Uncertainty Quantification for Hyperelastic Composite Material Simulations

Clients:
O.T. Turan
Y. Kato

Teaching Assistant:
Krzysztof Baran

Student names and student numbers:
Stijn Bier – 5408881
Anouk de Koning – 5109752
Erwin Bus – 5324270
Vincent Sprenger - 5173213
Daan van Haasteren - 4665465

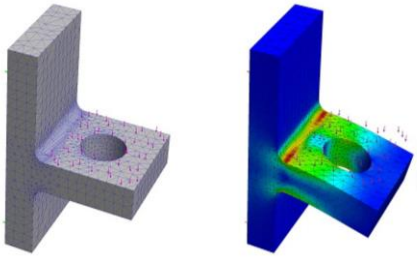


Figure 1. A visual representation of a FEM model, with prescribed loads and displacements [3]

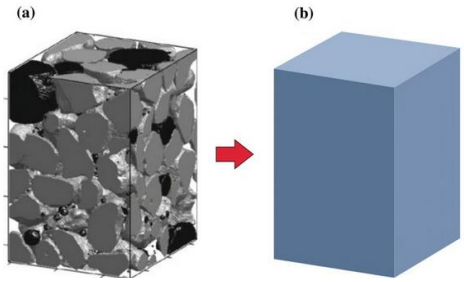


Figure 2. A material model is subjected to homogenisation [6]

1 INTRODUCTION

1.1 FE^2

In the past, if you wanted to build or design something, you would draw a plan, build a prototype, perform experiments, redesign it and continue this cycle until a sufficient design came up. With the advent of newer materials capable of taking more complex and irregular shapes, these techniques slowly became impractical. To keep up with the possibilities of newer materials and production techniques, the finite elements method or FEM was devised.

FEM divides a complex shape into tiny pieces, creating a mesh [1]. Boundary conditions are imposed on this mesh to imitate the loading conditions the structure would see in the real world. The deformations of the elements affected by these boundary conditions are calculated. Using these deformations, one can then calculate the stresses acting on this element. The deformations are used to calculate the deformations of the neighbouring elements, and so on, until the deformations and stresses of the entire model are in equilibrium [2]. Of course, all of this is an approximation, however, if a fine enough mesh is used, these approximations can accurately approximate the real world. This information about stress distributions and deformations can be used to identify weak points and potential points of failure in a given structure. In Figure 1, an example of the FEM analysis is shown. A load is put onto the structure, and the resulting deformations and stresses can also be seen in Figure 1.

Some modern engineering problems require materials which have a high performance. In order to achieve this high performance, composites can be used [4]. Some of these composites are materials which have in-homogeneity on multiple scales. So the micro-scale interactions between the components of these materials influence the performance of the material. However, FEM cannot properly analyse models with these inhomogeneities because FEM homogenises the elements and in doing so it makes necessary simplifications. With these simplifications in the mesomodel, the stress which can occur in those different scales is ignored. So this does not accurately represent the stress in a structure that uses these composites [5].

To get an accurate representation of composites, a method called computational homogenization is used. This method homogenises a representative element of the micro-scale of the composite as shown in Figure 2, in 2.a a heterogeneous material is shown and in 2 b an equivalent homogeneous material can be seen. When dealing with a nonlinear heterogeneous material, the FE^2method can be used.

In FE^2, each macroscopic integration point is associated with a Representative Volume Element (RVE). An RVE can be considered as a small, representative sample of a heterogeneous material that

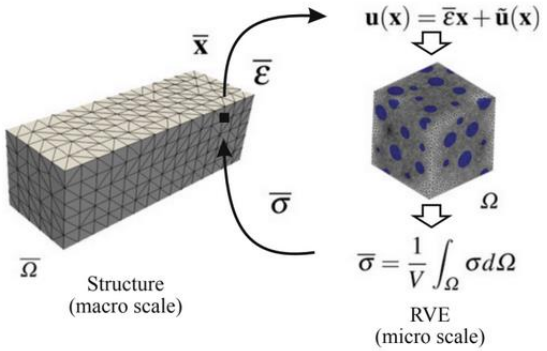


Figure 3. An FE² model with an RVE and relevant equations [6]

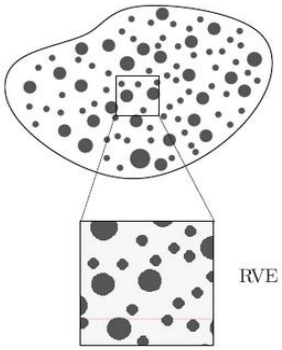


Figure 4. A non-homogenious material and an RVE of this material [8]

is used to predict the material's behaviour [7]. A macroscopic strain is put onto the integration point as the boundary conditions for the RVE. With these steps, it is possible to solve the relevant homogenization problem for every increment of the macroscopic loading put onto the structure. In Figure 3, the domain of the structure is defined by $\underline{\Omega} \subset \mathbb{R}^D$ here D is the dimension of the space, the boundary is described by $\partial\underline{\Omega}$. The domain of the associated RVE is, $\Omega \subset \mathbb{R}^D$ and the boundary is $\partial\Omega$. With a macroscopic strain of $\underline{\varepsilon}$ and a field of internal variables $\alpha(x,t)$ within the RVE, solving the local problem on the RVE with the following formulas:

$$\nabla * \sigma(u(x)) = 0 \; \forall x \in \Omega,$$

with

$$\sigma(x,t) = \mathcal{g}(\varepsilon(x,t), \alpha(x,t)),$$

and verifying

$$\langle \varepsilon(x,t) \rangle = \underline{\varepsilon}(t)$$

$\mathcal{g}$ is a nonlinear operator which is associated with the nonlinear behaviour of the phase within the RVE. With these equations, it becomes clear that the macroscopic response $\underline{\sigma}$ as seen in Figure 3, depends on the macroscopic tensor values of $\underline{\varepsilon}(t)$ and on the tensorial field $\alpha(x,t)$ within the whole RVE.

The FE^2 method works in three steps:

1. Defining the boundary conditions on the boundary $\partial\Omega$ of the RVE.
2. Solving the equations given above by using a numerical method.
3. Average the equilibrated stress solution over the RVE to get $\underline{\sigma}(t)$

In Figure 3 it is shown that the boundary condition of the RVE is $u(x) = \underline{\varepsilon}x + \tilde{u}(x)$. This formula is derived from the macroscopic strain which is applied to the structure. The last step is also shown in Figure 3, the stress results that were given by step 2 are integrated over the domain in order to get the macroscopic stress that the structure experiences.

Producing these RVE's is a tedious task, because the size of an RVE should strike a balance between being small enough for efficient computation and large enough to depict the microstructure [7] accurately**.** The micro model is considered representative only if increasing its size does not alter its average response [5]. These two properties of RVE's make it challenging to produce a correct RVE. Another problem with producing RVE's is the placement of particles which makes the composite in-homogeneous, these particles are placed randomly throughout the composite [5]. When these RVE's are created, the random particles can be placed from different distributions, for example a Gaussian or uniform distribution. In Figure 4, it can be observed that the particles are placed randomly throughout the RVE. Seeing the randomness in the RVE, uncertainty in the analysis is to be expected because the location of the particles, the microstructure, heavily influences how the RVE will react to the boundary conditions that are placed onto the RVE and also, in turn, affect the macroscopic stress

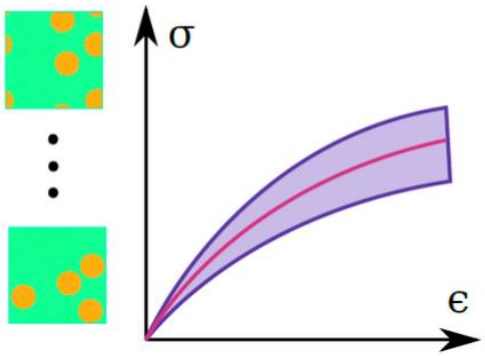


Figure 5. Stress-Strain curves and the uncertainty introduced by RVE's [4]

that the material experiences. In Figure 5, a schematic representation of the uncertainty in the stress reactions can be observed. However normally when doing an FE^2 analysis, many RVE's need to be created in order to find an RVE which best represents the microstructure and does not have a different response when the size is increased. This process takes a lot of time and computational power. A way to bypass this extensive RVE generation is to know the uncertainty that comes with a specific chosen RVE. In this experiment two different distributions were chosen, creating two types of RVE's. When strains are put onto these RVE's they all give slightly different values for the stress, with these variations it is possible to map the uncertainty that the type of RVE introduces. With this uncertainty, there is no longer a necessity for the most accurate RVE, because the uncertainty caused by the generation of random RVE's is known. This method of RVE generation saves much time and computational power.

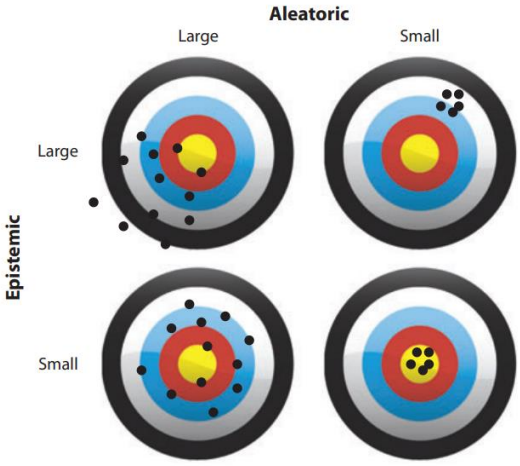


Figure 6. Aleatoric and epistemic uncertainty visualised in their similarity to precision/ accuracy concept [10]

1.2 Uncertainty Quantification

Computer simulations are used when problems become too complex to solve analytically [9]. Some of these problems are highly complex, so there is no similar reliable data to train models on. This means that more experimental data is used, however this experimental data also has varying degrees of uncertainty. These uncertainties are mathematical representations of lack of knowledge of that system [10]. Furthermore, in engineering frequently the available information is also subject to uncertainty. This can be caused by for example observations which are incomplete or inaccurate measurement equipment and of course human mistakes and errors in manufacturing are also present. This uncertain information contributes to uncertainty in a specific project or problem [11]**.** That is why it is vital to map these uncertainties because the findings that these computer simulations make highly influence engineering decisions. Quantifying the uncertainty in a model involves determining the error bounds or probability distributions of the model's parameters or input variables based on the data available for the system. [10].

In the case of this experiment, the stress results of the RVE's are where the UQ is necessary because the distributions of the stress at a specific strain will show how reliable the composite will be when the strain is put onto the structure. This experiment is done in a stochastic method, when the random particles were placed from either the Gaussian or uniform distributions the experiment became stochastic. With FE^2 and the specific RVE generation that was used, the stochasticity comes from a small scale and is being upscaled to a larger scale. This stochasticity brings uncertainty into the model. When material science experiments are done they can be deterministic, this means that all the model parameters can be determined with absolute certainty [10]. However, this experiment is stochastic, so the input variables are already uncertain and with the help of Bayesian machine learning models it is possible to map this uncertainty in the input variables.

There are two different types of uncertainty, aleatoric and epistemic uncertainty [12]. Where aleatoric uncertainty is known as the data uncertainty. This uncertainty is a result of the inherent random variability in, for example, the material structure [10]. Aleatoric uncertainty can be quantified in the form of a frequency distribution. Epistemic uncertainty is also known as knowledge uncertainty. When there is epistemic uncertainty regarding a specific variable it can be denoted in two types: a poorly known stochastic quantity or a poorly known deterministic quantity [11]. When the variable is a poorly

known stochastic quantity, the distribution type and/or the distribution parameters are uncertain. If the variable is a poorly known deterministic quantity, the wrong outcome or value is given to this variable. Epistemic uncertainty cannot be described by a probability distribution, additional knowledge or data can reduce the uncertainty. Figure 6 shows how aleatoric and epistemic uncertainty can be interpreted for a better understanding. In the scope of this experiment the aleatoric uncertainty is a result of the random placement of particles when creating the RVE's, because they are placed with different distributions introducing randomness into the model. The epistemic uncertainty in this experiment comes from the uncertainty in the model, which can be attributed to the model parameters. With this specific experiment, the total uncertainty will be mapped, there will not be a categorisation between the aleatoric and epistemic uncertainty.

1.3 Project aim and requirements

This report aims to investigate the effect of different random sampling methods of particles in an RVE for computational homogenization predictions of uncertainty for two different Bayesian models. More specifically a Bayesian neural network and a Gaussian Process. In order to investigate these topics, an understanding of Bayesian statistics is required, the two models which are a Bayesian neural networks and Gaussian Process, and the context of uncertainty quantification. The requirements which were set for this project are listed below. The requirements are formed using the MoSCoW analysis, which helps to prioritise the requirements.

1.3.1 Must have

- The developer must be able to explain the cause of the uncertainty in the given data.
- The developer must train and tune two probabilistic machine learning methods on the given data.
- The developer must display an understanding of the topic and the methods and models used. In order to do this, they must use the suggested papers and any other literature to create sufficient understanding.
- The developer must reflect on the log-likelihood performance given the predicted uncertainty.

1.3.2 Should have

- The developer should, using log likelihood, identify and reflect on the limitations of the chosen model and methods in the paper.
- The developer should observe heteroscedastic noise in data by performing kernel density estimation.

1.3.3 Could have

- If time permits, the developers could extend the model for the 3D input-output regression problem.

1.3.4 Won't have

- Won't be involved in the data generation process.
- Won't implement any machine learning algorithms from scratch.

2 LITERATURE REVIEW

2.1 Bayesian Statistics

A technique often used in the field of uncertainty quantification are Bayesian methods. Bayesian methods are a form of statistical inference. They compute the probability of an outcome using Bayes' theorem, where knowledge of prior probabilities are used to update one's beliefs about the probability of a specific outcome.

The Bayesian method, in the context of machine learning, follows three steps. First, a complete probabilistic model is set up, this is based on the available data and consistent with underlying natural phenomena and observation techniques. After this, a model is trained on the observable data and posterior distributions are created for the qualities of interest. Finally, the model must be evaluated, some metric must be chosen to evaluate how accurately a model fits to the real world. If this accuracy is insufficient, the scope of the model can be altered or expanded until a sufficiently accurate model is reached [13].

In this project, these techniques were used to perform a regression task. As discussed before the first step is to make a complete probabilistic model, linking the available or observed data denoted as y and the parameters that describe distribution, denoted as θ.

Taking the product of the distribution of these two, called the *prior distribution p(θ)* and the *sampling distribution p(y|θ)*, gives the *joint probability density function*:

$$p(\theta, y) = p(\theta)p(y|\theta)$$

Conditioning on the known values of y, using conditional probabilities known as Bayes' rule the *posterior distribution can be found:*

$$p(\theta|y) = \frac{p(\theta, y)}{p(y)} = \frac{p(\theta)p(y|\theta)}{p(y)}$$

Since y is the observations and thus does not change, *p(y)* can be considered constant and the above equation can be simplified to:

$$p(\theta|y) \propto p(\theta)p(y|\theta)$$

Finding this *posterior distribution* $p(\theta|y)$ is the main aim of the Bayesian method. Any Bayesian model is aimed at finding this posterior, as this is the key to make inferences about future data [14].

Before any data is considered, the unknown distribution of the observable data called the marginal distribution is given by:

$$p(y) = \int p(y, \theta)d\theta$$

After the data is observed, and the *posterior distribution* is found, predictions about unseen data can be made. This *posterior predictive distribution* or PPD is given by:

$$p(\tilde{y}|y) = \int p(\tilde{y}|\theta)p(\theta|y)d\theta$$

For a more in-depth derivation of this PPD please see "Bayesian Data Analysis" by Andrew Gelman, et al.[13]

2.2 Gaussian Processes

A Gaussian Process is a machine learning method which utilises the Bayesian method to make predictions about a dataset. Given a dataset, there are potentially an infinite number of possible functions that fit this data, a Gaussian Process assigns a probability to each of these functions. This allows one to make informed predictions about this dataset.

A Gaussian Process is defined by two parts, a mean function m($\boldsymbol{x}$) and a covariance function k($\boldsymbol{x}$,$\boldsymbol{x}$'). Where $\boldsymbol{x}$ is the training dataset and $\boldsymbol{x}$' is the test dataset. This covariance function can be thought of as a measure of similarity between two points. The measure one uses is called the *kernel.* Although many different kernels are available and can be combined to fit datasets of many different shapes, that is outside this project's scope and we will limit ourselves to the Radial Basis Function or RBF. this kernel, denoted $K(\boldsymbol{x}, \boldsymbol{x}'|\theta)$, is given by:

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 * exp(-\frac{1}{2}(\frac{\boldsymbol{x} - \boldsymbol{x}'}{L^2}))^2$$

Where σ^2 and L are predetermined hyperparameters, variance and length respectively. By adjusting these hyperparameters, the model can be tuned to most accurately represent the training data. Increasing the length scale will make the model behave more erratic thus leading to possibly more overfitting while decreasing the length scale will smoothen out the model thus leading to possibly underfitting [15].

Using this kernel, a kernel matrix is set up. This kernel matrix describes the prior distributions of the data that can later be used in the Bayesian method to determine the posterior distribution. The kernel matrix is a square |N| x |N| matrix where |N| is equal to the number of data points. The kernel matrix consists of a pairwise evaluation of the kernel function between every two training points. And a mean vector, which during the calculation phase is often assumed as 0 for the sake of simplifying calculations, as it can easily be added back on at the end [16] This kernel matrix is given by:

$$\mathbf{f}_* \sim N(0, K(X_*, X_*))$$

Where $\mathbf{f}_*$ is the random Gaussian vector, and $K(X_*, X_*)$ is the covariance between the two points. The second step is to combine this model with the observed data to get the posterior distribution. This distribution is given by:

$$y = f(\boldsymbol{x}) + \varepsilon$$

Where $f(\boldsymbol{x})$ gives the function of the mean of the dataset and ε is the error term, or the spread of the data, with variance σ_n^2. A visual explanation of this can be seen in Figure 5, where the red line follows the function of the mean, and width of the purple section is the variance.The prior covariance of this noisy function is then given by:

$$cov(\boldsymbol{y}) = K(X, X) + \sigma_n^2 I$$

This gets concatenated with the prior random Gaussian vector to give the joint distribution

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X, X) + \sigma_n^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right)$$

This vector can be used to calculate the posterior distribution:

$$\mathbf{f}_* | X, \mathbf{y}, X_* \sim N(\mathbf{f}_*, cov(\mathbf{f}_*)), \text{ where}$$
$$\mathbf{f}_* \triangleq E[\mathbf{f}_* | X, \mathbf{y}, X_*] = K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}\mathbf{y},$$
$$cov(\mathbf{f}_*) = K(X_*, X_*) - K(X_*, X)[K(X, X) + \sigma_n^2 I]^{-1}K(X, X_*) \text{ [16]}$$

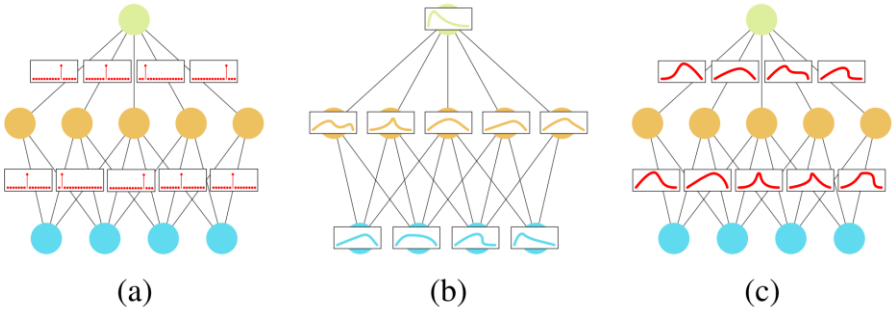


Figure 7. Neural Networks. a) An artificial neural network with weights. b) A Bayesian neural network with distributions over the weights. c) A Bayesian neural network with weights over the activations[17].

This is a Gaussian distribution with its mean equal to $\mathbf{f}_*$ and variance $cov(\mathbf{f}_*)$. The main benefit a Gaussian Process has over other machine learning algorithms is that it outputs a probabilistic model compared to other machine learning techniques that only output a single value.

One major drawback however is how computationally expensive it is when using large datasets. This is due to Its $O(N^3)$ time complexity. This scaling is so high due to having to invert the kernel matrix $K^{(x,x)}$ [16]. Another downside is that the model only outputs Gaussian functions as posterior density functions. This means that if the real data's uncertainty doesn't reflect a Gaussian distribution, the prediction given by the model will not be reliable.

2.3 Bayesian Neural Networks

In an artificial neural network deterministic values are assigned to the network's weights and biases, these are optimised in training to try and best fit a distribution. These deterministic values mean that once trained, an artificial neural network will always make the same predictions given the same data. A Bayesian neural network treats the weights as random variables rather than deterministic values on the weights and biases.

These distributions are the priors that are joined together in the layers of the neural network, to form the joint probabilistic distribution. The posterior distributions are then calculated and tuned, as the second and third step of the Bayesian framework. The priors are usually initiated as standard normal distribution. These prior distributions and the observations are evaluated to gain a posterior distribution. Finally the model is trained by slightly varying the priors and regressing to a maximum likelihood outcome.

This distribution rather than a deterministic value also means that, if sampled multiple times, a Bayesian neural network will make a different prediction each time. This gives it some immediate advantages over standard neural networks, as it is much better able to account for real life uncertainties. A major problem with many other machine learning models is that they overestimate their confidence, as they are unable to give uncertain or unclassified predictions, this causes them to often be overconfident in their predictions [17].

Bayesian neural networks, because they inherently incorporate the uncertainty in the data, are better at estimating the accuracy of their predictions [18]. Another advantage is that they give a prediction within a distribution, making them more flexible than other neural networks. Where a GP outputs a mean and variance, with which a normal distribution can be described immediately, a BNN outputs stochastic predictions, in order to find the distribution of these predictions, techniques such as Markov Chain Monte Carlo (MCMC), Hamiltonian Monte Carlo (HMC) or Variational Inference (VI) are used [19]. Where a Gaussian Process is only able to output Gaussian posterior distributions, a Bayesian

neural network is able to output any distribution. The disadvantages of a BNN is that they are more complex to set up and implement, as they require careful tuning of hyperparameters and an understanding of both neural networks and Bayesian statistics [20]. They are also computationally very expensive and require many epochs to converge to a confident model. Another limitation of BNN's is that the outcome of the network is influenced by the accuracy of the prior assumptions or model. As said before usually the priors are chosen as standard normal distributions, however the distributions of the actual data could be very different. The priors that are chosen influence how the model will be trained, which in turn can influence the predictions that the model will make [21].

2.4 Uncertainty Quantification

2.4.1 Posterior Predictive Distribution

PPD are used to quantify the uncertainty in model estimates. Knowing the distribution of a target value can be much more informative than simply regressing to a single value, and this knowledge can allow us to make more robust predictions about new data [22].

For Gaussian Process, the PPD is Gaussian, so to sample and obtain the PPD with a Gaussian Process, first the mean and the covariance has to be calculated, these will be used to form the PPD. For the Bayesian neural network, the PPD will be found using a kernel density estimation.

2.4.2 Predictive Density Function

With Gaussian Process the mean and standard deviation of the given dataset are determined, which together define a distribution over functions, given by the formula:

$$p(h) = \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{1}{2\sigma^2}(h(x_i) - \mu_i)^2)$$

With h being one of the possible functions [22].

2.4.3 Kernel Density Estimation

Kernel density estimation is a way to determine the predictive density function of a given point. With Bayesian Neural Networks Hamilton Monte Carlo is used, and HMC can be used to obtain a set of samples from the posterior distribution [23]. These samples can be used with KDE to estimate the density function in Figures 11 and 12 the results of the KDEs of the uniform and Gaussian datasets can be seen.

3 METHODOLOGY

3.1 Data Exploration

The data used in the experiment is generated by modelling 2000 RVE's. In half of these the particles were distributed uniformly, while in the other half they were distributed according to a Gaussian distribution. This created two separate datasets on which model training was performed and predictions were made. These two datasets will be referred to as the 'uniform' and 'Gaussian' datasets. Using FEM analysis these RVE's were subjected to increasing strain, and their internal average stress calculated. The strains the RVE's were subjected to, were the features that the machine learning models were trained on and the corresponding stresses the labels for the required prediction. These stress strain curves are shown in Figure 9a and 10a. They display a non-linear elasticity in line with other literature on hyperelastic materials [**24**]. The data roughly follows a root function, starting out steep and ending quite flat.

Since the distribution of the stress reactions is ultimately the goal, the variance in the distribution was also calculated. As can be seen in Figures 9b and 10b, the variance increases at larger strains. This increase of uncertainty is called heteroscedastic noise. This heteroscedasticity is much greater in the Gaussian dataset, reaching a variance of almost 0.7, compared to the uniform dataset, which reaches a variance of around 0.2.

To investigate the differences between the distributions a kernel density estimation of the data was performed for both datasets at strains $\varepsilon = [0, 0.375, 0.75, 1.125, 1.5]$. For these KDE's a Gaussian kernel is used with a bandwidth of 0.02. In Figure 11**,** and 12 these different KDE's can be seen, for the prescribed strains for both the uniform and Gaussian distributed RVE's. From these KDE's a number of different interesting observations can be made.

As can be seen in Appendix A4 The Gaussian dataset follows a multimodal distribution. There appears to be a normally distributed cluster of data points with low stress reactions at all strains. Under close inspection this clustering can also be seen in Appendix A2 Although we have no insight into the RVE's, this could be due to the Gaussian distribution of particles creating similar layouts of RVE's that react very similarly to each other. There are also two more distinct clusterings of the data. One at very high stress reactions that remains quite closely clustered at all strains, and another large group of datapoints, which get more and more stretched out as the strain increases.

Meanwhile the stress reactions from the RVE with uniformly distributed particles are grouped much more closely. This is reflected in the lower variance discussed before. The stress reactions from these RVE's are grouped about as closely as the central grouping in the Gaussian RVE's. At higher strains roughly 3 peaks begin to form in the distribution, with a large tail reaching towards higher stress reactions. There is also one clear outlier in this dataset reaching a stress of 26MPa which can also be very clearly seen in the Appendix A1 The outlier will be included in the training dataset, because it is only a single datapoint in the dataset and it will be unlikely that it affects the results too much [**25**]. Furthermore, including this outlier will give a better representation of the dataset, allowing the models to generalise better to unseen data.

3.2 Model Generation and Training

To perform the uncertainty quantification a number of points were sampled from the data and both a BNN and GP were trained on this data. The experiment was run using 10, 50 and 100 sampled data points. To sample these, random points were selected at every, every other, or every tenth value for the induced strain. These were then used to train the BNN and the GP.

The BNN that was used was built using Adam Cobb's Hamiltorch library. It consisted of an in and output layer consisting of one node, since the task is a 1-D regression task. And 2 hidden layers with 10 nodes each. The BNN architecture is based on the architecture written by Adam Cobb [**26**]. The architecture was not changed from the base architecture due to time constraints in the project. A graphical representation of the network can be seen in Figure 13. The priors were initialised to a $N(0,1)$ distribution. Training was then performed on these 10, 50 and 100 data points. Using a step size of 0.00011 and 1100 epochs, These hyperparameters were found to perform the best through a random search ranging the step size between 0.1 to 0.0001 in logarithmic steps and trying 100, 200, 500, 1000 and 1500 epochs. The hyperparameter 'Tau_out' was evaluated between 1 and 10, ideally a wider array would be more representative however this wasn't possible due to time and computational constraints. After this random search, a grid search to find the exact best hyperparameters was performed. During the grid and random search, the evaluating metric was always the negative log likelihood.
The resulting model was then sampled 1000 times. These 1000 samples were used to make a KDE, which in turn was used to perform the final validation and calculate the negative logarithmic likelihood of the experimental data being drawn from this distribution.

For the Gaussian Process the python GPy library was used. A kernel is defined, in this case a RBF kernel with an input dimension of 1. Then a model is defined with the training points X, labels y and the aforementioned kernel. Finally the model is restarted 10 times to optimise the kernel hyperparameters.

After finalising both models, they are trained on the generated data resulting in a probabilistic result. With these results a PDD is generated of both models. To make a comparison between these two different models the negative log likelihood is used. Computing this score for the BNN predictions was complicated due to the fact that it does not return any equation that the likelihood can be calculated on, this distribution is found experimentally instead. In order to get around this, a custom code was written to calculate the log likelihood. This code took all the experimental data points, and calculated their likelihood from the KDE, took the natural logarithm of this number and summed all of those. For the GP it was much more straightforward as it outputs a distribution from which it could calculate the log likelihood directly.

4 RESULTS AND DISCUSSION

This chapter will be dedicated to discussing the results from the KDE and the Bayesian models. This will include their performance compared to each other using log likelihood, the differences visually between all the models and reasoning why this may be the case.

The two models that were generated were able to make predictions as well as give uncertainty intervals about these predictions, the predictions and uncertainty intervals for the uniform dataset are shown in Appendix E1 For the Gaussian distributed dataset shown in Appendix E2 both the GP and BNN models have a low accuracy when using 10 training samples. This is most likely explained by the fact that these models are both quite flexible, and training on so few data points, this flexibility caused them to overfit. This causes the shapes of the predictions to not fit closely to the mean of the training data. Furthermore, as can be seen in Appendix B1.1 and Appendix B2.1 both models predict lower stresses than the actual data due to the sampled training data points having lower than average values, which is especially an issue with the Gaussian dataset due to it having quite a few data points that are well below the mean data.
With 50 samples the overfitting of the models becomes less prominent, however the models are still overfitting. This can be seen in Appendix B1.2, B2.2 and Appendix C1.2**,** C2.2 at a strain of 0.4 and

1.1, both models underestimate the mean of the data. When using 100 training samples, both models can be seen to perform quite well. As more points are sampled, the influence of outliers reduces drastically. The flexibility of these models is also more suitable for larger training sets, and overfitting can be observed to be less of an issue.

An interesting observation about the graphs of the BNN and the GP with the Gaussian datasets is that the size of the confidence interval depends on how many training points are sampled. At 10 training points the confidence interval of the GP is quite large compared to the confidence interval at 50 sampled points. At 100 sampled points the confidence interval has increased again. In Appendix B1.3 and Appendix B2.3 it can be observed that the sampled training points are more spread out, this could be the reason why the confidence interval gets wider with more training samples. The same concept happens with the BNN model, at 10 samples the confidence interval is relatively large, then at 50 samples the confidence interval decreases and at 100 samples the confidence interval increases again. Both the GP and BNN also use the same sampled data points, so it is logical that the confidence intervals roughly have the same size.

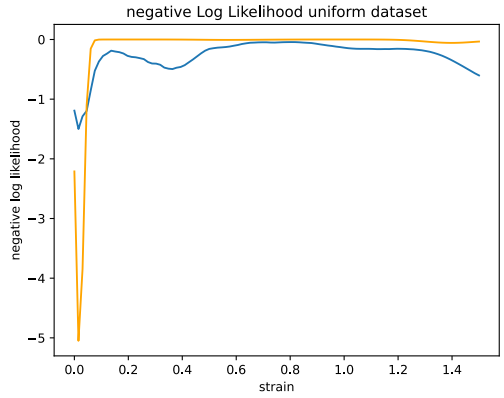
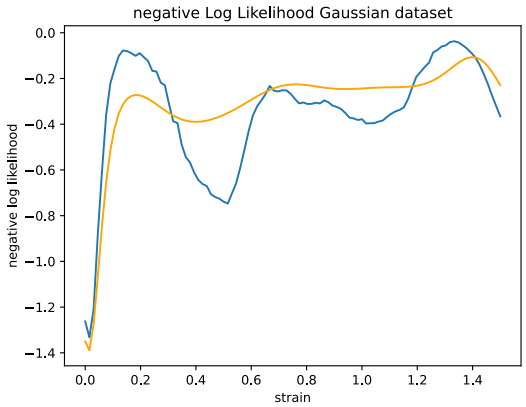
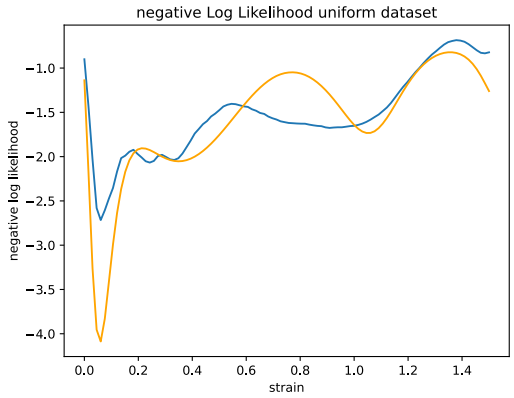
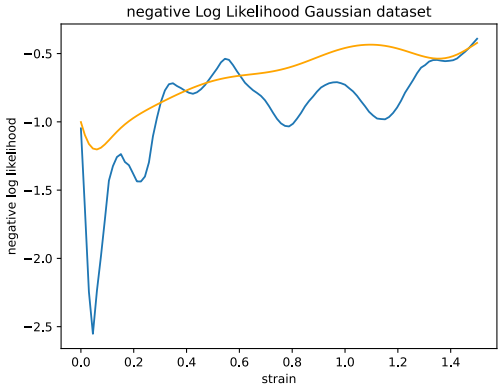
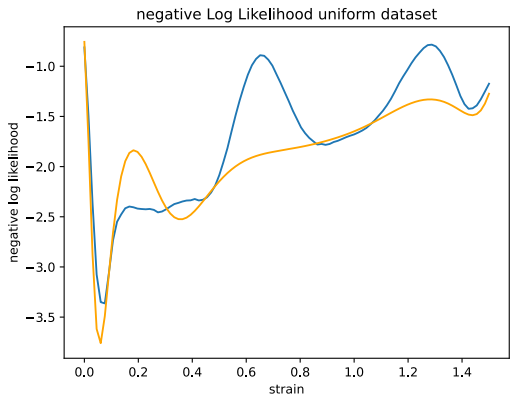
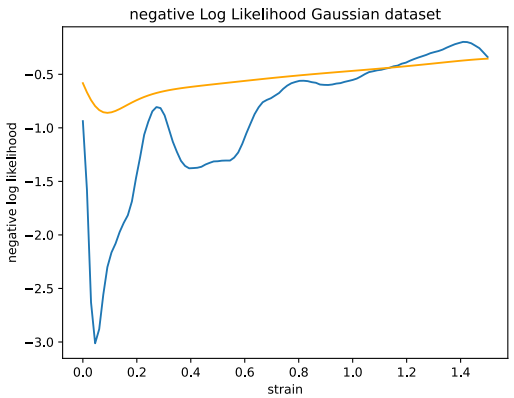
In the uniform dataset, with few training samples, a very similar issue arises as with the Gaussian dataset. The sampled training points just so happen to all be relatively low and the entire model predicts lower strains than it actually should. It can also be seen that the predicted curves vary quite a lot, this is due to the fact that there are few data points, so the model can't reach a high level of certainty about the shape of the dataset. The same pattern of vulnerability to outliers can be seen for the models when trained on 50 samples. For very high and very low strains the model fits quite well, but for strains between 0.4 and 1, both models underestimate the mean of the dataset. When trained with 100 data points both models perform much better.

When comparing the KDE of both the predicted and true uniform datasets, when 10 samples are used, in Appendix E1.1 the same pattern of underestimation can be seen in both the BNN predictions (blue) as the GP predictions (yellow line) at higher strains the models do seem to converge to the true distribution again. When trained with 50 and 100 training points, the models do seem to find the right mean, although the shape of the distribution and especially the variance of the BNN does not closely match that of the true dataset.

When comparing the true distribution of the Gaussian dataset with the predicted distributions, it becomes quite evident that both models are unable to properly predict the true distribution. For the GP this is easily explained by the fact that it can only predict normal distributions and the true distribution is very far from normal. In order to try and predict the very spread out data, the GP makes a very wide normal distribution. However, while most of the data does fall within the PPD made by the GP, the true distribution has spaces where no data points fall, making it less accurate. The BNN remains far more closely grouped at all levels of strain than the true distribution. The very wide normally distributed PPD of the GP makes it difficult to see, however, for the BNN the pattern of underestimation found in Appendix B1.1, can also be seen in the KDE in Appendix E2.1.

Table 1. negative log-Likelihood with uniform dataset (left) and Gaussian dataset (right).

Orange is GP & blue is BNN

10 samples 	*10 samples* 
50 samples 	*50 samples* 
100 samples 	*100 samples* 

A trend that can be immediately observed when looking at the negative log likelihood of the predictions is that both models, on all numbers of training points, perform the best when the induced strain is very low. In the Gaussian dataset this trend is particularly pronounced in the BNN model, whereas in the uniform dataset it is more pronounced in the GP. This is likely due to the fact that the predictions of the BNN tend to stay quite close together but are not necessarily normally distributed.

This allows them to perform well on the Gaussian dataset, which is very much not normally distributed, but only at low strains, where the stress reactions are still closely grouped. The GP does this better at low strains in the uniform dataset, as in this portion the data is still largely normally distributed. Another observation is that both models perform significantly better on the uniform dataset than the Gaussian dataset. This is due to the wide multimodal distribution, and the observations made earlier in this chapter. This is true for models trained on 50 and 100 data points, while the GP trained on 10 data points of the uniform dataset, after a brief segment where it works well, is essentially useless. Another observation that can be made about the Gaussian dataset is that the GP has a curve which is a lot smoother than the BNN, especially when only 10 samples were used to train it. However the average scores of the BNN are better than those of the GP. Furthermore when looking at the uniform dataset with 100 samples multiple peaks are seen in the BNN curve. These variations in performance can be attributed to the predicted distribution getting wider and narrower. Under close inspection of Appendix C1.3 it can be observed that while the predicted mean follows the true distribution quite well, the individual predictions don't follow a smooth curve and are quite noisy. This results in the PPD varying quite a bit at different strains.

5 CONCLUSION

The goal of this report was to investigate the effect of different random sampling methods of particles in an RVE for computational homogenization predictions of uncertainty for two different Bayesian models.

The results show that sampling the particles with the different distributions gives varying results for the computational homogenization predictions. The two models also give varying uncertainties which makes it hard to say what the eventual effect will be. Looking at the negative log likelihood, the BNN model performs better than the GP model on the Gaussian dataset. This could be the case because the Gaussian dataset does not follow a normal distribution, the GP model has more difficulties predicting such a distribution. On the uniform dataset the GP model performs a little bit better. However, both models perform worse on higher strains. This trend is visible in both datasets, this can be attributed to the fact that at higher strains the stress distributions become less normally distributed and thus more difficult for the GP to predict. The variance of the data also becomes larger, this makes it more difficult for the BNN to predict the distribution as it is limited by its hyperparameters. Furthermore the performance of the models is more accurate when more samples are used for training the models, this holds for both datasets as well, and can likely be attributed to the flexibility of both models being better suited to a larger training set.

A way to increase the performance of the BNN is tuning more hyperparameters. During this experiment only the number of epochs, the learning rate and 'Tau_out' were experimented with. However, it would benefit the BNN model if a more extensive grid search could be done on these and more hyperparameters of the BNN model and also more tuning on the architecture of the BNN. When the BNN model is tuned and making better predictions it will become more clear what the eventual effect of the random sampling will be. Furthermore, for the GP a RBF kernel is used, while other kernels or combinations were not considered. In the future any other type of kernels or combinations could be used, in order to get better predictions. Besides that the models are now only predicting the stress with only a one dimensional strain, while in the future the model could also be expanded to predict the stress when the strain is put on a three dimensional structure. Moreover the KDE's that were produced only looked at a few specific strains, if the KDE's would contain more strain values more insight could be obtained about the datasets and the predictions of both models.

6 GROUP PROCESS EVALUATION

6.1 Reflection process

Because the project required two different models which both required two different types of extra reading material тhe group quickly split up into two groups. One group mainly worked on the Gaussian Processes and the other group mainly worked on the BNN. This made it easier to look deeper into a subject after which a group member could explain it to the rest of the group. A mistake made earlier on in the project was being too eager to practically implement the model without exactly knowing what should be done. This mistake was made due to a misunderstanding between what we thought was the goal of the project and what the clients really wanted. Due to these misunderstandings the project started to fall behind schedule. This was resolved by having a long conservation with the clients about their expectations for this project. The lesson learned from this is to not make assumptions about the clients expectations and to create a better visualisation of their expectations / needs earlier on by more efficient and in general more communication.

6.1.1 Work organisation

The organisation of the project group can be divided in three different topics: general planning, information sharing and communication and lastly coding/ code sharing. These three will be tackled separately below.

6.1.2 General planning

For the general planning an application called ‘online gantt’ was used. This application made it easy to make a gantt chart and created a good visualisation of what needed to be done each week.

6.1.3 Information sharing and communication

There were different types of communication in this project. There were necessary meetings with the clients, meetings with the TA and meetings with the group itself. Most of the communication with the clients was done via mail using outlook and via a weekly meeting with the clients. If it was necessary to have an extra physical meeting this was communicated at the physical meeting or again using mail.

For communication with the TA two methods were used. There were two weekly physical meetings with the TA and other questions were asked using either WhatsApp or discord.

Communication and information sharing between group members was done throughout the week via either planned physical meetings or online meetings on discord. All group members were also present at the weekly TA meeting for further discussion except when noted beforehand to the group.

6.1.4 Coding / code sharing

For the coding two different types of applications were used. Pycharm was used to work on the code and github was used to store, share and access the code. These two work well together because pycharm has a built in add-on for github which makes using both more fluent. Github makes it easy to work on code, because it keeps track of the different versions and if a version breaks it is easy to go back to a version that did work.

6.2 Requirement validation

- M1 - (Completed): This requirement finished and explained in this paper using various citations
- M2 - (Completed): This requirement is finished and the results are shown in this paper and the code itself is attached in the appendix
- M3 - (Completed): In this paper multiple sources are cited to help explain the used methods therefore this requirement is completed
- M4 - (Completed): In the result section there are multiple negative log likelihood graphs on which is reflected further in the results therefore this requirement is finished.
- S1 - (Completed): Same as M4 this requirement can also be found in the results section of this paper where. Here is reflected on the models using the negative log likelihood.
- S2 - (Completed): Same as M4 this requirement can be found in the results section of this paper. Here is reflected on the data and can heteroscedastic noise be observed.
- C1 - (Not Completed): Because of the miscommunications in the project there was not enough time to also extend the model to a 3D input-output problem. Therefore this requirement is not completed.
- W1 - (Completed): The clients provided the data therefore no data generation was needed.
- W2 - (Completed): In the project no machine learning algorithms were implemented from scratch.

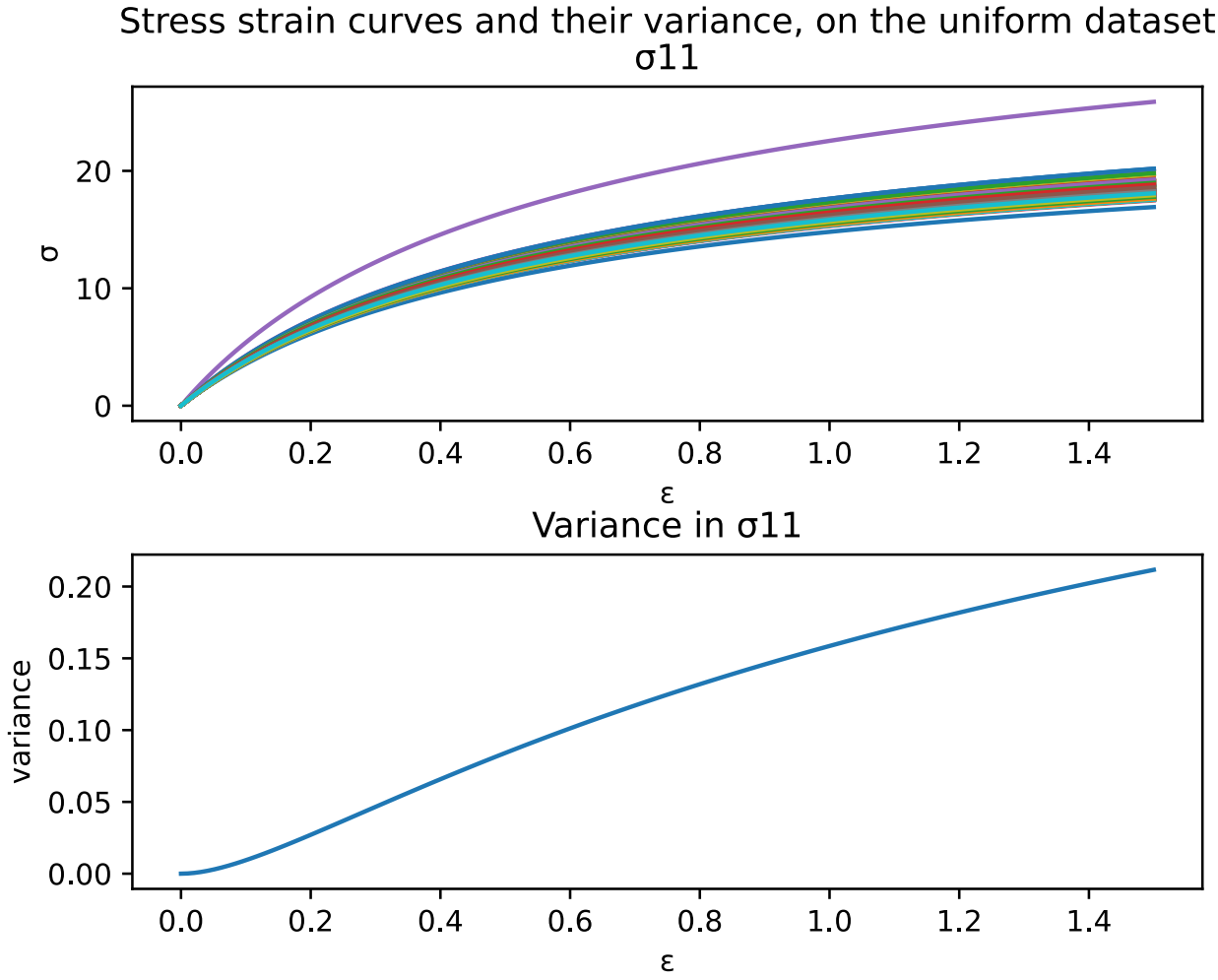
7 BIBLIOGRAPHY

[1] *Detailed Explanation of the Finite Element Method (FEM)*. (2017). Comsol. https://www.comsol.com/multiphysics/finite-element-method
[2] *What Is FEA | Finite Element Analysis? Documentation*. (2022, December 27). SimScale. https://www.simscale.com/docs/simwiki/fea-finite-element-analysis/what-is-fea-finite-element-analysis/
[3]*The Finite Element Method (FEM Simulation)*. (n.d.). Merkle. https://www.merkle-partner.de/en/competencies/fem-simulation
[4] Turan, T., & Kato, Y. (n.d.). Projectforum. https://projectforum.tudelft.nl/course_editions/56/generic_projects/1914
[5]van der Meer, F. P. (2016). Micromechanical validation of a mesomodel for plasticity in composites. *European Journal of Mechanics - a/Solids*, *60*, 58–69. https://doi.org/10.1016/j.euromechsol.2016.06.008
[6] Yvonnet, J. (2019). Computational Homogenization of Heterogeneous Materials with Finite Elements. *Solid Mechanics and Its Applications*. https://doi.org/10.1007/978-3-030-18383-7
[7] Turan, T. (2020). Surrogate Constitutive Models with Multi-fidelity Gaussian Processes for Composite Micromodels. TU Delft. https://repository.tudelft.nl/islandora/object/uuid%3A7438705a-da39-4f21-b8c0-8ee57faa47c1
[8] Nguyen, V. P., Stroeven, M., & Sluys, L. J. (2011). MULTISCALE CONTINUOUS AND DISCONTINUOUS MODELING OF HETEROGENEOUS MATERIALS: A REVIEW ON RECENT DEVELOPMENTS. *Journal of Multiscale Modelling*, *03*(04), 229–270.
[9] Chernatynskiy, A., Phillpt, S. R., & LeSar, R. (2013). Uncertainty Quantification in Multiscale Simulation of Materials: A Prospective. Annual review, vol. 43, 157-182. https://www.annualreviews.org/doi/pdf/10.1146/annurev-matsci-071312-121708#article-denial
[10] Honarmandi, P., & Arróyave, R. (2020). Uncertainty Quantification and Propagation in Computational Materials Science and Simulation-Assisted Materials Design. Integr Mater Manuf Innov 9, 103-143. https://doi.org/10.1007/s40192-020-00168-2 https://doi.org/10.1007/s10994-021-05946-3
[11] Möller, B., & Beer, M. (2008). Engineering computation under uncertainty—capabilities of non-traditional models. Computers & Structures, 86(10), 1024-1041. https://doi.org/10.1016/j.compstruc.2007.05.041
[12] Hüllermeier, E., Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Mach Learn* 110, 457–506.

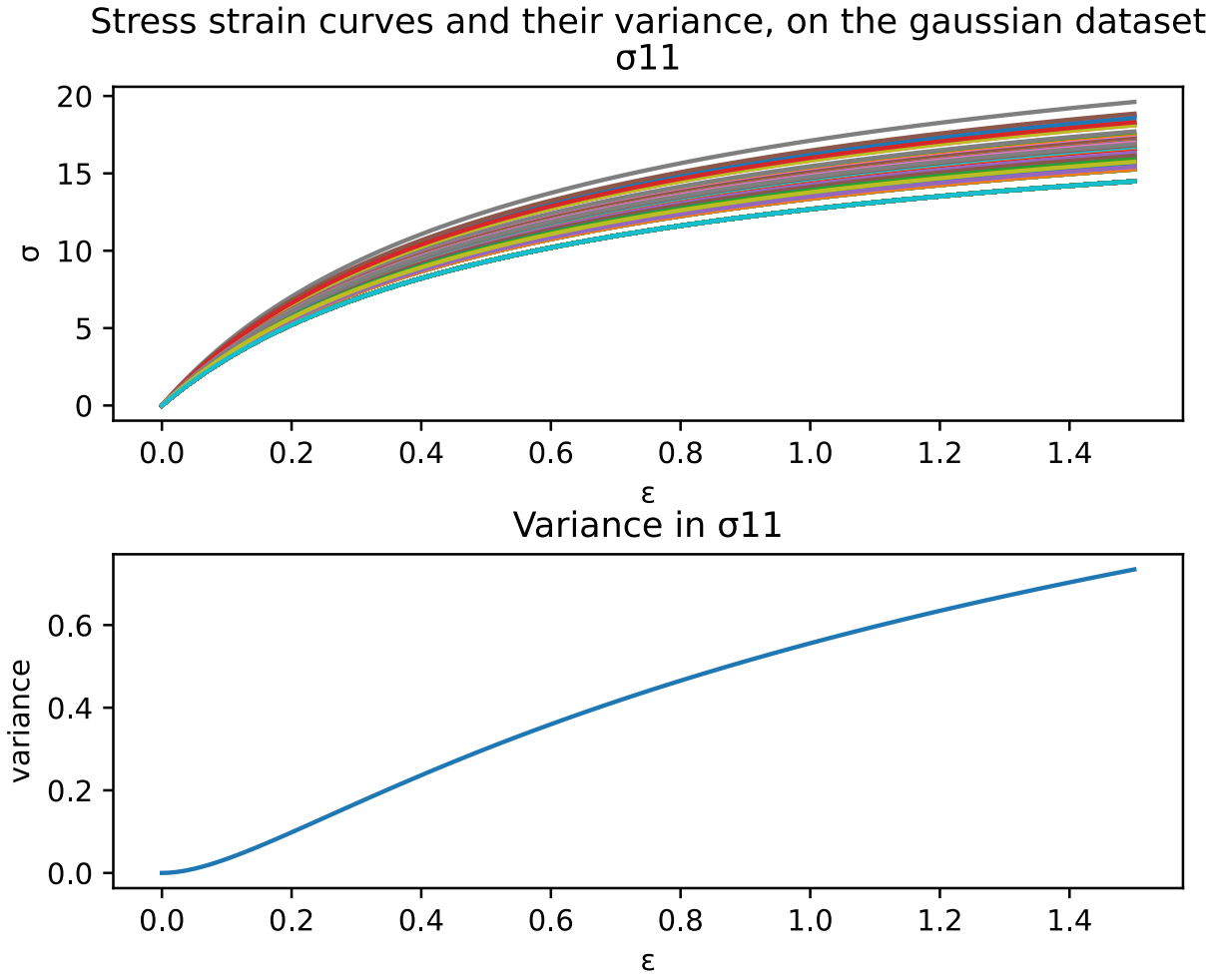
[13] Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., & Rubin, D. B. (2013). *Bayesian Data Analysis, Third Edition*. Taylor & Francis.
[14] Bishop, C. M. (2023). *Pattern Recognition and Machine Learning (text only) 2nd(Second) edition BY C.M. Bishop* (2nd printing edition). Mixture Density Networks.
[15] Görtler, J., Kehlbeck, R & Deussen, O. (2019). A Visual Exploration of Gaussian Processes, Distill. https://distill.pub/2019/visual-exploration-Gaussian-processes/
[16] Rasmussen, C. E., & Williams, C. K., I. (2005). *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning series)*. The MIT Press.
[17] Hands-on Bayesian Neural Networks – A Tutorial for Deep Learning Users
[18] L. V. Jospin, H. Laga, F. Boussaid, W. Buntine and M. Bennamoun, Hands-On Bayesian Neural Networks—A Tutorial for Deep Learning Users, in IEEE Computational Intelligence Magazine, vol. 17, no. 2, 29-48, May 2022, doi: 10.1109/MCI.2022.3155327.
[19] Ahamed, S. (2021, December 27). *Bayesian Neural network.* Towards Data Science. https://towardsdatascience.com/Bayesian-neural-network-7041dd09f2cc
[20] H. Cho, Y. Kim, E. Lee, D. Choi, Y. Lee and W. Rhee, Basic Enhancement Strategies When Using Bayesian Optimization for Hyperparameter Tuning of Deep Neural Networks, in IEEE Access, vol. 8, pp. 52588-52608, 2020, doi: 10.1109/ACCESS.2020.2981072.
[21] Libretexts. (2022, May 22). *13.5: Bayesian Network Theory*. Engineering LibreTexts. https://eng.libretexts.org/Bookshelves/Industrial_and_Systems_Engineering/Book:_Chemical_Process_Dynamics_and_Controls_(Woolf)/13:_Statistics_and_Probability_Background/13.05:_Bayesian_network_theory
[22] Do, C. B. (2007, December 1). *Gaussian processes*. Stanford. https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwj7grHI3dX8AhVXi_0HHUryCsIQFnoECAoQAQ&url=https%3A%2F%2Fsee.stanford.edu%2Fmaterials%2Faimlcs229%2Fcs229-gp.pdf&usg=AOvVaw31aywiYUYdIlBk1-1dfaUW
[23] Rogozhnikov, A. (2016). Hamiltonian Monte Carlo explained. Brilliantly wrong. https://arogozhnikov.github.io/2016/12/19/markov_chain_monte_carlo.html
[24] Rivlin, R.. (1948). Large Elastic Deformations of Isotropic Materials. IV. Further Developments of the General Theory. Philosophical Transactions of The Royal Society A: Mathematical, Physical and Engineering Sciences. 241. 379-397. 10.1098/rsta.1948.0024.
[25] Geron, A. (2019). Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow. O'Reilly Media.
[26] Cobb, A. (2020). Scaling HMC to larger data sets. Adam Cobb. https://adamcobb.github.io/journal/bnn.html

APPENDIX

A1

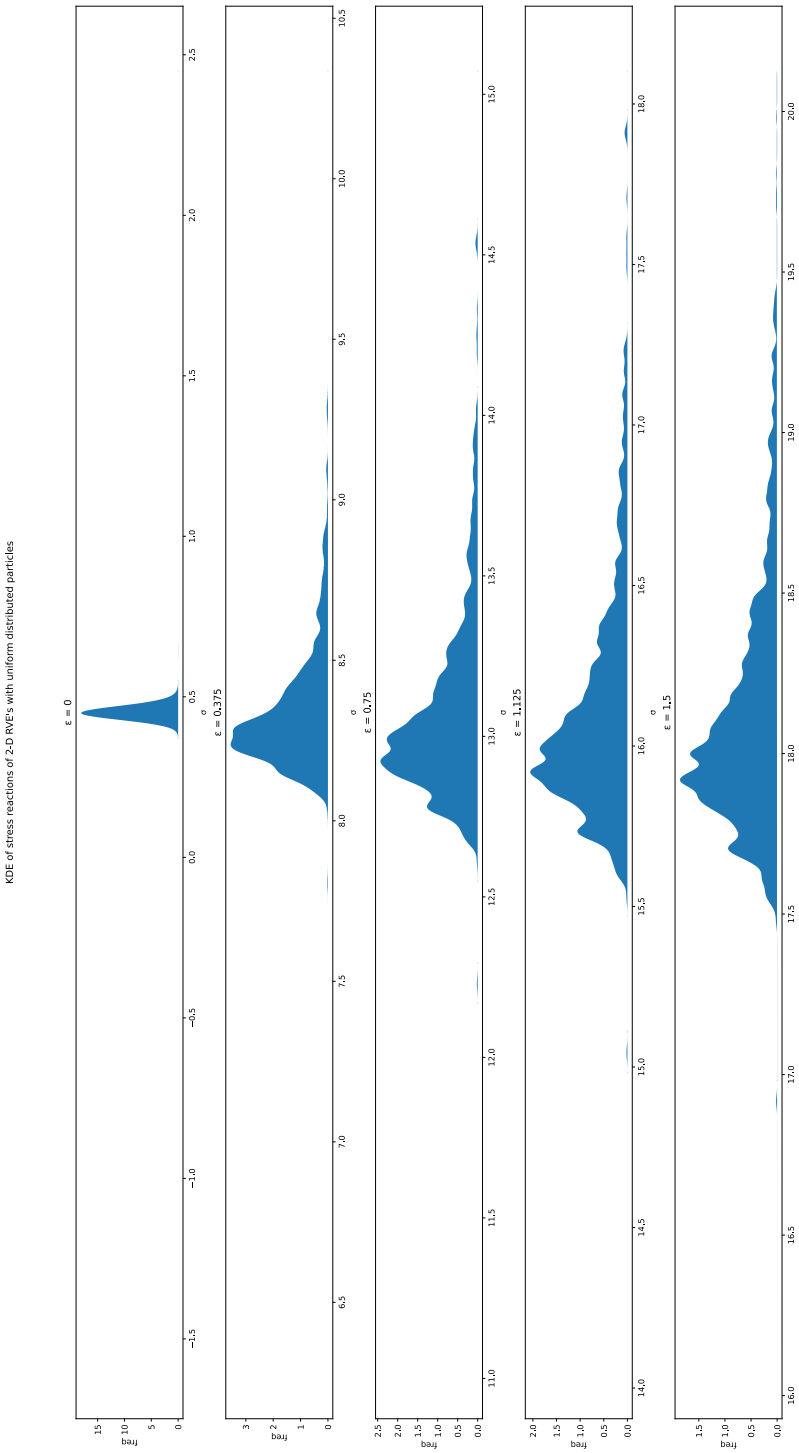

Stress strain curves and their variance, on the uniform dataset
σ11
σ
20
10
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
ε
Variance in σ11
variance
0.20
0.15
0.10
0.05
0.00
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
ε

A2

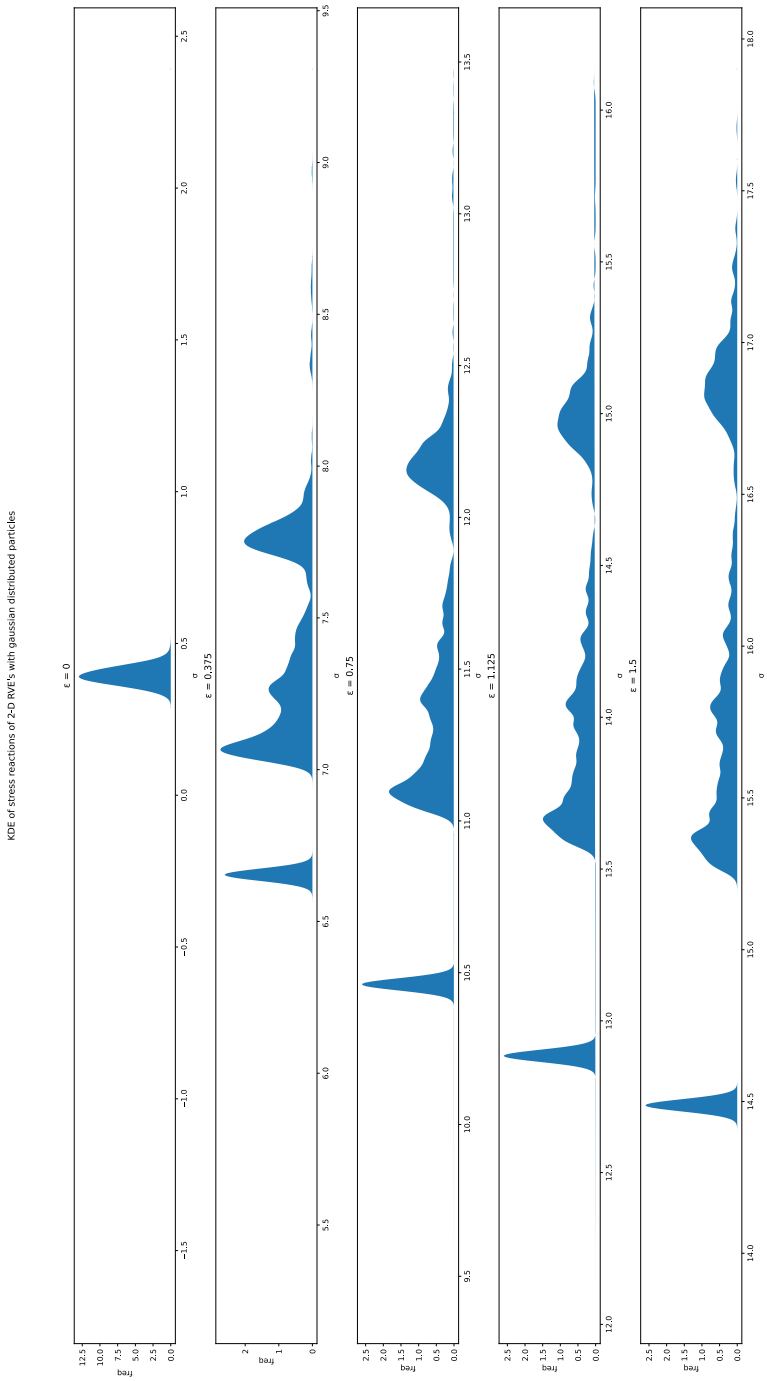

Stress strain curves and their variance, on the gaussian dataset
σ11
σ
ε
Variance in σ11
variance
ε

\

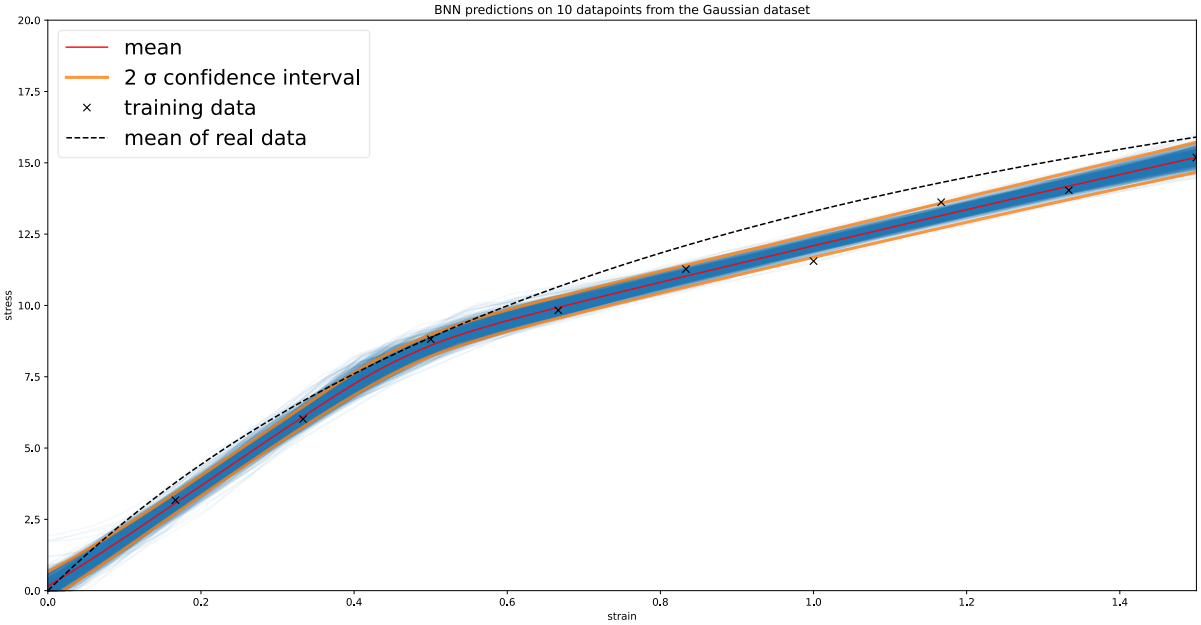
A3


KDE of stress reactions of 2-D RVE's with uniform distributed particles
ε = 0
ε = 0,375
ε = 0,75
ε = 1,125
ε = 1,5
freq
σ

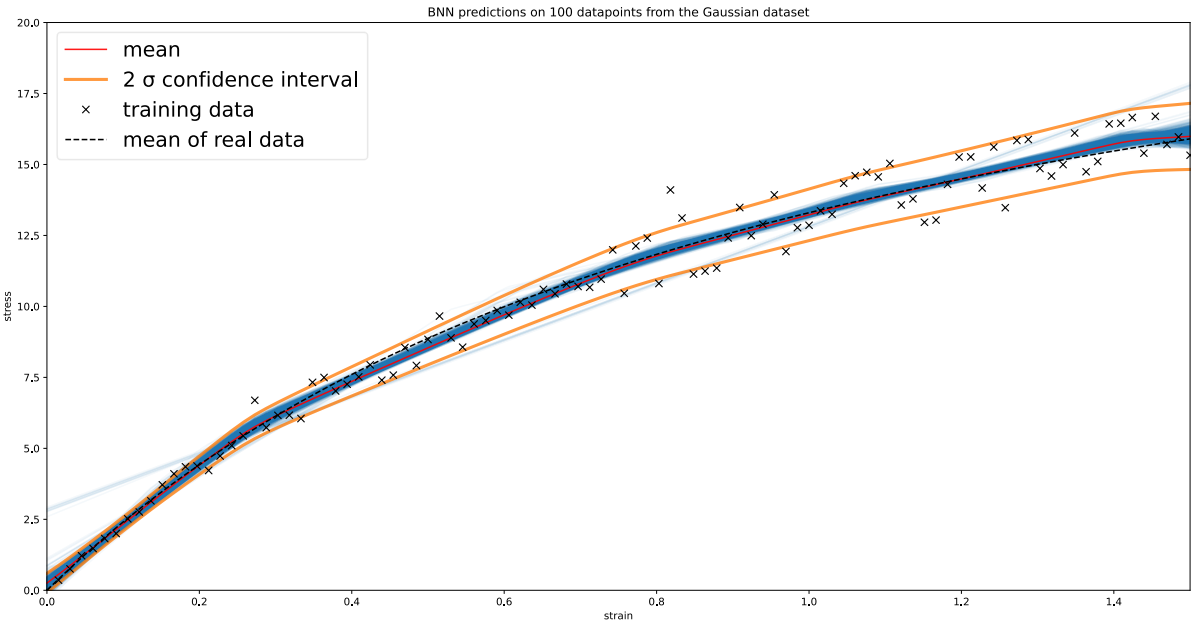
A4


KDE of stress reactions of 2-D RVE's with gaussian distributed particles
ε = 0
ε = 0,375
ε = 0,75
ε = 1,125
ε = 1,5
σ
freq

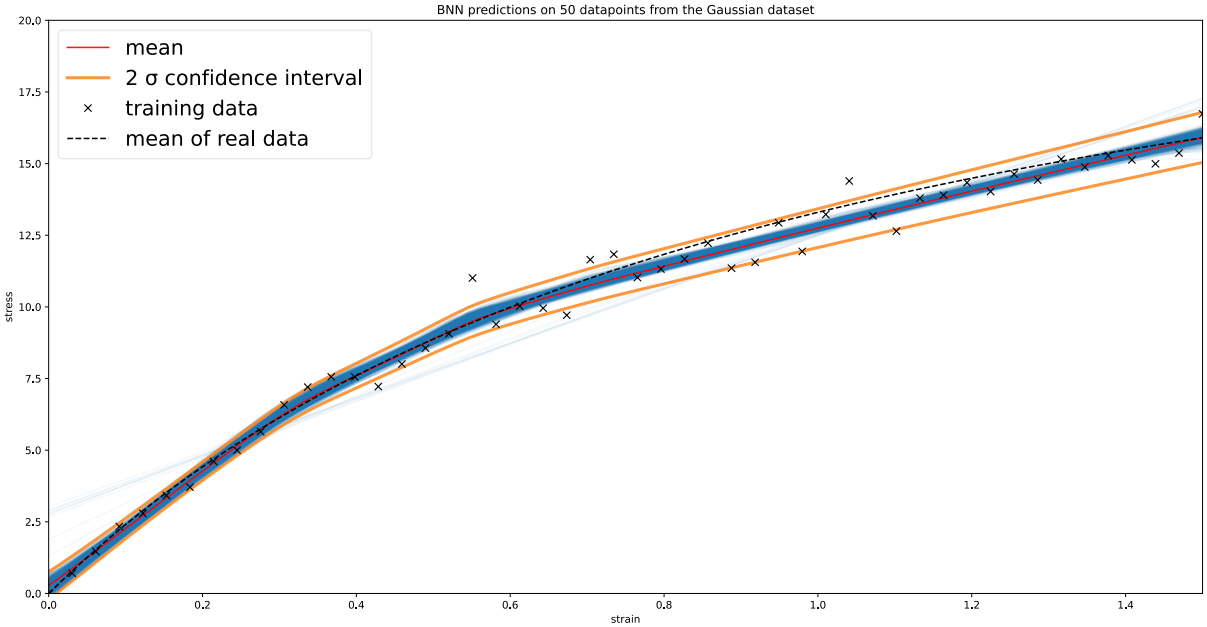
B1.1


BNN predictions on 10 datapoints from the Gaussian dataset
mean
2 σ confidence interval
training data
mean of real data
stress
strain
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

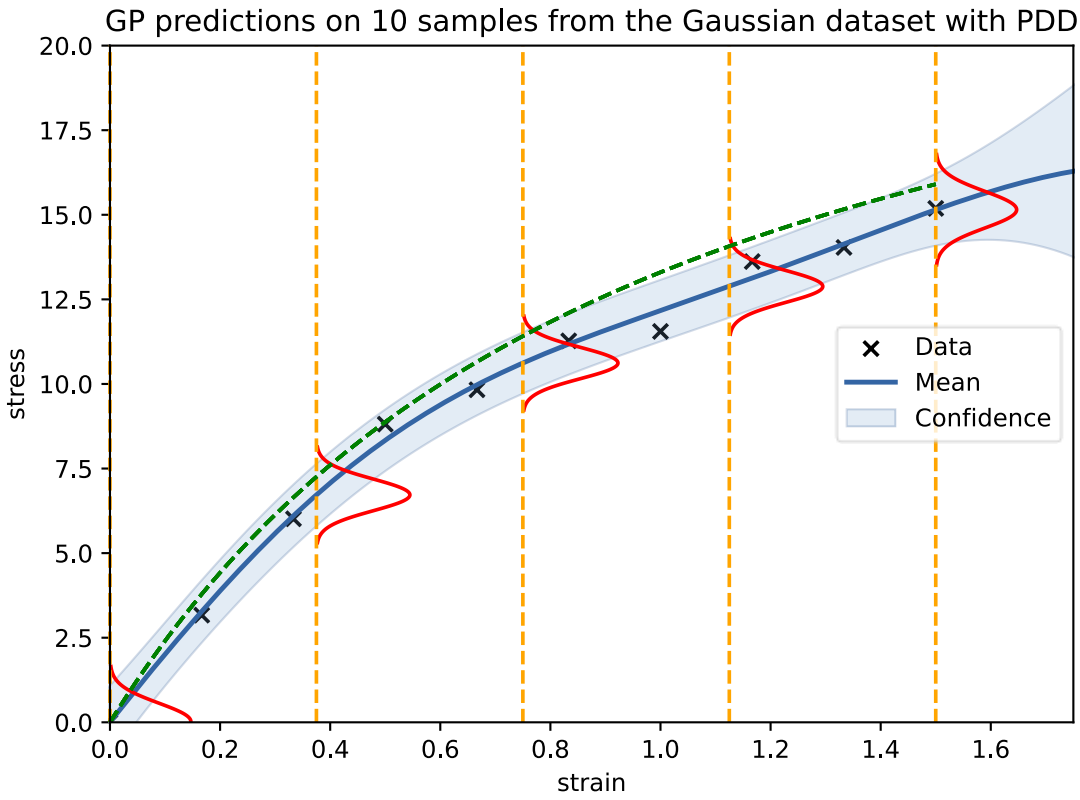
B1.2


BNN predictions on 100 datapoints from the Gaussian dataset
mean
2 σ confidence interval
training data
mean of real data
stress
strain
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

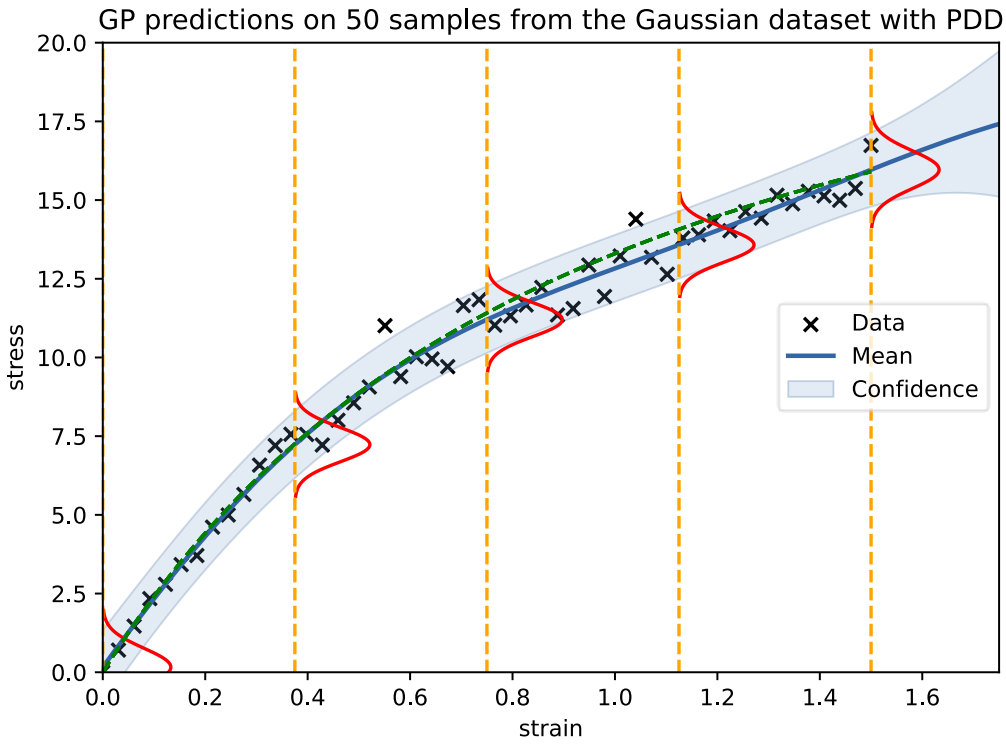
B1.3


BNN predictions on 50 datapoints from the Gaussian dataset
mean
2 σ confidence interval
training data
mean of real data
stress
strain

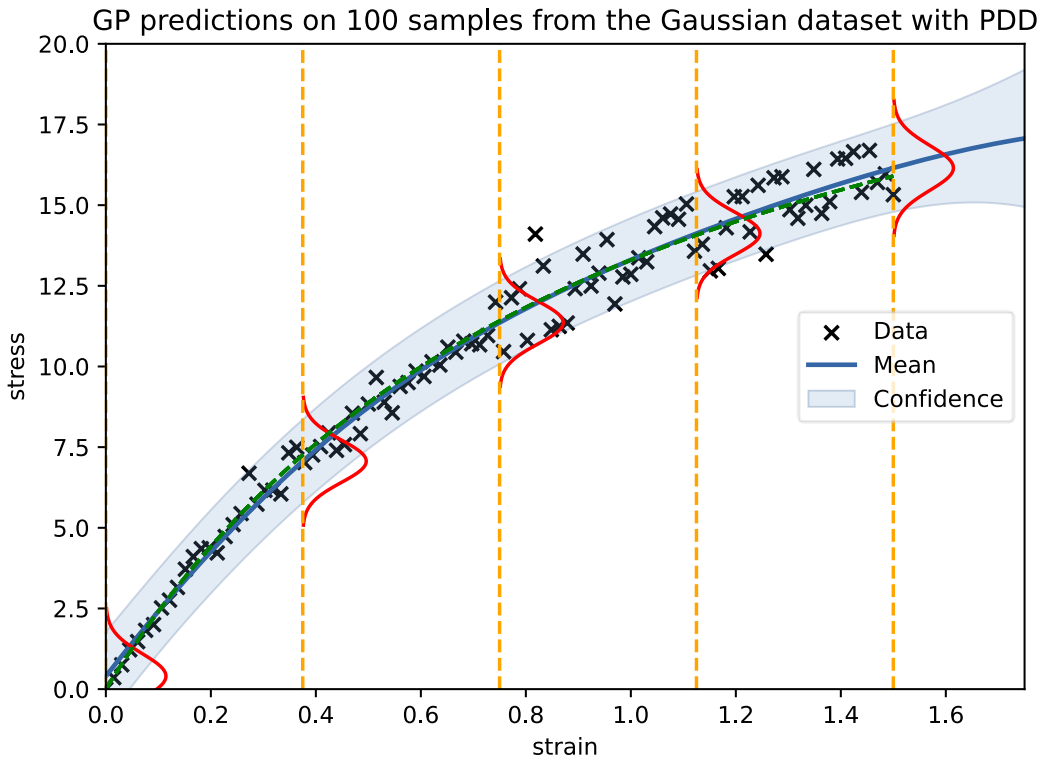
B2.1


GP predictions on 10 samples from the Gaussian dataset with PDD
Data
Mean
Confidence
stress
strain

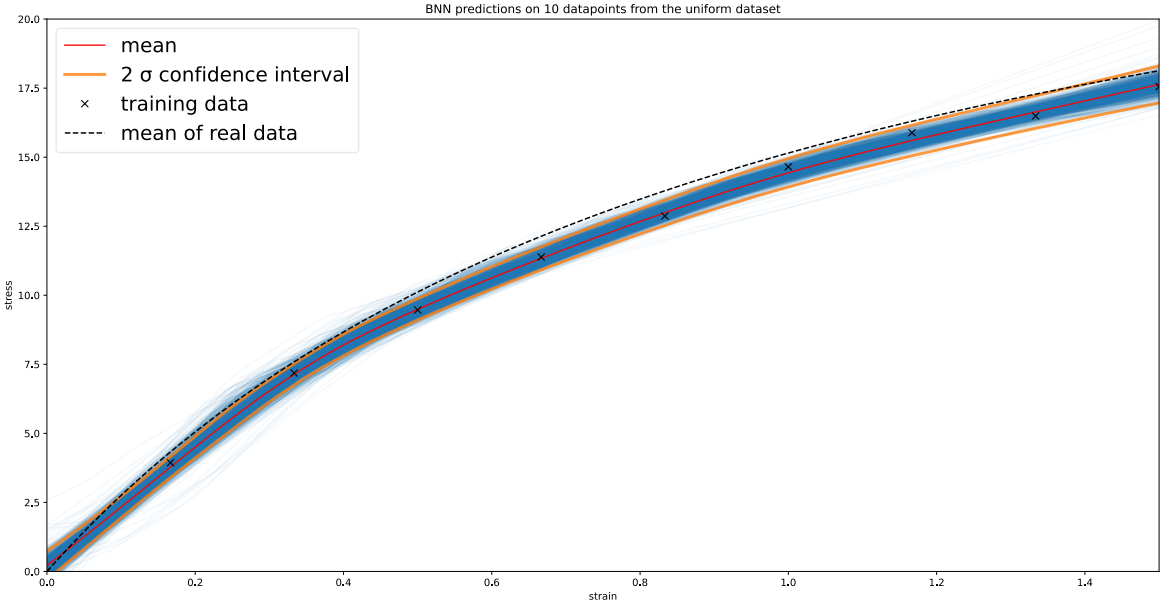
B2.2


GP predictions on 50 samples from the Gaussian dataset with PDD
stress
strain
Data
Mean
Confidence

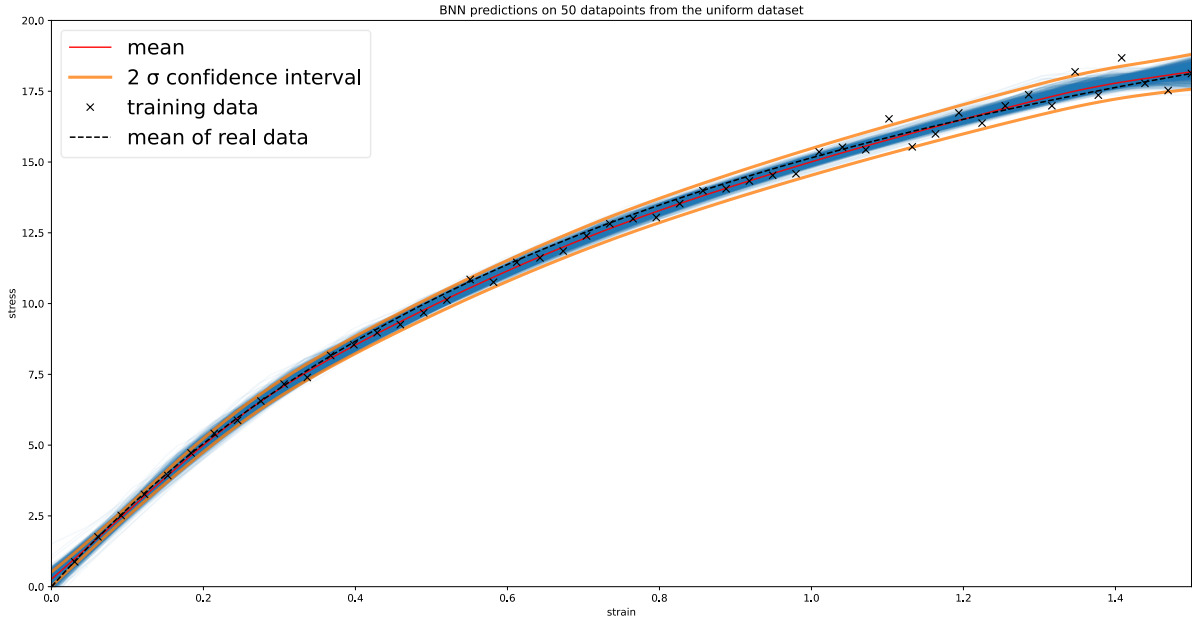
B2.3


GP predictions on 100 samples from the Gaussian dataset with PDD
stress
strain
Data
Mean
Confidence

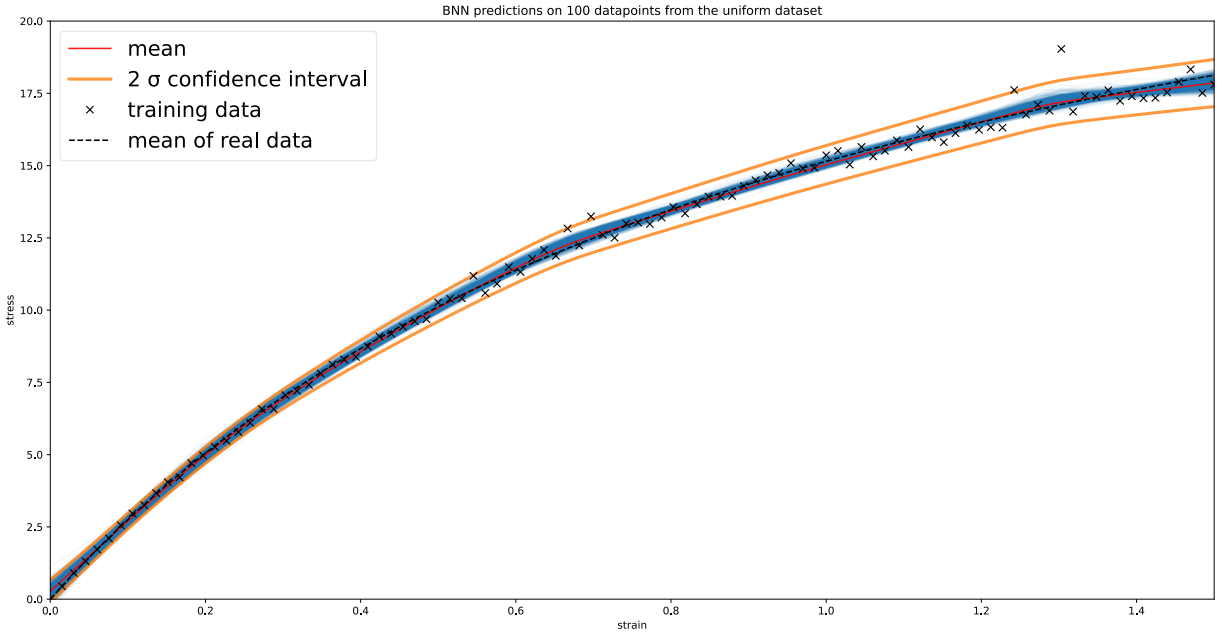
C1.1


BNN predictions on 10 datapoints from the uniform dataset
mean
2 σ confidence interval
training data
mean of real data
stress
strain
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

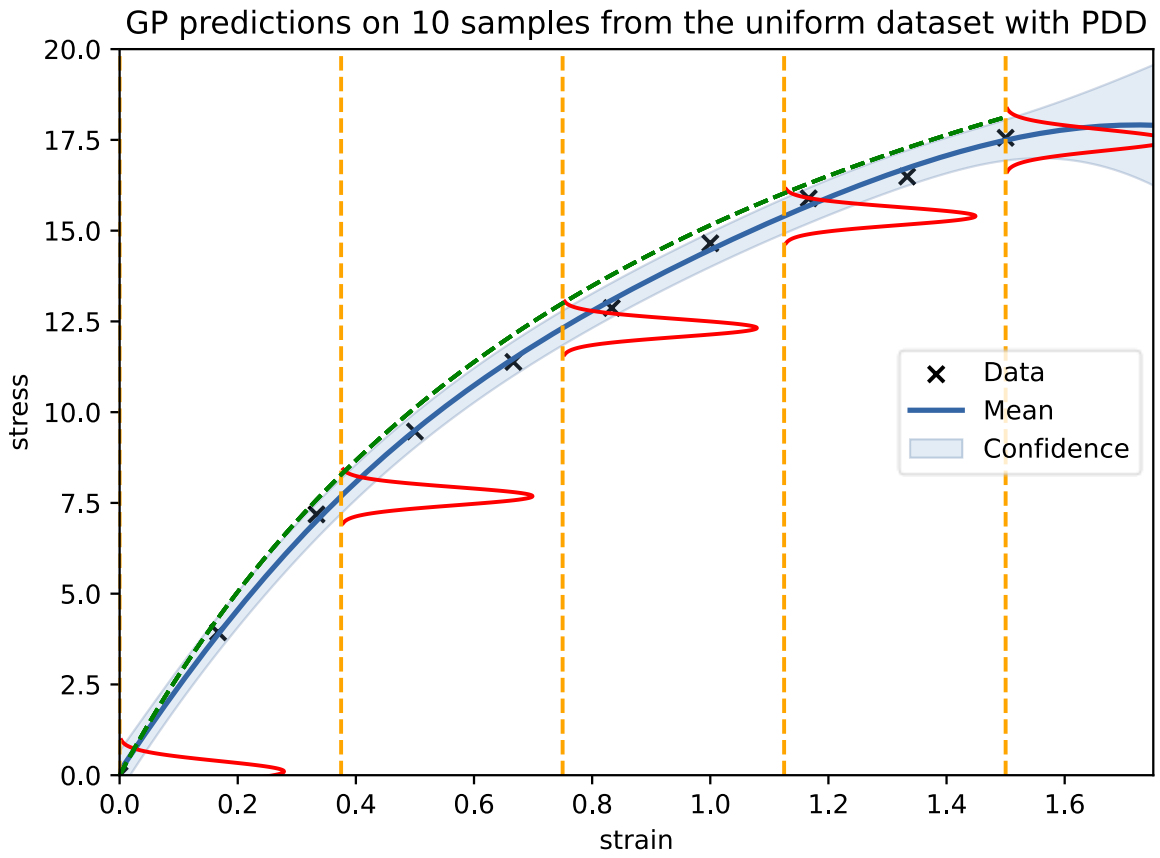
C1.2


BNN predictions on 50 datapoints from the uniform dataset
mean
2 σ confidence interval
training data
mean of real data
stress
strain
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4

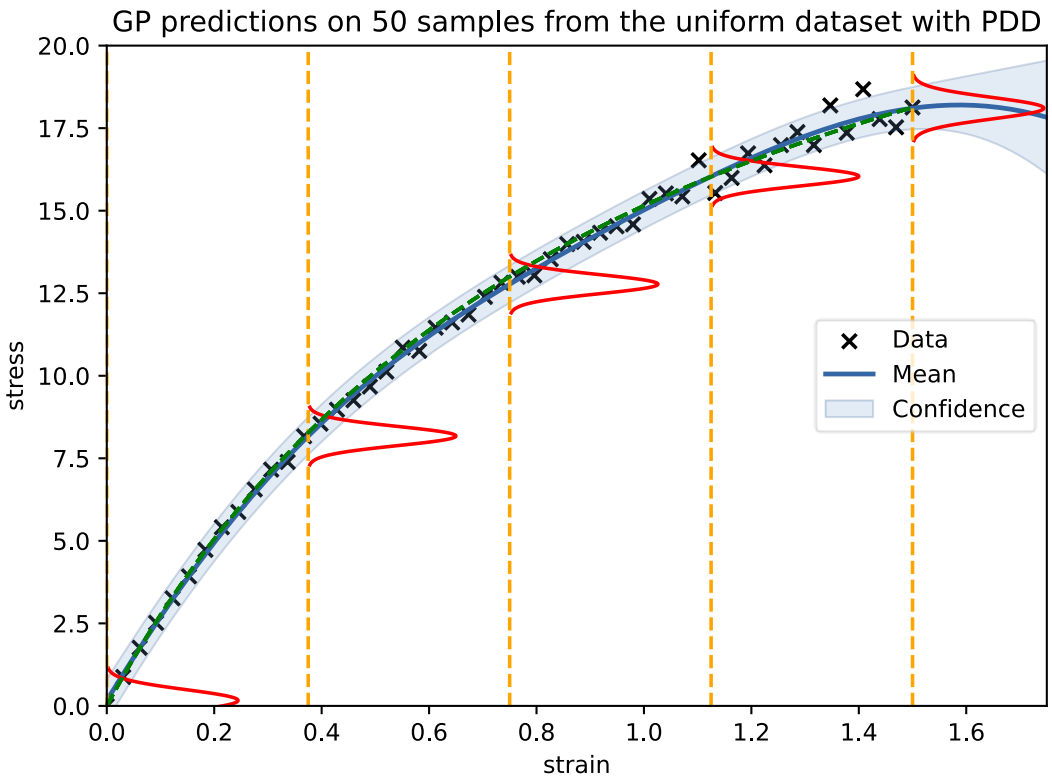
C1.3


BNN predictions on 100 datapoints from the uniform dataset
mean
2 σ confidence interval
training data
mean of real data
stress
strain

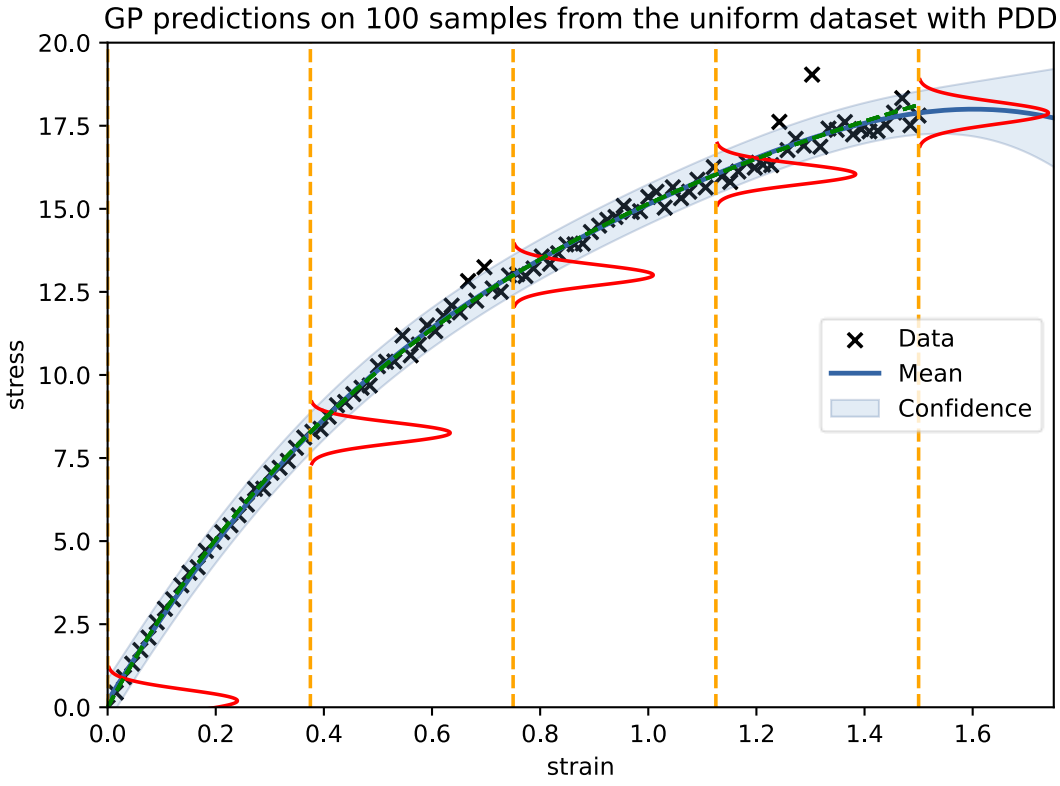
C2.1


GP predictions on 10 samples from the uniform dataset with PDD
Data
Mean
Confidence
stress
strain

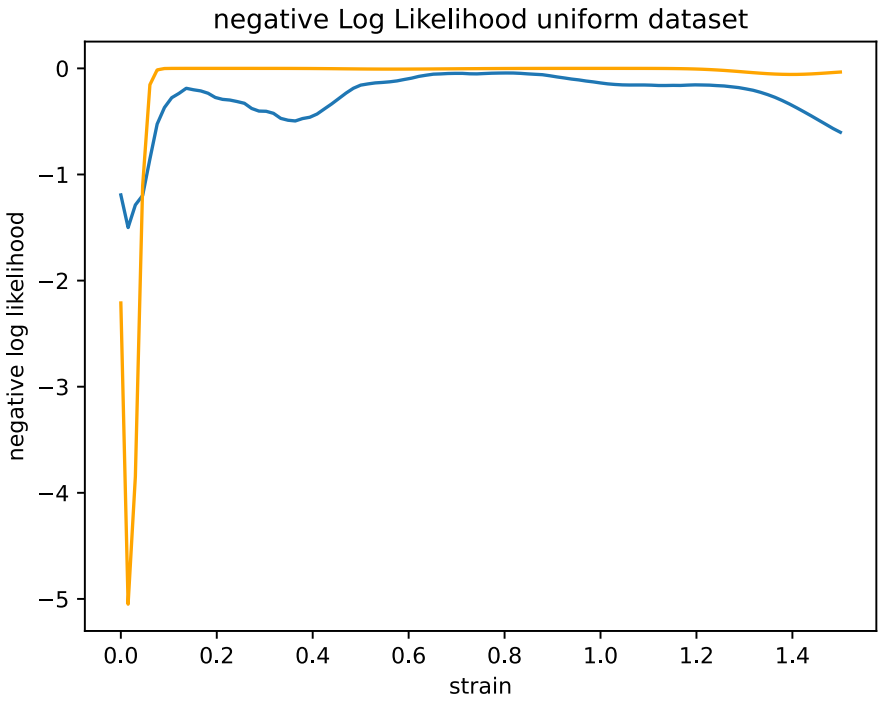
C2.2


GP predictions on 50 samples from the uniform dataset with PDD
stress
strain
Data
Mean
Confidence

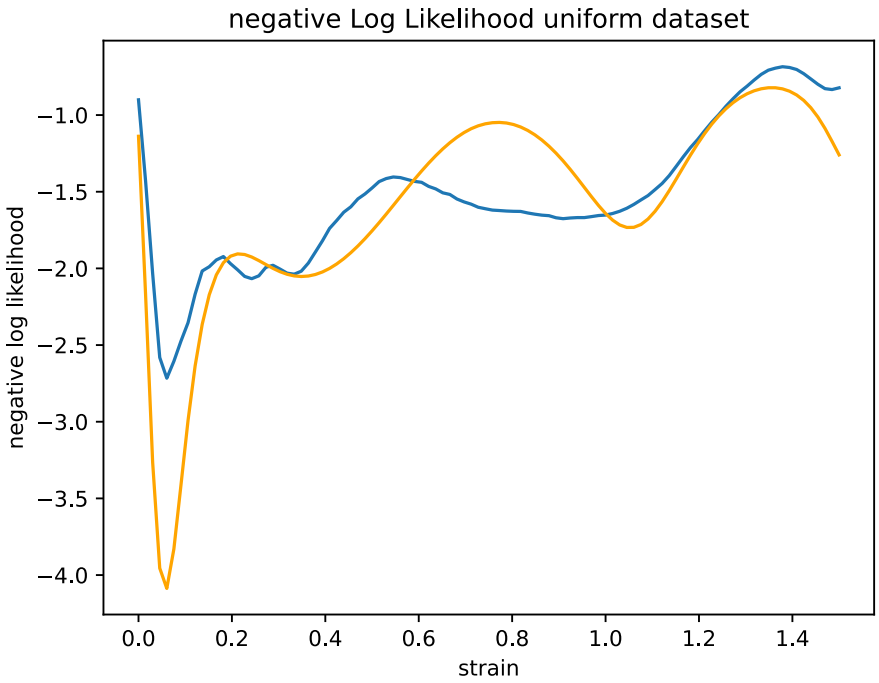
C2.3


GP predictions on 100 samples from the uniform dataset with PDD
stress
strain
Data
Mean
Confidence

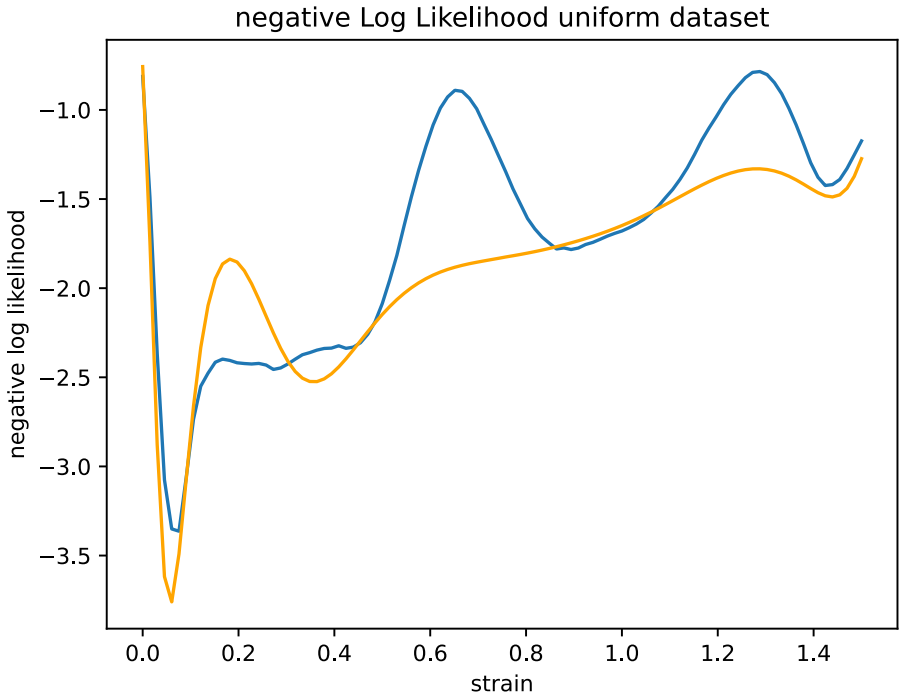
D1.1


negative Log Likelihood uniform dataset
negative log likelihood
0
−1
−2
−3
−4
−5
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
strain

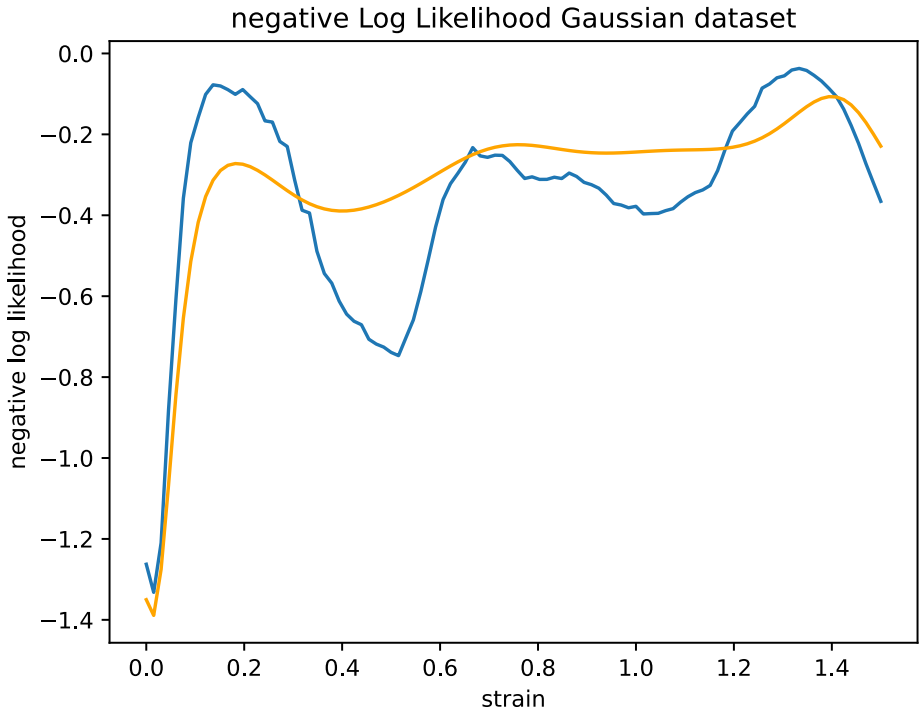
D1.2


negative Log Likelihood uniform dataset
negative log likelihood
−1.0
−1.5
−2.0
−2.5
−3.0
−3.5
−4.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
strain

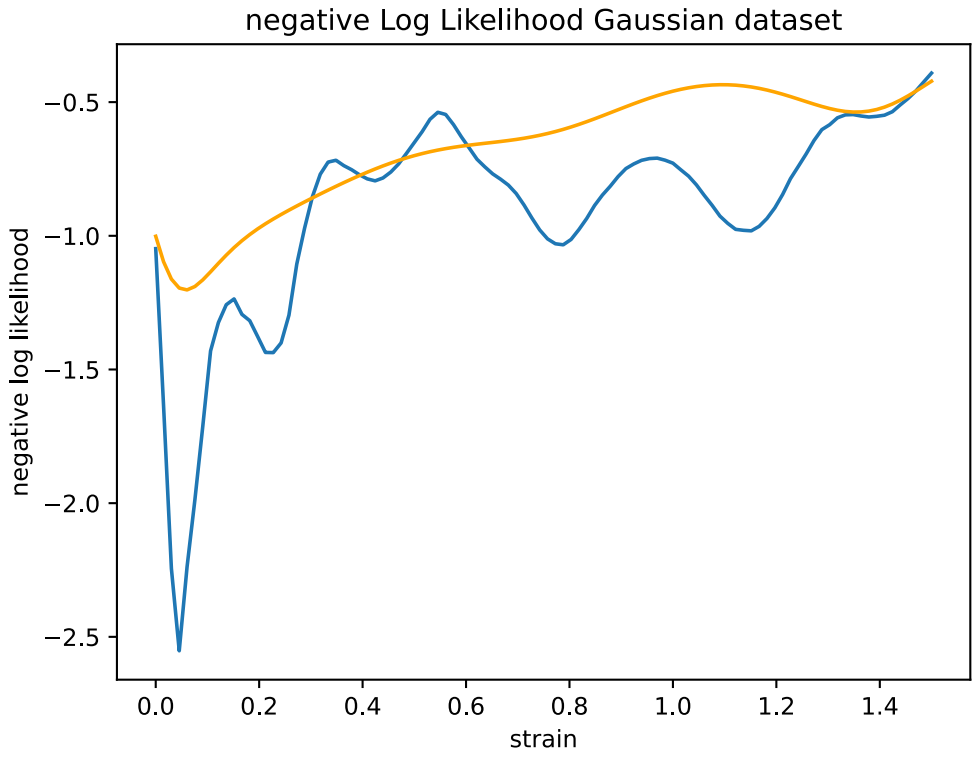
D1.3


negative Log Likelihood uniform dataset
negative log likelihood
−1.0
−1.5
−2.0
−2.5
−3.0
−3.5
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
strain

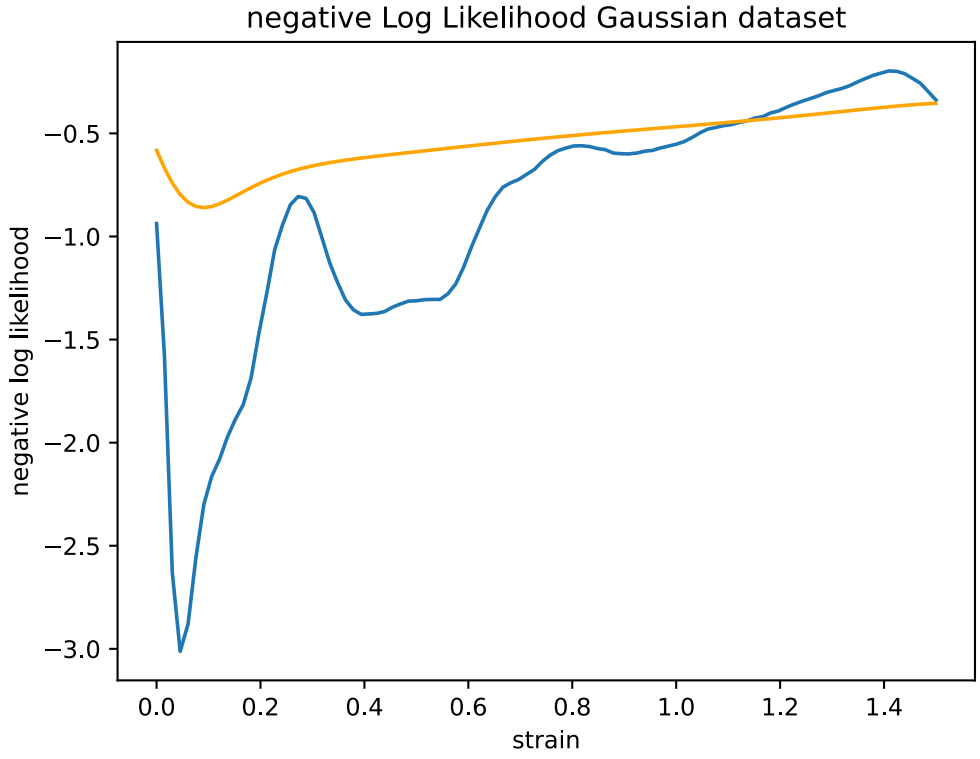
D2.1


negative Log Likelihood Gaussian dataset
negative log likelihood
0.0
−0.2
−0.4
−0.6
−0.8
−1.0
−1.2
−1.4
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
strain

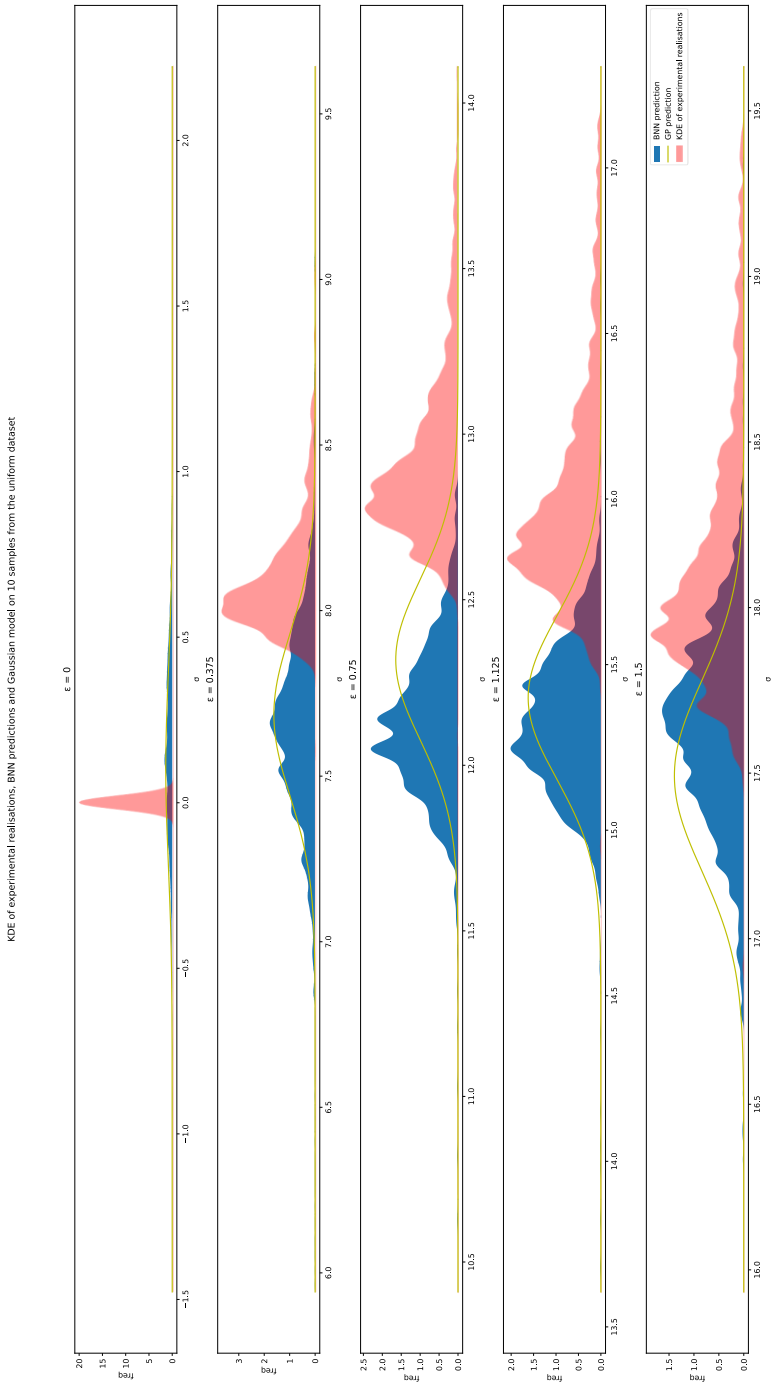
D2.2


negative Log Likelihood Gaussian dataset
negative log likelihood
−0.5
−1.0
−1.5
−2.0
−2.5
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
strain

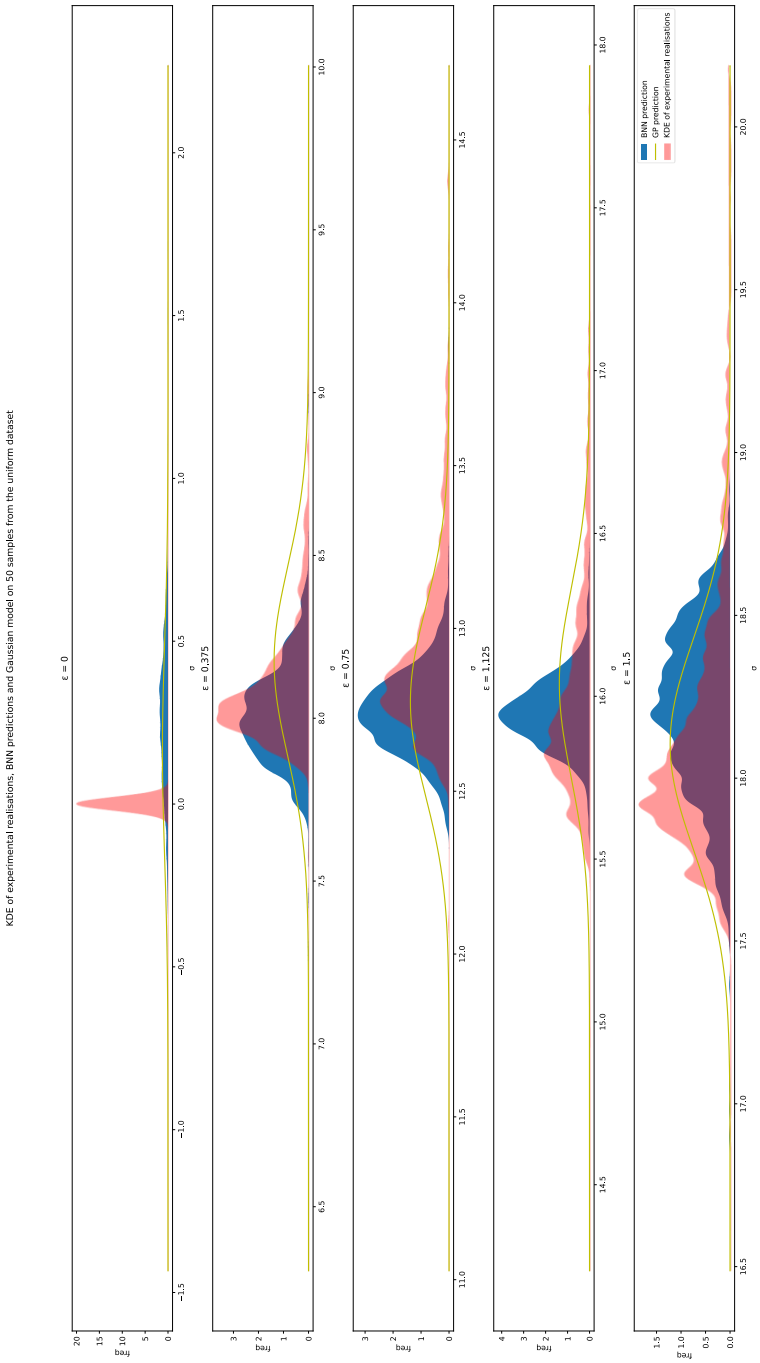
D2.3


negative Log Likelihood Gaussian dataset
negative log likelihood
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
strain

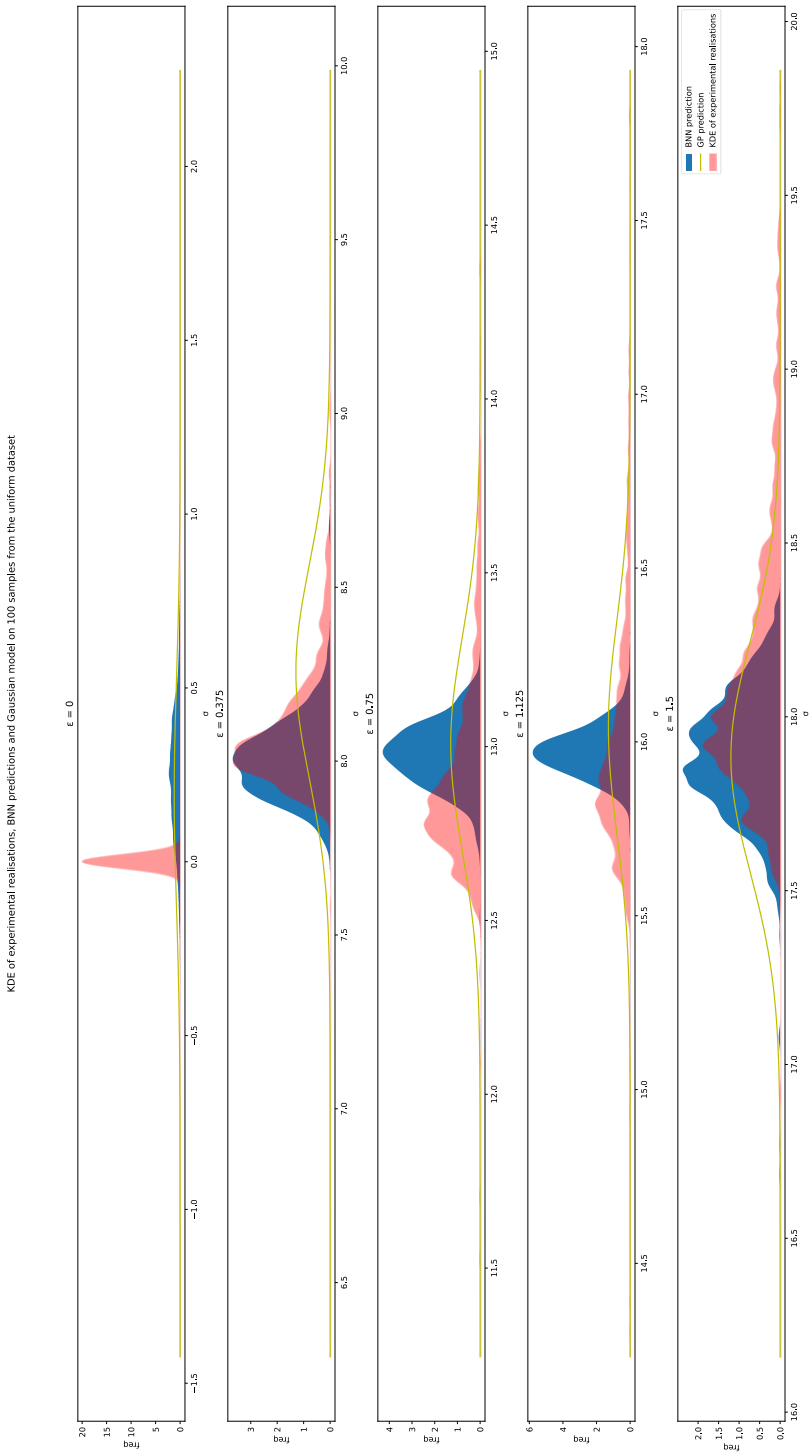
E1.1


KDE of experimental realisations, BNN predictions and Gaussian model on 10 samples from the uniform dataset
ε = 0
ε = 0.375
ε = 0.75
ε = 1.125
ε = 1.5
freq
σ
BNN prediction
GP prediction
KDE of experimental realisations

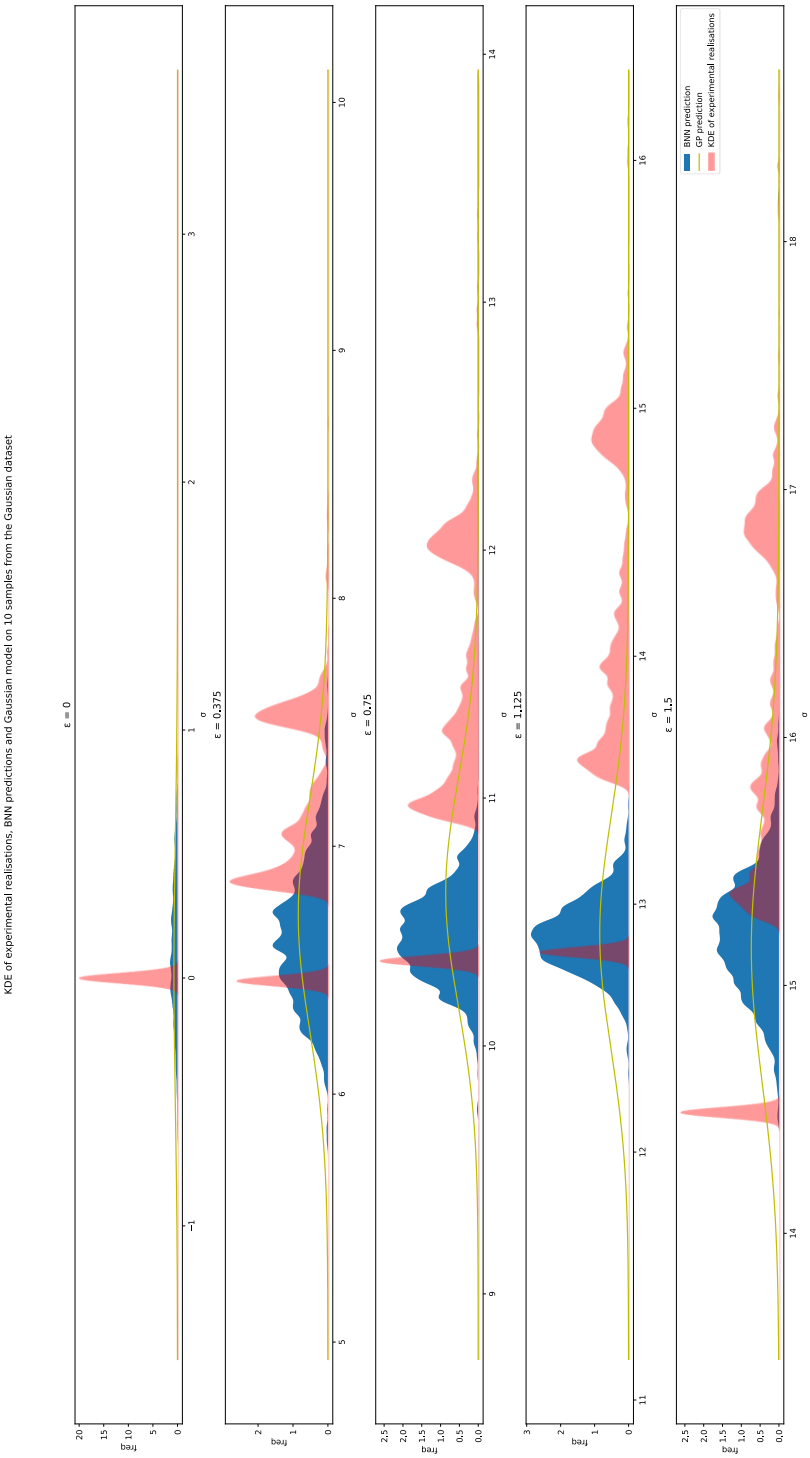
E1.2


KDE of experimental realisations, BNN predictions and Gaussian model on 50 samples from the uniform dataset
ε = 0
ε = 0,375
ε = 0,75
ε = 1,125
ε = 1,5
freq
σ
BNN prediction
GP prediction
KDE of experimental realisations

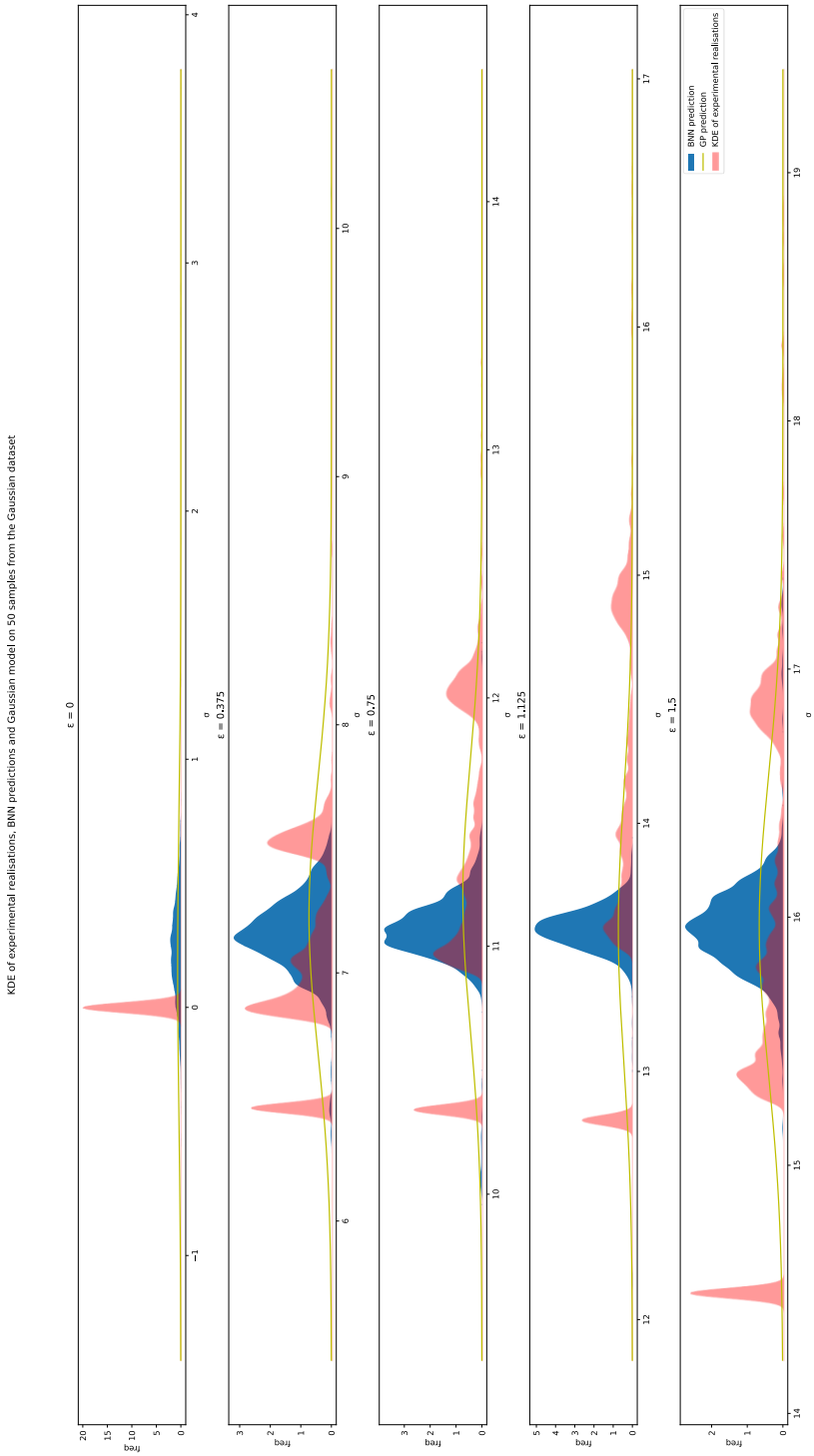
E1.3


KDE of experimental realisations, BNN predictions and Gaussian model on 100 samples from the uniform dataset
ε = 0
ε = 0.375
ε = 0.75
ε = 1.125
ε = 1.5
freq
σ
BNN prediction
GP prediction
KDE of experimental realisations

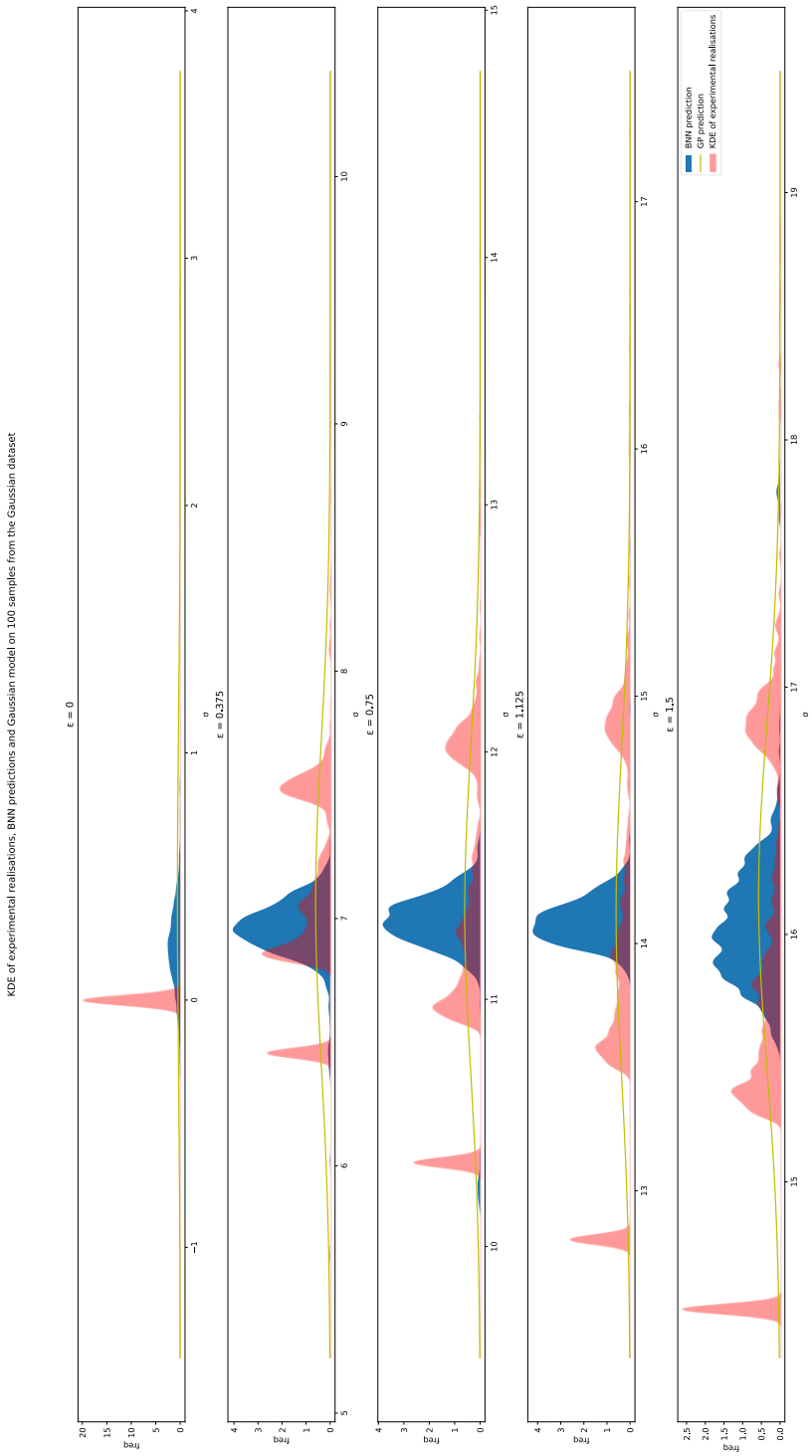
E2.1


KDE of experimental realisations, BNN predictions and Gaussian model on 10 samples from the Gaussian dataset
ε = 0
ε = 0,375
ε = 0,75
ε = 1,125
ε = 1,5
freq
σ
BNN prediction
GP prediction
KDE of experimental realisations

E2.2


KDE of experimental realisations, BNN predictions and Gaussian model on 50 samples from the Gaussian dataset
ε = 0
ε = 0,375
ε = 0,75
ε = 1,125
ε = 1,5
freq
σ
BNN prediction
GP prediction
KDE of experimental realisations

E2.3


KDE of experimental realisations, BNN predictions and Gaussian model on 100 samples from the Gaussian dataset
ε = 0
ε = 0,375
ε = 0,75
ε = 1,125
ε = 1,5
freq
σ
BNN prediction
GP prediction
KDE of experimental realisations