In their work they present promising results, even for little sets of data, which has led to great approval in this field and thus, was chosen as the main workhorse in this work.

In general, the list of reasons why and how errors enter a computational model in the first place is long. It can reach from e.g. a model error, model parametrizations or specific model assumptions, incomplete data sets, incomplete material properties, as well as boundary conditions. To quantify uncertainty in complex models, either high computational power or a lot of computational run time is necessary, most of the time even both. In case of already computationally expensive Finite Element simulations, the approach of brute force uncertainty quantification often becomes unfeasible. Thus, the idea is to train a surrogate model, based on a limited number of simulation runs, in order to predict a greater set of solutions of a FEM solver. As a consequence, uncertainty quantification can be performed by using the surrogate predictions instead of solving the actual PDE thousands of times. One of the essential ideas for handling high-dimensional data with surrogate models is to learn the latent input representation automatically by supervision with the output in regression tasks. This is the central idea of deep neural networks [106], especially convolutional neural networks (CNNs) [107, 108], which are known for learning information of spatial correlation within data. Convolutional neural networks consist of stacked layers of linear convolutions with nonlinear activations to automatically extract multi-scale features or concepts from high-dimensional input [109], thus alleviating the hand-craft feature engineering, such as searching for the right set of basis functions or relying on experts knowledge. However, the general perspective for using deep neural networks [107, 110] in the context of surrogate modelling is, that physical problems in uncertainty quantification (UQ) are not big data problems, thus are not suitable for addressing them with deep learning approaches [3]. With Bayesian Deep learning [111, 112, 113, 114, 105] it is possible to express prediction uncertainty. The Bayesian network can quantify the predictive uncertainty by treating the network parameters as random variables and by performing Bayesian inference on those uncertain parameters, even when the training data set is small.

To sum up, by combining the concepts of a convolutional encoder-decoder network, Bayesian deep learning, as well as the recently proposed Stein Variational Gradient Descent [105] a Bayesian surrogate was learned in order to perform uncertainty quantification of an uniaxial tensile test of heterogeneous aortic tissue.

### 5.2.1 Data Description

The Rolf-Pissarczyk-Holzapfel model, introduced in Chapter 4.3 describes a Finite Element (FE) simulation of an uniaxial tensile test on an input tissue $\mathcal{X}$, in this case random fields of a spatially correlated degradation parameter distribution, and in return gives the symmetric Cauchy-Stress Tensor at all Gaussian integration points. From this tensor the component $\sigma_{33}$ is extracted and relabelled here as the outcome $\mathcal{Y}$. Thus, the FE simulation can be considered as the mapping from an input space to a certain output space $\mathcal{X} \rightarrow \mathcal{Y}$. In order to overcome computational expensiveness of the model when performing uncertainty quantification, a surrogate function $\mathbf{f}(\cdot)$ is trained, using a subset of available FE in-/output data $\mathcal{D} = \{\mathbf{x}, \mathbf{y}\} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N}$, where $\boldsymbol{x}_i$ is a random field input sample, determining the degradation parameter and $\boldsymbol{y}_i$ is the true FE solution.

Moreover $\boldsymbol{\theta}$ holds the model parameters, such as weights and biases of the neural network, and $N$ is the number of available training samples, based on the number of FE simulations.

The (hyper)parameters of the surrogate are adapted/trained in order to approximate the real FE simulation best, and the neural network surrogate reads $\mathbf{y} = \mathbf{f}(\mathbf{x}, \boldsymbol{\theta})$. Considering the input dimensions, the number of evaluated grid points in the FE Solver are determined by the Gaussian integration points and can be described via an irregular grid of size $H \times W \times D$ for height, width and depth respectively. The box is shown in Fig. 21.

For the input cube one random field realization, is used twice as stacked layers, such that the input has the dimension $\boldsymbol{x}_i \in \mathcal{R}^{H \times W \times D}$ and the outcome of the solver similarly reads $\boldsymbol{y}_i \in \mathcal{R}^{H \times W \times D}$. However, for training the Convolutional Neural Network the third dimension in the input was neglected, meaning that only the two dimensional input $\boldsymbol{x}_i \in \mathcal{R}^{H \times W}$ was mapped by the surrogate function onto a single component of the Cauchy-Stress Tensor which was located on the sample plane, such that $\boldsymbol{y}_i \in \mathcal{R}^{H \times W}$. The surrogate model was treated as an image-to-image regression problem with pixel-to-pixel wise predictions. Thus, the regression function maps $\mathcal{R}^{H \times W} \rightarrow \mathcal{R}^{H \times W}$.

A total a number of 10.000 FE solutions were available, from which 5000 were used as a training set, with 4200 training samples and 800 test samples. The remaining 5000 samples could then be used for Uncertainty Quantification, since the data was 'unseen' by the surrogate model, yet, the 'true' FE solution was available to compare it to the NN predictions. The data structure was distributed in order to, first, have enough samples for training the surrogate with a total of 400 stochastic dimensions, and, second, to be left with enough unseen data samples, such that a Uncertainty Quantification could be conducted with enough samples.

### 5.2.2 Network Architecture

In this subchapter the main terms considering neural network architecture are explained and the main structure is illustrated. To get a glimpse of the proposed algorithm the architecture principle is outlined in Fig. 22 and the utilized network parameters are given by Table 2. Very briefly, Fig. 22 shows the encoding path (upper line) which takes

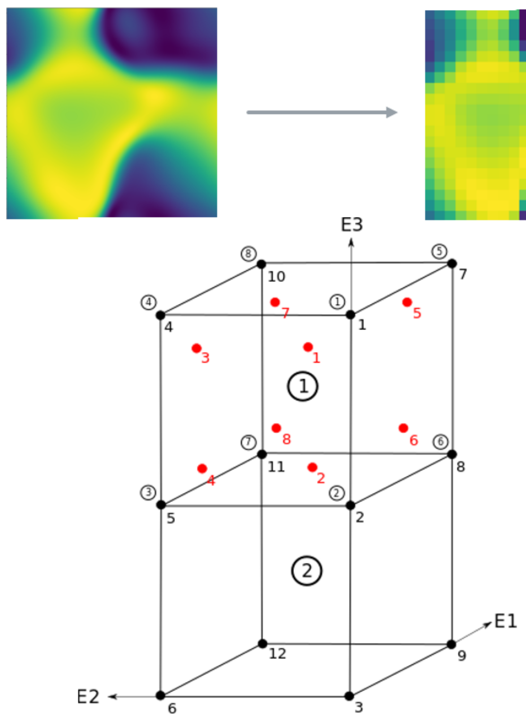


Figure 21: The images in the upper line show a high-resolution random field which is reduced to a lower-resolution image by taking only the Gaussian integration points used by the FE solver. The Gaussian integration points, indicated by red dots in the lower image, form a cube of size $(20 \times 20 \times 2)$ grid points. One box is labeled by a circled number and consists of 8 integration points, whereas the readout was performed from top to bottom. To obtain a 3-dimensional input, two identical low resolution images were stacked behind each other and used as an input in the FEM solver. The uniaxial tensile test could then perform a 3-dimensional calculation of aortic tissue based on the Rolf-Pissarczyk-Holzapfel model [2]. The coordinates were chosen, such that the stretch was applied in $\mathbf{E}_3$-direction. Moreover, the random field spans the $\mathbf{E}_2 \times \mathbf{E}_3$ plane and the $\mathbf{E}_1$-direction represents the wall depth.

random field realizations and feeds them through a convolutional layer. The extracted feature maps are handed to a number of dense blocks and encoding layers, as introduced in Fig. [23] and Fig. [24]. After the last dense block of transition layer the high level coarse feature maps are fed through dense layers, as outlined in dark green, and are subsequently fed into the decode path (lower line) of Fig. [22]. The decode path holds similar structure to the encoder, but with decoders instead. At the end of the last decode layer, predictions of the $\sigma_{33}$ output fields are made. The main architecture used for this work was initially proposed by [115] and afterwards modified by [3]. The initial algorithm by [115] is called DenseNet and aims to enhance information gradient flow through the network. Therefore any layer is connected to all subsequent layers, i.e. $x_l = h_l([x_{l-1}, x_{l-2}, \ldots, x_0])$. This means that if, e.g. an image has $K_0$ input channels (for a RGB image: $K_0 = 3$), each $l^{th}$ layer has a number of $K_0 + (l-1) \cdot K$ input feature maps. The total number of feature maps then grows linearly with every introduced layer to a total of $K_{\text{out}} = K_0 + L \cdot K$ output feature maps. This structure is embedded in a so called dense block. A dense block contains multiple densely connected layers whose input and output feature maps are the same size. Here, two design parameters

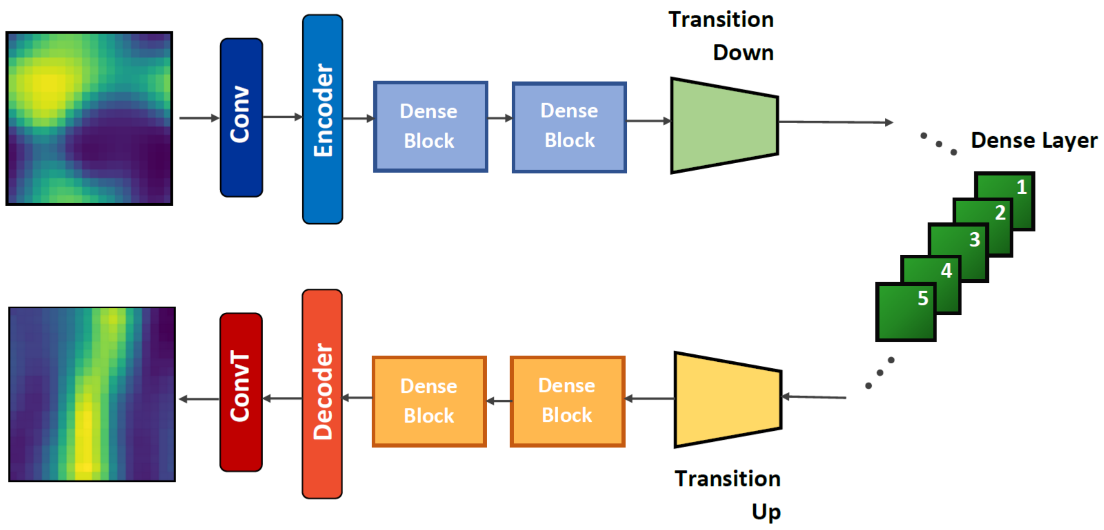


Figure 22: Network architecture used in this work. First, the input field of size $(20, 20)$ is 2D-convoluted with kernel size $k = (7, 7)$, stride $s = (2, 2)$ and padding $p = (3, 3)$. Afterwards, it enters the encoder with two Dense Blocks. Both are identical and consist of BatchNorm2D, ReLU and Conv2d, with $k = (3, 3)$, stride $s = (1, 1)$ and $p = (1, 1)$. Afterwards, is fed into a transition layer again holding BatchNorm2D, ReLU and Conv2d twice, with $k_1 = (1, 1)$, stride $s_1 = (1, 1)$, $p_1 = (0, 0)$ and $k_2 = (3, 3)$, stride $s_2 = (2, 2)$, $p_2 = (1, 1)$, respectively. Up until this point the Encoder shrinks the input down to size $(6, 6)$, see upper right handside. Then, the five dense layers, all consisting of BatchNorm2D, ReLU and Conv2d are used to arrive at the following Transition up layer, again with $k_1 = (1, 1)$, stride $s_1 = (1, 1)$, $p_1 = (0, 0)$ and $k_2 = (3, 3)$, stride $s_2 = (2, 2)$, $p_2 = (1, 1)$. What follows next are again two Dense Blocks of the same, yet inverted, construction. In the last step, called Sequential, BatchNorm2D, ReLU and Conv2d are used twice, but with ConvTranspose2d $k = (4, 4)$, stride $s = (2, 2)$, $p = (1, 1)$ in the last cycle. The outcome is the surrogate prediction of the mapped FE solution.

are introduced, namely $L$ which defines the **number of layers** within a dense block and $K$ which represents the **growth rate** and thus, defines the growth of input feature maps for each layer.

For image regression with encoder-decoder networks, down-sampling and up-sampling are required to change the size of feature maps, which makes concatenation of feature maps unfeasible. Thus, dense blocks and transition layers are introduced to overcome this issue [3].
In Fig. 23, a brief example of a dense block with parameters $K = 3, L = 3, K_0 = 3$ is outlined. Similarly to conventional CNNs, DenseNet includes **Batch Normalization** (Batch Norm) [116], Rectified Linear Unit (**ReLU**) [117] and convolution **Conv** or transposed convolution (**ConvT**) [118]. To reduce the number of feature maps between dense blocks, as well as the size of those, transition layers are used. More specifically, the encoding layer typically halves the size of feature maps, while the decoding layer doubles the feature map size. Both of the two layers reduce the number of feature maps [3], with an illustration shown in Fig. 24. Moreover, batch normalization layers

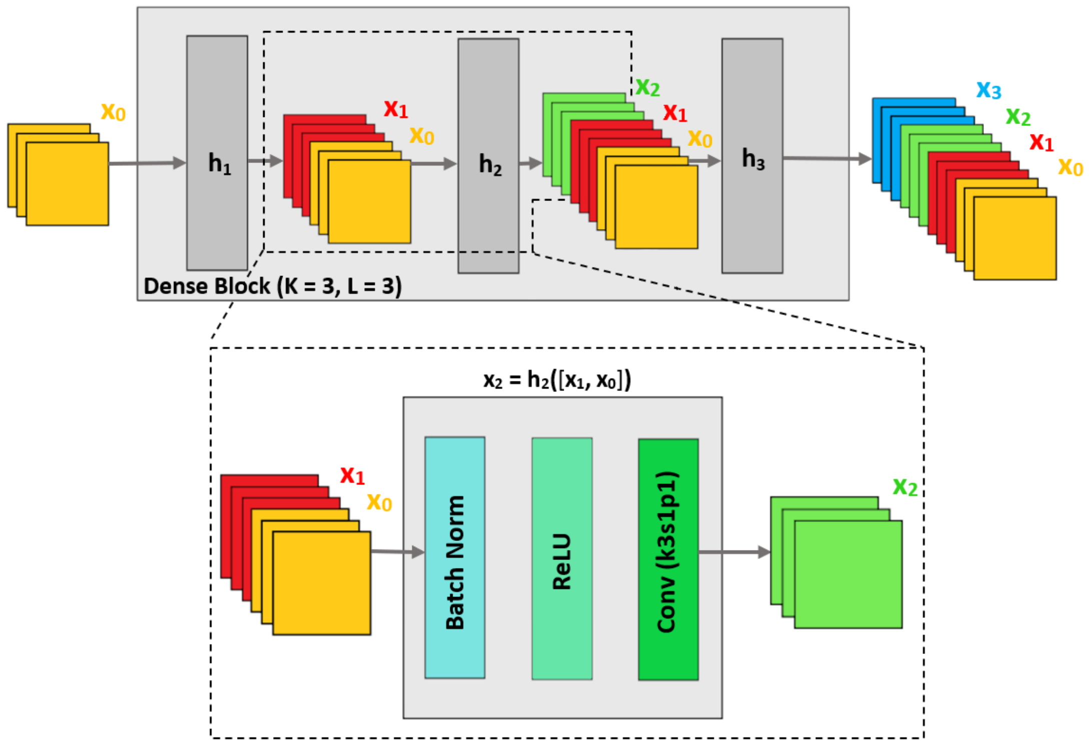
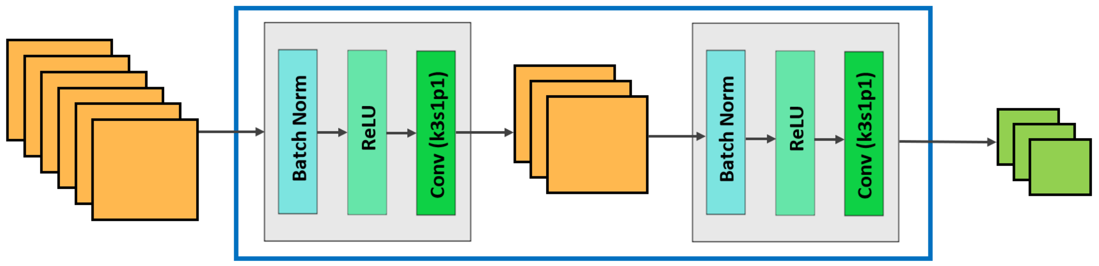
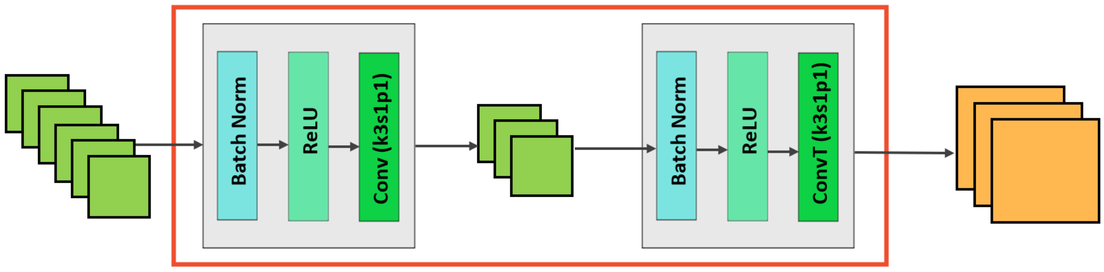


Figure 23: The upper image shows the structure of a dense block containing $L = 3$ layers called $h_1$, $h_2$, $h_3$ with growth rate $K = 3$. The input is given by $K_0 = 3$ channels, the output respectively by $K_{\text{out}} = 12$ feature maps. The lower image outlines the second layer $h_2$ of the dense block, where $x_2 = h_2([x_1, x_0])$ represents the output feature map. Notice that the input to the third layer is the concatenation of the output and input features of $h_2$, i.e. $[x_2, x_1, x_0]$. In addition, each layer consists of Batch Normalization [119](BatchNorm), Rectified Linear Unit [117](ReLU) and Convolution (Conv). In this case, the convolution kernel has size $k = 3$, stride $s = 1$ and zero padding $p = 1$, which keep the size of the feature maps the same as the input.

are used after each convolutional layer, since this can also be considered as an effective regularizer [119] and is (nowadays) commonly applied in deep neural networks in order to replace dropout [101]. As proposed in [120] fully-convolutional networks (FCN) are the extension of CNNs for pixel-to-pixel wise predictions, where FCN replace the fully connected layers of CNNs with convolutional layers. Moreover, up-sampling layers are added in the end to restore the input spatial resolution and skip connections between feature maps are included for the down- and up-sampling path, see [3]. An introduction of fully convolutional DenseNets can be found here [121]. This work heavily relies on the principles of [3], which proposed a very similar approach to DenseNet with FCNs, with the main difference of dropping the concatenation of feature maps between the encode paths and decode paths. This means, that while in [121] only the last feature map of the convolutional layer is fed into the transition layer, [3] propose to keep all feature maps, concatenate them before passing it to the transition layer. In addition, skipping of connections is avoided because of weak correspondence and no max-pooling in encoding layers was used, hence the compensation with a stride of 2. Furthermore, the authors designate their modification of a DenseNet with FCN the term **DenseED**. An illustrative example of a DenseED network is shown in Fig. 22. Table 2 shows the



(a) Encoder structure



(b) Decoder structure

Figure 24: Both the (a) encoding and (b) decoding layer contains two convolutions. In this case, the first convolution reduces the number of feature maps while keeping their size the same, by using a kernel with parameters $k = 1$, $s = 1$, $p = 0$, the second convolution changes the size of feature maps, but not their number using a kernel $k = 3$, $s = 2$, $p = 1$. The main difference between (a) and (b) is the type of second convolution, which is *Conv*, for down-sampling, and *ConvT*, for up-sampling respectively, whereby no pooling is used in the transition layer. The colors of feature maps used here are independent of feature maps shown in other figures. This figure was inspired by [3].

main architectural features used in this work.

### 5.2.3 Total Uncertainty

Bayesian Neural Networks (BNN), in contrast to deterministic ones, treat their parameters as random variables, since uncertainties can be introduced by a lack of training data. The BNN takes an input $\mathbf{x}$ as well as the set of random variables as parameters $\boldsymbol{\omega}$ and return $\mathbf{f}(\mathbf{x}, \boldsymbol{\omega})$ as the output. In addition, an additive noise $\mathbf{n}$ is commonly added, see [3] to model aleatoric uncertainty which can not be reduced elsewise, e.g. by having more observations. Thus, the probabilistic model reads

$$\mathbf{y} = \mathbf{f}(\mathbf{x}, \boldsymbol{\omega}) + \mathbf{n}. \tag{120}$$

In general, finding appropriate priors for probabilistic neural networks can be a challenging task, because of the difficult interpretability of the parameters. Moreover, any prior used has to satisfy low memory and computational costs. This motivates the use of sparse priors [121, 122]. In this model, analogously to the paper of Zhu et. al. (2018), a fully factorized Gaussian prior with zero mean and a Gamma distributed precision $\alpha$

Table 2: Key properties of Bayesian neural network

| **Network** | | **Data** | |
|---|---|---|---|
| **property** | **value** | **property** | **value** |
| Batch size | 350 | Total number | 10.000 |
| Dense Blocks | [2, 5, 2] | Training set | 4200 |
| Epochs | 500 | Test set | 800 |
| growth rate | 2 | validation set | 5000 |
| bottleneck size | 1 * growth rate | | |
| learning rate | 0.03 with cosine annealing | Input size [px] | 20 x 20 |
| number of single predictions for mean field predictions | 20 | Output size [px] | 20 x 20 |
| Dense Blocks Encoder | 2 | | |
| Dense Blocks Decoder | 2 | | |
| Dense Layers | 5 | | |
| Epochs | 500 | | |
| bottleneck size | 1 $\times$ growth rate | | |

on the parameters of $\boldsymbol{\omega}$ is assumed.

$$p(\boldsymbol{\omega} \mid \alpha) = \mathcal{N}(\boldsymbol{\omega} \mid \mu = 0, \alpha^{-1}\mathbf{I}), \qquad p(\alpha) = \text{Gamma}(\alpha \mid a_0, b_0) \tag{121}$$

The result is a Student's t-prior, which is known for its heavy tails and mass close to the origin. Regarding the additive noise, one can distinguish between (i) output wise noise (same for all output pixels), (ii) channel wise noise (same across each output channel) and (iii) pixel wise noise (distinct for every output pixel). Similarly to [3] only a homoscedastic centered Gaussian output noise $\mathbf{n} = \sigma\boldsymbol{\epsilon}$ with $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ was considered.

### 5.2.4 Stein Variational Gradient Descent (SVGD)

Finding a good approximation for any high dimensional model can be a daunting task, especially when the surrogate model needs to learn a high-dimensional posterior distribution with millions of random variables, while having only a very limited set of training data available. Thus, non-parametric method called Stein Variational Gradient Descent (SVGD) [72, 105] was adopted to replace standard gradient descent with the benefit of maintaining the efficiency of point methods. In short, Bayesian inference, by Stein Variational Gradient Descent, is an one line algorithm, where the gradient pushes the samples towards the high posterior mass region. Fig. 25 shows a schematic representation of the algorithm.

In a Bayesian Neural Networks with homoscedastic Gaussian noise one can summarize the hyperparameters as $\boldsymbol{\theta} = \{\boldsymbol{\omega}, \sigma, \alpha\}$ with the random variable $\boldsymbol{\omega}$, the output noise $\sigma$ and Gamma distributed precision $\alpha$, see Section 5.2.3. Then, for a prescribed probabilistic model one can specify a likelihood function $p(\mathbf{y} \mid \boldsymbol{\theta}, \mathbf{x})$ and a prior $p_0(\boldsymbol{\theta})$. The posterior distribution reads $p(\boldsymbol{\theta} \mid \mathcal{D})$, where $\mathcal{D}$ denote i.i.d. (independent and identically distributed) observations (= training data) summarized in $\mathcal{D} = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N}$.

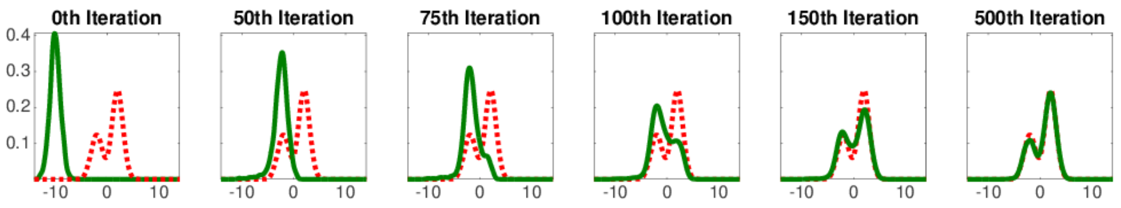


Figure 25: This toy example illustrates the SVGD algorithm with a 1D Gaussian mixture. The red dashed lines are the target density function and the solid green lines represent the densities of the particles at different iterations of the algorithm. It shows how the gradient pushes the samples towards the high posterior region. This image was taken from [105]. Even though the initial distribution had almost zero overlap with the target one, SVGD is capable of recovering the distribution after 500 iterations of training. In this case the number of particles used was $n = 100$.

Bayesian inference enables to determine the posterior distribution

$$p(\boldsymbol{\theta} \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})}{p(\mathcal{D})} \tag{122}$$

by approximating it with a variational distribution. The unnormalized posterior reads

$$\tilde{p}(\boldsymbol{\theta} \mid \mathcal{D}) = Z \cdot p(\mathcal{D} \mid \boldsymbol{\theta})\, p_0(\boldsymbol{\theta}) = \Pi_{i=1}^{N}\, p(\boldsymbol{y}_i \mid \ \boldsymbol{\theta}, \boldsymbol{x}_i)\, p_0(\boldsymbol{\theta}) \quad \text{with } Z \ = \int \tilde{p}(\boldsymbol{\theta} \mid \mathcal{D})\, d\boldsymbol{\theta} \tag{123}$$

being the normalization constant also called model evidence. This constant is usually computationally intractable, but can be ignored when optimizing the $\mathcal{KL}$ divergence [3]. The approximating variational distribution $q^*(\boldsymbol{\theta})$ lies in a restricted set of distributions with $q \in \mathcal{Q}$, such that by minimizing the Kullback-Leibler $\mathcal{KL}$ divergence between the two probabilities one obtains

$$q^*(\boldsymbol{\theta}) = \arg\min \mathcal{KL}\Big(q(\boldsymbol{\theta}) \mid \ p(\boldsymbol{\theta} \mid \mathcal{D})\Big) = \arg\min \mathbb{E}_q\Big[\log\big(q(\boldsymbol{\theta})\big) - \log\big(\tilde{p}(\boldsymbol{\theta} \mid \mathcal{D})\big) + \log\big(Z\big)\Big],$$

Even though most of the network architecture was kept through this analysis, minor modifications, with e.g. block numbers and the step-size scheduler were adapted. Zhu et. al. [3] p. 16, utilize a learning rate scheduler, which decreases by a factor 10 when the regression loss function remained on a plateau during training. Moreover, they applied RMSE for the regression loss function in SVGD, see Chapter 5.2.4. Here, the learning rate scheduler was replaced by cosine annealing and the regression loss function was replaced by smooth L1 loss **smooth L1**.

In principle, smooth L1 loss, which is also referred to as **Huber loss** [123] when parametrized with a delta [124], acts as a combination of L1 and L2 loss, meaning that it behaves like a L1-loss in case the absolute value of the argument is high, and like a L2-loss in case of low argument values. Thus, smooth L1 loss should lead to a faster convergence rate [125, 126]. Mathematically speaking, smooth L1 can be expressed as

$$\mathrm{L}_{1\text{ smooth}}(w, q) = \frac{1}{N} \sum_{i}^{N} l_{1\text{ smooth}}(w_i, q_i) \tag{124}$$

$$l_{1\text{ smooth}}(w_i, q_i) = \begin{cases} \frac{1}{2}(w_i - q_i)^2 & \text{for } |w_i - q_i| \leq \beta, \\ \beta(|w_i - q_i| - \frac{1}{2}\beta) & \text{otherwise.} \end{cases} \tag{125}$$

with $w, q$ being arbitrary shape parameters, $N$ being the total number of elements and $\beta$ representing a parameter of free choice. Having introduced the idea of $L_{1\text{smooth}}$, the loss function used in this work yields

$$L_{1\text{ smooth}}(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}) = \frac{1}{N}\sum_{i}^{N} l_{1\text{ smooth}}(\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{\theta}) \tag{126}$$

and analogous to Eq. (125) with parameter $\beta = 1$

$$l_{1\text{ smooth}} = \begin{cases} \frac{1}{2}\Big(\boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{\theta}) + \boldsymbol{n} - \boldsymbol{y}_i\Big)^2 & \text{for } |\boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{\theta}) + \boldsymbol{n} - \boldsymbol{y}_i| \leq 1, \\ \big|\boldsymbol{f}(\boldsymbol{x}_i, \boldsymbol{\theta}) + \boldsymbol{n} - \boldsymbol{y}_i\big| - \frac{1}{2} & \text{otherwise.} \end{cases} \tag{127}$$

For the learning rate scheduler **cosine annealing** was used. In this case, a cosine function is used as the learning rate annealing function, because the cosine function has shown to perform very promising compared to alternatives like simple linear annealing as explained in [127]. From the official website [128] follows the description that $\eta_{max}$ is set to the initial learning rate, $\eta_{min}$ is the minimum learning rate, $T_{cur}$ is the number of epochs since the last restart and $T_{max}$ is the maximum number of iterations, which yields

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})\left(1 + \cos\left(\frac{T_{cur}}{T_{max}}\pi\right)\right) \tag{128}$$

It is also a common choice to combine cosine annealing with **warm restarts**, as proposed by [129], however in this work we stick to the simple case.

For a more detailed explanation the reader is referred to [105].

# 6 Results and Discussion

## 6.1 Numerical Implementation

The investigated physical system describes a degradation parameter of elastic fibres of an aortic tissue within an uniaxial tensile test as modeled in the recently published Rolf-Pissarczyk-Holzapfel model [2]. The underlying distribution of the degradation within the fibers was assumed to be uniform and was modelled as uniform random fields, see Chapter 3.5. Therefore, Gaussian random fields were sampled via FFT with size $(2048 \times 2048)$ pixel and were mapped onto Uniform random fields. Two identical layers of this high resolution grid were stacked behind each other, in order to obtain a three dimensional mesh. In order to keep computational run time of the Finite Element Solver feasible, only the Gaussian integration points of the FE Simulation were selected from the grid, as shown in Fig. 18. In the next step, the solution of the uniaxial tensile test, namely the elements of the Cauchy-Stress tensor $\boldsymbol{\sigma}$ were investigated, see Chapter 4.3. The elements of the displacement vector $\vec{\boldsymbol{u}} = \left(\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3\right)^T$ of the uniaxial deformation is shown in Fig. 26. When analysing the Cauchy-stress elements, the difference of almost two orders of magnitude were reason enough to neglect all tensor elements other than $\sigma_{33}$. For a comparison of Cauchy-stress tensor elements see Fig. 27.

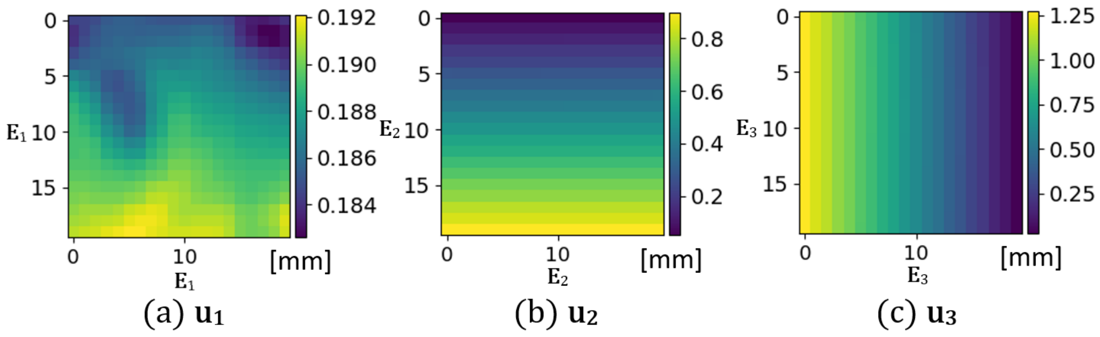


Figure 26: Coordinate displacement done by uniaxial tensile test. The observed tissue was stretched in $\mathbf{E}_3$-direction, which is outlined in subfigure (c). In contrast, subfigures (a) and (b) have a much smaller distortion scale and represent axes $\mathbf{E}_1$ and $\mathbf{E}_2$, respectively.

Attempts to also learn all tensor elements by one network all together were made, but only resulted in noise predictions rather than meaningful outcomes. Thus, a complete Cauchy-Tensor prediction with possibly individual neural networks stays a task for future projects. Fig. 28 outlines the performance of the Bayesian network. In subfigure (a), the input random field is displayed and figure (b) shows the FEM uniaxial tensile output. The network prediction is plotted above label (d) and the absolute difference between FEM output and NN prediction can be seen in (c). From comparing the true output and its network prediction it is visible that the network is not able to predict small scale fluctuations within the data. Therefore the network prediction looks rather smooth compared to the true simulation field. Those fluctuations may be assumed as noise within the data by the model, and are in fact responsible for the main network prediction uncertainty, as can bee seen from the absolute difference in (c) and predicted standard deviation in subfigure (e). Even though at first sight, the network prediction seems to

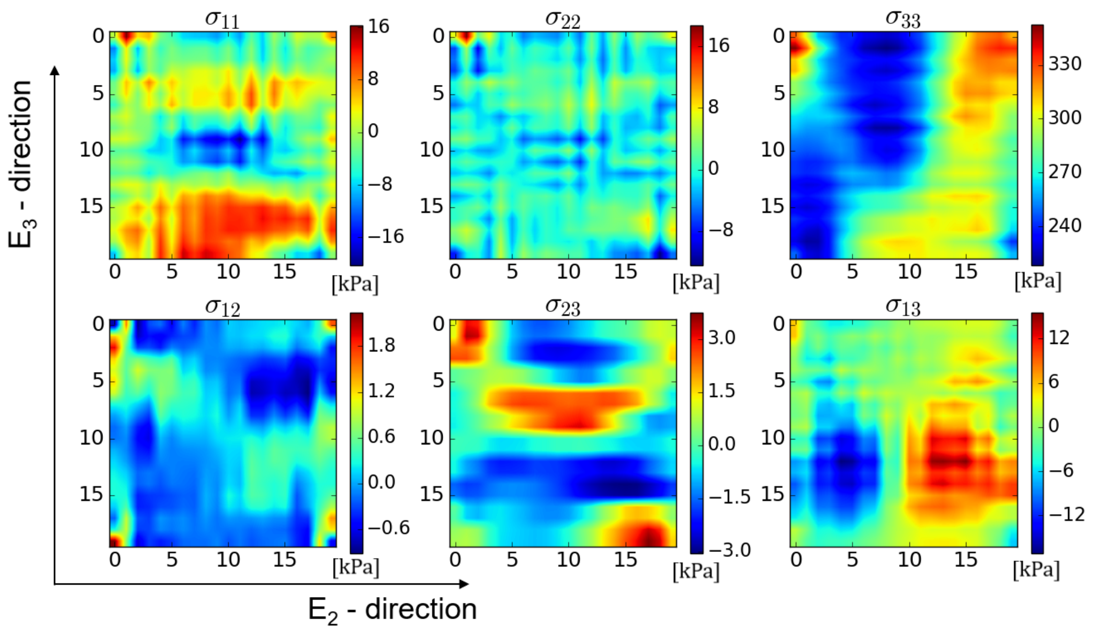


Figure 27: Comparison of Cauchy-Stress tensor of one sample output in all different plane directions.

capture the output well, it is the minor fluctuations which in the end cause relative errors up to 20%.

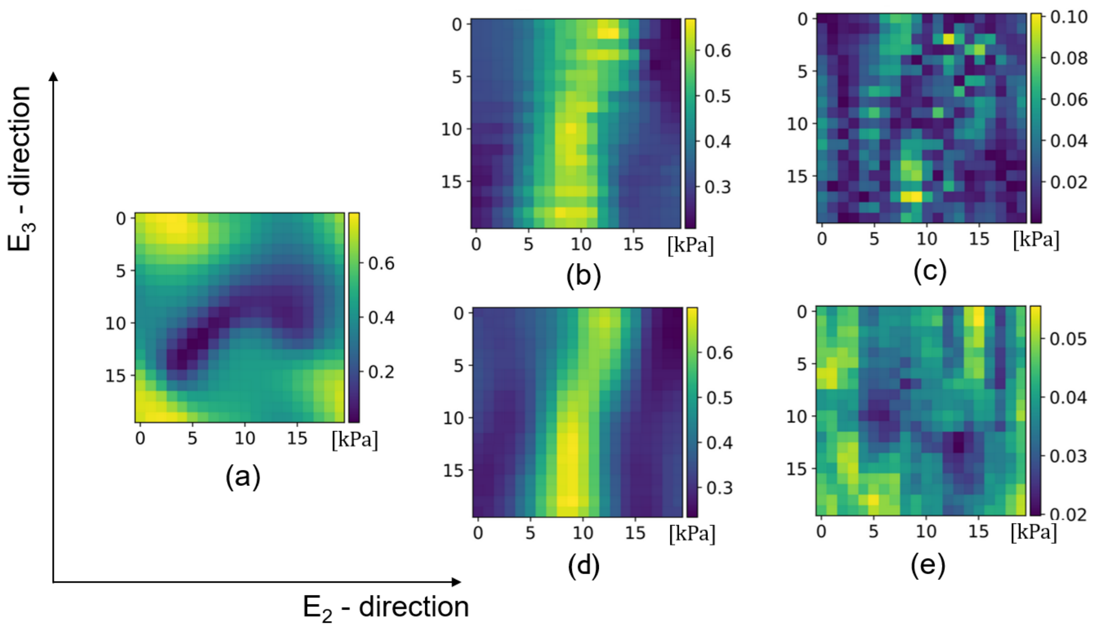


Figure 28: Trained surrogate model. A random field input $\boldsymbol{x}_i$ at index $i$, shown in (a), is mapped onto the target FEM solution $\boldsymbol{y}_i$ in (b) with a network mean prediction $\boldsymbol{\mu}_{pred}$ in (d). The absolute difference between network prediction and true solution is depicted in (c) as absolute difference. In addition, the standard deviation of the prediction is shown in (e).

Three more samples of FEM output and network predictions are outlined in Fig. 29, where the true FEM data lies on the upper left side, the upper right hand side shows the predicted output, the input random field is shown in the lower left image and on the lower right one there is the absolute difference between true output and network prediction. From those predictions it is again visible that the true FEM solution shows much more variations, thus looks less 'smooth' compared to the neural network prediction. This could have various reasons. Firstly, it could either result from a rather poor network architecture, which is incapable of learning small changes in the model, or secondly, it could be, that when the mean prediction is computed from a set of single predictions, as in this model 20 single predictions, that such local changes are smoothened as an inherent feature of Bayesian NNs. On the other side, from a physical model perspective, it can also very well be that those local fluctuations stem from fiber degradations, which arise independently in the model, due to exclusion of fibers above the exclusion angle. Since these excluded fibers stem from randomly arranged fibres the network might, independent on its architecture assume those distortions to be noise. Beside that, a smaller kernel window within the Dense Layers might also improve the results.

In order to investigate the individual predictions and their mean field in more detail, a comparison between the true field, predicted mean field, their absolute difference and three single predictions is shown in Fig. 30. From that, one can see that even if the single predictions are less smooth compared to the mean field, the individual predictions yet don't show distortions in $\mathbf{E}_2$-direction, as in the FEM solution.

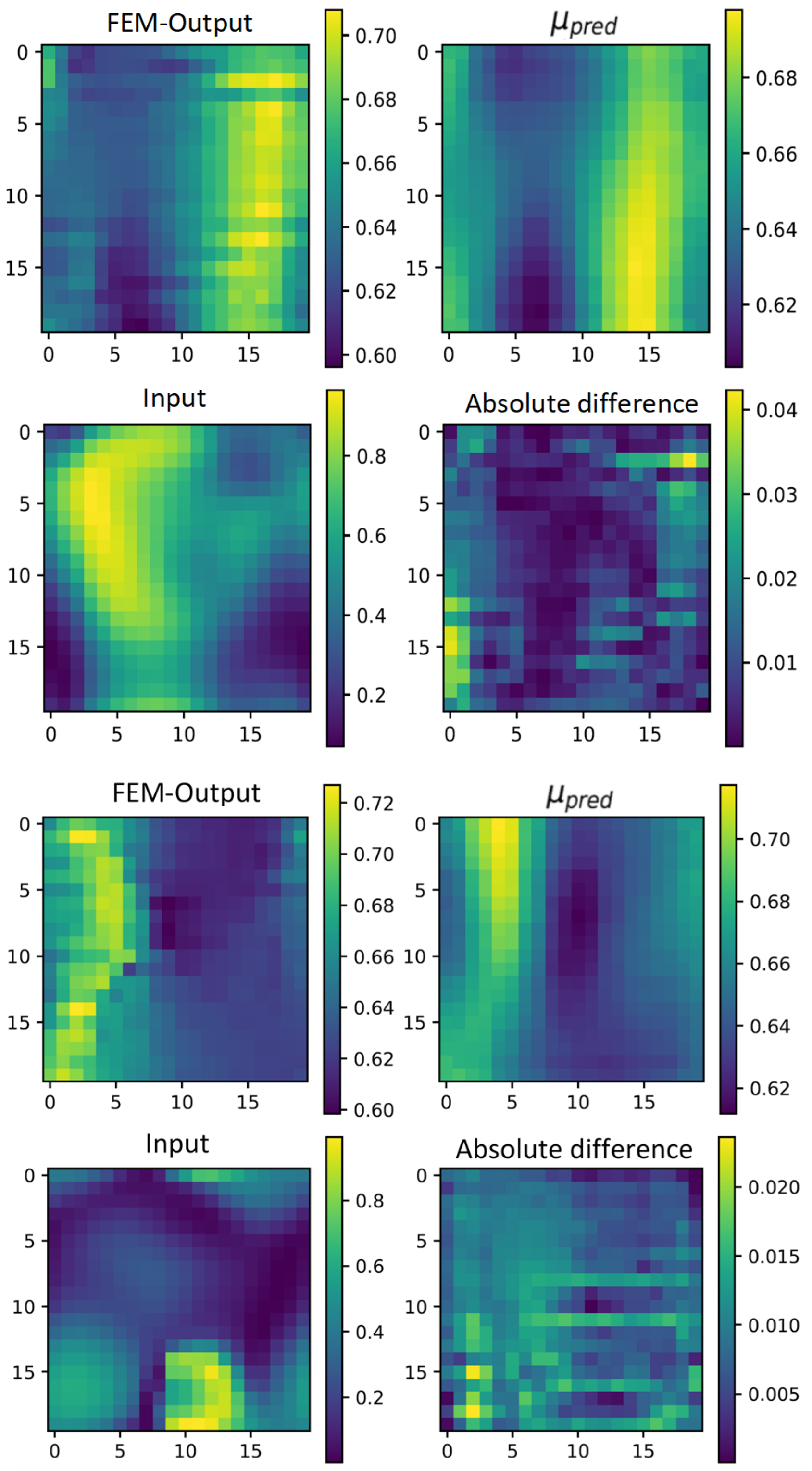


Figure 29: Two different samples of random field Input (lower left), FEM-Output (upper left), $\boldsymbol{\mu}_{pred}$ (upper right) and the absolute difference of true FEM solution and mean prediction (lower right). The colorbar indicates the value range in [kPa].

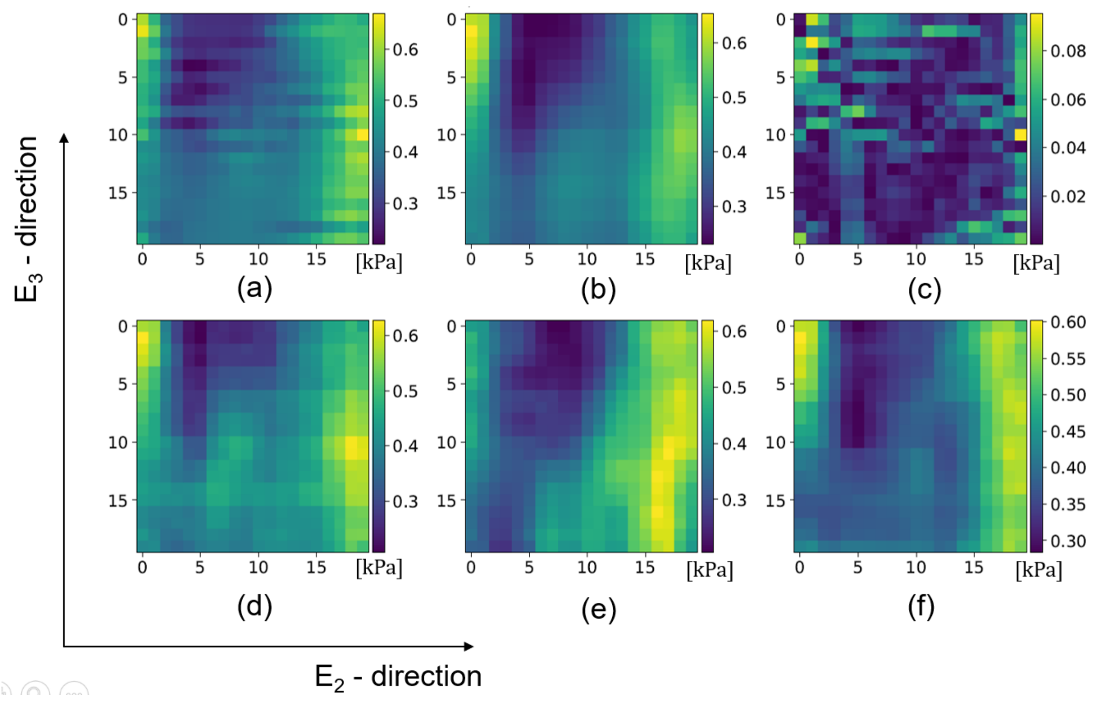


Figure 30: BNN - principle. The true target (a) is learned by the surrogate function and 20 individual predictions are made, whith mean field $\boldsymbol{\mu}_{pred}$, see (b), is returned by the network. Fig. (c) shows the absolute difference of (a) and (b). In the lower row, three out of 20 single predictions are depicted to compare the results. Fig. (b) looks rather smooth, which comes from averaging over all single predictions. It seems that by looking at the individual predictions (d) - (f), which are samples drawn from the learned distribution, that the surrogate is incapable of learning local fluctuations of the stress tensor component $\sigma_{33}$.

## 6.2 Uncertainty Quantification of Rolf-Pissarczyk-Holzapfel model

This Section contains the centerpiece of this thesis. After setting up a surrogate model, training and adapting hyperparameters, as well as architecture of it, this Section concentrates on quantifying the uncertainty of the uniaxial tensile test surrogate model. For the surrogate, a Bayesian neural network was trained and its hyperparamters as well as the architecture adapted from [3]. Therefore, random fields, which were introduced in Section 3, describing degraded elastic fibers of a physical uniaxial tensile test, as proposed by Rolf-Pissarczyk-Holzapfel Section 4.3, were sampled from a uniform and spatially correlated distribution and used as input of a Finite Element Solver, performing an uniaxial tensile test of aortic tissue. Then, by taking $\sigma_{33}$, the principal Cauchy-stress-tensor component as the quantity of interest, a Bayesian Autoencoder was trained to approximate the mapping of random fields, as an input, onto the stress-tensor-field-component, as an output. Results of the network predictions are outlined in Section 6.1. To quantify uncertainty, it is useful to compare surrogate predictions at certain locations to the true solution. Figures 31 - 33 outline the predicted probability of the $\sigma_{33}$-stress-tensor component at three different locations to the true tensor components. The sampled locations on the two-dimensional grid were chosen analogously to Zhu et. al. (2018), reading $(1.5, 4.5)$ for Fig. 31, $(12.5, 14.5)$ in Fig. 32 and finally $(18.5, 15.5)$ for Fig. 33.

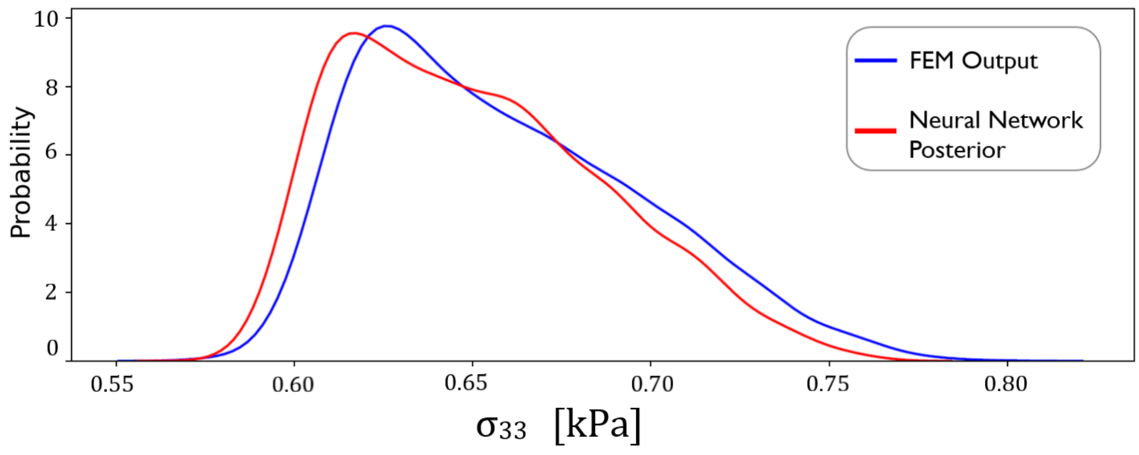


Figure 31: Posterior distribution at position $(1.5, 4.5)$.

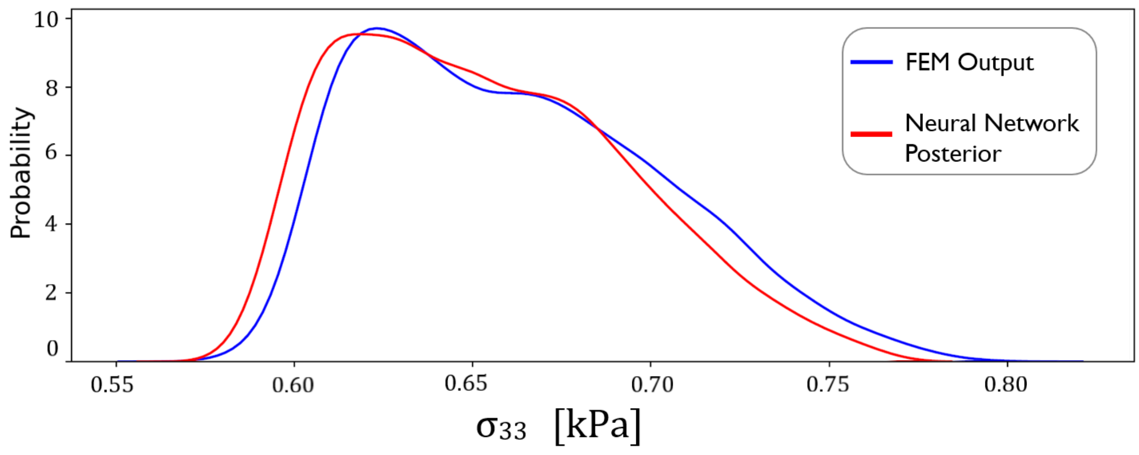


Figure 32: Posterior distribution at position $(12.5, 14.5)$.

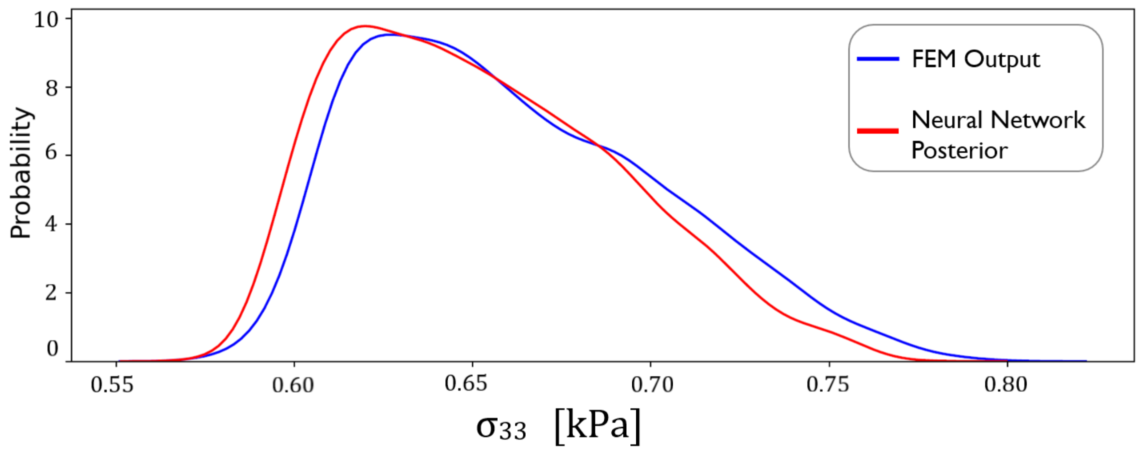


Figure 33: Posterior distribution at position $(18.5, 15.5)$.

From the three posterior distributions, one can see that the surrogate model is able to capture and predict the probability distribution at those three different locations quite well. It is notable that more training data leads to a more valuable prediction, simply because a greater sample size means a greater variety, which leads to a better approximation of the underlying distribution by the BNN. However, this only holds if the initial training set is large enough. Too little data during training might not fully depict the distribution and its variance, with the consequence of poor prediction accuracy.

In addition to the FEM and NN distribution, a direct comparison of local stresses between the single predictions within the Bayesian network, their mean values and the true FEM solution were considered. Therefore, figure 34 (a) shows the three locations at which the FEM solution was compared to the posterior predictions of the Bayesian neural network. Figure 34 (b) provides an intuitive outline of how the histogram of local stress distribution in Fig. 35 was obtained. After training with the first 5000 samples, predictions of the unseen $\sigma_{33}$-stress-components were made and their frequency was plotted in a histogram.

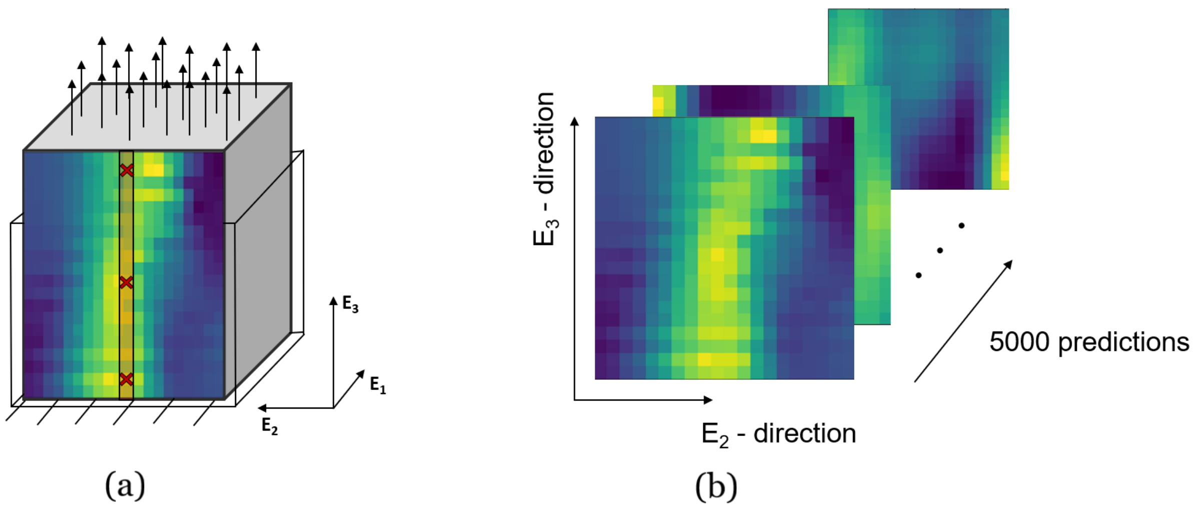


Figure 34: In subfigure (a) the locations of the evaluated posterior distribution of the Cauchy-Stress tensor in tensile direction are indicated with red ticks. The following figures 31 - 33, then show all individual cases of the red markers. Fig. (b) illustrates how posterior predictions of the Bayesian neural network were obtained. First, one location is set, then 5000 predictions of the unseen data are evaluated by the neural network and its posterior distributions are plotted, as outlined in fig. 31 - 33.

The single predictions in Fig. 35 are in fact the histogram over all individual posterior predictions made by the neural network. Because of the Bayesian architecture, the network is trained to learn an underlying distribution of the tensor element $\sigma_{33}$ of the uniaxial tensile test, such that one can sample from this distribution. Those extracted samples are then used to calculate the statistics of the network, meaning its mean and variance. Moreover, it would be possible to calculate higher momenta, which in this case was not necessary, but is generally possible. In this architecture 20 single predictions at each location of each field were made to characterize the statistics. Further investigations could be the study of influence of the number of single predictions for the network performance.

From Fig. 35 one can see, that the posterior of single predictions, in this case 20-times

5000 individual predictions (blue) approximate the underlying FEM Output (red) not as good, as when taking the mean of all 20-predictions at every location and plot their histogram over all 5000 samples (orange).

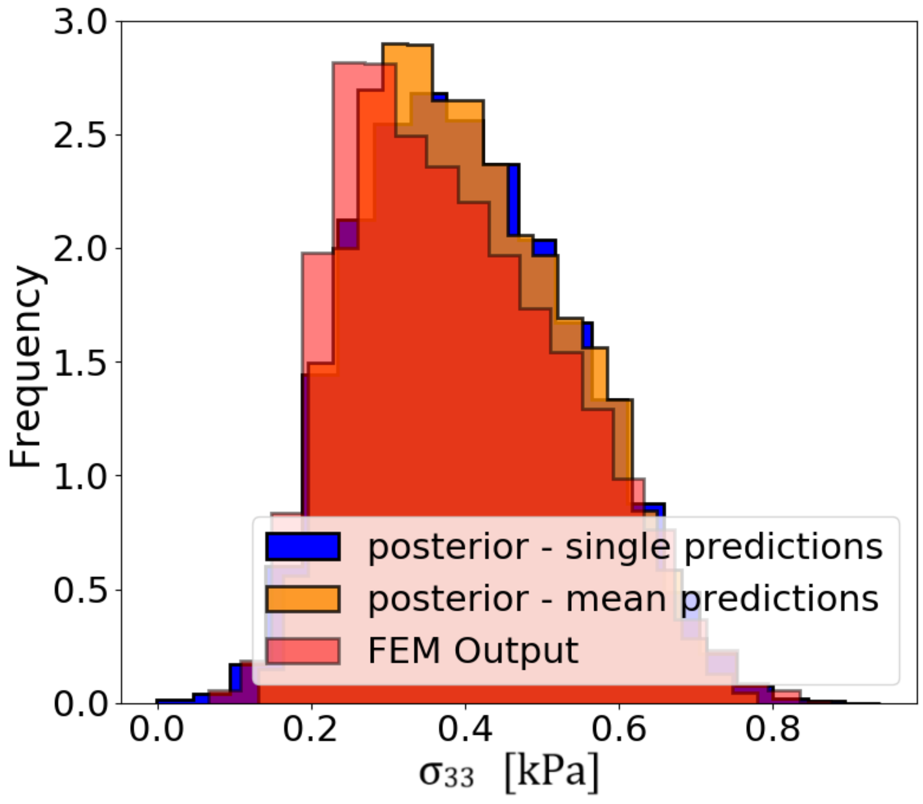


Figure 35: Histogram of local stress at position of the most uppest red cross in 34. This figure shows the posterior of all single predictions (blue), their mean prediction $\boldsymbol{\mu}_{pred}$ in orange, as well as the true target histogram in red.

In the next figures, one mean field prediction of one of the 5000 samples was analysed. In detail, a stripe along the middle of the mean field prediction was cut out, as shown by the grey shaded area in Fig. 36, and was plotted along $\mathbf{E}_2$-direction as x-axis and $\mathbf{E}_3$-direction for the y-axis, see Fig. 37. The cyan colored dots, represent the 20-single-predictions at one location, over which the network takes its mean, which is orange. The standard deviation at each predicted location is marked in light blue and the true FEM solution in dark blue. The benefit from using a BNN now is, that even mean and standard variation predictions can be equipped with error bands.

It is legitimate to ask, why the prediction uncertainty does not always reach out far enough to capture all FEM solutions within the error bands, as shown in Fig. 37. Therefore, at first sight, it may seem that the neural network is incapable of predicting the error bands correctly, but one step at a time.

The total uncertainty of the model consists of

$$\Delta_{\text{Total}} = \Delta_{\text{Model}} + \Delta_{\text{Data}} \tag{129}$$

where $\Delta_{\text{Data}}$ is also called **aleatoric** uncertainty and stems from e.g. obtaining dif-

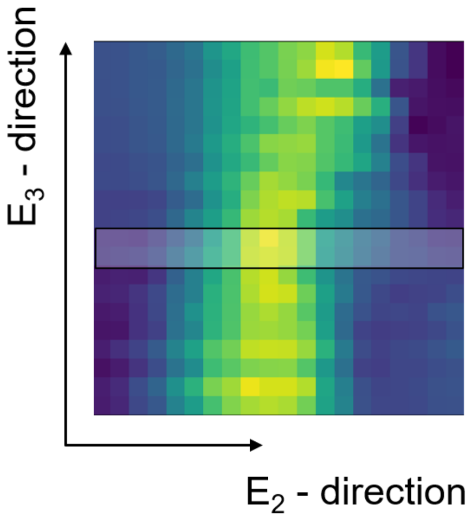


Figure 36: Explanation of the extracted data in Fig. 37. The data was taken from predictions along a line starting from location (1,10) up to (20,10) are outlined, which represents a horizontal cut through the Cauchy-Stress-Tensor distribution along in $\sigma_{33}$ direction.

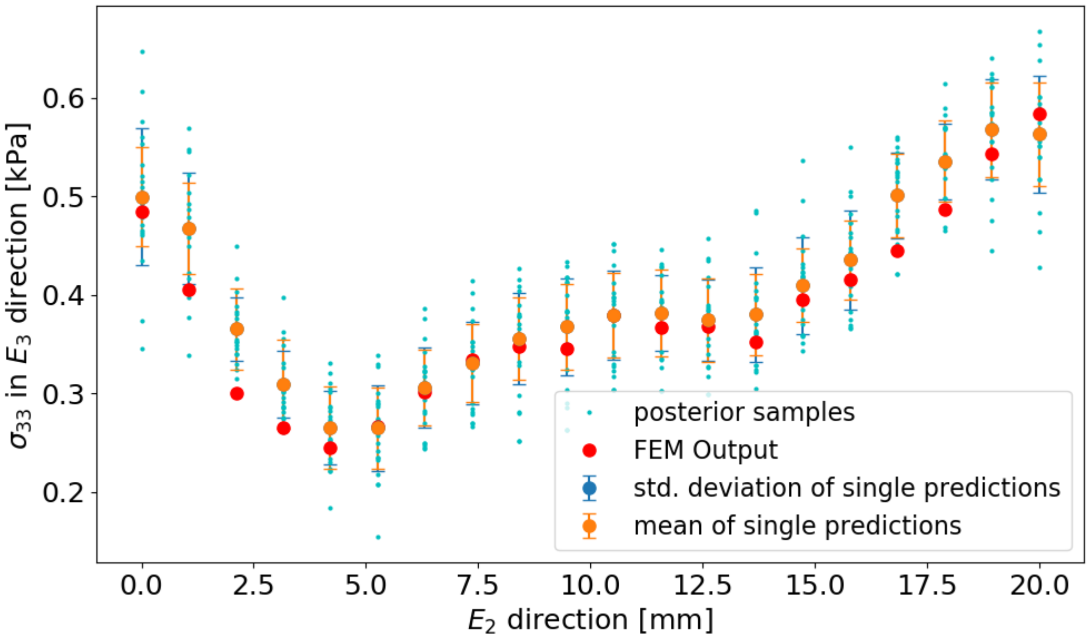


Figure 37: Predictions along a line starting from location (1,10) up to (20,10), including error bands. The single posterior predictions are marked in cyan, their mean, including error bands in orange, the standard deviation of prediction is marked in blue, including its error bands. The true output solution is marked in red.

ferent measurement results at the same initial conditions, whereas $\Delta_{\text{Model}}$ consists of **experimental**, **parameter**, **algorithmic**, **structural** uncertainties, and others.

Because the data was simulated the experimental uncertainty is assumed to be zero and only the additional noise of the input fields was considered. For the neural network uncertainty the view gets slightly more complex. Here, the uncertainty is assumed to be neglectable, also the algorithmic one. Uncertainty of the neural network mainly stems from approximating that the network truly finds the optimal hyperparameters within its training, which may not be completely true if the network settles at a good, yet local minimum.

Here, the hyperparameter $\boldsymbol{\theta} = (\boldsymbol{\omega}, \sigma, \alpha)$ contains the weights vector $\boldsymbol{\omega}$, homoscedastic Gaussian noise $\sigma$ and the precision of the distribution of weights $\alpha$ in the neural network. $\mathcal{D}$ represents the data of the fields, $\mathbf{x}$ the unseen input fields for the prediction and $\mathbf{y}$ the prediction of the neural network on unseen data, such that the neural network prediction for $\mathbf{y} = \boldsymbol{f}(\mathbf{x} \mid \boldsymbol{\theta})$ reads

$$p(\mathbf{y} \mid \mathbf{x}, \mathcal{D}) = \int p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}) \cdot p(\boldsymbol{\theta} \mid \mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} \tag{130}$$

whereby brute force calculation is impossible, but with NN one aims to find the best hyperparameters. Those resemble the minimum of the loss function, or respectively, the maximum aposteriori. Here, it is assumed that the optimal hyperparameters are found by the network as $\boldsymbol{\theta}^*$, with no additional error considered in the uncertainty calculation. This can be assumed to be true if most of the probability mass lies densely around those optimal parameters. Thus,

$$p(\boldsymbol{\theta}^* \mid \mathcal{D}) \approx \delta(\boldsymbol{\theta} - \boldsymbol{\theta}^*) \tag{131}$$

$$p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}^*) \approx p(\mathbf{y} \mid \mathbf{x}, \mathcal{D}). \tag{132}$$

The total uncertainty in this model stems from

- data: firstly, the added noise when simulating the input random fields. Secondly, the Gauss-Konrod integration points in the unit cube of the uniaxial tensile test are not equidistantly spread. This error damage was reduced by sampling high resolution images of size $2048 \times 2048$ which were reduced to the closest GIP yielding $20 \times 20$ pixel images. Thirdly, the FEM calculation is only a model of the aorta and no stress or strain equation may describe the real world completely, giving rise to model errors. In addition, the decimal digits when sampling and evaulating the FE solver may cause additional uncertainties, which are not considered. Moreover, within the constitutive model the discretized number of collagen fibre densities also plays a major role. Here, a finer grid of discretization triangles, see [2] p. 6 would lead to a more accurate model, thus, smaller errors. Having said this, due to limited capacities only the random field noise was included when propagating the uncertainties in this work. Furthermore, numerical errors also are assumed to be zero in this case.

- neural network: firstly, a there is model uncertainty, by using the surrogate. Moreover, even if the NN mimics the FEM quite well, one needs to consider that it might still not have the optimal hyperparameter. However, this uncertainty is assumed to be zero, since Eq. (131). Secondly, there is a prediction uncertainty, because the output is assumed to be Gaussian, when predicting the first and second moment. On top of that, even the predicted mean and standard deviation have error bands, which are outlined in Fig. 37.

Another interesting quantity is the rupture probability of the material. Therefore, the inverse cumulative distribution of the Cauchy-stress tensor in tensile direction was plotted in Fig. 38, to demonstrate, that, when given a physiological value, e.g. a critical stress value $\sigma_{\text{crit}}$, above which the aortic tissue is highly likely to fail, the surrogate model can be used to support FEM calculations quite accurately. However, this does not mean

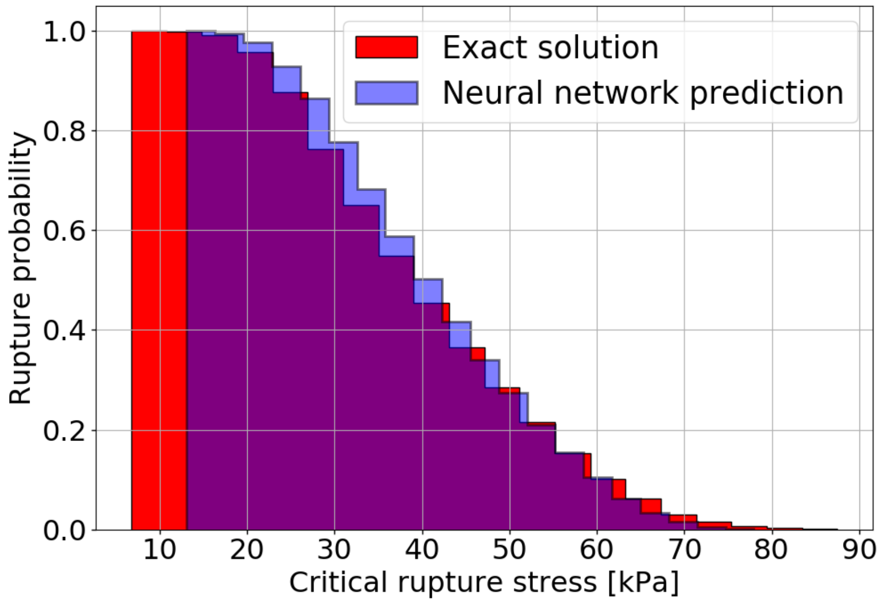


Figure 38: Inverse cumulative of stress histogram at location (1,10). This figure shows, that the surrogate is capable of predicting a rupture probability when given a critical value. The red bars belong to the true Finite Element solution, whereas the Neural Network prediction is drawn in blue.

that FEM solvers might be outdated in the future, but that the user can benefit from combining reliable, well validated and trustworthy methods, like FEM, with modern approaches of neural networks, in order to boost the efficiency and computational run time without losing too much precision.

Nevertheless, even if Neural Networks show promising results in more and more fields, it is up to the user to asses first the appropriateness use of a Neural Network, and if so, to adapt existing tools to a model dependent task accurately and most importantly to quantify and compare the introduced uncertainties of the neural network. Here, one might run into major issues of NNs, which tackles the uncertainty of the network itself. It is a vivid area of research which concerns this problem [130, 131, 132]. Moreover, the approach of using predictions of a neural network may sometimes work better, sometimes worse, without any, at first sight, obvious reason. To demonstrate this, a comparison of 'good' vs. 'failed' in quality predicted results in shown in Fig. 39. One reason leading to failed predictions could be, that the training sample set was too small, or, that the surrogate model obtained similar input, which had to be mapped onto output with larger variation, which in return gives a larger error. Furthermore, local distortions as discussed in 30, may contribute as well. Therefore it is recommended to also look at the quality of the predictions, i.e. by plotting the reliability diagram, as shown in figure 40. Even if the optimal neural network prediction analysis should be close to the diagonal line, which is not always the case, this does not automatically mean a bad result in general. One needs to keep in mind the high complexity of the model. Moreover, for training the

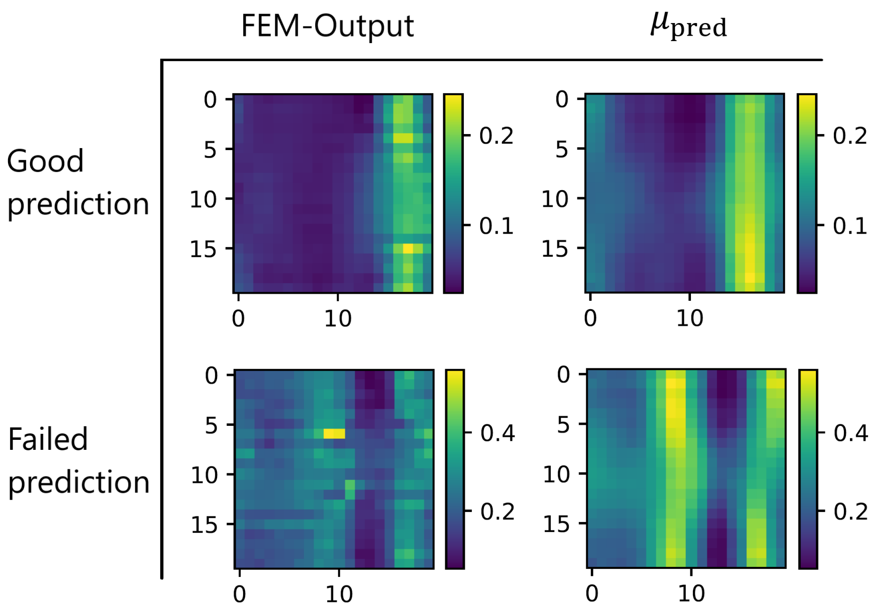


Figure 39: Comparison of one ’trustworthy’ vs. a ’failed’ prediction result. In the left column the FEM solutions are outlined and in the right column the mean prediction of the neural network.

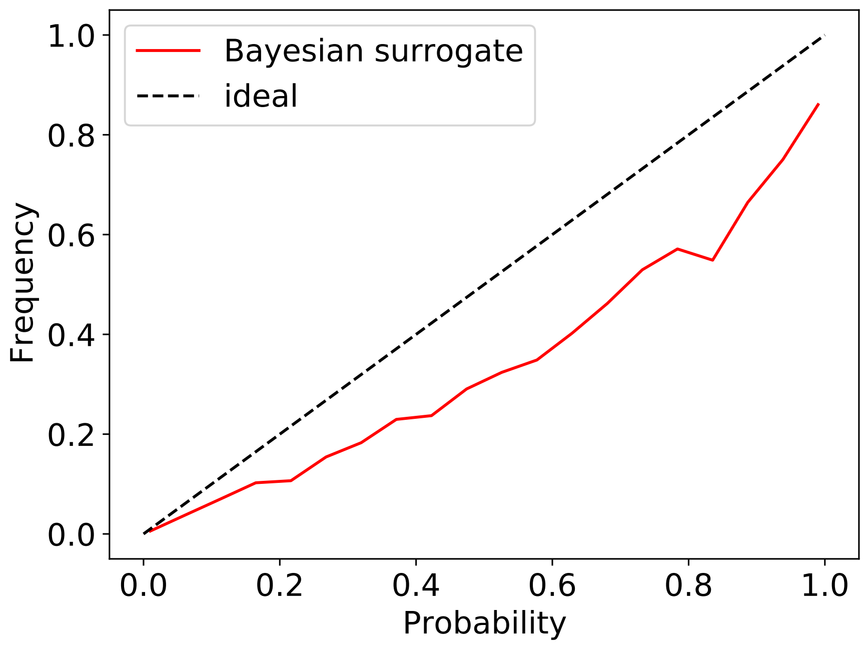


Figure 40: Reliability diagram of the surrogate model. The ideal progression is plotted as dotted black line and the Bayesian surrogate in red. The model frequency was evaluated at 30 points with a maximum of 86.0%.

network a total of 5000 training samples were used. As discussed in Zhu (2018), remark 4, with a greater set of training samples also comes a greater variety of mapping options. This means, that a larger set of highly variable input, which maps onto almost the same output, will make the network automatically less confident with its prediction than if it would be only given half or even less training data, i.e. making the mapping more distinguishable. In that case however, the neural network could get overly confident with its predictions by seeing too little data. Therefore, a rather linear correlation in Fig. 40 is favourable, but one also needs to take the size of training and evaluation data as well as complexity of the model into considerations.

## 6.3 Predicting the Principal Absolute Value with Bayesian Autoencoder Surrogate

This final subsection presents the surrogate predictions for a slightly different objective, namely the **Principal Absolute Value**, short **PAV**. The principal absolute value $\hat{\sigma}_{\text{total}_{loc}}$ can be calculated by taking the sum of the squared Cauchy-Stress-tensor elements for each location as

$$||\hat{\sigma}_{\text{total}_{loc}}||^2 = \sum_{i,j} \sigma_{ij}^2 \quad \text{with } i,j \in \{1,2,3\}. \tag{133}$$

which are the elements taken from the symmetric Cauchy-stress-tensor

$$\hat{\sigma}_{\text{Cauchy}} = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{12} & \sigma_{22} & \sigma_{23} \\ \sigma_{13} & \sigma_{23} & \sigma_{33} \end{pmatrix}.$$

The summation over space of all local PAV values multiplied with the strain $\epsilon$ gives an information about the deformation energy $E$ contained within a material reading

$$E = \frac{1}{2} \int \text{tr}\left(\hat{\sigma}_{\text{Cauchy}}\, \epsilon\right)\, \text{d}V \;\propto \int ||\hat{\sigma}_{\text{total}_{loc}}||^2\, \text{d}V. \tag{134}$$

The existing network architecture was adopted to learn a surrogate model, which is able to predict not only single stress-tensor components, but also the principal absolute value and its maximum. Further investigations strongly motivated that approximating the Cauchy-stress-tensor in tensile direction only, i.e. $\sigma_{33}$, the surrogate is capable to result almost the same as the true outcome when taking the complete Cauchy stress tensor. Hence, the approximation reads

$$\tilde{\hat{\sigma}}_{\text{Cauchy}} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \sigma_{33} \end{pmatrix}.$$

These additionally introduced uncertainties were neglected, because firstly, the stress-tensor-components span almost two orders of magnitude, see Fig. 27 and secondly, because learning different channels with different noises by the same neural network is counterproductive and leads to even greater noise assumptions by the model. One way around could be to train each output channel individually, for which one needs to adapt the architecture individually. This could be an interesting, yet challenging task for future projects. In this case, the simplified model was assumed and the comparison of the total PAV of the FEM solver and the predicted reduced PAV of the neural network are outlined in Fig. 41 at location $(1, 10)$.

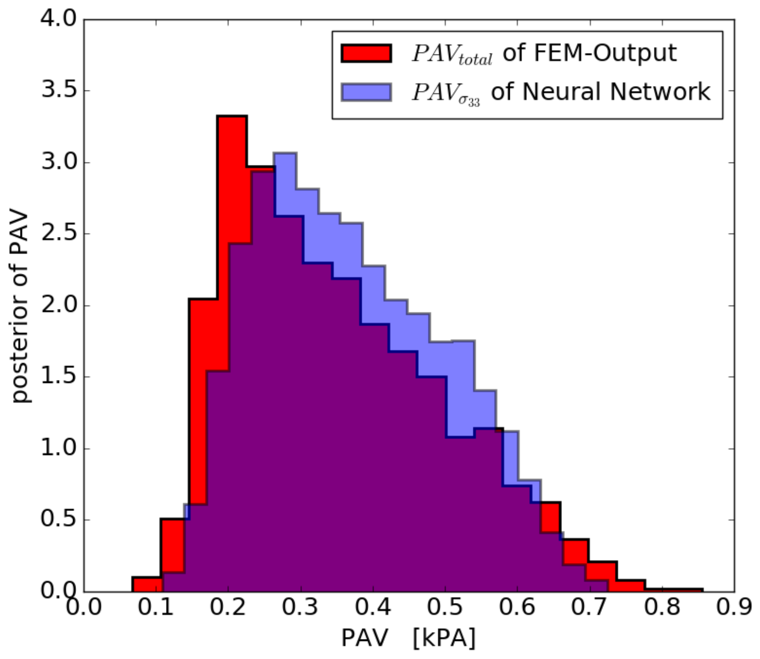


Figure 41: PAV of the FEM solver in comparison to surrogate prediction of reduced PAV. The PAV was calculated by taking the sum of the square elements of once the full Cauchy-Stress-tensor for the FEM solution and once the reduced Cauchy-Stress-tensor for the surrogate model.

The reliability diagram of of the uncertainty analysis is shown in 42 as well as three posterior samples at locations $(1.5, 4.5)$ for Fig. 43, $(12.5, 14.5)$ in Fig. 44 and $(18.5, 15.5)$ for Fig. 45.

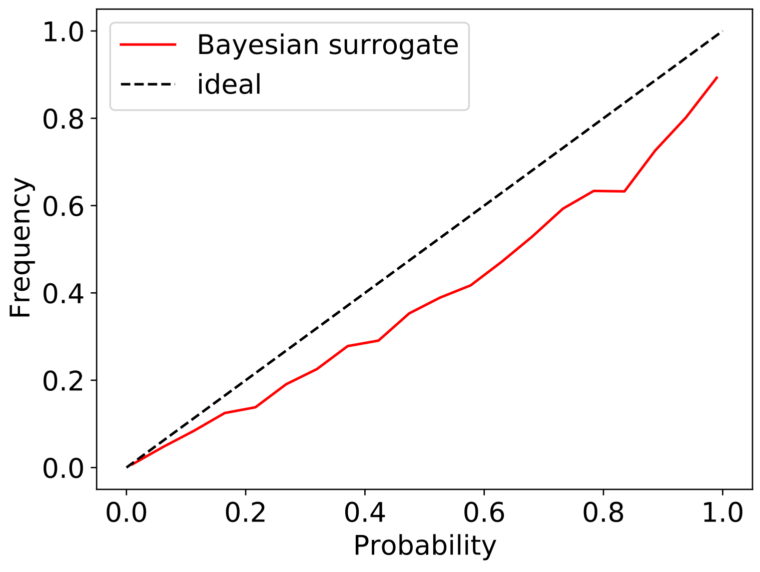


Figure 42: Reliability diagram of the PAV surrogate model. The ideal progression is plotted as dotted black line and the Bayesian surrogate in red. The model frequency was evaluated at 30 points with a maximum value of 89.2% accuracy.

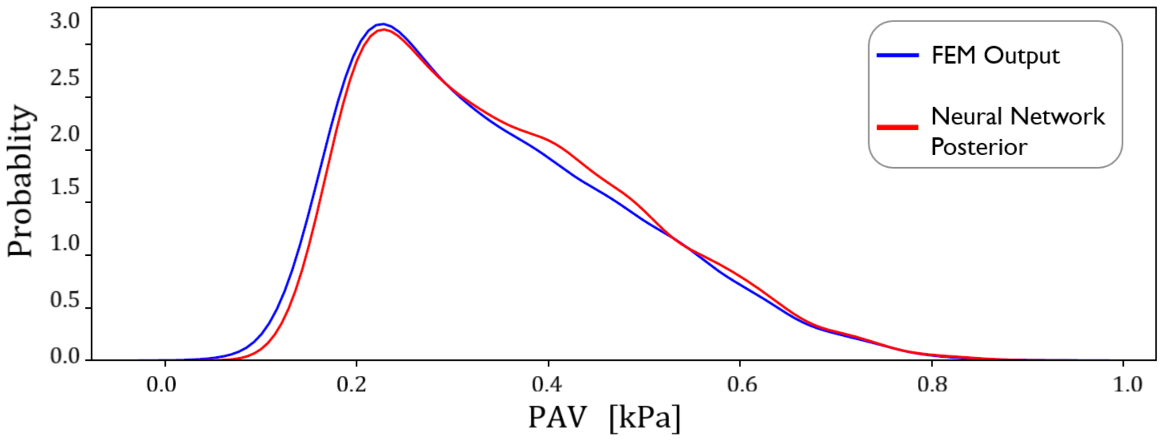


Figure 43: Posterior distribution at position $(1.5, 4.5)$.

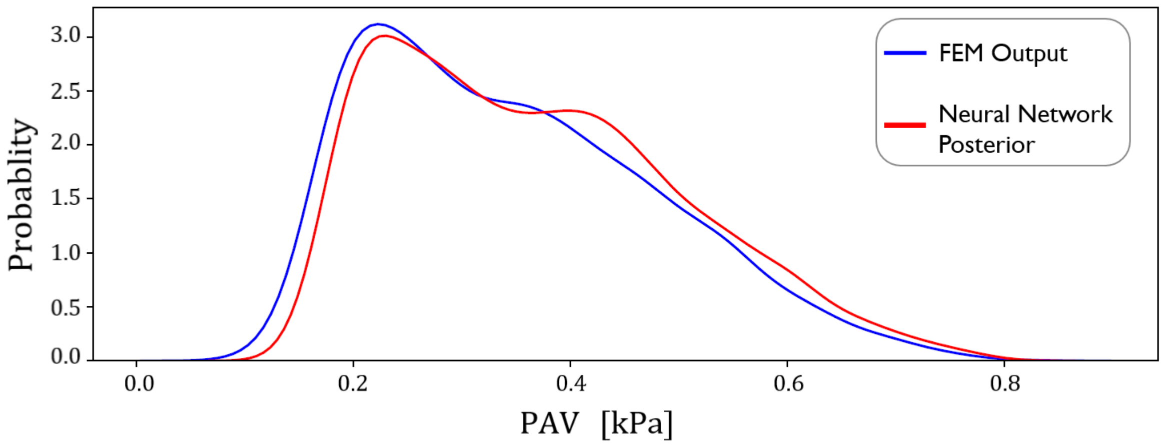


Figure 44: Posterior distribution at position $(12.5, 14.5)$.

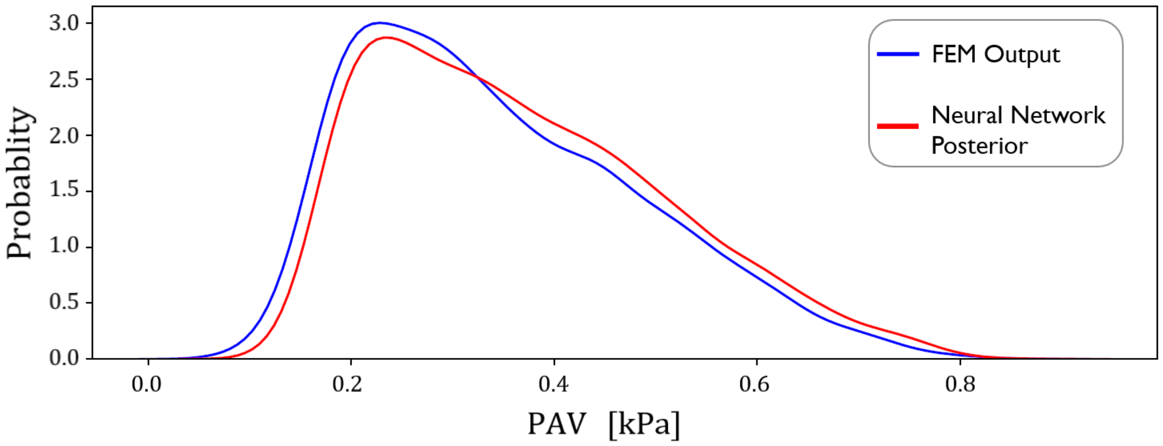


Figure 45: Posterior distribution at position $(18.5, 15.5)$.

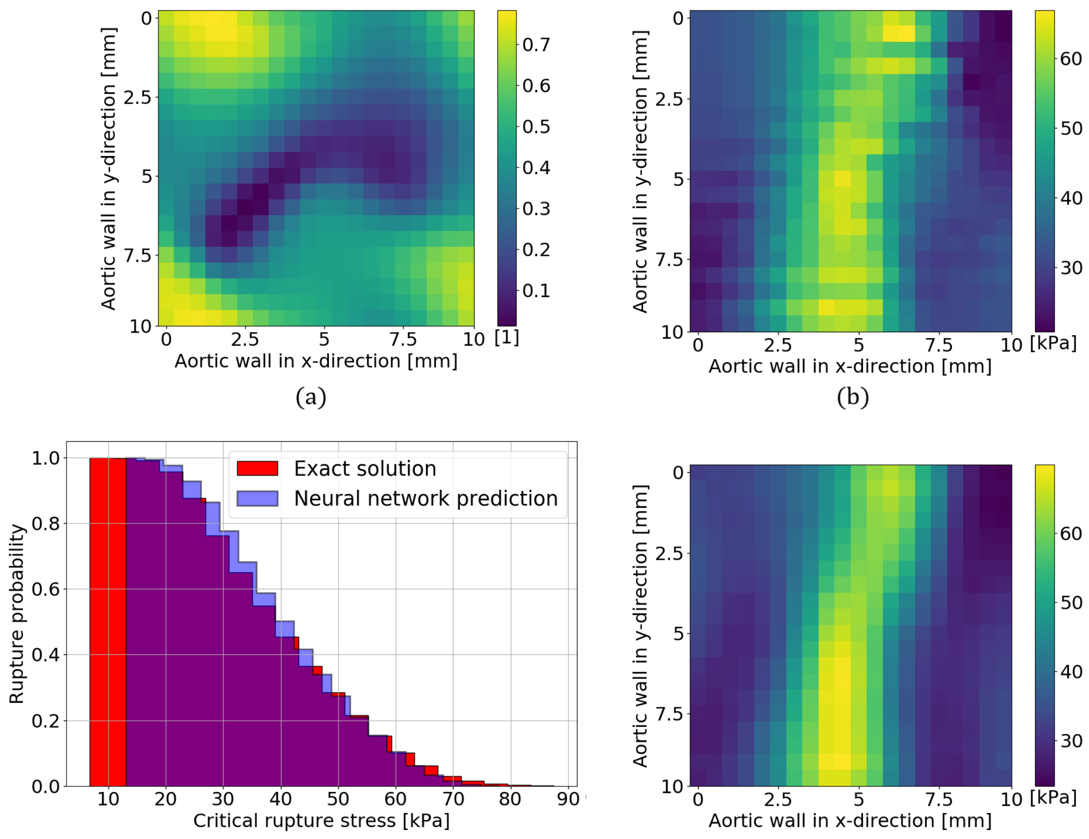
# 7 Summary and Outlook

To sum up, in this work Uncertainty Quantification of a degradation parameter used for modelling aortic walls in case of Aortic Dissection is investigated, for which an uniaxial tensile test is performed on hyperelastic, heterogeneous tissue.

First, random field simulation techniques were investigated to model the spatial distribution of a degradation parameter, see Section 3. The Spectral method was chosen through which random fields were generated. Those were used as an input to FEM calculations to the Rolf-Pissarczyk-Holzapfel model [2]. Since Uncertainty Quantification of such calculations are computationally expensive, see Section 4, a surrogate model was trained to learn the mapping from an input to the output of the uniaxial tensile test on hyperelastic tissue. This surrogate model consist of a Bayesian Deep Convolutional Encoder-Decoder, short Bayesian Autoencoder, see Section 5.1, which was adapted from [3]. The network architecture was investigated in Section 5 and Uncertainty Quantification via the surrogate model was analysed in Section 6.2 including quantities like the critical rupture stress and the principal absolute value, Section 6.3. An accuracy of 86% for the constitutive model surrogate and 89.2% for the principal absolute value surrogate was achieved. A more detailed network performance is outlined Section 6.2. Fig. 46 gives a brief summary of this work, including a random field sample, subfigure (a), the FEM output in subfigure (b), the neural network prediction in (d) and finally, the inverse cumulative stress histogram to model the critical rupture stress in subfigure (c).

Further investigations of this work could include sampling and training the model with higher level precision data, moreover one could also train the network with more or less data and compare the results. On another perspective considering the data, one can further analyse the differences between the neo-Hookean and the Rolf-Pissarczyk-Holzapfel model including their PAV. In this case, the Cauchy stress tensor was reduced to its component $\sigma_{33}$, because of two orders magnitude difference to the other elements. However, one can further investigate to train the network to predict all Cauchy stress tensor channels individually to probably obtain an even better result. Furthermore, one could also focus and compare different network architectures and their prediction performances, including different kernel sizes. Another intriguing aspect is the uncertainty quantification of the neural network itself, which was neglected in this work.

On top of that, one could modify the network to incorporate physics-information of the tensile test. This state-of-the art research is called Physics-informed Neural Network [133, 134, 135, 136, 137] and seems a promising modification for future investigations of surrogate models in biomechanics.



(a) (b) (c) (d)

Figure 46: Summary of this work. Uniformly distributed random fields, as shown in (a), were sampled and used as input to a FEM simulation. This simulation performed an uniaxial tensile test on hyperelastic tissue, as proposed by [2]. The mapping from input to output, subfigure (b), was learned by a Bayesian Autoencoder network, subfigure (d), in order to perform Uncertainty Quantification. In addition, quantities like the critical rupture stress, subfigure (c), could then be predicted by the surrogate model.

## References

[1] R. L. Taylor. FEAP - Finite Element Analysis Program, 2014. Accessed 20/01/2021.
http://www.ce.berkeley/feap

[2] M. Rolf-Pissarczyk, K. Li, D. Fleischmann, G. A. Holzapfel. *A discrete approach for modeling degraded elastic fibers in aortic dissection.* Computer Methods in Applied Mechanics and Engineering **373**.
doi:10.1016/j.cma.2020.113511

[3] Y. Zhu, N. Zabaras. *Bayesian deep convolutional encoder–decoder networks for surrogate modeling and uncertainty quantification.* Journal of Computational Physics **366** (2018) 415.
doi:10.1016/j.jcp.2018.04.018

[4] P. G. Constantine, E. Dow, Q. Wang. *Active subspace methods in theory and practice: Applications to kriging surfaces.* SIAM Journal on Scientific Computing **36** (2014).

[5] P. G. Constantine. Active Subspaces. https://github.com/paulcon/active_subspaces. Github repository. Accessed 10/01/2021.

[6] J. Deutsch. Nugget Effect. http://www.geostatisticslessons.com/lessons/nuggeteffect. Accessed 10/01/2021.

[7] G. A. Holzapfel. *Nonlinear Solid Mechanics. A Continuum Approach for Engineering.* Chichester, New York: John Wiley & Sons, 2001.

[8] M. Faber. Aortic Dissection, MSD Manuals. https://www.msdmanuals.com/professional/cardiovascular-disorders/diseases-of-the-aorta-and-its-branches/aortic-dissection. Accessed 10/01/2021.

[9] D. P. Kingma, M. Welling. *Auto-Encoding Variational Bayes.* CoRR (2014). https://arxiv.org/pdf/1312.6114.pdf, Accessed 21/01/2021.

[10] C. M. Durkan. Variational Autoencoder. https://github.com/conormdurkan/variational-autoencoder. Github repository. Accessed 10/01/2021.

[11] P. Isola, J.-Y. Zhu, T. Zhou, A. A. Efros. *Image-to-Image Translation with Conditional Adversarial Networks.* 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 5967.

[12] J. Brownlee. How to Develop a Pix2Pix GAN for Image-to-Image Translation. https://machinelearningmastery.com/how-to-develop-a-pix2pix-gan-for-image-to-image-translation/. Github repository. Accessed 10/01/2021.

[13] J. Rocca. Understanding Generative Adversarial Networks (GANs). Building, step by step, the reasoning that leads to GANs. https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29. Blog entry. Accessed 10/01/2021.

[14] N. Sucia. *Diffusion in Random Fields. Applications to Transport in Groundwater*, Band 1. Springer, 2019.
doi:10.1007/978-3-030-15081-5

[15] A. Seifried, J. Baker. Characterization of random fields at multiple scales: an efficient conditional simulation procedure and applications in geomechanics, 2011. http://web.stanford.edu/~bakerjw/Publications/Baker_et_al_(2011)_Multiscale_RF,_ICASP.pdf, Accessed 10/01/2021.

[16] B. Wandelt. *Gaussian Random Fields in Cosmostatistics*, Band 2. Springer, New York, 2012.
doi:10.1007/978-1-4614-3508-2_5

[17] J. Bardeen, J. R. Bond, N. Kaiser. *The Statistics of Peaks of Gaussian Random Fields.* Astrophysics Journal **308** (1986) 83.
doi:10.1086/164143

[18] T. Baldauf. Lecture notes: Advanced Cosmology Statistics, non-Gaussianity and non-Linearity, 2018. http://www.damtp.cam.ac.uk/user/tb561/AdvCosmo/AdvCosmo18_notes1403.pdf, Accessed 21/01/2021.

[19] B. Wandelt. *MAGIC: Exact Bayesian Covariance Estimation and Signal Reconstruction for Gaussian Random Fields* https://arxiv.org/pdf/astro-ph/0401623.pdf, Accessed 21/01/2021.

[20] B. Noetinger, L. Hume, R. Chatelin, P. Poncet. *Effective viscosity of a random mixture of fluids.* Physical Review Fluids **3** (2018).
doi:10.1103/PhysRevFluids.3.014103

[21] PyTorch. PyTorch. https://pytorch.org/. Accessed 10/01/2021.

[22] S. Otte. Deep Neural Networks with PyTorch — PyData Berlin 2018. https://www.youtube.com/watch?v=_H3aw6wkCv0. Accessed 10/01/2021.

[23] S. Otte. PyTorch Tutorial. https://github.com/sotte/pytorch_tutorial. Github repository. Accessed 10/01/2021.

[24] J. Biehler. *Efficient Uncertainty Quantification for Large-Scale Biomechanical Models Using a Bayesian Multi-Fidelity Approach.* Dissertation, Technische Universität München, 2016.

[25] W. von der Linden, V. Dose, U. Toussaint. *Bayesian Probability Theory: Applications in the Physical Sciences.* Cambridge, 2014.

[26] M. Goossens, F. Mittelbach, A. Samarin. *Bayesian Statistics the Fun Way*

- *Understanding Statistics and Probability with Star Wars, LEGO, and Rubber Ducks*. No Starch Press, 2019.

[27] R. Price. *An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes* (1763). https://royalsocietypublishing.org/doi/10.1098/rstl.1763.0053. Acessed 21/01/2021.

[28] J. Biehler, M. W. Gee, W. A. Wall. *Towards efficient uncertainty quantification in complex and large-scale biomechanical problems based on a Bayesian multi-fidelity scheme*. Biomechanics and Modeling in Mechanobiology **14** (2015) 489.
doi:10.1007/s10237-014-0618-0

[29] A. M. Yaglom. *Correlation Theory of Stationary and Related Random Functions*, Band 1. Springer-Verlag, 1986.

[30] P. Abrahamsen. *A Review of Gaussian Random Fields and Correlation Functions* **2** (1997).
doi:10.13140

[31] R. J. Adler, J. E. Taylor. *Random Fields and Geometry*. Springer-Verlag New York, 2007.

[32] G. Lindgren. *Lectures on Stationary Stochastic Processes. A Course for PhD Students in Mathematical Statistics and other fields.* (2006). http://www.maths.lth.se/matstat/staff/georg/Publications/lecture2006.pdf, Accesses 20/01/2021.

[33] D. Bolin, F. Lindgren. *Spatial models generated by nested stochastic partial differential equations, with an application to global ozone mapping* (2011).
doi:10.1214/10-AOAS383

[34] P. Sidén, F. Lindsten. *Deep Gaussian Markov random fields*. arXiv (2020). Abs/2002.07467.

[35] V. S. Pugachev. *Theory of Random Functions and its Application to Control Problems*. Pergamon Press, 1995.

[36] T. T. Soong, M. Grigoriu. *Random vibration of mechanical and structural systems*. Prentice Hall, 1993.

[37] D. Duvenaud. *Automatic model construction with Gaussian processes*. Dissertation, 2014. Dissertation. University of Toronto.

[38] J. miller. Machine Learning. https://www.youtube.com/watch?v=yDLKJtOVx5c&list=PLD0F06AA0D2E8FFBA. Accessed 10/01/2021.

[39] C. E. Rasmussen, C. K. Williams. *Gaussian Processes for Machine Learning.* The MIT Press, Cambridge, MA, USA, 2006.

[40] M. Shinozuka, G. Deodatis. *Simulation of Multi-Dimensional Gaussian Stochastic Fields by Spectral Representation.* Applied Mechanics Reviews **49** (1996) 29.

[41] G. Wolkerstorfer. DeepUQ with CNNs. `https://github.com/wolke26/DeepUQ-MasterThesis`, 2021. Acessed 21/01/2021.

[42] L. De Carvalho, V. Bouvier, R. Cottereau, D. C. Paludo. *Scalable parallel scheme for sampling of Gaussian random fields over very large domains.* International Journal for Numerical Methods in Engineering **117** (2019). doi:10.1002/nme.5981

[43] R. Gray. *Toeplitz and Circulant Matrices: A Review.* Found. Trends Commun. Inf. Theory **2** (2005).

[44] M. L. Ravalec, B. Noetinger, L. Hu. *The FFT Moving Average (FFT-MA) Generator: An Efficient Numerical Method for Generating and Conditioning Gaussian Simulations.* Mathematical Geology **32** (2000) 701.

[45] C. E. Powell. *Generating Realisations of Stationary Gaussian Random Fields by Circulant Embedding* .

[46] C. R. Dietrich, G. N. Newsam. *Fast and exact Simulation of Stationary Gaussian Processes through Circulant Embedding of the Covariance Matrix.* Sci Comput, Siam J **18** (1997) 9.

[47] E. Aune, J. Eidsvik, Y. Pokern. *Iterative numerical methods for sampling from high dimensional Gaussian distributions.* Statistics and Computing **23** (2013) 501. doi:10.1007/s11222-012-9326-8

[48] E. Chow, Y. Saad. *Preconditioned krylov subspace methods for sampling multivariate gaussian distributions.* SIAM Journal on Scientific Computing **36** (2014). doi:10.1137/130920587

[49] A. M. Panunzio, R. Cottereau, G. Puel. *Large scale random fields generation using localized Karhunen–Loève expansion.* Advanced Modeling and Simulation in Engineering Sciences **5** (2018). doi:10.1186/s40323-018-0114-7

[50] M. Shinozuka, G. Deodatis. *Simulation of Stochastic Processes by Spectral Representation.* Applied Mechanics Reviews **44** (1991) 191.

[51] E. P. Igúzquiza, M. C. Olmo. *The Fourier Integral Method: An efficient spectral method for simulation of random fields.* Mathematical Geology **25** (1993).

[52] M. D. Shields, H. Kim. *Simulation of higher-order stochastic processes by spectral representation.* Probabilistic Engineering Mechanics **47** (2017) 1. doi:10.1016/j.probengmech.2016.11.001

[53] S. O. Rice. *Mathematical Analysis of Random Noise.* Bell System Technical Journal in Vol. 23, July 1944 and in Vol. 24, January 1945 (1954).

[54] B. A. Benowitz. *Modeling and Simulation of Random Processes and Fields in Civil Engineering and Engineering Mechanics.* Dissertation. columbia university, 2013.

[55] B. A. Benowitz, M. D. Shields, G. Deodatis. *Determining evolutionary spectra from non-stationary autocorrelation functions.* Probabilistic Engineering Mechanics **41** (2015) 73.
doi:10.1016/j.probengmech.2015.06.004

[56] K. Fourmon. *Fast Fourier Transform for Non-Equidistant Meshes and Tomographic Applicatio.* Dissertation, Technische Universität Münster, 1999.

[57] D. Potts, G. Steid. *Fast summation at nonequispaced knots by NFFT.* SIAM J. Sci. Compu **24** (2003).

[58] B. J. Fischer. The Cross-Correlation and Wiener-Khinchin theorems. https://authors.library.caltech.edu/11363/2/FISjns08supp.pdf. Accessed 10/01/2021.

[59] N. Wiener. *Generalized harmonic analysis.* Acta Mathematica **55** 117.

[60] M. B. Priestley. *Spectral Analysis and Time Series.* Academic Press, 1982.

[61] A. Papoulis, S. U. Pillai. *Probability, Random Variables and Stochastic Processes.* Mcgraw-Hill Higher Education, 1965.
doi:10.2307/1266379

[62] E. Brigham. *Fast Fourier Transform and its Applications.* Prentice-hall Signal Processing Series, 1988. ISBN: 9780133075052.

[63] F. Lindgren, H. Rue, J. Lindström. *An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach* **73** (2011) 423.

[64] Havard Rue, Leonhard Held. *Gaussian Markov Random Fields Theory and Applications.* 104, 2005.

[65] A. Lang, J. Potthoff. *Fast simulation of Gaussian random fields.* Monte Carlo Methods Appl. (2011).

[66] D. Bolin, K. Kirchner. *The Rational SPDE Approach for Gaussian Random Fields With General Smoothness.* Journal of Computational and Graphical Statistics **29** (2017) 274 .

[67] J. Lindström. Gaussian Markov Random Fields. https://sites.stat.washington.edu/peter/PASI/Lecture-GMRF.pdf. Accessed 10/01/2021.

[68] P. Whittle. *On stationary processes in the plane.* Biometrika **41** (1954).

[69] P. Whittle. *Stochastic processes in several dimensions.* Bull. Inst. Int. Statist. **40** (1963).

[70] Y. Rozanov. *Markov random fields and stochastic partial differential equations.* Math. Sb. **103** (1977).

[71] H. Rue, H. Tjelmeland. *Fitting Gaussian Markov random fields to Gaussian fields.* Scandinavian Journal of Statistics **29** (2002) 31.
doi:10.1111/1467-9469.00058

[72] M. Stein. *Statistical Interpolation of Spatial Data: Some Theory for Kriging*, 1999.
doi:10.2307/2669494

[73] X. Hu, D. Simpson, F. Lindgren, H. Rue. *Multivariate Gaussian Random Fields Using Systems of Stochastic Partial Differential Equations.* arXiv: Methodology (2013).

[74] N. Cressie. *Statistics for spatial data.* John Wiley & Sons, 1993.

[75] A. S. D. Miller, R. Glennie. *Understanding the stochastic partial differential equation approach to smoothing.* JABES **25** (2020).
doi:https://doi.org/10.1007/s13253-019-00377-z

[76] E. Aune, D. Simpson, J. Eidsvik. *Parameter estimation in high dimensional Gaussian distributions.* Statistics and Computing **24** (2014) 247.

[77] B. Staber. Stochastic analysis, simulation and identification of hyperelastic constitutive equations, 2018. `https://tel.archives-ouvertes.fr/tel-01982185` Accessed 10/01/2021.

[78] S. Pezzuto, A. Quaglino, M. Potse. On Sampling Spatially-Correlated Random Fields for Complex Geometries. *Lecture Notes in Computer Science*, Band 11504 LNCS. Springer Verlag, 2019 S. 103–111.

[79] F. Lindgren. Stochastic PDEs and Markov random fieldswith ecological applications. `http://www.craigmile.com/peter/MBI/files/Lindgren_OSU2015.pdf`. Accessed 10/01/2021.

[80] A. Lang. *Simulation of Stochastic Partial Differential Equations and Stochastic Active Contours.* Dissertation, Technische Universität Mannheim, 2007.

[81] M. Grigoriu. *Applied Non-Gaussian Processes, Examples, Theory, Simulation, Linear Random Vibration nd MATLAB Solution.* Prentice Hall, 1995.

[82] M. Grigoriu. *Crossings of Non-Gaussian Translation Processes.* Journal of Engineering Mechanics **110** (1984).

[83] M. Grigoriu. *Simulation of Stationary Non-Gaussian Translation Processes.* Journal of Engineering Mechanics **124** (1998).

[84] M. Grigoriu. *Stochastic Calculus: Applications in Science and Engineering.* Birkäuser Verlag, 2002.

[85] M. Grigoriu. *Spectral Representation for a Class of Non-Gaussian Processes.* Journal of Engineering Mechanics-asce **130** (2004) 541.

[86] F. Yamazaki, M. Shinozuka. *Digital generation of non-Gaussian stochastic fields.* Journal of Engineering Mechanics **114** (1988) 1183.
doi:10.1061/(ASCE)0733-9399(1988)114:7(1183)

[87] R. Popeseu, G. Deodatis, J. H. Prevost. *Simulation of homogeneous non-Gaussian stochastic vector fields.* Prob. Engng. Mech **13** (1998) 1.

[88] M. D. Shields, G. Deodatis, P. Bocchini. *A simple and efficient methodology to approximate a general non-Gaussian stationary stochastic process by a translation process.* Probabilistic Engineering Mechanics **26** (2011) 511.
doi:10.1016/j.probengmech.2011.04.003

[89] R. E. Melchers. *Structural Reliability Analysis and Prediction.* John Wiley & Sons Ltd, 2017.
doi:10.1002/9781119266105

[90] R. Vio, P. Andreani, W. Wamsteker. *Numerical Simulation of Non-Gaussian Random Fields with Prescribed Correlation Structure.* Publications of the Astronomical Society of the Pacific **113** (2001) 1009.

[91] R. Trandafir, S. Demetriu. Numerical Simulation of Non-Gaussian Random Fields, 2005 S. p. 231–237. Proceedings of 7st Balkan Conference on Operational Research.

[92] P. Bocchini, G. Deodatis. *Critical review and latest developments of a class of simulation algorithms for strongly non-Gaussian random fields.* Probabilistic Engineering Mechanics **23** (2008) 393.
doi:10.1016/j.probengmech.2007.09.001

[93] K. K. Choi, Y. Noh, L. Du. Reliability Based Design Optimization with Correlated Input Variables. *SAE Technical Paper.* SAE International, 2007 .
`https://doi.org/10.4271/2007-01-0551`

[94] R. Lebrun, A. Dutfoy. *An innovating analysis of the Nataf transformation from the copula viewpoint.* Probabilistic Engineering Mechanics **24** (2009) 312.
doi:10.1016/j.probengmech.2008.08.001

[95] V. Ramnath. *Analysis of approximations of GUM supplement 2 based non-Gaussian PDFs of measurement models with Rosenblatt Gaussian transformation mappings.* International Journal of Metrology and Quality Engineering **11** (2020)

2.
doi:10.1051/ijmqe/2019018

[96] M. D. Shields, G. Deodatis. *Estimation of evolutionary spectra for simulation of non-stationary and non-gaussian stochastic processes.* Computers and Structures **126** (2013) 149.
doi:10.1016/j.compstruc.2013.02.007

[97] J. D. Humphrey. *Possible mechanical roles of glycosaminoglycans in thoracic aortic dissection and associations with dysregulated transforming growth factor-.* J Vasc Res 50 (1) (2013).

[98] G. A. Holzapfel, R. W. Ogden, S. Sherifova. *On fibre dispersion modelling of soft biological tissues: a review.* Proc Math Phys Eng Sci 475 (2224) (2019).
doi:10.1098/rspa.2018.0736

[99] K. Li, R. W. Ogden, G. A. Holzapfel. *A discrete fibre dispersion method for excluding fibres under compression in the modelling of fibrous tissues.* Journal of Royal Society Interface 15 (2018).
doi:10.1098/rsif.2017.0766

[100] G. Sanderson. Convolutions in image processing, Week 1, MIT 18.S191 Fall 2020. https://www.youtube.com/watch?v=8rrHTtUzyZA. Accessed 10/01/2021.

[101] Reddit. What happened to DropOut. https://www.reddit.com/r/MachineLearning/comments/5l3f1c/d_what_happened_to_dropout/. Accessed 10/01/2021.

[102] S. Saha. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. Accessed 10/01/2021.

[103] W. Kurt. Kullback-Leibler Divergence Explained. Accessed 10/01/2021.
https://www.countbayesie.com/blog/2017/5/9/kullback-leibler-divergence-explained

[104] S. Brunton. Principal Component Analysis (PCA). https://www.youtube.com/watch?v=fkf4IBRSeEc. Accessed 10/01/2021.

[105] Q. Liu, D. Wang. Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm. *NIPS*, 2016 .

[106] Y. Bengio. *Learning Deep Architectures for AI.* Found. Trends Mach. Learn. **2** (2007).

[107] Y. LeCun, Y. Bengio, G. Hinton. *Deep Learning.* Nature (2012). Accessed 10/01/2021.

[108] I. Goodfellow, Y. Bengio, A. C. Courville. *Deep Learning.* Nature **521** (2015) 436.

[109] M. D. Zeiler, R. Fergus. Visualizing and Understanding Convolutional Networks, 2014 S. 818–833.

[110] G. E. Hinton, S. Osindero, Y. Teh. *A Fast Learning Algorithm for Deep Belief Nets.* Neural Computation **18** (2006) 1527.

[111] G. E. Hinton, R. Neal. *Bayesian learning for neural networks* (1995). doi:DOI:10.1007/978-1-4612-0745-0

[112] D. MacKay. *A Practical Bayesian Framework for Backpropagation Networks.* Neural Computation **4** (1992) 448.

[113] Y. Gal, Z. Ghahramani. *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning.* ArXiv **abs/1506.02142** (2016).

[114] P. Baldi, P. Sadowski, D. Whiteson. *Searching for exotic particles in high-energy physics with deep learning.* Nature communications **5** (2014) 4308.

[115] G. Huang, Z. Liu, K. Q. Weinberger. *Densely Connected Convolutional Networks.* 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 2261.

[116] S. Ioffe, C. Szegedy. *Batch normalization: accelerating deep network training by reducing internal covariate shift.* Conference on Machine Learning (2015).

[117] X. Glorot, A. Bordes, Y. Bengio. *Deep Sparse Rectifier Neural Networks* (2011).

[118] Theano. A Python framework for fast computation of mathematical expressions, 2020. Accessed 10/01/2021.
`http://arxiv.org/abs/1605.02688`

[119] S. De, S. L. Smith. *Batch Normalization has Multiple Benefits: An Empirical Study on Residual Networks.* ICLR 2020 Conference Blind Submission (2020). Accessed 10/01/2021.

[120] J. Long, E. Shelhamer, T. Darrell. *Fully convolutional networks for semantic segmentation.* IEEE Trans Pattern Anal March Intell (2016). doi:DOI:10.1109/TPAMI.2016.2572683

[121] S. Jégou, M. Drozdzal, D. Vázquez, A. Romero, Y. Bengio. *The One Hundred Layers Tiramisu: Fully Convolutional DenseNets for Semantic Segmentation.* 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) (2017) 1175.

[122] C. Louizos, M. Welling. *Multiplicative Normalizing Flows for Variational Bayesian Neural Networks.* ArXiv **abs/1703.01961** (2017).

[123] Huber Loss. `https://en.wikipedia.org/wiki/Huber_loss`. Accessed 10/01/2021.

[124] A. Koepf. Smooth L1 loss vs. Huber loss. https://github.com/torch/nn/issues/579, 2016. Github repository. Accessed 10/01/2021.

[125] R. B. Girshick. *Fast R-CNN.* 2015 IEEE International Conference on Computer Vision (ICCV) (2015) 1440.

[126] PyTorch. Conv2D. https://pytorch.org/docs/stable/generated/torch.nn.SmoothL1Loss.html. Accessed 10/01/2021.

[127] A. Bilogur. Cosine annealed warm restart learning schedulers. https://www.kaggle.com/residentmario/cosine-annealed-warm-restart-learning-schedulers, 2019.

[128] PyTorch. Learning Rate Scheduler - Cosine Annealing. https://pytorch.org/docs/stable/optim.html, 2020. Accessed 10/01/2021.

[129] I. Loshchilov, F. Hutter. *SGDR: Stochastic Gradient Descent with Warm Restarts.* arXiv: Learning (2017).

[130] C. Leibig, V. Allken, M. Ayhan, P. Berens, S. Wahl. *Leveraging uncertainty information from deep neural networks for disease detection.* Scientific Reports **7** (2017).

[131] A. Malaa. Deep-learning-uncertainty. https://github.com/ahmedmalaa/deep-learning-uncertainty, 2020. GitHub repository. Acessed 21/01/2021.

[132] T. yu Song, W. Ding, H. Liu, J. Wu, H. Zhou, J. Chu. *Uncertainty Quantification in Machine Learning Modeling for Multi-Step Time Series Forecasting: Example of Recurrent Neural Networks in Discharge Simulations.* Water **12** (2020) 912.

[133] M. Raissi, P. Perdikaris, G. Karniadakis. *Physics Informed Deep Learning (Part II): Data-driven Discovery of Nonlinear Partial Differential Equations.* ArXiv **abs/1711.10566** (2017).

[134] M. Raissi. *Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations.* ArXiv **abs/1801.06637** (2018).

[135] L. Lu, X. Meng, Z. Mao, G. Karniadakis. *DeepXDE: A Deep Learning Library for Solving Differential Equations.* ArXiv **abs/1907.04502** (2020).

[136] G. Pang, M. D'Elia, M. Parks, G. Karniadakis. *nPINNs: nonlocal Physics-Informed Neural Networks for a parametrized nonlocal universal Laplacian operator. Algorithms and Applications.* ArXiv **abs/2004.04276** (2020).

[137] H. Gao, L. Sun, J. X. Wang. *PhyGeoNet: Physics-Informed Geometry-Adaptive Convolutional Neural Networks for Solving Parametric PDEs on Irregular Domain.* Journal of Computational Physics **428** (2021) 110079. doi:https://doi.org/10.1016/j.jcp.2020.110079

# Appendix

The following code snippet can be used to simulate Gaussian Random fields and analytic non-Gaussian random fields via the FFT-Method:

```
import numpy as np
import matplotlib.pyplot as plt
from math import pi
from scipy.stats import beta
from tqdm import tqdm

def PSDF_2D(w,k, ell1,ell2, sig):
    """ Calculate the Power Spectral Density, see eq. (65)."""
    praefac = sig*(ell1)*(ell2)/ (4*pi)
    xi1 =  np.sum(w**2,1).reshape(-1,1) + np.sum(w**2,1)
    xi2 =  np.sum(k**2,1).reshape(-1,1) + np.sum(k**2,1)
    ker = praefac * np.exp(-.25* (xi1 * ell1**2 + xi2 * ell2**2))
    return(ker / np.sum(ker))

def Lognstat(mu, sigma):
    """Calculate the mean of and variance of the Lognormal distribution
   given
    the mean (‘mu‘) and standard deviation (‘sigma‘), of the associated
   normal
    distribution."""
    m = np.exp(mu + sigma**2 / 2.0)
    var = np.exp(2 * mu + sigma**2) * (np.exp(sigma**2) - 1)
    return (m, var)

def Gamma(sample_field,n):
    """Calculate a Gamma field as in eq. (95)"""
    lim = int(2*n)
    G = np.sum([ (np.square(np.array(sample_field)[i])) for i in range(
   lim)],0)
    return(G)

def Beta(Gamma1, Gamma2):
    """Calculate a Beta field as in eq. (100)"""
    return(Gamma1 / (Gamma1 + Gamma2))


def Logn(sample_field,mu_g, sig_g):
    """Calculate a Lognormal field as in eq. (108)"""
    log_field = np.exp(mu_g + sig_g*sample_field)
    log_mu,log_sig = Lognstat(mu_g,sig_g)
    return(log_field,  log_mu, log_sig)

def Unif(Gamma1, Gamma2):
    """Calculate a Uniformly disrtibuted field as a special case of Beta
   field"""
    return(0.5 * Gamma1 / (0.5* Gamma1 + 0.5* Gamma2))

def mult(A,B):
    return(A*B)

# Frequency domain
Nw = 2**7
Nk = 2**7

dw = 0.0781
dk = 0.0781
```

```

kmax = dk*Nk
wmax = dw*Nw

w = np.dot(dw , range(0,Nw-1)).reshape(-1,1)
k = np.dot(dk ,range(0,Nk-1)).reshape(-1,1)

# Spatial Domain
Mw = 2*Nw
Mk = 2*Nk

t = 2*pi/dw *np.linspace(0,1,Mw+1)
Mt = len(t)
x = 2*pi/dk *np.linspace(0,1,Mk+1)
Mx = len(x)

# Define field correlation lengthscales
corr1 = 10
corr2 = 10
sig = 1

S_freq_domain = PSDF_2D(w,k,corr1,corr2,sig)
S_spatial_domain = np.zeros(shape =(Mk,Mw))
S_spatial_domain[0:Nk-1,0:Nw-1] = S_freq_domain


Log_fields = []
Beta_fields = []
Uni_fields = []
Gaussian_fields =  []

Beta_22  = []
Beta_0505= []
Beta_24= []
Beta_41= []

set_seed = 12
for numbers in tqdm(range(0,10)):

    gamma_8 = []
    gamma_4 = []

    for m in (range(set_seed)):
        np.random.seed(m+numbers*set_seed)

        phi1 = np.random.rand(Mk,Mw)*2*pi
        phi2 = np.random.rand(Mk,Mw)*2*pi
        B1 = 2*np.array(list(map(mult,np.sqrt(S_spatial_domain*dk*dw), np
    .exp(1j*phi1))))
        B2 = 2*np.array(list(map(mult,np.sqrt(S_spatial_domain*dk*dw), np
    .exp(1j*phi2))))

        F1 =  (Mk*2*pi) * np.fft.ifft(B1,Mx,0)
        F2 = (Mk*2*pi)  * np.fft.ifft(B2,Mx,0)

        F1 = Mw * np.fft.ifft(F1,Mt,1)
        F2 =   np.fft.fft(F2,Mt,1)


        # Sample of Gaussian Random Field via FFT-method
        y = np.real(F1+F2)
        Gaussian_fields.append(y)
```

```python


        if m < 8:
            gamma_8.append(y)
        if m > 7 :
            gamma_4.append(y)



    Gamma4 = np.array(Gamma(gamma_8,4))
    Gamma2 = np.array(Gamma(gamma_4,2))
    Uniform1 = np.array(Gamma(gamma_8,1))
    Uniform2 = np.array(Gamma(gamma_4,1))
    Gamma1_1 = np.array(Gamma(gamma_8,1))
    Gamma1_2 = np.array(Gamma(gamma_8,2))
    Gamma2_2 = np.array(Gamma(gamma_8,2))
    Gamma05_1 = np.array(Gamma(gamma_8,.5))
    Gamma05_2 = np.array(Gamma(gamma_4,.5))

    Beta_22.append(Beta(Gamma2,Gamma2_2))
    Beta_0505.append(Beta(Gamma05_1,Gamma05_2))
    Beta_24.append(Beta(Gamma2 ,Gamma4))
    Beta_41.append(Beta(Gamma4,Gamma1_1))


    Log_fields.append(Logn(y,np.mean(y),np.std(y)))
    Beta_fields.append(Beta(Gamma4,Gamma2))
    Uni_fields.append(Unif(Uniform1, Uniform2))

# Plot the result



def PlotBetaPdf():
    """  Define the distribution parameters to be plotted """
    alpha_values = [4, 2,2,  4]
    beta_values =  [2, 2,4,  1]
    linestyles = ['-', '--', ':','-']
    x = np.linspace(0, 1, 1002)[1:-1]

    for a, b, ls in zip(alpha_values, beta_values, linestyles):
        dist = beta(a, b)

        plt.plot(x, dist.pdf(x), ls=ls,
                 label=r'$m=%.1f,\ n=%.1f$' % (a, b), color= 'k')

    plt.xlim(0, 1)
    plt.ylim(0, 3)

    plt.xlabel('$x$', size = 22)
    plt.ylabel(r'$p(x|m,n)$', size = 22)
    leg = plt.legend(loc=0, fontsize = 18)
    plt.tick_params(labelsize=20)
    plt.show()
    plt.tight_layout()
    return

i = 0 # index for plot, number in [0,10]
fig = plt.figure(figsize=(5, 7))
plt.subplots_adjust(wspace= 0.25, hspace= 0.25)

sub1 = fig.add_subplot(3,2,1)
```

```
ax = plt.gca()
plt.axis('off')
plt.title('Beta(4,2)', size = 16)
im = ax.imshow(Beta_fields[i])
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
cbar = plt.colorbar(im, cax=cax)
cbar.ax.tick_params(labelsize=16)

sub2 = fig.add_subplot(3,2,2) # two rows, two columns, second cell
ax = plt.gca()
plt.axis('off')
plt.title('Beta(2,2)', size = 16)
im = ax.imshow(Beta_22[i])
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
cbar = plt.colorbar(im, cax=cax)
cbar.ax.tick_params(labelsize=16)

sub3 = fig.add_subplot(3,2,3)
ax = plt.gca()
plt.axis('off')
plt.title('Beta(2,4)', size = 16)
im = ax.imshow(Beta_24[i])
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
cbar = plt.colorbar(im, cax=cax)
cbar.ax.tick_params(labelsize=16)

sub4 = fig.add_subplot(3,2,4)
ax = plt.gca()
plt.axis('off')
plt.tick_params(labelsize=16)
plt.title('Beta(4,1)', size = 16)
im = ax.imshow( Beta_41[i])
divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
cbar = plt.colorbar(im, cax=cax)
cbar.ax.tick_params(labelsize=16)
sub5 = fig.add_subplot(3,2,(5,6))

plt.tight_layout()
PlotBetaPdf()
```

Listing 1: Code snippet to generate Gaussian and Non-Gaussian random fields with plots.