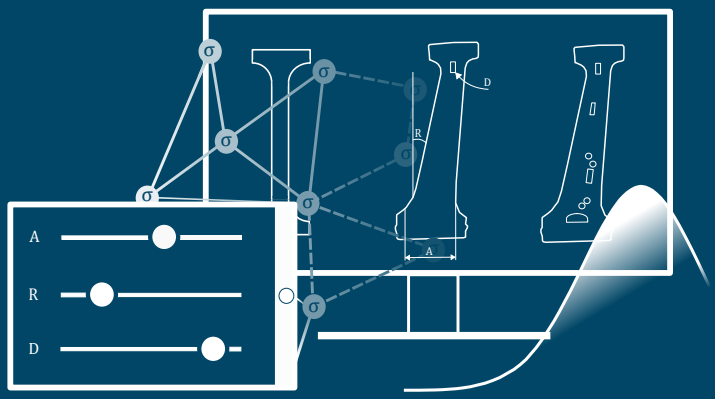


Towards an interactive crash simulation

Bram Pieter van de Weg

TOWARDS AN INTERACTIVE CRASH SIMULATION

Bram Pieter van de Weg

TOWARDS AN INTERACTIVE CRASH SIMULATION

DISSERTATION

to obtain
the degree of doctor at the University of Twente,
on the authority of the rector magnificus,
prof. dr. ir. A. Veldkamp,
on account of the decision of the Doctorate Board,
to be publicly defended
on Tuesday the 12$^{\text{th}}$ of December 2023 at 16.45 hours

by

Bram Pieter van de Weg

born on the 4$^{\text{th}}$ of December 1994
in Assen, The Netherlands

This dissertation has been approved by:

Supervisor:
prof.Dr.-Ing. B. Rosić

Co-supervisor:
L. Greve, Ph.D.

The research in this dissertation was carried out at the Volkswagen Aktiengesellschaft in Wolfsburg, Germany, in close collaboration with the Applied Mechanics and Data Analysis group within the faculty of Engineering Technology at the University of Twente in Enschede, The Netherlands. The funding of this research is fully covered by the Volkswagen Aktiengesellschaft. The results, opinions and conclusions expressed in this thesis are not necessarily those of Volkswagen Aktiengesellschaft.

Cover design: Bram Pieter van de Weg
Printed by: Gildeprint
Lay-out: Bram Pieter van de Weg
ISBN (print): 978-90-365-5878-5
ISBN (digital): 978-90-365-5879-2
URL: https://doi.org/10.3990/1.9789036558792

Graduation Committee:

Chair / secretary

prof.dr.ir. H.F.J.M. Koopman	University of Twente

Promotor

prof.Dr.-Ing. B. Rosić	University of Twente

Co-promotor

L. Greve, Ph.D.	Volkswagen Aktiengesellschaft

Committee Members

prof.dr.ir. A.H. van de Boogaard	University of Twente
prof.dr. A.R. Thornton	University of Twente
prof.Dr.-Ing. S. Elgeti	Vienna University of Technology
assoc.prof. A. Kučerová, Ph.D.	Czech Technical University in Prague

Acknowledgements

The work set out in this thesis is performed at Volkswagen Aktiengesellschaft in Wolfsburg, Germany, and is supervised by the University of Twente in the Netherlands. Despite the presence of many uncertainties at the onset of this research project, the close collaboration and incredible support provided by all people involved has motivated me to keep going, no matter how big the hurdles may be been. Hence, I would like to express my gratitude towards all people that have provided their support, on both professional and personal front.

I would like to first thank my direct supervisors prof.Dr.-Ing. Bojana Rosić and Lars Greve, Ph.D. Bojana, thank you for accepting me as a doctoral student. Despite your busy schedule you were always prepared to guide me in the right direction and provide feedback where required. Your patience and hard work is unmatched and a great encouragement for me to stay focused. The advice you gave in our regular meetings always helped me to further progress, stay critical and think of solutions I could have never imagined. Lars, the many discussions we have had in the past years have helped me to always stay focused. Your incredible expertise and ideas to solve problems have greatly supported my progress over the years. I am very grateful to have such an incredibly dedicated colleague that is willing to provide valuable feedback on a daily basis and could not have wished for a better team of supervisors.

Furthermore, I would like to express my gratitude towards the great team at Volkswagen Aktiengesellschaft. Thank you for the warm welcome in the team. Your insights and willingness to provide feedback from an application point of view have been of great value to me. The care, trust and support you have given helped me tremendously throughout the assignment.

At last (but not least), a very special thank you to my family for all their love and patience. The support and opportunities you have given me will always leave me in debt.

Summary

To comply with ever-increasing requirements in vehicle safety, light weight design and energy footprint, the automotive industry is currently undergoing a paradigm change where computationally expensive finite element simulations are substituted by more efficient surrogate modeling approaches. An initial step towards surrogate modeling of finite element simulations is the identification of the nonlinear large-deformation finite element model, the parameters of interest in the finite element model and the parameterization accordingly. To strive towards the goal of computationally efficient optimizations in crash-related problems, the current state of research is first evaluated using data generalized by parametrized finite element models and on the basis of a currently available, well known and proven long short-term memory surrogate architecture. To subsequently overcome the problems of generalization and handling of data noise a novel optimization methodology is developed that provides a flexible and self-configuring surrogate model. By means of benchmark examples and crash-related load cases this methodology is proven robust, while simultaneously allowing the quantification of (relative) uncertainty over the imposed domains. A strongly related continuation of this approach is then developed to further improve the description of the present domains (i.e. time, stochastic and spatial) using the principle of domain separation and corresponding suitable surrogate frameworks. This approach implicitly entails a low-rank approximation of the approximated problem to elevate the computational efficiency of the surrogate model. It is found that with the in this work developed novel optimization methodologies in the various surrogate frameworks a robust, flexible and self-configuring surrogate model is provided that allows for an efficient and interactive evaluation of structural optimization problems.

Samenvatting

Om aan steeds strengere eisen voor voertuigveiligheid, lichtgewicht ontwerp en energie voetafdruk te kunnen voldoen, ondergaat de automobielindustrie op het moment een verandering van paradigma, waar rekenkundig dure eindigende elementen simulaties worden vervangen door een efficiëntere substitionele modelleeraanpak. Een eerste stap in de richting van het substitioneel modelleren van eindigende elementen simulaties is het identificeren van het eindigende elementen model met niet-lineaire, grote vervormingen, de parameters die van belang zijn in het eindigende elementen model en de daarbij behorende parameterisatie. Om naar het doel van rekenkundig efficiënte optimalisaties in crash-gerelateerde problemen te streven, wordt de huidige staat van onderzoek eerst geevalueerd met behulp van data dat is generaliseerd door geparameteriseerde eindigende elementen modellen en op basis van de op het moment beschikbare bekende en bewezen "long short-term memory" substitionele modelarchitectuur. Om vervolgens de problemen van generalisatie en de omgang met data ruis te overwinnen is een nieuwe optimalisatiemethode ontwikkeld die een flexibele en zelfconfigurende substitionele modelarchitectuur biedt. Met het gebruik van maatstafvoorbeelden en crash-gerelateerde opstellingen is de robuustheid van deze methode bewezen, terwijl tegelijkertijd de kwantificatie van (relatieve) onzekerheid over de opgelegde domeinen mogelijk wordt gemaakt. Een sterk gerelateerde voortzetting van deze aanpak is daarna ontwikkeld om de beschrijving van de aanwezige domeinen (d.w.z. tijd, stochastisch en ruimtelijk) verder te verbeteren met gebruik van het principe van de scheiding van domeinen en de daarbij geschikte substitionele modellen. Deze aanpak brengt impliciet ook een lage-orde benadering van het benaderde probleem met zich mee om de rekenkundige efficiëntie van het substitutionele model te verheffen. Het is gebleken dat met de in dit werk ontwikkelde nieuwe optimalisatiemethoden in de verschillende substitutionele modellen een robuust, flexibel en zelfconfigurerend substitutioneel model is ontwikkeld dat een efficiënte, interactieve evaluatie van structuuroptimalisatieproblemen mogelijk maakt.

Contents

Nomenclature

The most commonly used symbols and abbreviations are listed here.

Abbreviations

ARD	automatic relevance determination
BCC	body-centered cubic
CAE	computer aided engineering
DMD	dynamic mode decomposition
DOE	design of experiment
EI	expected improvement
ELBO	evidence (or expected) lower bound
FCC	face-centered cubic
FEM	finite element method
GPR	Gaussian process regression
GPU	graphics processing unit
GRU	gated recurrent unit
HSR	Hill Stören-Rice (necking model)
KDE	kernel density estimation
KL	Kullback-Leibler
KLE	Karhunen-Loève expansion
LHS	Latin Hypercube sampling
LSE	least squared error
LSTM	long short-term memory
MAP	maximum a posteriori (estimation)
MC	Monte Carlo method
MK	Marciniak-Kuczynski (necking model)
MLE	maximum likelihood estimation
MLP	multi-layer perceptron
mMC	modified Mohr-Coulomb
MOR	model order reduction
MP	most probable
MSE	mean squared error
NN	neural network
ODE	ordinary differential equation
PCA	principal component analysis
PCE	polynomial chaos expansion
PDE	partial differential equation
PDF	probability density function
PGD	proper generalized decomposition
PINN	physics-informed neural network

POD	proper orthogonal decomposition
QoI	quantity of interest
RBF	radial basis function
RED-LSTM	ARD-PCE-POD-LSTM model combination
RNN	recurrent neural network
RV	random variable
SBL	sparse Bayesian learning
SLP	single-layer perceptron
S-S	stress-strain (curve)
SVD	singular value decomposition
SVM	support vector machine
SVR	support vector regression
UQ	uncertainty quantification
VPS	virtual performance solution

Symbols

Latin Symbols

A_l	affine map for neural network layer l
a_H	exponent of the Hershey yield criterion
a_V	free model parameter for Voce hardening law
b_V	free model parameter for Voce hardening law
C	damping
$\boldsymbol{c}_\ell$	LSTM cell state at time step ℓ
$\boldsymbol{c}$	system state
$\boldsymbol{c}_\tau$	trajectory of the solution in the past
$\dot{\boldsymbol{c}}(\cdot)$	functional form of a dynamic model
$\tilde{\boldsymbol{c}}_\ell$	candidate gate state at time step ℓ
c_θ	modified Mohr-Coulomb fracture model parameter
$D_\mathrm{KL}(\cdot)$	KL-divergence measure
d_f	fracture indicator
$\mathcal{E}$	discrepancy measure
$\mathcal{E}_\mathrm{e}$	variational free energy
F_ℓ	functional recurrent neural network representation
$\mathcal{F}$	force
$\boldsymbol{f}_\ell$	forget gate state at time step ℓ
$\mathfrak{F}$	σ-algebra
$F_\mathbf{w}$	functional neural network representation
$\boldsymbol{g}$	nonlinear, saturating and invertible function of a state
$G(\cdot)$	general notation for a finite element model
$G_\mathrm{S}(\cdot)$	general surrogate model notation
$G_\ell(\cdot)$	incremental (in time discretized) surrogate model notation
$\mathbf{H}$	Hessian matrix
$\boldsymbol{h}_\ell$	LSTM hidden state at time step ℓ

$\mathcal{H}$	hidden state space
$\mathbf{I}$	identity matrix
$\mathcal{J}$	mean squared loss
k_S	free model parameter for Swift hardening law
K	stiffness
k_V	free model parameter for Voce hardening law
l	length; element length
$\mathcal{I}(\cdot,\cdot)$	Laplacian probability density function
$\mathcal{L}$	(log) likelihood loss objective function
$\mathbf{m}_1, \mathbf{m}_2$	first and second order adaptive moment
M	mass
$\mathcal{N}(\cdot,\cdot)$	Gaussian probability density function
n_b	batch size
n_c	number of stochastic elements
n_d	number of designs (i.e. data samples)
n_e	number of elements of the finite element model
n_f	number of input parameters (i.e. features)
n_i	LSTM gate input dimension
n_L	number of neural network layers
n_m	state dimension, i.e. number of units
n_n	number of nodes of the finite element model
n_o	spatial (i.e. output) dimension
n_id	node identification index
n_s	number of Monte Carlo samples
n_r	number of stochastic elements per input parameter
n_S	free model parameter for Swift hardening law
n_t	number of time steps
n_pc	number of principal components
n_g	polynomial degree in PCE
n_p	number of polynomials in PCE
n_v	number of variables
n_V	free model parameter for Voce hardening law
$\boldsymbol{o}_\ell$	output gate state at time step ℓ
$\mathcal{O}$	transition parameter space
$\mathbb{P}$	probability measure
$p(\cdot)$	probability
$\mathcal{Q}$	input parameter space
Q_V	free model parameter for Voce hardening law
$\boldsymbol{q}$	input parameter set
$\boldsymbol{q}_i$	input parameter set for observation i
$\hat{\boldsymbol{q}}$	predictive input parameter set
R^2	coefficient of determination
$\mathbb{R}$	set of real numbers
$\mathbf{R}$	precision matrix
$\boldsymbol{s}^\lambda$	singular values
$\boldsymbol{s}$	local state

$\mathcal{T}$	time domain
u	displacement
$\dot{u}$	rate of displacement; velocity
$\ddot{u}$	acceleration
$\mathbf{u}$	unitary matrix
$\mathcal{U}(\cdot,\cdot)$	uniform probability density function
ν	degrees of freedom on a probability density function
$\mathbf{v}$	collection of eigenvectors
$\boldsymbol{w}$	weight vector
$\mathbf{w}$	collection of weights
w_b	bias
$\boldsymbol{w}_\mathrm{s}$	linear scaling weight vector
$\mathbf{w}_{\mathrm{c},\ell}$	polynomial chaos expansion coefficient set at time step ℓ
$\mathbf{w}^\mathrm{h}$	recurrent part of recurrent neural network weight collection
$\mathbf{w}^\mathrm{q}$	subset of the weight collection corresponding to the input
$\mathcal{X}$	spatial domain
$\boldsymbol{y}_\ell$	output parameter set at time step ℓ
$\tilde{\boldsymbol{y}}$	forecast of the output not accounting for modeling errors
$\mathcal{Y}$	output parameter space
$\mathcal{Y}_{n_\mathrm{n}}$	discretized output parameter space (in n_n basis functions)
$\hat{\boldsymbol{y}}$	prediction of the real observation
$\underset{\sim}{\boldsymbol{y}}_\ell$	output data set in the reduced subspace at time step ℓ
$\tilde{\underset{\sim}{\boldsymbol{y}}}_\ell$	predictive output in the reduced subspace at time step ℓ
$\boldsymbol{z}_\ell$	input gate state at time step ℓ

Greek Symbols

$\boldsymbol{\alpha}$	prior hyperparameter parameter
$\boldsymbol{\beta}$	likelihood hyperparameter
$\boldsymbol{\Sigma}$	covariance
δ_{mn}	Kronecker delta function
$\overline{\varepsilon}_\mathrm{f}$	equivalent strain to fracture
$\tilde{\boldsymbol{\varepsilon}}$	prediction of the measurement/modeling error
$\overline{\varepsilon}^\mathrm{p}$	accumulated equivalent plastic strain
$\varepsilon_{0,\mathrm{S}}$	initial plastic strain in Swift hardening law
$\boldsymbol{\xi}$	set of random variables
ι	time delay
$\mathcal{J}$	multi-index set to (multivariate) polynomials in PCE
κ	sample selection parameter
λ	learning rate
λ_n	n-th eigenvalue for the Lyapunov stability critaria
λ_m	mixing parameter in the combined Swift-Voce hardening law
η	stress triaxiality
Ω	sample space
ω	the event in a σ-algebra
ϕ_y	Hershey yield criterion

Ψ	collection of orthonormal polynomial basis functions
$\boldsymbol{\Phi}$	surrogate model input state basis function
ϱ	adaptive moment blow-up prevention factor
σ_{S}	Swift hardening law
σ_{V}	Voce hardening law
σ_{SV}	combined Swift-Voce hardening law
$\sigma_1, \sigma_2, \sigma_3$	principal stresses
$\overline{\sigma}$	equivalent stress
σ_{m}	hydrostatic stress
$\sigma(\cdot)$	activation function
$\hat{\boldsymbol{\sigma}}^2$	predictive variance
σ^2	stochastic variance
σ_{h}	transition function of a recurrent neural network layer
σ_{y}	transition function of a neural network output layer
σ_{s}	deviatoric stress
$\overline{\sigma}_{\mathrm{Y}}$	yield stress
$\boldsymbol{\theta}$	collection of internal layer parameters
$\Theta(\cdot)$	computational order indication function
$\overline{\theta}$	normalized Lode angle
$\boldsymbol{\theta}$	collection of dependent parameters in KL-divergence
τ	time moment
τ_ℓ	time moment of discretized time sequence at time step ℓ
μ	stochastic mean
φ	input parameter value
υ	set of weights to the separate domain representation
ϑ_1, ϑ_2	exponential decay rate factors
χ^2	Chi squared distribution function
$\boldsymbol{\zeta}$	collection of output layer parameters
γ_n^{p}	orthonormality measure for a set of polynomial functions
$\hat{\boldsymbol{\mu}}$	predictive mean
$\Upsilon_j(t)$	basis function representation of the time domain
$\Psi_i(\omega)$	basis function representation of the stochastic domain
$\Gamma_k(x)$	basis function representation of the spatial domain

List of Figures

List of Tables

1 Introduction

In this chapter an introduction to the contents of this thesis is provided, with a main focus on the pursued goal to develop a novel robust and computationally efficient methodology for the optimization of vehicle crashworthiness.

Automotive transportation is globally intensifying. To ensure passenger safety and simultaneously enforce minimal material usage, the automotive industry is devoting increased attention to both vehicle crashworthiness and light weight design. Assessment of vehicle crashworthiness mainly relies on destructive physical tests, also known as crash tests, examples of which are the frontal impact test, moderate overlap test, side impact test and rear impact test, among others [4–7]. These are done with the help of the test dummies playing the role of passengers. With the help of sensors and monitoring techniques one may then identify places where vehicles need to be made safer and/or reinforced. However, such experiments require huge investments, especially if there is a need for their repetition. Today, these investments can be substantially reduced, and the structural design can be significantly improved using virtual testing simulations. The main idea is to virtually mimic the vehicle behavior during crash, and improve its structural design with the help of a computer aided engineering (CAE)-based optimization [8–10]. CAE starts with the physics-based or mathematical model of a vehicle, and optimizes its design by tuning all the relevant features until the corresponding mechanical safety criterion is satisfied.

Despite the fact that CAE-based optimization shows a great promise, its direct utilization is not so straightforward. To use CAE in practice one has to design the digital phantom, i.e. a virtual model that mechanically behaves in the same manner as the corresponding vehicle. In this thesis the digital phantom is understood as a physics-based model discretized by a finite element method, i.e. the so-called finite element model (FEM) [3, 9, 11–15]. However, a realistic FEM model of a vehicle in the specific crash relevant situations is not existing. One cannot model such a complex system in a great detail. Furthermore, the boundary and excitation conditions in real traffic situations are often not fully known or are non-deterministic. Next to this, the model requires the definition of the constitutive material law that may vary with the material choice during optimization. Similar holds for any design change in geometrical features at the system or the component level. Thus, one cannot stay with the simple deterministic representation of a vehicle. Instead, one has to be able to alter vehicle properties or introduce any possible variations without directly changing the digital phantom. To achieve this, the finite element model has to be extended to its stochastic-FEM (SFEM) counterpart [16–18]. Next to the proper choice of explicit/implicit discretization of the model in the time domain [9], this extension may improve models accuracy but rise its computational cost.

Due to increasing customer demands and stricter governmental and industrial standards, the simulation society is under continuous pressure to make a FEM model very detailed. A higher level of detail leads to a more realistic design optimization. However, highly detailed computations also cause an elevated pressure on the available computational resources and optimization time, and hence cause growth of the computing carbon footprint. To overcome these issues, the simulation effort in industrial environment is currently subject to a paradigm change, by the help of which the demanding finite element simulations are replaced by a real time (i.e. interactive) substitute in the form of an efficient surrogate[1] model.

[1]Someone or something that replaces or is used instead of someone or something else; a substitute for another.

As FEM-based models can describe the physical switch from linear to nonlinear behavior of a system or component, their efficient counterpart therefore has to be adapted accordingly. In other words, the surrogate model has to generalize and represent both linear and nonlinear situations accurately. It is therefore of key importance to identify any present nonlinearity in the FEM model, and select the appropriate surrogate model accordingly. In addition to the (non)linearity of the finite element problem considered, one may desire to investigate the effect of a varying parameter on the often unknown and complex system/component response. Accordingly, the appropriate surrogate models must be flexible such that both deterministic and stochastic problems can be described. To provide such flexibility one may distinguish two types of surrogate model approaches: a physics based or data driven approach. The former imposes the inclusion of domain knowledge in the proxy model in the form of physics laws. The latter denotes an approach in which the surrogate framework is constructed only using the data provided.

Despite an expectation with respect to the computational efficiency, surrogate modeling of both deterministic and stochastic problems may not necessarily reduce the computational complexity compared to the original phantom model. This may become troublesome in more complicated, high-dimensional problems. Further advances to reduce the computational complexity (and thus the computational effort) in surrogate modeling can be achieved by employing model order reduction schemes, an example of which is the low rank decomposition approach [19–23]. Using decomposition schemes one may recognize dependency within the data set generated by a virtual phantom model and subsequently map the original data set to a reduced subspace. By truncating the decomposition of the input and/or response of the phantom model one obtains a low-rank approximation of the described problem, hereby ensuring model order reduction (MOR).

Although promising, in contrast to FEM, surrogate-based modeling (and accompanying model order reduction) is challenging due to the absence of a unique approach of making a model. As a result, over the last decade a multitude of surrogate modeling approaches have emerged. Most of these models are obtained in a data-driven way in which the surrogate model is trained given the data set collected by repeated FEM simulations for different parameter values within a predefined parameter space.

1.1 State of the art

An example of a surrogate model is an artificial neural network (NN) [24], the main topic of this thesis. Artificial NNs may be depicted as an interconnected collection of neurons, where each neuron exhibits a predetermined (nonlinear) transform of its own input, also known as the activation functions. By means of these activation functions and NN architecture, one can approximate the parameter-response map of the FEM model or its stochastic counterpart in both linear and nonlinear cases. Thanks to the adaptability of the network architecture and its activation functions NNs are considered to be universal approximators [25]. The unknown coefficients (i.e. weights) present in NNs are estimated using the theory of back-propagation [26],

also known as the gradient based approach in which the weights are updated from the last layer to the first one in a sequential manner based on a specific loss function and its gradient. In the majority of cases, this function is taken to be the mean squared error, but other choices also exist [27].

The general applicability of NNs in linear and nonlinear application problems is fundamental for its employment and gain of popularity in the field of nonlinear (solid) mechanics. The first applications of NNs in structural fracture mechanics mainly focus on low-dimensional problems, namely crack propagation and forming limit diagrams [28–33]. Due to absence of constitutive laws for new types of materials, NNs are mainly used to approximate the stress-strain relationship of material behavior given pure experimental data sets. Mainly, for this purpose the so-called plain feed-forward NN architectures [28–33] are used. In feed-forward NN architectures [34] the information only flows in a single direction, from the (input) parameter to the (output) response. In [28, 31] a single layer perceptron (SLP) is used. This NN only consists of one activation function that directly maps the input to the response. If activation function is of the linear type the input-output map is approximated by a linear regression. As an SLP with linear activation functions cannot properly describe nonlinearity, in [30, 33, 34] the authors used a multi-layer perceptron (MLP) that consists of several layers each described by a nonlinear activation function. This type of approximation decomposes the parameter-response map to several local maps, each obtained by composition of the previous layer with the following one. Among different network architectures it is found that the nonlinearity between the input parameter set and the output response strongly affects the required network dimension in terms of number of neurons and (hidden) layers.

The application of NNs in multi-dimensional fracture applications are found in [35, 36]. In [35] a feed-forward NN is found to outperform the classical necking model (enhanced Hill/Stören-Rice (eHSR) [35, 37, 38]) in terms of accuracy on the applied load cases. The enhanced Hill/Stören-Rice model assumes damage accumulation for defect-free homogeneous metals and proportional loading and the NN imposes a unique parameter-response map over the complete time domain. In [35] the applied NN is additionally found to outperform the more advanced Marciniak and Kuczynski (MK) necking model [39] in terms of computational efficiency. In contrast to the enhanced Hill/Stören-Rice necking model, the model proposed by Marciniak and Kuczynski entails nonlinear damage accumulation. The history dependency of this nonlinear damage accumulation is assumed to be intrinsically defined at any given time moment by a set of predetermined parameters. These parameters are in-plane stress tensors, accumulated plastic strains, equivalent plastic strain, and the stress triaxiality. The applied NN employs these parameters for the imposed parameter-response map.

Besides plain feed forward NNs, in recent years other surrogate models have also found a role in making a proxy model for the finite element models [40–42]. In [42] a performance analysis of various surrogate models used to approximate displacements, stresses and strains has been set out under varying geometrical and material parameters. Examples of applied models are support vector regression (SVR) [43, 44] and

physics informed neural networks (PINN) [45]. Support vector regression is a derived type of the support vector machine-model(SVM) [46] proposed in [43, 47]. SVR focuses on finding an input parameter dependent function of the nonlinear type. The map is based on the kernel transformation of the data set from a nonlinear to linear space in which then the data are approximated by a linear regression. The last one is made such that most of the data points fall inside of the predetermined margin. The main advantage of SVR models is the independence of the computational effort regarding the dimension of the input parameter set. Besides the choice of kernel, the a priori determination of a suitable margin may therefore be increasingly difficult given the nonlinearity of the response. In [42] is hence shown that SVR cannot well approximate the given FEM response. On the other side, PINNs are defined by incorporating physics-based constraints derived from the fundamental partial differential equations (PDEs) in the networks architecture (e.g. in the error function). Although not necessarily restricted to finite element modeling, the application of physics-constrained surrogate frameworks is gaining popularity in (nonlinear) finite element modeling in recent years [42, 48, 49]. However, the use of PINNs requires changing the NN architecture and the corresponding loss every time new physics phenomena appears. Therefore, this approach is not considered in this thesis.

In a classical surrogate modeling approach, one focuses on the approximation of the FEM response in a particular spatial point (node or element) and specific time moment. This then promotes the use of parallel computing, but requires allocation of a huge memory space for storing all the approximations in different spatial points and time moments. As FEM responses in different nodes/elements are highly correlated, one can take this dependency into account to reduce memory requirements, and at the same time increase computational efficiency. An example exploring linear dependency is the Karhunen-Loève expansion (KLE) [19], which in an empirical sense is also known as the principal component analysis (PCA), or as proper orthogonal decomposition (POD) [20–23]. In PCA the response is decomposed by means of orthogonal eigenvectors (i.e. principal components) and eigenvalues that are obtained given the response covariance matrix. A well known extension of the linear PCA is its non-linear kernel-based equivalent (kernel-PCA) [50–52]. A further generalization that does not necessarily impose orthogonal functions is the proper generalized decomposition (PGD) [53, 54]. In contrast to POD, the PGD approach is an iterative self-enriching nonlinear method, developed to solve boundary value problems and hence is capable to approximate nonlinear problems. A related approach that uses decomposition by evaluating frequency response functions [55, 56] is dynamic mode decomposition (DMD) [57].

One may classify model-order reduction techniques in an intrusive and non-intrusive sense [15, 58–62]. A non-intrusive application defines the use of the surrogate model in such sense that it does not require modification of the source code. Contrary, the intrusive reduction requires changing of the source code. In [58–61] decomposition schemes have been applied in an intrusive sense to reduce the computational effort evolved with the size of the stiffness matrix and internal force vector in a finite element simulation. An example of a crash-relevant non-intrusive application of aforementioned decomposition approaches is found in [62]. In this contribution a

global high-fidelity model of a parametrized crash-relevant component (B-pillar) is reduced using the PGD approach. Another example of non-intrusive application of decomposition schemes in crash is devoted to uncertainty quantification (UQ) and robustness analysis [15]. In [15] a local nonlinear kernel PCA is applied to identify the spatial dependency and to simplify the model description of the plastic strain in a finite element model of a simplified B-pillar. However, the aforementioned surrogate models and decomposition schemes do not impose the propagation of information between time steps and are thus unable to describe a time-dependent relationship as found in the material state during a finite element simulation.

To provide a surrogate model for crashworthiness evaluation of structural components in FEM, one must account for the time dependency in which the history evolution of internal states has to be taken into account. Typical examples are recurrent neural networks (RNN) [26, 63]. An RNN propagates state information in a recurrent sense over a (time) sequence. Every step in this sequence therefore partly depends on the propagated state from the previous step. Examples of RNNs, besides its plain architecture are the gated recurrent unit (GRU) [64] and the long short-term memory (LSTM) RNN [65–67]. These networks have originally been developed for language processing but are currently applied for a wide range of surrogate applications [68–72]. Examples of RNNs in nonlinear mechanics applications are found in [70, 71]. In [70] a non-intrusive application of the RNN model is used to perform an interactive parameter optimization for a structural component in crash. In [71] the pre-trained recurrent proxy model is implemented in the finite element solver to accelerate the multiscale finite element simulation. Despite this clear distinction in surrogate modeling approaches, both contributions impose the description of a nonlinear material model in the chosen recurrent surrogate frameworks.

The previously reviewed works had a focus on the surrogate modeling of the deterministic FEM problems. On the other hand, stochastic FEM problems are more difficult to substitute by an efficient surrogate approach (e.g. [42, 73–78]). Besides the aforementioned application of SVR, in [42] Gaussian process regression (GPR) is applied to approximate the FEM-based displacement, stresses and strains under plastic deformation. Gaussian process regression is a nonparameteric technique that does not evaluate parameters of fixed basis functions, but strives to infer the correlation among the data points. A priori this correlation is described using a predetermined kernel. Gaussian process regression can therefore be employed to describe the (nonlinear) parameter-response mapping in a continuous sense. In [42] Gaussian process regression is found to fails to accurately approximate plastic deformation, indicating limited applicability in highly nonlinear problems that are not smooth. Furthermore, in [74–77] a polynomial chaos expansion (PCE) [79–83] is employed to model nonlinear behavior obtained from a structural finite element response. In polynomial chaos expansion the uncertain input parameters (i.e. random variables or random fields) of the model are described using a predetermined distribution. Accordingly, the parameter-response mapping is approximated by means of polynomials that are orthogonal with respect to the joint distribution over the random variables. The use of these orthogonal polynomials hence allows to efficiently take into account uncertainty in the aforementioned parameter-response mapping. Another popular technique is to use a

radial basis function (RBF) surrogate approach [84]. For example, in [78] this method is used to analyze damage in a one-dimensional reinforced concrete beam. A radial basis function defines its output by the distance (i.e. the radius) of an input value to the origin. A single radial basis function is employed for each input parameter of the parameter-response mapping. In the radial basis function approach a response is then approximated by linearly weighted combinations of the predetermined radial basis functions. Therefore, the quality of the approximation of the parameter-response mapping is primarily determined by the choice of radial basis function. A major advantage of these previously described stochastic surrogate approaches (thus GPR, PCE and RBF) is their ability to include stochastic dependencies in the parameter-response mapping. However, the adaptability of these surrogate architectures is limited compared to the aforementioned NN frameworks. On the other hand, NNs are in their traditional form unable to properly capture modeling uncertainty, but have a highly adaptable architecture.

In this thesis the NN framework is further considered as an appropriate surrogate model. Although one may argue that other mentioned approaches can be used as well, in this thesis the focus is on the NN as an universal approximator in contrast to for example the polynomial model. Furthermore, the focus is on NN models that use probabilistic theory for their training instead of classical deterministic gradient based approaches. If one oughts to approximate the deterministic FEM model by a NN model, then the probabilistic perspective on the weight estimation of such a model allows new ways of training that are more suitable for industrial practice. Please note that in such a case one estimates the deterministic parameters of the NN through a probabilistic approach. This is known as probabilistic numerics [85]. However, such an approach is also general as it can also cover models that represent noisy data. Namely, the input and output to the NN can be both subjected to noise. If noisy data are used in combination with deterministic gradient approach then the weight estimation may become severely ill-posed, and the estimate would highly depend on the amount of noise in the data. However, in a probabilistic approach the presence of noise can be easily incorporated without changing the training procedure. Furthermore, if one aims at approximating a stochastic FEM model by a NN approximator sampling-wise, then one can also use the previously described approach. Please note that the structure-related uncertainty then lies in the input and output of the network, whereas the uncertainty introduced in the weights is not structure-related but is epistemic uncertainty and represents our knowledge about NN weights. Thus, this uncertainty is reduced by gathering more data. In case of infinite data set one would then obtain a deterministic estimate of the NN weights.

The most straightforward way for the probabilistic estimation of NN models is the Bayesian theory [86, 87]. By using this approach one describes the NN weights apriori as random variables to which one assigns a probability density function. This uncertainty is epistemic as said previously and represents our knowledge on the weight value. Once the data are fed one uses Bayes' rule to update the prior knowledge into the so-called posterior distribution. An example of this are works in [2, 88–97] in which Bayesian theory is used on (recurrent) NNs, primarily to improve regularization. In many of these contributions the weights are described using Kullback-Leibler

(KL) divergence [2, 98]. KL-divergence describes an approximation over the true distribution function that is further estimated by costly sampling, making the approach computationally intractable for deep and wide NNs. In Bayesian or approximate Bayesian approaches next to the way of estimating posterior, one has to also pay attention to the prior definition. Not all priors give the same estimate of the posterior. As in classical NNs one expects that many of the neural connections are not needed in the final model description, one may model weights a priori by using the so-called sparsity induced priors. This type of prior then promotes automatic dropout procedure. However, introducing such a prior brings more complication on the estimation side. Namely, only sample based approaches can be then used. However, sample-based approaches then lead to computationally infeasible solutions that are not of practical use for industry. The training of only one NN can take several days.

To enforce a scheme that automatically and efficiently determines which weights are important in a NN architecture, and hence promote its self-configuration, in this thesis a new way of training is suggested. The training takes the probabilistic form by employing the theory of automatic relevance determination (ARD) [99]. In this direction there are already few attempts [90, 93, 100, 101]. Despite the implementation of ARD for time series prediction in [100, 101] the applied surrogate models are purely feed-forward. The proposed approach in [93] aims to describe the relationship between data samples over time, but is limited to linearly weighted kernels. In [90] the RNN output connections serve as basis functions that are linearly weighted, the relevance of these weights is then assessed using a fast sparse Bayesian learning (fast-SBL) scheme [102]. Yet, the proposed approach in this thesis employs ARD to promote the self-configuration of a NN in a nonlinear, time dependent setting that is not yet studied in the literature.

1.2 Purpose of the study

The main purpose of this work is to serve the aforementioned paradigm change from a finite element-driven structural simulation towards a robust surrogate model-driven approximation.

The first step in this process is to parametrize a finite element model describing nonlinear deformations of structural components as found in crash applications. The aim of the parametrization is to include all possible variations (material, geometrical, etc.) in the overall description of a mechanical design. As design varies, in this thesis a probabilistic parametrization is taken into account in which both design variables as well as other influential factors are described as uncertain, i.e. modeled as random variables. Given such a stochastic finite element model numerical data sets are created and comprehensive surrogate models are trained based on the existing LSTM NN architectures that are used in other areas of application.

Furthermore, the goal is to improve the training capabilities of the considered surrogate frameworks with a focus on nonlinear parameter-response mappings found in crashworthiness simulations. For this purpose, in this thesis a novel probabilistic approach is introduced to estimate the deterministic weights of an LSTM network.

The approach extends the linear automatic relevance vector machine to a nonlinear estimation problem that corresponds to a weight estimation in a nonlinear neuron. Furthermore, this method is combined with the backpropagation algorithm that allows weight estimation in a NN architecture. Due to its Bayesian nature the newly proposed method is thus allowing self-configuration of the network and its auto-adaption to the existing data. LSTM architecture is further extended by the PCE that models stochastic input data. In combination with the POD approximation of the output layer, the newly designed architecture is also made self-configuring. This new model then represents an efficient reduced order surrogate model that is capable of describing strong nonlinear mechanical behavior.

The introduction of stochastics not only enables the self-configuration of the surrogate models to the given problem, but also allows the propagation and subsequent evaluation of present uncertainties (i.e. data noise and modeling uncertainty). The paradigm change is thus expected to be accompanied by relieving the pressure of prior knowledge of the engineer in future finite element applications and optimization procedures using surrogate frameworks that self-configure and therefore provide more robust proxy models throughout.

1.3 Objectives

Following the previously described purpose of the study, several objectives are identified as listed further:

1. investigate the applicability of surrogate modeling approaches on non-linear deformation found in crash-derived load cases defined in a finite element model;
2. build a novel self-configuring surrogate framework and accompanying novel training approach;
3. incorporate model order reduction into the NN architecture;
4. and to develop an understanding of remaining difficulties in the generalized translation from FEM to surrogate models.

1.4 Scope of the research

This work pursues the goal to develop a robust surrogate approach that allows the engineer to efficiently analyze a predetermined QoI and subsequently optimize the component of interest by means of engineering decisions. To develop an understanding on the fundamentals of surrogate frameworks, extensions are gradually introduced to (in the basis) relatively straightforward surrogate models.

Special interest is devoted to the formulation and application of surrogate approaches in finite element simulations in large deformation (crash) load cases. These simulations are based on an explicit scheme, where the underlying material mechanics are adapted from previous works [3, 13, 14], further elaborated in Section 2. The accompanying data is numerical and purely based on and limited to finite element models,

under the assumption of constant mesh topologies. The accuracy of the data is thus considered to lie within machine precision.

The considered quantities of interest are e.g. equivalent plastic strain, maximum stress, etc., i.e time- and spatially-varying quantities. The focus of the study is the design phase of crash relevant components (e.g. B-pillar, bumper, etc.), and the goal is therefore to construct an efficient surrogate model for a geometrically parameterized problem. However, extension of this approach to other parametrization types is straightforward. The methodology does not depend on this, and can also easily be generalized to other application areas that do not include engineering problems.

The surrogate model of key interest in this work is the LSTM NN. The classical LSTM NNs are assumed to approximate data in a relatively low-noise environment, obtained from the finite element model. However, the novel suggested approach of their training can also be used for applications with a higher noise, as the general methodology accounts for noise. Furthermore, this work aims to provide an opportunity to achieve modular (flexible, self-configuring) surrogate architectures. The proposed training approach may therefore also be further employed for new architectures.

1.5 Thesis outline

This thesis is set out to comprise a chapter-wise insight in the development of a robust surrogate modeling approach. In Chapter 2 a global overview on component modeling is given in its traditional form, followed by the general formulation of an accompanying surrogate approximation. This chapter also includes the description of various procedures for the creation of data sets used to train the surrogate model.

After providing a general description on surrogate approaches and the generation of training data sets, a deterministic way of obtaining the surrogate framework is considered in Chapter 3. Here, attention is devoted to the general applicability and methodological explanations of the selected surrogate frameworks. The chapter starts with a general formulation of feed forward NNs, which is further adapted towards a recurrent LSTM architecture. In the final section two structural load cases are considered to evaluate the LSTM framework under a range of architectural variations in practice.

A novel stochastic approach to train the LSTM networks is introduced in Chapter 4. The goal of this approach is to turn the selected model of Chapter 3 in a more robust, flexible and self-configuring form. In addition to benchmark examples, the applicability of a newly designed network is subsequently evaluated by means of a structural academic example, first applied in Chapter 3.

To increase computational efficiency and simultaneously provide a more compact proxy model an alternative surrogate approach is proposed in Chapter 5. This approach further employs model order reduction in combination with the functional approximation of random variables that describe varying design parameters. For the purpose of evaluation one of the application examples from Chapter 3 is used and compared.

The final conclusions of the thesis and a possible outlook are further discussed in Chapter 6.

1

2 Modeling and simulation of structural components in crash

In this chapter the physics-based model of vehicle components undergoing crash-based type of deformations is elaborated. In particular, the dynamics type of model that incorporates the non-linear mechanical constitutive law is presented. This model takes into account the non-linear stress-strain relation that can account for the material damage in high-strength steels. As material choice and the vehicle component shape can have a huge effect on the energy absorption during the crash, the physics-based model is further parametrized to investigate optimal design of the component. The parametrization is done in a probabilistic sense meaning that each of the model parameters is assigned the corresponding probability distribution given the experts (designers) knowledge. Such a description then leads to the stochastic-physics based model that is inherently high-dimensional compared to its deterministic counterpart, and thus cannot be discretized and evaluated in a real time. To reduce the computational effort, in this chapter the so-called data driven surrogate models are introduced. Subsequently, the transition from a finite element model towards a data driven surrogate model by the use of designs of experiments is explained.

2.1 Vehicle crashworthiness optimization

In crashworthiness optimization of vehicles one desires to minimize vehicle weight, while maximizing the energy absorption during a crash. However, both of the mentioned requirements are contradictory. The vehicle weight reduction on the one hand is a known strategy to address growing concerns about fuel consumption and CO_2 emissions, whereas weight reduction on the other hand reduces the structural performance. When comparing vehicles of different weight at the same speed, heavier vehicles have a higher kinetic energy and can therefore recover more energy during periods of deceleration. However, the opposite holds during the acceleration phase. In this light, the problem of optimizing the vehicle weight is not easy. Current trends are to analyze the individual components of the overall system (i.e. vehicle), and thus apply weight reduction on each of the components separately while keeping structural performance of the overall system. One way of achieving this is by use of the structural optimization that is focusing on the optimization of the shape, size and topology of the components [59, 103].

With an increased focus of the automotive industry on sustainability in recent years, structural optimization of vehicle components goes beyond sole weight reduction, as environmental impact of the design starts playing an important role. In the context of a circular economy the use of advanced or additive manufacturing technologies based on the multi-material concept is therefore gaining interest [104]. Novel manufacturing processes enrich the design capabilities of structural components and hence allow for an increased focus on a high strength-to-weight ratio of the structural component [105]. The latter is achieved by using materials that also have a high strength-to-weight ratio and accompanying formability. Examples of such materials may range from traditional high-strength steels to conventional aluminum or magnesium alloys or composites [1, 13, 35, 105–108]. The choice of material, manufacturing process and the accompanying structural component design optimization are yet open.

In this thesis the focus is devoted to the design of two vehicle components that form an integral part of the vehicle structural body: the B-pillar and bumpers. The B-pillar is the vertical or nearly vertical roof support pillar located in the car's midsection. The bumper is a structural body across the front or rear of a car and is characterized by a location dependent design. The main function of these two components is to absorb as much energy as possible during the crash, and hence prevent the impact energy to transfer to the other vehicle components (e.g. the battery) and passengers. In this manner passenger safety is ensured.

As explained previously, weight reduction of the vehicle can be done by focusing on its individual components, here to be understood solely as either the B-pillar or bumper. From a structural optimization point of view each of the mentioned components can be optimized by mathematically describing the desired mechanical safety or lightweightness objective as a function of the specific quantity of interest (QoI). The goal is therefore to estimate the QoI given all possible variations (changes) in the design. Once the QoI is generalized to all variations in the structural design, one can improve the structural design by maximizing or minimizing the corresponding

objective function. To predict the QoI – the main goal of this thesis – one has to first formulate a model that can mimic the component behavior, i.e. the QoI, given design variations and expected loading/boundary conditions, which further can be adopted as the model parameters. Examples of these parameters can be anything like geometrical features, or material characteristics, whereas the QoI of the component may be the energy absorption during a crash, maximal deformation of a part or material damage, among others.

In the scientific society today the QoI estimation given the design variations is usually made with the help of the physics-based approach, i.e. the mathematical formulation of the component behavior based on the first principles. In other words, for a predefined load case (crash) one has to solve the corresponding time-dependent partial differential equations that describe the non-linear dynamics behavior of the component. This is usually done in a finite element setting [8, 10] by using explicit or implicit procedures. However, there are few key challenges in this type of modeling and discretization such as: i) the definition of the constitutive law that characterizes material behavior especially in a multi-material design approach ii) prescription of possibly unknown material/structural parameters or loads/boundary conditions in the model iii) generalized modeling of the design variations in a geometric sense (e.g. change of shape) iv) efficient estimation of QoI given high-dimensionality of the discretized problem and small time step size. The first and second challenge are generally tackled by calibrating existing parametrized (non)linear constitutive laws (i.e. prescribed stress-strain relations) [37, 39] to experimentally obtained data sets, generally obtained from tensile and bending tests [13]. The third challenge largely depends on engineering experience and is thus subject to interpretation. Efficient estimation of the QoI is of key importance for a real-time optimization process in which user can directly interact with the structural design and for any introduced variation obtain the global optimum of the selected objective. Therefore, in this chapter the focus is on the fourth mentioned challenge assuming that the first three are already resolved.

2.2 Constitutive modeling of high-strength steels

To compensate for body weight reduction and simultaneously ensure vehicle crashworthiness by high energy absorption during a crash, the use of high strength steels in structural vehicle design is of great interest [1, 13, 35, 108]. For example, one requires the use of low hardness steels in the bottom part of a B-pillar due to the energy absorption requirement. On the other hand, the upper B-pillar section has to be made of high hardness steel so that the intrusion resistance is increased and hence passenger safety ensured [1, 109]. To satisfy previously mentioned requirements the B-pillar is made of a 22MnB5 high-strength steel (commercially denoted to as Usibor® 1500) in various hardness grades [1]. The 22MnB5 steel is therefore considered throughout this work as the chosen material for all given applications.

In order to calculate the response of high-strength steel structures in crash test scenarios and to analyze the vehicle component behavior more accurately, one has to first

understand the material behavior under prescribed loading conditions (i.e. monotonic or cyclic load). This is achieved by defining the constitutive relation between two physical quantities namely stress and strain, or force and displacement. For high-strength steel this relationship, i.e. the S-S (stress-strain) curve, is known to be nonlinear, meaning that the material under monotonic load undergoes irreversible changes. In the S-S curves of most metals and alloys obtained by a uniaxial tensile test, the yielding phenomenon is observed as a gradual transition from elastic to elastic-plastic deformation with a steadily increasing flow stress (hereinafter referred to as continuous yielding). The onset of plastic deformation in the 22MnB5 steel is described by a generalized Yld2000-2d yield function [110]. Initially proposed by [111], this yield criterion is generally applicable for body-centered cubic (BCC) and face-centered cubic (FCC) crystalline material structures. In this work the focus is put only on BCC as the general structure for high-strength steels [111–113]. The yield function ϕ_{y} is assumed to be an isotropic approximation of the Hershey yield criterion:

$$\phi_{\mathrm{y}} = \overline{\sigma} - \overline{\sigma}_{\mathrm{Y}} \leq 0 \tag{2.1}$$

with $\overline{\sigma}_{\mathrm{Y}}$ being the yield stress. Here, the equivalent stress $\overline{\sigma}$ is given by

$$2\overline{\sigma}^{a_{\mathrm{H}}} = |\sigma_1 - \sigma_2|^{a_{\mathrm{H}}} + |\sigma_2 - \sigma_3|^{a_{\mathrm{H}}} + |\sigma_3 - \sigma_1|^{a_{\mathrm{H}}}, \tag{2.2}$$

in which σ_1, σ_2 and σ_3 are the principal stresses, $|\cdot|$ is the absolute value and $a_{\mathrm{H}} = 6$ is the exponent value commonly used for BCC structures [1, 112]. For $\phi_{\mathrm{y}} = 0$ the equivalent stress equals the yield stress, hereby denoting the onset of plastic deformation. However, the 22MnB5 steel is subject to work hardening under plastic deformation and hence to describe continuous yield one has to specify the corresponding hardening model. As the S-S curve for 22MnB5 (for various hardness grades) shows a sudden gradient increase under continuous yielding [1], one may model the strain hardening behavior as a linear combination of the Voce and Swift hardening laws:

$$\sigma_{\mathrm{SV}}\left(\overline{\varepsilon}^{\mathrm{p}}\right) = \left(1 - \lambda_{\mathrm{m}}\right)\sigma_{\mathrm{S}}\left(\overline{\varepsilon}^{\mathrm{p}}\right) + \lambda_{\mathrm{m}}\sigma_{\mathrm{V}}\left(\overline{\varepsilon}^{\mathrm{p}}\right), \tag{2.3}$$

in which $\overline{\varepsilon}^{\mathrm{p}}$ is the accumulated equivalent plastic strain [1, 114, 115], σ_{V} is the yield stress described by Voce law [114] and σ_{S} is the yield stress described by Swift law [115]. Here, λ_{m} represents the non-linear mixing parameter

$$\lambda_{\mathrm{m}} = \lambda_{\mathrm{m}}(\overline{\varepsilon}^{\mathrm{p}}) = \lambda_{\mathrm{m,final}} + \left(\lambda_{\mathrm{m,initial}} - \lambda_{\mathrm{m,final}}\right) e^{-a_{\mathrm{shape}}\overline{\varepsilon}^{\mathrm{p}}}, \tag{2.4}$$

with the shape parameter a_{shape} controlling the effect of the equivalent plastic strain on the change of the mixing parameter ranging from the lower ($\lambda_{\mathrm{m,initial}}$) to the larger ($\lambda_{\mathrm{m,final}}$) plastic strain regions. With $\lambda_{\mathrm{m,initial}} > \lambda_{\mathrm{m,final}}$ the mixing parameter λ_{m} deteriorates for an increasing equivalent plastic strain, giving more significance to the Swift hardening law in Eq. (2.3).

According to the Voce hardening law [114] the yield stress is given as

$$\sigma_{\mathrm{V}}(\overline{\varepsilon}^{\mathrm{p}}) = k_{\mathrm{V}} + Q_{\mathrm{V}}(1 - e^{-b_{\mathrm{V}}\overline{\varepsilon}^{\mathrm{p}}}) + a_{\mathrm{V}}\overline{\varepsilon}^{\mathrm{p}}, \tag{2.5}$$

for the free model parameters k_{V}, Q_{V}, a_{V}, b_{V} and the asymptotic hardening rate a_{V} [1, 108]. Furthermore, the Swift law describes the yield stress as:

$$\sigma_{\mathrm{S}}(\overline{\varepsilon}^{\mathrm{p}}) = k_{\mathrm{S}}(\overline{\varepsilon}^{\mathrm{p}} + \varepsilon_{0,\mathrm{S}})^{n_{\mathrm{S}}}. \tag{2.6}$$

where k_{S}, $\varepsilon_{0,\mathrm{S}}$, and n_{S} are free model parameters [115].

With an increase in plastic strain and thus under continued yielding, the risk for material damage increases. Although material damage strongly depends on material irregularities and discontinuities, such defects are generally difficult to realistically prescribe in a finite element model. One way of introducing a detailed damage description is via the computationally expensive multi-scale analysis [59, 116], which is not within the scope of this work. In the further text it is assumed that the homogenized damage model is sufficient enough to represent the damage severity in relation to the equivalent plastic strain $\overline{\varepsilon}^{\mathrm{p}}$.

2.2.1 Material damage model

Given plastic strain dependency of the material, one may prescribe the severity of material damage as an accumulation over all plastic steps $\mathrm{d}\overline{\varepsilon}^{\mathrm{p}}$. In other words, the material damage accumulation can be written in an integral form:

$$d_{\mathrm{f}} = \int_0^{\overline{\varepsilon}^{\mathrm{p}}} \frac{\mathrm{d}\overline{\varepsilon}^{\mathrm{p}}}{\overline{\varepsilon}_{\mathrm{f}}}, \tag{2.7}$$

where $d_{\mathrm{f}} \in [0,1]$ denotes the fracture indicator and $\overline{\varepsilon}_{\mathrm{f}}$ is the critical strain to fracture. Accordingly, $d_{\mathrm{f}} = 1$ indicates the onset of material fracture and thus the onset of its loss in strength [108]. To evaluate the damage one requires knowledge on the critical equivalent fracture strain $\overline{\varepsilon}_{\mathrm{f}}$. For 22MnB5 steel this quantity can be modeled by the modified strain-based formulation of the Mohr-Coulomb (mMC) ductile fracture model [117], which according to [1] properly captures the fracture behavior of 22MnB5 steel. In contrast to the traditional stress-based Mohr-Coulomb fracture criterion, the modified version provides a strain-based formulation that can be written in the form:

$$\begin{aligned}\overline{\varepsilon}_{\mathrm{f}}\left(\eta,\overline{\theta}\right) = \Bigg[\frac{k_{\mathrm{S}}}{c_2}\left(c_3 + \frac{\sqrt{3}\left(c_\theta - c_3\right)}{2-\sqrt{3}}\left(\sec\frac{\overline{\theta}\pi}{6} - 1\right)\right) \\ \cdot\left(\sqrt{\frac{1+c_1^2}{3}}\cos\frac{\overline{\theta}\pi}{6} + c_1\left(\eta + \frac{1}{3}\sin\frac{\overline{\theta}\pi}{6}\right)\right)\Bigg]^{\frac{1}{n_{\mathrm{S}}}},\end{aligned} \tag{2.8}$$

given the stress triaxiality

$$\eta = \frac{\sigma_{\mathrm{m}}}{\overline{\sigma}}, \tag{2.9}$$

in which σ_{m} is the hydrostatic stress and $\overline{\sigma}$ is the Von Mises equivalent stress. The normalized Lode angle $\overline{\theta}$ in Eq. (2.8) is defined by

$$\overline{\theta} = 1 - \frac{2}{\pi}\arccos\left(\frac{r}{\overline{\sigma}}\right)^3 \text{ with } \quad r := \left(\frac{27}{2}\det\,\sigma_{\mathrm{s}}\right)^{\frac{1}{3}}, \tag{2.10}$$

with σ_{s} being the deviatoric stress. Additionally, in Eq. (2.8) the parameter c_θ is defined as:

$$c_\theta = \begin{cases} 1, & \text{if } \overline{\theta} \geq 0 \\ c_4, & \text{else} \end{cases} \tag{2.11}$$

whereas the parameter pair k_S and n_S are predefined in the Swift power law in Eq. (2.6). The remaining free parameters $c_1 - c_4$ are experimentally determined and are hardness grade dependent as explained in [1, 108].

Given the material hardening and damage laws, one is able to define and assess the large deformation behavior of vehicle components made of 22MnB5 steel by a finite element approximation as described in the further text.

2.3 Nonlinear dynamic finite element analysis of vehicle components

In underlying mechanics describing crash tests one can generally distinguish three types of non-linearity: i) geometrical non-linearity [108, 118], as caused by large deformations in which one may further distinguish large strains and large displacements; ii) non-linear material behavior, as previously discussed in Section 2.2 and iii) non-linearity present in modeling contacts i.e. the interaction of the components during the crash [119]. All present non-linearities can be modeled by means of mathematical models that are guided by the first principles in the framework of continuum mechanics and irreversible thermodynamics assuming isothermal conditions. By use of the principle of virtual work [11], one may derive the equation of motion by using either total [120] or updated Lagrangian approach [120]. Due to spatial and time dependence of involved terms in both formulations one requires spatial discretization by the finite element method [11] and either explicit [11, 121–124] or implicit [11, 124, 125] time discretization. The finite element discretization of the principle of virtual work with n_e elements and n_n nodes results in a matrix equation

$$\boldsymbol{M}\ddot{\boldsymbol{u}}(\tau) + \boldsymbol{C}\dot{\boldsymbol{u}}(\tau) + \boldsymbol{K}(\boldsymbol{u})\boldsymbol{u}(\tau) = \boldsymbol{\mathcal{F}}(\tau), \quad \boldsymbol{u}(0) = \boldsymbol{u}_0, \tag{2.12}$$

in which $\boldsymbol{u}(\tau) \in \mathbb{R}^{n_n}$ is the unknown displacement in n_n FEM nodes at time τ, $\boldsymbol{M} \in \mathbb{R}^{n_n \times n_n}$ is the mass matrix, $\boldsymbol{C} \in \mathbb{R}^{n_n \times n_n}$ is the damping matrix, $\boldsymbol{K}(\boldsymbol{u}) \in \mathbb{R}^{n_n \times n_n}$ is the stiffness matrix and $\boldsymbol{\mathcal{F}}(\tau) \in \mathbb{R}^{n_n}$ denotes the external excitation. Due to the non-linear constitutive laws describing 22MnB5 steel the stiffness matrix $\boldsymbol{K}(\boldsymbol{u})$ nonlinearly depends on the displacement. The provided initial condition $\boldsymbol{u}_0$ indicates the displacement $\boldsymbol{u}(\tau)$ at the onset of the analysis ($\tau = 0$).

Given that n_n may quickly grow for larger and more finely discretized load cases, an efficient integration scheme is needed to solve Eq. (2.12). Yet, Eq. (2.12) can be solved in both implicit and explicit form. Examples of implicit schemes are the first order solving backward Euler method or the second order Newmark scheme [124, 125]. Examples of explicit integration methods are the first order forward Euler method or the Newmark explicit method [121–124]. Implicit integration schemes are primarily used for static and quasi-static analyses, whereas explicit procedures are common for dynamic analyses due to their ability to solve highly nonlinear transient problems in a less accurate but computationally more efficient sense [9, 119]. Explicit integration schemes are generally also more time-efficient than implicit schemes due to the ability to parallelize these schemes [119]. In this work the central difference explicit Virtual Performance Solution (VPS) crash solver PAM-CRASH [126–128] is

used to solve Eq. (2.12). Accordingly, time domain is discretized in a finite number of dimensionless time steps n_t.

The model presented in Eq. (2.12) can be used to describe one particular example of a structural design. In other words, this model does not include possible design variations or uncertainties in the prescribed boundary conditions or load cases. Thus, any parameter change would result in a new model of the type given in Eq. (2.12) and thus would result in a new explicit FEM simulation. This is computationally expensive. As in design for crashworthiness one would like to optimize the model parameters given a specific objective function, the variation of the parameter set has to be directly incorporated in the model in Eq. (2.12).

To include all possible variations and existing uncertainties in the model, one may rewrite Eq. (2.12) as:

$$\boldsymbol{M}(\omega)\ddot{\boldsymbol{u}}(\omega,\tau)+\boldsymbol{C}(\omega)\dot{\boldsymbol{u}}(\omega,\tau)+\boldsymbol{K}(\boldsymbol{u}(\omega,\tau))\boldsymbol{u}(\omega,\tau)=\boldsymbol{\mathcal{F}}(\omega,\tau),\boldsymbol{u}(0)=\boldsymbol{u}_0,a.s. \quad (2.13)$$

in which all parameters (mass, damping, stiffness, etc.) and the loading conditions are modeled as uncertain in a joint probability space defined by the triplet $(\Omega,\mathfrak{F},\mathbb{P})$. Here, ω represents one elementary event in the sampling space Ω, $\mathfrak{F}$ is the sigma-algebra and $\mathbb{P}$ is the probability measure. Hence, for one specific elementary event ω (further called sample) one can obtain a deterministic model as presented in Eq. (2.12). Although the previously described model can include design variations, its stochastic formulation makes the dynamic analysis more cumbersome than in the specific design case. Namely, next to the spatial and time discretizations, the model in Eq. (2.13) also requires parametric discretization, i.e. the discretization of the probabilistic space. This can be achieved by the so-called stochastic finite element method [16–18, 74] in which the probabilistic variables are approximated by the orthogonal stochastic shape functions alike the finite element ones. However, such an approach would require significant changes in the existing FEM simulation software. Another way to solve Eq. (2.13) is to use the so-called data driven approach. This approach relies on the approximation of the solution or QoI of Eq. (2.13) in a functional approximation form (also known as a surrogate model), i.e. the function of the design parameters. This function is estimated by use of the sampling data collected by sampling the random parameters in Eq. (2.13). Such an approach is known to be non-intrusive as it does not require change of the existing software as can be seen in Fig. 2.1. The QoI must therefore be directly expressed as a function of the design parameters. Here, $G(a^\mathrm{f},b^\mathrm{f},c^\mathrm{f})$ represents a detailed finite element model of the crash simulation that depends on few input parameters , e.g. a^f, b^f and c^f. The variation of parameters is modeled as shown on the left side of Fig. 2.1 with the help of a probability density functions based on the experts/designer knowledge. Note that one could have also used intervals instead [35, 129, 130]. As the finite element model G is deterministic, its inputs are only samples of the parameter set, meaning that each change in the input would lead to a corresponding new expensive simulation. To circumvent this, the data collected by a few full simulations are used to train simpler analytic function G_S, also known as a surrogate model, that depends on the same parameter set, but is computationally cheaper than G for the desired accuracy level.

Although the use of a surrogate model reduces the computational demands, its application brings along several other issues. For example, the analytic function G_{S} is not a priori known and has to be selected among a vast number of candidates. Therefore, often is hard to find an optimal surrogate model. In practice one usually fixes the model form based on the previous experience. Next to this, once the model is chosen, one has to carefully select the sample points used to train the model. In other words, the number of samples as well as their positioning has to be optimized. When employing a sub-optimal surrogate model shape and/or sub-optimal sampling, one may risk poor accuracy of the surrogate approximation. Although the choice of the surrogate model family can be appropriate, its direct discretization may not be. For example, one may choose to have a model within same family (e.g. feed-forward NN) that is overly complex, which may lead to the over-fitting problem [87, 131, 132]. On the contrary, making the surrogate model too simple may lead to the under-fitting problem [87, 132]. To prevent an improper choice one strives for a data-driven algorithm that may automatically detect the proper complexity form of the surrogate model. Accordingly, the surrogate configuration can be adapted, hereby preventing over-fitting. In this thesis the focus is put on the NN family of models due to its proven general applicability and adaptivity in approximating both linear and non-linear functions (see Section 1.1).

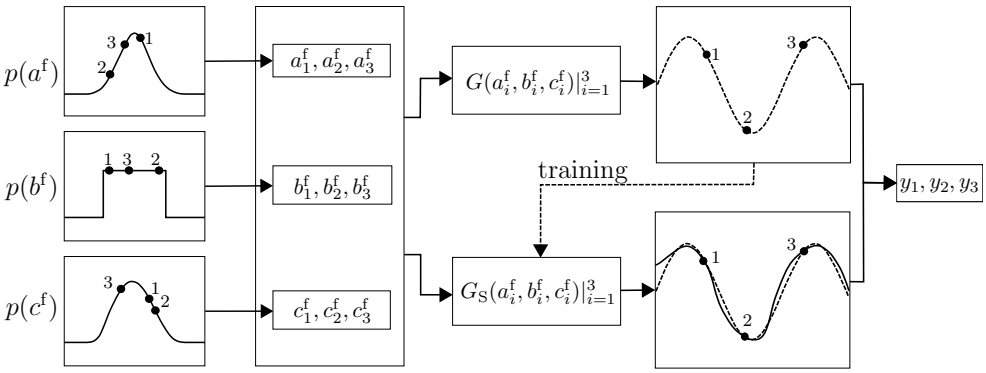


Figure 2.1: Overview of the process chain from continuous uncertain input parameters $(p(a^{\mathrm{f}}), p(b^{\mathrm{f}}), p(c^{\mathrm{f}}))$ defined by a probability density function on the parameter space, its discretization by sampling to sets of $(a_i^{\mathrm{f}}, b_i^{\mathrm{f}}, c_i^{\mathrm{f}})$, the corresponding FEM results $y_i = G(a_i^{\mathrm{f}}, b_i^{\mathrm{f}}, c_i^{\mathrm{f}})$ and the subsequent surrogate model giving $y_i \approx G_{\mathrm{S}}(a_i^{\mathrm{f}}, b_i^{\mathrm{f}}, c_i^{\mathrm{f}})$. For simplicity, time steps are disregarded in this example.

2.4 Surrogate modeling

The first step to adapt the parametrized finite element model towards a surrogate modeling approach is to write the QoI as a function of parameter set $\boldsymbol{q}$. By collecting all design parameters from Eq. (2.13) into a parameter set $\boldsymbol{q} \in \mathcal{Q}$[1] one may model $\boldsymbol{q}$

[1] $\mathcal{Q}$ is a space in which the parameters take a value, and can be a space of values described by real numbers, positive numbers, negative numbers, etc. depending on the properties of parameter $\boldsymbol{q}$.

as

$$\boldsymbol{q}(\omega) : \Omega \to \mathcal{Q} \tag{2.14}$$

i.e. a random vector with finite second order moments on a probability space $(\Omega, \mathfrak{F}, \mathbb{P})$ [80, 133]. In case that parameters $\boldsymbol{q}$ are spatially dependent as in a heterogeneous material [134], one may extend Eq. (2.14) to:

$$\boldsymbol{q}(x, \omega) : \mathcal{X} \times \Omega \to \mathcal{Q}, \tag{2.15}$$

in which $\mathcal{X}$ denotes the spatial domain of interest that may be discretized by the finite element approach. Note that in Eq. (2.12) the external influence $\boldsymbol{\mathcal{F}}$, as well as initial conditions $\boldsymbol{u}_0$ can also be included in the parameter set. The theory presented in the upcoming does not depend on this choice, and is general enough to cover all of the mentioned cases.

Given Eq. (2.12) and the model description of $\boldsymbol{q}$, one is interested in the estimation of the QoI $\boldsymbol{y}(\boldsymbol{q}, \tau)$ at the time moment $\tau \in \mathcal{T}$ described by a possibly non-linear operator G[2]. Moreover, one aims to approximate

$$\boldsymbol{y}(\tau) = G(\boldsymbol{u}(\tau), \boldsymbol{q}, \tau) \approx G_{\mathrm{S}}(\boldsymbol{q}, \tau) \tag{2.16}$$

by a surrogate function G_{S} that describes the time evolution of the QoI.

After the appropriate time discretization by an explicit method, Eq. (2.16) can be substituted by its incremental version

$$\boldsymbol{y}(\tau_\ell) = G_\ell(\boldsymbol{w}, \boldsymbol{q}, \tau_\ell), \quad \ell = 1, ..., n_{\mathrm{t}} \tag{2.17}$$

in which $\boldsymbol{w}$ represents the parameterization of the surrogate model. Here, n_{t} time increments are not necessarily taken as equidistant. Additionally, the QoI $\boldsymbol{y}$ is parametrically dependent and thus the continuous parameter space can be discretized by a sampling type of approach (e.g. by Latin Hypercube sampling [135]). Let $\boldsymbol{q}_i := [q_i(\omega_j)]_{j=1}^{n_{\mathrm{f}}}$ be the set of n_{f} design parameters at the sample point i, then $\boldsymbol{y}(\tau_\ell)$ can be written in the complete discretized form as:

$$\forall i = 1, ..., n_{\mathrm{b}}, \quad \forall \ell = 1, ..., n_{\mathrm{t}}, : \quad \boldsymbol{y}(\boldsymbol{q}_i, \tau_\ell) \in \mathcal{Y}_{n_{\mathrm{o}}}, \tag{2.18}$$

in which n_{b} is the sampling batch size, i.e. part of the complete data set counting n_{d} samples obtained by repeated evaluation of the finite element method. From the previous equation, it is immediately seen that the QoI is defined by a collection of finite element solutions also known as designs of experiments (DOEs).

Once the data set in Eq. (2.18) is collected, one may estimate the unknown parameters w in G_ℓ by a regression type of approach [87, 129, 136]. To obtain the well-posed solution for w one has to collect enough data points. As not only the number of sampling points but also their location in the probability space matters, one can have different strategies designing such an experiment as further elaborated. The optimal choice of the sampling procedure for surrogate modeling applications is becoming an

[2]This operator is an abstract notation for the definition of e.g. the equivalent plastic strain, equivalent stress or any other quantity one can indirectly estimate by FEM.

increasingly debated topic in recent years [137–141]. This choice strongly depends on the type of parameters present in the parameter set, as well as on their number.

For an unbounded input parameter set one is required to model random variables using a boundless probability density function and subsequently sample by Monte-Carlo simulation [142, 143]. In this work, only bounded parameter sets are considered. In case of bounded RVs with a uniform probability distribution one may use a space filling full factorial sampling procedure [144]. This deterministic sampling in a tensor product form defines a mesh grid over the pre-determined input parameter discretization [144], see Fig. 2.2a. Alternatively a general random uniform Monte-Carlo (MC) sampling [142] may be used, see Fig. 2.2b. However, among others, a more progressive space-filling procedure can be achieved by Latin Hypercube (Fig. 2.2c), Sobol sequence (Fig. 2.2d) or Halton sequence (Fig. 2.2e) sampling [135, 145, 146]. For visualization purposes an overview of the aforementioned sampling procedures for a two-dimensional RV over $n_\mathrm{d} = 4$ samples has been represented in Fig. 2.2. The quasi-

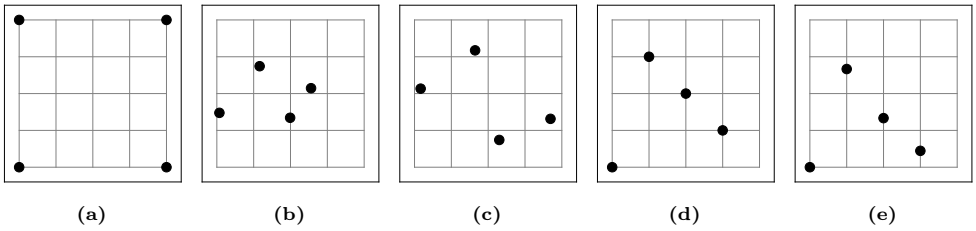


Figure 2.2: Sampling of random vector described by two random variables (4 samples): (a) Full factorial, (b) uniform random, (c) Latin Hypercube, (d) Sobol sequence and (e) Halton sequence sampling procedures. Additionally, the hyper-cubes are indicated using the given grid lines.

random Latin Hypercube procedure divides the complete sampling interval into n_d equally probable sub-intervals (hypercubes) for n_d samples in all n_f dimensions, see Fig. 2.2. Subsequently, the procedure fills the parameter space by a joint-uniform random sampling in the sub-intervals. In contrast to the Latin Hypercube procedure, both the Sobol and the Halton sequence are quasi-random low-discrepancy sequences [135, 145–148]. The discrepancy is a measure of (non)uniformity of the filled sample space: the lower the discrepancy, the more equally distributed the sample points in the sample space are. The most widely studied measure for discrepancy is the star discrepancy, which is characterized by having each subset anchored at the origin [149, 150]. The quasi-random low-discrepancy Sobol sequence enforces a sample space coverage over the multi-dimensional space and subdivides each dimension in 2^κ points, with κ being the selection parameter for the number of samples taken. This yields that the number of samples is constraint by $n_\mathrm{d} = 2^\kappa$. The quasi-random low-discrepancy Halton sequence is based on coprime numbers and reduces to a Van der Corput sequence in a single dimensional case [145, 151, 152]. Alike Sobol, the Halton sequence space-covering quality strongly depends on the number of samples considered, but also on the chosen sample base. Hence, in Fig. 2.2e a $(2, 3)$-Halton sequence provides insufficient space coverage. Here, $(2, 3)$ denotes the base of respectively the first and second dimension. Both the quasi-random Sobol and the Halton sequence remain

unchanged after re-sampling. In contrast, the Latin Hypercube procedure may vary after re-sampling. Hence, to ensure uniform space coverage for arbitrary numbers of samples n_{d} in this thesis the Latin Hypercube sampling method is employed.

3 Deterministic surrogate modeling

In this chapter the theory of the deterministic type of training deep neural network surrogate models is explained. Due to their recent popularity, universal capability of approximation and computational efficiency, these soft computing methods are further investigated on application examples that follow the theory described in Chapter 2.

3.1 Neural network surrogate model

The previously described physics-based analyses require a large number of simulation runs, where each run takes a different combination of design parameters as the inputs to the FEM simulation. In a real industrial application only one simulation run can take days to finish. To address this issue, one may construct a computationally cheaper model, also known as a surrogate or meta-model. Subsequently, the trained surrogate model can be deployed to replace the original computer simulation in emulating the physics-based environment as accurately as possible in the design task. At the same time surrogate models also need to be interpretable. In this thesis a NN family of surrogate models is considered. As the construction of such a model is based on data driven methods, in particular supervised learning [87], the following text is focusing on establishing a unified mathematical framework of both the model and the training method.

Let be given the QoI $\boldsymbol{y}(\boldsymbol{q}(\omega), t)$ at the time moment $t := \tau_\ell$. The goal is to approximate $\boldsymbol{y}(\boldsymbol{q}) := \boldsymbol{y}(\boldsymbol{q}(\omega), t)$ by a NN for a given data set $(\boldsymbol{q}_i, \boldsymbol{y}_i), i = 1, ..., n_\mathrm{b}$ collected at the time moment t.

Before showing how this works, one may first shortly review the definition of a NN. If both the input and output quantities are defined by real valued scalars, then one may approximate $y(q)$ by a function

$$\hat{y}(q) = \sigma(q), \tag{3.1}$$

representing a single input single output neuron described by the activation function

$$\sigma : \mathbb{R} \to \mathbb{R}. \tag{3.2}$$

An example of such activation function is the sigmoid function, generally described for an arbitrary scalar $x \in \mathbb{R}$ by (see Fig. 3.1a):

$$\sigma(x) = \frac{1}{1 + e^x}, \quad \sigma : \mathbb{R} \to [0, 1], \tag{3.3}$$

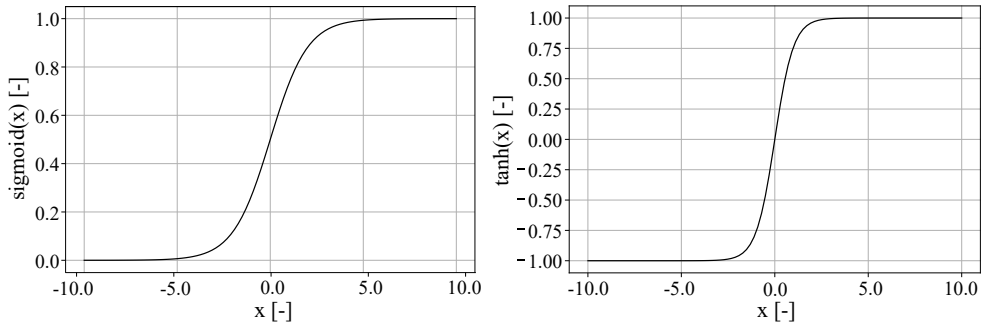
or the hyperbolic tangent function (see Fig. 3.1b):

$$\sigma(x) = \frac{e^{2x} - 1}{e^{2x} + 1}, \quad \sigma : \mathbb{R} \to [-1, 1], \tag{3.4}$$

which are applied in an element-wise form over the input x if x represents a vector, matrix or tensor. Both the sigmoid and hyperbolic tangent functions are part of the smooth continuous (bounded) sigmoidal function family. In [153] the concept of sigmoidal functions is further generalized and prescribes the definition of an activation function to be any measurable function $\sigma(\cdot) : \mathbb{R} \to [a, b]$ that satisfies:

$$\lim_{x \to -\infty} \sigma(x) = a \quad \wedge \quad \lim_{x \to +\infty} \sigma(x) = b \tag{3.5}$$

with $a \neq b$.



(a) The sigmoid activation function.

(b) The hyperbolic tangent activation function.

Figure 3.1: The sigmoid and hyperbolic tangent activation functions, as applied in an LSTM cell, plotted for $x = [-10.0, 10.0]$.

As single inputs are not common in practice, the previous definition of a neuron (Eq. (3.1)) can be expanded to a multi-input single output neuron

$$\hat{y}(\boldsymbol{q}) = \sigma\left(\sum_{i=1}^{n_\mathrm{f}} w_{\mathrm{s},i} q_i + w_\mathrm{b}\right), \tag{3.6}$$

in which $\boldsymbol{q} \mapsto \mathbb{R}^{n_\mathrm{f}}$ is the input vector, $w_{\mathrm{s},i} \in \mathbb{R}, i = 1, ..., n_\mathrm{f}$ are the unknown linear scaling weights and $w_\mathrm{b} \in \mathbb{R}$ is the unknown bias. In a vector form the previous equation can be also written as

$$\hat{y}(\boldsymbol{q}) = \sigma(\langle \boldsymbol{w}_\mathrm{s}, \boldsymbol{q} \rangle + w_\mathrm{b}) \tag{3.7}$$

with $\boldsymbol{w}_\mathrm{s} := [w_{\mathrm{s},1}, ..., w_{\mathrm{s},n_\mathrm{f}}] \in \mathbb{R}^{n_\mathrm{f}}$ being the unknown vector of linear scaling weights, and $\langle \cdot, \cdot \rangle$ being the Euclidean inner product. A more intuitive graphical representation of Eq. (3.7) is given in Fig. 3.2. By collecting all unknown weights into $\boldsymbol{w} := [\boldsymbol{w}_\mathrm{s}\ w_\mathrm{b}] \in$

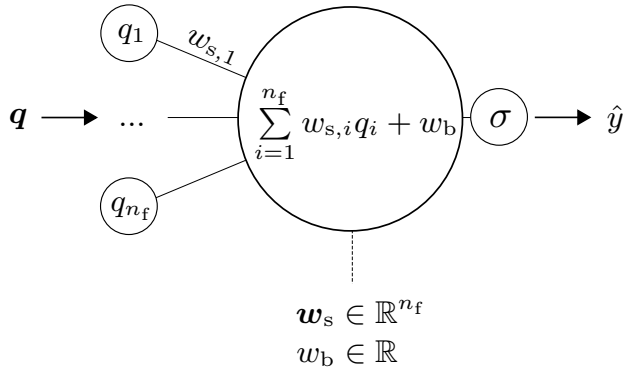


Figure 3.2: A multi-input single output feed-forward NN, visualizing the representation of Eq. (3.7).

$\mathbb{R}^{n_\mathrm{f}+1}$, one may rewrite the previous equation as

$$\hat{y} = \sigma(\boldsymbol{\Phi}\boldsymbol{w}) \tag{3.8}$$

in which $\boldsymbol{\Phi} := [1\ \boldsymbol{q}] \mapsto \mathbb{R}^{1\times(n_\mathrm{f}+1)}$. By defining the local state $s := \boldsymbol{\Phi}\boldsymbol{w}$, the previous equation leads to

$$\hat{y} = \sigma(s). \tag{3.9}$$

The previously described multi-input single output neuron often serves as a basis for the description of the NN architectures used to approximate the vector valued QoI. An example is the simple single layer multi-output feed-forward NN, the output of which is element-wise described by

$$\hat{y}_k(\boldsymbol{q}) = \sum_{j=1}^{n_\mathrm{m}} w^{(2)}_{\mathrm{s},jk}\sigma\left(\sum_{i=1}^{n_\mathrm{f}} w^{(1)}_{\mathrm{s},ji}q_i + w^{(1)}_{\mathrm{b},j}\right) + w^{(2)}_{\mathrm{b},k}, \quad k = 1, ..., n_\mathrm{o} \tag{3.10}$$

with n_m being the number of neurons in the layer. Here, $w^{(1)}_{\mathrm{s},ji} \in \mathbb{R}$ is the set of unknown linear scaling weights in the input layer, whereas $w^{(2)}_{\mathrm{s},jk} \in \mathbb{R}$ is the set of unknown linear scaling weights in the output layer. Similar holds for biases $w^{(1)}_{\mathrm{b},j} \in \mathbb{R}$ and $w^{(2)}_{\mathrm{b},k} \in \mathbb{R}$, respectively. These are further collected to

$$\boldsymbol{W}^{(1)}_\mathrm{s} \in \mathbb{R}^{n_\mathrm{f}\times n_\mathrm{m}}, \boldsymbol{W}^{(1)}_\mathrm{s} := [w^{(1)}_{\mathrm{s},ij}], \quad i = 1, ..., n_\mathrm{f}, j = 1, ..., n_\mathrm{m},$$

$$\boldsymbol{W}^{(2)}_\mathrm{s} \in \mathbb{R}^{n_\mathrm{m}\times n_\mathrm{o}}, \boldsymbol{W}^{(2)}_\mathrm{s} := [w^{(2)}_{\mathrm{s},ij}], \quad i = 1, ..., n_\mathrm{m}, j = 1, ...n_\mathrm{o}$$

$$\boldsymbol{w}^{(1)}_\mathrm{b} \in \mathbb{R}^{n_\mathrm{m}}, \quad \boldsymbol{w}^{(1)}_\mathrm{b} := [w^{1)}_{\mathrm{b},i}], \quad i = 1, ..., n_\mathrm{m}$$

and

$$\boldsymbol{w}^{(2)}_\mathrm{b} \in \mathbb{R}^{n_\mathrm{o}}, \quad \boldsymbol{w}^{(2)}_\mathrm{b} := [w^{(2)}_{\mathrm{b},i}], \quad i = 1, ..., n_\mathrm{o}$$

such that

$$\hat{\boldsymbol{y}}(\boldsymbol{q}) = (\boldsymbol{W}^{(2)}_\mathrm{s})^T\boldsymbol{\sigma}\left((\boldsymbol{W}^{(1)}_\mathrm{s})^T\boldsymbol{q} + \boldsymbol{w}^{(1)}_\mathrm{b}\right) + \boldsymbol{w}^{(2)}_\mathrm{b} \tag{3.11}$$

holds. Here, $\boldsymbol{\sigma}(\boldsymbol{h})$ is a vector valued function made of n_m scalar valued functions on $\mathbb{R}$:

$$\boldsymbol{\sigma}(\boldsymbol{h}) = [\sigma(h_1), ..., \sigma(h_{n_\mathrm{m}})] \tag{3.12}$$

acting element-wise on the input argument $\boldsymbol{h} \mapsto \mathbb{R}^{n_\mathrm{m}}, \quad \boldsymbol{h} := (\boldsymbol{W}^{(1)}_\mathrm{s})^T\boldsymbol{q} + \boldsymbol{w}^{(1)}_\mathrm{b}$, also known as the hidden state. To simplify the previous notation, one may introduce the affine map:

$$A_l : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}, \quad n_l \in \mathbb{N},\ l = 1, 2 \tag{3.13}$$

of the form $\boldsymbol{x} \mapsto \boldsymbol{W}^{(l)}\boldsymbol{x} + \boldsymbol{b}^{(l)}$ for a matrix $\boldsymbol{W}^{(l)}$ and vector $\boldsymbol{b}^{(l)}$. Here $n_0 = n_\mathrm{f}$, $n_1 = n_\mathrm{m}$ and $n_2 = n_\mathrm{o}$. Then, a single layer NN is a function

$$F_\mathbf{w} : \mathbb{R}^{n_\mathrm{f}} \to \mathbb{R}^{n_\mathrm{o}} \tag{3.14}$$

parametrized by all unknown weights in different layers collected to $\mathbf{w}$ such that

$$F_{\mathbf{w}}(\boldsymbol{x}) = (A_2 \circ \boldsymbol{\sigma} \circ A_1)(\boldsymbol{x}) \tag{3.15}$$

holds. Here, $\boldsymbol{\sigma}(\boldsymbol{h})$ is a vector valued function made of n_{m} scalar valued functions on $\mathbb{R}$ given in Eq. (3.12) and acting element-wise on the input argument $\boldsymbol{h} := A_1(\boldsymbol{x})$. Following this, the vector-valued QoI can be approximated as

$$\hat{\boldsymbol{y}}(\boldsymbol{q}) = F_{\mathbf{w}}(\boldsymbol{q}). \tag{3.16}$$

A function in Eq. (3.15) is known to be a universal approximator if the activation function is of the sigmoidal type, meaning that $\sigma(x) \to 1$ for $x \to \infty$ and $\sigma(x) \to 0$ for $x \to -\infty$. According to the universal approximation theory [25] a single hidden layer NN with an activation function that satisfies Eq. (3.5) is always capable to approximate a multi-variate continuous function with arbitrary accuracy given unconstrained width of the NN.

By generalizing Eq. (3.15), one may define a feed-forward NN as a function:

$$F_{\mathbf{w}} : \mathbb{R}^{n_{\mathrm{f}}} \to \mathbb{R}^{n_{\mathrm{o}}} \tag{3.17}$$

in the form

$$F_{\mathbf{w}}(\boldsymbol{x}) = (A_{n_{\mathrm{L}}} \circ \boldsymbol{\sigma} \circ A_{n_{\mathrm{L}}-1} \circ ... \circ \boldsymbol{\sigma} \circ A_1)(\boldsymbol{x}). \tag{3.18}$$

Here, $A_l : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}, \quad n_l \in \mathbb{N}$ is affine for each $l = 1, ..., n_{\mathrm{L}}$ with $n_0 = n_{\mathrm{f}}$ and $n_{\mathrm{L}} = n_0$. Thus, the vector-valued output is given by

$$\hat{\boldsymbol{y}}(\boldsymbol{q}) = F_{\mathbf{w}}(\boldsymbol{q}). \tag{3.19}$$

This model can further be used to approximate the QoI in Eq. (2.16).

The previous derivations are made for the approximation of QoI at the specific time moment. However, in a transient analysis one is interested in approximating the QoI in all observed time moments. One way of achieving this is to concatenate the output vectors at all time moments τ_ℓ, $\ell = 1, ..., n_{\mathrm{t}}$ to $\boldsymbol{y} \in \mathbb{R}^{n_{\mathrm{o}} n_{\mathrm{t}}}, \quad \boldsymbol{y} := [\boldsymbol{y}(\boldsymbol{q}, \tau_\ell)]$, $\ell = 1, ..., n_{\mathrm{t}}$. Thus, one may use a feed-forward NN of the form as given in Eq. (3.17) with the only difference that the output dimension is not n_{o} but $n_{\mathrm{o}} n_{\mathrm{t}}$. Another way to approximate the time series is to feed the NN with the input $\boldsymbol{q}$, and the QoI in the previous time moments in order to predict the next time moment. This is also known as time delayed feed-forward NN [154]. No matter what type of the NN architecture is chosen, for all of them the form of the approximation as given in Eq. (3.17) holds. The only difference lies in the input, and output dimension of the network. Thus, with this, also the number of unknown scaling weights and biases can vary.

In order to evaluate if the proposed model $F_{\mathbf{w}}$ is a good approximation, one requires a measure of the goodness of a model. This is achieved by defining a loss function:

$$\mathcal{J} : \mathbb{R}^{n_{\mathrm{y}}} \times \mathbb{R}^{n_{\mathrm{y}}} \to \mathbb{R}_+ \tag{3.20}$$

that measures the cost of the discrepancy between the real output $\boldsymbol{y} \in \mathbb{R}^{n_\mathrm{y}}$[1] and the approximated one $\hat{\boldsymbol{y}}$ given input and output observations $(\boldsymbol{q}_i, \boldsymbol{y}_i), i = 1, ..., n_\mathrm{b}$, respectively. The goal is to find a multi-layer feed-forward NN such that

$$\boldsymbol{y}_i \approx F_{\mathbf{w}}(\boldsymbol{q}_i) \tag{3.21}$$

holds. A typical empirical loss function that is used in the machine learning practice is the mean squared error [27, 87, 155] defined as:

$$\mathcal{J}(\mathbf{w}) = \frac{1}{n_\mathrm{b}} \sum_{i=1}^{n_\mathrm{b}} \|\boldsymbol{y}_i - F_{\mathbf{w}}(\boldsymbol{q}_i)\|_2^2. \tag{3.22}$$

The unknown weights are then found by solving the following optimization problem:

$$\mathbf{w}^* = \arg\min \mathcal{J}(\mathbf{w}), \tag{3.23}$$

as described in the following text.

3.2 Supervised training by point-estimation

Given the cost function in Eq. (3.22), for a single layer feed-forward NN one may estimate the unknown weights using a gradient descend based approach. Using Eq. (3.9) and Eq. (3.22) one may compute the gradient of the cost function with respect to the weights. For a 1$^{\mathrm{st}}$-order gradient descent optimization, one then updates the weights of NN as:

$$\mathbf{w}^{e+1} = \mathbf{w}^e - \lambda \frac{\partial \mathcal{J}}{\partial \mathbf{w}^e} \tag{3.24}$$

in which λ is the learning rate and e is the current iteration step. With a fixed learning rate, one may risk fast convergence to a local minimum [156, 157]. Therefore, one may look at more advanced convergence algorithms with an adapting learning rate as elaborated further in the text.

For a multi-layer NN as described in Eq. (3.18), one may propagate the gradient of the cost function backward through the networks layers. Thus, the gradient in the p^{th} layer can be expressed as:

$$\frac{\partial \mathcal{J}}{\partial \mathbf{w}^p} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{\sigma}(\boldsymbol{s}^{n_\mathrm{L}})} \odot \frac{\partial \boldsymbol{\sigma}(\boldsymbol{s}^{n_\mathrm{L}})}{\partial \boldsymbol{s}^{n_\mathrm{L}}} \frac{\partial \boldsymbol{s}^{n_\mathrm{L}}}{\partial \boldsymbol{\sigma}(\boldsymbol{s}^{n_\mathrm{L}-1})} \odot ... \odot \frac{\partial \boldsymbol{\sigma}(\boldsymbol{s}^{p})}{\partial \boldsymbol{s}^{p}} \frac{\partial \boldsymbol{s}^{p}}{\partial \mathbf{w}^{p}} \tag{3.25}$$

with n_L being the last NN layer, $\odot$ being the Hadamard product, and $\boldsymbol{s}^p := \boldsymbol{\Phi}^p \mathbf{w}^p$ being the weighted input of layer p.

Gradient descent algorithms as in Eq. (3.25) search for a local minimum of a differentiable function, here a mean-squared error, see Eq. (3.22) [158, 159]. To improve robustness and stability, various improvements to the 1$^{\mathrm{st}}$-order gradient descent algorithm can be made. An example of such is the adaptive moment (Adam) optimizer

[1] Please note that here the output dimension is denoted by n_y in order to generalize the approximation in the specific time moment to the approximation of the whole time series, e.g. $n_\mathrm{y} = n_\mathrm{o} n_\mathrm{t}$

[158]. The goal of the Adam optimizer is to adapt the learning rate over all iterations (in the machine learning community also known as epochs) according to an adapted moment and hence deliver a more robust solution to the iterative scheme. The Adam optimizer takes the general formulation:

$$\mathbf{w}^{e+1} = \mathbf{w}^{e} - \lambda \frac{\hat{\mathbf{m}}_1^{e+1}}{\sqrt{\hat{\mathbf{m}}_2^{e+1}} + \varrho}. \tag{3.26}$$

Here, λ is the initial learning rate, whereas ϱ is the constant blow-up prevention factor, usually in the order of 1e−8. Parameters $\hat{\mathbf{m}}_1^{e+1}$ and $\hat{\mathbf{m}}_2^{e+1}$ are respectively the first and second weight dependent gradients at the iteration step $e+1$:

$$\hat{\mathbf{m}}_1^{e+1} = \frac{\mathbf{m}_1^{e+1}}{1-\vartheta_1^{e+1}}, \qquad \hat{\mathbf{m}}_2^{e+1} = \frac{\mathbf{m}_2^{e+1}}{1-\vartheta_2^{e+1}}, \tag{3.27}$$

with

$$\begin{aligned} \mathbf{m}_1^{e+1} &= \vartheta_1 \mathbf{m}_1^{e} + (1-\vartheta_1) \left(\frac{\partial \mathcal{J}}{\partial \mathbf{w}^e} \right), \\ \mathbf{m}_2^{e+1} &= \vartheta_2 \mathbf{m}_2^{e} + (1-\vartheta_2) \left(\frac{\partial \mathcal{J}}{\partial \mathbf{w}^e} \right)^2. \end{aligned} \tag{3.28}$$

Here, ϑ_1 and ϑ_2 are the exponential decay rate factors set at $\vartheta_1 = 0.900$ and $\vartheta_2 = 0.999$, respectively. Although many other gradient-based optimization algorithms have been developed and may be suitable, the Adam optimizer has gained great popularity in recent years because of its robustness [35, 160–163], and thus is used in this thesis.

3.3 Recurrent neural networks

The system given in Eq. (3.18) describes a purely feed-forward parameter-response mapping, despite the QoI $\boldsymbol{y}_\ell$ being time dependent. As described previously a plain feed-forward NN could be applied per time step to describe the QoI over time. However, given a structural nonlinear finite element simulation, during plastic deformation the material properties change over time and may be described by means of a time-dependent recurrent relationship. In other words, the current state of the structure is history dependent. The application of an individual feed-forward NNs per time step is memory intensive and would fail to describe the history dependent relationship of the parameter-response mapping between each of the time increments. The history dependency of the data set may therefore be further employed by a NN architecture that enforces recurrent relationships in the parameter-response mapping.

Let be given the QoI $y(q,t)$ at the time moment $t := \tau_\ell$, see Eq. (2.16), discretized by $[\boldsymbol{y}_\ell]_i := \boldsymbol{y}(\boldsymbol{q}_i, t) \in \mathbb{R}^{n_\circ}$ for i-th data sample, see Eq. (2.18) in Chapter 2. The approximation of the QoI by different versions of the feed-forward NN as described

in the previous sections is often not optimal due to the already mentioned reasons. Therefore, one may approximate $\boldsymbol{y}_\ell(\boldsymbol{q}),\ \ell = 1, ..., n_\mathrm{t}$ by an RNN [67]:

$$\boldsymbol{h}_\ell = \boldsymbol{\sigma}_\mathrm{h}(\boldsymbol{h}_{\ell-1}, \boldsymbol{q}, \mathbf{w}), \tag{3.29}$$

in which $\boldsymbol{\sigma}_\mathrm{h}$ acts element-wise on its argument. The accompanying approximation function is then given by:

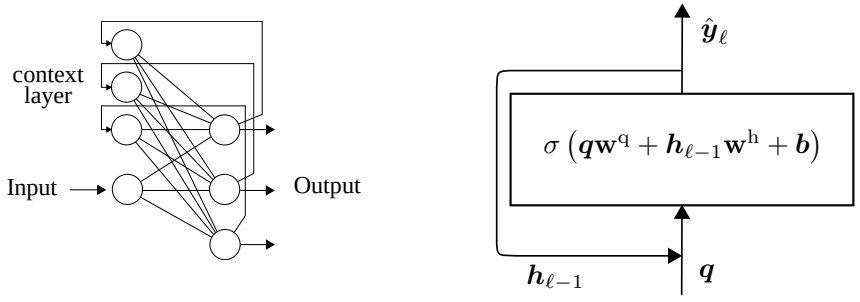
$$\hat{\boldsymbol{y}}_\ell = \boldsymbol{\sigma}_\mathrm{y}(\boldsymbol{h}_\ell, \mathbf{w}_\mathrm{y}). \tag{3.30}$$

Here, $\mathbf{w}_\mathrm{y}$ are the parameters (weights) of the output layer and $\boldsymbol{h}_\ell$ is the hidden state consisting of n_m states at time step ℓ. The initial hidden state $\boldsymbol{h}_{\ell=0}$ is unknown and thus can be either learned as a parameter or initialized to some fixed known vector. In this work $\boldsymbol{h}_{\ell=0}$ is considered to be a zero valued initial state. Moreover, the parameters $\mathbf{w}$ and $\mathbf{w}_\mathrm{y}$ are intra-dependent as further explained in the text.

Following previous statements, the generic approximation of Eq. (2.16) by means of an RNN can be formulated as a composition of functions:

$$\hat{\boldsymbol{y}}_\ell = \boldsymbol{\sigma}_\mathrm{y} \circ \boldsymbol{\sigma}_\mathrm{h}^\ell \circ ... \circ \boldsymbol{\sigma}_\mathrm{h}^1, \tag{3.31}$$

for a single recurrent hidden layer, also known as Elman network, see Fig. 3.3a [63]. The depicted Elman network is an elaboration of a feed-forward NN in which the predictive output is re-entered as input parameters for the subsequent state, hereby ensuring the recurrent relationship over the time sequence. The recurrent state therefore equals the dimension of the predictive output (i.e. the QoI).



(a) Schematic single layer overview.

(b) Single layer representation in terms of matrix multiplications.

Figure 3.3: Elman RNN, comprised of a single layer.

For simplification reasons, one may further rewrite the transition function in Eq. (3.29) in a matrix-vector form:

$$\boldsymbol{h}_\ell := \boldsymbol{\sigma}_\mathrm{h}(\boldsymbol{\Phi}_\ell \mathbf{w}) \tag{3.32}$$

with $\boldsymbol{\Phi}_\ell := [1\ \boldsymbol{q}\ \boldsymbol{h}_{\ell-1}]$ and $\mathbf{w} \in \mathbb{R}^{(n_\mathrm{f}+n_\mathrm{m}+1)\times n_\mathrm{m}}$. Here, the weights $\mathbf{w}$ are decoupled to the part that corresponds to the 1 in $\boldsymbol{\Phi}_\ell$, also called the bias, the part $\mathbf{w}^\mathrm{q}$ that corresponds to the parameter set $\boldsymbol{q}$ and the part $\mathbf{w}^\mathrm{h}$ that corresponds to the hidden state $\boldsymbol{h}_{\ell-1}$. The bias may be described by $\boldsymbol{w}_\mathrm{b} := \boldsymbol{w}_0 \in \mathbb{R}^{n_\mathrm{m}}$ and the subset of $\mathbf{w}$ corresponding to the input $\boldsymbol{q}$ is given by $\mathbf{w}^\mathrm{q} := [\boldsymbol{w}_j]_{j=1}^{n_\mathrm{f}} \in \mathbb{R}^{n_\mathrm{f}\times n_\mathrm{m}}$. Accordingly,

one may define the recurrent weight $\mathbf{w}^{\mathrm{h}}$ as a subset of $\mathbf{w}$ ($\mathbf{w}^{\mathrm{h}} \subset \mathbf{w}$) by defining $\mathbf{w}^{\mathrm{h}} := [\boldsymbol{w}_j]_{j=n_{\mathrm{f}}+1}^{n_{\mathrm{f}}+n_{\mathrm{m}}+1} \in \mathbb{R}^{n_{\mathrm{m}} \times n_{\mathrm{m}}}$. A graphical representation of the Elman network [63] in terms of the described individual weight subsets ($\mathbf{w}^{\mathrm{q}}$, $\mathbf{w}^{\mathrm{h}}$ and $\boldsymbol{b}$) is given in Fig. 3.3b. To map the hidden state to the QoI one may further employ Eq. (3.30) to obtain:

$$\hat{\boldsymbol{y}}_\ell := \boldsymbol{\sigma}_{\mathrm{y}}(\boldsymbol{\Phi}_{\mathrm{y},\ell}\mathbf{w}_{\mathrm{y}}), \tag{3.33}$$

with $\boldsymbol{\Phi}_{\mathrm{y},\ell} := [1\ \boldsymbol{h}_\ell]$ and $\mathbf{w}_{\mathrm{y}}$ being the weights of the final layer.

Estimation of the weights in Eq. (3.32) is usually done in a classical mean squared sense by minimizing the loss function as given in Eq. (3.22), and already described in Section 3.2. The structure of Eq. (3.31) provides the basis for many complex NN architectures. Optimization of these vanilla RNNs is performed over a gradient-based backward propagation through time [164, 165].

3.3.1 Recurrent neural network from differential equations

Besides the introduction of a recurrent neural framework from a data science point of view, one may also derive the recurrent architecture from a dynamics point of view. In other words, the map between the input parameter and QoI in time can be modeled by a continuous dynamical model with delay:

$$\dot{\boldsymbol{c}}(\tau) = f\left(\boldsymbol{c}(\tau), \boldsymbol{c}_\tau, \boldsymbol{q}, \tau\right) \tag{3.34}$$

in which $\boldsymbol{c} \in \mathbb{R}^{n_{\mathrm{m}}}$ represents the state of the system and $\boldsymbol{q} \in \mathbb{R}^{n_{\mathrm{f}}}$ represents the input parameter set. Furthermore,

$$\boldsymbol{c}_\tau = \{\boldsymbol{c}(\overleftarrow{\tau}) : \overleftarrow{\tau} \leq \tau\}$$

represents the trajectory of the solution in the past. The dynamic system in Eq. (3.34) can be written in terms of the discrete delay:

$$\dot{\boldsymbol{c}}(\tau) = f\left(\boldsymbol{c}(\tau), \boldsymbol{c}(\tau - \overleftarrow{\tau}_1), ..., \boldsymbol{c}(\tau - \overleftarrow{\tau}_{n_{\mathrm{t}}}), \boldsymbol{q}, \tau\right) \tag{3.35}$$

in which $\overleftarrow{\tau}_1 > \cdots > \overleftarrow{\tau}_{n_{\mathrm{t}}} \geq 0$ denotes the memory of the system given the delay $\overleftarrow{\tau}$ over n_{t} time steps.

In a special case when the function $f(\cdot)$ in Eq. (3.35) can be described in a linear sense and having only one step delay [67] the system in Eq. (3.35) can be decoupled to

$$\dot{\boldsymbol{c}}(\tau) = \mathbf{w}_{\mathrm{A}}\boldsymbol{c}(\tau) + \mathbf{w}_{\mathrm{B}}\boldsymbol{h}(\tau - \overleftarrow{\tau}_0) + \mathbf{w}_{\mathrm{C}}\boldsymbol{q} + \boldsymbol{w}_{\mathrm{D}}, \quad \boldsymbol{h} := \boldsymbol{g}(\boldsymbol{c}(\tau - \overleftarrow{\tau}_0)) \tag{3.36}$$

in which $\boldsymbol{g}(\cdot)$ is a nonlinear, saturating, and invertible function of a state. Additionally, the pair $\mathbf{w}_{\mathrm{A}} \in \mathbb{R}^{n_{\mathrm{m}} \times n_{\mathrm{m}}}, \mathbf{w}_{\mathrm{B}} \in \mathbb{R}^{n_{\mathrm{m}} \times n_{\mathrm{m}}}$ are the system matrices, $\mathbf{w}_{\mathrm{C}} \in \mathbb{R}^{n_{\mathrm{f}} \times n_{\mathrm{m}}}$ is the input matrix and $\boldsymbol{w}_{\mathrm{D}} \in \mathbb{R}^{n_{\mathrm{m}}}$ is the bias vector. Taking $\overleftarrow{\tau}_0 = \Delta\tau$, and after discretization of Eq. (3.36) by means of the implicit backward Euler technique [67, 121, 122, 166], one obtains:

$$\boldsymbol{c}_\ell = \mathbf{w}^{\mathrm{c}}\boldsymbol{c}_{\ell-1} + \mathbf{w}^{\mathrm{h}}\boldsymbol{h}_{\ell-1} + \mathbf{w}^{\mathrm{q}}\boldsymbol{q} + \boldsymbol{w}_{\mathrm{b}}, \quad \boldsymbol{h}_\ell := \boldsymbol{g}(\boldsymbol{c}_\ell) \tag{3.37}$$

with

$$\begin{aligned}\mathbf{w}^{\mathrm{c}} &:= (I - \Delta\tau \mathbf{w}_{\mathrm{A}})^{-1},\\ \mathbf{w}^{\mathrm{h}} &:= \Delta\tau \mathbf{w}_{\mathrm{B}} \mathbf{w}^{\mathrm{c}},\\ \mathbf{w}^{\mathrm{q}} &:= \Delta\tau \mathbf{w}_{\mathrm{C}} \mathbf{w}^{\mathrm{c}},\\ \boldsymbol{w}_{\mathrm{b}} &:= \Delta\tau \boldsymbol{w}_{\mathrm{D}} \mathbf{w}^{\mathrm{c}}.\end{aligned} \tag{3.38}$$

According to the Lyapunov stability criteria [167] the system in Eq. (3.36) is stable if every eigenvalue of $\mathbf{w}^{\mathrm{ch}} := \mathbf{w}^{\mathrm{c}} + \mathbf{w}^{\mathrm{h}}$ lies within the complex-valued unit circle. Within the bounds of the stability requirement one can freely choose $\mathbf{w}_{\mathrm{A}}$ and $\mathbf{w}_{\mathrm{B}}$. Hence, to simplify Eq. (3.37) one may define $\mathbf{w}_{\mathrm{A}}$ as a diagonal matrix with large negative values on its main diagonal (i.e. $w_{\mathrm{A},nn} \ll 0, w_{\mathrm{A},n\neq m} = 0$). One then obtains the system matrix [67]:

$$\mathbf{w}^{\mathrm{c}} = (I - \Delta\tau \mathbf{w}_{\mathrm{A}})^{-1} \approx -\mathbf{w}_{\mathrm{A}}^{-1} \approx \mathbf{0},$$

meaning that the stability condition can be defined using only the matrix $\mathbf{w}^{\mathrm{ch}} \approx \mathbf{w}^{\mathrm{h}} \approx -\mathbf{w}_{\mathrm{B}} \mathbf{w}_{\mathrm{A}}^{-1}$. Thus, the sufficient condition for stability becomes $0 < \lambda_n < 1$ in which λ_n is the n-th eigenvalue of $\mathbf{w}^{\mathrm{ch}}$.

As $\mathbf{w}^{\mathrm{c}}$ vanishes, dynamical system described in Eq. (3.37) becomes

$$\boldsymbol{c}_\ell = \mathbf{w}^{\mathrm{q}} \boldsymbol{q} + \mathbf{w}^{\mathrm{h}} \boldsymbol{h}_{\ell-1} + \boldsymbol{w}_{\mathrm{b}}, \quad \boldsymbol{h}_\ell := \boldsymbol{g}(\boldsymbol{c}_\ell), \tag{3.39}$$

which can therefore also be written as

$$\boldsymbol{h}_\ell = \boldsymbol{\sigma}_{\mathrm{h}}(\mathbf{w}^{\mathrm{q}} \boldsymbol{q} + \mathbf{w}^{\mathrm{h}} \boldsymbol{h}_{\ell-1} + \boldsymbol{w}_{\mathrm{b}}), \tag{3.40}$$

with the observable output defined as:

$$\boldsymbol{y}_\ell := \boldsymbol{\sigma}_{\mathrm{y}}(\mathbf{w}_{\mathrm{y}} \boldsymbol{h}_\ell + \boldsymbol{w}_{\mathrm{b,y}}), \tag{3.41}$$

in which σ_{y} is possibly a nonlinear observation operator. When collecting $\mathbf{w} := \{\mathbf{w}^{\mathrm{h}}, \mathbf{w}^{\mathrm{q}}, \boldsymbol{w}_{\mathrm{b}}, \mathbf{w}_{\mathrm{y}}, \boldsymbol{w}_{\mathrm{b,y}}\}$ from Eq. (3.40) and Eq. (3.41) and applying recurrency in the time interval $[0, \Delta\tau, ..., \ell\Delta\tau]$, one can further introduce an RNN as a composition of ℓ functions:

$$\boldsymbol{F}_\ell(\boldsymbol{h}, \boldsymbol{q}, \mathbf{w}) := (\boldsymbol{g}_\ell \circ \boldsymbol{g}_{\ell-1} \circ \boldsymbol{g}_{\ell-2} \circ ... \circ \boldsymbol{g}_1)(\boldsymbol{h}, \boldsymbol{q}, \mathbf{w})$$

in which

$$\boldsymbol{g}_n(\boldsymbol{h}, \boldsymbol{q}, \mathbf{w}) := \boldsymbol{\sigma}_{\mathrm{h}}(\mathbf{w}^{\mathrm{h}} \boldsymbol{h}_{n-1} + \mathbf{w}^{\mathrm{q}} \boldsymbol{q}_n + \boldsymbol{w}_{\mathrm{b}}), \quad n = 1, ..., \ell - 1$$

and

$$\boldsymbol{g}_\ell(\boldsymbol{h}, \boldsymbol{q}, \mathbf{w}) := \boldsymbol{\sigma}_{\mathrm{y}}(\mathbf{w}_{\mathrm{y}} \boldsymbol{h}_\ell + \boldsymbol{w}_{\mathrm{b,y}})$$

such that

$$\boldsymbol{y}_\ell = \boldsymbol{F}_\ell(\boldsymbol{h}, \boldsymbol{q}, \mathbf{w})$$

holds.

Given the real dynamical system that is represented by a possibly noisy data set $(\boldsymbol{q}^{(i)}, \boldsymbol{y}_\ell^{(i)})_{i=1,...,n_{\mathrm{b}}}$ the goal is to estimate $\mathbf{w}$ using the minimization described in Eq. (3.22).

As one may notice in Eq. (3.32) and Eq. (3.33) the state variables $\boldsymbol{h}_\ell$ are propagated through time, although the weights and activation functions are not dependent on time. One major drawback of the RNN training approach is that the error exponentially decays/grows with time (i.e. the vanishing/exploding gradient problem), and hence the weights cannot be updated anymore due to its updated increment going to zero or infinity, see [67]. The main reason for an exponentially decaying/growing error gradient in the RNN training approach is that the gradient directly depends on both the norm of the hidden state matrix $\mathbf{w}^{\mathrm{h}}$ as well as on the norm of the derivative of the activation function $\boldsymbol{g}'(\cdot)$. If all eigenvalues of the hidden state matrix are smaller than one, i.e. $0 < \lambda_j^{\mathbf{w}^{\mathrm{h}}} < 1$, then $\|\mathbf{w}^{\mathrm{h}}\| < 1$. In such a case the gradient is vanishing if $\|\boldsymbol{g}'(\cdot)\| < 1$. If any $\lambda_j^{\mathbf{w}^{\mathrm{h}}} > 1$ then the term $\|\mathbf{w}^{\mathrm{h}}\|$ will exponentially grow. In such a case when $\|\boldsymbol{g}'(\cdot)\| = \mathbf{0}$ (the flat regions of the activation function), then the gradient vanishes, otherwise if $\|\boldsymbol{g}'(\cdot)\| \neq \mathbf{0}$ (quasi-linear regions of the activation function) the gradient explodes. Thus, the RNNs suffer from the so-called gradient problem when used in a long term integration. Additionally, one must prevent the propagation of conflicting information through the networks layers over time. Examples of conflicting information are [65, 168]:

- Input weight conflict: As presented in Fig. 3.3b the input weight matrix $\mathbf{w}^{\mathrm{q}}$ determines for every time step ℓ what information of the input parameter set $\boldsymbol{q}$ is propagated or ignored. However, with $\mathbf{w}^{\mathrm{q}}$ being time-independent this may quickly lead to conflicting gradients, denoting the desire to store certain inputs at specific time steps, while ignoring others, and thus describing gradients that are pointing away from each other. This makes learning difficult, asking for control of the writing operations. As a consequence, the gradient with the largest magnitude will dominate the other, and therefore prioritize certain information.
- Output weight conflict: The hidden state $\boldsymbol{h}_\ell$ of a plain RNN (Fig. 3.3b) arises from a weighted connection that originates at the input parameter set. This hidden state is used to both store information for the next time step and propagate information to the output (layer). As a result, the hidden state may be subject to propagating a mixture of information for both the output layer and the next time step. Hence, in order to prevent any disruptions caused by this hidden layer $\boldsymbol{h}_\ell$, while simultaneously propagating the information given by the hidden state, the output layer weight may be subject to conflicting update gradients [168].

With strong time-varying data and with an eye on the vanishing gradient problem, a more advanced recurrent architecture that overcomes these gradient issues is favorable. Among others, one can distinguish two well known and established RNNs that are condensed of multiple weight matrices per layer and address a vanishing gradient. These networks are the long short-term memory (LSTM)-based network and the Gated Recurrent Unit (GRU) [64–66].

3.4 Long short-term memory

To make the system robust against the aforementioned RNN training instabilities and information conflicts, one may first generalize the hidden state from $\boldsymbol{h}_\ell = \boldsymbol{g}(\boldsymbol{c}_\ell)$ to:

$$\boldsymbol{h}_\ell = \boldsymbol{o}_\ell(\ell) \odot \boldsymbol{g}(\boldsymbol{c}_\ell) \tag{3.42}$$

in which $\boldsymbol{o}_\ell(\ell)$ is a continuous, differentiable, monotonically increasing function that maps the domain $(-\infty, \infty)$ to $(0, 1)$, i.e. $0 \leq \boldsymbol{o}_\ell(\ell) \leq 1$ (or $[-1, 1]$) [67, 169]. Hence, the introduced function $\boldsymbol{o}_\ell(\ell)$ protects the memory of output connections (i.e. $\boldsymbol{h}_\ell$) from irrelevant input values/variables stored in hidden layers, which thus solves the output weight conflict. Furthermore, Eq. (3.37) can be generalized to

$$\boldsymbol{c}_\ell = \boldsymbol{f}_\ell(\ell) \odot (\mathbf{w}^{\mathrm{c}} \boldsymbol{c}_{\ell-1}) + \boldsymbol{z}_\ell(\ell) \odot \boldsymbol{\sigma}_{\tilde{\mathrm{c}}}(\boldsymbol{s}_\ell)$$

in which

$$\boldsymbol{s}_\ell := \mathbf{w}^{\mathrm{h}} \boldsymbol{h}_{\ell-1} + \boldsymbol{g}_{\mathrm{q}}(\ell) \odot \mathbf{w}^{\mathrm{q}} \boldsymbol{q} + \boldsymbol{w}_{\mathrm{b}}.$$

Here, all controls are continuous, differentiable, monotonically increasing functions that map the domain $(-\infty, \infty)$ into the range $(0, 1)$, i.e. $\mathbf{0} \leq \boldsymbol{f}_\ell(\ell), \boldsymbol{g}_{\mathrm{q}}(\ell), \boldsymbol{z}_\ell(\ell) \leq \mathbf{1}$ or into the range $[-1, 1]$. Taking $\boldsymbol{g}_{\mathrm{q}}(\ell) = 1$ and $\mathbf{w}^{\mathrm{c}} = \mathbf{I}$ one obtains

$$\boldsymbol{c}_\ell = \boldsymbol{f}_\ell(\ell) \odot \boldsymbol{c}_{\ell-1} + \boldsymbol{z}_\ell(\ell) \odot \boldsymbol{\sigma}_{\tilde{\mathrm{c}}}(\boldsymbol{s}_\ell)$$

which is a core constituent of the set of formulas defining the cell of the vanilla LSTM network [168]. The applied function $\boldsymbol{z}_\ell(\ell) \in (0, 1)$ protects the memory $\boldsymbol{c}_\ell$ from irrelevant inputs arising from $\boldsymbol{s}_\ell$, hereby solving the input weight conflict.

Thus, the LSTM cell can be further written as

$$\boldsymbol{c}_\ell = \boldsymbol{f}_\ell(\ell) \odot \boldsymbol{c}_{\ell-1} + \boldsymbol{z}_\ell(\ell) \odot \boldsymbol{\sigma}_{\tilde{\mathrm{c}}}(\boldsymbol{s}_\ell), \tag{3.43}$$

with

$$\begin{aligned}
\tilde{\mathbf{c}}_\ell = \boldsymbol{\sigma}_{\tilde{\mathrm{c}}}(\boldsymbol{s}_\ell), \quad & \boldsymbol{s}_\ell := \mathbf{w}^{\mathrm{h}} \boldsymbol{h}_{\ell-1} + \mathbf{w}^{\mathrm{q}} \boldsymbol{q} + \boldsymbol{w}_{\mathrm{b}} \\
\boldsymbol{f}_\ell(\ell) = \boldsymbol{\sigma}_{\mathrm{f}}(\boldsymbol{s}_{\mathrm{f},\ell}), \quad & \boldsymbol{s}_{\mathrm{f},\ell} := \hat{\mathbf{w}}^{\mathrm{c}} \boldsymbol{c}_{\ell-1} + \hat{\mathbf{w}}^{\mathrm{q}} \boldsymbol{q} + \hat{\mathbf{w}}^{\mathrm{h}} \boldsymbol{h}_{\ell-1} + \boldsymbol{w}_{\mathrm{b,f}} \\
\boldsymbol{o}_\ell(\ell) = \boldsymbol{\sigma}_{\mathrm{o}}(\boldsymbol{s}_{\mathrm{o},\ell}), \quad & \boldsymbol{s}_{\mathrm{o},\ell} := \tilde{\mathbf{w}}^{\mathrm{c}} \boldsymbol{c}_\ell + \tilde{\mathbf{w}}^{\mathrm{q}} \boldsymbol{q} + \tilde{\mathbf{w}}^{\mathrm{h}} \boldsymbol{h}_{\ell-1} + \boldsymbol{w}_{\mathrm{b,o}} \\
\boldsymbol{z}_\ell(\ell) = \boldsymbol{\sigma}_{\mathrm{z}}(\boldsymbol{s}_{\mathrm{z},\ell}), \quad & \boldsymbol{s}_{\mathrm{z},\ell} := \bar{\mathbf{w}}^{\mathrm{c}} \boldsymbol{c}_{\ell-1} + \bar{\mathbf{w}}^{\mathrm{q}} \boldsymbol{q} + \bar{\mathbf{w}}^{\mathrm{h}} \boldsymbol{h}_{\ell-1} + \boldsymbol{w}_{\mathrm{b,z}}
\end{aligned}$$

and the output (observable) defined as

$$\boldsymbol{y}_\ell = \boldsymbol{\sigma}_{\mathrm{y}}(\boldsymbol{s}_{\mathrm{y},\ell}), \quad \boldsymbol{s}_{\mathrm{y},\ell} := \mathbf{w}_{\mathrm{y}} \boldsymbol{h}_\ell + \boldsymbol{w}_{\mathrm{b,y}}.$$

For simplification of the further notation, the target states and output may be denoted by $\boldsymbol{s}_{\mathrm{g},\ell}$ with the index $\mathrm{g} \in \{\mathrm{f}, \mathrm{z}, \tilde{\mathrm{c}}, \mathrm{o}, \mathrm{y}\}$[2]. Similarly, the notation $\boldsymbol{\sigma}_{\mathrm{g},\ell}$ is adopted with the index $\mathrm{g} \in \{\mathrm{f}, \mathrm{z}, \tilde{\mathrm{c}}, \mathrm{o}, \mathrm{y}\}$ for different activation functions that act element-wise on the input argument. Generally, a sigmoid function (see Fig. 3.1a) is applied for $\boldsymbol{\sigma}_{\mathrm{o}}$,

[2] In the notation there is one exemption with $\boldsymbol{s}_{\tilde{\mathrm{c}},\ell} := \boldsymbol{s}_\ell$

$\boldsymbol{\sigma}_{\mathrm{z}}$ and $\boldsymbol{\sigma}_{\mathrm{f}}$, and for the candidate gate the hyperbolic tangent function (see Fig. 3.1b) is applied.

In peephole LSTM $\mathbf{w}^{\mathrm{h}}$, $\hat{\mathbf{w}}^{\mathrm{h}}$, $\tilde{\mathbf{w}}^{\mathrm{h}}$ and $\bar{\mathbf{w}}^{\mathrm{h}}$ are all identical to zero, whereas in a classical LSTM cell with the forget gate –which is considered in this work– $\mathbf{w}^{\mathrm{c}}, \hat{\mathbf{w}}^{\mathrm{c}}, \tilde{\mathbf{w}}^{\mathrm{c}}, \bar{\mathbf{w}}^{\mathrm{c}}$ are all equal to zero. Here, $\boldsymbol{\sigma}_{\mathrm{g}}$ is the activation function that acts element-wise on the input argument, $\boldsymbol{q}$ is the input state and $\boldsymbol{h}_\ell$ is the hidden state at time ℓ. One may also distinguish the forget gate's activation vector $\boldsymbol{f}_\ell$, the input gate's activation vector $\boldsymbol{z}_\ell$, the output gate's activation vector $\boldsymbol{o}_\ell$, the cell candidate activation vector $\tilde{\boldsymbol{c}}_\ell := \boldsymbol{\sigma}_{\mathrm{c}}(\boldsymbol{s}_\ell)$, and the cell state $\boldsymbol{c}_\ell$ [66].

To simplify the notation, let $\boldsymbol{\Phi}_\ell = [1\ \boldsymbol{q}\ \boldsymbol{h}_{\ell-1}] \in \mathbb{R}^{1\times(n_{\mathrm{f}}+n_{\mathrm{m}}+1)}$ be "the basis function", i.e. the full state of the cell, and $\mathbf{w}_{\mathrm{g}}$ –with $\mathbf{w}_{\mathrm{g}} \in \mathbb{R}^{(n_{\mathrm{f}}+n_{\mathrm{m}}+1)\times n_{\mathrm{m}}}$ expect for $g = \mathrm{y}$ when $\boldsymbol{\Phi}_{\mathrm{y},\ell} = [1\ \boldsymbol{h}_\ell] \in \mathbb{R}^{1\times(n_{\mathrm{m}}+1)}$ and $\mathbf{w}_{\mathrm{y}} \in \mathbb{R}^{(n_{\mathrm{m}}+1)\times n_{\mathrm{o}}}$– be collection of the corresponding weights for the specific gate $\mathrm{g} \in \{\mathrm{f}, \mathrm{z}, \tilde{\mathrm{c}}, \mathrm{o}\}$ e.g. $\mathbf{w}_{\mathrm{f}} := [\hat{\mathbf{w}}^{\mathrm{c}}, \hat{\mathbf{w}}^{\mathrm{u}}, \hat{\mathbf{w}}^{\mathrm{h}}, \boldsymbol{w}_{\mathrm{b,f}}]$[3]. The LSTM cell state (schematically represented in Fig. 3.4) can then be also written as [65]:

$$
\begin{aligned}
\boldsymbol{f}_\ell &= \boldsymbol{\sigma}_{\mathrm{f}}(\boldsymbol{\Phi}_\ell \mathbf{w}_{\mathrm{f}}) := \boldsymbol{\sigma}_{\mathrm{f}}(\boldsymbol{s}_{\mathrm{f},\ell}),\\
\boldsymbol{z}_\ell &= \boldsymbol{\sigma}_{\mathrm{z}}(\boldsymbol{\Phi}_\ell \mathbf{w}_{\mathrm{z}}) := \boldsymbol{\sigma}_{\mathrm{z}}(\boldsymbol{s}_{\mathrm{z},\ell}),\\
\tilde{\boldsymbol{c}}_\ell &= \boldsymbol{\sigma}_{\tilde{\mathrm{c}}}(\boldsymbol{\Phi}_\ell \mathbf{w}_{\tilde{\mathrm{c}}}) := \boldsymbol{\sigma}_{\tilde{\mathrm{c}}}(\boldsymbol{s}_{\tilde{\mathrm{c}},\ell}),\\
\boldsymbol{o}_\ell &= \boldsymbol{\sigma}_{\mathrm{o}}(\boldsymbol{\Phi}_\ell \mathbf{w}_{\mathrm{o}}) := \boldsymbol{\sigma}_{\mathrm{o}}(\boldsymbol{s}_{\mathrm{o},\ell}),\\
\boldsymbol{c}_\ell &= \boldsymbol{f}_\ell \odot \boldsymbol{c}_{\ell-1} + \boldsymbol{z}_\ell \odot \tilde{\boldsymbol{c}}_\ell,\\
\boldsymbol{h}_\ell &= \boldsymbol{o}_\ell \odot \boldsymbol{\sigma}_{\mathrm{c}}(\boldsymbol{c}_\ell).
\end{aligned}
\tag{3.44}
$$

The output layer is accordingly defined by:

$$
\hat{\boldsymbol{y}}_\ell = \boldsymbol{\sigma}_{\mathrm{y}}\left(\boldsymbol{\Phi}_{\mathrm{y},\ell}\mathbf{w}_{\mathrm{y}}\right) \tag{3.45}
$$

in which $\hat{\boldsymbol{y}}_\ell \in \mathbb{R}^{n_{\mathrm{o}}}$. A graphical representation of the NN comprising of an LSTM cell and a subsequent output layer is given on the left side in Fig. 3.4. The network architecture consisting of an LSTM cell and subsequent output layer is the architecture that is further applied throughout this work. Thus, if it is not stated differently then under architecture one means the LSTM one.

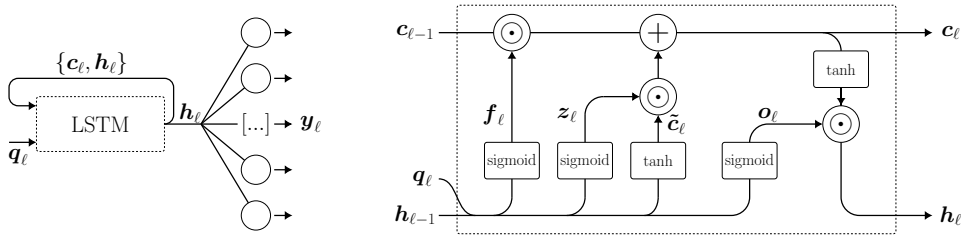


Figure 3.4: LSTM-based NN (left) and the LSTM cell (right), comprising the continuous cell state $\boldsymbol{c}_\ell$ and hidden state $\boldsymbol{h}_\ell$ for time step ℓ.

[3] In the notation there is one exemption with $\mathbf{w}_{\tilde{\mathrm{c}},\ell} := [\mathbf{w}^{\mathrm{h}}, \mathbf{w}^{\mathrm{u}}, \boldsymbol{w}_{\mathrm{b}}]$

3.4.1 Optimization by backward propagation

Similarly as for general feed-forward networks, the LSTM weights are optimized by a backward propagation procedure. By using the same definition of the mean-squared function as in Eq. (3.22) one may propagate the error backward through the network by finding the corresponding partial derivatives:

$$\begin{aligned}
\frac{\partial \mathcal{J}_{\ell:n_\mathrm{t}}}{\partial \boldsymbol{h}_\ell} &:= \frac{\partial \mathcal{J}_\ell}{\partial \boldsymbol{h}_\ell} + \frac{\partial \mathcal{J}_{(\ell+1):n_\mathrm{t}}}{\partial \boldsymbol{h}_\ell} \\
\frac{\partial \mathcal{J}_{\ell:n_\mathrm{t}}}{\partial \boldsymbol{c}_\ell} &:= \partial \boldsymbol{h}_\ell \odot \boldsymbol{o}_\ell \odot \left(1 - \tanh^2(\boldsymbol{c}_\ell)\right) + \partial \boldsymbol{c}_{\ell+1} \odot \boldsymbol{f}_{\ell+1} \\
\frac{\partial \mathcal{J}_{\ell:n_\mathrm{t}}}{\partial \boldsymbol{s}_{\tilde{\mathrm{c}},\ell}} &:= \partial \boldsymbol{c}_\ell \odot \boldsymbol{z}_\ell \odot (1 - \tilde{\boldsymbol{c}}_\ell^2) \\
\frac{\partial \mathcal{J}_{\ell:n_\mathrm{t}}}{\partial \boldsymbol{s}_{\mathrm{z},\ell}} &:= \partial \boldsymbol{c}_\ell \odot \tilde{\boldsymbol{c}}_\ell \odot \boldsymbol{z}_\ell \odot (1 - \boldsymbol{z}_\ell) \\
\frac{\partial \mathcal{J}_{\ell:n_\mathrm{t}}}{\partial \boldsymbol{s}_{\mathrm{f},\ell}} &:= \partial \boldsymbol{c}_\ell \odot \boldsymbol{c}_{\ell-1} \odot \boldsymbol{f}_\ell \odot (1 - \boldsymbol{f}_\ell) \\
\frac{\partial \mathcal{J}_{\ell:n_\mathrm{t}}}{\partial \boldsymbol{s}_{\mathrm{o},\ell}} &:= \partial \boldsymbol{h}_\ell \odot \tanh(\boldsymbol{c}_\ell) \odot \boldsymbol{o}_\ell \odot (1 - \boldsymbol{o}_\ell)
\end{aligned} \tag{3.46}$$

in which $\mathcal{J}_{\ell:n_\mathrm{t}}$ denotes the mean squared error at time step ℓ up to time step $\ell = n_\mathrm{t}$ with $\mathcal{J}_\ell$ being the mean squared error as defined by:

$$\mathcal{J}_\ell(\mathbf{w}) = \frac{1}{n_\mathrm{b}} \sum_{i=1}^{n_\mathrm{b}} \|[\mathbf{y}_\ell]_i - [\hat{\mathbf{y}}_\ell]_i\|_2^2, \tag{3.47}$$

with $[\hat{\mathbf{y}}_\ell]_i := \hat{\boldsymbol{y}}(\boldsymbol{q}_i, t) \in \mathbb{R}^{n_\mathrm{o}}$ being the prediction of the NN for the i-th data sample at time moment $t = \tau_\ell$. The iterative weight update for each of the four gate weights can be determined using Eq. (3.24).

3.4.2 Quality measures

As aforementioned, the generally used optimization metric for deterministic networks is the mean squared error. Using this metric one obtains an estimate of the variance of all residuals. A direct drawback is the dependency on the data magnitude. Therefore, a closely related normalized form of the mean squared, known as the coefficient of determination, is of better use in any quality assessment. For the complete sample set ($n_\mathrm{b} = n_\mathrm{d}$) one defines the coefficient of determination R^2 as:

$$\mathrm{R}^2 = 1 - \sum_{i=1}^{n_\mathrm{d}} \sum_{k=1}^{n_\mathrm{o}} \sum_{\ell=1}^{n_\mathrm{t}} \frac{\|[\mathbf{y}_\ell]_{ik} - [\hat{\mathbf{y}}_\ell]_{ik}\|_2^2}{\|[\mathbf{y}_\ell]_{ik} - \overline{y}\|_2^2} \tag{3.48}$$

in which

$$\overline{y} := \frac{1}{n_\mathrm{d} n_\mathrm{o} n_\mathrm{t}} \sum_{i=1}^{n_\mathrm{d}} \sum_{k=1}^{n_\mathrm{o}} \sum_{\ell=1}^{n_\mathrm{t}} [\mathbf{y}_\ell]_{ik}$$

is the overall mean of the observed data. A model that exactly matches the presented data (i.e. $\|[\mathbf{y}_\ell]_{ik} - [\hat{\mathbf{y}}_\ell]_{ik}\|_2^2 = 0$) yields a maximum of $\mathrm{R}^2 = 1.0$. One may also observe that for $[\hat{\mathbf{y}}_\ell]_{ik} = \overline{y}$ one obtains $\mathrm{R}^2 = 0.0$.

3.5 Structural applications

In this section a traditional point-estimate deterministic LSTM-based network is applied to describe the time series prediction of two academic examples:

- a geometrically parametrized tensile specimen featuring a hardness transition zone [1, 108]. The NN is used to describe the fracture indicator parameter over time. The fracture indicator shows a strong bifurcation, which the LSTM should be capable to depict with sufficient generalization within the given sample space Ω.
- a metal strip specimen subjected to a parametrized punch impact position. The time series prediction is merely used to describe the deformed mesh in order to identify the LSTM generalization dependencies and capabilities.

The goal is to identify the strengths and weaknesses regarding point estimate LSTM-based NNs for structural applications.

3.5.1 Tapered tensile specimen

A tapered tensile specimen is considered to represent an extreme simplification of a B-pillar, including a hardness transition as used in the automotive industry [108]. The specimen is geometrically parametrized by a width variable φ defined at the narrow side of the gauge section, see Fig. 3.5, and is mechanically described by the material model presented in [1, 108] that is experimentally validated. Thus, the considered mechanical behavior is driven by a hardness transition in the gauge section. In

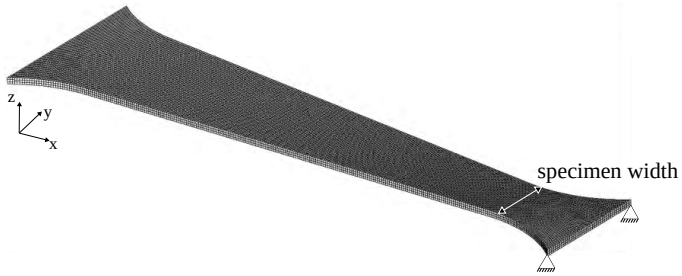


Figure 3.5: Tapered tensile specimen finite element model with section width φ.

Fig. 3.6 the hardness transition in the gauge section is visualized as a function of the longitudinal coordinate. The specimens used for the experiments are extracted from a 1.5 mm thick tailor tempered hot-formed section ensuring a hardness transition in the gauge section of the tapered tensile specimen. In [1, 108] a comparison is made between the experimentally obtained fracture results and a simulation model composed of hexagonal solid elements. The explicit Virtual Performance Solution crash solver [126] is used for the simulations and a classical reduced integration is applied for the hexagonal solid elements [108]. In addition, a stiffness method using hourglass shape vectors (Flanagan–Belytschko) is used [170, 171]. The tensile specimen is constrained in longitudinal direction on the narrow (i.e. right) side edge, as shown in Fig. 3.6. By means of a constant (tensile) velocity of $\dot{u} = 0.01\,\mathrm{mm\,s^{-1}}$

applied to the constraint edge on the wide (i.e. left) side of the specimen a quasi-static strain rate in the localization area is ensured. On both edges the constraint is applied uniformly over the full edge length.

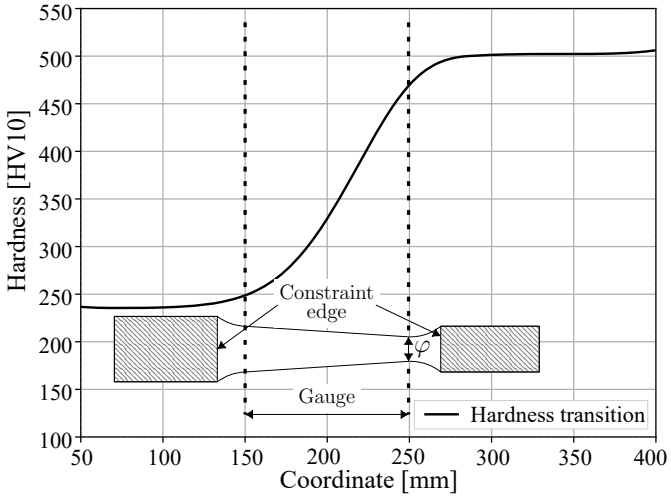


Figure 3.6: Hardness transition in the gauge section of the tapered tensile specimen, adapted from [1].

A width of $\varphi = 28.0\,\text{mm}$ reduces the tapered tensile specimen to a standard uni-axial tensile specimen with parallel edges in the gauge section. For a smaller width φ, the location of fracture is found to switch to the narrow part of the gauge section.

An element length of $l = 0.5\,\text{mm}$ is taken, giving total of $n_\text{e} = 34\,809$ elements in the considered specimen. In this application, mesh topology refers to the internal connectivity of all considered nodes and elements. Implementing a constant mesh topology in the training procedure of a NN significantly simplifies the application, as the internal node and element connectivity remain unchanged. It has to be noted that this can only be done for relatively minor geometry changes when no additional geometrical features are introduced or left out [129]. At $\varphi = 16.0\,\text{mm}$ the characteristic element length increases to $l = 0.52\,\text{mm}$. The used fracture model is known to be highly mesh size dependent, especially in strain localization zones where high strain gradients are found [108]. The modified Mohr-Coulomb (mMC) fracture model as described in Section 2.2 accounts for different mesh sizes by linear interpolation between fracture surfaces, which have been calibrated to reference mesh sizes.

The resulting force-displacement curves of the tensile experiments performed in [1] are given in Fig. 3.7 for $\varphi = \{13.0\,\text{mm}, 13.5\,\text{mm}, 14.0\,\text{mm}\}$. The solid-based simulation shows great similarity to the performed experiments. The data generated by the FEM simulation is therefore assumed to be valid and hence applicable for further use.

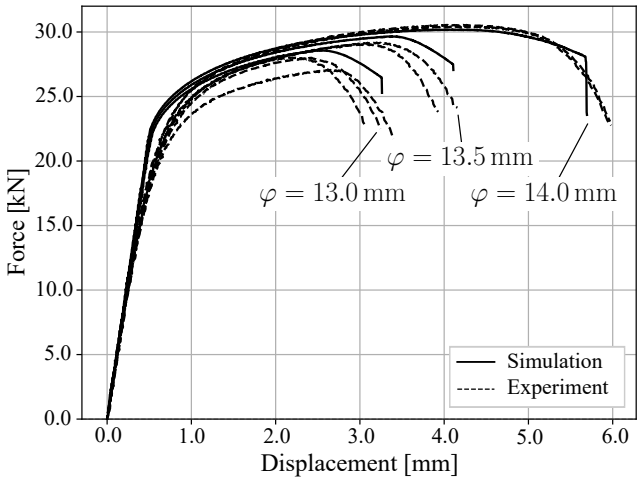


Figure 3.7: Validation of the adapted solid element-based model and experiments at widths $\varphi = \{13.0\,\text{mm}, 13.5\,\text{mm}, 14.0\,\text{mm}\}$ for VPS v2013 with the experimental results taken from [1].

Training data generation

In Table 3.1 an overview of the considered DOEs with their corresponding widths φ is presented. Four different sets of DOEs for training are considered: $n_\text{d} = 10$, $n_\text{d} = 5$, $n_\text{d} = 3$ and $n_\text{d} = 2$ width samples, sampled using the Latin Hypercube sampling (LHS) procedure for the range $\varphi \in [13.0\,\text{mm}, 16.0\,\text{mm}]$, with each sample located in the center of the created hypercubes [135]. Besides a reduced time in creating the DOEs, the advantage of a lower required number of samples is the decrease of the training data set, leading to a reduction in network training time. However, fewer samples also reduce the accuracy of the surrogate model over the parameter domain and should thus be handled with care.

Table 3.1: Training sample sets n_d in combination with their corresponding widths, selected using LHS sampling.

n_d	φ [mm]									
10	13.15	13.45	13.75	14.05	14.35	14.65	14.95	15.25	15.55	15.85
5	13.30	13.90	14.50	15.10	15.70					
3	13.50	14.50	15.50							
2	13.75	15.25								

A simulation time of 40 ms is considered, with a corresponding linearly increasing displacement adding up to 7.0 mm at $t = t_\text{end}$. Due to the strain-rate insensitive material model, the higher strain rates are neglected. For the creation of training data, a time step output frequency of $0.5\,\text{m}\,\text{s}^{-1}$ is applied, yielding a total of $n_\text{t} = 21$ time steps for all n_d load cases with $n_\text{d} \in \{10, 5, 3, 2\}$.

The QoI $\boldsymbol{y}_\ell$ are fracture indicator values which are defined in a range $d_\mathrm{f} \in [0.0, 1.0]$, as described in Eq. (2.7). Unity denotes the initiation of fracture, after which the element is eliminated, causing the specimen to start losing its structural integrity. At continued deformation this causes further progression of the rupture, eventually leading to complete fracture separating the specimen into two parts. The proposed LSTM NN requires a constant connectivity description between all considered finite elements to ensure a robust map can be described between the varying input parameter set $\boldsymbol{q}$ -here consisting of the width φ- and the response $\boldsymbol{y}_\ell$, here defined as the fracture indicator. Therefore, the deleted elements must be maintained in the resulting fracture matrix. This is done by keeping the values of the corresponding indices at unity.

The sampled input parameter values are scaled from the range $\varphi \in [13.0\,\mathrm{mm}, 16.0\,\mathrm{mm}]$ to $[0.0, 1.0]$. With a focus on sample domain dependency, a single LSTM architecture with $n_\mathrm{m} = 128$ with a linear time-distributed output layer has been applied, which corresponds to a total of 4 556 921 unknown weights. However, only 1.46% of these weights are located in the LSTM cell, as the strong majority of the weights is located in the output layer to map the LSTM width ($n_\mathrm{m} = 128$) to the number of elements ($n_\mathrm{e} = 34\,809$) in the structural component. The used mean squared error is given by Eq. (3.22). Parameter optimization is performed over 3000 epochs, with a batch size n_b equal to the number of samples $n_\mathrm{d} \in \{10, 5, 3, 2\}$. The classical Adam-optimizer is used (see Section 3.2), combined with a learning rate of $\lambda = 0.001$ [158].

A single Nvidia Quadro P5000 GPU has been used for training and application, utilizing the Tensorflow v1.8.0 framework.

Assessment of the model convergence

For all network training procedures a continuously decreasing loss –defined by the mean squared error– is observed, indicating a stable training towards a (local) minimum, see Fig. 3.8.

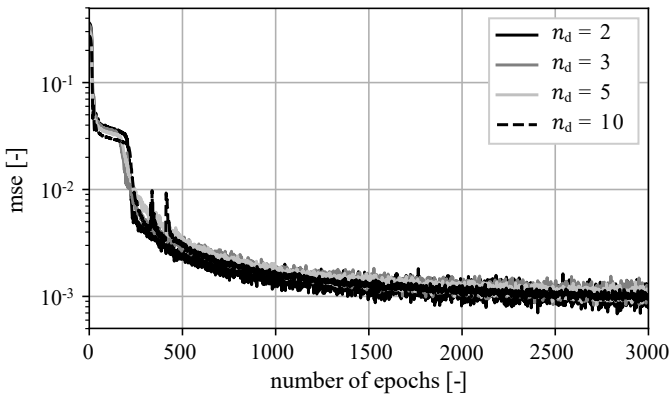
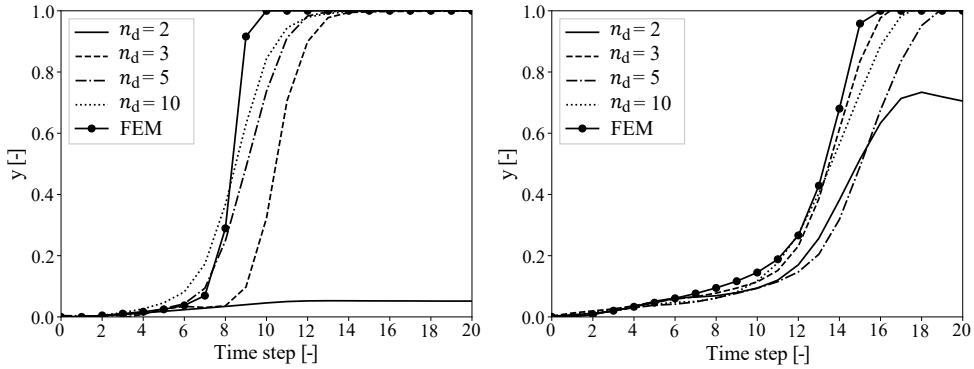


Figure 3.8: Development of the mean squared error during training over the number of epochs, convergence is reached after 1500 epochs.

Assessment of the sample space generalization

The considered tapered tensile specimen is characterized by two width dependent main deformation and fracture bifurcation modes: for $\varphi < 13.70\,\text{mm}$, deformation localization is observed at the narrow region of the gauge section. For $\varphi \geq 13.70\,\text{mm}$, deformation localization is found at the wider region of the gauge section. These observations are in accordance with these in [108].

In Fig. 3.9a and Fig. 3.9b the fracture indicator is plotted over time for the first element reaching $y = 1.0$ in the FEM simulation. Large deviations are found for the network trained with $n_\text{d} = 2$: the LHS sampling does not include the fracture location shift from the narrow to the wide side of the specimen, causing the LSTM model to predict fracture at the wide side for $\varphi = 13.0\,\text{mm}$. At $16.0\,\text{mm}$ the fracture magnitude is accurate (Fig. 3.9b), but it is predicted for the element different than the actual one. This leads to a lower predicted fracture value for this critical element, explaining the major deviation at $t = t_\text{end}$ in Fig. 3.9b. The predicted fracture for the network trained on $n_\text{d} = 3$ shown in Fig. 3.9b strongly corresponds to the fracture development of the FEM reference, and has a smaller deviation compared the fracture predicted by the network trained on $n_\text{d} = 2$, $n_\text{d} = 5$ and $n_\text{d} = 10$.



(a) Single element fracture development over time for $\varphi = 13.0\,\text{mm}$.

(b) Single element fracture development over time for $\varphi = 16.0\,\text{mm}$.

Figure 3.9: Fracture development for a single critical element over time for $\varphi = 13.0\,\text{mm}$ and $\varphi = 16.0\,\text{mm}$.

Using the largest DOE data set $n_\text{d} = 10$ a relatively low determination coefficient of $\text{R}^2_\text{train} = 0.979$ (Table 3.2) is found. This low value (compared to the networks trained on $n_\text{d} = 2$, $n_\text{d} = 3$ and $n_\text{d} = 5$) is primarily caused by various elements not catching up with a strongly increasing fracture magnitude at the through-thickness outer edges of the specimen, see Fig. 3.9a and Fig. 3.9b. The R^2_train-value is sensitive to the number of samples n_d used to trained the network. Hence, R^2_train increases with a decreasing n_d (see Table 3.2), because the same network architecture is used to describe fewer data points. The coefficient of determination must therefore also be evaluated on a validation set. An additional inter/extrapolation error comparison is thus introduced with R^2_val denoting the coefficient of determination on the validation

data set. Here, the trained LSTM model is used to predict the fracture indicator for the remaining number of data points in the training set, see Table 3.2.

Table 3.2: R^2 metrics for all considered sample domains $n_d \in \{2, 3, 5, 10\}$.

n_d		2	3	5	10
R^2_{train}		0.991	0.987	0.986	0.979
R^2_{val}	$n_d = 2$	-	0.189	0.601	0.975
	$n_d = 3$	0.332	-	0.953	0.937
	$n_d = 5$	0.467	0.660	-	0.903
	$n_d = 10$	0.493	0.617	0.842	-

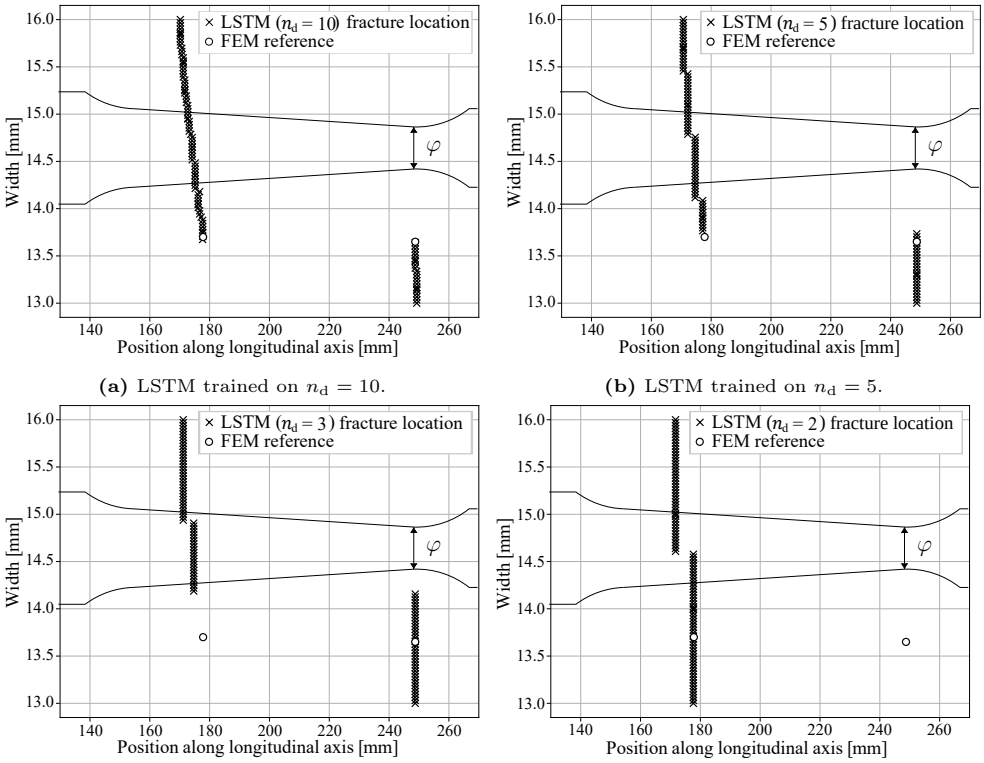
The framework trained for $n_d = 10$ indicates respectable overall generalization (R^2_{train} and R^2_{val}) with $R^2_{val} > 0.9$ for all of its validation data sets $n_d = 2$, $n_d = 3$ and $n_d = 5$. The networks with a lower number of trained samples show less compatibility for generalization, as indicated by lower R^2_{val} values.

The fracture behavior is characterized by the distinct bifurcation regions at either the narrow or wide side of the gauge section, depending on the width φ. Thus the representation of the bifurcation location is of key importance: the critical shift from the narrow to the wide side or vise versa.

In order to obtain more insight in the validity of the performed prediction in the bifurcation region, the fracture location over the complete considered domain is represented using the observed bifurcation locations for 100 linearly spread prediction samples for $\varphi \in [\varphi = 13.0\,\text{mm}, \varphi = 16.0\,\text{mm}]$ for the networks trained on $n_d = 2$, $n_d = 3$, $n_d = 5$ and $n_d = 10$, see Fig. 3.10. Here the locations of elements with the highest fracture value at $t = t_{end}$ for $\varphi = 13.70\,\text{mm}$ and $13.65\,\text{mm}$ are shown. Between these critical values for φ the bifurcation is observed, corresponding to the observations in the experiments.

Analogous to the observations on the generalization of the LSTM networks over the validation sets, the network with a larger n_d shows the closest relation to the performed FEM simulations over the full sample domain. However, the network trained with $n_d = 3$ is already able to fully capture the two distinct wide/narrow-side bifurcation modes: a single fracture longitudinal coordinate on the narrow side, where the wide side shows a linear trend for an increasing width. The more samples in the domain $\varphi \geq 13.70\,\text{mm}$, the more accurate the linear trend can be depicted.

Accurate prediction of the two bifurcation modes therefore strongly depends on the chosen sample locations and is found to be superior to the number of samples. Without a priori knowledge of the considered fracture system the proper choice of sample positions majorly depends on the chosen sampling approach, which indicates a considerable weakness of non-intrusive surrogate models and could lead to unnecessary computational expenses.



(a) LSTM trained on $n_\mathrm{d} = 10$.

(b) LSTM trained on $n_\mathrm{d} = 5$.

(c) LSTM trained on $n_\mathrm{d} = 3$.

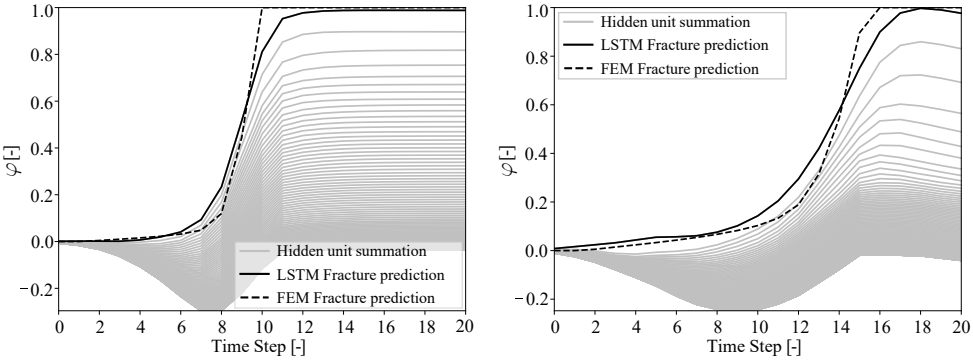
(d) LSTM trained on $n_\mathrm{d} = 2$.

Figure 3.10: For the LSTM networks trained on $n_\mathrm{d} = 2$, $n_\mathrm{d} = 3$, $n_\mathrm{d} = 5$ and $n_\mathrm{d} = 10$ the longitudinal position of the hexagonal element with the highest fracture indicator at the final time step is depicted for 100 linearly spread samples of $\varphi \in [13.0\,\mathrm{mm}, 16.0\,\mathrm{mm}]$.

LSTM state development for bifurcating elements

To obtain more insight in the LSTM state, an analysis on the LSTM state development over time for $n_\mathrm{m} = 128$ is performed for two elements that are selected using a FEM simulation at $\varphi = 13.15\,\mathrm{mm}$ and $\varphi = 15.85\,\mathrm{mm}$, obtained from $n_\mathrm{d} = 10$ as described previously. A single element of interest is selected at both sides of the location shift.

Due to a linear output layer activation, the predicted output, the fracture indicator d_f directly translates to the hidden state of the LSTM cell. As the fracture indicator is a linear combination of the hidden states one may inspect the cumulative contribution of each hidden state to the output. These are shown in Fig. 3.11 in gray, and their importance is sorted by significance. The total summation is shown by a solid black curve. The element specific d_f is largely determined by a strong minority of the available network weights, indicating sparsity of the output layer. At $\varphi = 13.15\,\mathrm{mm}$ (Fig. 3.11a) the first three major state curves describe 23.4% of the total fracture value at $t = t_\mathrm{end}$, but at $\varphi = 15.85\,\mathrm{mm}$ (Fig. 3.11b) the three most significant curves describe 41.2%.



(a) Summed up individual fracture contributors for $\varphi = 13.15\,\text{mm}$.

(b) Summed up individual fracture contributors for $\varphi = 15.85\,\text{mm}$.

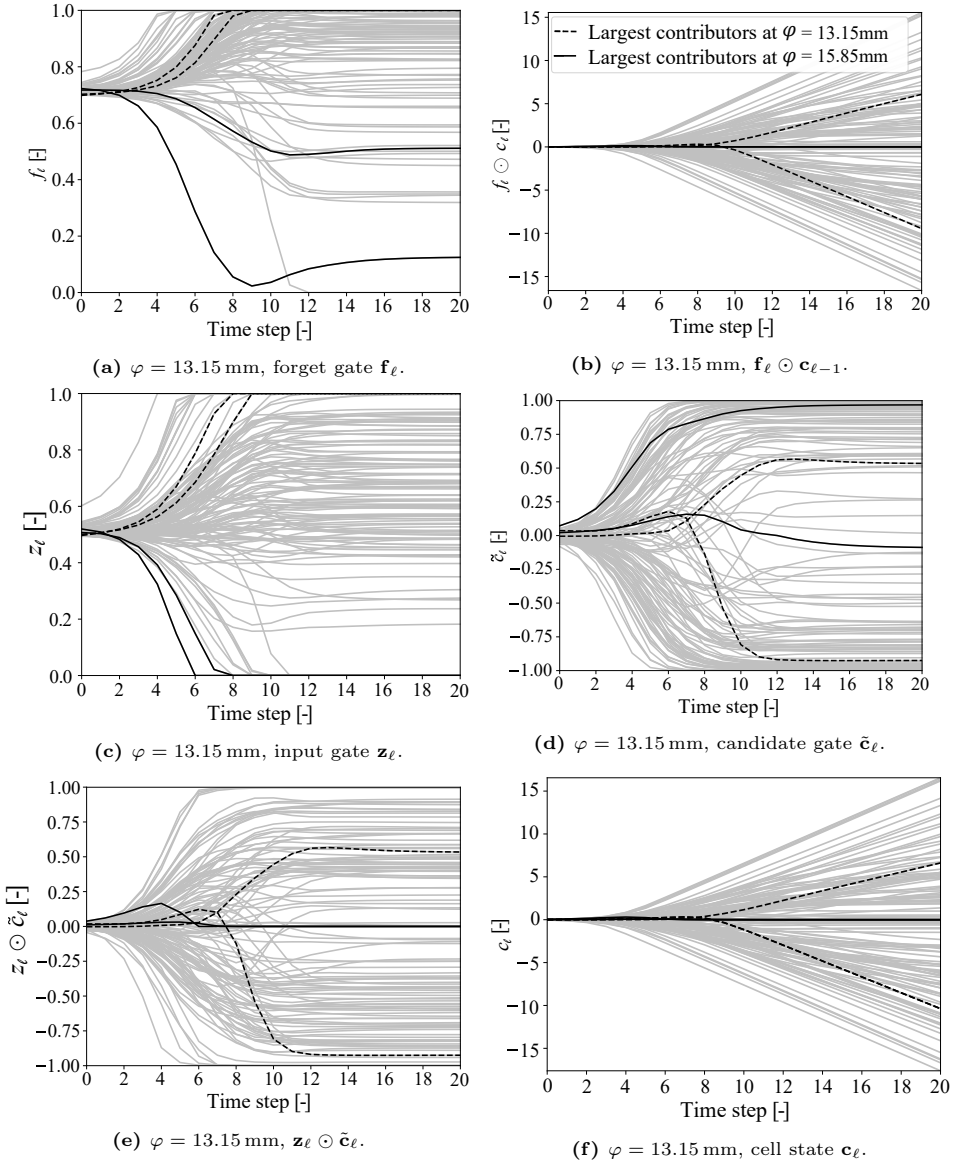
Figure 3.11: Summation of all individual hidden units for the critical fracture element at $\varphi = 13.15\,\text{mm}$ and $\varphi = 15.85\,\text{mm}$, sorted by significance and leading to the fracture prediction curve.

Forward propagation of the LSTM hidden state through time starts by the introduction of the normalized width, together with a zero initial hidden state, visualized in Fig. 3.12. The two units with the largest contribution (i.e. significance) to the critical elements as indicated in Fig. 3.11 are emphasized by black curves in Fig. 3.12. At a zero cell state, the forget gate exerts no influence, holding that a differentiation from the zero state is introduced by the input gate and candidate gate. In these gates, the decision is made which unit values to update based on the initial width and hidden state. At $\varphi = 13.0\,\text{mm}$, both the initial hidden state and normalized width input are equal to zero, holding that the non-multiplicative bias is the only parameter with the ability to create a nonzero state. For the input gate $\mathbf{z}_\ell$ (see Eq. (3.44)), a value of 1.0 yields a largely affected unit value, where 0.0 transfers a limited information flow.

The input gate and candidate gate combination (Fig. 3.12e, Fig. 3.12n) keeps the states that correspond to a low value of d_f close to zero, ensuring a "close to zero" input to the cell state $\mathbf{c}_\ell$ over time. The output gate can also exert no influence on this cell state $\mathbf{c}_\ell$: a (close to) zero-state output is thus obtained, limiting the continuous increase of d_f. In the hidden state evolution, the bifurcation between the two elements is represented by values approaching zero-state for the lower values of d_f and upper bound values for higher values of d_f, see Fig. 3.12i. Beware that a negative value can be compensated by an equal sign weight in the output layer. At the onset of bifurcation, the fracture development is greatly influenced by its history, see Fig. 3.12a and Fig. 3.12j, by raising the forget gate state value to unity.

The continuous polarization of the cell state values at the onset of bifurcation indicate a strong history dependency. When the value of fracture increases the forget gate indicates to store most cell state information as the highest density of the forget gate values is found between 0.5 and 1.0. The bifurcation of both selected (finite) elements is initiated at the combination of the input gate and candidate gate: a sample and history-dependent decision boundary is defined. After initiation, the bifurcation is

amplified through time by the remainder of gates.



(a) $\varphi = 13.15\,\mathrm{mm}$, forget gate $\mathbf{f}_\ell$.

(b) $\varphi = 13.15\,\mathrm{mm}$, $\mathbf{f}_\ell \odot \mathbf{c}_{\ell-1}$.

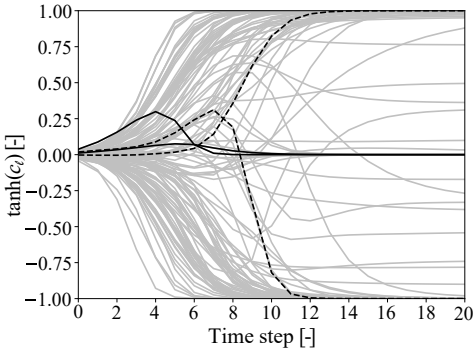
(c) $\varphi = 13.15\,\mathrm{mm}$, input gate $\mathbf{z}_\ell$.

(d) $\varphi = 13.15\,\mathrm{mm}$, candidate gate $\tilde{\mathbf{c}}_\ell$.

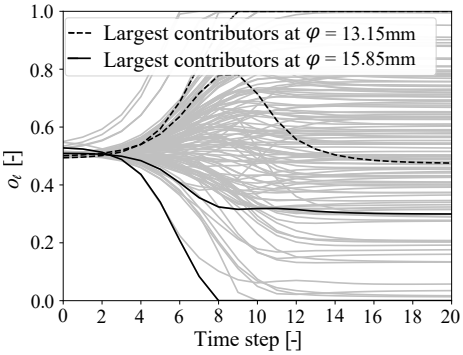
(e) $\varphi = 13.15\,\mathrm{mm}$, $\mathbf{z}_\ell \odot \tilde{\mathbf{c}}_\ell$.

(f) $\varphi = 13.15\,\mathrm{mm}$, cell state $\mathbf{c}_\ell$.

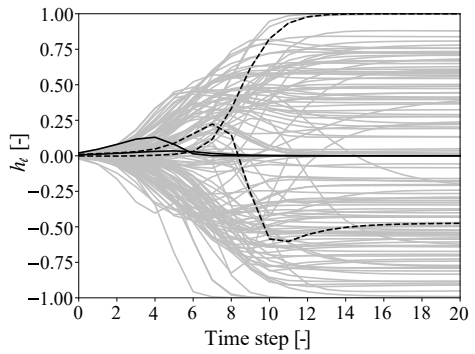
Figure 3.12: LSTM cell gate output and state evolution over time for critical fracture elements at $\varphi = 13.15\,\mathrm{mm}$ and $\varphi = 15.85\,\mathrm{mm}$, shown for the with $n_\mathrm{d} = 10$ sample trained network. *(cont.)*



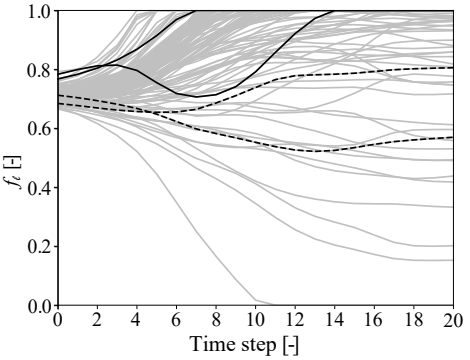
(g) $\varphi = 13.15\,\mathrm{mm}$, $\tanh(\mathbf{c}_\ell)$.



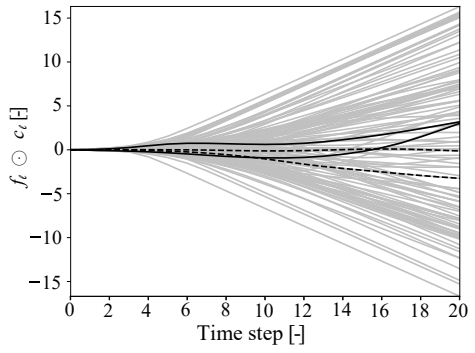
(h) $\varphi = 13.15\,\mathrm{mm}$, output gate $\mathbf{o}_\ell$.



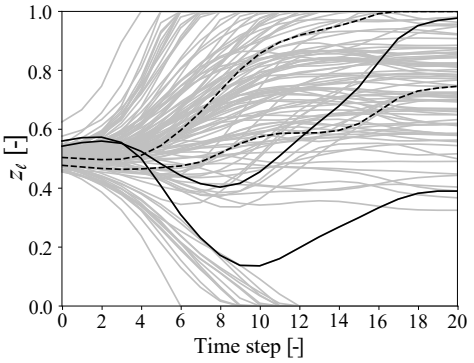
(i) $\varphi = 13.15\,\mathrm{mm}$, hidden state $\mathbf{h}_\ell$.



(j) $\varphi = 15.85\,\mathrm{mm}$, forget gate $\mathbf{f}_\ell$.

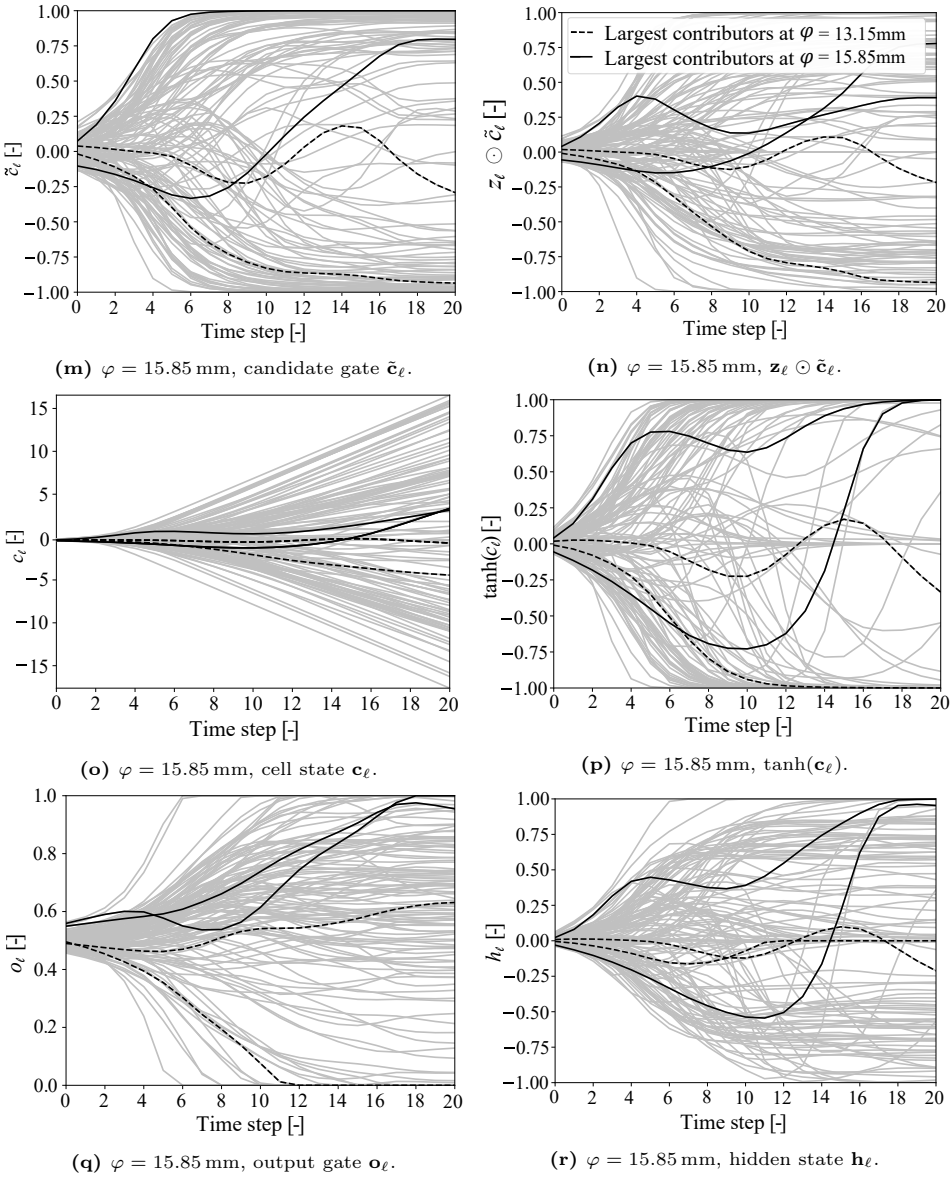


(k) $\varphi = 15.85\,\mathrm{mm}$, $\mathbf{f}_\ell \odot \mathbf{c}_{\ell-1}$.



(l) $\varphi = 15.85\,\mathrm{mm}$, input gate $\mathbf{z}_\ell$.

Figure 3.12: LSTM cell gate output and state evolution over time for critical fracture elements at $\varphi = 13.15\,\mathrm{mm}$ and $\varphi = 15.85\,\mathrm{mm}$, shown for the with $n_\mathrm{d} = 10$ sample trained network. *(cont.)*



(m) $\varphi = 15.85\,\text{mm}$, candidate gate $\tilde{\mathbf{c}}_\ell$.

(n) $\varphi = 15.85\,\text{mm}$, $\mathbf{z}_\ell \odot \tilde{\mathbf{c}}_\ell$.

(o) $\varphi = 15.85\,\text{mm}$, cell state $\mathbf{c}_\ell$.

(p) $\varphi = 15.85\,\text{mm}$, $\tanh(\mathbf{c}_\ell)$.

(q) $\varphi = 15.85\,\text{mm}$, output gate $\mathbf{o}_\ell$.

(r) $\varphi = 15.85\,\text{mm}$, hidden state $\mathbf{h}_\ell$.

Figure 3.12: LSTM cell state evolution over time for critical fracture elements at $\varphi = 13.15\,\text{mm}$ and $\varphi = 15.85\,\text{mm}$, shown for the with $n_\text{d} = 10$ sample trained network.

A closer look is taken at the decision made in the LSTM cell that ensures proper description of the bifurcation (i.e. fracture) location shift. Again, the $n_\text{d} = 10$ samples training data set is selected, but reduced to only the two aforementioned critical elements of the finite element structure: a single element at both sides of the bifurcation location shift. In addition, the number of units in the LSTM cell is reduced from $n_\text{m} = 128$ to $n_\text{m} = 2$. In order to obtain more insight in the bifurcation location

shift, the gate outputs are plotted at the final time step over a width ranging between its predefined boundaries $\varphi \in [13.0\,\text{mm}, 16.0\,\text{mm}]$. In Fig. 3.13 the resulting curves are plotted for all gates. As shown with Fig. 3.10, a fracture location shift is present between $\varphi = 13.65\,\text{mm}$ and $\varphi = 13.70\,\text{mm}$. In this region the predicted fracture d_f values change in magnitude according to the fracture location. This alteration in predicted fracture values is initiated at the gates, visible by the alteration in unit with the largest magnitude. For the input gate no shift in critical unit is present, which is also not required due to the multiplicative nature of both the candidate gate and the input gate in combination with the sign change of the candidate gate. Al-

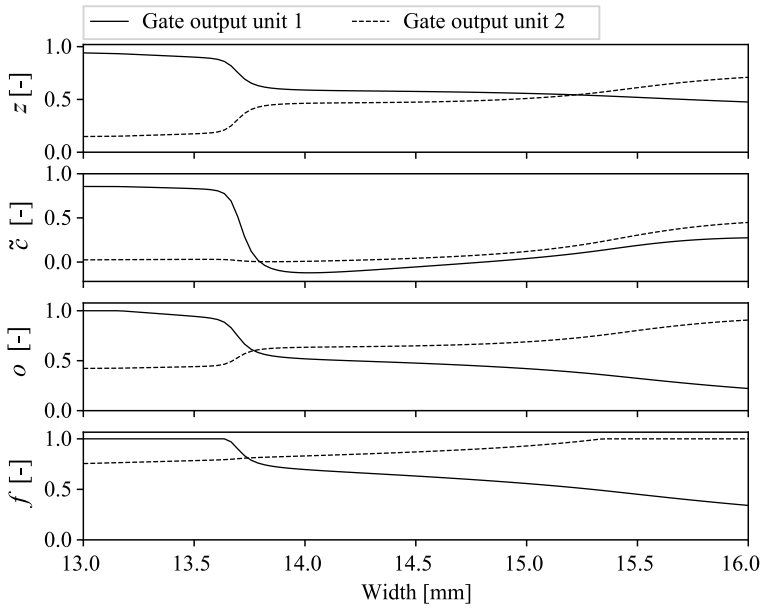


Figure 3.13: Gate output values at $t = t_\text{end}$ as function of the width φ.

though heavily simplified with only two elements, the LSTM model is able to make the distinction. The presence of such decision indicates great capabilities of LSTM-based NNs in the prediction of history dependent bifurcating quantities in arbitrary structures of varying complexity.

3.5.2 Bending specimen strip

The second academic example is defined by a metal strip, subjected to a punch impact, parametrized by its position along the longitudinal x-axis.

Structural model adaption

The metal specimen strip is 1.5 mm thick and 150 mm long, as represented in Fig. 3.14. The strip is fixed using rotational bearings on both lateral sides. The punch is given a constant mass of 10 kg and an initial impact speed of $2.0\,\mathrm{m\,s^{-1}}$. Additionally, the simulation time is set to 20 ms with an output frequency of 2 kHz, hence $n_\mathrm{t} = 41$ time steps. The goal is to predict the nodal displacement (x, y, z) using the data present on a total of $n_\mathrm{d} = 7$ samples, by equidistant sampling uniform distribution of the punch position over the given parameter space.

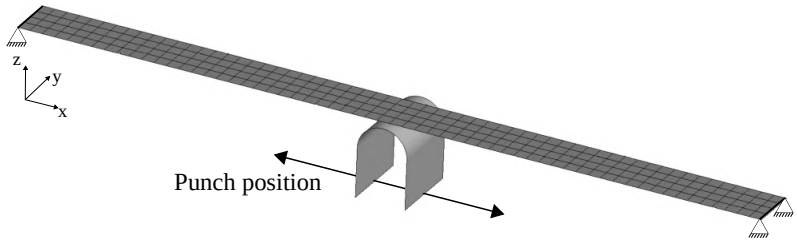


Figure 3.14: Bending test overview, with indicated the varying punch position along the longitudinal axis.

The punch is considered rigid and is limited to a displacement in z-direction only. The material used in the specimen is the aforementioned 22MnB5 hot-forming steel. The material is modeled by an isotropic elasto-plastic strain hardening according to a Hershey yield criterion. A more elaborated description of the applied fundamental material model is given in Section 2.2. The explicit VPS 2017 solver has been applied, with extensions for shells as described in [14].

Optimization parameters

In Table 3.3 $n_\mathrm{d} = 7$ data samples (i.e. designs) are chosen within the parameter space $\varphi \in [-60\,\mathrm{mm}, 60\,\mathrm{mm}]$ with the longitudinal midpoint as specimen center. In Fig. 3.15 the maximum deformation is visualized in the xz-plane indicating a convex envelope that describes maximum deformations for varying punch positions.

Table 3.3: DOE specification.

Design number	1	2	3	4	5	6	7
φ [mm]	−60	−40	−30	0	20	40	60

The constant mesh contains of 305 nodes, and hence a total of $n_\mathrm{o} = 915$ (x, y, z) displacement output values are to be predicted per time step. With $n_\mathrm{m} \in \{16, 32, 64, 128\}$

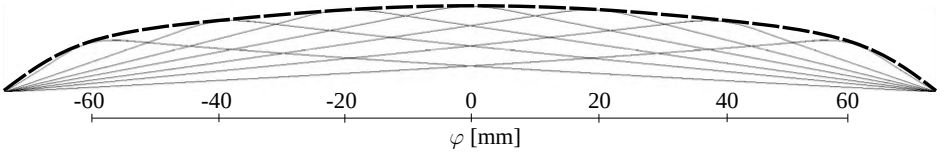


Figure 3.15: Bending specimen designs location indication and overlaying convex envelope.

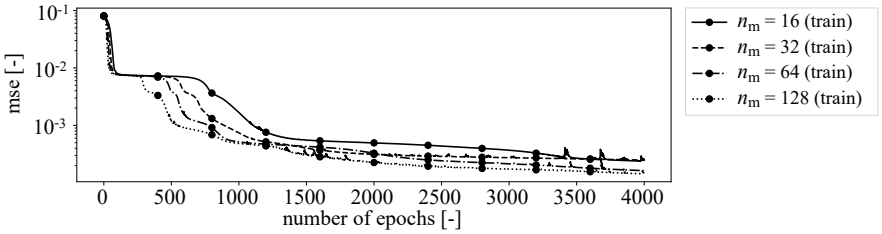
a variety of LSTM widths are analyzed. Additionally, for every considered network width n_m four networks are trained, each with a different sample point $\varphi \in \{0\,\mathrm{mm}, 20\,\mathrm{mm}, 40\,\mathrm{mm}, 60\,\mathrm{mm}\}$ as validation, with the remainder of the data samples used to train the network with. As example, with validation set $\varphi = 0\,\mathrm{mm}$ the network is trained using $\varphi \in \{-60\,\mathrm{mm}, -40\,\mathrm{mm}, -20\,\mathrm{mm}, 20\,\mathrm{mm}, 40\,\mathrm{mm}, 60\,\mathrm{mm}\}$. Additionally, a case without validation set is considered, according to which the model is thus trained on all $n_\mathrm{d} = 7$ data samples. In the upcoming the trained networks are distinguished by their width n_m and choice of validation sample.

The learning rate is set to $\lambda = 0.001$ and a maximum number of 4000 epochs is considered. The deterministic network is optimized over a single batch, hence $n_\mathrm{d} = n_\mathrm{b}$. The LSTM networks have been created and optimized using the Tensorflow v2.1.0 framework [172] on a single Nvidia Quadro P5000 GPU.

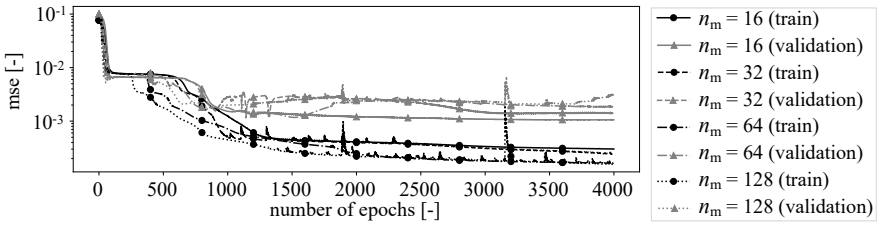
Assessment of the model convergence

For all network sizes n_m and validation samples $\varphi \in \{\text{no validation}, 0\,\mathrm{mm}, 20\,\mathrm{mm}, 40\,\mathrm{mm}, 60\,\mathrm{mm}\}$, the mean square optimization curves are given in Fig. 3.16. A relatively stable optimization was found for all validation samples and state dimensions n_m. Despite quickly converging validation losses (gray curves) for most validation samples, the network dimension n_m only weakly affects the final loss and rate of convergence of the validation domain. The closer the validation sample is to the outer boundary ($\varphi_\mathrm{val} : 0\,\mathrm{mm} \rightarrow 60\,\mathrm{mm}$), the higher the overall validation loss, where for $\varphi_\mathrm{val} = 20\,\mathrm{mm}$ strong over-fitting is observed for $n_\mathrm{m} = 32$ and $n_\mathrm{m} = 64$, see Fig. 3.16c.

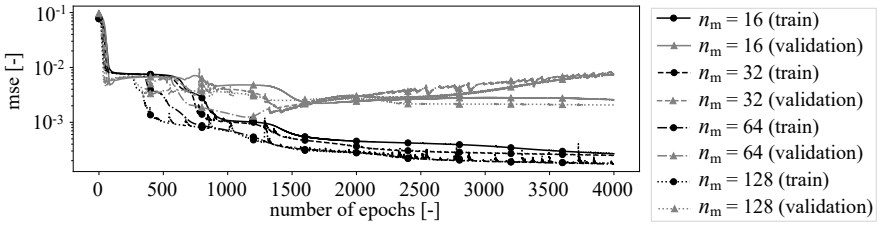
Additionally, n_m is shown to exert only minor effect on the mean squared error for the training samples. Hence, one may question the generalization performance regarding potential under- or over-fitting caused by an improper network dimension n_m and validation sample location.



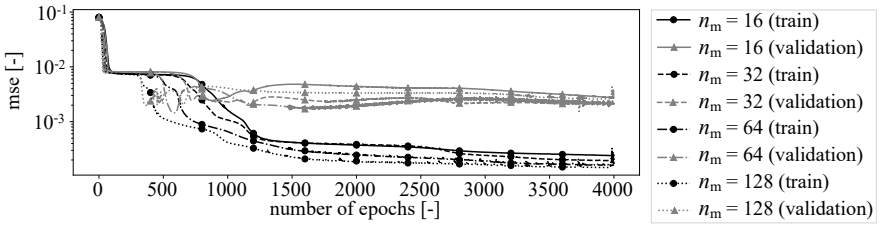
(a) No validation set.



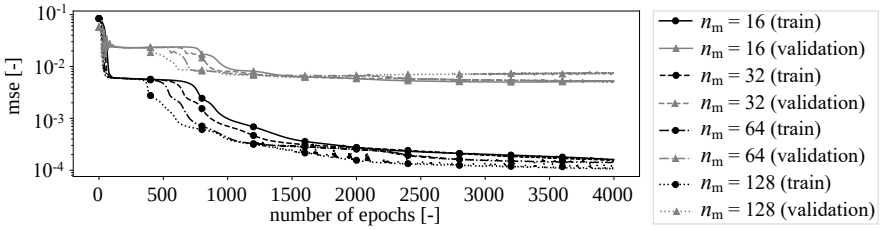
(b) $\varphi_{\mathrm{val}} = 0\,\mathrm{mm}$.



(c) $\varphi_{\mathrm{val}} = 20\,\mathrm{mm}$.



(d) $\varphi_{\mathrm{val}} = 40\,\mathrm{mm}$.



(e) $\varphi_{\mathrm{val}} = 60\,\mathrm{mm}$.

Figure 3.16: Development of the mean squared error during training over the number of epochs for $n_{\mathrm{m}} \in \{16, 32, 64, 128\}$ and five different validation samples.

Assessment of the sample space generalization

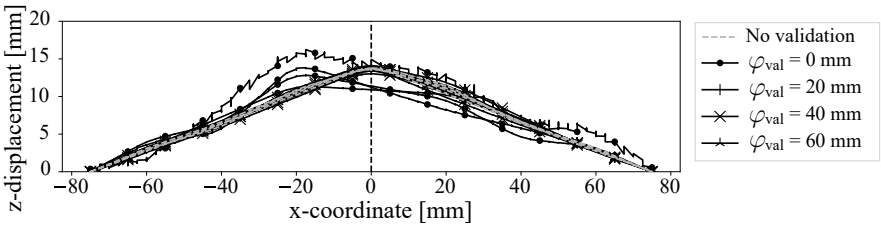
The R^2 metrics between the predictive output $\hat{\mathbf{y}}_\ell(\varphi)$ and corresponding design $\mathbf{y}_\ell(\varphi)$ with $\ell = 1, ..., n_\mathrm{t}$ for the LSTM network with $n_\mathrm{m} \in \{16, 32, 64, 128\}$ are given in Table 3.4. With lower determination coefficients found for R^2_vali as for R^2_train, each network behaves as expected in the given validation/train sampling domain. The strong over-fitting found for $\varphi_\mathrm{val} = 20\,\mathrm{mm}$ is also indicated by lower values for R^2_vali.

The global accuracy of the trained networks significantly differs, as indicated by R^2_vali. Hence, with a different validation sample φ_val one may obtain a significant difference in generalization, with the weakest generalization found at the outer boundary sample for $\varphi_\mathrm{val} = 60\,\mathrm{mm}$.

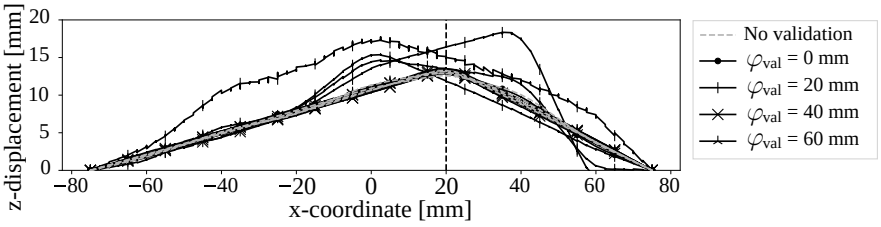
Table 3.4: Optimization metrics for all considered network widths ($n_\mathrm{m} \in \{16, 32, 64, 128\}$.

φ_val	-				$\varphi = 0\,\mathrm{mm}$				$\varphi = 20\,\mathrm{mm}$			
n_m	16	32	64	128	16	32	64	128	16	32	64	128
R^2_train	0.993	0.993	0.995	0.996	0.991	0.993	0.995	0.995	0.992	0.992	0.995	0.995
R^2_vali	-	-	-	-	0.978	0.970	0.961	0.935	0.941	0.829	0.818	0.953
R^2_all	0.993	0.993	0.995	0.996	0.988	0.988	0.988	0.983	0.983	0.963	0.963	0.987
φ_val					$\varphi = 40\,\mathrm{mm}$				$\varphi = 60\,\mathrm{mm}$			
n_m					16	32	64	128	16	32	64	128
R^2_train					0.993	0.994	0.995	0.994	0.996	0.996	0.996	0.997
R^2_vali					0.918	0.934	0.933	0.915	0.688	0.684	0.693	0.557
R^2_all					0.983	0.986	0.987	0.983	0.975	0.974	0.975	0.967

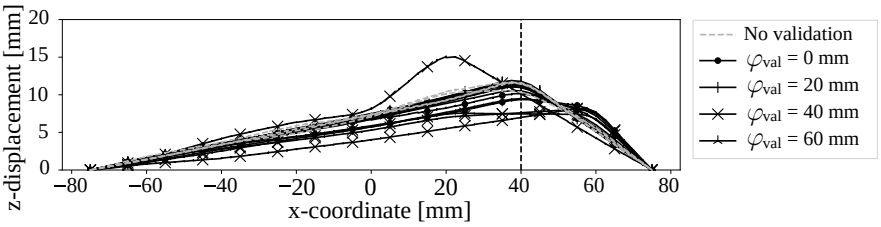
With traditional point-estimate driven NNs one does not obtain any information from the trained network regarding the information content in the chosen data point, nor its importance. In similar manner it is difficult to specify the number of required points for training. To visualize the impact of lacking knowledge on the sampling domain, the trained networks with $n_\mathrm{m} \in \{16, 32, 64, 128\}$ are applied to predict the z-displacement at $\varphi \in \{0\,\mathrm{mm}, 20\,\mathrm{mm}, 40\,\mathrm{mm}, 60\,\mathrm{mm}\}$ (see Fig. 3.17a-d), all chosen on one side of the specimen because of spatial symmetry. In addition, an interpolation position $\varphi = -10\,\mathrm{mm}$ is considered (see Fig. 3.17e), which has not been seen in the training set for any of the trained networks. In Fig. 3.17a-e the z-displacement at the final time step for input values of $\varphi \in \{0\,\mathrm{mm}, 20\,\mathrm{mm}, 40\,\mathrm{mm}, 60\,\mathrm{mm}\}$ is given for the networks with the network dimensions $n_\mathrm{m} \in \{16, 32, 64, 128\}$ that are trained with different data samples for validation $\varphi_\mathrm{val} \in \{\text{no validation}0\,\mathrm{mm}, 20\,\mathrm{mm}, 40\,\mathrm{mm}, 60\,\mathrm{mm}\}$ where the remainder of the $n_\mathrm{d} = 7$ is used as training set. The predicted displacement curves are separated by the choice of validation sample and thus per validation sample φ_val a total of four network widths are applied. For every chosen validation sample φ_val, the predictive z-displacement curve shown from the xz-plane is represented for all four pre-defined network state dimensions n_m.



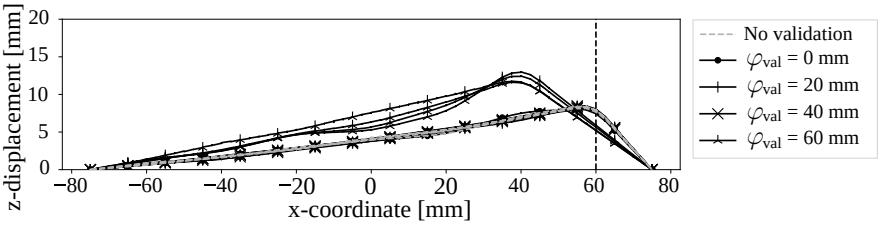
(a) Predicted deformation at $\varphi = 0\,\text{mm}$.



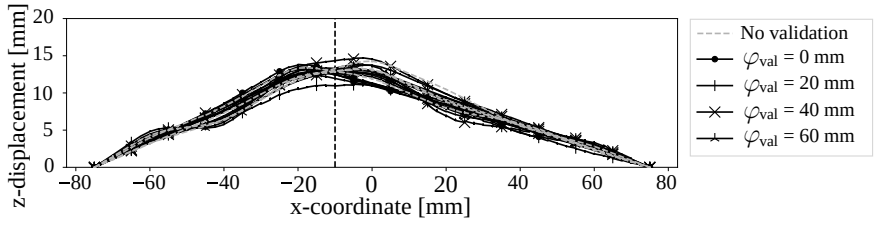
(b) Predicted deformation at $\varphi = 20\,\text{mm}$.



(c) Predicted deformation at $\varphi = 40\,\text{mm}$.



(d) Predicted deformation at $\varphi = 60\,\text{mm}$.



(e) Predicted deformation at $\varphi = -10\,\text{mm}$ (interpolation).

Figure 3.17: z-displacement at $t = t_{\text{end}}$ for different input punch positions φ on the set of trained networks for $n_{\text{m}} \in \{16, 32, 64, 128\}$.

For all network dimensions n_{m} trained without validation set one can observe a proper prediction of the imposed deformation at all evaluated positions. At the interpolation $\varphi = -10\,\mathrm{mm}$, see Eq. (3.17e), the evaluated network dimensions for all considered validation sets all approach the FEM solution with sufficient accuracy. However, the use of a different validation sample (and thus of a different train sample set) causes the networks to behave significantly differently over the complete domain due to a lack of provided information in the region of the chosen validation sample, as one may see in Fig. 3.17a-d for values of $\varphi \in \{0\,\mathrm{mm}, 20\,\mathrm{mm}, 40\,\mathrm{mm}, 60\,\mathrm{mm}\}$. Hence, one may question the accuracy in interpolation and the selection of a properly positioned –sufficiently large– training and validation sample set.

LSTM models are sensitive to the choice of the training data set. Next to this, the issue is the validation of the model. In case of small data sets, one cannot easily choose the training as well as validation set. For example, if the training set is small one can get a good R^2-value although the accuracy of the model compared to reality can be small. Similarly, if the validation set is not well chosen, one may obtain a misleading conclusion. With mean-based point-estimate driven networks, one is unable to solve this issue. Thus, the approach as described in the following chapter is needed.

With a relatively simple academic example it has been shown that the LSTM cell is well capable to distinguish a bifurcating structural fracture response, indicating its capabilities in systems alike. However, with the model accuracy and generalization strongly depending on the selected sample domain and chosen network width, one requires elaborated system knowledge to properly describe the system of unknown equations with a surrogate model. This is again confirmed by the second academic example, where a different validation set φ_{val} results in a strong difference in generalization over this validation set (see Table 3.4). With various network dimensions considered in Table 3.4, one may observe –depending on the chosen φ_{val}– a strongly fluctuating R^2_{vali}. For different validation samples on small data sets the described problem complexity may increase or decrease.

Depending on the chosen network type and architecture, one may have high risk to obtain a framework with a parameter-response mapping that is either too shallow or too complex for the problem given. Such phenomenon is also known as respectively under- or over-fitting. A popular approach to reduce the risk of over-fitting is to reduce the network dimension during optimization by the parameter (i.e. weight) dropout [131]. In the random dropout procedure a predetermined percentage of randomly selected weights is disregarded per iteration step. However, random dropout affects the predictability of the network due to the random selection of weights. Additionally, the amount of random dropout required is unknown a priori. To ensure that a proper network flexibility in its architecture is provided, one needs to evaluate the relevance of every individual weight and its neural connections in the framework. To assess the relevance of network weights one may first describe the weights in a probabilistic sense, as further proposed in this work.

4 Probabilistic surrogate modeling

In this chapter the deterministic neural network framework is transformed to a probabilistic one. At first, the basic principles of probabilistic formulations are elaborated, followed by two distinct methodologies to assimilate the neural network weights in a stochastic sense using the data provided. After selection of the appropriate methodology for stochastic weight optimization, this approach is applied for the recurrent LSTM neural network scheme.

4.1 Relevance determination

The classical deterministic training of NNs is deeply rooted in the numerical practice today. However, the deterministic formulation is focused on the point-based estimates of the model parameters, and hence is prone to instabilities when data are subjected to noise. Thus, to have reliable estimates one is required to pre-process the data and remove the noise in both input and output. Next to this, the gradient-based approaches as described in the previous chapter do not promote self- and incremental learning [82, 87], meaning that the NN weights can be learned in an online manner but only with the existing data set. If the new data set is to be provided, then the network has to be retrained. This partially can be overcome by transfer learning [173]. The main idea is to fix the pre-trained NN models in some part of the architecture, whereas the remaining part is re-trained to conform with the new data set. In this manner one may re-use existing NN architectures for data sets coming from different application areas. Furthermore, the deterministic modeling also does not account for the modeling errors and uncertainties that are related to the lack of the modelers knowledge such as the choice of the NN architecture needed to describe the problem, the choice of its width and height.

To partially overcome previously described issues, and hence promote for the self-organized network, in this thesis a new way of learning is proposed that is framed in the probabilistic framework. Here, under self-organized network one understands the network that can learn its shape, i.e. importance of its weighted connections, as well as its width and height. Thus, the newly proposed training algorithm leads to a network that can self-adapt on the fly, and can learn from the incremental data. The algorithm is based on probabilistic numerics [85], in which the deterministic version of the algorithm is observed as its stochastic counterpart. Namely, instead of the direct weight estimation by the gradient descent approach, one introduces the probabilistic learning problem in which weights are described by a priori experts knowledge further assimilated with the measurement data via Bayes' rule [86, 87, 99]. To encourage self-adaptivity, a special type of sparse prior distributions is proposed. Thus, the new learning rule allows automatic identification and elimination of the irrelevant weight connections by the help of which the NN self-adapts. Due to its probabilistic nature, the new algorithm also allows incremental learning in which data can be sequentially assimilated on the fly.

To describe the new procedure, one may observe a multi-input single output neuron, see Eq. (3.8), described by

$$\hat{y} = \sigma(\boldsymbol{\Phi}\boldsymbol{w}) \tag{4.1}$$

in which $\boldsymbol{\Phi} := [1\ \boldsymbol{q}]$ is a basis function consisting of $n_{\mathrm{f}} + 1$ elements, and $\boldsymbol{w} := [\boldsymbol{w}_{\mathrm{s}}\ w_{\mathrm{b}}] \in \mathbb{R}^{n_{\mathrm{f}}+1}$ is the collection of all unknown weights. The activation function σ may be linear or nonlinear. The goal is to estimate the unknown weights $\boldsymbol{w}$ given the input-output data set $(\boldsymbol{q}_i, y_i), \quad i = 1, ..., n_{\mathrm{b}}$. As the weights in Eq. (4.1) are unknown they may be modeled as uncertain, and hence described as random variables (RVs) $\boldsymbol{w}(\omega_w)$ with finite variance in a probability space $(\Omega_w, \mathfrak{F}_w, \mathbb{P}_w)$[1], see Fig. 4.1. On the

[1]Note that these random variables are independent from the input parameter set. They also do

left side of Fig. 4.1 a schematic overview of a plain feed-forward NN with deterministic weights is presented, and a schematic overview of a similar network but alternatively with the weights being modeled as random variables is presented on the right.

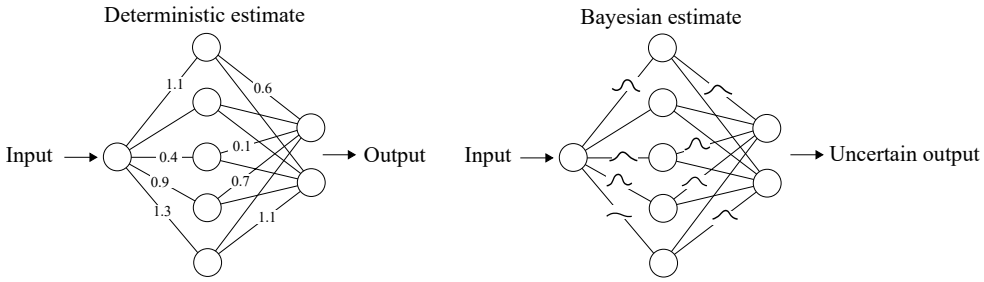


Figure 4.1: Comparison between a plain feed-forward NN with deterministic weights (left) and probabilistic weights (right), adapted from [2].

By describing weights as uncertain, the NN output becomes uncertain as well. Following Eq. (4.1) one may forecast the output as:

$$\tilde{y}(\omega_w) = \sigma(\boldsymbol{\Phi}\boldsymbol{w}(\omega_w)), \tag{4.2}$$

whereas the prediction of the real observation is given by

$$\hat{y}(\omega_w) = \tilde{y}(\omega_w) + \varepsilon(\omega_\varepsilon), \tag{4.3}$$

in which y is a scalar real-valued QoI, $\boldsymbol{\Phi} := [1\ \boldsymbol{q}]$ is the basis function defined by the input and $\boldsymbol{w}$ are the $n_\mathrm{f} + 1$ unknown weights. Thus, the prediction of the network output with n_b samples can be written as:

$$\tilde{\boldsymbol{y}}(\omega_w) = \boldsymbol{\sigma}(\boldsymbol{s}); \qquad \boldsymbol{s} := \boldsymbol{\Phi}\boldsymbol{w}(\omega_w), \tag{4.4}$$

in which $\boldsymbol{\Phi} := [..., \boldsymbol{\Phi}_i, ...], i = 1, ..., n_\mathrm{b}$ is the sampled input parameter set and $\boldsymbol{\sigma}$ is a vector-valued activation function consisting of n_b scalar-valued activation functions σ on $\mathbb{R}$ that are applied element-wise, i.e. $\boldsymbol{\sigma}(\boldsymbol{s}) = [\sigma(s_1), ..., , \sigma(s_{n_\mathrm{b}})] = [\sigma(\boldsymbol{\Phi}_1\boldsymbol{w}(\omega_w)), ..., \sigma(\boldsymbol{\Phi}_{n_\mathrm{b}}\boldsymbol{w}(\omega_w))]$.

With the help of Eq. (4.4) one can predict the data set by:

$$\hat{\boldsymbol{y}}(\omega_w, \omega_\varepsilon) = \tilde{\boldsymbol{y}}(\omega_w) + \tilde{\boldsymbol{\varepsilon}}(\omega_\varepsilon). \tag{4.5}$$

Here, $\tilde{\boldsymbol{\varepsilon}}(\omega_\varepsilon)$ denotes the prediction of the measurement/modeling error and is described by a random variable with a finite variance on a probability space $(\Omega_\varepsilon, \mathfrak{F}_\varepsilon, \mathbb{P}_\varepsilon)$ that is independent of $\boldsymbol{w}(\omega_w)$ and $\boldsymbol{q}(\omega)$. Denoting the joint space $(\Omega_s, \mathfrak{F}_s, \mathbb{P}_s)$ with $\Omega_s := \Omega_w \times \Omega_\varepsilon$, one may rewrite Eq. (4.5) as:

$$\hat{\boldsymbol{y}}(\omega_s) = \tilde{\boldsymbol{y}}(\omega_s) + \tilde{\boldsymbol{\varepsilon}}(\omega_s). \tag{4.6}$$

not have the same meaning. Whereas the uncertainty in the parameter set is of the "aleatoric" type, the uncertainty in the weights is epistemic and represents our uncertainty about their values.

Assuming that the weights $\boldsymbol{w}(\omega_s)$ follow the prior probability density function [2] $p(\boldsymbol{w})$, one may estimate their value given noisy data $\boldsymbol{y} \in \mathbb{R}^{n_\mathrm{b}}$, $\boldsymbol{y} := [..., y_i, ...], i = 1, ..., n_\mathrm{b}$ by the use of Bayes' theorem [86, 87]:

$$p(\boldsymbol{w}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{w})p(\boldsymbol{w})}{P(\boldsymbol{y})}. \tag{4.7}$$

In other words, the prior $p(\boldsymbol{w})$ is updated to the posterior probability density function $p(\boldsymbol{w}|\boldsymbol{y})$ given the likelihood function $p(\boldsymbol{y}|\boldsymbol{w})$ and the evidence $P(\boldsymbol{y})$ (i.e. the normalization factor). The likelihood function $p(\boldsymbol{y}|\boldsymbol{w})$ describes the likelihood of the data set $\boldsymbol{y}$ given the parameters $\boldsymbol{w}$, and is a function defined by the shape of the measurement/modeling error.

The normalization factor in Eq. (4.7) can be rewritten to:

$$P(\boldsymbol{y}) = \int p(\boldsymbol{y}|\boldsymbol{w})p(\boldsymbol{w})\mathrm{d}\boldsymbol{w}, \tag{4.8}$$

in which the integral denotes the evidence of the data set. However, for the computation of $P(\boldsymbol{y})$ by sampling one must account for a large number of samples, making the posterior estimation computationally intractable [2]. Therefore, in practice algorithms are often used that do not focus on the full posterior distribution estimation, but its moments or the corresponding point estimates. An example of such a point estimate is the maximum likelihood estimation (MLE). This estimate is obtained by maximizing the log-likelihood function [2]:

$$\boldsymbol{w}^{\mathrm{MLE}} = \underset{\boldsymbol{w}}{\operatorname{argmax}} \log\, p(\boldsymbol{y}|\boldsymbol{w}). \tag{4.9}$$

Since the log is a monotonic function, the log-likelihood has maxima at exactly the same places where the likelihood has maxima. On the other hand, one may search for a maximum a posteriori (MAP) [2, 174] point estimate over the weights according to:

$$\boldsymbol{w}^{\mathrm{MAP}} = \underset{\boldsymbol{w}}{\operatorname{argmax}} \log\, p(\boldsymbol{w}|\boldsymbol{y}) = \operatorname{argmax}\ \log\, p(\boldsymbol{y}|\boldsymbol{w}) + \log\, p(\boldsymbol{w}). \tag{4.10}$$

In contrast to MLE, this estimate describes the mode of the a posteriori distribution in Eq. (4.7), and is further focus of this work. To obtain more insight in Eq. (4.10) one may further simplify Eq. (4.4) and assume that the activation function $\sigma(\cdot)$ is of the linear type as further described in the text.

4.1.1 Linear relevance determination

Assuming that the activation function $\sigma(\cdot)$ in Eq. (4.2) is linear, one has:

$$\tilde{y}(\omega_s) = \boldsymbol{\Phi}\boldsymbol{w}(\omega_s) \tag{4.11}$$

[2] the joint distribution of all weights is Gaussian with a diagonal covariance, meaning that they are all assumed to be independent

which further simplifies Eq. (4.10) and leads to an MAP estimate that may take an analytical form depending on the choice of the prior and likelihood function. If the prior $p(\boldsymbol{w})$ is assumed to be Gaussian (visually presented in Fig. 4.2a):

$$p(\boldsymbol{w}) = \prod_{j=1}^{n_\mathrm{f}+1} p(w_j); \qquad p(w_j) \sim \mathcal{N}(0, \sigma_w^2) \tag{4.12}$$

and the measurement noise is of the white type:

$$p(\boldsymbol{y}|\boldsymbol{w}) = \prod_{i=1}^{n_\mathrm{b}} \mathcal{N}(y_i|\tilde{y}_i, \sigma_y^2), \tag{4.13}$$

then one has:

$$p(\boldsymbol{w}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{w})p(\boldsymbol{w}), \tag{4.14}$$

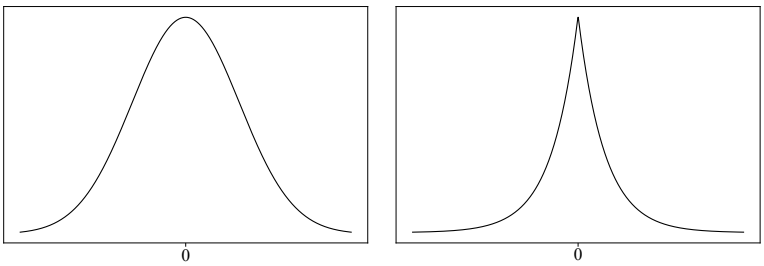
i.e.

$$p(\boldsymbol{w}|\boldsymbol{y}) \propto \prod_{i=1}^{n_\mathrm{b}} \mathcal{N}(y_i|\tilde{y}_i, \sigma_y^2) \prod_{j=1}^{n_\mathrm{f}+1} \mathcal{N}(w_j|0, \sigma_w^2). \tag{4.15}$$

Taking the logarithm of both the right hand side and the left hand side in the previous equation and eliminating the constants, one obtains:

$$\log\, p(\boldsymbol{w}|\boldsymbol{y}) \propto -\frac{1}{\sigma_y^2}\langle \boldsymbol{y} - \tilde{\boldsymbol{y}}, \boldsymbol{y} - \tilde{\boldsymbol{y}} \rangle - \frac{1}{\sigma_w^2}\boldsymbol{w}^2 + \text{const.} \tag{4.16}$$

By maximizing the Gaussian likelihood (i.e. MLE) with respect to $\boldsymbol{w}$ one obtains for the posterior estimate the weighted least square solution [87] in which the weight plays the role of the measurement covariance matrix. The least squares optimization can therefore be understood in a probabilistic sense. On the other hand, when maximizing the posterior estimate (i.e. MAP), one obtains the solution which corresponds to the regularized weighted least squares solution in which the prior term corresponds to the ℓ_2 regularization [87].



(a) Gaussian distribution function. **(b)** Laplacian distribution function.

Figure 4.2: Prior distributions, with (a) a Gaussian prior and (b) a heavily sparsity favoring Laplace prior.

As made visible in Fig. 4.2a a Gaussian prior is not sharply peaked around the imposed zero-mean and is hence not strongly favoring zero-weights. This is also clear from a

l_2 regularization perspective. Thus, a Gaussian prior takes all neural connections into account, and does not discard any of them. Hence, the neuron cannot self-adapt. In practice some of the existing connections are not important, i.e. the output is insensitive to some of the input values and thus this has to be taken into account. A more sparsity favoring alternative would be the sharply peaked Laplacian distribution, as visually represented in Fig. 4.2b. In case when the prior over the neuron weights is Laplacian $\mathcal{I}$ [175] one has:

$$p(\boldsymbol{w}) = \prod_{j=1}^{n_\mathrm{f}+1} p(w_j), \qquad p(w_j) \sim \mathcal{I}(0, \sigma_w) \tag{4.17}$$

such that for Gaussian likelihood one may write the posterior distribution, see Eq. (4.7), as:

$$p(\boldsymbol{w}|\boldsymbol{y}) = \prod_{i=1}^{n_\mathrm{b}} \mathcal{N}(y_i|\tilde{y}_i, \sigma_y^2) \prod_{j=1}^{n_\mathrm{f}+1} \mathcal{I}(w_j|0, \sigma_w). \tag{4.18}$$

By taking the logarithm one obtains:

$$\log\, p(\boldsymbol{w}|\boldsymbol{y}) = -\frac{1}{\sigma_y^2}\langle \boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w}, \boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w}\rangle - \frac{1}{\sigma_w}|\boldsymbol{w}| + \mathrm{const}, \tag{4.19}$$

with the basis function being $\boldsymbol{\Phi} \in \mathbb{R}^{n_\mathrm{b}\times(n_\mathrm{f}+1)}$, the weights being $\boldsymbol{w}$ and the QoI being defined by $\boldsymbol{y} \in \mathbb{R}^{n_\mathrm{b}}$. From the previous equation one may see that the MAP estimate in case of a Laplace prior corresponds to the weighted least squared error (LSE) in which one applies ℓ_1 regularization for the weights [87, 176].

To solve Eq. (4.19) in a computationally efficient manner, one requires a prior distribution that is conjugate to the likelihood [87]. If conjugacy is present one is capable to analytically determine the posterior distribution and hence obtain an exact solution. However, the posterior in Eq. (4.19) cannot be evaluated directly as a Laplace distribution is not conjugate with the Gaussian likelihood function. To still pursue a sparsity favoring prior distribution the posterior must be estimated by using approximation approaches such as variational inference. Variational Bayesian methods are a family of techniques for approximating intractable integrals arising in Bayesian inference and machine learning [87, 155]. An example of the variational inference approach is the Kullback-Leibler-divergence (KL-divergence) method [2, 98]. This method is elaborated in Appendix A.1. However, the KL-divergence procedure heavily depends on the chosen type of approximate distribution, the shape of the prior and posterior as well as the reparametrization noise. Additionally, the KL approach is based on a sampling type of approach that is often computationally unattractive in NNs applications [89, 94]. To ensure a cheaper way of estimating the posterior distribution and simultaneously enforce sparsity in the NN framework, a novel computational approach is proposed subsequently.

To estimate the posterior given in Eq. (4.19) one may employ the theory of automatic relevance determination (ARD) [99]. In ARD the problem of non-conjugacy of the Laplace prior and the Gaussian likelihood is overcome by defining hyperpriors that

emulate a Laplace distribution. Following the ARD scheme one may describe the weights by

$$p(\boldsymbol{w}|\boldsymbol{\alpha}) = \prod_{j=1}^{n_f+1} \mathcal{N}(w_j|0, \alpha_j^{-1}) \tag{4.20}$$

in which the prior variance is described by a uniform (non-informative) type of hyperprior $p(\boldsymbol{\alpha})$. On a logarithmic scale the latter one matches with a non-informative Gamma distribution [99]:

$$p(\boldsymbol{\alpha}) = \prod_{j}^{n_f+1} \Gamma(a)^{-1} b^a \alpha_j^{a-1} \exp(-b\alpha_j) \tag{4.21}$$

in which $\Gamma(a) := \int_0^\infty t^{a-1} \exp(-t)\mathrm{d}t$ and a, b are the identical shape and scale parameters taken over all weights, respectively.

Assuming that the model in Eq. (4.4) is described by a zero-mean Gaussian noise (due to modeling or data error), one may model the noise term by a variance σ_G, taken to be the same for all data points. Note that each data point is assumed to be characterized by an independent noise term of the same distribution type, and thus the same variance σ_G. As the noise variance σ_G is often unknown, one can also model it as uncertain, and hence infer from data. Here, this is achieved by modeling the precision $\beta := \sigma_G^{-1}$ by a hyperprior:

$$p(\beta) = \Gamma(c)^{-1} d^c \beta^{c-1} \exp(-d\ \beta), \tag{4.22}$$

with c, d being respectively the shape and scale parameter of the Gamma distribution. In a special case when $a = b = c = d = 0$ one obtains a uniform hyperprior.

Under previously made assumptions, one may write the full Bayes rule for estimating unknown weights, its hyperparameters and the measurement precision as:

$$p(\boldsymbol{w}, \boldsymbol{\alpha}, \beta|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{w}, \boldsymbol{\alpha}, \beta) p(\boldsymbol{w}, \boldsymbol{\alpha}, \beta)}{P(\boldsymbol{y})}, \tag{4.23}$$

in which $p(\boldsymbol{w}, \boldsymbol{\alpha}, \beta)$ is the (hyper)prior over all unknowns, and the remaining terms have the similar meaning as in Eq. (4.7). Due to the previously mentioned reasons, the estimation of the full posterior distribution in Eq. (4.23) is computationally infeasible. To reduce computational effort, in this thesis the mean-field approximation [177] of the posterior distribution is considered in which the posterior $p(\boldsymbol{w}, \boldsymbol{\alpha}, \beta|\boldsymbol{y})$ is factorized as a product of the posterior over the weights only $p(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{\alpha}, \beta)$ and the posterior over the hyperparameters $p(\boldsymbol{\alpha}, \beta|\boldsymbol{y})$, i.e.:

$$p(\boldsymbol{w}, \boldsymbol{\alpha}, \beta|\boldsymbol{y}) = p(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{\alpha}, \beta) p(\boldsymbol{\alpha}, \beta|\boldsymbol{y}). \tag{4.24}$$

The term $p(\boldsymbol{w}, \boldsymbol{\alpha}, \beta|\boldsymbol{y})$ in Eq. (4.24) is obtained via:

$$p(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{\alpha}, \beta) = \frac{p(\boldsymbol{y}|\boldsymbol{w}, \beta) p(\boldsymbol{w}|\boldsymbol{\alpha})}{p(\boldsymbol{y}|\boldsymbol{\alpha}, \beta)}, \tag{4.25}$$

which represents a convolution of Gaussians, and hence the corresponding posterior can be computed directly. Using the derivation given in Appendix A.2 the posterior $p(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{\alpha}, \beta)$ in the previous equation follows a Gaussian distribution, the mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ of which are respectively given by

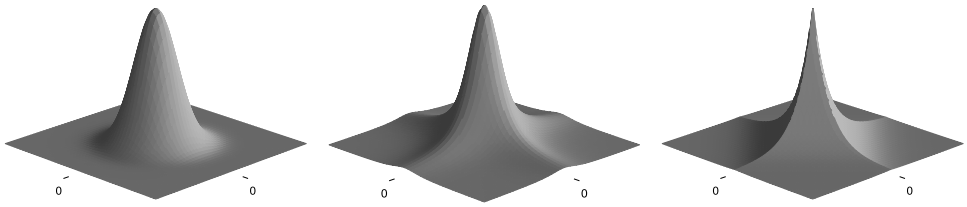
$$\begin{aligned} \boldsymbol{\mu} &:= \beta \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \boldsymbol{y}, \\ \boldsymbol{\Sigma} &:= (\beta \boldsymbol{\Phi}^T \boldsymbol{\Phi} + \boldsymbol{\alpha} \odot \mathbf{I})^{-1} \end{aligned} \tag{4.26}$$

with $\boldsymbol{\Phi} \in \mathbb{R}^{n_\mathrm{b} \times (n_\mathrm{f}+1)}$, $\boldsymbol{y} \in \mathbb{R}^{n_\mathrm{b}}$ and $\mathbf{I} \in \mathbb{R}^{(n_\mathrm{f}+1)\times(n_\mathrm{f}+1)}$. Additionally, $\boldsymbol{\Sigma} \in \mathbb{R}^{(n_\mathrm{f}+1)\times(n_\mathrm{f}+1)}$ and $\boldsymbol{\mu} \in \mathbb{R}^{n_\mathrm{f}+1}$ yielding hyperparameter $\boldsymbol{\alpha} \in \mathbb{R}^{n_\mathrm{f}+1}$ and $\beta \in \mathbb{R}$.

To evaluate the effect of hyperprior $p(\boldsymbol{\alpha})$ on the Gaussian hierarchical prior $p(\boldsymbol{w}|\boldsymbol{\alpha})$ (see Fig. 4.3a) one needs to integrate out $\boldsymbol{\alpha}$:

$$\begin{aligned} p(\boldsymbol{w}) &= \int p(\boldsymbol{w}|\boldsymbol{\alpha}) p(\boldsymbol{\alpha}) \mathrm{d}\boldsymbol{\alpha} \\ &= \frac{\boldsymbol{b}^{\boldsymbol{a}} \Gamma(\boldsymbol{a} + \frac{1}{2})}{(2\pi)^{\frac{1}{2}} \Gamma(\boldsymbol{a})} (\boldsymbol{b} + \frac{\boldsymbol{w}^2}{2})^{-(\boldsymbol{a}+\frac{1}{2})}. \end{aligned} \tag{4.27}$$

The obtained result describes a Student-t distribution over $p(\boldsymbol{w})$, see Fig. 4.3b [99]. In a special situation when $\boldsymbol{a} = \boldsymbol{b} = 0$ one obtains $p(\boldsymbol{w}) \propto 1/\sqrt{\boldsymbol{w}^2}$, which resembles the Laplace prior, and hence favors sparsity, see Fig. 4.3c [178]. Although a relatively non-sparse Gaussian prior $p(\boldsymbol{w}|\boldsymbol{\alpha})$ is assumed, the addition of a uniform hyperprior ensures sparsity, making the prior sharply peak around zero. However, as $\boldsymbol{\alpha}$ is integrated out, $p(\boldsymbol{w})$ is non-Gaussian, and cannot be described as a convolution of Gaussians.



(a) Multivariate Gaussian. **(b)** Multivariate Student-t. **(c)** Multivariate Laplace.

Figure 4.3: Multivariate prior distributions, with (a) a multivariate Gaussian prior; (b) a Student-t prior and (c) a heavily sparsity favoring Laplace distribution.

Given the first term in Eq. (4.24), it remains to estimate $p(\boldsymbol{\alpha}, \beta|\boldsymbol{y})$. Assuming independence of $p(\boldsymbol{\alpha})$ and $p(\beta)$, one may further write:

$$p(\boldsymbol{\alpha}, \beta|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{\alpha}, \beta) p(\boldsymbol{\alpha}) p(\beta). \tag{4.28}$$

According to [82, 99] one may approximate the posterior by the most probable value:

$$p(\boldsymbol{\alpha}, \beta|\boldsymbol{y}) \approx \delta(\boldsymbol{\alpha}^{\mathrm{MP}}, \beta^{\mathrm{MP}}) \tag{4.29}$$

for computational simplicity. Here, $\boldsymbol{\alpha}^{\mathrm{MP}}$ and β^{MP} denote the most probable values of respectively $\boldsymbol{\alpha}$ and β. This completes the description of the decomposed posterior in Eq. (4.24).

In order to estimate the hyperparameters $\boldsymbol{\alpha}$ and β given the data $\boldsymbol{y}$ the objective is to maximize the marginal likelihood:

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{\alpha},\beta) &= \int p(\boldsymbol{y}|\boldsymbol{w},\beta)p(\boldsymbol{w}|\boldsymbol{\alpha})\mathrm{d}\boldsymbol{w} \\ &= (2\pi)^{-n_\mathrm{d}/2}\left|\beta^{-1}+\boldsymbol{\Phi}(\boldsymbol{\alpha}^{-1}\odot\mathbf{I})\boldsymbol{\Phi}^T\right|^{-1/2} \\ &\quad \exp\left[-\frac{1}{2}\boldsymbol{y}(\beta^{-1}+\boldsymbol{\Phi}(\boldsymbol{\alpha}^{-1}\odot\mathbf{I})\boldsymbol{\Phi}^T)^{-1}\boldsymbol{y}^T\right]. \end{aligned} \tag{4.30}$$

After taking the logarithm the objective function reads [99]:

$$\mathcal{L}^{\alpha,\beta} = -\frac{1}{2}\left[\log\left|\beta^{-1}+\boldsymbol{\Phi}(\boldsymbol{\alpha}^{-1}\odot\mathbf{I})\boldsymbol{\Phi}^T\right| + \boldsymbol{y}^T\left(\beta^{-1}+\boldsymbol{\Phi}(\boldsymbol{\alpha}^{-1}\odot\mathbf{I})\boldsymbol{\Phi}^T\right)^{-1}\boldsymbol{y}\right]. \tag{4.31}$$

Finally, $\boldsymbol{\alpha}$ and β can be estimated via gradient based optimization [99] to obtain:

$$\frac{\partial\mathcal{L}^{\alpha,\beta}}{\partial\log\,\alpha_j} = \frac{1}{2}\left[1-\alpha_j(\mu_j^2+d_j)\right], \qquad j=1,...,n_\mathrm{f}+1. \tag{4.32}$$

with d_j being the jth diagonal element of the covariance matrix $\boldsymbol{\Sigma}$ defined in Eq. (4.26). From optimal conditions one obtains:

$$\alpha_j = \frac{1}{\mu_j^2+d_j}. \tag{4.33}$$

Similarly, one may write

$$\frac{\partial\mathcal{L}_k^{\alpha,\beta}}{\partial\log\,\beta} = \frac{1}{2}\left[\frac{n_\mathrm{d}}{\beta}-\|\boldsymbol{y}-\boldsymbol{\Phi}\boldsymbol{\mu}\|^2-\beta^{-1}\sum_j\gamma_j\right], \qquad j=1,...,n_\mathrm{f}+1, \tag{4.34}$$

giving

$$\beta = \frac{n_\mathrm{d}-\sum_j\gamma_j}{\|\boldsymbol{y}-\boldsymbol{\Phi}\boldsymbol{\mu}\|^2}. \tag{4.35}$$

Parameter γ_j is defined by [99]:

$$\gamma_j = 1-\alpha_j d_j. \tag{4.36}$$

According to this equation $\gamma_j\to 0$ if w_j is majorly constrained by its prior α_j while being uncertain in its likelihood ($d_j\approx\alpha_j^{-1}$). In such case the corresponding weight μ_j can be set to zero.

Posterior predictive

With optimized most probable values for $\boldsymbol{\alpha}$ and β one can subsequently compute the posterior predictive $p(\hat{\boldsymbol{y}}|\boldsymbol{y})$ [99] using the likelihood $p(\hat{\boldsymbol{y}}|\boldsymbol{w},\beta^{\mathrm{MP}})$ and posterior over the weights $p(\boldsymbol{w}|\boldsymbol{y},\boldsymbol{\alpha}^{\mathrm{MP}},\beta^{\mathrm{MP}})$, giving:

$$p(\hat{\boldsymbol{y}}|\boldsymbol{y}) = \mathcal{N}(\hat{\boldsymbol{y}}|\hat{\boldsymbol{\mu}}^{\mathrm{y}},\hat{\boldsymbol{\sigma}}^2), \tag{4.37}$$

with

$$\hat{\boldsymbol{\mu}}^{\mathrm{y}} = \boldsymbol{\Phi}\boldsymbol{\mu} \tag{4.38}$$

and

$$\hat{\sigma}_i^2 = (\beta^{\mathrm{MP}})^{-1} + \boldsymbol{\Phi}_i^T \boldsymbol{\Sigma} \boldsymbol{\Phi}_i, \quad i = 1, ..., n_{\mathrm{b}}. \tag{4.39}$$

Here, $\boldsymbol{\Phi}_i := [1\ \boldsymbol{q}_i]$, $\hat{\boldsymbol{\mu}}^{\mathrm{y}}$ and $\hat{\boldsymbol{\sigma}}^2 := [\hat{\sigma}_i^2]_{i=1}^{n_{\mathrm{b}}}$ are the input feature vector, predictive mean and predictive variance, respectively.

4.1.2 Nonlinear relevance determination

The previously presented theory is related to the estimation of weights in a linear setting. However, the neuron in Eq. (4.2), or in general NNs, are characterized by nonlinear activation functions $\sigma(\cdot)$. Therefore, the weight estimation becomes inherently a nonlinear estimation problem for which the previously mentioned ARD theory does not hold. In order to extend and generalize the ARD theory, one has to introduce a framework which deals with nonlinearity. This is not so straightforward as in such a case the estimation of the posterior weights and the corresponding hyperparameters as shown in Eq. (4.30) do not have an analytical form, and thus the sampling approaches are often used [2, 89, 179]. Even in its optimal form the training based on the sampling approach is computationally expensive, and thus not feasible. Therefore, in this work a novel nonlinear ARD theory is suggested in which one imposes the sparsity priors on weights, and approximates the full Bayesian posterior by a computationally cheaper yet sufficiently representative estimate. To achieve this, an approximate Bayes rule is employed that introduces both nonlinearity and sparsity as presented in the following text.

To allow for weight relevance determination in nonlinear frameworks with arbitrary continuous nonlinear activation functions $\sigma(\cdot)$ one may first restate Eq. (4.4) on a joint space $(\Omega_s, \mathfrak{F}_s, \mathbb{P}_s)$ to represent a multi-input single output neuron with predicted output (QoI):

$$\hat{y}(\omega_s) := \sigma(s\,(\omega_s)) + \varepsilon(\omega_s), \tag{4.40}$$

in which

$$s := \boldsymbol{\Phi}\boldsymbol{w}(\omega_s) + \epsilon(\omega_s). \tag{4.41}$$

Here, ε is the measurement/FEM modeling error and ϵ is the modeling error that comes from the approximation of the local state and is unknown. Accordingly, one may define the same hierarchical priors over the unknown weights as in Section 4.1.1. Once the prior uncertainty is propagated through the nonlinear activation function in Eq. (4.40), the forecast of the target data is obtained, thus shaping the likelihood function. However, the newly defined likelihood function is not Gaussian anymore as presented in Section 4.1.1, and is implicitly defined by a nonlinear activation function. This means that the estimation scheme presented in the previous section does not hold anymore. To be able to still apply the linear ARD scheme, one has to separate the weight $\boldsymbol{w}$ learning problem from a the so-called local target $\boldsymbol{s}$ estimation problem. Thus, the main goal is to first estimate the local target $\boldsymbol{s}$ given the measurement data and once the state/local target is known to estimate the corresponding weight $\boldsymbol{w}$ given the estimated $\boldsymbol{s}$ in an ARD manner.

Let us assume that both the local targets $\boldsymbol{s}$ and the weights $\boldsymbol{w}$ are unknown a priori, and hence can be modeled as uncertain. Given a random variable $\boldsymbol{w}(\omega_s)$ following prior distribution $p(\boldsymbol{w})$, the prediction of the local state $\boldsymbol{s}(\omega_s)$ corresponding to n_b samples of the QoI, i.e. $\boldsymbol{s}(\omega_s) = [..., s_i(\omega_s), ...], i = 1, ..., n_\mathrm{b}$, follows the distribution $p(\boldsymbol{s}|\boldsymbol{w})$ implicitly defined by Eq. (4.41). To estimate the posterior joint distribution $p(\boldsymbol{s}, \boldsymbol{w}|\boldsymbol{y})$ of both the local state and the unknown weights, one may use the Bayes rule in the form:

$$p(\boldsymbol{s}, \boldsymbol{w}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{s}, \boldsymbol{w}) p(\boldsymbol{s}|\boldsymbol{w}) p(\boldsymbol{w}). \tag{4.42}$$

Here, $p(\boldsymbol{y}|\boldsymbol{s}, \boldsymbol{w})$ is the likelihood function conditional on the target state $\boldsymbol{s}$ and the parameter $\boldsymbol{w}$, and is implicitly defined by the quantity of interest given in Eq. (4.40). Assuming that the joint distribution in Eq. (4.42) exists and that the conditional distributions $p(\boldsymbol{s}|\boldsymbol{w}, \boldsymbol{y})$ over the local target and $p(\boldsymbol{w}|\boldsymbol{s}, \boldsymbol{y})$ over the weights are compatible [180, 181], one may estimate the aforementioned conditionals by:

$$p(\boldsymbol{s}|\boldsymbol{w}, \boldsymbol{y}) := p(\boldsymbol{s}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{s}) p(\boldsymbol{s}|\boldsymbol{w}) \tag{4.43}$$

and

$$p(\boldsymbol{w}|\boldsymbol{s}, \boldsymbol{y}) := p(\boldsymbol{w}|\boldsymbol{s}) \propto p(\boldsymbol{s}|\boldsymbol{w}) p(\boldsymbol{w}), \tag{4.44}$$

alternatively. Here, the likelihood function $p(\boldsymbol{y}|\boldsymbol{s})$ is shaped by the measurement error ε, whereas $p(\boldsymbol{s}|\boldsymbol{w})$ is shaped by the modeling error ϵ. Both of the errors are assumed to follow a Gaussian distribution with zero mean and unknown precision. The posterior $p(\boldsymbol{s}|\boldsymbol{w}, \boldsymbol{y})$ is further estimated by employing the gradient descent algorithm, whereas the posterior $p(\boldsymbol{w}|\boldsymbol{s}, \boldsymbol{y})$ is obtained via the linear ARD scheme. In the upcoming, the estimation of the former is first described, followed by the description of the latter.

Local target estimation

Estimation of the local target state $\boldsymbol{s}$ (containing n_b elements) is done using a slight alteration of the classical back-propagation as described in Section 3.4.1. Let the conditional distribution over the local targets be $\pi(\boldsymbol{s}) := p(\boldsymbol{s}|\boldsymbol{w}, \boldsymbol{y})$, which is the same as the posterior:

$$p(\boldsymbol{s}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{s}) p(\boldsymbol{s}|\boldsymbol{w}). \tag{4.45}$$

Due to the non-linear character of the activation function $\sigma(\cdot)$ (see Eq. (4.4)), the likelihood $p(\boldsymbol{s}|\boldsymbol{w})$ may describe a non-Gaussian probability density function. Alike, the prior may also be non-Gaussian and hence the posterior $p(\boldsymbol{s}|\boldsymbol{y})$ can consequently be non-Gaussian as well. The likelihood $p(\boldsymbol{y}|\boldsymbol{s})$ and prior $p(\boldsymbol{s}|\boldsymbol{w})$ might therefore be non-conjugate, causing the posterior estimation to be reliant on costly (Monte-Carlo) sampling [142], which makes the estimation computationally expensive. An approximation of the posterior over the local targets is therefore desired.

One may approximate the posterior distribution by seeking:

$$\pi(\boldsymbol{s}) \approx \tilde{\pi}(\boldsymbol{s}) \sim \mathcal{N}(\overline{\boldsymbol{s}}, \mathbf{C}_{\boldsymbol{s}}) \tag{4.46}$$

in which the mean $\overline{\boldsymbol{s}}$ is unknown. This corresponds to the variational inference problem (see Appendix A.1), and the objective is thus to minimize the evidence lower bound:

$$L(\tilde{\pi}) := -\mathbb{E}_{\tilde{\pi}(\boldsymbol{s})}(\log\, p(\boldsymbol{y}|\boldsymbol{s})) + D_\mathrm{KL}(\tilde{\pi}(\boldsymbol{s})||\pi(\boldsymbol{s})) \tag{4.47}$$

in which $D_{\mathrm{KL}}(\tilde{\pi}(\boldsymbol{s})||\pi(\boldsymbol{s}))$ describes the Kullback-Leibler divergence between the distributions $\tilde{\pi}(\boldsymbol{s})$ and $\pi(\boldsymbol{s})$, as also found in Section A.1. Given

$$\mathcal{L}^s := \mathcal{L}(\boldsymbol{s}) = \log\, p(\boldsymbol{y}|\boldsymbol{s}), \tag{4.48}$$

Eq. (4.47) becomes:

$$L(\tilde{\pi}) := -\mathbb{E}_{\tilde{\pi}(\boldsymbol{s})}(\mathcal{L}^s) + D_{\mathrm{KL}}(\tilde{\pi}(\boldsymbol{s})||\pi(\boldsymbol{s})), \tag{4.49}$$

and has to be minimized. The approximating probability density function $\tilde{\pi}(\boldsymbol{s})$ is assumed to belong to the exponential family of distributions generally described by

$$\tilde{\pi}(\boldsymbol{s}) = v(\boldsymbol{s}) \exp\left(\langle \boldsymbol{\lambda}, T(\boldsymbol{s})\rangle - A(\boldsymbol{\lambda})\right) \tag{4.50}$$

or more specifically by

$$\begin{aligned} v(\boldsymbol{s}) &= (2\pi)^{-P/2} \exp(-\frac{1}{2}\boldsymbol{s}^T\boldsymbol{s}), \\ \boldsymbol{\lambda} &= \overline{\boldsymbol{s}}, \\ T(\boldsymbol{s}) &= \overline{\boldsymbol{s}}^T\boldsymbol{s}. \end{aligned} \tag{4.51}$$

Here, $A(\boldsymbol{\lambda})$ is a normalization factor. The problem given in Eq. (4.49) may then be minimized by the so-called Bayesian learning rule [182]:

$$\boldsymbol{\lambda}_{e+1} = (1 - \rho_e \mathbf{C}_{\boldsymbol{s}})\boldsymbol{\lambda}_e - \rho_e \mathbf{C}_{\boldsymbol{s}} \nabla_{\overline{\boldsymbol{s}}} \mathbb{E}_{\tilde{\pi}_e(\boldsymbol{s})} \left(\mathcal{L}(\boldsymbol{s}) + \log\, v(\boldsymbol{s})\right), \tag{4.52}$$

in which e denotes the iteration step and $\tilde{\pi}_e(\boldsymbol{s})$ is the approximating probability density function at iteration step e. One may note that the expectation $\mathbb{E}_{\tilde{\pi}_e(\boldsymbol{s})}(\log\, v(\boldsymbol{s})) = -\overline{\boldsymbol{s}}$ is constant and can thus be taken out of the objective. After removing all constant terms one obtains:

$$\boldsymbol{\lambda}_{e+1} = \boldsymbol{\lambda}_e - \rho_e \mathbf{C}_{\boldsymbol{s}} \nabla_{\boldsymbol{\mu}} \mathbb{E}_{\tilde{\pi}_e(\boldsymbol{s})} \left(\mathcal{L}(\boldsymbol{s})\right)|_{\boldsymbol{\mu}=\boldsymbol{\lambda}_e} \tag{4.53}$$

which in terms of the unknown $\overline{\boldsymbol{s}}$ yields:

$$\overline{\boldsymbol{s}}^{e+1} = \overline{\boldsymbol{s}}^e - \rho_e \mathbf{C}_{\boldsymbol{s}} \nabla_{\boldsymbol{\mu}} \mathbb{E}_{\tilde{\pi}_e(\boldsymbol{s})} \left(\mathcal{L}(\boldsymbol{s})\right)|_{\boldsymbol{\mu}=\overline{\boldsymbol{s}}^e}. \tag{4.54}$$

The previous equation involves the expectation of the loss function (denoted as $\mathbb{E}_{\tilde{\pi}_e(\boldsymbol{s})} \left(\mathcal{L}(\boldsymbol{s})\right)$) and the covariance $\mathbf{C}_{\boldsymbol{s}}$ which makes the previous updating rule different from the classical gradient descent algorithm (Section 3.4.1). For the classical version, one would have to assume that $\mathbf{C}_{\boldsymbol{s}} = \mathbf{I}$ and

$$\nabla_{\overline{\boldsymbol{s}}} \mathbb{E}_{\tilde{\pi}_e(\boldsymbol{s})} \left(\mathcal{L}(\boldsymbol{s})\right) \approx \nabla_{\overline{\boldsymbol{s}}} \mathcal{L}(\overline{\boldsymbol{s}}), \qquad \mathbb{E}\left(\mathcal{L}(\boldsymbol{s})\right) = \mathcal{L}(\mathbb{E}(\boldsymbol{s})) \tag{4.55}$$

to obtain

$$\overline{\boldsymbol{s}}^{e+1} = \overline{\boldsymbol{s}}^e - \rho_e \nabla_{\boldsymbol{\mu}} \mathcal{L}(\overline{\boldsymbol{s}}^e) \tag{4.56}$$

which one may recognize from Eq. (3.24). However, in a general nonlinear case the approximation of Eq. (4.55) does not hold. By using Bonnets theorem [183] and $\mathbf{C}_{\boldsymbol{s}} = I$ one may further state

$$\nabla_{\overline{\boldsymbol{s}}} \mathbb{E}_{\tilde{\pi}_e(\boldsymbol{s})} \left(\mathcal{L}(\boldsymbol{s})\right) = \mathbb{E}_{\tilde{\pi}(\boldsymbol{s})} \nabla_{\overline{\boldsymbol{s}}} \left(\mathcal{L}(\boldsymbol{s})\right), \tag{4.57}$$

meaning that the gradient of the expected value equals the expected value of the gradient. After substitution in Eq. (4.54) one then obtains:

$$\overline{\boldsymbol{s}}^{e+1} = \overline{\boldsymbol{s}}^{e} - \rho_e \mathbb{E}_{\tilde{\pi}_e(\boldsymbol{s})} \nabla_{\boldsymbol{\mu}} \left(\mathcal{L}(\boldsymbol{s}) \right) |_{\boldsymbol{\mu}=\overline{\boldsymbol{s}}^{e}}. \tag{4.58}$$

To estimate the gradient $\boldsymbol{V}_e := \mathbb{E}_{\tilde{\pi}_e(\boldsymbol{s})} \nabla_{\boldsymbol{\mu}} \left(\mathcal{L}(\boldsymbol{s}) \right) |_{\boldsymbol{\mu}=\overline{\boldsymbol{s}}^{e}}$, one may use the first order approximation of the likelihood function

$$\mathcal{L}(\boldsymbol{s}) = \mathcal{L}(\overline{\boldsymbol{s}}^{e}) + \boldsymbol{V}_e(\boldsymbol{s} - \overline{\boldsymbol{s}}^{e}) + \varepsilon_{\mathcal{L}} = \boldsymbol{V}_e \boldsymbol{s} + \boldsymbol{b}_{V,e} + \varepsilon_{\mathcal{L}} \tag{4.59}$$

and the data-driven approach given the loss $\mathcal{L}(\boldsymbol{s})$ as a data set and (Monte-Carlo) sampling of $\boldsymbol{s}$. Thus, one may assume that $\boldsymbol{V}_e$, $\boldsymbol{b}_{V,e}$ and $\varepsilon_{\mathcal{L}}$ are unknown and estimate them. For this one may use a stochastic approach in which these quantities are first assumed as uncertain by use of prior knowledge and then estimated in a Bayesian manner. The gradient $\boldsymbol{V}_e$ is thereafter found in a Bayesian manner. Due to linearity of the problem, the gradient thus can be estimated by use of the ARD approach explained in Section 4.1.1 and in [82]. For a better accuracy Eq. (4.59) can also be approximated by higher order terms to better cover the possible nonlinearity. However, in this work only the linear term is considered to keep the same framework as in classical training approaches by means of back-propagation. The possible approximation error is then fully covered by the estimated approximation error $\varepsilon_{\mathcal{L}}$.

Sparsity-induced weight estimation

By using the algorithm presented in the previous section, one can estimate the mean $\overline{\boldsymbol{s}}^{e} \in \mathbb{R}^{n_\mathrm{b}}, \quad \overline{\boldsymbol{s}}^{e} = [..., \overline{s}_i^{e}, ...], i = 1, ..., n_\mathrm{b}$ of the local target state $\boldsymbol{s}^{e}$. Once the mean of the local target is estimated, one may further adopt it as a new data set instead of $\boldsymbol{y}$. This new data set is, however, characterized by a new measurement error ϵ_s that is further assumed to be unknown. The part of this new virtual measurement error describes the observation noise inherited from $\boldsymbol{y}$, and the other part covers the stochastic spread that exists in the approximated local target posterior.

Given the new data $\overline{\boldsymbol{s}}^{e}$, the goal is to estimate the weight $\boldsymbol{w}^{e}$ knowing the relation given in Eq. (4.41). Assuming that the weight $\boldsymbol{w}^{e}$ and the error ϵ_s are uncertain and described in a Laplace hyperprior setting (see Section 4.1.1), one may further predict the local target observation according to:

$$\hat{\overline{\boldsymbol{s}}}^{e}(\omega_s) = \boldsymbol{\Phi} \boldsymbol{w}^{e}(\omega_s) + \boldsymbol{\epsilon}_s(\omega_s), \tag{4.60}$$

in which $\boldsymbol{w}^{e}(\omega_s)$ and $\boldsymbol{\epsilon}_s(\omega_s)$ are the prior weights and the prior modeling error, respectively. The iteration index e is dropped in the upcoming text for the ease of understanding. Hence, the previous equation can be written as:

$$\hat{\overline{\boldsymbol{s}}}(\omega_s) = \boldsymbol{\Phi} \boldsymbol{w}(\omega_s) + \boldsymbol{\epsilon}_s(\omega_s), \tag{4.61}$$

which represents a linear problem with respect to the weights $\boldsymbol{w}(\omega_s)$ and the error $\boldsymbol{\epsilon}_s(\omega_s)$.

Following previous assumptions, one may further follow the approach presented in Section 4.1.1 to obtain:

$$p(\boldsymbol{w}, \boldsymbol{\alpha}, \beta|\overline{\boldsymbol{s}}) = \frac{p(\overline{\boldsymbol{s}}|\boldsymbol{w}, \boldsymbol{\alpha}, \beta)p(\boldsymbol{w}, \boldsymbol{\alpha}, \beta)}{P(\overline{\boldsymbol{s}})}. \tag{4.62}$$

Here, $\boldsymbol{\alpha}$ is the hyperparameter describing the weight, whereas the parameter β describes the unknown error $\boldsymbol{\epsilon}_s$ [3]. Note that compared to the Bayes rule in Eq. (4.23) the only difference is that the real data set $\boldsymbol{y}$ is substituted by a virtual data set $\overline{\boldsymbol{s}}$ representing the estimated local target, and that the real observation noise is substituted by the one inside of the neuron, $\boldsymbol{\epsilon}_s$. Given Eq. (4.24) and Eq. (4.62), the joint posterior over the unknown parameters can be represented as the product

$$p(\boldsymbol{w}, \boldsymbol{\alpha}, \beta|\overline{\boldsymbol{s}}) = p(\boldsymbol{w}|\overline{\boldsymbol{s}}, \boldsymbol{\alpha}, \beta)p(\boldsymbol{\alpha}, \beta|\overline{\boldsymbol{s}})$$

in which $p(\boldsymbol{w}|\overline{\boldsymbol{s}}, \boldsymbol{\alpha}, \beta)p(\boldsymbol{\alpha}, \beta|\overline{\boldsymbol{s}})$ obtains a Gaussian form with the weight mean $\boldsymbol{\mu}$ and the covariance $\boldsymbol{\Sigma}$:

$$\begin{aligned} \boldsymbol{\mu} &:= \beta \boldsymbol{\Sigma} \boldsymbol{\Phi}^T \overline{\boldsymbol{s}}, \\ \boldsymbol{\Sigma} &:= (\beta \boldsymbol{\Phi}^T \boldsymbol{\Phi} + \boldsymbol{\alpha} \odot \mathbf{I})^{-1}. \end{aligned} \tag{4.63}$$

Here, $\mathbf{I} \in \mathbb{R}^{(n_\mathrm{f}+1)\times(n_\mathrm{f}+1)}$ is the diagonal identity matrix, $\boldsymbol{\Sigma} \in \mathbb{R}^{(n_\mathrm{f}+1)\times(n_\mathrm{f}+1)}$, $\boldsymbol{\Phi} \in \mathbb{R}^{n_\mathrm{b}\times(n_\mathrm{f}+1)}$ and $\overline{\boldsymbol{s}} \in \mathbb{R}^{n_\mathrm{b}}$. Additionally, $\boldsymbol{\mu} \in \mathbb{R}^{n_\mathrm{f}+1}$, yielding vector $\boldsymbol{\alpha} \in \mathbb{R}^{n_\mathrm{f}+1}$.

Accordingly, the description of the most probable estimate of the hyperparameter posterior (see Eq. (4.29)) is given by [82, 99]:

$$p(\boldsymbol{\alpha}, \beta|\overline{\boldsymbol{s}}) \approx \delta(\boldsymbol{\alpha}^{\mathrm{MP}}, \beta^{\mathrm{MP}}). \tag{4.64}$$

Using Eq. (4.31) the marginal likelihood objective function is then given by [99]:

$$\mathcal{L}^{\alpha,\beta}(\overline{\boldsymbol{s}}) = -\frac{1}{2}\left[\log|\mathbf{C}| + \overline{\boldsymbol{s}}^T \mathbf{C}^{-1} \overline{\boldsymbol{s}}\right], \tag{4.65}$$

where $\mathbf{C} := \beta^{-1}\mathbf{I} + \boldsymbol{\Phi}(\boldsymbol{\alpha} \odot \mathbf{I})^{-1}\boldsymbol{\Phi}^T$. From optimal conditions and gradient-based optimization [99] the hyperparameters $\boldsymbol{\alpha}$ and β then yield:

$$\begin{aligned} \alpha_j &= \frac{1}{\mu_j^2 + d_j}, \qquad j = 1, ..., n_\mathrm{f} + 1; \\ \beta &= \frac{n_\mathrm{d} - \sum_j \gamma_j}{\|\overline{\boldsymbol{s}} - \boldsymbol{\Phi}\boldsymbol{\mu}\|^2} \end{aligned} \tag{4.66}$$

with d_j being the jth diagonal element of $\boldsymbol{\Sigma}$. The prior dominance indicator γ_j is given by

$$\gamma_j = 1 - \alpha_j d_j. \tag{4.67}$$

Once the previous posterior is estimated, one may predict the local target state as:

$$p(\hat{\overline{\boldsymbol{s}}}|\overline{\boldsymbol{s}}) = \mathcal{N}(\hat{\overline{\boldsymbol{s}}}|\hat{\boldsymbol{\mu}}^{\mathrm{s}}, \hat{\boldsymbol{\sigma}}^2), \tag{4.68}$$

[3] The error is assumed to follow a Gaussian distribution, although this assumption can be too strict. The extension to a more general case will be studied in the future research

with

$$\begin{aligned}\hat{\boldsymbol{\mu}}^{\mathrm{s}} &= \boldsymbol{\Phi}\boldsymbol{\mu} \\ \hat{\sigma}_i^2 &= (\beta^{\mathrm{MP}})^{-1} + \boldsymbol{\Phi}_i^T \boldsymbol{\Sigma} \boldsymbol{\Phi}_i, \quad i = 1, ..., n_{\mathrm{b}}\end{aligned} \tag{4.69}$$

defining $\hat{\boldsymbol{\mu}}^{\mathrm{s}}$ and $\hat{\boldsymbol{\sigma}}^2$ as the predictive mean and variance that may be propagated through the nonlinear activation function by:

$$\hat{\boldsymbol{y}} = \boldsymbol{\sigma}(\hat{\overline{\boldsymbol{s}}}) + \boldsymbol{\varepsilon}. \tag{4.70}$$

Given the newly introduced optimization scheme to obtain the weights it is now possible to automatically identify the importance of weighted connections and introduce sparsity in a single nonlinear neuron. However, deep feed forward neural networks with multiple neurons (i.e. layers) are yet left untouched. To allow for the automatic identification of weighted connections and introduction of sparsity in a multi-layer nonlinear framework one is required to estimate the local target state and weights of every specific layer and neuron. This requires the introduction of a more elaborate local target and weight estimation scheme through back-propagation. Furthermore, vehicle crashworthiness evaluation is generally performed in a time-dependent setting as described in Section 3.3. Thus, the recurrent relationship within the parameter-response mapping must be accounted for. Despite the choice of activation function, the introduced framework only applies for a single layer purely feed-forward setting.

4.2 Recurrent relevance determination

In the previous section a novel optimization scheme is introduced that allows for sparse weight estimation in nonlinear neuron. Yet, to comply with the recurrent and nonlinear relationship in the data obtained from FEM for crash simulations, the previous scheme has to be extended to a NN with time dependency. Hence in this section the nonlinear single neuron framework as presented in Section 4.1.2 is further extended to an approach that corresponds to a nonlinear recurrent neural network (RNN). The vanishing gradient problem as presented in Section 3.3 is directly addressed by employing the LSTM framework to describe the desired nonlinear RNN architecture.

In a time-dependent setting the QoI $\boldsymbol{y}_\ell$ has to be predicted with respect to the time index ℓ (at time moment τ_ℓ), see Section 3.3. For this the following assumptions are made:

- all the weights describing the LSTM cell and hence the network are assumed to be uncertain, and independent. This means that in contrast to the classical LSTM network the weights (see Fig. 3.4) are not shared but are assumed to be time dependent. However, as the traditional LSTM scheme of gates is followed, one ensures that the recurrent relationship of the propagated state over time remains intact, while adapting and switching on and off gate weights per time step. Here, due to simplicity reasons one assumes full independency, although other type of scenarios can be assumed as well.

- the observation noise at each time moment τ_ℓ is assumed to be independent of the observation noise at all other time moments τ_{tt}, i.e. $\tilde{\boldsymbol{\varepsilon}}_\ell$ is independent of $\tilde{\boldsymbol{\varepsilon}}_{tt}$.

After defining the joint space $(\Omega_s, \mathfrak{F}_s, \mathbb{P}_s)$ of all epistemic uncertainties that are introduced, one may further use Eq. (3.44) to predict the stochastic state of the LSTM cell:

$$\begin{aligned}
\boldsymbol{f}_\ell(\omega_s) &= \boldsymbol{\sigma}_{\mathrm{f}}(\boldsymbol{\Phi}_\ell \mathbf{w}_{\mathrm{f},\ell}(\omega_s) + \boldsymbol{\epsilon}_{\mathrm{f},\ell}(\omega_s)) := \boldsymbol{\sigma}_{\mathrm{f}}(\boldsymbol{s}_{\mathrm{f},\ell}(\omega_s)) \\
\boldsymbol{z}_\ell(\omega_s) &= \boldsymbol{\sigma}_{\ell}(\boldsymbol{\Phi}_\ell \mathbf{w}_{\mathrm{z},\ell}(\omega_s) + \boldsymbol{\epsilon}_{\mathrm{z},\ell}(\omega_s)) := \boldsymbol{\sigma}_{\mathrm{z}}(\boldsymbol{s}_{\mathrm{z},\ell}(\omega_s)) \\
\tilde{\boldsymbol{c}}_\ell(\omega_s) &= \boldsymbol{\sigma}_{\tilde{\mathrm{c}}}(\boldsymbol{\Phi}_\ell \mathbf{w}_{\tilde{\mathrm{c}},\ell}(\omega_s) + \boldsymbol{\epsilon}_{\tilde{\mathrm{c}},\ell}(\omega_s)) := \boldsymbol{\sigma}_{\tilde{\mathrm{c}}}(\boldsymbol{s}_{\tilde{\mathrm{c}},\ell}(\omega_s)) \\
\boldsymbol{o}_\ell(\omega_s) &= \boldsymbol{\sigma}_{\mathrm{o}}(\boldsymbol{\Phi}_\ell \mathbf{w}_{\mathrm{o},\ell}(\omega_s) + \boldsymbol{\epsilon}_{\mathrm{o},\ell}(\omega_s)) := \boldsymbol{\sigma}_{\mathrm{o}}(\boldsymbol{s}_{\mathrm{o},\ell}(\omega_s)) \\
\boldsymbol{c}_\ell(\omega_s) &= \boldsymbol{f}_\ell(\omega_s) \odot \boldsymbol{c}_{\ell-1}(\omega_s) + \boldsymbol{z}_\ell(\omega_s) \odot \tilde{\boldsymbol{c}}_\ell(\omega_s) \\
\boldsymbol{h}_\ell(\omega_s) &= \boldsymbol{o}_\ell(\omega_s) \odot \boldsymbol{\sigma}_{\mathrm{c}}(\boldsymbol{c}_\ell(\omega_s)).
\end{aligned} \tag{4.71}$$

Using Eq. (3.45) one may further predict the response of the output layer:

$$\boldsymbol{y}_\ell(\omega_s) = \boldsymbol{\sigma}_{\mathrm{y}}\left(\boldsymbol{\Phi}_{\mathrm{y},\ell} \mathbf{w}_{\mathrm{y},\ell}(\omega_s) + \boldsymbol{\epsilon}_{\mathrm{y},\ell}(\omega_s)\right) + \boldsymbol{\varepsilon}_{\mathrm{y},\ell}(\omega_s), \tag{4.72}$$

in which $\boldsymbol{\Phi}_{\mathrm{y},\ell} := [1\ \mathbb{E}(\boldsymbol{h}_\ell)]$. Given Eq. (4.71) and Eq. (4.73) one may extract the local targets $\boldsymbol{s}_{\mathrm{g},\ell}$ for all gates $\mathrm{g} \in \{\mathrm{f}, \mathrm{z}, \tilde{\mathrm{c}}, \mathrm{o}, \mathrm{y}\}$ in a form of

$$\boldsymbol{s}_{\mathrm{g},\ell} := \boldsymbol{\Phi}_\ell \mathbf{w}_{\mathrm{g},\ell}(\omega_s) + \boldsymbol{\epsilon}_{\mathrm{g},\ell}(\omega_s), \quad \mathrm{g} \in \{\mathrm{f}, \mathrm{z}, \tilde{\mathrm{c}}, \mathrm{o}, \mathrm{y}\}, \tag{4.73}$$

with the weights $\mathbf{w}_{\mathrm{g},\ell}$ consisting of $n_{\mathrm{i}} \times n_{\mathrm{m}}$ random variables and the basis function $\boldsymbol{\Phi}_\ell := [1\ \boldsymbol{q}\ \mathbb{E}(\boldsymbol{h}_{\ell-1})]$ consisting of $n_{\mathrm{i}} := 1 + n_{\mathrm{f}} + n_{\mathrm{m}}$ elements. Note that the basis function in Eq. (4.72) and Eq. (4.73) only contains the expected value of the hidden state. The main reason for this is that the fluctuating part is shifted towards the virtual observation noise $\boldsymbol{\epsilon}_{\mathrm{g},\ell}(\omega_s)$. The virtual observation noise $\boldsymbol{\epsilon}_{\mathrm{g},\ell}$ for all $\mathrm{g} \in \{\mathrm{f}, \mathrm{z}, \tilde{\mathrm{c}}, \mathrm{o}, \mathrm{y}\}$ is furthermore assumed to be uncertain, and independent of all other virtual noises. The description of the quantity of interest can finally be adapted from Eq. (3.31):

$$\tilde{\boldsymbol{y}}_\ell(\omega_s) = \boldsymbol{\sigma}_{\mathrm{y}}(\omega_s) \circ \boldsymbol{\sigma}_{\mathrm{h}}^{\ell}(\omega_s) \circ \ldots \circ \boldsymbol{\sigma}_{\mathrm{h}}^{1}(\omega_s), \tag{4.74}$$

in which $\boldsymbol{\sigma}_{\mathrm{h}}^{\ell}$ denotes the hidden state arising from the LSTM cell at time step ℓ.

4.2.1 ARD-LSTM

To incorporate the ARD estimation of the weights as well as the states in the whole LSTM network, one requires an adapted algorithm.

Initially, one may assume that weights of LSTM are described by prior distribution $p([\mathbf{w}_{\mathrm{g},\ell}]_{:,k} | [\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k}) \sim \mathcal{N}(0, [\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k}^{-1})$, i.e. the Laplace like hyperprior:

$$p(\mathbf{w}_{\mathrm{g},\ell} | \boldsymbol{\alpha}_{\mathrm{g},\ell}) = \prod_{j=1}^{n_{\mathrm{i}}} \prod_{k=1}^{n_{\mathrm{m}}} \mathcal{N}([\mathbf{w}_{\mathrm{g},\ell}]_{jk} | 0, [\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{jk}^{-1}). \tag{4.75}$$

The hyperparameter $\boldsymbol{\alpha}_{\mathrm{g},\ell}$ is further defined by:

$$p(\boldsymbol{\alpha}_{\mathrm{g},\ell}) = \prod_{j=1}^{n_{\mathrm{i}}} \prod_{k=1}^{n_{\mathrm{m}}} \Gamma(a)^{-1} b^a [\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{jk}^{a-1} \exp(-b[\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{jk}) \tag{4.76}$$

for all gates besides the output transition for which

$$p(\boldsymbol{\alpha}_{\mathrm{y},\ell}) = \prod_{j=1}^{n_\mathrm{m}+1} \prod_{k=1}^{n_\mathrm{o}} \Gamma(a)^{-1} b^a [\boldsymbol{\alpha}_{\mathrm{y},\ell}]_{jk}^{a-1} \exp(-b[\boldsymbol{\alpha}_{\mathrm{y},\ell}]_{jk}). \tag{4.77}$$

Here, $[\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{jk}$ are the elements of the matrix $\boldsymbol{\alpha}_{\mathrm{g},\ell}$ at the time step ℓ.

In a similar manner, one may further define the prior information for the precision of the modeling errors $\boldsymbol{\epsilon}_{\mathrm{g},\ell}$ in Eq. (4.73):

$$p(\boldsymbol{\beta}_{\mathrm{g},\ell}) = \prod_{k}^{n_\mathrm{m}} \Gamma(c)^{-1} d^c [\boldsymbol{\beta}_{\mathrm{g},\ell}]_k^{c-1} \exp(-d\ [\boldsymbol{\beta}_{\mathrm{g},\ell}]_k), \tag{4.78}$$

except for the output layer as presented in Eq. (4.72), for which the prior information for the precision of the modeling errors $\boldsymbol{\varepsilon}_{\mathrm{y},\ell}$ is given by:

$$p(\boldsymbol{\beta}_{\mathrm{y},\ell}) = \prod_{k}^{n_\mathrm{o}} \Gamma(c)^{-1} d^c [\boldsymbol{\beta}_{\mathrm{y},\ell}]_k^{c-1} \exp(-d\ [\boldsymbol{\beta}_{\mathrm{y},\ell}]_k). \tag{4.79}$$

In this work the Monte-Carlo approach [142] is chosen to sample the hyperprior distributions. The covariance of the zero-mean weight prior is then accordingly obtained from:

$$[\boldsymbol{\Sigma}_{\mathrm{g},\ell}]_{:,:,k} = ([\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k} \odot \mathbf{I})^{-1}. \tag{4.80}$$

With the obtained weight prior in each gate $\mathrm{g} \in \{\mathrm{f}, \mathrm{z}, \tilde{\mathrm{c}}, \mathrm{o}, \mathrm{y}\}$ one may forward propagate the prior information through the ARD-LSTM framework as described previously in Eq. (4.71)-Eq. (4.74).

As aforementioned in Section 4.1.1 and Section 4.1.2 the estimation of the weights given the observation data $\mathbf{y}_\ell$ is not straightforward in the nonlinear LSTM framework. A solution is introduced by separating the estimation into two smaller subproblems consisting of a (local) target estimation and the subsequent weight estimation. To do so, the alternative minimization principle is employed. In other words, first the hyperparameters are fixed in their most probable values [99] to estimate the (local) targets by means of gradient descent. Once the local targets for all gates for all time steps are estimated, then they are fixed and the hyperparameters in all gates $\mathrm{g} \in \{\mathrm{f}, \mathrm{z}, \tilde{\mathrm{c}}, \mathrm{o}, \mathrm{y}\}$ are found.

Let the hyperparameters $\boldsymbol{\alpha}_{\mathrm{g},\ell}$ and $\boldsymbol{\beta}_{\mathrm{g},\ell}$ for all gates and output $\mathrm{g} \in \{\mathrm{f}, \mathrm{z}, \tilde{\mathrm{c}}, \mathrm{o}, \mathrm{y}\}$ be fixed in their most probable values [99] (see Eq. (4.64)), then the goal is to estimate the (local) targets by means of gradient descent. Starting at the output layer ($\mathrm{g} = \mathrm{y}$) the joint posterior distribution $p(\mathbf{s}_{\mathrm{y},\ell}, \mathbf{w}_{\mathrm{y},\ell}|\mathbf{y}_\ell)$, between the local target $\mathbf{s}_{\mathrm{y},\ell}$ and weights $\mathbf{w}_{\mathrm{y},\ell}$ at time step ℓ is estimated by using the approximated Bayesian rule given data $\mathbf{y}_\ell$ as described in Section 4.1.2. Hence, for the joint posterior one obtains:

$$p(\mathbf{s}_{\mathrm{y},\ell}, \mathbf{w}_{\mathrm{y},\ell}|\mathbf{y}_\ell) \propto p(\mathbf{y}_\ell|\mathbf{s}_{\mathrm{y},\ell}, \mathbf{w}_{\mathrm{y},\ell}) p(\mathbf{s}_{\mathrm{y},\ell}|\mathbf{w}_{\mathrm{y},\ell}) p(\mathbf{w}_{\mathrm{y},\ell}). \tag{4.81}$$

The likelihood $p(\mathbf{y}_\ell|\mathbf{s}_{\mathrm{y},\ell}, \mathbf{w}_{\mathrm{y},\ell})$ is implicitly defined by the output layer given in Eq. (4.74) and is nonlinear. Following Eq. (4.43), one further has:

$$\pi(\mathbf{s}_{\mathrm{y},\ell}) := p(\mathbf{s}_{\mathrm{y},\ell}|\mathbf{y}_\ell) \propto p(\mathbf{y}_\ell|\mathbf{s}_{\mathrm{y},\ell}) p(\mathbf{s}_{\mathrm{y},\ell}|\mathbf{w}_{\mathrm{y},\ell}), \tag{4.82}$$

which is approximated by

$$\pi(\mathbf{s}_{\mathrm{y},\ell}) \approx \tilde{\pi}(\mathbf{s}_{\mathrm{y},\ell}) \sim \mathcal{N}(\bar{\mathbf{s}}_{\mathrm{y},\ell}, \mathbf{C}_{\mathbf{s}_{\mathrm{y},\ell}}), \tag{4.83}$$

in which $\bar{\mathbf{s}}_{\mathrm{y},\ell}$ is unknown. One may further define the loss:

$$\mathcal{L}^{\mathrm{s}}_{\mathrm{y},\ell} := \mathcal{L}^{\mathrm{s}}(\mathbf{s}_{\mathrm{y},\ell}) = \log\, p(\mathbf{y}_\ell|\mathbf{s}_{\mathrm{y},\ell}). \tag{4.84}$$

Following the derivation in Section 4.1.2, one can adapt Eq. (4.58) to the output layer of the LSTM framework, giving the estimation:

$$\bar{\mathbf{s}}^{e+1}_{\mathrm{y},\ell} = \bar{\mathbf{s}}^{e}_{\mathrm{y},\ell} - \rho_e \mathbb{E}_{\tilde{\pi}_e(\mathbf{s}_{\mathrm{y},\ell})} \nabla_{\boldsymbol{\mu}} \left(\mathcal{L}^{\mathrm{s}}(\mathbf{s}_{\mathrm{y},\ell})\right)|_{\boldsymbol{\mu}=\bar{\mathbf{s}}^{e}_{\mathrm{y},\ell}}. \tag{4.85}$$

One may estimate $\boldsymbol{V}_e := \mathbb{E}_{\tilde{\pi}_e(\mathbf{s}_{\mathrm{y},\ell})} \nabla_{\boldsymbol{\mu}} \left(\mathcal{L}^{\mathrm{s}}(\mathbf{s}_{\mathrm{y},\ell})\right)|_{\boldsymbol{\mu}=\bar{\mathbf{s}}^{e}}$ using the first order approximation of the likelihood function (see Eq. (4.88)):

$$\mathcal{L}^{\mathrm{s}}(\mathbf{s}_{\mathrm{y},\ell}) = \mathcal{L}^{\mathrm{s}}(\bar{\mathbf{s}}^{e}_{\mathrm{y},\ell}) + \boldsymbol{V}_e(\mathbf{s}_{\mathrm{y},\ell} - \bar{\mathbf{s}}^{e}_{\mathrm{y},\ell}) + \varepsilon_{\mathcal{L}}, = \boldsymbol{V}_e \mathbf{s}_{\mathrm{y},\ell} + \boldsymbol{b}_{V,e} + \varepsilon_{\mathcal{L}}. \tag{4.86}$$

As a result of the recurrent state description in the LSTM cell the prediction of the local state in the output layer at every time step ℓ depends on the previous time steps. Hence, starting at the output layer all hyperparameters $\boldsymbol{\alpha}_{\mathrm{g},\ell}$ and $\boldsymbol{\beta}_{\mathrm{g},\ell}$ for $\mathrm{g} \in \{\mathrm{f}, \mathrm{z}, \tilde{\mathrm{c}}, \mathrm{o}, \mathrm{y}\}$, in the current time step depend on those in the previous time steps, i.e. $\boldsymbol{\alpha}_{\mathrm{g},\ell-m}$ and $\boldsymbol{\beta}_{\mathrm{g},\ell-m}$ with $m \geq 1$ and $m < n_{\mathrm{t}}$. The latter gets propagated through the hidden state $\mathbf{h}_\ell$, see Eq. (4.71). Following the LSTM architecture (see Eq. (4.71) and Fig. 3.4) one may note that the hidden state $\mathbf{h}_\ell$ is described by a nonlinear combination of the local targets $\mathbf{s}_{\mathrm{g},\ell}$ of all LSTM gates at time step ℓ (and their corresponding weights). Therefore, the weight dependency on the hidden state $\mathbf{h}_\ell$ and thus on the local targets $\mathbf{s}_{\mathrm{g},\ell}$ of the LSTM gates must be included as additional parameters in the ARD estimate [99].

Given the estimated local target $\bar{\mathbf{s}}_{\mathrm{y},\ell}$ (see Eq. (4.85)), the loss function for the hyperparameters $\boldsymbol{\alpha}_{\mathrm{y},\ell}$ and $\boldsymbol{\beta}_{\mathrm{y},\ell}$ in the output layer ($\mathrm{g} = \mathrm{y}$) reads:

$$\mathcal{L}^{\alpha,\beta}_{\mathrm{y},\ell} := \mathcal{L}^{\alpha,\beta}(\bar{\mathbf{s}}_{\mathrm{y},\ell}) = \log\, p(\bar{\mathbf{s}}_{\mathrm{y},\ell}|\boldsymbol{\alpha}_{\mathrm{y},\ell}, \boldsymbol{\beta}_{\mathrm{y},\ell}) \tag{4.87}$$

and is shaped by the virtual observation error $\boldsymbol{\epsilon}_{\mathrm{y},\ell}$. Following Eq. (4.31) the previous equation can be rewritten as [99]:

$$[\mathcal{L}^{\alpha,\beta}_{\mathrm{y},\ell}]_k = -\frac{1}{2}\left[\log\,|[\mathbf{C}_{\mathrm{y},\ell}]_{:,:,k}| + [\bar{\mathbf{s}}_{\mathrm{y},\ell}]^T_{:,k}[\mathbf{C}_{\mathrm{y},\ell}]^{-1}_{:,:,k}[\bar{\mathbf{s}}_{\mathrm{y},\ell}]_{:,k}\right], \tag{4.88}$$

where $[\mathbf{C}_{\mathrm{y},\ell}]_{:,:,k} := [\boldsymbol{\beta}_{\mathrm{y},\ell}]^{-1}_k \mathbf{I} + \boldsymbol{\Phi}_\ell([\boldsymbol{\alpha}_{\mathrm{y},\ell}]_{:,k} \odot \mathbf{I})^{-1}\boldsymbol{\Phi}^T_\ell$ with the basis function $\boldsymbol{\Phi}_\ell := [1\ \mathbf{h}_\ell]$. Note that in the likelihood in Eq. (4.88) we have the basis function $\boldsymbol{\Phi}_\ell$ that depends on the hidden states $\mathbf{h}_\ell$, and hence here enters the dependency of the weight on all previous hyperparameters as well as the local targets $\mathbf{s}_{\mathrm{g},\ell}$ in the gates. In other words, the likelihood has to be maximized not only with respect to $\boldsymbol{\alpha}_{\mathrm{y},\ell}$ and $\boldsymbol{\beta}_{\mathrm{y},\ell}$ but also for the optimal value of the hidden state $\mathbf{h}_\ell$, i.e. the local targets in the gates. This again can be achieved by alternating minimization.

Given Eq. (4.88), one may fix the hyperparameters $\boldsymbol{\alpha}_{\mathrm{y},\ell}$ and $\boldsymbol{\beta}_{\mathrm{y},\ell}$. The goal is to maximize the marginal likelihood in Eq. (4.87) for the local targets $\mathbf{s}_{\mathrm{g},\ell}$ in the LSTM gates. As here one is not aiming at obtaining the distribution of the local targets in the gates but the point estimate $\overline{\mathbf{s}}_{\mathrm{g},\ell}$, one may again employ the gradient descent approach similarly to the one described in Section 4.1.2. The main issue in such a gradient descent algorithm is the estimation of the gradient itself due to the back-propagation through the cell.

For a single time step (i.e. $n_{\mathrm{t}} = 1$) the gradient of the marginal likelihood objective $\mathcal{L}_{\mathrm{y},\ell}^{\alpha,\beta}$ in Eq. (4.88) with respect to the local targets $\mathbf{s}_{\mathrm{g},\ell}$ in the LSTM gates $\mathrm{g} \in \{\mathrm{f}, \mathrm{z}, \tilde{\mathrm{c}}, \mathrm{o}\}$ is given by:

$$\frac{\partial \mathcal{L}_{\mathrm{y},\ell}^{\alpha,\beta}}{\partial \mathbf{s}_{\mathrm{g},\ell}} = \frac{\partial \mathcal{L}_{\mathrm{y},\ell}^{\alpha,\beta}}{\partial \mathbf{h}_{\ell}} \odot \frac{\partial \mathbf{h}_{\ell}}{\partial \mathbf{s}_{\mathrm{g},\ell}} \tag{4.89}$$

and is defined by the chain rule. The first term in Eq. (4.89) reads:

$$\frac{\partial \mathcal{L}_{\mathrm{y},\ell}^{\alpha,\beta}}{\partial \mathbf{h}_{\ell}} = \sum_{k=1}^{n_{\mathrm{o}}} \left[[\mathbf{C}_{\mathrm{y},\ell}]_{:,:,k}^{-1} [\overline{\mathbf{s}}_{\mathrm{y},\ell}]_{:,k} [\overline{\mathbf{s}}_{\mathrm{y},\ell}]_{:,k}^{T} [\mathbf{C}_{\mathrm{y},\ell}]_{:,:,k}^{-1} - [\mathbf{C}_{\mathrm{y},\ell}]_{:,:,k}^{-1} \right] \mathbf{h}_{\ell} ([\boldsymbol{\alpha}_{\mathrm{y},\ell}]_{:,k}^{\mathrm{h}} \odot \mathbf{I})^{-1} \tag{4.90}$$

with $[\mathbf{C}_{\mathrm{y},\ell}]_{:,:,k} := [\boldsymbol{\beta}_{\mathrm{y},\ell}]_{k}^{-1} \mathbf{I} + \mathbf{h}_{\ell} ([\boldsymbol{\alpha}_{\mathrm{y},\ell}]_{:,k}^{\mathrm{h}} \odot \mathbf{I})^{-1} \mathbf{h}_{\ell}^{T}$ and $[\boldsymbol{\alpha}_{\mathrm{y},\ell}]_{:,k}^{\mathrm{h}} := [[\boldsymbol{\alpha}_{\mathrm{y},\ell}]_{jk}]_{j=1}^{n_{\mathrm{m}}}$, whereas, the second term in Eq. (4.89) is nonlinear and arises from the LSTM cell architecture. In other words, following Eq. (3.46) one obtains

$$\begin{aligned}
\frac{\partial \mathbf{h}_{\ell}}{\partial \mathbf{s}_{\tilde{\mathrm{c}},\ell}} &:= \frac{\partial \mathbf{h}_{\ell}}{\partial \mathbf{c}_{\ell}} \odot \mathbf{z}_{\ell} \odot (1 - \tilde{\mathbf{c}}_{\ell}^{2}) \\
\frac{\partial \mathbf{h}_{\ell}}{\partial \mathbf{s}_{\mathrm{z},\ell}} &:= \frac{\partial \mathbf{h}_{\ell}}{\partial \mathbf{c}_{\ell}} \odot \tilde{\mathbf{c}}_{\ell} \odot \mathbf{z}_{\ell} \odot (1 - \mathbf{z}_{\ell}) \\
\frac{\partial \mathbf{h}_{\ell}}{\partial \mathbf{s}_{\mathrm{f},\ell}} &:= \frac{\partial \mathbf{h}_{\ell}}{\partial \mathbf{c}_{\ell}} \odot \mathbf{c}_{\ell-1} \odot \mathbf{f}_{\ell} \odot (1 - \mathbf{f}_{\ell}) \\
\frac{\partial \mathbf{h}_{\ell}}{\partial \mathbf{s}_{\mathrm{o},\ell}} &:= \tanh(\mathbf{c}_{\ell}) \odot \mathbf{o}_{\ell} \odot (1 - \mathbf{o}_{\ell})
\end{aligned} \tag{4.91}$$

in which

$$\frac{\partial \mathbf{h}_{\ell}}{\partial \mathbf{c}_{\ell}} := \mathbf{o}_{\ell} \odot \left(1 - \tanh^{2}(\mathbf{c}_{\ell})\right) + \frac{\partial \mathbf{h}_{\ell+1}}{\partial \mathbf{c}_{\ell+1}} \odot \mathbf{f}_{\ell+1}. \tag{4.92}$$

After back-propagation for one time step is completed one may estimate the local targets $\overline{\mathbf{s}}_{\mathrm{g},\ell}$ in the LSTM gates by:

$$\overline{\mathbf{s}}_{\mathrm{g},\ell}^{e+1} = \overline{\mathbf{s}}_{\mathrm{g},\ell}^{e} - \rho_{e} \mathbb{E} \left(\frac{\partial \mathcal{L}_{\mathrm{y},\ell}^{\alpha,\beta}}{\partial \mathbf{s}_{\mathrm{g},\ell-1}^{e}} c) \right) \Bigg|_{\boldsymbol{\mu} = \overline{\mathbf{s}}_{\mathrm{g},\ell}}. \tag{4.93}$$

Here ρ_e is the learning rate, yet for ARD-LSTM the Adam-optimizer [158] is applied for a more robust gradient-based estimation.

The previous ARD-backpropagation algorithm is derived for only one time cell. However, in practice (i.e. if $n_{\mathrm{t}} > 1$) we deal with several LSTM cells, each characterized

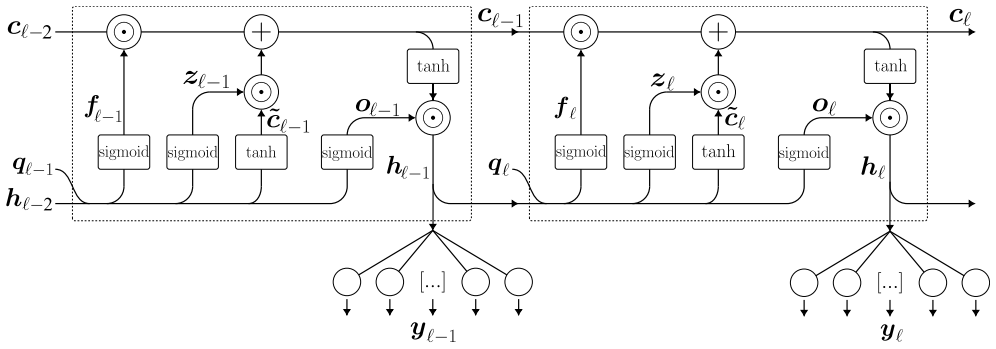


Figure 4.4: LSTM-based NN for two consecutive time steps $\ell - 1$ and ℓ.

4

with its own output, see Fig. 4.4. In Fig. 4.4 two consecutive time steps are depicted for an LSTM-based NN, hereby emphasizing the propagation of state information through the hidden state $\mathbf{h}_\ell$ and the cell state $\mathbf{c}_\ell$. Thus, the LSTM cell at this time moment has an output that depends on $\mathbf{h}_\ell$, which further depends on $\mathbf{s}_{\mathrm{g},\ell}$. However, $\mathbf{s}_{\mathrm{g},\ell}$ depends on the hidden state of the previous cell $\mathbf{h}_{\ell-1}$. This means that if we have estimated the mean for the local targets $\bar{\mathbf{s}}_{\mathrm{g},\ell}$ via Eq. (4.93), now we may use this estimate to get the estimate for the weights in the previous cell, and we can repeat this sequentially in a backward manner. For this purpose, one may further define the marginal likelihood for the LSTM gate weights as:

$$\mathcal{L}_{\mathrm{g},\ell}^{\alpha,\beta} := \mathcal{L}^{\alpha,\beta}(\bar{\mathbf{s}}_{\mathrm{g},\ell}) = \log\, p(\bar{\mathbf{s}}_{\mathrm{g},\ell}|\boldsymbol{\alpha}_{\mathrm{g},\ell}, \boldsymbol{\beta}_{\mathrm{g},\ell}), \tag{4.94}$$

in which $\bar{\mathbf{s}}_{\mathrm{g},\ell}$ is assumed to be known from Eq. (4.93) and thus already estimated. In a similar manner as for the estimation of the output local state in Eq. (4.87) the likelihood in Eq. (4.94) is characterized by a virtual measurement noise $\boldsymbol{\epsilon}_{\mathrm{g},\ell}$. The previous equation then leads to the objective function of the marginal likelihood in the LSTM gates [99]:

$$[\mathcal{L}_{\mathrm{g},\ell}^{\alpha,\beta}]_k = -\frac{1}{2}\left[\log\,|[\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}| + [\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k}^T [\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}^{-1} [\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k}\right], \tag{4.95}$$

where $[\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k} := [\boldsymbol{\beta}_{\mathrm{g},\ell}]_k^{-1}\mathbf{I} + \boldsymbol{\Phi}_\ell([\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k} \odot \mathbf{I})^{-1}\boldsymbol{\Phi}_\ell^T$ with the basis function $\boldsymbol{\Phi}_\ell := [1\ \mathbf{q}\ \mathbf{h}_{\ell-1}]$.

At time step ℓ the hyperparameters in all gates $\mathrm{g} \in \{\mathrm{f}, \mathrm{z}, \tilde{\mathrm{c}}, \mathrm{o}\}$ rely on the hidden state $\mathbf{h}_{\ell-1}$. Furthermore, at time $\ell - 1$ the hyperparameters in the output layer ($\mathrm{g} = \mathrm{y}$) also relies on $\mathbf{h}_{\ell-1}$, see Fig. 4.4. Hence, in back-propagation the hidden state $\mathbf{h}_{\ell-1}$ (and thus the local target $\mathbf{s}_{\mathrm{g},\ell-1}$) relies on both likelihoods $\mathcal{L}_{\mathrm{y},\ell-1}^{\alpha,\beta}$ and $\mathcal{L}_{\mathrm{g},\ell}^{\alpha,\beta}$, which are assumed to be independent due to the independent virtual measurement noise $\boldsymbol{\epsilon}_{\mathrm{g},\ell}$ in the gates $\mathrm{g} \in \{\mathrm{f}, \mathrm{z}, \tilde{\mathrm{c}}, \mathrm{o}\}$ and output ($\mathrm{g} = \mathrm{y}$). Due to the assumed independent likelihoods one may express the aforementioned reliance of the local target $\mathbf{s}_{\mathrm{g},\ell-1}$ using the product of both likelihoods. However, as both $\mathcal{L}_{\mathrm{y},\ell-1}^{\alpha,\beta}$ and $\mathcal{L}_{\mathrm{g},\ell}^{\alpha,\beta}$ are expressed in logarithmic scale the product of both likelihoods becomes the sum. Thus, using Eq. (4.89) and following the classical LSTM back-propagation scheme in Eq. (3.46)

for multiple time steps one may define:

$$\mathbf{D}_{\mathbf{s}_{\mathrm{g},\ell-1}} := \left(\frac{\partial \mathcal{L}^{\alpha,\beta}_{\mathrm{g},\ell}}{\partial \mathbf{h}_{\ell-1}} + \frac{\partial \mathcal{L}^{\alpha,\beta}_{\mathrm{y},\ell-1}}{\partial \mathbf{h}_{\ell-1}} \right) \odot \frac{\partial \mathbf{h}_{\ell-1}}{\partial \mathbf{s}_{\mathrm{g},\ell-1}}. \tag{4.96}$$

The first term in the previous equation reads:

$$\frac{\partial \mathcal{L}^{\alpha,\beta}_{\mathrm{g},\ell}}{\partial \mathbf{h}_{\ell-1}} = \sum_{k=1}^{n_\mathrm{m}} \Big[[\mathbf{C}_{\mathrm{g},\ell}]^{-1}_{:,:,k} [\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k} [\bar{\mathbf{s}}_{\mathrm{g},\ell}]^{T}_{:,k} [\mathbf{C}_{\mathrm{g},\ell}]^{-1}_{:,:,k} - [\mathbf{C}_{\mathrm{g},\ell}]^{-1}_{:,:,k} \Big] \mathbf{h}_{\ell-1} ([\boldsymbol{\alpha}_{\mathrm{g},\ell}]^{\mathrm{h}}_{:,k} \odot \mathbf{I})^{-1} \tag{4.97}$$

with $[\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k} := [\boldsymbol{\beta}_{\mathrm{g},\ell}]^{-1}_{k} \mathbf{I} + \mathbf{h}_{\ell-1} ([\boldsymbol{\alpha}_{\mathrm{g},\ell}]^{\mathrm{h}}_{:,k} \odot \mathbf{I})^{-1} \mathbf{h}^{T}_{\ell-1}$ and the hyperparameter $[\boldsymbol{\alpha}_{\mathrm{g},\ell}]^{\mathrm{h}}_{:,k}$ $:= [[\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{jk}]^{n_\mathrm{i}}_{j=n_\mathrm{i}-n_\mathrm{m}}$ where $n_\mathrm{i} = 1 + n_\mathrm{f} + n_\mathrm{m}$. Accordingly, before continuing back-propagation to the next time step, in every gate $\mathrm{g} \in \{\mathrm{f}, \mathrm{z}, \tilde{\mathrm{c}}, \mathrm{o}\}$ the local target is then first estimated using:

$$\bar{\mathbf{s}}^{e+1}_{\mathrm{g},\ell-1} = \bar{\mathbf{s}}^{e}_{\mathrm{g},\ell-1} - \rho_e \mathbb{E}\left(\mathbf{D}_{\mathbf{s}_{\mathrm{g},\ell-1}} \right) \big|_{\boldsymbol{\mu} = \bar{\mathbf{s}}_{\mathrm{g},\ell-1}}. \tag{4.98}$$

After updating the local targets at all time steps for the current iteration e the hyperparameters $\boldsymbol{\alpha}_{\mathrm{g},\ell}$ and $\boldsymbol{\beta}_{\mathrm{g},\ell}$ can be updated.

Fixing the local targets (and $\mathbf{h}_\ell$) and using gradient-based optimization to minimize Eq. (4.94) the hyperparameters $[\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k}$ and $[\boldsymbol{\beta}_{\mathrm{g},\ell}]_k$ then gives:

$$\begin{aligned} [\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{jk} &= \frac{1}{[\boldsymbol{\mu}_{\mathrm{g},\ell}]^2_{jk} + d_{jk}}, \qquad j = 1, ..., n_\mathrm{i}; k = 1, ..., n_\mathrm{m}; \\ [\boldsymbol{\beta}_{\mathrm{g},\ell}]_k &= \frac{n_\mathrm{d} - \sum_j [\boldsymbol{\gamma}_{\mathrm{g},\ell}]_{jk}}{\| [\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k} - \boldsymbol{\Phi}_\ell [\boldsymbol{\mu}_{\mathrm{g},\ell}]_{:,k} \|^2} \end{aligned} \tag{4.99}$$

with d_{jk} being the jth diagonal element of $[\boldsymbol{\Sigma}_{\mathrm{g},\ell}]_{:,:,k}$. The prior dominance indicator $[\boldsymbol{\gamma}_{\mathrm{g},\ell}]_{jk}$ is defined per time step as:

$$[\boldsymbol{\gamma}_{\mathrm{g},\ell}]_{jk} = 1 - [\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{jk} d_{jk}. \tag{4.100}$$

In a similar sense, for the (observable) output layer ($\mathrm{g} = \mathrm{y}$) using Eq. (4.87) one obtains:

$$\begin{aligned} [\boldsymbol{\alpha}_{\mathrm{y},\ell}]_{jk} &= \frac{1}{[\boldsymbol{\mu}_{\mathrm{y},\ell}]^2_{jk} + d_{jk}}, \qquad j = 0, ..., n_\mathrm{m}; k = 1, ..., n_\mathrm{o}; \\ [\boldsymbol{\beta}_{\mathrm{y},\ell}]_k &= \frac{n_\mathrm{d} - \sum_j [\boldsymbol{\gamma}_{\mathrm{y},\ell}]_{jk}}{\| [\bar{\mathbf{s}}_{\mathrm{y},\ell}]_{:,k} - \boldsymbol{\Phi}_\ell [\boldsymbol{\mu}_{\mathrm{y},\ell}]_{:,k} \|^2} \end{aligned} \tag{4.101}$$

with

$$[\boldsymbol{\gamma}_{\mathrm{y},\ell}]_{jk} = 1 - [\boldsymbol{\alpha}_{\mathrm{y},\ell}]_{jk} d_{jk}. \tag{4.102}$$

Having completed the local target estimation and hyperparameter updates for all time steps back-propagation is completed for the current iteration. The overall process of forward and backward propagation with accompanying alternative local target and

hyperparameter estimation through all time steps is repeated until the convergence criteria

$$\left| \sum_{\ell=1}^{n_t} \sum_{k=1}^{n_o} [\mathcal{L}_{y,\ell}^{\alpha,\beta,e-20}]_k - [\mathcal{L}_{y,\ell}^{\alpha,\beta,e}]_k \right| \leq 2e{-}2 \tag{4.103}$$

has been met in two successive iterations of the local target state and hyperparameter estimation. An overview of the iterative procedure for the LSTM cell has been set out in Algorithm 1.

Algorithm 1 ARD-LSTM optimization.

Require: Normalize input
Require: Prior weight: $p([\mathbf{w}_{g,\ell}]_{:,k}|[\boldsymbol{\alpha}_{g,\ell}]_{:,k}) \sim \mathcal{N}(0, [\boldsymbol{\alpha}_{g,\ell}]_{:,k}^{-1})$
Require: Hyperprior parameter: $[\boldsymbol{\alpha}_{g,\ell}]_{:,k} \leftarrow \text{Gamma}_{g,\ell}([\boldsymbol{\alpha}_{g,\ell}]_{:,k}|0,0)$
Require: Likelihood hyperprior: $[\boldsymbol{\beta}_{g,\ell}]_k \leftarrow \text{Gamma}_{g,\ell}([\boldsymbol{\beta}_{g,\ell}]_k|0,0)$
Require: $\boldsymbol{h}_0 = \mathbf{0}$, $\boldsymbol{c}_0 = \mathbf{0}$ at $\ell = 0$, $e = 0$ ▷ State initialization

While $\mathcal{L}_{g,\ell}^{\alpha,\beta}$ is not converged: ▷ Loops through epochs
  Forward propagation
  $[\mathbf{s}_{g,\ell}]_{:,k} = \boldsymbol{\Phi}_{g,\ell}[\mathbf{w}_{g,\ell}]_{:,k} + [\boldsymbol{\epsilon}_{g,\ell}]_{:,k}, \quad \boldsymbol{\Phi}_{g,\ell} := [\mathbf{1}\ \mathbf{q}\ \mathbb{E}(\mathbf{h}_{\ell-1})]$
  Back-propagation
  Estimate $\mathbb{E}_{\tilde{\pi}_e([\mathbf{s}_{g,\ell}]_{:,k})} \nabla_{\boldsymbol{\mu}} \left(\mathcal{L}^s([\mathbf{s}_{g,\ell}]_{:,k}) \right) |_{\mu=[\bar{\mathbf{s}}_{g,\ell}^e]_{:,k}}$ in the output layer.
  Update output local target state:
  $[\bar{\mathbf{s}}_{g,\ell}^{e+1}]_{:,k} = [\bar{\mathbf{s}}_{g,\ell}^e]_{:,k} - \rho_e \mathbb{E}_{\tilde{\pi}_e([\mathbf{s}_{g,\ell}]_{:,k})} \nabla_{\boldsymbol{\mu}} \left(\mathcal{L}^s([\mathbf{s}_{g,\ell}]_{:,k}) \right) |_{\boldsymbol{\mu}=[\bar{\mathbf{s}}_{g,\ell}^e]_{:,k}}$
  Estimate $\dfrac{\partial \mathcal{L}_{y,\ell}^{\alpha,\beta}}{\partial \mathbf{h}_\ell}$ and $\dfrac{\partial \mathcal{L}_{g,\ell}^{\alpha,\beta}}{\partial \mathbf{h}_\ell}$ in the LSTM cell to propagate through time.
  Update LSTM gate local target state:
  $[\bar{\mathbf{s}}_{g,\ell}^{e+1}]_{:,k} = [\bar{\mathbf{s}}_{g,\ell}^e]_{:,k} - \rho_e \mathbb{E} \left([\mathbf{D}_{\mathbf{s}_{g,\ell}}]_{:,k} \right)_{\boldsymbol{\mu}=[\bar{\mathbf{s}}_{g,\ell}^e]_{:,k}}$
  While weight ARD is not converged: ▷ ARD weight update
    Update $[\boldsymbol{\alpha}_{g,\ell}]_{:,k}$ and $[\boldsymbol{\beta}_{g,\ell}]_k$
    Estimate moments of the posterior weight:
    $[\boldsymbol{\Sigma}_{g,\ell}]_{:,:,k} \leftarrow \mathcal{N}([\mathbf{w}_{g,\ell}]_{:,k}|[\bar{\mathbf{s}}_{g,\ell}]_{:,k}, [\boldsymbol{\alpha}_{g,\ell}]_{:,k}, [\boldsymbol{\beta}_{g,\ell}]_k)$
    $[\boldsymbol{\mu}_{g,\ell}]_{:,k} \leftarrow \mathcal{N}([\mathbf{w}_{g,\ell}]_{:,k}|[\bar{\mathbf{s}}_{g,\ell}]_{:,k}, [\boldsymbol{\alpha}_{g,\ell}]_{:,k}, [\boldsymbol{\beta}_{g,\ell}]_k)$
  return $[\boldsymbol{\mu}_{g,\ell}]_{:,k}, [\boldsymbol{\Sigma}_{g,\ell}]_{:,:,k}$
  Set the next iteration step $e = e + 1$
return $[\boldsymbol{\mu}_{g,\ell}]_{:,k}, [\boldsymbol{\Sigma}_{g,\ell}]_{:,:,k}, [\mathbf{s}_{g,\ell}]_{:,k}$

4.2.2 Acceleration of the ARD-LSTM algorithm

When dissecting Eq. (4.88) one may note the computationally expensive inversion of $[\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}$. However, Eq. (4.97) can be conveniently rewritten [99] into:

$$\frac{\partial \mathcal{L}^{\alpha,\beta}_{\mathrm{g},\ell}}{\partial \mathbf{h}_{\ell-1}} = \sum_{k}^{n_{\mathrm{m}}} [\boldsymbol{\beta}_{\mathrm{g},\ell}]_k \left[\left([\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k} - \mathbf{h}_{\ell-1} [\boldsymbol{\mu}_{\mathrm{g},\ell}]^{\mathrm{h}}_{:,k} \right) \left([\boldsymbol{\mu}_{\mathrm{g},\ell}]^{\mathrm{h}}_{:,k} \right)^T - \mathbf{h}_{\ell-1} [\boldsymbol{\Sigma}_{\mathrm{g},\ell}]^{\mathrm{h}}_{:,:,k} \right], \quad (4.104)$$

which hereby prevents the inversion of $[\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}$. Here, the weights $[\boldsymbol{\mu}_{\mathrm{g},\ell}]^{\mathrm{h}}_{:,k} := [\boldsymbol{\mu}_{\mathrm{g},\ell}]_{j=n_{\mathrm{i}}-n_{\mathrm{f}},\ldots,n_{\mathrm{i}},k}$ and covariance $[\boldsymbol{\Sigma}_{\mathrm{g},\ell}]^{\mathrm{h}}_{:,:,k} := [\boldsymbol{\Sigma}_{\mathrm{g},\ell}]_{j=n_{\mathrm{i}}-n_{\mathrm{f}},\ldots,n_{\mathrm{i}},l=n_{\mathrm{i}}-n_{\mathrm{f}},\ldots,n_{\mathrm{i}},k}$ with superscript h denoting the part of the weight matrix that propagates information of the hidden state. Similarly, one can find the gradient of the output layer ($\mathrm{g} = \mathrm{y}$).

By omitting the inverse of $[\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}$, one significantly reduces the computational effort involved. However, one may note that $[\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}$ and its inverse must still be computed to obtain the log likelihood objective itself, see Eq. (4.88). Yet, this is only partly true, as Eq. (4.88) may be further altered. Having:

$$|[\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k} \odot \mathbf{I}| \, |[\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}| = |[\boldsymbol{\beta}_{\mathrm{g},\ell}]_k^{-1} \mathbf{I}| |[\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k} \odot \mathbf{I} + [\boldsymbol{\beta}_{\mathrm{g},\ell}]_k \boldsymbol{\Phi}_\ell^T \boldsymbol{\Phi}_\ell|, \quad (4.105)$$

as shown in [99], one can rewrite the determinant to

$$\begin{aligned} \log |[\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}| &= \log |[\boldsymbol{\beta}_{\mathrm{g},\ell}]_k^{-1} \mathbf{I} + \boldsymbol{\Phi}_\ell ([\boldsymbol{\alpha}_{\mathrm{g},\ell}]^{-1}_{:,k} \odot \mathbf{I}) \boldsymbol{\Phi}_\ell^T| \\ &= -\log |[\boldsymbol{\Sigma}_{\mathrm{g},\ell}]_{:,:,k}| - n_{\mathrm{d}} \log [\boldsymbol{\beta}_{\mathrm{g},\ell}]_k - \log |[\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k} \odot \mathbf{I}| \end{aligned} \quad (4.106)$$

the terms in which are also known as the Ockham (or Occam) factors [99, 184]. In addition, from the Woodbury inversion identity [99, 185], one obtains:

$$\begin{aligned} [\bar{\mathbf{s}}_{\mathrm{g},\ell}]^T_{:,k} [\mathbf{C}_{\mathrm{g},\ell}]^{-1}_{:,:,k} [\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k} &= [\bar{\mathbf{s}}_{\mathrm{g},\ell}]^T_{:,k} \left([\boldsymbol{\beta}_{\mathrm{g},\ell}]_k^{-1} \mathbf{I} + \boldsymbol{\Phi}_\ell ([\boldsymbol{\alpha}_{\mathrm{g},\ell}]^{-1}_{:,k} \odot \mathbf{I}) \boldsymbol{\Phi}_\ell^T \right)^{-1} [\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k} \\ &= [\boldsymbol{\beta}_{\mathrm{g},\ell}]_k [\bar{\mathbf{s}}_{\mathrm{g},\ell}]^T_{:,k} [\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k} - [\boldsymbol{\beta}_{\mathrm{g},\ell}]_k^2 [\bar{\mathbf{s}}_{\mathrm{g},\ell}]^T_{:,k} \boldsymbol{\Phi}_\ell [\boldsymbol{\Sigma}_{\mathrm{g},\ell}]_{:,:,k} \boldsymbol{\Phi}_\ell^T [\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k} \\ &= [\boldsymbol{\beta}_{\mathrm{g},\ell}]_k [\bar{\mathbf{s}}_{\mathrm{g},\ell}]^T_{:,k} \left([\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k} - \boldsymbol{\Phi}_\ell [\boldsymbol{\mu}_{\mathrm{g},\ell}]_{:,k} \right) \\ &= [\boldsymbol{\beta}_{\mathrm{g},\ell}]_k \left\| [\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k} - \boldsymbol{\Phi}_\ell [\boldsymbol{\mu}_{\mathrm{g},\ell}]_{:,k} \right\|^2 + [\boldsymbol{\mu}_{\mathrm{g},\ell}]^T_{:,k} \left([\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k} \odot \mathbf{I} \right) [\boldsymbol{\mu}_{\mathrm{g},\ell}]_{:,k} \end{aligned} \quad (4.107)$$

giving the log-likelihood function:

$$\begin{aligned} [\boldsymbol{\mathcal{L}}^{\alpha,\beta}_{\mathrm{g},\ell}]_k = &- \frac{1}{2} [-\log |[\boldsymbol{\Sigma}_{\mathrm{g},\ell}]_{:,:,k}| - n_{\mathrm{d}} \log [\boldsymbol{\beta}_{\mathrm{g},\ell}]_k - \log |[\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k} \odot \mathbf{I}| \\ &+ [\boldsymbol{\beta}_{\mathrm{g},\ell}]_k \| [\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k} - [\boldsymbol{\mu}_{\mathrm{g},\ell}]_{:,k} \boldsymbol{\Phi}_\ell \|^2 + [\boldsymbol{\mu}_{\mathrm{g},\ell}]^T_{:,k} ([\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k} \odot \mathbf{I}) [\boldsymbol{\mu}_{\mathrm{g},\ell}]_{:,k}], \end{aligned} \quad (4.108)$$

with a posterior-mean penalty described by Eq. (4.107). Maximization of the previous equation is equal to the minimization of its negative, referred to as the negative log-likelihood error.

It should be noted that values in $[\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k}$ may tend towards infinity to describe a low prior variance (and be limited to the set maximum). As a result, its determinant $|[\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k} \odot \mathbf{I}|$ may quickly reach infinity. Although this does not affect the optimization procedure, it strongly affects the likelihood objective metric and an imposed

maximum for numerical stability may be useful. In addition, as the local target function $[\mathbf{s}_{\mathrm{g},\ell}]_{:,k}$ is unknown in advance, the first iterative steps may have a poorly defined covariance matrix due to a strongly fluctuating sparsity percentage leading to numerical instabilities in the Cholesky decomposition for $\log|[\boldsymbol{\Sigma}_{\mathrm{g},\ell}]_{:,:,k}|$. Therefore, in this thesis the original formulation as given in Eq. (4.97) is applied to stimulate numerical stability.

4.3 Numerical results

To evaluate the numerical capability of the proposed framework, a purely numerical verification test is performed on one neuron example with a nonlinear activation function. The neuron is first modeled by known weights, and then the data are collected by sampling. In other words given:

$$\hat{\boldsymbol{y}} = \boldsymbol{\sigma}(\boldsymbol{s}), \quad \boldsymbol{s} := \boldsymbol{\Phi}\mathbf{w}, \tag{4.109}$$

where the basis $\boldsymbol{\Phi} := [\boldsymbol{q}]$ is made of a varying input signal $\boldsymbol{q}$, whereas the activation function $\sigma(\cdot)$ is taken to be a sigmoid function. A true weight set $\mathbf{w} \in \mathbb{R}^{1\times 100}$ is defined, the entries of which are created by sampling from a standard normal distribution $\mathcal{N}(0,1)$ with a pre-determined sparsity percentage. The input data is collected in matrix $\boldsymbol{\Phi} := [\mathbf{q}] \in \mathbb{R}^{n_\mathrm{d}\times n_\mathrm{f}}$ in which input signal $\mathbf{q} := [..., \boldsymbol{q}_i, ...], i = 1, ..., n_\mathrm{d}$ is sampled from a uniform distribution $\mathcal{U}(0,1)$, The weights are kept fixed whereas the input $\mathbf{q}$ is varied, and hence $\mathbf{y} := [..., \boldsymbol{y}_i, ...], i = 1, ..., n_\mathrm{d}$ with $\mathbf{y} \in \mathbb{R}^{n_\mathrm{d}\times 100}$ is collected for the n_d samples taken from the input. Additionally, the model is assumed perfect, meaning that measurement noise is not taken into account.

Given the gathered data $(\mathbf{q}, \mathbf{y})$ the goal is to estimate the local target and the weights as presented in Eq. (4.109) in the aforementioned alternating form. With $\boldsymbol{\Phi}$ and $\mathbf{y}$ known, both the local target $\mathbf{s}$ and weights $\mathbf{w}$ are assumed unknown and estimated as described in Section 4.2.1. To analyze the numerical convergence of the algorithm to the true values of $\mathbf{w}$, a default hyperparameter setting is taken as shown in Table 4.1. The study is undergone by varying each of the hyperparameters. In the iteration process the pruning is applied for the elements of $\boldsymbol{\gamma}$ being $\boldsymbol{\gamma} \leq 1\mathrm{e}{-8}$. This less strict pruning threshold (compared to the proposal in [99]) has been set to conform to the imposed constraints on $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$, which disallows $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ to go to the infinity. Note that this threshold still ensures that the posterior covariance of the weights can still be strongly dominated by the prior over the likelihood.

Table 4.1: Default parameter setting for numerical validation.

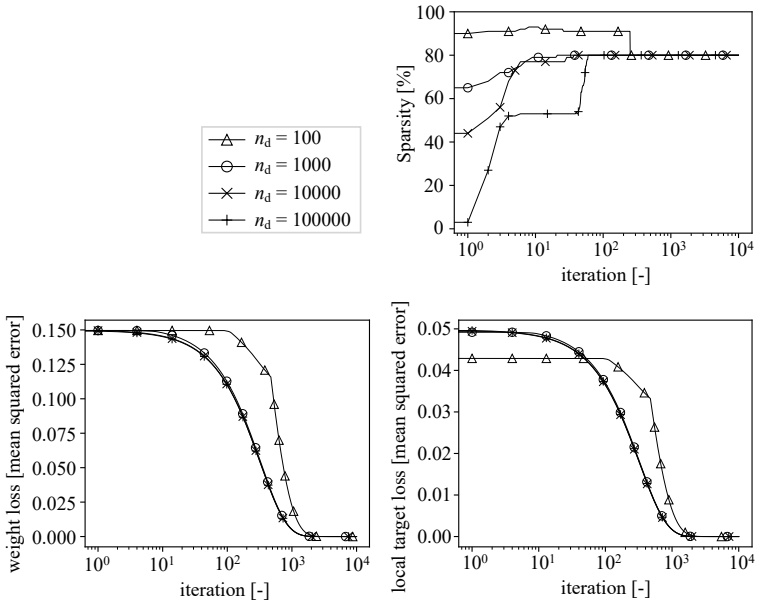
Parameter	n_d	λ	$\boldsymbol{\beta}$	$\boldsymbol{\alpha}$	sparsity
Default value (range)	10 000	0.001	[1e1, 1e6]	[1e1, 1e12]	80%

The convergence is measured in the mean squared sense by computing the mean squared error between the assimilated weights and the true weights. Also, the sparsity of the posterior weights is kept track of. In a similar manner, the convergence of the

local target is monitored by estimating the mean squared error between the posterior local target and the exact local target that is obtained by inverting the activation function given the true response, i.e. $\mathbf{s} := \boldsymbol{\sigma}^{-1}(\mathbf{y})$.

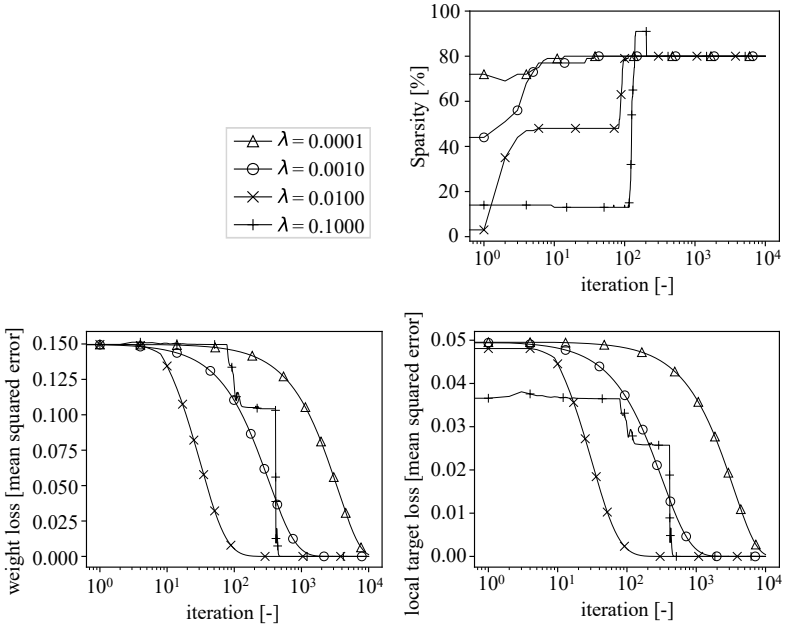
In Fig. 4.5a one may see the convergence (rate) as a function of the number of data points (i.e. samples) $n_\mathrm{d} \in \{100, 1000, 10\,000, 100\,000\}$ taken for assimilation purpose. As depicted the algorithm is converging faster to the desired solution with the increase of the number of samples n_d, as expected. At the same time the algorithm is reaching the true value sparsity percentage for each of the data set size, even for the small one ($n_\mathrm{d} = 100$) that corresponds to the number of terms in the weight vector.

To analyze the influence of the learning parameter on the assimilation accuracy, its values are varied for $\lambda \in \{0.0001, 0.001, 0.01, 0.1\}$, see Fig. 4.5b and Eq. (3.26). Given $\lambda = 0.1$ the system continuously over-estimates the weight updates, due to which fluctuations in the convergence are observed. When reducing the learning rate to a more sensible value $\lambda = 0.001$ robust convergence is observed. Further decreasing the learning rate still leads to convergence, but requires an increasing number of iterations. A learning rate of $\lambda = 0.001$ corresponds to the proposed default in the adaptive moment scheme [158].

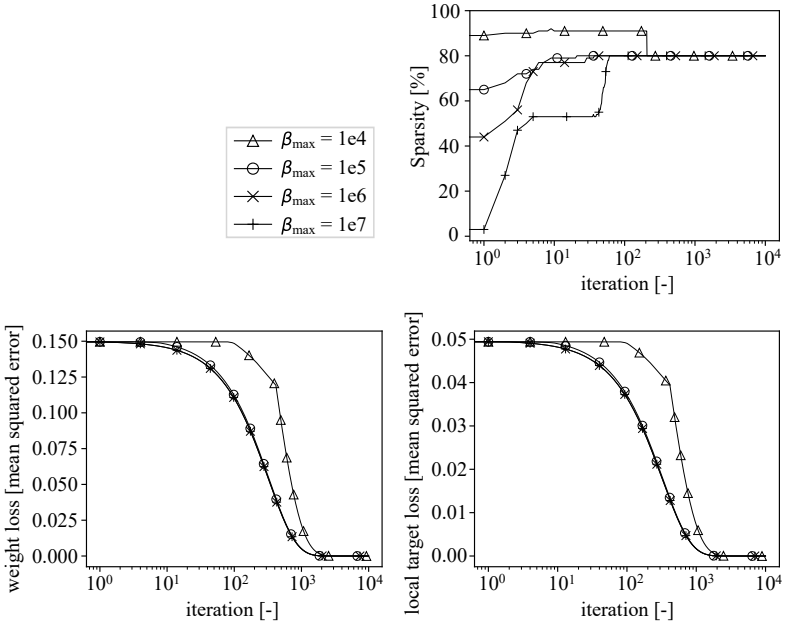


(a) Convergence for varying numbers of data samples n_d.

Figure 4.5: Convergence assessment on the true value weight sparsity, true value weight loss and local target loss of the algorithm in a single-layer system. *(cont.)*

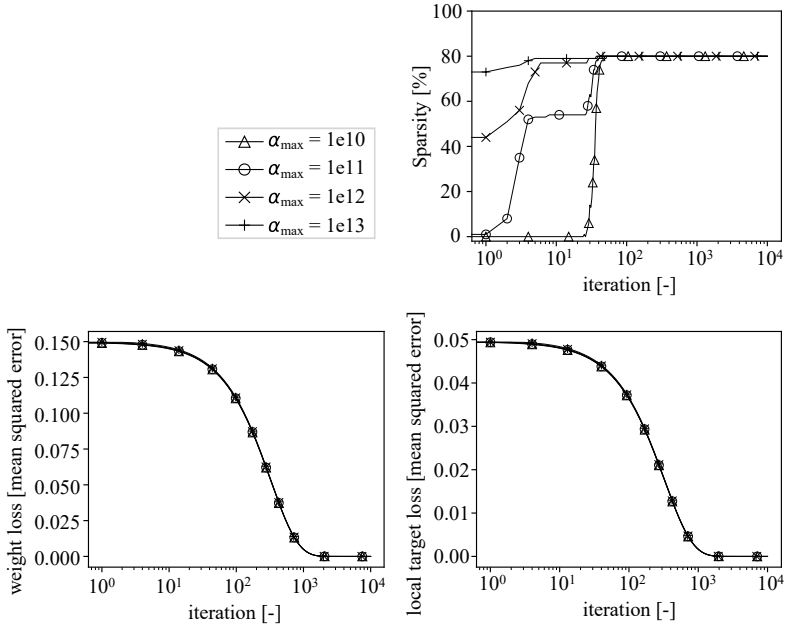


(b) Convergence for varying learning rates λ.

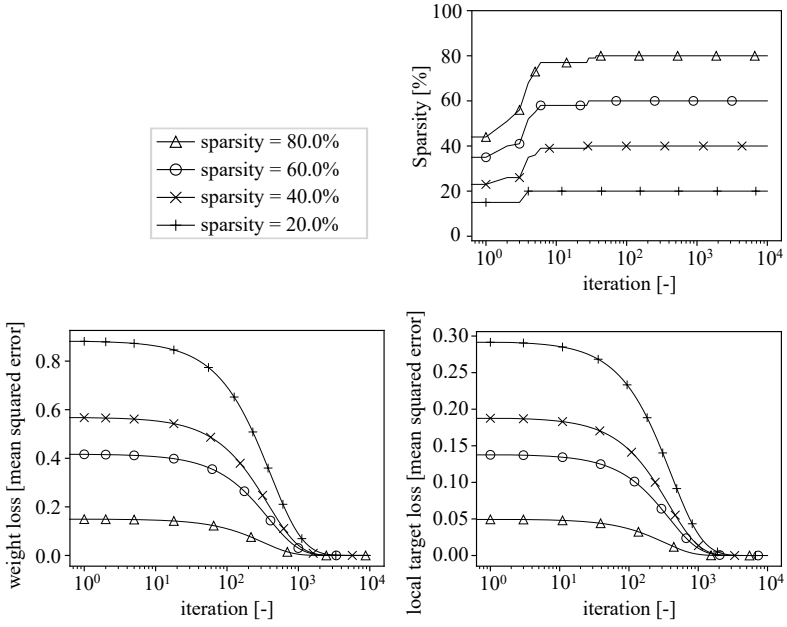


(c) Convergence for varying upper range limits for $\boldsymbol{\beta}$.

Figure 4.5: Convergence assessment on the true value weight sparsity, true value weight loss and local target loss of the algorithm in a single-layer system. *(cont.)*



(d) Convergence for varying upper range limits for $\boldsymbol{\alpha}$.



(e) Convergence for varying sparsity percentages on the true value weights.

Figure 4.5: Convergence assessment on the true value weight sparsity, true value weight loss and local target loss of the algorithm in a single-layer system.

In a similar fashion, the upper range limit of the hyperparameters used to model

the weights ($\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$) are altered to $\boldsymbol{\beta}_{\max} \in \{1e4, 1e5, 1e6, 1e7\}$ (see Fig. 4.5c) and $\boldsymbol{\alpha}_{\max} \in \{1e10, 1e11, 1e12, 1e13\}$ (see Fig. 4.5d), respectively. Robust convergence is observed for all $\boldsymbol{\beta}_{\max}$ and $\boldsymbol{\alpha}_{\max}$ in the given ranges, where a smaller value of $\boldsymbol{\beta}_{\max}$ causes a decrease in the rate of convergence. A larger $\boldsymbol{\beta}$ means assuming less noise in the data, and hence a more precise measurement, which is considered in this text. In such case the posterior estimate moves towards the measurement and away from the prior. Thus, the prior has a smaller influence on the posterior estimate. An increased maximum range limit of $\boldsymbol{\alpha}$ increases the influence of the prior and hence promotes sparsity.

Finally, the experiments are repeated for varying the true sparsity percentage in the weights with values taken in $\{20\%, 40\%, 60\%, 80\%\}$, see Fig. 4.5e. As can be seen in Fig. 4.5e the algorithm is robust and well performing in all of the considered cases. However, here one may note that the number of sparse entries in the target strongly impacts the accuracy at the onset of optimization. This is primarily due to the identification of constant zeros in any target and automatic pruning accordingly (hence the use of the local target in the mean estimation in Eq. (4.63)).

Next to the validation of the algorithm on one simple neuron case, a more complicated case in which two simple neurons are composed is also investigated. The two-layer system is described by:

$$\begin{aligned} \boldsymbol{h} &= \boldsymbol{\sigma}(\boldsymbol{s}^{(1)}), \quad \boldsymbol{s}^{(1)} := \boldsymbol{\Phi}\mathbf{w}^{(1)}, \\ \boldsymbol{y} &= \boldsymbol{\sigma}(\boldsymbol{s}^{(2)}), \quad \boldsymbol{s}^{(2)} := \boldsymbol{h}\mathbf{w}^{(2)}, \end{aligned} \tag{4.110}$$

with the indices (1) and (2) denoting the respective input and output layer. As one may notice, the described system represents a general feed-forward nonlinear multi-layer perceptron. The input parameter set $\boldsymbol{q}$ builds the basis function $\boldsymbol{\Phi} := [\boldsymbol{q}] \in \mathbb{R}^{1\times 1}$, whereas $\boldsymbol{\sigma}(\cdot)$ represents a sigmoid activation function that acts element-wise on its input argument $\boldsymbol{s}$. Other activation functions can be also used, yet this choice is made due to invertability condition of the activation function which then allows estimation of the true target states. The true weights $\mathbf{w}^{(1)} \in \mathbb{R}^{1\times 1}$ and $\mathbf{w}^{(2)} \in \mathbb{R}^{1\times 100}$ are sampled from a standard normal distribution with a pre-determined sparsity percentage. The input data is collected in matrix $\boldsymbol{\Phi} := [\mathbf{q}] \in \mathbb{R}^{n_\mathrm{d}\times 1}$ in which input signal $\mathbf{q} := [..., \boldsymbol{q}_i, ...], i = 1, ..., n_\mathrm{d}$ is sampled from a uniform distribution $\mathcal{U}(0, 1)$, and the forecast of the output is done by a Monte Carlo simulation using 1000 samples. The experimental setup needed to train the network is given in Table 4.2.

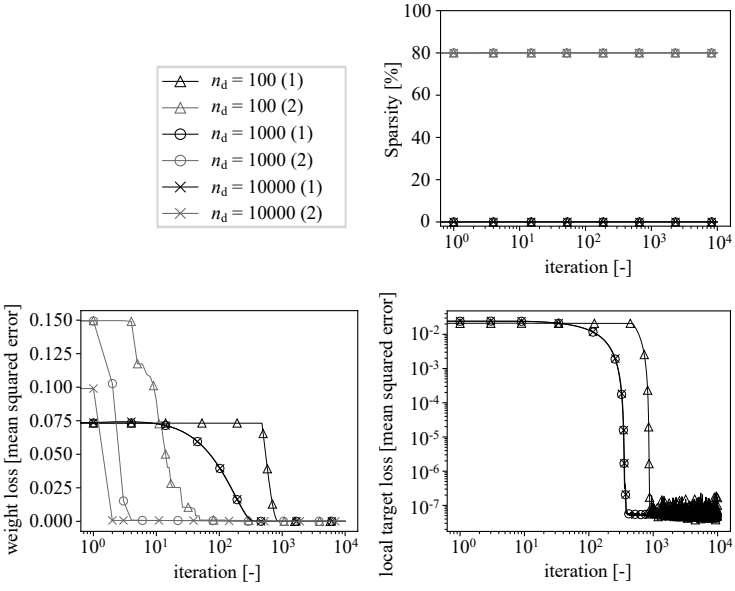
Table 4.2: Default parameter setting for numerical validation.

Parameter	n_d	λ	$\boldsymbol{\beta}$	$\boldsymbol{\alpha}$	sparsity
Default value (range)	1000	0.001	[1e4, 1e6]	[1e1, 1e12]	80%

Alike the single layer system considered, convergence is analyzed for different numbers of data samples $n_\mathrm{d} \in \{100, 1000, 10\,000\}$ in Fig. 4.6a excluding $n_\mathrm{d} = 100\,000$ due to high memory requirements. The more data points used, the faster the algorithm

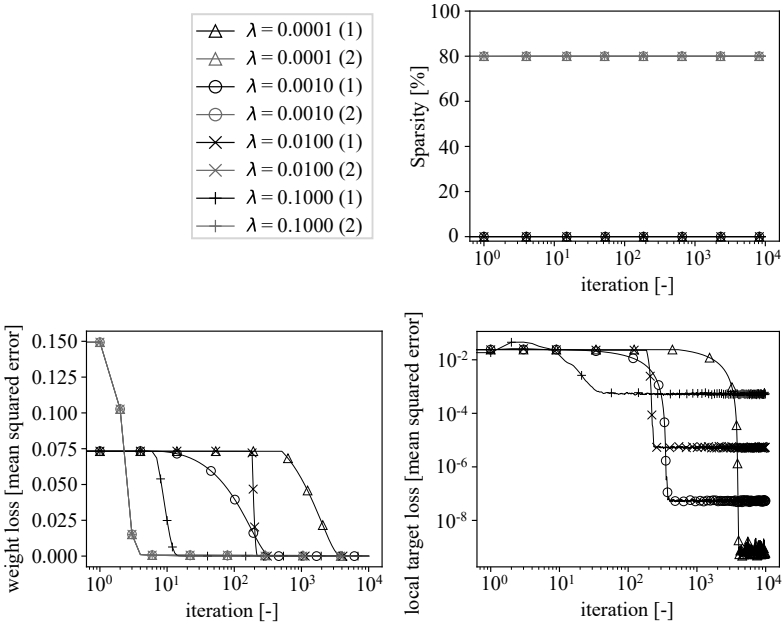
converges towards the true sparsity. In Fig. 4.6b the effect of the learning rate variation is investigated. For the largest considered learning rate ($\lambda = 0.1$) the algorithm is not robust and shows fluctuations at the onset in the local target loss in mean squared sense and corresponding weight loss. For $\lambda \leq 0.001$ the algorithm shows robust convergence for all considered weights and targets up to a high degree of accuracy.

When varying the upper range limits for $\boldsymbol{\beta}$, i.e. $\boldsymbol{\beta}_{\max} \in \{1e5, 1e6, 1e7\}$, one may observe robust convergence for all values, see Fig. 4.6c. However, the smaller the upper range limit for $\boldsymbol{\beta}$ is, the more restricted is its adaptability. This leads to slower convergence, which can be seen in the mean squared loss development on the local target. By varying $\boldsymbol{\alpha}_{\max} \in \{1e10, 1e11, 1e12, 1e13\}$ as shown in Fig. 4.6d no instabilities and distinguishable alterations in convergence of the local target are observed. Although at the onset of all iterations the mean squared error of the weights is larger in layer (2) for larger $\boldsymbol{\alpha}_{\max}$, sparsity percentages and weight values match the true weights, and an accurate depiction of the target data is achieved for all considered scenarios. By varying the sparsity percentage of the true weights the convergence over the weight matrices remains robust as can be seen in Fig. 4.6e. Similarly to the results of the single layer system, an increase in sparsity leads to an improvement in predictive accuracy as one has smaller number of non-zero weights and more data points per non-zero weight.

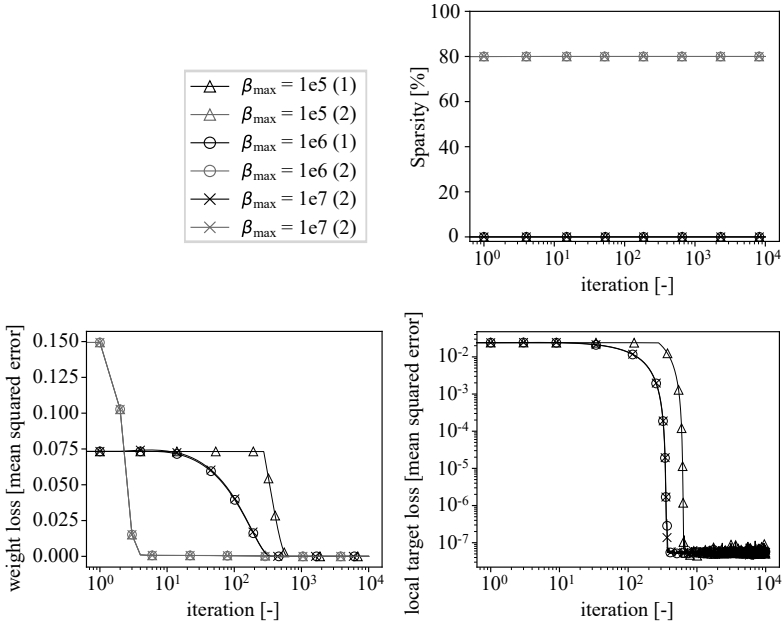


(a) Convergence for varying numbers of data samples n_d.

Figure 4.6: Convergence assessment on the true value weight sparsity, true value weight loss and local target loss of the algorithm in a two-layer system. *(cont.)*

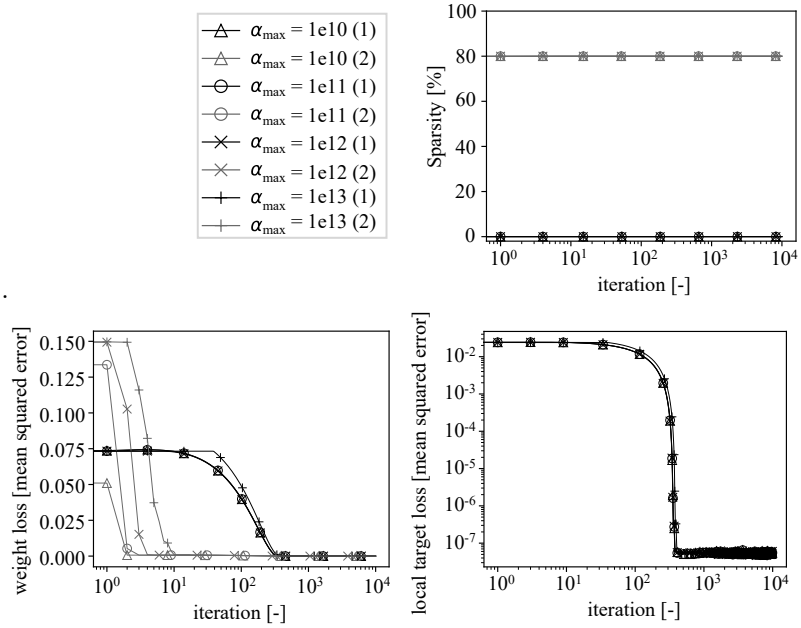


(b) Convergence for varying learning rates λ.

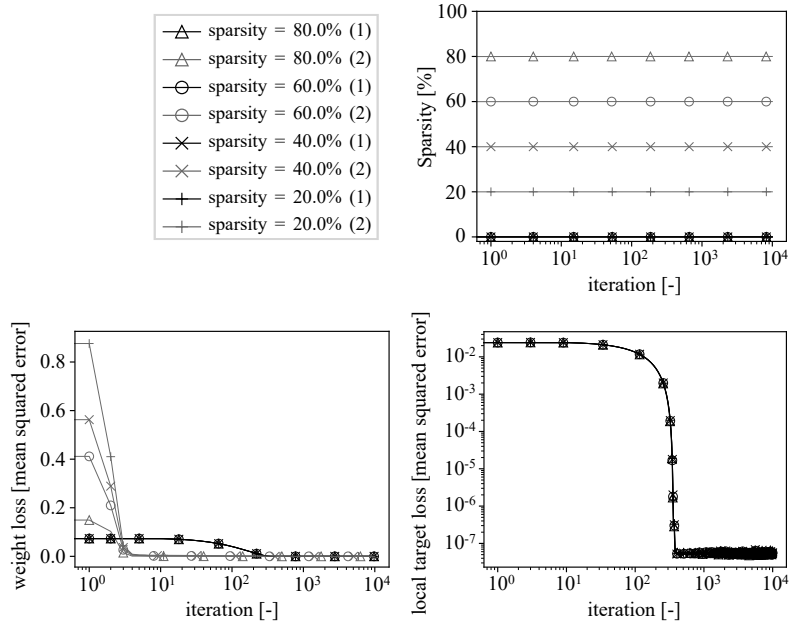


(c) Convergence for varying upper range limits for $\boldsymbol{\beta}$.

Figure 4.6: Convergence assessment on the true value weight sparsity, true value weight loss and local target loss of the algorithm in a two-layer system. *(cont.)*



(d) Convergence for varying upper range limits for $\boldsymbol{\alpha}$.



(e) Convergence for varying sparsity percentages on the true value weights.

Figure 4.6: Convergence assessment on the true value weight sparsity, true value weight loss and local target loss of the algorithm in a two-layer system.

A final convergence evaluation is performed by using a lower number of data samples $n_\mathrm{d} \in \{5, 8, 10, 20\}$ under the assumption of $\boldsymbol{\alpha} \in \{1\mathrm{e}1, 1\mathrm{e}6\}$ and $\boldsymbol{\beta} \in \{1\mathrm{e}4, 1\mathrm{e}6\}$ and an 80% true weight sparsity percentage in layer (2). Layer (1) has no sparsity included due to its dimension given $\mathbf{w}^{(2)} \in \mathbb{R}^{1\times 1}$. The upper range limits of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are chosen to increase numerical stability under a lower number of data samples. Accordingly, pruning is applied for the elements of $\boldsymbol{\gamma}$ being $\boldsymbol{\gamma} \leq 1\mathrm{e}{-4}$.

In Fig. 4.7 the corresponding optimization metrics are given, where robust convergence is observed under all considered variations of n_d. The assimilated weight values are found to converge faster to the true weight values for a larger number of data samples. The true value weight sparsity percentage is matched for all.

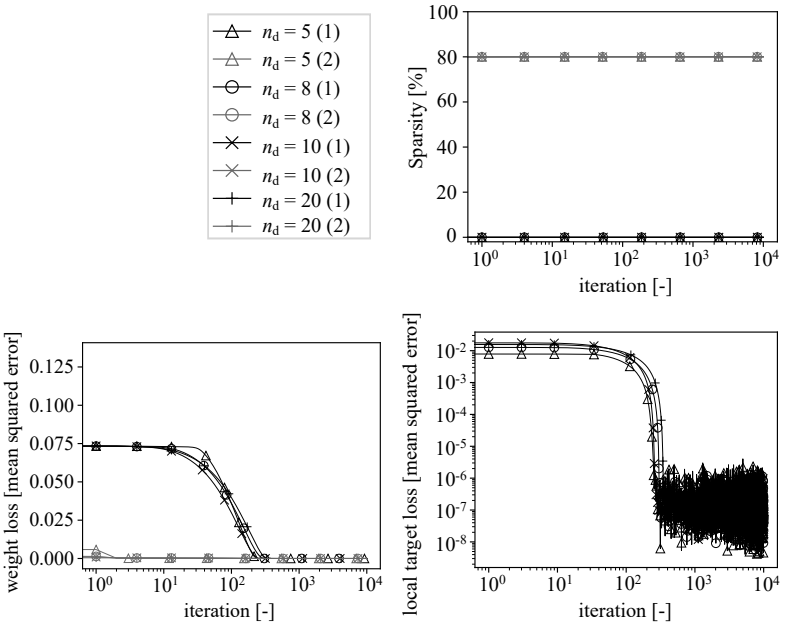


Figure 4.7: Convergence assessment on the true value weight sparsity, true value weight loss and local target loss of the algorithm in a two-layer system under the variation of the number of data samples (designs) $n_\mathrm{d} \in \{5, 8, 10, 20\}$.

4.3.1 Pruning threshold

Adequate convergence results have been found for a relatively conservative $\boldsymbol{\gamma} \leq 1\mathrm{e}{-4}$. By increasing the pruning threshold (i.e. $\boldsymbol{\gamma} > 1\mathrm{e}{-4}$) one increases the influence of the prior and hence promote sparsity. However, an increased pruning threshold naturally affects the models generalization capabilities as well. An alternative approach to pruning would be to evaluate the value of $\boldsymbol{\alpha}$ and prune for any of upper boundary values in $\boldsymbol{\alpha}$ (if $\boldsymbol{\alpha} \to \infty$ then $\boldsymbol{\mu} \to 0$, see Eq. (4.26)) [99, 186]. This alternative is further left out of consideration.

The proposed alternating update scheme on both the local target $[\mathbf{s}_{\mathrm{g},\ell}]_{:,k}$ and hyperparameters has been found to be unstable for unbounded hyperparameters. In such a case one may obtain a numerically singular Hessian matrix $[\mathbf{H}_{\mathrm{g},\ell}]_{:,:,k} := [\boldsymbol{\beta}_{\mathrm{g},\ell}]_k \boldsymbol{\Phi}_\ell^T \boldsymbol{\Phi}_\ell + [\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k} \odot \mathbf{I}$, where $[\boldsymbol{\Sigma}_{\mathrm{g},\ell}]_{:,:,k} = [\mathbf{H}_{\mathrm{g},\ell}]^{-1}_{:,:,k}$ (see Eq. (4.63)). The numerically singular Hessian matrix is also observed in [99]. One may prevent ill-conditioning in regular relevance vector machine by removing the entries in $[\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k}$ and $\boldsymbol{\Phi}_\ell$ that correspond to the pruned weights, as these entries are left unemployed. The removal of these entries is straightforward in regular relevance vector machine as the basis function and output dimensions are equal. In ARD-LSTM, with $\mathbf{w}_{\mathrm{g},\ell} \in \mathbb{R}^{n_\mathrm{i} \times n_\mathrm{m}}$ the basis function dimension $n_\mathrm{i} = 1 + n_\mathrm{f} + n_\mathrm{m}$ consists of the complete network width n_m and over all n_o basis functions in the subsequent output layer ($\mathbf{w}_{\mathrm{y},\ell} \in \mathbb{R}^{(n_\mathrm{m}+1) \times n_\mathrm{o}}$). One may be able to prune corresponding basis function if and only if the basis function $\boldsymbol{\Phi}_\ell$ is redundant over the complete network width n_m. Due to this unfeasible requirement, the basis function removal becomes very insignificant and therefore does not prevent any ill-conditioning. In the convergence evaluation it is shown that by using a bounded $[\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k}$ the ill-conditioning can also be prevented.

A similar approach on the threshold, as performed for the gates g, is also followed for the output layer on $\mathbf{w}_{\mathrm{y},\ell}$.

4.4 Structural application

With the predictive capabilities of an LSTM network in terms of strong bifurcations shown in Section 3.5.1 and the identified lack of generalization in Section 3.5.2, the bending specimen strip load case of Section 3.5.2 is reconsidered for the ARD-LSTM framework.

4.4.1 Structural model adaption

An elaborate description of the employed constitutive model is presented in Section 2.2. A variety of ARD-LSTM widths is analyzed ($n_\mathrm{m} \in \{16, 32, 64, 128\}$) to compare the sparsity effect and convergence speed to the traditional deterministic LSTM for varying network dimensions. For ARD-LSTM training the Adam optimizer learning rate is set to $\lambda = 0.005$ and a maximum number of 4000 epochs is considered. To allow for proper comparison, initialization of both traditional- and ARD-LSTM networks is performed using the same sampling and thus ensures an equal initial weight (mean). Additionally, all data points are assumed to have the same type of noise magnitude, the hyperparameter $[\boldsymbol{\beta}_{\mathrm{g},\ell}]_k$ is initialized in the domain $[\boldsymbol{\beta}_{\mathrm{g},\ell}]_k \in [1\mathrm{e}4, 1\mathrm{e}5]$. During optimization the hyperparameter boundaries are set by $[\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k} \in [1\mathrm{e}1, 1\mathrm{e}6]$ and $[\boldsymbol{\beta}_{\mathrm{g},\ell}]_k \in [1\mathrm{e}4, 1\mathrm{e}6]$. Sampling of the local target state $[\hat{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k}$ in the ARD-LSTM network is done by $n_\mathrm{s} = 100$ samples and pruning is applied for $[\boldsymbol{\gamma}_{\mathrm{g},\ell}]_{jk} \leq 1\mathrm{e}{-4}$. The same holds for the output layer ($\mathrm{g} = \mathrm{y}$). This less strict pruning threshold has been set to conform to the imposed boundaries on $[\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k}$ and $[\boldsymbol{\beta}_{\mathrm{g},\ell}]_k$, which disallows $[\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{jk} \to \infty$ and therefore demotivates $[\boldsymbol{\Sigma}_{\mathrm{g},\ell}]_{jjk} \approx [\boldsymbol{\alpha}_{\mathrm{g},\ell}]^{-1}_{jk}$ (see Eq. (4.36), Section 4.3.1). The deterministic network has no weight regularization and is optimized over a single batch. Both network types have been created and opti-

mized using the Tensorflow $v2.1.0$ framework [172] on a single Nvidia Quadro P5000 GPU.

4.4.2 Assessment of the model convergence

In Fig. 4.8 the negative log likelihood for all given widths n_m in ARD-LSTM is plotted. Whereas the wider networks are converging steadily, the smaller one for $n_\mathrm{m} = 16$ leads to a noisy estimate. This indicates underfitting on the given data set. An overview of

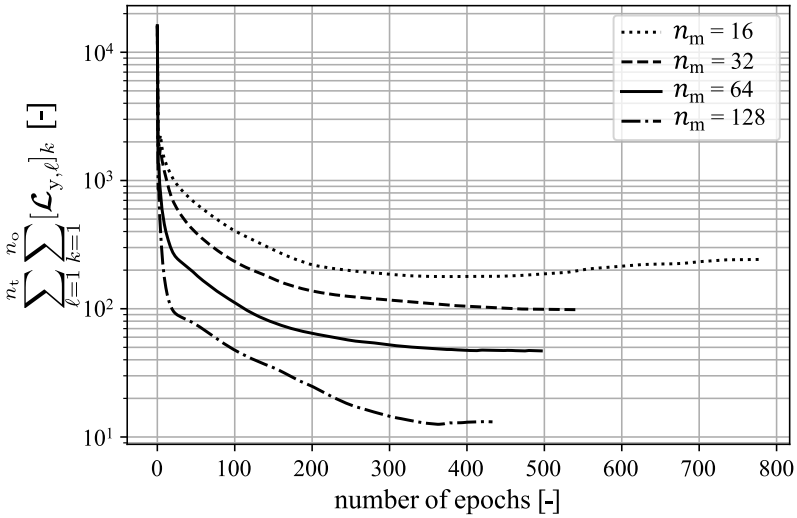
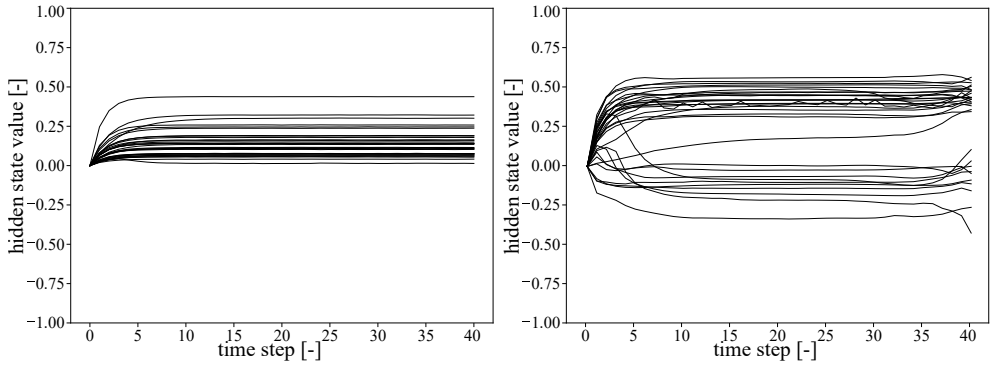


Figure 4.8: Negative log likelihood optimization up to convergence.

the optimization metrics is given in Table 4.3. Due to the increase in the number of free variables variables ARD-LSTM requires a longer computational time overall as well as per epoch compared to its deterministic counterpart. Although ARD-LSTM may be more memory intensive, it converges faster and hence does not require as many epochs as the classical LSTM.

In contrast to the deterministically optimized LSTM (see Section 3.5.2) the chosen network dimension n_m significantly affects the value of the optimization objective and the negative log likelihood significantly decreases for an increasing network dimension indicating an increase in the predictive likelihood. Although one may expect the proposed ARD-LSTM optimization scheme to negatively affect the number of required iteration steps, it has been found that the local target data relatively quickly converge to a stable setting. This is not only caused by an initial large error gradient, but is also stimulated by a non-vanishing gradient propagation through the LSTM cell. With the weights being stimulated towards a strongly peaked zero-mean a priori, and a hidden state bounded by $[-1, 1]$, the LSTM gate activation functions are only rarely saturated and thus provide a strong contribution to the gradient propagation (see Section 3.4). This is confirmed in Fig. 4.9. On the left side one can see the hidden state development for $n_\mathrm{m} = 32$ in design $i = 4$ ($\varphi = 0\,\mathrm{mm}$) at the first epoch ($e = 0$),

and on the right hand side the hidden state at epoch $e = 400$, when the network approaches convergence. In the initial steps the hidden state is strongly bounded, and gradually altered by the weights over subsequent epochs to maximize the likelihood objective.



(a) Hidden state at epoch $e = 1$.

(b) Hidden state at epoch $e = 400$.

Figure 4.9: Hidden state $\mathbb{E}(\mathbf{h}_i)$ development over time of all LSTM states in n_m for design $i = 4$ ($\varphi = 0\,\mathrm{mm}$).

4.4.3 Assessment of the sample space generalization

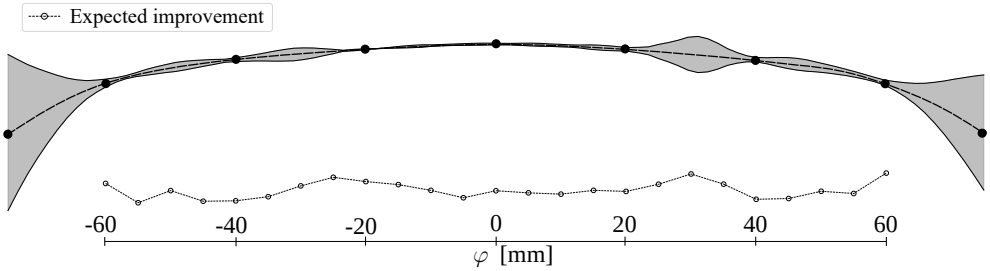
For comparison ease, the R^2 values of the point-estimate LSTM – trained without validation set (see Table 3.4) – are added in Table 4.3. For ARD-LSTM R^2 values are found to be higher over all widths, leading to a lower squared error. Whereas ARD-LSTM is able to cover 99.5 % of data variance at $n_\mathrm{m} = 32$, the deterministic framework only reaches a similar result at $n_\mathrm{m} = 128$. For ARD-LSTM a significant increase in time per epoch is found for $n_\mathrm{m} = 128$. The cause of this is the required batch-wise computation of the covariance to fit the available memory.

Table 4.3: Optimization metrics for all considered network widths $n_\mathrm{m} \in \{16, 32, 64, 128\}$.

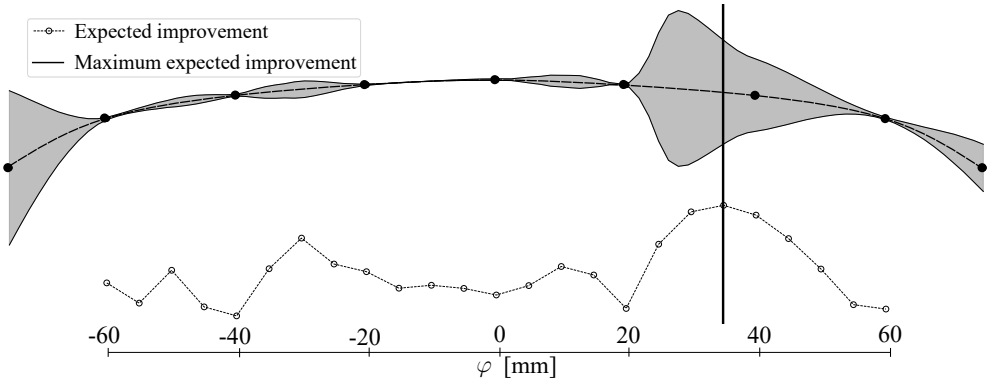
	n_m	epochs [−]	time [s]	time per epoch [s]	R^2 [−]
deterministic LSTM	16	4000	189	0.047	0.994
	32	4000	189	0.048	0.993
	64	4000	190	0.048	0.994
	128	4000	191	0.048	0.996
ARD-LSTM	16	779	153	0.20	0.993
	32	541	151	0.28	0.995
	64	497	317	0.64	0.998
	128	434	1403	3.23	0.998

In Fig. 4.10a the normalized maximum predictive standard deviation σ for $n_\mathrm{m} = 32$ over all time steps is plotted as an overlay on the convex envelope of Fig. 3.15. The

input φ is sampled 100 times linearly over the range $\varphi \in [-75\,\text{mm}, 75\,\text{mm}]$. Hence, all values outside of the $\varphi \in [-60\,\text{mm}, 60\,\text{mm}]$ interval are extrapolated. Although the magnitude of the normalized predictive standard deviation is small for $n_\text{m} = 32$ (see Fig. 4.10a), ARD-LSTM still clearly captures the difference in uncertainty between training samples, interpolation and extrapolation. A secondary overlay is given in Fig. 4.10b, the results of which are obtained by using a network with $n_\text{m} = 32$ trained without design $i = 6$ ($\varphi = 40\,\text{mm}$, Table 3.3) in the training set. Due to the lack of domain knowledge around $\varphi = 40\,\text{mm}$ the predictive uncertainty strongly increases and a local standard deviation increases from $\sigma = 1.8\text{e}{-2}$ to $\sigma = 5.2\text{e}{-2}$. This corresponds to a predictive standard deviation of 0.88 mm, where a predictive mean displacement of 7.03 mm is found.



(a) Normalized standard deviation overlay of ARD-LSTM with $n_\text{m} = 32$ trained on all data samples.



(b) Normalized standard deviation overlay of ARD-LSTM with $n_\text{m} = 32$ trained without design 6 ($\varphi = 40\,\text{mm}$).

Figure 4.10: Bending test data samples location indication and overlaying normalized maximum standard deviation plots on the convex envelope described in Fig. 3.15.

4.4.4 Expected improvement

With an eye on expensive training data generation, one desires to minimize the number of data samples required. It may therefore be useful to analyze the location where the maximum expected improvement is found. At every φ the expected improvement acquisition function can be expressed as [187, 188]:

$$\text{EI}(\varphi) = \mathbb{E}\left[\max\left((\hat{y}^*(\varphi) - y(\varphi), 0\right)\right] \tag{4.111}$$

in which $y(\varphi)$ describes the data sample and $\hat{y}^*(\varphi) \sim \mathcal{N}(\overline{y}^*, (\sigma^2)^*)$ describes the predictive output $\hat{y}^*(\varphi)$, see Eq. (4.68). The expected improvement is a measure that indicates the potential improvement of given $\hat{y}^*(\varphi)$ compared to the target data target data $y(\varphi)$ at a point φ. For $\sigma^*(\varphi) > 0$ this yields:

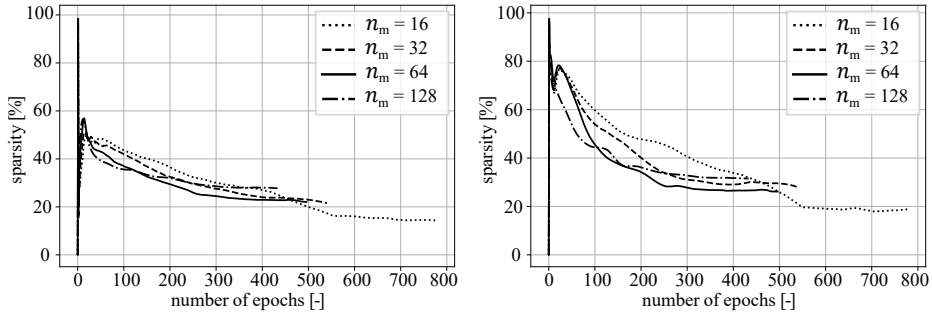
$$\mathrm{EI}(\varphi) = (\overline{y}^*(\varphi) - y(\varphi))\, \Phi_{\mathcal{N}} \left(\frac{\overline{y}^*(\varphi) - y(\varphi)}{\sigma^*(\varphi)} \right) + \sigma^*(\varphi) \mathcal{N} \left(\frac{\overline{y}^*(\varphi) - y(\varphi)}{\sigma^*(\varphi)} \right) \tag{4.112}$$

with $\Phi_{\mathcal{N}}$ being the cumulative distribution function of the standard Gaussian random variable. Here, $\mathrm{EI}(\varphi)$ is considered to be the maximum value over all nodes and time steps. In other words, a prediction $\hat{y}^*(\varphi)$ with a mean close to the data set $y(\varphi)$ and a small variance will have a small expected improvement. On the contrary, if the prediction has either a mean that strongly deviates from the target data or a high predictive uncertainty the expected improvement increases.

The most optimal design point is found by discretizing the domain $\varphi \in [-60\,\mathrm{mm}, 60\,\mathrm{mm}]$ by 25 designs that are linearly spread. The maximum expected improvements over all nodes and time steps corresponding to the 25 designs are depicted in Fig. 4.10a and Fig. 4.10b, below the variance overlay on the convex envelope. As a result of an increased predictive uncertainty (see Fig. 4.10b), the maximum expected improvement is found for $\varphi = 35\,\mathrm{mm}$, which is close to the removed data sample at $\varphi = 40\,\mathrm{mm}$. Hence, application of ARD-LSTM analysis provides the additional benefit of obtaining domain knowledge on the basis of predictive uncertainty. This domain knowledge allows the selection of a minimal-sized data set for satisfying interpolation uncertainty and thus model generalization.

4.4.5 Perspectives on sparsity

Sparsity is heavily stimulated at the onset of ARD-LSTM hyperparameter optimization caused by the prior-driven γ in initialization, see Eq. (4.36) and Section 4.2. In Fig. 4.11 the convergence of weights sparsity in the output layer and LSTM cell is shown for different architectures denoted by n_{m}. One can see that starting from a high sparsity percentage, the ARD framework re-introduces weights according to their relevance.



(a) Output layer sparsity. **(b)** LSTM cell sparsity.

Figure 4.11: Sparsity percentage for the ARD-LSTM cell and output layer.

A decreasing sparsity percentage for a more complex architecture may seem counter-intuitive. However, a varying network complexity (i.e. a different network width n_{m}) potentially leads to a different optimum in optimization. Note that sparsity in smaller networks may still be present due to weight redundancy.

The (mean) weight magnitudes of ARD-LSTM and the corresponding point-estimate weights are depicted in Fig. 4.12. The set of weights in ARD-LSTM are all Laplace-like shaped. The deterministic LSTM only gets small-valued weights on larger network widths. Although the deterministic weights are valued around zero, none exactly equals it.

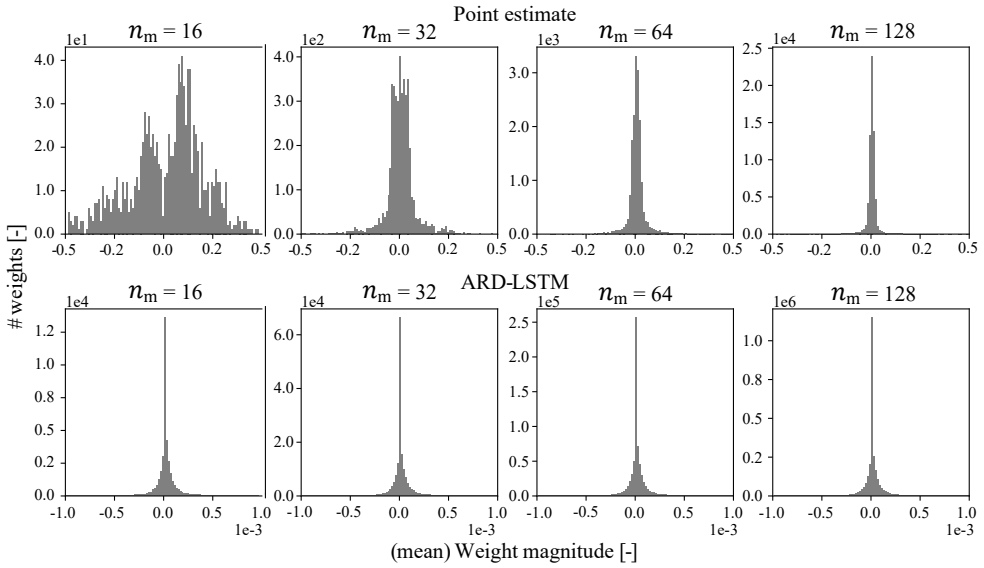


Figure 4.12: LSTM cell weight magnitude for the deterministic LSTM (top row) and ARD-LSTM (bottom row).

4.4.6 Findings on ARD-LSTM

With the introduction of Bayesian automatic relevance determination to the LSTM framework, a higher degree of generalization for a regression case is obtained, while showing a relatively fast convergence compared to the point-estimate equivalent. By automatically adapting the frameworks architecture to the data provided one may easily obtain the proper LSTM architecture for this very data set. As automatic relevance determination, in contrast to KL-divergence, does not require any expensive sampling in all linearized sections of the larger system it provides a relatively efficient stochastic solution. In Table 4.3 the optimization time and time per epoch of both the deterministic LSTM and the ARD-LSTM are listed for comparison purposes. Despite the increased time per epoch for ARD-LSTM, convergence at a lower number of iterations was found, leading to only minor differences in optimization time for the smaller networks.

A major drawback found in stochastic frameworks is the use of the covariance matrices. With the covariance $[\mathbf{\Sigma}_{\mathrm{g},\ell}]_{:,:,k} \in \mathbb{R}^{n_\mathrm{i} \times n_\mathrm{i}}$ for $n_\mathrm{i} = 1 + n_\mathrm{m} + n_\mathrm{f}$ and $[\mathbf{\Sigma}_{\mathrm{y},\ell}]_{:,:,k} \in \mathbb{R}^{(n_\mathrm{m}+1)\times(n_\mathrm{m}+1)}$, the corresponding computational expenses for estimating the covariance matrix are in the order of $\Theta(n_\mathrm{t} \times n_\mathrm{o} \times n_\mathrm{m}^3)$ – with $n_\mathrm{o} \equiv n_\mathrm{m}$ inside the LSTM cell – and grow exponentially with the network width n_m. Additionally, with time domain refinement in FEM or an increase in the number of FEM basis functions (mesh refinement), the covariance matrices linearly grow along. Besides computational effort, the ARD framework does not solve any under-fitting and may in this case cause a noisy optimization. Therefore, ensuring that the network is able to describe the problem complexity remains of great importance.

Where complex finite element simulations may require refinements on both the time and spatial domain, stochastic frameworks quickly become computationally intractable. A potential solution could be found by the identification of data dependency and separation of the time domain, spatial domain and stochastic domain. Hence, to further address both computational efficiency and large FEM-based data sets, one may use low-rank approximations as further discussed.

5 Low-rank approximations

In this chapter the stochastic, time and spatial dependency of the quantity of interest are analyzed and a corresponding separate representation is suggested. The goal is to provide a computationally efficient approximation of the stochastic FEM solution. At first, the reasoning and theory of separate domain representations is elaborated, followed by an explanation of the applied surrogate approaches for each of the aforementioned domains. The theory of automatic relevance determination is introduced to the applied surrogate models and subsequently evaluated using the previously introduced tapered tensile specimen.

5.1 Separate domain representation

As previously explained in Section 4.2, the QoI is approximated by a single NN that takes into account the parameter set as the input, and the time-spatial dependent output. Thus, the approximate model has many output neurons due to the high-dimensionality of the output. Although the newly proposed ARD-LSTM network is highly adaptable to various applications, such a model is still computationally expensive due to large number of input features and a very fine discretization of the stochastic FEM solution. To overcome this problem, some type of reduction approach to both the input and output data has to be applied. Under the assumption of separate representations [189], one may reduce the number of input parameters as well as both spatial and time related degrees of freedom by applying separate reduction approaches in each of the aforementioned domains. In other words, the goal is to obtain a QoI approximation of the type:

$$y(\omega, \tau, x) = \sum_{ijk} \upsilon_{ijk} \Psi_i(\omega) \Upsilon_j(\tau) \Gamma_k(x), \tag{5.1}$$

in which ω represents the probabilistic dependency as presented in Chapter 2, τ and x are time and the spatial dependencies, respectively. In Eq. (5.1) each of the discretization domains is approximated by the corresponding basis function that is in some sense optimal. Although for each of the mentioned basis functions one can make various types of choices, in this thesis the focus is not on finding the most optimal basis functions from available dictionary. Instead, the focus is on the efficient calculation of the selected basis function by utilizing the approach suggested in the previous chapter. In this light, basis functions are not all chosen to be of the NN type. Instead, only the time domain is considered in a NN setting, whereas both spatial and parameteric domains are approximated by already proven approaches. Therefore, the time dependency is further approximated by an ARD-LSTM network. Furthermore, the stochastic dependency is approximated by the well-established polynomial chaos expansion (PCE) [79–83]. The remaining spatial domain is represented by basis function $\Gamma_k(x)$ coming from the proper orthogonal decomposition (POD) [20–22], in which the given domain is decomposed to a reduced subspace using a (truncated) set of orthogonal eigenvectors. Although both PCE and ARD-LSTM are suitable for nonlinear approximations, the POD approach is not. To overcome this, the proposed approximation does not have a form as given in Eq. (5.1). Instead, the solution is first approximated by PCE and POD, and then the remaining time-dependent coefficients are approximated by the ARD-LSTM network. In this way the modeling error of the POD approach is accounted for.

5.1.1 Multi-element polynomial chaos expansion

To describe the QoI dependency on the parameter set $\boldsymbol{q}$, consisting of n_{f} independent varying design parameters, one may approximate the scalar output of QoI y_ℓ in the spatial point x and time moment ℓ by a polynomial chaos expansion [79, 80]:

$$y_\ell(\omega, x) := y(\omega, x, \tau_\ell) = \sum_{\alpha \in \mathcal{J}_*} y_\ell^{(\alpha)}(x) \boldsymbol{\Psi}^{(\alpha)}(\boldsymbol{\xi}(\omega)) \tag{5.2}$$

Here, input vector $\boldsymbol{q}(\omega)$ depends on a set of independent identically distributed (i.i.d.) random variables $\boldsymbol{\xi}(\omega) = (\xi_i(\omega))_{i=1}^{n_\mathrm{f}}$. Furthermore, $\boldsymbol{\Psi}$ are the orthogonal basis functions defined in a probabilistic manner and $\mathcal{J}_*$ is a multi-index set related to the multivariate polynomials $\boldsymbol{\Psi}^{(\alpha)}(\boldsymbol{\xi}(\omega))$ [134]. Further elaboration on the orthogonality condition of the basis functions is given in Appendix A.4.

As a consequence of orthogonality, one can truncate the expansion to a specific degree n_g depending on the desired model accuracy and generalization. Given Eq. (5.2), one may further approximate the QoI [134] as:

$$\tilde{y}_\ell(\omega, x) = \sum_{\alpha \in \mathcal{J}_p} y_\ell^{(\alpha)}(x) \boldsymbol{\Psi}^{(\alpha)}(\boldsymbol{\xi}(\omega)) \tag{5.3}$$

with $\mathcal{J}_p$ being the prospective truncated multi-index set of tuplets and $n_\mathrm{p} = \mathrm{card}(\mathcal{J}_p)$, arising from the number of input parameters n_f and the polynomial degree n_g. Hence, in discretized form the time-varying output of Eq. (5.3) may be rewritten to:

$$\tilde{y}_\ell(\omega_i, x_k) = \sum_{\alpha \in \mathcal{J}_p} y_\ell^{(\alpha)}(x_k) \boldsymbol{\Psi}^{(\alpha)}(\boldsymbol{\xi}(\omega_i)), \quad i = 1, ..., n_\mathrm{b};\ k = 1, ..., n_\mathrm{o}, \tag{5.4}$$

Here, n_o represents the spatial dimension as the number of elements or nodes in the FEM mesh and n_b is the number of data samples considered. Using Eq. (5.4) one may further write:

$$\tilde{\boldsymbol{y}}_{\ell k} = \boldsymbol{\Psi} \boldsymbol{w}_{\mathrm{c},\ell k}, \tag{5.5}$$

with the approximated QoI vector $\tilde{\boldsymbol{y}}_{\ell k} := [..., \tilde{y}_\ell(\omega_i, x_k), ...] \in \mathbb{R}^{n_\mathrm{b}}$. Furthermore, $\boldsymbol{w}_{\mathrm{c},\ell k} \in \mathbb{R}^{n_\mathrm{p}}$ are the PCE coefficients and $\boldsymbol{\Psi} \in \mathbb{R}^{n_\mathrm{b} \times n_\mathrm{p}}$ represents the collection of basis function samples. Accordingly one has $\tilde{\mathbf{y}}_\ell := [\tilde{\boldsymbol{y}}_{\ell k}]_{k=1}^{n_\mathrm{o}} \in \mathbb{R}^{n_\mathrm{b} \times n_\mathrm{o}}$ and $\mathbf{w}_{\mathrm{c},\ell} := [\boldsymbol{w}_{\mathrm{c},\ell k}]_{k=1}^{n_\mathrm{o}} \in \mathbb{R}^{n_\mathrm{p} \times n_\mathrm{o}}$.

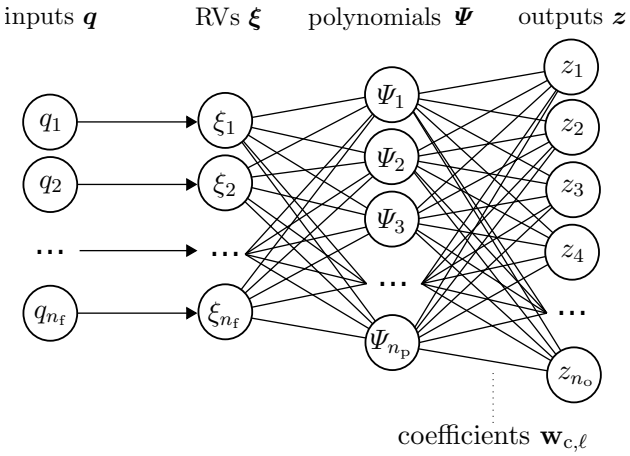


Figure 5.1: Graphical representation of a polynomial chaos model with $z_k := \tilde{\boldsymbol{y}}_\ell(x_k, \omega)$ and n_f input parameters.

If the PCE is to be represented as a nonlinear feed-forward NN, then one can depict Eq. (5.5) at the time step ℓ as shown in Fig. (5.1). The input layer with assumed independent parameters $\boldsymbol{q}$ is linearly or nonlinearly transformed to the natural random variables $\boldsymbol{\xi}(\omega)$ which are mapped to the output layer via nonlinear polynomial activation functions $\boldsymbol{\Psi}(\boldsymbol{\xi}(\omega))$. The weights of the last layer are the weights (i.e. coefficients) of the PCE, whereas the weights in the previous layer are weights of the corresponding Legendre polynomial. Thus, the training of the PCE coefficients $\mathbf{w}_{\mathrm{c},\ell}$ can be done in a classical gradient-descent approach as the one used in the traditional machine learning community. In other words, one may estimate the PCE coefficients as:

$$\boldsymbol{w}_{\mathrm{c},\ell k} = \min \mathcal{J}_{\ell k}, \tag{5.6}$$

in which the objective $\mathcal{J}_{\ell k}$ is given by

$$\mathcal{J}_{\ell k} = \frac{1}{n_{\mathrm{b}}} \sum_{i=1}^{n_{\mathrm{b}}} \| y_\ell(\omega_i, x_k) - \tilde{y}_{\ell k}(\boldsymbol{q}(\omega_i), \boldsymbol{w}_{\mathrm{c},\ell k}) \|_2^2, \quad k = 1, ..., n_{\mathrm{o}}. \tag{5.7}$$

in which $(\boldsymbol{q}_i, y_\ell(\omega_i, x_k)), i = 1, ..., n_{\mathrm{b}}$ denotes the real data set at the spatial point k and $\tilde{y}_\ell(\omega_i, x_k), i = 1, ..., n_{\mathrm{b}}$ represents the PCE model. Next to the regression approach presented in Eq. (5.7), one may also use the Moore-Penrose inverse $(\boldsymbol{\Psi}^T \boldsymbol{\Psi})^{-1} \boldsymbol{\Psi}^T$, and hence find the parameter by [190] by:

$$\boldsymbol{w}_{\mathrm{c},\ell k} = (\boldsymbol{\Psi}^T \boldsymbol{\Psi})^{-1} \boldsymbol{\Psi}^T \boldsymbol{y}_{\ell k}, \tag{5.8}$$

with the QoI matrix $\boldsymbol{y}_{\ell k} := [..., y_\ell(\omega_i, x_k), ...] \in \mathbb{R}^{n_{\mathrm{b}}}$, giving an iteration-free optimization procedure.

Due to the strong nonlinear dependency between the parameter set and the QoI and thus the potential multi-modality of the QoI, the approximation in Eq. (5.4) may not lead to the satisfactory results. Therefore, the multi-element approximation [191] is taken into account. The input vector $\boldsymbol{q}(\omega)$ depends on a set of i.i.d. random variables $\boldsymbol{\xi}(\omega) = (\xi_f(\omega))_{f=1}^{n_{\mathrm{f}}}$ in which the random variable $\xi_f(\omega)$ is defined by a uniform distribution on $\Xi_f := [a_f, b_f]$ with a_f and b_f being either finite or infinite in $\mathbb{R}$. Thus, the joint domain $\mathcal{D}$ of $\boldsymbol{\xi}(\omega)$, defined as $\mathcal{D} := \times_{f=1}^{n_{\mathrm{f}}} \Xi_f = \times_{f=1}^{n_{\mathrm{f}}} [a_f, b_f]$ - see Fig. 5.2 for the bi-variate case - can be further decomposed as:

$$\mathcal{D} = \cup_{m=1}^{n_{\mathrm{c}}} \mathcal{D}_m, \quad \mathcal{D}_m = [a_1^m, b_1^m) \times [a_2^m, b_2^m) \times ... \times [a_{n_{\mathrm{f}}}^m, b_{n_{\mathrm{f}}}^m] \tag{5.9}$$

such that $\mathcal{D}_m \cap \mathcal{D}_n = \emptyset$ if $m \neq n$. Here, n_{c} denotes the number of stochastic elements (not to be confused with elements arising from a FEM mesh discretization). Given the indicator variable $I_{\mathcal{D}_m}$, defined as $I_{\mathcal{D}_m} = 1$ if $\boldsymbol{\xi} \in \mathcal{D}_m$ and zero otherwise, one may define the multi-element PCE approximation[1] at the spatial point x_k by:

$$\tilde{y}_\ell(\omega, x_k) = \sum_{m=1}^{n_{\mathrm{c}}} \tilde{y}_{\ell,m} I_{\mathcal{D}_m} = \sum_{m=1}^{n_{\mathrm{c}}} \sum_{\alpha_m \in \mathcal{J}_m} y_{\ell,m}^{(\alpha_m)}(x_k) \Psi^{(\alpha_m)}(\boldsymbol{\xi}(\omega)) I_{\mathcal{D}_m} \tag{5.10}$$

[1] convergence in L_2 sense is shown in [191]

according to which one may also write the input vector $\boldsymbol{q}(\omega)$ in the form:

$$\boldsymbol{q}(\omega) = \sum_{m=1}^{n_\mathrm{c}} \boldsymbol{q}_m(\omega) I_{\mathcal{D}_m}. \tag{5.11}$$

As can be seen in the previous two equations, the QoI is approximated by the local PCE's, instead of the global one. Accordingly, one may rewrite the local PCE approximation given in Eq. (5.10) using Eq. (5.5) in a matrix-vector form:

$$\tilde{\boldsymbol{y}}_{\ell k} = \sum_{m}^{n_\mathrm{c}} \boldsymbol{\Psi}^{(m)} \boldsymbol{w}_{\mathrm{c},\ell k}^{(m)} I_{\mathcal{D}_m}, \quad k = 1, ..., n_\mathrm{o} \tag{5.12}$$

given the PCE coefficients $\mathbf{w}_{\mathrm{c},\ell}^{(m)} \in \mathbb{R}^{n_\mathrm{p} \times n_\mathrm{o}}$.

When subdividing each random parameter in $\boldsymbol{q}$ into n_r elements, one obtains a total of $n_\mathrm{c} = n_\mathrm{r}^{n_\mathrm{f}}$ base variables, see Fig. 5.2. Here is assumed that the base variables are of the uniform type defined over the domain $[-1, 1]$ and the data samples are indicated by the black dots.

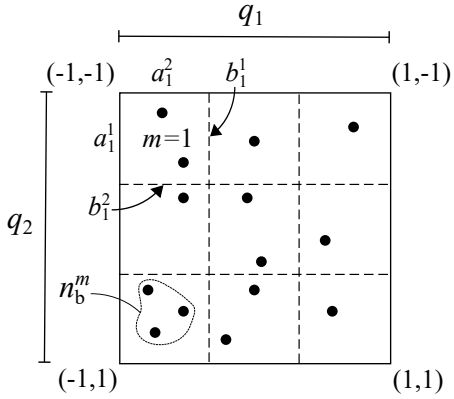


Figure 5.2: Schematic representation for the $(m = 1, ..., n_\mathrm{c})$, $n_\mathrm{c} = 9$ stochastic elements from $n_\mathrm{r} = 3$ subdivisions of a $n_\mathrm{f} = 2$ bi-variate (q_1, q_2) case, yielding $\boldsymbol{a}_m \in \mathbb{R}^2$ and $\boldsymbol{b}_m \in \mathbb{R}^2$.

An equidistant element separation in a probabilistic space is in general not optimal, and one may therefore search for n_c optimal elements with their corresponding intervals. Hence, when subdividing each of the parameters in $\boldsymbol{q}$ into n_r disjoint elements, the aim is to find the optimal position of the non-equidistant n_r elements, i.e. their boundaries. When assuming a pre-defined number of n_r elements, one may therefore define an optimization problem that incorporates the optimization objective for the estimation of the optimal local PCE approximation of the QoI on each element and for the estimation of the optimal boundaries of the stochastic elements.

Let the unknown boundary positions of random vectors $\boldsymbol{\xi}_m(\omega)$ be defined by parameter sets $\boldsymbol{a}_m \in \mathbb{R}^{n_\mathrm{f}}$ and $\boldsymbol{b}_m \in \mathbb{R}^{n_\mathrm{f}}$, representing respectively the lower and upper boundary values of all input parameters at element m, see Fig. 5.2. Accordingly one may define the complete set of unknown stochastic element boundaries by

$\mathbf{a} = [\boldsymbol{a}_m]_{m=1}^{n_c} \in \mathbb{R}^{n_c \times n_f}$ and $\mathbf{b} = [\boldsymbol{b}_m]_{m=1}^{n_c} \in \mathbb{R}^{n_c \times n_f}$. Next to the unknown boundaries, one has to also find the local PCE coefficients that correspond to the orthogonal basis function expressed in terms of $\boldsymbol{\xi}_m$. For this purpose one has to define an objective function that measures the discrepancy between the approximated QoI in Eq. (5.10) and the real data set. If the discrepancy is measured in L_2 sense one has:

$$\begin{aligned}(\mathbf{w}_c^*, \mathbf{a}^*, \mathbf{b}^*) &= \min_{\mathbf{w}_c, \mathbf{a}, \mathbf{b}} \mathcal{E}(\boldsymbol{w}_{c,\ell k}^{(m)}, \boldsymbol{a}_m, \boldsymbol{b}_m) \\ &= \min_{\mathbf{w}_c, \mathbf{a}, \mathbf{b}} \sum_m^{n_c} \sum_\ell^{n_t} \sum_k^{n_o} \mathcal{J}_{\ell k}^{(m)}(\boldsymbol{w}_{c,\ell k}^{(m)}, \boldsymbol{a}_m, \boldsymbol{b}_m)\end{aligned} \tag{5.13}$$

in which

$$\mathcal{J}_{\ell k}^{(m)}(\boldsymbol{w}_{c,\ell k}^{(m)}, \boldsymbol{a}_m, \boldsymbol{b}_m) := \frac{1}{n_b^m} \sum_{i=1}^{n_b^m} \| y_\ell^{(m)}(\omega_i, x_k) - \tilde{y}_{\ell k}^{(m)}(\boldsymbol{q}^{(m)}(\omega_i), \boldsymbol{w}_{c,\ell k}^{(m)}, \boldsymbol{a}_m, \boldsymbol{b}_m) \|_2^2 \tag{5.14}$$

given the collection of weights $\mathbf{w}_c := \{...\boldsymbol{w}_{c,\ell k}...\}$ and with n_b^m denoting the fraction of n_b data samples located in stochastic element m, see Fig. 5.2. Here, one must beware that two adjacent stochastic elements in a rectilinear grid of stochastic elements share a boundary and hence to ensure a rectilinear grid using Eq. (5.9) one may deduce that a total of $n_j = (n_r - 1)^{n_f}$ unknown boundaries positions can be defined with $n_j < n_c$. For the case described in Fig. 5.2 one obtains $n_j = 4 = (3-1)^2$, as indicated by the two horizontal and vertical dashed lines. Thus, a total of $n_j = 4$ unknown stochastic boundaries remain to be estimated.

To solve the previously defined optimization problem, one can use the alternative minimization principle. In other words, one can initialize and fix boundaries $(\mathbf{a}, \mathbf{b})$ to values $(\mathbf{a}^{(e)}, \mathbf{b}^{(e)})$ to find the optimal PCE coefficients by minimizing:

$$\boldsymbol{w}_{c,\ell k}^{(e),(m)} = \min_{\boldsymbol{w}_{c,\ell k}^{(m)}} \mathcal{J}_{\ell k}^{(m)}(\boldsymbol{w}_{c,\ell k}^{(m)}, \boldsymbol{a}_m^{(e)}, \boldsymbol{b}_m^{(e)}). \tag{5.15}$$

The previous minimization problem resembles a reformulation of the problem presented in Eq. (5.6), now depending on the stochastic elements. Hence, Eq. (5.15) can be solved by use of Eq. (5.7) or Eq. (5.8). However, this approach is not promoting for their sparsity. Thus, here a new method is proposed that is further explained in Section 5.1.2.

Once the PCE coefficients have converged by using the corresponding minimization procedure, one may fix them to $\boldsymbol{w}_{c,\ell k}^{(e),(m)}$, and search for the element boundaries $(\mathbf{a}^{(e+1)}, \mathbf{b}^{(e+1)})$. These then can be obtained by minimizing:

$$(\mathbf{a}^{(e+1)}, \mathbf{b}^{(e+1)}) = \min_{\mathbf{a}, \mathbf{b}} \sum_m^{n_c} \sum_\ell^{n_t} \sum_k^{n_o} \mathcal{J}_{\ell k}^{(m)}(\boldsymbol{w}_{c,\ell k}^{(e),(m)}, \boldsymbol{a}_m, \boldsymbol{b}_m). \tag{5.16}$$

The boundary adaption by minimizing the found discrepancy is performed using the numerical simplex based Nelder-Mead scheme [192] which can be employed to solve

unconstrained optimization problems in the general form. An alternative optimization method, also known as Powell's method, is proposed by [193]. In contrast to Nelder-Mead, Powells method is gradient based and handles all free parameters subsequently. Powells method, however, handles the free parameters independently in a sequential sense, and thus does not account the possible influence of a single free parameter change on the global optimum and is therefore not applied in this thesis.

This alternating process of estimating the local polynomial chaos expansions and adapting the stochastic element boundaries is repeated until the discrepancy reaches a predetermined bottom threshold $\mathcal{E}^*$. A global algorithmic view of the optimization is given in Algorithm 2. As commented in previous chapters, estimation of the

Algorithm 2 Multi-element PCE boundary optimization.

Require: Set the number of elements: $m = 1, ..., n_\mathrm{c}$

Require: Scale the input parameters: $q_j(\omega) \in [-1, 1],\ j = 1, ..., n_\mathrm{f}$

▷ Uniform distribution $q_j \sim \mathcal{U}(-1, 1)$ assumed

Require: Initialize the boundaries by sampling $\boldsymbol{q}(\omega)$: $(\boldsymbol{a}_m^{(e)}, \boldsymbol{b}_m^{(e)}) \in [-1, 1]$

Require: Define the bottom threshold discrepancy: $\mathcal{E}^*$

1: **While** $\mathcal{E} > \mathcal{E}^*$: ▷ Iterate

Describe the PCE objective:

2: $\mathcal{J}_{\ell k}^{(m)}(\boldsymbol{w}_{\mathrm{c},\ell k}^{(m)}, \boldsymbol{a}_m, \boldsymbol{b}_m) = \frac{1}{n_\mathrm{b}^m} \sum_{i=1}^{n_\mathrm{b}^m} \| y_\ell^{(m)}(\omega_i, x_k) - \tilde{y}_{\ell k}^{(m)}(\boldsymbol{q}^{(m)}(\omega_i), \boldsymbol{w}_{\mathrm{c},\ell k}^{(m)}, \boldsymbol{a}_m, \boldsymbol{b}_m) \|_2^2$

Fix the boundaries $(\boldsymbol{a}_m^{(e)}, \boldsymbol{b}_m^{(e)})$ and estimate the local PCE coefficients by:

3: $\boldsymbol{w}_{\mathrm{c},\ell k}^{(e),(m)} = \min_{\boldsymbol{w}_{\mathrm{c},\ell k}^{(m)}} \mathcal{J}_{\ell k}^{(m)}(\boldsymbol{w}_{\mathrm{c},\ell k}^{(m)}, \boldsymbol{a}_m^{(e)}, \boldsymbol{b}_m^{(e)})$

Describe the discrepancy:

4: $\mathcal{E}(\boldsymbol{w}_{\mathrm{c},\ell k}^{(m)}, \boldsymbol{a}_m, \boldsymbol{b}_m) = \sum_m^{n_\mathrm{c}} \sum_\ell^{n_\mathrm{t}} \sum_k^{n_\mathrm{o}} \mathcal{J}_{\ell k}^{(m)}(\boldsymbol{w}_{\mathrm{c},\ell k}^{(m)}, \boldsymbol{a}_m, \boldsymbol{b}_m)$

Fix the weights $\boldsymbol{w}_{\mathrm{c},\ell k}^{(m)}$ and adapt the boundaries by minimizing the discrepancy using Nelder-Mead:

5: $(\mathbf{a}^{(e+1)}, \mathbf{b}^{(e+1)}) = \min_{\mathbf{a},\mathbf{b}} \mathcal{E}(\boldsymbol{w}_{\mathrm{c},\ell k}^{(m)}, \boldsymbol{a}_m, \boldsymbol{b}_m)$

6: Set the next iteration step $e = e + 1$

Return the optimized boundaries and PCE coefficients for all n_c elements:

7: **return** $\mathbf{a}^*, \mathbf{b}^*, \mathbf{w}_\mathrm{c}^*$

PCE coefficients (or NN weights) in a mean squared (i.e. deterministic) sense does not impose self-configuration of the given framework. Alike the introduced novel optimization scheme in LSTM by means of ARD, one may therefore further employ ARD to estimate the PCE coefficients over each element with the goal of introducing a self-configuring architecture.

5.1.2 Automatic relevance based polynomial chaos expansion

The first step towards a probabilistic polynomial chaos estimation is to redefine Eq. (5.7) in a probabilistic sense, alike Eq. (4.4). With the coefficients $\mathbf{w}_{\mathrm{c},\ell}$ considered unknown, one may model them as uncertain $\mathbf{w}_{\mathrm{c},\ell}(\omega_c)$ in $(\Omega_c, \mathfrak{F}_c, \mathbb{P}_c)$. Using Eq. (5.4) and Eq. (4.4) one can predict the time-varying QoI:

$$\hat{\boldsymbol{y}}_\ell(\omega_c, \omega_e) = \tilde{\boldsymbol{y}}_\ell(\omega_c) + \boldsymbol{\varepsilon}_\ell(\omega_e) \tag{5.17}$$

in which $\boldsymbol{\varepsilon}_\ell(\omega_e)$ is the prediction of the unknown modeling and measurement error here assumed to be independent of $\mathbf{w}_{\mathrm{c},\ell}(\omega_c)$. Introducing the joint space $(\Omega_s, \mathfrak{F}_s, \mathbb{P}_s)$ with $\Omega_s := \Omega_c \times \Omega_\varepsilon$, the previous equation can be written as:

$$\hat{\boldsymbol{y}}_\ell(\omega_s) = \tilde{\boldsymbol{y}}_\ell(\omega_s) + \boldsymbol{\varepsilon}_\ell(\omega_s). \tag{5.18}$$

The error term $\boldsymbol{\varepsilon}_\ell(\omega_s)$ is independent of $\boldsymbol{\varepsilon}_m(\omega_s)$ for any time step $\ell \neq m$.

Assuming that the PCE coefficients follow a Gaussian prior distribution function $p([\mathbf{w}_{\mathrm{c},\ell}^{(m)}]_{:,k})$, one may employ Bayesian theory to estimate them as [99]:

$$p([\mathbf{w}_{\mathrm{c},\ell}^{(m)}]_{:,k}|[\mathbf{y}_\ell]_{:,k}) = \frac{p([\mathbf{y}_\ell]_{:,k}|[\mathbf{w}_{\mathrm{c},\ell}^{(m)}]_{:,k})p([\mathbf{w}_{\mathrm{c},\ell}^{(m)}]_{:,k})}{P([\mathbf{y}_\ell]_{:,k})}, k=1,...,n_{\mathrm{o}}; m=1,...,n_{\mathrm{c}}. \tag{5.19}$$

Here, the spatial dimension n_{o} is defined by the number of elements or nodes in the FEM mesh and n_{c} is the number of stochastic elements. The Bayesian description in Eq. (5.19) may be employed to find the optimized set of coefficients in the local PCE. In Eq. (5.19) the prior probability density function $p([\mathbf{w}_{\mathrm{c},\ell}^{(m)}]_{:,k})$ is updated using the likelihood $p([\mathbf{y}_\ell]_{:,k}|[\mathbf{w}_{\mathrm{c},\ell}^{(m)}]_{:,k})$ and evidence $P([\mathbf{y}_\ell]_{:,k})$ to eventually deliver the posterior over the coefficients $p([\mathbf{w}_{\mathrm{c},\ell}^{(m)}]_{:,k}|[\mathbf{y}_\ell]_{:,k})$, giving the posterior mean and variance of the density function considered.

To introduce sparsity to the estimation of the PCE coefficients, one may further use the same theory as presented in Chapter 4. In contrast to the ARD-LSTM estimation, the QoI is linear in the unknown coefficients in the PCE, and hence one may use the straightforward ARD scheme as presented in Section 4.1.1. Thus, the PCE coefficients are assumed to follow:

$$p(\mathbf{w}_{\mathrm{c},\ell}^{(m)}|\boldsymbol{\alpha}_\ell^{(m)}) = \prod_{j=1}^{n_{\mathrm{p}}} \prod_{k=1}^{n_{\mathrm{o}}} \mathcal{N}(0, [\boldsymbol{\alpha}_\ell^{(m)}]_{jk}^{-1}) \tag{5.20}$$

in which $\boldsymbol{\alpha}_\ell^{(m)}$ is described by a non-informative (i.e. uniform) hyperprior $p(\boldsymbol{\alpha}_\ell^{(m)})$. On a logarithmic scale such hyper-parameter dependent distribution matches a Gamma distribution for shape and scale parameters $a = b = 0$, as given in Eq. (4.21) [99]. Hence, rewriting Eq. (4.21) for $\boldsymbol{\alpha}_\ell^{(m)}$ gives:

$$p(\boldsymbol{\alpha}_\ell^{(m)}) = \prod_{j=1}^{n_{\mathrm{p}}} \prod_{k=1}^{n_{\mathrm{o}}} \Gamma(a)^{-1} b^a [\boldsymbol{\alpha}_\ell^{(m)}]_{jk}^{a-1} \exp(-b[\boldsymbol{\alpha}_\ell^{(m)}]_{jk}), \tag{5.21}$$

where $\Gamma(a) = \int_0^\infty t^{a-1} \exp(-t)\mathrm{d}t$. Similarly, hyperprior $\boldsymbol{\beta}_\ell^{(m)}$ may be defined using Eq. (4.22) to describe the existing noise variance in the likelihood:

$$p(\boldsymbol{\beta}_\ell^{(m)}) = \prod_{k=1}^{n_\mathrm{o}} \Gamma(v)^{-1} d^v [\boldsymbol{\beta}_\ell^{(m)}]_k^{v-1} \exp(-d\ [\boldsymbol{\beta}_\ell^{(m)}]_k), \quad \ell = 1, ..., n_\mathrm{t}, \tag{5.22}$$

with v and d being the scale parameters. Note that the described conjugacy only holds for a system that describes a linear relation between the coefficients and the quantity of interest, which is the case in PCE.

Alike Eq. (4.23) the posterior in Eq. (5.19) is factorized into two posteriors: the product over the posterior over the coefficients $p(\mathbf{w}_{\mathrm{c},\ell}^{(m)}|\mathbf{y}_\ell, \boldsymbol{\alpha}_\ell^{(m)}, \boldsymbol{\beta}_\ell^{(m)})$ and the posterior over the hyperparameters $p(\boldsymbol{\alpha}_\ell^{(m)}, \boldsymbol{\beta}_\ell^{(m)}|\mathbf{y}_\ell)$, i.e.:

$$p(\mathbf{w}_{\mathrm{c},\ell}^{(m)}, \boldsymbol{\alpha}_\ell^{(m)}, \boldsymbol{\beta}_\ell^{(m)}|\mathbf{y}_\ell) = p(\mathbf{w}_{\mathrm{c},\ell}^{(m)}|\mathbf{y}_\ell, \boldsymbol{\alpha}_\ell^{(m)}, \boldsymbol{\beta}_\ell^{(m)}) p(\boldsymbol{\alpha}_\ell^{(m)}, \boldsymbol{\beta}_\ell^{(m)}|\mathbf{y}_\ell). \tag{5.23}$$

where the posterior over the coefficients $p(\mathbf{w}_{\mathrm{c},\ell}^{(m)}|\mathbf{y}_\ell, \boldsymbol{\alpha}_\ell^{(m)}, \boldsymbol{\beta}_\ell^{(m)})$ is a convolution of Gaussians, thus giving a multivariate Gaussian distribution of which the mean $\boldsymbol{\mu}_\ell^{(m)}$ and covariance $\boldsymbol{\Sigma}_\ell^{(m)}$ (with corresponding matrix elements $[\boldsymbol{\Sigma}_\ell^{(m)}]_{:,:,k} \in \mathbb{R}^{n_\mathrm{p} \times n_\mathrm{p}}$) read:

$$\begin{aligned} [\boldsymbol{\mu}_\ell^{(m)}]_{:,k} &:= [\boldsymbol{\beta}_\ell^{(m)}]_k [\boldsymbol{\Sigma}_\ell^{(m)}]_{:,:,k} (\boldsymbol{\Psi}^{(m)})^T [\mathbf{y}_\ell]_{:,k}, \quad k = 1, ..., n_\mathrm{o}, \\ [\boldsymbol{\Sigma}_\ell^{(m)}]_{:,:,k} &:= ([\boldsymbol{\beta}_\ell^{(m)}]_k (\boldsymbol{\Psi}^{(m)})^T \boldsymbol{\Psi}^{(m)} + [\boldsymbol{\alpha}_\ell^{(m)}]_{:,k} \odot \mathbf{I})^{-1}, \end{aligned} \tag{5.24}$$

for $\mathbf{I} \in \mathbb{R}^{n_\mathrm{p} \times n_\mathrm{p}}$, the basis functions $\boldsymbol{\Phi}_\ell \in \mathbb{R}^{n_\mathrm{b} \times n_\mathrm{p}}$ and the data set $[\mathbf{y}_\ell]_{:,k} \in \mathbb{R}^{n_\mathrm{b}}$. Additionally, $[\boldsymbol{\mu}_\ell^{(m)}]_{:,k} \in \mathbb{R}^{n_\mathrm{p}}$, $[\boldsymbol{\alpha}_\ell^{(m)}]_{:,k} \in \mathbb{R}^{n_\mathrm{p}}$ and $[\boldsymbol{\beta}_\ell^{(m)}]_k \in \mathbb{R}$.

The second term in Eq. (5.23) may be approximated by its most probable (MP) value $p(\boldsymbol{\alpha}_\ell^{(m)}, \boldsymbol{\beta}_\ell^{(m)}|\mathbf{y}_\ell) \approx \delta((\boldsymbol{\alpha}_\ell^{(m)})^\mathrm{MP}, (\boldsymbol{\beta}_\ell^{(m)})^\mathrm{MP})$, similarly as in Section 4.1.1.

As aforementioned, the introduction of a Bayesian coefficient description may not only lead to the introduction of coefficient sparsity, but also impose propagation of modeling and measurement errors in the surrogate model. The posterior PCE coefficients can be further propagated through PCE expansion to obtain the posterior predictive. In other words, the obtained PCE approximation next to the input ("physical") uncertainty also contains the epistemic uncertainty that gives the confidence in the approximated values. The epistemic uncertainty thus can be estimated by a convolution of Gaussians to obtain [99]:

$$p([\hat{\mathbf{y}}_\ell]_{:,k}|[\mathbf{y}_\ell]_{:,k}) = \mathcal{N}([\hat{\mathbf{y}}_\ell]_{:,k}|[\hat{\boldsymbol{\mu}}_\ell^\mathrm{y}]_{:,k}, [\hat{\boldsymbol{\sigma}}_\ell^2]_{:,k}), \tag{5.25}$$

where the predictive mean $[\hat{\boldsymbol{\mu}}_\ell^\mathrm{y}]_{:,k}$ and finite variance $[\hat{\boldsymbol{\sigma}}_\ell^2]_{:,k}$ are given by

$$\begin{aligned} [\hat{\boldsymbol{\mu}}_\ell^\mathrm{y}]_k &= \sum_m^{n_\mathrm{c}} \boldsymbol{\Psi}^{(m)} [\boldsymbol{\mu}_\ell^{(m)}]_{:,k} I_{\mathcal{D}_m}, \quad k = 1, ..., n_\mathrm{o}, \\ [\hat{\boldsymbol{\sigma}}_\ell^2]_{ik} &= \sum_m^{n_\mathrm{c}} \left([(\boldsymbol{\beta}_\ell^{(m)})^\mathrm{MP}]_k^{-1} + \boldsymbol{\Psi}_i^{(m)} [\boldsymbol{\Sigma}_\ell^{(m)}]_{:,:,k} (\boldsymbol{\Psi}_i^{(m)})^T \right) I_{\mathcal{D}_m}, \quad i = 1, ..., n_\mathrm{b}. \end{aligned} \tag{5.26}$$

In the upcoming, the automatic relevance determination (ARD) scheme applied to PCE may be denoted by ARD-PCE.

Given that the ARD-PCE automatically prunes coefficients according to its determined relevance in the problem description provided, the ARD-PCEs descriptive complexity is automatically adapted.

With the number of polynomials in a multivariate situation growing exponentially, the coefficient matrix grows strongly along, making its usage and storage memory intensive. Additionally, the number of nodes and elements in a finite element solution grow with a desire to (locally) improve the solution accuracy. With the stochastic domain handled by ARD-PCE one may further use the spatial domain decomposition in order to represent the given coefficient matrix ($\mathbf{w}_{\mathrm{c},\ell} \in \mathbb{R}^{n_\mathrm{p} \times n_\mathrm{o}}$) in a smaller subspace, and hence to increase computational efficiency while preventing information loss [194–198].

5.1.3 Spatial decomposition

As aforementioned in Section 1.1 a large range of spatial decomposition approaches are available. From the linear orthogonal decomposition POD [20–23] to its nonlinear equivalent kernel-PCA [50–52] or the further generalized PGD [53, 54]. Additionally, functional time series may be decomposed by means of dynamic mode decomposition [57] or methods alike. All mentioned approaches pursue the goal of identifying eigenmodes in the spatial domain, also known as information redundancy, and subsequently prescribe a reduced subspace that remains capable to describe the spatial variance. In this thesis the reduced space is found by use of the relatively straightforward linear POD approximation. Although the linear assumption is not appropriate for this type of problem, the decomposition is further augmented by an unknown modeling error that is identified.

Proper orthogonal decomposition

The first step towards the decomposition of the spatial dependency into an approximation in a smaller subspace is to first collect the FEM solutions in all n_o elements or nodes for the number of stochastic samples n_d at each time step $\ell = 1, ..., n_\mathrm{t}$. The collection then reads $\mathbf{y} \in \mathbb{R}^{(n_\mathrm{d} n_\mathrm{t}) \times n_\mathrm{o}}$. The corresponding output is normalized to a zero mean and unit variance. In order to find an orthogonal vector combination that spans the maximum covariance $\mathbf{\Sigma}_\mathrm{y} \in \mathbb{R}^{n_\mathrm{o} \times n_\mathrm{o}}$ of the QoI, one may look for an n_o-dimensional collection of eigenvectors $\mathbf{v}$ that satisfies

$$\mathrm{argmax}_{\|\mathbf{v}\|=1} \left[\mathbf{v}^T \mathbf{\Sigma}_\mathrm{y} \mathbf{v} \right], \tag{5.27}$$

where $\|\mathbf{v}\| \equiv \mathbf{v}\mathbf{v}^T$. One way to find $\mathbf{v}$ is by applying the singular value decomposition (SVD) [20–22, 199] to $\mathbf{y}$, providing $\mathbf{u}$, $\boldsymbol{s}$ and $\mathbf{v}$ according to:

$$\mathbf{y} = \mathbf{u}(\boldsymbol{s}^\lambda \odot \mathbf{I})\mathbf{v}^T, \tag{5.28}$$

where $\mathbf{u} \in \mathbb{R}^{(n_\mathrm{d} n_\mathrm{t}) \times (n_\mathrm{d} n_\mathrm{t})}$ is the unitary matrix, $(\boldsymbol{s}^\lambda \odot \mathbf{I}) \in \mathbb{R}^{(n_\mathrm{d} n_\mathrm{t}) \times n_\mathrm{o}}$ is the rectangular diagonal matrix of singular values and $\mathbf{v} \in \mathbb{R}^{n_\mathrm{o} \times n_\mathrm{o}}$ is the matrix of eigenvectors. The

covariance matrix $\boldsymbol{\Sigma}_{\mathrm{y}} \in \mathbb{R}^{n_{\mathrm{o}} \times n_{\mathrm{o}}}$ is then given by:

$$\boldsymbol{\Sigma}_{\mathrm{y}} = \mathbf{v} \frac{(\boldsymbol{s}^{\lambda} \odot \mathbf{I})^2}{(n_{\mathrm{d}} n_{\mathrm{t}}) - 1} \mathbf{v}^T. \tag{5.29}$$

Truncation to n_{pc} principal components where $n_{\mathrm{pc}} \ll n_{\mathrm{o}}$ is achieved by selecting the first n_{pc} eigenvectors giving $\underset{\sim}{\mathbf{v}} \in \mathbb{R}^{n_{\mathrm{o}} \times n_{\mathrm{pc}}}$. Following this one may decouple the uncertain from the spatial part by defining:

$$\underset{\sim}{\mathbf{y}}_{\ell}(\omega, \tau_{\ell}) = \mathbf{y}_{\ell}(\omega, x, \tau_{\ell}) \underset{\sim}{\mathbf{v}}(x), \quad \ell = 1, ..., n_{\mathrm{t}}, \tag{5.30}$$

in which $\underset{\sim}{\mathbf{y}}_{\ell} \in \mathbb{R}^{n_{\mathrm{b}} \times n_{\mathrm{pc}}}$ represents the reduced approximation of the QoI $\mathbf{y}_{\ell} \in \mathbb{R}^{n_{\mathrm{b}} \times n_{\mathrm{o}}}$ for $n_{\mathrm{pc}} < n_{\mathrm{o}}$.

As the coefficients $\underset{\sim}{\mathbf{y}}_{\ell}$ in Eq. (5.30) are uncertain, one may approximate them by a the multi-element PCE approach presented in Eq. (5.10) to obtain:

$$\hat{\underset{\sim}{y}}_{\ell}(\omega) = \sum_{m=1}^{n_{\mathrm{c}}} \underset{\sim}{y}_{\ell}^{m}(\omega) I_{\mathcal{D}_m} \approx \sum_{m=1}^{n_{\mathrm{c}}} \sum_{\alpha_m \in \mathcal{J}_m} \underset{\sim}{y}_{\ell,m}^{(\alpha_m)}(\tau_{\ell}) \boldsymbol{\Psi}^{(\alpha_m)}(\boldsymbol{\xi}(\omega)) I_{\mathcal{D}_m}. \tag{5.31}$$

The previous equation can be written in a matrix-vector form:

$$\hat{\underset{\sim}{\mathbf{y}}}_{\ell} \approx \sum_{m}^{n_{\mathrm{c}}} \boldsymbol{\Psi}^{(m)} \underset{\sim}{\mathbf{w}}_{\mathrm{c},\ell}^{(m)} I_{\mathcal{D}_m}, \tag{5.32}$$

given the reduced set of coefficients $\underset{\sim}{\mathbf{w}}_{\mathrm{c},\ell}^{(m)} \in \mathbb{R}^{n_{\mathrm{p}} \times n_{\mathrm{pc}}}$ and the approximation of $\underset{\sim}{\mathbf{y}}_{\ell} \in \mathbb{R}^{n_{\mathrm{b}} \times n_{\mathrm{pc}}}$ by $\hat{\underset{\sim}{\mathbf{y}}}_{\ell} \in \mathbb{R}^{n_{\mathrm{b}} \times n_{\mathrm{pc}}}$. The coefficients can be further estimated as in Eq. (5.23) by the use of the ARD procedure. Finally, the QoI approximation obtained by combination of POD and PCE reads:

$$\hat{\mathbf{y}}_{\ell} \approx \left(\sum_{m}^{n_{\mathrm{c}}} \boldsymbol{\Psi}^{(m)} \hat{\underset{\sim}{\mathbf{w}}}_{\mathrm{c},\ell}^{(m)} I_{\mathcal{D}_m} \right) \underset{\sim}{\mathbf{v}}^T, \quad \ell = 1, ..., n_{\mathrm{t}}. \tag{5.33}$$

A visual representation of Eq. (5.32) for a single element m in a NN format is given in Fig. 5.3. Starting on the left side with the input parameter set $\boldsymbol{q}$ consisting of n_{f} independent variables one may model each parameter as a random variable. Accordingly, a truncated multi-index polynomial set $\boldsymbol{\Psi}(\boldsymbol{\xi}^{(m)})$ can be defined. By weighting the multi-index set for every time step ℓ individually one obtains an approximation of the reduced output $\underset{\sim}{\boldsymbol{z}} := [..., \hat{\underset{\sim}{y}}_{\ell,k}, ...], k = 1, ..., n_{\mathrm{pc}}$.

Despite the reduction of $\mathbf{y}_{\ell}$ to $\underset{\sim}{\mathbf{y}}_{\ell}$, the (reduced) set of PCE coefficients, the approximation still remains to be defined for every time step ℓ individually. For a large number of time steps, the set of coefficients may therefore still grow excessively large, and thus their storing becomes memory intensive. Next to this, one may explore the time dependency between solutions in a similar manner as in the spatial domain. However, in contrast to spatial dependency that is often linear, the time dependency can be strongly nonlinear. This makes POD approximation in time domain $\mathcal{T}_{n_{\mathrm{t}}}$ unsuitable. A solution can be found by approximating the time-dependency of $\underset{\sim}{\mathbf{y}}_{\ell}$ (or $\mathbf{y}_{\ell}$) using RNNs, where in this work specific interest is devoted to ARD-LSTM NNs.

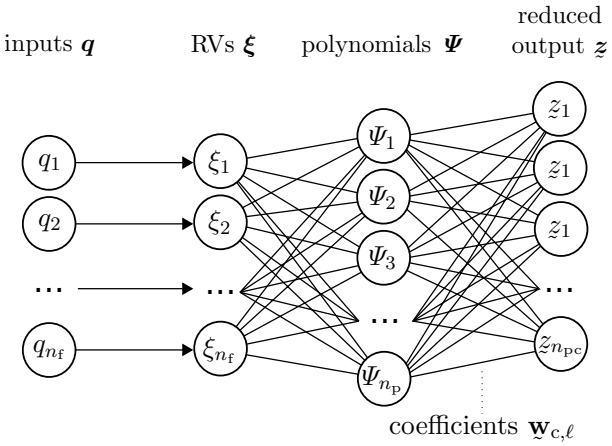


Figure 5.3: Feed-forward NN point of view on the PCE-POD representation of the QoI.

5.1.4 Full separate representation of the quantity of interest

Starting from Eq. (5.31) one may approximate the unknown time dependent local PCE coefficients $\underset{\sim}{y}_{\ell,m}^{(\alpha_m)}$ by the ARD-LSTM NN:

$$\underset{\sim}{y}_{\ell,m}^{(\alpha_m)} = \sigma_{\mathrm{y},m}^{\ell,(\alpha_m)} \circ \sigma_{\mathrm{h},m}^{\ell,(\alpha_m)} \circ ... \circ \sigma_{\mathrm{h},m}^{1,(\alpha_m)}, \tag{5.34}$$

in which $\sigma_{\mathrm{y},m}^{\ell,(\alpha_m)}$ is the functional description of the ARD-LSTM output layer at time step ℓ and for stochastic element m, see Chapter 4. The arising fully separated multi-element ARD-PCE-POD-LSTM model is further abbreviated as RED-LSTM (aRd-pcE-poD-LSTM). In this manner, the time dependency is taken into account. As ARD-LSTM identifies the modeling error in $\mathbf{y}_\ell$ (see Eq. (4.73) and Algorithm 1) the correction of the POD error is also accounted for.

A visual representation of Eq. (5.33) in the form of a NN architecture is given in Fig. 5.4 for time step ℓ. In this figure the input parameters are mapped to base variables which are then mapped to the PCE orthogonal polynomials. These are further mapped to reduced output QoI $\underset{\sim}{\boldsymbol{z}} := [..., \underset{\sim}{\hat{y}}_{\ell,k}, ...], k = 1, ..., n_{\mathrm{pc}}$. The mapping coefficients in the last layer are approximated by an LSTM NN.

5.2 Structural application

In this section is considered the structural application of the previously presented theory on the separate representation of the displacement QoI of the tapered tensile specimen experiment, aforementioned in Section 3.5.1, is considered. In contrast to the earlier numerical analysis presented in Chapter 3 and Chapter 4 in which the effect of the number of designs and their location on the accuracy of the solution is evaluated, in this section the analysis is focused on the evaluation of domain representation and accompanying low-rank sparse approximations.

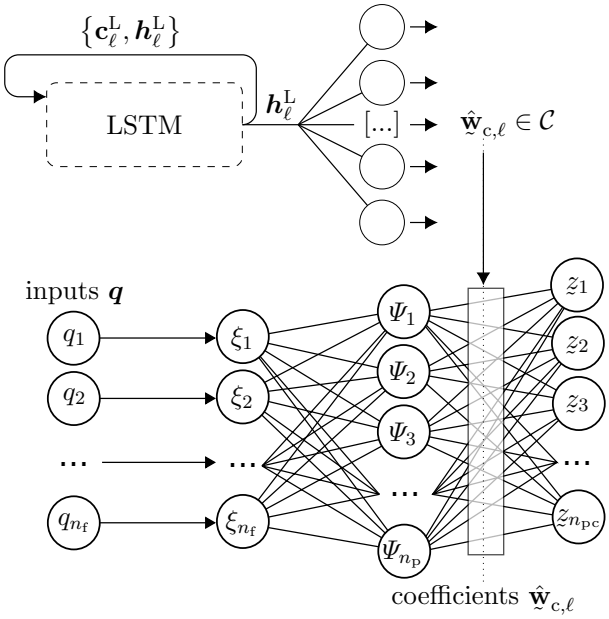


Figure 5.4: A NN point of view of the full separate representation of the QoI.

5.2.1 Tapered tensile specimen

For computational simplicity, the aforementioned tapered tensile specimen (see Section 3.5.1) is first simplified with a relatively coarse shell element-based model ($l_c = 4.0\,\mathrm{mm}$) with 282 quadraliteral elements, see Fig. 5.5.

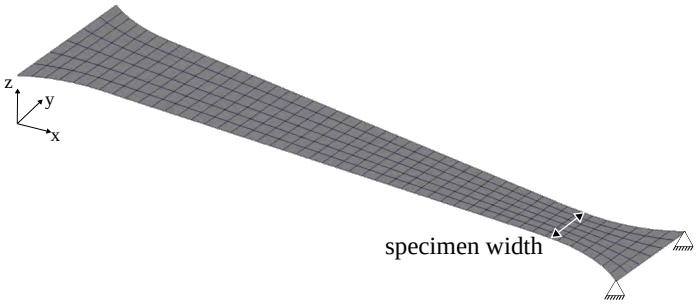
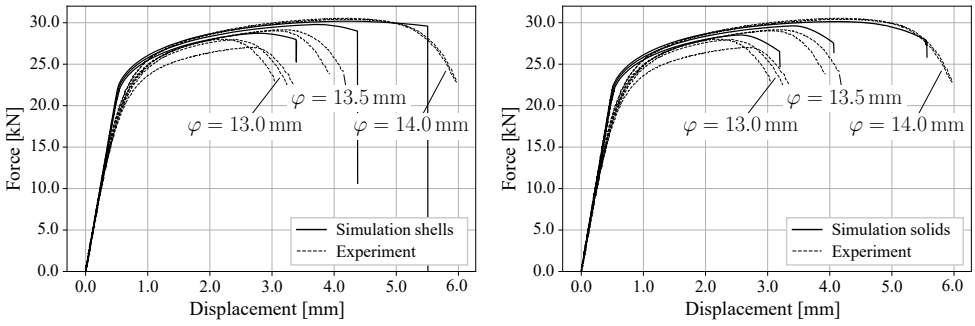


Figure 5.5: Tapered tensile specimen with 282 quadraliteral shell elements, parametrized by the specimen width φ at the narrow side of the gauge section. The specimen is fixed on the narrow side and a tensile deformation is applied in the $-x$ direction.

Additionally, the more detailed and experimentally validated hexagonal solid-element based model is used to evaluate the proposed domain representation on a larger data scale. The 2017 version of the explicit finite element solver VPS is again considered [126]. In Fig. 5.6 the force-displacement curves of the experiments performed in [1, 3]

and the corresponding curves for both shell- and solid-based simulations are set out for different specimen widths $\varphi = \{13.0\,\text{mm}, 13.5\,\text{mm}, 14.0\,\text{mm}\}$. Although the shell-based results show a relatively large deviation due to an increased elemental stiffness, the solid-based simulation shows very similar behavior compared to the experimentally obtained curves, equivalently to the result found in Section 4.2.



(a) $n_\text{e} = 282$ shells. **(b)** $n_\text{e} = 34\,809$ solids.

Figure 5.6: Validation of the adapted shell element-based and solid element-based model using VPS 2017 and experiments at widths $\varphi = \{13.0\,\text{mm}, 13.5\,\text{mm}, 14.0\,\text{mm}\}$ with the experimental results taken from [1, 3].

At first, a base polynomial chaos-based surrogate model is created for multiple polynomial degrees. This analysis excludes any low-rank approximation by POD and ARD-LSTM. Additionally, this base model excludes the aforementioned multi-element approach and hence only models a traditional polynomial chaos architecture, optimized in an ARD sense. For ARD-PCE, the coefficients are pruned for $[\boldsymbol{\gamma}_\ell]_{jk} \leq 1\text{e}{-4}$ whereas $[\boldsymbol{\alpha}_\ell]_{jk} \in [1\text{e}1, 1\text{e}6]$ and $[\boldsymbol{\beta}_\ell]_k \in [1\text{e}4, 1\text{e}6]$, with an equal hyperparameter setup for the weights as in the approach used for ARD-LSTM, see Section 4.2. With finite element simulation taken to collect the data set for the QoI, the assumed modeling error is relatively small, yielding relatively high value bounds for $[\boldsymbol{\beta}_\ell]_k$.

Subsequently, this polynomial chaos framework is applied to various predefined numbers of stochastic elements for the input random parameters $\boldsymbol{q} \in \mathbb{R}^{n_\text{f}}$, where the element boundaries are optimized accordingly. With the results for the multi-element approach evaluated, the low rank approximation by means of ARD-LSTM and POD in each model is then further considered.

Base model

The shell-based tapered tensile specimen width $\varphi \in [13.0\,\text{mm}, 16.0\,\text{mm}]$ (uniform distribution) is sampled $n_\text{d} = 200$ times by the Latin Hypercube procedure [135]. With the given number of time steps $n_\text{t} = 21$ the complete data set of the QoI (x-displacement) belongs to $\mathbb{R}^{n_\text{d} \times n_\text{t} \times n_\text{n}} = \mathbb{R}^{200 \times 21 \times 336}$, in which $n_\text{n} = 336$ denotes the number of FEM nodes.

In order to assess the surrogate models suitability, a total of three critical nodes ($n_\text{id} \in \{1, 2, 3\}$, see Fig. 5.7) are selected: two at both sides of the right bifurcation

and one at the center of the gauge section. In Fig. 5.7 probability density functions are shown of the final x-deformation of the mentioned nodes. For this purpose a Gaussian kernel density estimation is used with Scotts rule applied for bandwidth computation [200].

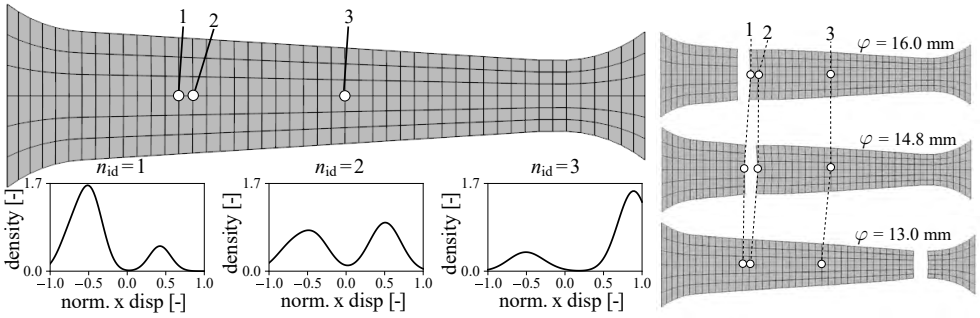


Figure 5.7: Top view of the shell-based tapered tensile specimen with kernel density estimations of the x-displacement.

The x-deformation is modeled by an ARD-PCE, see Section 5.1.2 model with $n_g \in \{1, 2, 3, 4\}$ polynomial degrees for Legendre polynomials. With the ARD-training procedure the estimated mean and covariance of the deformation PCE coefficients (see Eq. (5.24)) converge after 50 iterations of the algorithm. The coefficient of determination R^2 for $n_g \in \{1, 2, 3, 4\}$ is given in Table 5.1. As can be seen in

Table 5.1: Coefficient of determination R^2 for all considered base models for $n_g \in \{1, 2, 3, 4\}$, analyzed in $n_{id} \in \{1, 2, 3\}$.

n_{id}	1				2				3			
n_g	1	2	3	4	1	2	3	4	1	2	3	4
R^2	0.971	0.983	0.990	0.990	0.972	0.975	0.984	0.984	0.661	0.884	0.917	0.921

Table 5.1 the coverage of data variance does not (significantly) improve for all nodes by increasing the polynomial order n_g beyond the second order. Hence, despite the increase in descriptive complexity for a larger n_g, the model fails to properly describe the variance covered by the provided data samples. As $R^2 = 1.0$ is desired, one may see that the classical ARD-PCE approach is not able to provide a satisfying result.

Multi-element ARD-PCE

To discretize the input random variable φ, i.e. the width of the tensile specimen, into a fixed number of n_c stochastic elements the Nelder-Mead optimization is used, see Algorithm 2. The number of elements is kept fixed with different options $n_c = \{2, 3, 4\}$, whereas the boundaries of the elements are optimized. Furthermore, each element is approximated by its local ARD-PCE model of the order $n_p = 3$. While estimating the PCE coefficients it is made sure that each element has enough samples, i.e. the minimum number of Latin Hypercube sampling points in each segment is taken to be 5% of the total number of data points describing the input variable.

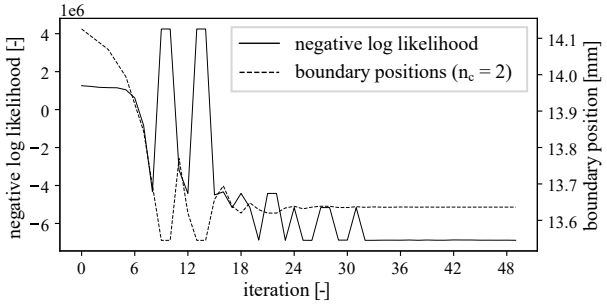
In Fig. 5.8 for all stochastic elements $n_c = \{2, 3, 4\}$ the negative log likelihood of the PCE estimated hyperparameters is plotted over all iterations of the alternating optimization procedure presented in Algorithm 2. The multi-element boundary optimization procedure is performed over a maximum of 100 iterations, or until convergence ($\mathcal{L}_{tol} = 1e{-}4$), starting with boundary values sampled from a uniform distribution as indicated in Algorithm 2. The boundary positions (in terms of the input value φ) obtained from the Nelder-Mead optimization procedure for $n_c = 2$ elements are $[13.00\,\text{mm}, 13.64\,\text{mm}]$ and $[13.64\,\text{mm}, 16.00\,\text{mm}]$. For $n_c = 3$ elements the boundaries are given by $[13.00\,\text{mm}, 13.63\,\text{mm}]$, $[13.63\,\text{mm}, 13.90\,\text{mm}]$ and $[13.90\,\text{mm}, 16.00\,\text{mm}]$. In a similar sense, for $n_c = 4$ elements the boundaries obtained from the optimization are $[13.00\,\text{mm}, 13.63\,\text{mm}]$, $[13.63\,\text{mm}, 14.66\,\text{mm}]$, $[14.66\,\text{mm}, 15.20\,\text{mm}]$ and $[15.20\,\text{mm}, 16.00\,\text{mm}]$.

With a varying n_c, the corresponding optimum boundary positions may also change. Hence, when increasing the number of elements, one does not necessarily obtain the same boundary positions. The first boundary for $n_c = 2$ at $\varphi = 13.63\,\text{mm}$ corresponds to the point of bifurcation, where the location of fracture leaps from the narrow side of the bifurcation to the wide side, or vice versa (Fig. 5.7). This corresponds to bimodality of the maximum deformation probability density function as shown in Fig. 5.7. The remainder of boundary positions are found to correlate to minor shifts in fracture location by a single element row for an increased specimen width φ, see Fig. 5.7.

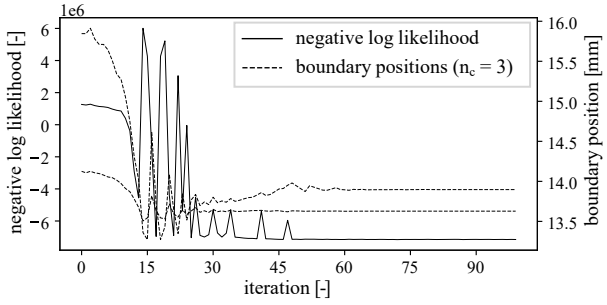
In Table 5.2 the R^2-values are presented for the node indices $n_{id} \in \{1, 2, 3\}$ with the polynomial degree $n_g \in \{1, 2, 3, 4\}$ on $n_c = \{2, 3, 4\}$ elements. By subdividing each input random variable into two random variables, the overall predictive accuracy given a fixed degree n_g improves. A significant increase in the coefficient of determination R^2 is found for $n_{id} = 3$ and polynomial degrees $n_g \in \{1, 2, 3, 4\}$ compared to the values found in Table 5.1.

One may note that increasing either n_g or n_c results in a general increase of the coefficient of determination for the data samples provided. The decision which of n_g or n_c to keep small strongly depends on the number of input parameters n_f and on the size of the spatial domain n_o of the QoI as well as on the computational resources available. The effect of these dependencies may be found in the size of the coefficient matrix $\mathbf{w}_{c,\ell}$ of the corresponding ARD-PCE model. Hence, in Table 5.2 the number of variables n_v in the ARD-PCE coefficients $\mathbf{w}_{c,\ell}$ over all time steps is shown for reference. According to the number of random variables n_v shown in Table 5.2, one may argue that $R^2 \geq 0.99\ \forall n_{id} \in \{1, 2, 3\}$ is first achieved for the polynomial degree $n_g = 1$ and $n_c = 4$ elements and thus this combination is selected as a proper choice of model. However, $n_g = 1$ provides a purely linear system over all elements and thus does not represent the probability density function of the response well. The increase in R^2 found for $n_g = 2$ and $n_c = 4$ indicates the presence of non-linearity in the mapping $\mathbf{q} \rightarrow \mathbf{y}_\ell$, requiring $n_g \geq 2$ for the proper local PCE model selection.

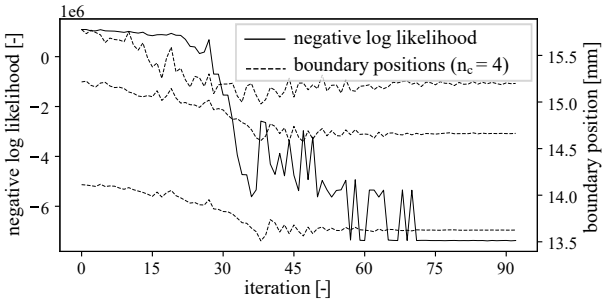
For the relatively small shell-based example, a significant increase in modeling accuracy has been found by using the multi-element approach. However, for an increasing n_g or n_c the number of variables n_v significantly grows (Table 5.2).



(a) $n_\mathrm{c} = 2$



(b) $n_\mathrm{c} = 3$



(c) $n_\mathrm{c} = 4$

Figure 5.8: Multi-element PCE boundary optimization procedure.

Multi-element low-rank ARD-PCE approximation

The data samples for the solid-element based tapered tensile specimen are created in an equal manner as for the shell-based approximation. Thus, $n_\mathrm{d} = 200$ Latin Hypercube samples are used to sample the uniform specimen width $\varphi \in [13.0\,\mathrm{mm}, 16.0\,\mathrm{mm}]$. With the total number of nodes $n_\mathrm{n} = 47\,712$ and the number of time steps $n_\mathrm{t} = 21$ this yields the data set belongs to $\mathbb{R}^{200\times21\times47712}$. The Nelder-Mead optimization of the multi-element boundaries (see Eq. (5.13)) is set up analogously to the one applied in the shell-based example. The boundaries (in terms of the input value φ) are found using the Nelder-Mead optimization. For $n_\mathrm{c} = 2$ elements are $[13.00\,\mathrm{mm}, 13.69\,\mathrm{mm}]$ and

Table 5.2: Coefficient of determination R^2 for all considered base models for $n_g \in \{1,2,3,4\}$, analyzed in $n_{id} \in \{1,2,3\}$, evaluated over the shell-based tapered tensile specimen.

n_{id}	1											
n_g	1			2			3			4		
n_c	2	3	4	2	3	4	2	3	4	2	3	4
R^2	0.974	0.978	0.993	0.989	0.990	0.995	0.991	0.991	0.996	0.991	0.992	0.997

n_{id}	2											
n_g	1			2			3			4		
n_c	2	3	4	2	3	4	2	3	4	2	3	4
R^2	0.976	0.976	0.990	0.981	0.986	0.994	0.986	0.987	0.996	0.990	0.991	0.997

n_{id}	3											
n_g	1			2			3			4		
n_c	2	3	4	2	3	4	2	3	4	2	3	4
R^2	0.994	0.995	0.995	0.996	0.996	0.997	0.996	0.997	0.997	0.997	0.997	0.997

n_v	2.8e4	4.2e4	5.6e4	4.2e4	6.3e4	8.5e4	5.6e4	8.5e4	11e4	7.1e4	11e4	14e4

[13.69 mm, 16.00 mm]. Accordingly, for $n_c = 3$ elements the boundaries are given by [13.00 mm, 13.69 mm], [13.69 mm, 14.00 mm] and [14.00 mm, 16.00 mm] and for $n_c = 4$ elements the boundaries obtained from the optimization are [13.00 mm, 13.69 mm], [13.69 mm, 14.60 mm], [14.60 mm, 15.19 mm], and [15.19 mm, 16.00 mm]. Similarly to the nodes selected for the shell-based example, three representative FEM nodes are chosen to evaluate the multi-element ARD-PCE model, see Fig. 5.9. The selected nodes $n_{id} \in \{1,2,3\}$ shown in Fig. 5.9 are chosen at two sides of the right bifurcation and at the center of the gauge section of the solid-based tapered tensile specimen. Alike for Fig. 5.7 the shown Gaussian kernel density estimations (KDE) for the x-displacement of selected notes are computed over all $n_d = 200$ designs at the final time step.

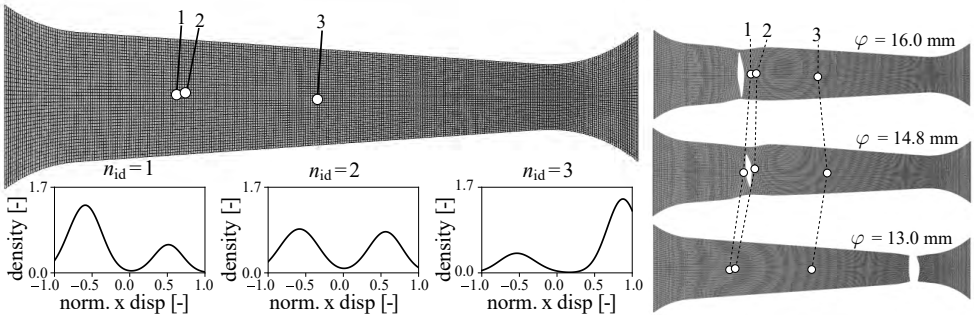


Figure 5.9: Solid-based tapered tensile specimen with Gaussian KDEs for the x-displacement of three selected nodes at the final time step.

The R^2-values of the ARD-PCE model for $n_{\mathrm{id}} = \{1, 2, 3\}$ are presented in Table 5.3, analogously to Table 5.2 to indicate the accuracy of the model over the provided designs given a varying polynomial degree n_{g} and number of elements n_{c}.

Table 5.3: Coefficient of determination R^2 for all considered base models for $n_{\mathrm{g}} \in \{1, 2, 3, 4\}$, analyzed in $n_{\mathrm{id}} \in \{1, 2, 3\}$, evaluated over the tapered tensile specimen approximated by solid based FEM elements.

n_{id}	1											
n_{g}		1			2			3			4	
n_{c}	2	3	4	2	3	4	2	3	4	2	3	4
R^2 ARD-PCE	0.974	0.981	0.998	0.986	0.986	0.999	0.988	0.991	0.999	0.994	0.994	0.999
R^2 RED-LSTM	0.974	0.981	0.998	0.986	0.986	0.999	0.988	0.991	0.999	0.994	0.994	0.999

n_{id}	2											
n_{g}		1			2			3			4	
n_{c}	2	3	4	2	3	4	2	3	4	2	3	4
R^2 ARD-PCE	0.976	0.977	0.997	0.980	0.989	0.998	0.989	0.990	0.999	0.992	0.995	0.999
R^2 RED-LSTM	0.976	0.976	0.997	0.980	0.989	0.998	0.989	0.990	0.999	0.992	0.995	0.999

n_{id}	3											
n_{g}		1			2			3			4	
n_{c}	2	3	4	2	3	4	2	3	4	2	3	4
R^2 ARD-PCE	0.998	0.998	0.998	0.999	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
R^2 RED-LSTM	0.998	0.998	0.998	0.999	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000

n_{v} ARD-PCE	4.0e6	6.0e6	8.0e6	6.0e6	9.0e6	12e6	8.0e6	12e6	16e6	10e6	15e6	20e6
n_{v} RED-LSTM	4.2e3	6.3e3	8.4e3	6.3e3	9.5e3	13e3	8.4e3	13e3	17e3	11e3	16e3	21e3

Furthermore, after pure PCE approximation the next step is to approximate the QoI by using the POD+PCE approach as explained in Section 5.1.3 and truncated at $n_{\mathrm{pc}} = 50$ principal components. For a full separate representation of the QoI (x-displacement) one may further employ ARD-LSTM to predict the ARD-PCE coefficients over time, as elaborated in Section 5.1.4. Thus, a distinction between ARD-PCE and RED-LSTM is made in Table 5.3. For the RED-LSTM model the number of units representing the size of the LSTM cell state is $n_{\mathrm{m}} = 4$. The ARD-LSTM framework used in RED-LSTM is optimized over 50 epochs.

The R^2 of RED-LSTM for previously explained variations in n_{g} and n_{c} is given in Table 5.3. In the table it is shown that RED-LSTM is capable to estimate the coefficients $\mathbf{w}_{\mathrm{c},\ell}\mathbf{w}_{\mathrm{c},\ell}$ without any noticeable loss. The results show insignificant reduction in R^2 compared to the multi-element ARD-PCE. Yet, RED-LSTM strongly reduces the number of variables n_{v} in the coefficient matrix. The predictive accuracy for $n_{\mathrm{id}} = 3$, located in the center of the gauge section, is found to quickly converge to $R^2 = 1.000$, first reaching this value for $n_{\mathrm{c}} = 3$ and $n_{\mathrm{g}} = 2$.

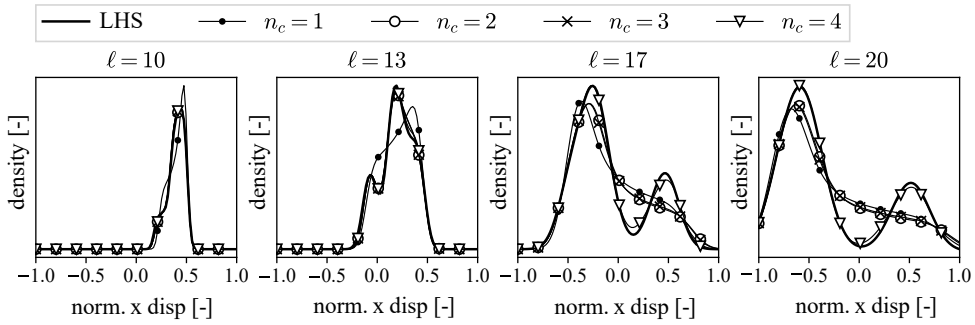
As mentioned for the shell-based example, the increase in R^2 when increasing the polynomial degree $n_{\mathrm{g}} = 1 \rightarrow 2$ indicates the presence of non-linearity in the map between the input parameter and the QoI. Hence, the choice of model is constrained by $n_{\mathrm{g}} \geq 2$. For input randomness being approximated by $n_{\mathrm{g}} = 2$ and $n_{\mathrm{c}} \in \{1, 2, 3, 4\}$ elements,

and the map between the input and output being approximated by RED-LSTM, the corresponding probability density functions of the normalized x-displacement for different time steps are shown in Fig. 5.10. The density functions of the x-displacement are plotted using Gaussian kernel density estimations with 100 linearly spread samples in the range $[-1, 1]$ for every selected node $n_{\mathrm{id}} \in \{1, 2, 3\}$ given four time steps ($\ell \in \{10, 13, 17, 20\}$) along the sequence. The KDE obtained from the aforementioned $n_{\mathrm{d}} = 200$ pure Latin Hypercube samples (LHS) used to create the QoI data set (see Section 5.2.1) is added as a reference. For any $n_{\mathrm{c}} > 1$ a strong increase of the descriptive quality of the RED-LSTM model on $n_{\mathrm{id}} \in \{1, 2, 3\}$ is found. Although the number of elements $n_{\mathrm{c}} = 2$ is sufficient for $n_{\mathrm{id}} = 3$ ($\mathrm{R}^2 = 0.999$), at least $n_{\mathrm{c}} = 4$ is required to ensure that the model properly describes the KDE at the place of the nodes with indicator $n_{\mathrm{id}} = 1$ and $n_{\mathrm{id}} = 2$.

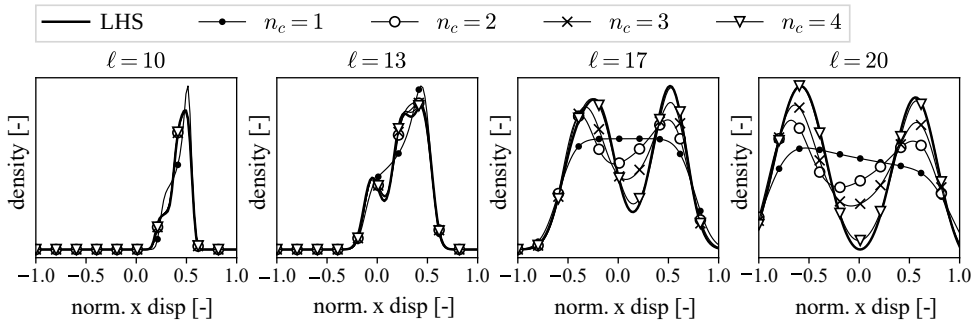
To evaluate the predictive uncertainty of the RED-LSTM model, one may apply Eq. (5.26) and Eq. (5.33). In Fig. 5.11 the predictive uncertainty at the final time step n_{t} is given of the x-displacement at the nodes described by the indicators $n_{\mathrm{id}} \in \{1, 2, 3\}$. For the approximation the RED-LSTM model is used with polynomial degree $n_{\mathrm{g}} = 2$ and input randomness described by $n_{\mathrm{c}} = 4$ elements. Furthermore, the RED-LSTM model used for Fig. 5.11a, Fig. 5.11c and Fig. 5.11e is trained with all aforementioned $n_{\mathrm{d}} = 200$ data samples. To give more insight in the predictive variance of the RED-LSTM model all data samples in the critical range $\varphi \in [13.3\,\mathrm{mm}, 14.0\,\mathrm{mm}]$ are disregarded in training the RED-LSTM model, see Fig. 5.11b, Fig. 5.11d and Fig. 5.11f. In this parameter range, the bifurcation was found to switch from the narrow side to the wider side of the gauge section, or vice versa (see Fig. 5.9). In the figures the predictive mean $\hat{\bar{y}}$ is shown at the final time step over the variation range of parameter φ. Next to this, the shaded area represents $(Q(5\%), Q(95\%))$ quantiles and the surrounded area represents $(Q(1\%), Q(99\%))$ quantiles. Furthermore, the vertical lines are used to indicate the boundaries of the stochastic elements. These plots are compared to the sampled mean obtained by LHS sampling method for all available designs.

In Fig. 5.11c one may note a strong deviation between the predicted mean displacement and the purely LHS (thus the QoI data set used to train the RED-LSTM model) mean for the input random variable in the range $14.60\,\mathrm{mm} \leq \varphi \leq 15.20\,\mathrm{mm}$. This deviation is also visible in Fig. 5.10b and is caused by a slight mismatch between the stochastic element boundary and the abrupt change in deformation around $\varphi \approx 14.60\,\mathrm{mm}$, see Fig. 5.11c. The RED-LSTM model therefore shows a curvature that tries to capture data samples at the left side of the stochastic element that has a much lower mean value compared to the rest of the stochastic element. A similar phenomenon is shown in Fig. 5.11a, where a strong deviation is found between the predicted mean x-displacement and the purely LHS mean for the input random variable in the range $15.20\,\mathrm{mm} \leq \varphi \leq 16.00\,\mathrm{mm}$. Here, the left stochastic element boundary has a different mean compared to the rest of this stochastic element.

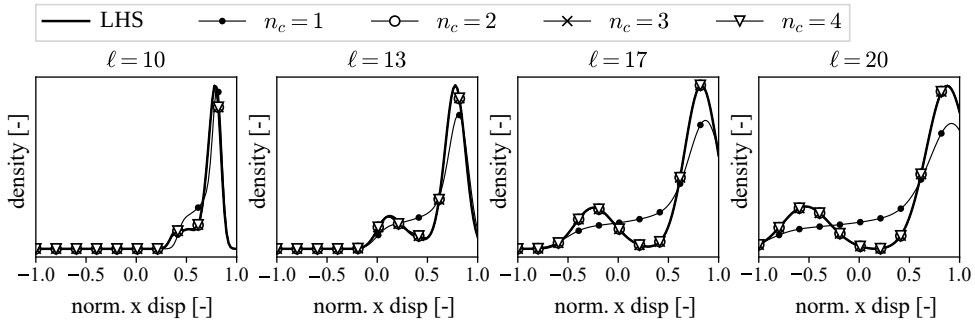
As $n_{\mathrm{d}} = 200$ data samples sufficiently fill the parameter space, the predictive uncertainty –shown using the $(Q(5\%), Q(95\%))$ quantiles– is relatively constant over every stochastic element as can be seen in Fig. 5.11a, Fig. 5.11c and Fig. 5.11e. As a di-



(a) $n_{\mathrm{id}} = 1$.



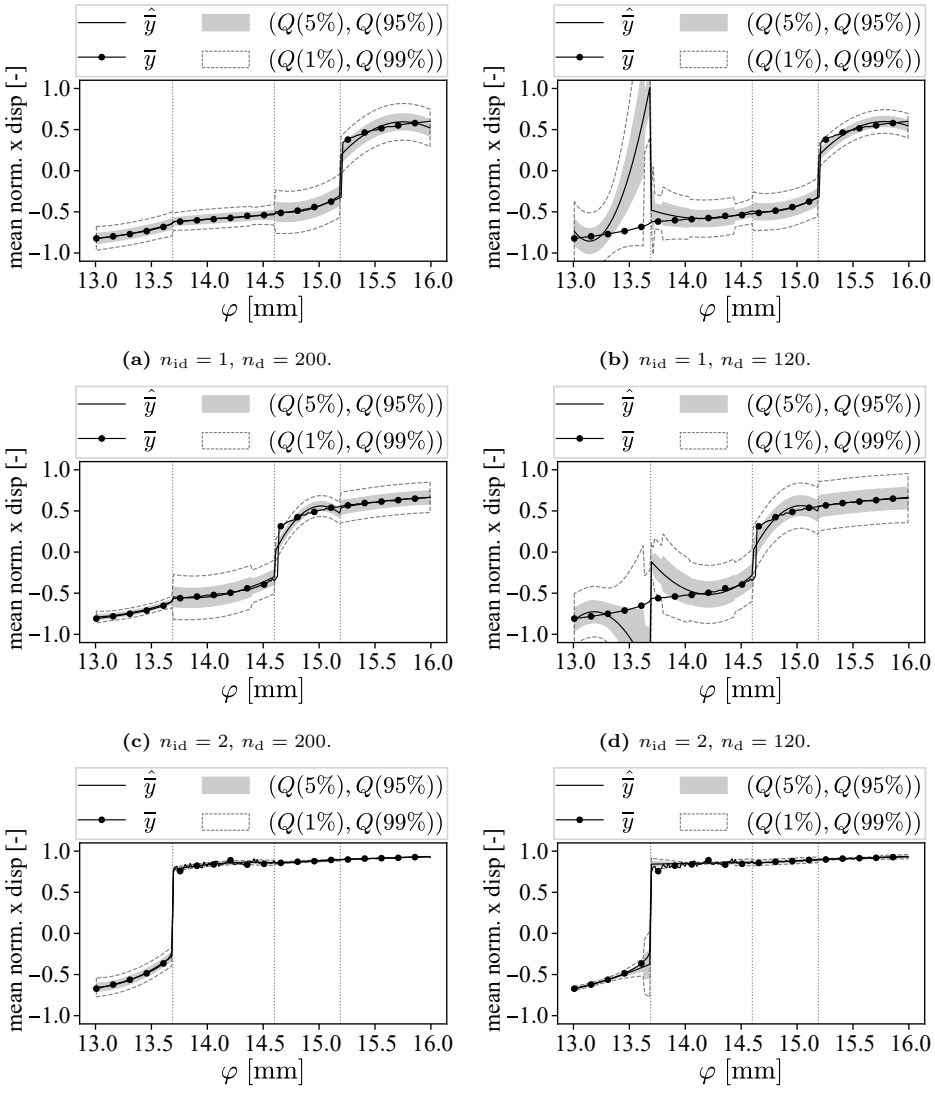
(b) $n_{\mathrm{id}} = 2$.



(c) $n_{\mathrm{id}} = 3$.

Figure 5.10: Gaussian kernel density estimations for $n_{\mathrm{id}} \in \{1, 2, 3\}$ and $n_{\mathrm{c}} \in \{1, 2, 3, 4\}$ with $n_{\mathrm{g}} = 2$.

rect result of removing all data samples in the critical range $\varphi \in [13.3\,\mathrm{mm}, 14.0\,\mathrm{mm}]$ the predictive uncertainty in this region (see Fig. 5.11b, Fig. 5.11d and Fig. 5.11f) increases significantly, accompanied by a faulty mean prediction close to the bifurcation around $\varphi \approx 13.69\,\mathrm{mm}$, indicating that the sample space must be better filled in this region. Although the mean prediction may deviate in the untrained region, its uncertainty well captures the true DOE-given mean by the $(Q(1\%), Q(99\%))$ quantiles on all $n_{\mathrm{id}} \in \{1, 2, 3\}$.



(a) $n_{\mathrm{id}} = 1$, $n_{\mathrm{d}} = 200$.

(b) $n_{\mathrm{id}} = 1$, $n_{\mathrm{d}} = 120$.

(c) $n_{\mathrm{id}} = 2$, $n_{\mathrm{d}} = 200$.

(d) $n_{\mathrm{id}} = 2$, $n_{\mathrm{d}} = 120$.

(e) $n_{\mathrm{id}} = 3$, $n_{\mathrm{d}} = 200$.

(f) $n_{\mathrm{id}} = 3$, $n_{\mathrm{d}} = 120$.

Figure 5.11: The Latin Hypercube samples mean $\overline{y}$ and predictive mean $\hat{\overline{y}}$ and the $(Q(5\%), Q(95\%))$ and $(Q(1\%), Q(99\%))$ quantiles at the final time step.

5.3 Computational efficiency of the covariance

The computational cost of the ARD-PCE coefficients (weights) covariance is in the order $\Theta(n_{\mathrm{t}} \times n_{\mathrm{o}}, \times n_{\mathrm{p}}^3)$ and $\Theta(n_{\mathrm{t}} \times n_{\mathrm{pc}}, \times n_{\mathrm{p}}^3)$ for coefficients covariance of the RED-LSTM model. Hence, with a higher polynomial degree and a larger input feature set, the involved computational expenses strongly increase. The application of POD

reduces the computational efficiency only in a linear sense. By using ARD-PCE in a multi-element approach, each sub-ARD-PCE is required to describe a system with reduced complexity. Thus, by the use of stochastic domain separation a lower polynomial degree per stochastic element is generally required, and hence reduces the computational effort in a cubic sense. However, a larger number of stochastic elements increases the computational effort in a linear sense, as each of the stochastic elements is approximated by a separate ARD-PCE (or RED-LSTM) model. The overall benefit of the multi-element approach in terms of computational expenses is therefore a reduction in quadratic rather than cubic sense.

Despite the increase in computational efficiency, the multi-element approach requires optimization of the stochastic element boundaries. In the optimization process one has to estimate the local ARD-PCE coefficients in each iteration and then assemble the global ARD-PCE. The PCE coefficients can be estimated in a mean square sense by inversion (see Eq. (5.8)). Although the introduction of ARD causes the need for an iterative estimation of the coefficients, ARD-PCE (and RED-LSTM) is found to converge by using a small number of iterations. Hence, the computational cost is not increasing drastically. To get a better insight in the computational costs, an overview for the ARD-PCE iterative weight estimation times is given for 50 iterations over the data set under the variation of the polynomial degree $n_\mathrm{g} \in \{1, 2, 3, 4\}$ and the output dimension n_o, see Fig. 5.12.

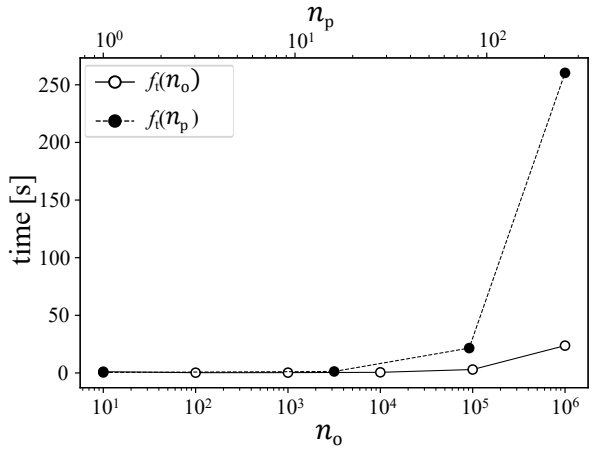


Figure 5.12: ARD-PCE optimization time $f_t(n_\mathrm{o})$ and $f_t(n_\mathrm{g})$.

The figure shows the ARD-PCE optimization time $f_t(n_\mathrm{o})$ as a function of n_o for a set of models with $n_\mathrm{g} = 2$ and the ARD-PCE optimization time $f_t(n_\mathrm{g})$ as a function of n_g for a set of models with $n_\mathrm{o} = 1000$. The analysis is performed using a multivariate example data set $(\boldsymbol{q}_i, \boldsymbol{y}_i), i = 1, ..., n_\mathrm{d}$ with $\boldsymbol{q} \in \mathbb{R}^{n_\mathrm{f}}$ and $\boldsymbol{y} \in \mathbb{R}^{n_\mathrm{o}}$ created by $n_\mathrm{d} = 100$ samples the random input variables from independent uniform distributions $\mathcal{U}(0, 1)$ with $n_\mathrm{f} = 4$ for a single time step ($n_\mathrm{t} = 1$) and a single stochastic element ($n_\mathrm{c} = 1$). The random input variables therefore belong to $\mathbb{R}^{100\times4}$ and the

QoI belongs to $\mathbb{R}^{100\times n_\mathrm{o}}$. Furthermore, with $n_\mathrm{g} \in \{1,2,3,4\}$ and $n_\mathrm{f} = 4$ one obtains $n_\mathrm{p} \in \{4,16,64,256\}$, respectively. The Tensorflow v2.2.0 framework is applied for the optimization on a single Nvidia Quadro P5000 GPU. In Fig. 5.12 one may observe that the polynomial degree n_g significantly affects the estimation time. The exponential relation between the number of multi-index tuplets n_p and n_g may therefore quickly make the ARD-PCE coefficient estimation computationally intractable. By reducing the spatial domain to a subspace ($n_\mathrm{pc} \ll n_\mathrm{o}$) one can partly compensate for this. A further reduction of the computational expenses may be found by applying the multi-element approach and reducing the polynomial degree n_g on each of the stochastic elements where possible.

6 Conclusion & Outlook

In this chapter conclusions are drawn from the obtained results on the developed novel automatic relevance determination scheme for a nonlinear dynamics based FEM analysis. Accordingly, further exploitation of the possible future research directions is given.

In the present work the paradigm change that explains the focus shift from computationally expensive parametrized finite element models towards computationally less demanding surrogate models is exploited. The application domain of the proposed research is vehicle crashworthiness and its possible component optimization with respect to mechanical safety. Hence, the FEM based model used in this thesis is characterized by large nonlinear deformations and a nonlinear material model.

The QoIs considered in this thesis are fracture and displacement, representing key performance indicators in crash worthiness evaluations. The design parameter is taken to be a geometrical feature of the vehicle component, which is further varied in a probabilistic sense. The geometrical parameters and hence the QoIs are further sampled by Latin Hypercube sampling strategies, hence the corresponding virtual (FEM-based) training sets are built assuming a constant mesh topology. In other words, each sample of the QoI given the varying input feature is computed on the same mesh.

The QoI is further approximated by several surrogate approaches. The initial approximation by the very well known deterministic LSTM NNs with subsequent dense layer is shown not to be a suitable choice due to the fixed NN architecture that cannot adapt to the data set. Changing the size of the data set used to train the NN leads to a strongly fluctuating accuracy. Hence, the surrogate model shows a strong sample space dependency, with the employed frameworks being unable to indicate its predictive (un)certainty. Additionally, the chosen network dimension is a priori unknown and modeling therefore becomes more difficult as several LSTM models of different widths have to be considered separately.

Despite previously described issues, the LSTM model has shown the capabilities to predict the evolution of fracture by distinguishing its bifurcation. Therefore, in this thesis a novel approach is proposed, the ARD-LSTM model, which employs an LSTM NN as a suitable mathematical model, but changes the way its weights are trained. Instead of the classical backpropagation algorithm in this thesis a corresponding probabilistic framework is proposed by means of automatic relevance determination. As the latter one can only be used for linear estimation, in this work its nonlinear version is proposed that can deal with nonlinearity of the activation functions in the network. By applying an efficient sparse Bayesian learning framework the LSTM network is trained to be self-organized, i.e. to adapt its architecture optimally. This is achieved by describing the unknown, yet to be estimated NN weights, by a zero-mean Laplace distribution. This distribution enforces sparsity, and hence all weights that do not contribute to the model response are automatically removed from the network. To achieve so, the probabilistic backpropagation algorithm is introduced. In contrast to the classical training, in this algorithm first the so-called internal state of the neural cell is estimated by a modified version of the gradient descent approach. Once the internal states are known the unknown weights are obtained using a Bayesian approach by taking the internal state as measurement. The newly proposed ARD-LSTM method is analyzed on a structural application example by the help of which is shown that the ARD-LSTM framework is very well capable to describe the given data set by adapting the NN complexity accordingly. However, the new approach requires

a probabilistic optimization procedure, which further increases the computational effort when compared to its deterministic equivalent. This is due to the need to propagate probabilistic information through the nonlinear activation functions in the ARD-LSTM gates, which is here done in a Monte Carlo manner. However, the increase of the computational effort is compensated by faster weight convergence compared to the corresponding deterministic framework. Yet the ARD-LSTM network can still face under-fitting as in the classical NN version. Ensuring that the network is able to robustly describe the nonlinear mechanics problem therefore remains of great importance.

Next to the ARD-LSTM model, in this thesis the ARD-PCE model is also investigated. The ARD-PCE model approximates the QoI by polynomial functions of simple random variables describing the input parameter set. In a novel proposed ARD framework PCE coefficients are estimated by an ARD scheme, and hence their sparsity is enforced. This approach is further used to find a low rank approximation of the solution by employing the POD method. When combined with the ARD-LSTM model, the ARD-PCE approximation of the input parameter and the low-rank approximation of the output introduce a new machine learning architecture, the coefficients of which have a physical meaning. These coefficients can then also be estimated in an ARD setting leading to the self-organized RED (aRd-pcE-poD)-LSTM architecture.

The applied ARD-PCE and accompanying RED-LSTM surrogates are found to converge relatively fast ($<$50 epochs), thus minimizing computational overhead. To handle the inability of the PCE framework to properly describe multi-modal QoIs, a multi-element approach has been introduced. The newly proposed approach is analyzed on a structural application example, by the help of which is shown that the multi-model ARD-PCE approximation is superior when compared to the classical PCE. When using pure ARD-LSTM the network width is larger than when using ARD-LSTM in combination with a multi-element ARD-PCE approximation of the input and a POD approximation of the output. This strongly underlines the computational benefit provided by the proposed RED-LSTM approach, and may form the basis for further larger-scale nonlinear mechanics applications.

By the subsequent analysis of strengths and weaknesses for every considered surrogate approach, starting with deterministic LSTM and finalizing with multi-element RED-LSTM, a robust and computationally efficient auto-adapting training procedure has been developed that provides a possible solution to uncertainty quantification for structural finite element model problems.

6.1 Outlook

The developed multi-element RED-LSTM model provides a promising surrogate solution to structural finite element model problems. However, with the relation between the parameter set and the QoI being implicitly defined, it remains difficult to identify the optimal locations and ranges of the stochastic elements used to approximate the probabilistic space. Although in this work the uniformly distributed input parameter set is subdivided into a multitude of smaller-ranged uniform distributions (i.e. the

stochastic elements), one may further investigate the options to combine (i.e. merge) stochastic elements in case of multiple input parameters and hence reduce the imposed exponential relation between the number of elements and the number of parameters in the parameter set. Additionally, the vastly growing number of imposed elements in the multi-element approach also requires a large number of designs in the DOE to sufficiently fill each of the defined elements. Continued research may be performed on more intrinsic and advanced element boundaries that allow to minimize the number of required designs.

Any change of geometrical parameters in a parameterized finite element model may alter the structures geometry. With geometrical changes of a predetermined structural component one generally requires a reconstruction of the finite element mesh. Such reconstruction causes the mesh to have a varying connectivity table and/or number of elements and nodes depending on a varying geometrical parameter. This then causes variations in the data set size for different designs of experiments. The evaluated NN structures are unable to handle differences in data set dimensions (apart from variations in the batch size n_b). This holds that n_o must be constant throughout the n_d designs. Hence, one is required to unify the number of outputs (elements or nodes) n_o when these vary. Additionally, with n_o being constant, one may still be subjected to a varying connectivity table that defines the (re-)constructed mesh. Although often being visually similar or connected, in the spatial domain the connectivity may be unknown. As a result, any information logic between nodes or elements in different designs may be lost. Although losing this connectivity information between different designs may not terminate any surrogate applications, the loss of this very information may significantly affect the complexity of in-data relationships that one wishes to approximate. A possible solution to varying mesh topologies could be to create a generalized node cloud from the varying mesh topologies, hereby creating a fixed nodal structure over all designs [70]. The resulting generalized node cloud has a fixed dimension over all designs and one may thus directly apply the in this thesis evaluated NN structures. An alternative node driven approach of the surrogate framework employs the spatial dependency of values taken by the QoI. Here, one strives to approximate the QoI of the geometry in a node-by-node sense, where the coordinates of the current node are added to the input parameter set. These coordinates enforce the spatial dependency of the QoI values on the nodes, hereby omitting the necessity of a generalized node cloud.

A Appendix

A.1 Kullback-Leibler divergence

The posterior distribution obtained by the (full) Bayesian rule in a general case is not known and can be obtained by sampling based approaches such as MCMC [142, 143]. However, to speed up computations one can approximate the posterior distribution by some known parametrized distribution. Thus, the goal is to obtain the approximation as accurate as possible. For this purpose then one may use the so-called variational framework. Namely, one may assume that the unknown true posterior $p(\boldsymbol{w}|\boldsymbol{y})$ can be parametrized by some unknown parameters $\boldsymbol{\theta}$, and one may assume that its approximation $\pi(\boldsymbol{w}|\boldsymbol{\theta})$ is determined by minimizing the difference between the approximated variational $\pi(\boldsymbol{w}|\boldsymbol{\theta})$ and the unknown true posterior $p(\boldsymbol{w}|\boldsymbol{y})$ [2, 98, 201, 202]. Hence, one may describe the optimization problem by:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left[D_{\mathrm{KL}} \left[\pi(\boldsymbol{w}|\boldsymbol{\theta}) \parallel p(\boldsymbol{w}|\boldsymbol{y}) \right] \right], \tag{A.1}$$

in which the term D_{KL} denotes the KL-divergence measure between the approximated posterior $\pi(\boldsymbol{w}|\boldsymbol{\theta})$ and the true posterior $p(\boldsymbol{w}|\boldsymbol{y})$, described by:

$$D_{\mathrm{KL}} \left[\pi(\boldsymbol{w}|\boldsymbol{\theta}) \parallel p(\boldsymbol{w}|\boldsymbol{y}) \right] = \int \pi(\boldsymbol{w}|\boldsymbol{\theta}) \log \left(\frac{\pi(\boldsymbol{w}|\boldsymbol{\theta})}{p(\boldsymbol{w}|\boldsymbol{y})} \right) \mathrm{d}\boldsymbol{w}. \tag{A.2}$$

From Eq. (4.7) one may write the true posterior as:

$$p(\boldsymbol{w}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{w}) p(\boldsymbol{w})}{P(\boldsymbol{y})}, \tag{A.3}$$

according to which one may further write out the KL-divergence measure to obtain:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \int \pi(\boldsymbol{w}|\boldsymbol{\theta}) \log \left(\frac{\pi(\boldsymbol{w}|\boldsymbol{\theta}) P(\boldsymbol{y})}{p(\boldsymbol{y}|\boldsymbol{w}) p(\boldsymbol{w})} \right) \mathrm{d}\boldsymbol{w}. \tag{A.4}$$

The evidence $\log P(\boldsymbol{y})$ is constant and can therefore be left out of consideration for the minimization problem. Hence one obtains

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \int \pi(\boldsymbol{w}|\boldsymbol{\theta}) \log \left(\frac{\pi(\boldsymbol{w}|\boldsymbol{\theta})}{p(\boldsymbol{y}|\boldsymbol{w}) p(\boldsymbol{w})} \right) \mathrm{d}\boldsymbol{w}, \tag{A.5}$$

which may also be written as:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \; D_{\mathrm{KL}} \left[\pi(\boldsymbol{w}|\boldsymbol{\theta}) \parallel p(\boldsymbol{w}) \right] - \mathbb{E}_{\pi(\boldsymbol{w}|\boldsymbol{\theta})} \left[p(\boldsymbol{y}|\boldsymbol{w}) \right], \tag{A.6}$$

in which a prior dependent and a data dependent part is described. The prior dependent part may be further derived:

$$\begin{aligned} D_{\mathrm{KL}}\left[\pi(\boldsymbol{w}|\boldsymbol{\theta}) \parallel p(\boldsymbol{w})\right] &= \mathbb{E}_{\pi(\boldsymbol{w}|\boldsymbol{\theta})}\left[\log \frac{\pi(\boldsymbol{w}|\boldsymbol{\theta})}{p(\boldsymbol{w})}\right] \\ &= \mathbb{E}_{\pi(\boldsymbol{w}|\boldsymbol{\theta})}\left[\log \pi(\boldsymbol{w}|\boldsymbol{\theta}) - \log p(\boldsymbol{w})\right]. \end{aligned} \tag{A.7}$$

Using Eq. (A.6) one then obtains

$$\begin{aligned} \boldsymbol{\theta}^* = \underset{\theta}{\operatorname{argmin}} \left[\mathbb{E}_{\pi(\boldsymbol{w}|\boldsymbol{\theta})}\left[\log \pi(\boldsymbol{w}|\boldsymbol{\theta}) - \log p(\boldsymbol{w})\right]\right. \\ \left. - \mathbb{E}_{\pi(\boldsymbol{w}|\boldsymbol{\theta})}\left[\log p(\boldsymbol{y}|\boldsymbol{w})\right]\right], \end{aligned} \tag{A.8}$$

which describes the variational free energy $\mathcal{E}_{\mathrm{e}}$, also known as evidence lower bound (ELBO):

$$\begin{aligned} \mathcal{E}_{\mathrm{e}}(\boldsymbol{y}, \boldsymbol{\theta}) = \mathbb{E}_{\pi(\boldsymbol{w}|\boldsymbol{\theta})}[\log \pi(\boldsymbol{w}|\boldsymbol{\theta}) - \log p(\boldsymbol{w})] \\ - \mathbb{E}_{\pi(\boldsymbol{w}|\boldsymbol{\theta})}\left[\log p(\boldsymbol{y}|\boldsymbol{w})\right]. \end{aligned} \tag{A.9}$$

In the previous equation all terms depend on the expectation w.r.t. $\pi(\boldsymbol{w}|\boldsymbol{\theta})$. Minimizing the free energy embodies finding an optimum between the approximation of data set $\boldsymbol{y}$, while satisfying prior $p(\boldsymbol{w})$. The variational free energy can be found by approximating the continuous variable problem as a discrete variable problem. The sampling-based procedure [2] for such approximation can be written in the form:

$$\mathcal{E}_{\mathrm{e}}(\boldsymbol{y}, \boldsymbol{\theta}) \approx \sum_{\kappa=1}^{n_{\mathrm{s}}} \log(\pi(\boldsymbol{w}^{(\kappa)}|\boldsymbol{\theta})) - \log(p(\boldsymbol{w}^{(\kappa)})) - \log p(\boldsymbol{y}|\boldsymbol{w}^{(\kappa)}) \tag{A.10}$$

with $\boldsymbol{w}^{(\kappa)}$ being the κth Monte-Carlo sample taken from $\pi(\boldsymbol{w}^{(\kappa)}|\boldsymbol{\theta})$.

The objective function for the approximation of $\boldsymbol{\theta}^*$ using the minimization of the variational free energy is then given by:

$$\boldsymbol{\theta}^* \approx \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ \mathcal{E}_{\mathrm{e}}(\boldsymbol{y}, \boldsymbol{\theta}) \approx \sum_{\kappa=1}^{n_{\mathrm{s}}} \log(\pi(\boldsymbol{w}^{(\kappa)}|\boldsymbol{\theta})) - \log(p(\boldsymbol{w}^{(\kappa)})) - \log p(\boldsymbol{y}|\boldsymbol{w}^{(\kappa)}). \tag{A.11}$$

Hence, an approximation of $\boldsymbol{\theta}^*$ leads to $\pi(\boldsymbol{w}|\boldsymbol{\theta}^*)$, which approximates the posterior $p(\boldsymbol{w}|\boldsymbol{y})$.

A.2 Bayesian conditional Gaussian description

Given the prior distribution on the weights $p(\boldsymbol{w})$, one may estimate their value given the measurement $\boldsymbol{y}$ using Bayes rule:

$$p(\boldsymbol{w}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{w})p(\boldsymbol{w})}{P(\boldsymbol{y})} \tag{A.12}$$

in which the measurement is linearly dependent on the weight, i.e.

$$\hat{\boldsymbol{y}} = \boldsymbol{\Phi}\boldsymbol{w}. \tag{A.13}$$

Thus, the output data set $\boldsymbol{y} \in \mathbb{R}^{n_\mathrm{b}}$ defined over the previous equation, $\boldsymbol{\Phi} \in \mathbb{R}^{n_\mathrm{b} \times n_\mathrm{f}}$ and $\boldsymbol{w} \in \mathbb{R}^{n_\mathrm{f}+1}$. For a zero-mean Gaussian prior on weights [87]:

$$p(\boldsymbol{w}) = \prod_{j=1}^{n_\mathrm{f}+1} \mathcal{N}(w_j|0, \alpha_j^{-1}); \tag{A.14}$$

one obtains the likelihood function

$$p(\boldsymbol{y}|\boldsymbol{w}) = \prod_{i=1}^{n_\mathrm{b}} \prod_{j=1}^{n_\mathrm{f}+1} \mathcal{N}(y_i|\phi_{ij}w_j, \beta^{-1}), \tag{A.15}$$

in which one may define $\boldsymbol{\Phi} := [..., \phi_{ij}, ...], i = 1, ..., n_\mathrm{b}, j = 1, ..., n_\mathrm{f} + 1$. Furthermore, $\boldsymbol{\alpha}$ and β are precision parameters that are used to describe the prior and the measurement variance. The product of the prior and likelihood function in Eq. (A.12) gives the joint distribution over $\boldsymbol{w}$ and $\boldsymbol{y}$. By introducing

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{y} \end{bmatrix} \tag{A.16}$$

one can write:

$$\begin{aligned} \log p(\boldsymbol{z}) &= \log p(\boldsymbol{w}) + \log p(\boldsymbol{y}|\boldsymbol{w}) \\ &= \frac{1}{2}\boldsymbol{w}^T\boldsymbol{\alpha}\boldsymbol{w} - \frac{1}{2}(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w})^T\beta(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w}) \end{aligned} \tag{A.17}$$

As a Gaussian distribution is conjugate to itself, $p(\boldsymbol{z})$ is Gaussian as well. Its mean and covariance can thus be determined from the previous equation by letting the left hand side be Gaussian as well, and then making the corresponding terms equal:

$$\begin{aligned} &-\frac{1}{2}(\boldsymbol{\alpha} \odot \mathbf{I} + \beta\boldsymbol{\Phi}^T\boldsymbol{\Phi})\boldsymbol{w} - \frac{1}{2}\beta\boldsymbol{y}^T\boldsymbol{y} + \frac{1}{2}\boldsymbol{y}^T\beta\boldsymbol{\Phi}\boldsymbol{w}\frac{1}{2}\boldsymbol{w}^T\boldsymbol{\Phi}^T\beta\boldsymbol{y} \\ &= \frac{1}{2}\begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{y} \end{bmatrix}^T \begin{bmatrix} \boldsymbol{\alpha} \odot \mathbf{I} + \beta\boldsymbol{\Phi}^T\boldsymbol{\Phi} & -\beta\boldsymbol{\Phi}^T \\ -\beta\boldsymbol{\Phi} & \beta \end{bmatrix} \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{y} \end{bmatrix} \\ &= -\frac{1}{2}\boldsymbol{z}^T\mathbf{R}\boldsymbol{z}, \end{aligned} \tag{A.18}$$

with $\mathbf{I}$ being the diagonal unit matrix and $\mathbf{R}$ representing the precision matrix. Accordingly, as the covariance is the inverse of the precision, the covariance $\mathbf{\Sigma}_{\mathrm{z}}$ of joint Gaussian distribution $p(\boldsymbol{z})$ is given by:

$$\mathbf{\Sigma}_{\mathrm{z}} = \mathbf{R}^{-1} = \begin{bmatrix} \boldsymbol{\alpha}^{-1} \odot \mathbf{I} & (\boldsymbol{\alpha} \odot \mathbf{I})^{-1}\mathbf{\Phi}^T \\ \mathbf{\Phi}(\boldsymbol{\alpha} \odot \mathbf{I})^{-1} & \beta^{-1} + \mathbf{\Phi}(\boldsymbol{\alpha} \odot \mathbf{I})^{-1}\mathbf{\Phi}^T \end{bmatrix}. \tag{A.19}$$

In a similar fashion, the mean $\boldsymbol{\mu}_{\mathrm{z}}$ is given by evaluating the first order terms in Eq. (A.17). However, as a zero mean has been imposed in Eq. (A.14) and by defining the system merely by $\mathbf{\Phi}\boldsymbol{w}$ (disregarding a bias), no first-order terms arise and hence the mean $\boldsymbol{\mu}_{\mathrm{z}} = 0$. A derivation using a nonzero-mean and bias may be found in [87].

The joint distribution $p(\boldsymbol{z}) \sim \mathcal{N}(\boldsymbol{\mu}_{\mathrm{z}}, \mathbf{\Sigma}_{\mathrm{z}})$ can be used to describe the mean and covariance of marginal distribution $p(\boldsymbol{y})$, after marginalizing out $\boldsymbol{w}$. From $\boldsymbol{\mu}_{\mathrm{z}} = 0$, it follows that $\boldsymbol{\mu}_{\mathrm{y}} = 0$. The covariance in Eq. (A.19) can be written by a set of partitions:

$$\begin{bmatrix} \boldsymbol{\alpha}^{-1} \odot \mathbf{I} & (\boldsymbol{\alpha} \odot \mathbf{I})^{-1}\mathbf{\Phi}^T \\ \mathbf{\Phi}(\boldsymbol{\alpha} \odot \mathbf{I})^{-1} & \beta^{-1} + \mathbf{\Phi}(\boldsymbol{\alpha} \odot \mathbf{I})^{-1}\mathbf{\Phi}^T \end{bmatrix} = \begin{bmatrix} \Lambda_{\mathrm{ww}} & \Lambda_{\mathrm{wy}} \\ \Lambda_{\mathrm{yw}} & \Lambda_{\mathrm{yy}} \end{bmatrix}, \tag{A.20}$$

and thus, with $\Lambda_{\mathrm{yy}} = \mathbf{\Sigma}_{\mathrm{y}}$, one obtains the mean and covariance for $p(\boldsymbol{y})$:

$$\begin{aligned} \boldsymbol{\mu}_{\mathrm{y}} &= 0 \\ \mathbf{\Sigma}_{\mathrm{y}} &= \beta^{-1} + \mathbf{\Phi}(\boldsymbol{\alpha} \odot \mathbf{I})^{-1}\mathbf{\Phi}^T \end{aligned} \tag{A.21}$$

With $p(\boldsymbol{z})$ and $p(\boldsymbol{y})$ known, one may seek the expression for the conditional distribution $p(\boldsymbol{w}|\boldsymbol{y})$. Knowing that in such conditional formulation the covariance can be found from the precision matrix (Eq. (A.18)), one obtains $\mathbf{\Sigma}_{\mathrm{w|y}} = \mathbf{R}_{\mathrm{ww}}^{-1}$, and thus [87]:

$$\mathbf{\Sigma}_{\mathrm{w|y}} = (\boldsymbol{\alpha} \odot \mathbf{I} + \beta\mathbf{\Phi}^T\mathbf{\Phi})^{-1}. \tag{A.22}$$

Additionally, the mean $\boldsymbol{\mu}_{\mathrm{w|y}}$ can be found by first evaluating the general description of the exponential in a general Gaussian formulation $\mathcal{N}(\boldsymbol{y}|\boldsymbol{\mu}, \mathbf{\Sigma})$, a full derivation of which is given in [87]:

$$\begin{aligned} \boldsymbol{\mu}_{\mathrm{w|y}} &= -\mathbf{\Sigma}_{\mathrm{w|y}}\mathbf{R}_{\mathrm{wy}}\boldsymbol{y} \\ &= \beta\mathbf{\Sigma}_{\mathrm{w|y}}\mathbf{\Phi}^T\boldsymbol{y} \end{aligned} \tag{A.23}$$

and hence:

$$p(\boldsymbol{w}|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{w}|\beta\mathbf{\Sigma}_{\mathrm{w|y}}\mathbf{\Phi}^T\boldsymbol{y}, \mathbf{\Sigma}_{\mathrm{w|y}}) \tag{A.24}$$

for $\mathbf{\Sigma}_{\mathrm{w|y}} = (\boldsymbol{\alpha} \odot \mathbf{I} + \beta\mathbf{\Phi}^T\mathbf{\Phi})^{-1}$.

A.3 Marginal likelihood gradient

One may reformulate the linear system described in Appendix A.2 to describe the linearized LSTM gate g:

$$[\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k} = \mathbf{\Phi}_\ell [\mathbf{w}_{\mathrm{g},\ell}]_{:,k}, \quad k = 1, ..., n_{\mathrm{m}}, \tag{A.25}$$

with $\bar{\mathbf{s}}_{\mathrm{g},\ell} \in \mathbb{R}^{n_{\mathrm{b}} \times n_{\mathrm{m}}}$ being the local target mean at time step ℓ, $\mathbf{\Phi}_\ell := [1\ \mathbf{q}\ \mathbb{E}(\mathbf{h}_\ell)]$ being the gate input with $\mathbf{\Phi}_\ell \in \mathbb{R}^{n_{\mathrm{b}} \times n_{\mathrm{i}}}$ for $n_{\mathrm{i}} = 1 + n_{\mathrm{f}} + n_{\mathrm{m}}$ and $\mathbf{w}_{\mathrm{g},\ell} \in \mathbb{R}^{n_{\mathrm{i}} \times n_{\mathrm{m}}}$ being the weights. Using Appendix A.2 one can formulate the marginal likelihood $p(\bar{\mathbf{s}}_{\mathrm{g},\ell} | \boldsymbol{\alpha}_{\mathrm{g},\ell}, \boldsymbol{\beta}_{\mathrm{g},\ell})$ objective function by:

$$[\mathcal{L}_{\mathrm{g},\ell}]_k = -\frac{1}{2} \left[\log |[\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}| + [\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k}^T [\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}^{-1} [\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k} \right], \tag{A.26}$$

in which (following Appendix A.2) one may denote $[\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k} := [\boldsymbol{\beta}_{\mathrm{g},\ell}]_k^{-1} \mathbf{I} + \mathbf{\Phi}_\ell ([\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k} \odot \mathbf{I})^{-1} \mathbf{\Phi}_\ell^T$. One may estimate the likelihood gradient which can be obtained by:

$$\frac{\partial \mathcal{L}_{\mathrm{g},\ell}}{\partial \mathbf{\Phi}_\ell} = \sum_{k=1}^{n_{\mathrm{m}}} \frac{\partial [\mathcal{L}_{\mathrm{g},\ell}]_k}{\partial [\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}} \frac{\partial [\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}}{\partial \mathbf{\Phi}_\ell}. \tag{A.27}$$

From Eq. (A.26) and using

$$\begin{aligned} \frac{\partial \log |[\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}|}{\partial [\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}} &= \frac{\partial \log |[\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}|}{\partial \mathrm{tr}(\log ([\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}))} \\ &= [\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}^{-1} \end{aligned} \tag{A.28}$$

the first term of Eq. (A.26) can be differentiated:

$$\begin{aligned} \frac{\partial \log |[\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}|}{\partial \mathbf{\Phi}_\ell} &= [\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}^{-1} \frac{\partial [\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}}{\partial \mathbf{\Phi}_\ell} \\ &= 2 [\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}^{-1} \mathbf{\Phi}_\ell ([\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k} \odot \mathbf{I})^{-1}, \end{aligned} \tag{A.29}$$

and in a similar sense one can derive the second term of Eq. (A.26):

$$\begin{aligned} \frac{\partial ([\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k}^T [\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}^{-1} [\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k})}{\partial \mathbf{\Phi}_\ell} &= -([\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}^{-1} [\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k} [\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k}^T [\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}^{-1}) \frac{\partial [\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}}{\partial \mathbf{\Phi}_\ell} \\ &= -2([\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}^{-1} [\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k} [\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k}^T [\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}^{-1}) \\ &\quad \cdot \mathbf{\Phi}_\ell ([\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k} \odot \mathbf{I})^{-1}. \end{aligned} \tag{A.30}$$

Using Eq. (A.26) and Eq. (A.27) and combining Eq. (A.29) and Eq. (A.30) one then obtains the desired gradient:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{\Phi}_\ell} &= \sum_{k=1}^{n_{\mathrm{m}}} -\frac{1}{2} \Big[2 [\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}^{-1} \mathbf{\Phi}_\ell ([\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k} \odot \mathbf{I})^{-1} \\ &\qquad -2([\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}^{-1} [\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k} [\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k}^T [\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}^{-1}) \mathbf{\Phi}_\ell ([\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k} \odot \mathbf{I})^{-1} \Big] \\ &= \sum_{k=1}^{n_{\mathrm{m}}} \left[[\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}^{-1} [\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k} [\bar{\mathbf{s}}_{\mathrm{g},\ell}]_{:,k}^T [\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}^{-1} - [\mathbf{C}_{\mathrm{g},\ell}]_{:,:,k}^{-1} \right] \mathbf{\Phi}_\ell ([\boldsymbol{\alpha}_{\mathrm{g},\ell}]_{:,k} \odot \mathbf{I})^{-1} \end{aligned} \tag{A.31}$$

A.4 Orthogonal polynomials

A set of polynomials may be called orthogonal when

$$\begin{aligned}\int \psi_n(\xi)\psi_m(\xi)\mathrm{d}p(\xi) &= \int \psi_n(\xi)\psi_m(\xi)f(\xi)\mathrm{d}\xi, \quad m,n=1,...,n_\mathrm{g}\in\mathbb{N} \\ &= \gamma_n\delta_{mn},\end{aligned} \tag{A.32}$$

with δ_{mn} the Kronecker delta function ($\delta_{mn}=1$ if $n=m$ else 0), p being the probability measure and f is the probability density function for ξ and n_g the polynomial degree considered. For $\gamma_n=1$ the system is orthonormal. When assuming random variable ξ to be uniformly distributed, i.e. $p(\xi)\sim\mathcal{U}(-1,1)$, the PDF $f(\xi)=\frac{1}{2}$ then gives us the orthogonality requirement

$$\int_{-1}^{1}\psi_n(\xi)\psi_m(\xi)\frac{1}{2}\mathrm{d}\xi=\gamma_n\delta_{mn}, \quad m,n=1,...,n_\mathrm{g}\in\mathbb{N} \tag{A.33}$$

which is satisfied by Legendre polynomials with $\psi_0(\xi)=1$, $\psi_1(\xi)=\xi$ and $\psi_2(\xi)=\frac{1}{2}(3\xi^2-1)$ being the first three terms. A graphical representation of the first six Legendre polynomial degrees is given in Fig. A.1.

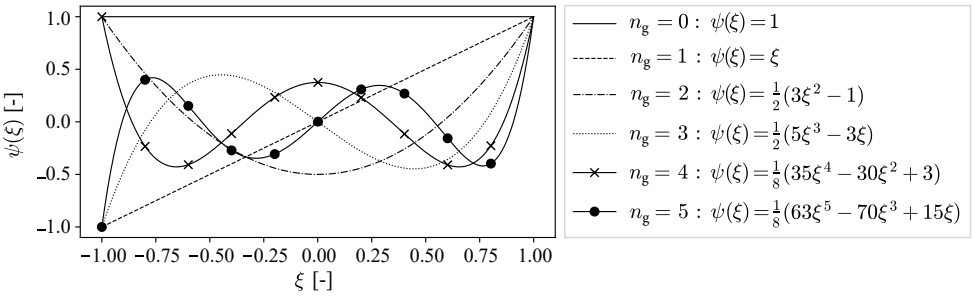


Figure A.1: Monovariate Legendre polynomials of degree n_g in their respected range $[-1,1]$.

Similarly, one finds the Hermite polynomials to be orthogonal for a Gaussian distributed basis in $[-\infty,\infty]$.

Bibliography

[1] T.K. Eller. *Modeling of tailor hardened boron steel for crash simulation.* PhD thesis, University of Twente, 2016.

[2] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. *arXiv preprint*, 2015, doi: 10.48550/ARXIV.1505.05424.

[3] T.K. Eller, L. Greve, M. Andres, M. Medricky, V.T. Meinders, and A.H. van den Boogaard. Determination of strain hardening parameters of tailor hardened boron steel up to high strains using inverse FEM optimization and strain field matching. *Journal of Materials Processing Technology*, 228:43–58, 2016, doi: 10.1016/j.jmatprotec.2015.09.036.

[4] C. Parenteau, S. White, R. Burnett, G. Stephens, and D. Michalski. Rear-end impacts - part 1: Field and test data analysis of crash characteristics. *SAE Int. J. Adv. & Curr. Prac. in Mobility*, 4(6):2106–2119, 2022, doi: 10.4271/2022-01-0859.

[5] R. Arbelaez, B. Mueller, M. Brumbelow, and E. Teoh. Next steps for the IIHS side crashworthiness evaluation program. In *62nd Stapp Car Crash Conference*, 5–7, 2019, doi: 10.4271/SC18-22-0002.

[6] Euro NCAP. General protocolls. https://www.euroncap.com/en/for-engineers/protocols/general/, 2022. Accessed: 2022.

[7] Insurance Institute for Highway Safety (IIHS). Test protocols and technical information. https://www.iihs.org/ratings/about-our-tests/test-protocols-and-technical-information, 2022. Accessed: 2022.

[8] R. Blumhardt. FEM - crash simulation and optimization. *International Journal of Vehicle Design*, 26:331–347, 2001.

[9] J.M. Chang, T. Tyan, M. El-bkaily, J. Cheng, A. Marpu, Q. Zeng, and J. Santini. Implicit and explicit finite element methods for crash safety analysis. *SAE Transactions*, 116:1025–1037, 2007.

[10] K.J. Hickey and S.T. Xiao. Finite element modeling and simulation of car crash. *International Journal of Modern Studies in Mechanical Engineering (IJMSME)*, 3:1–5, 2017, doi: 10.20431/2454-9711.0301001.

[11] O.C. Zienkiewicz, R.L. Taylor, and J.Z. Zhu. *The finite element method: its basis and fundamentals.* Elsevier, 7 edition, 2005.

[12] J.N. Reddy. *An introduction to the finite element method.* McGraw-Hill, New York, 3 edition, 2004.

[13] T.K. Eller, L. Greve, M. Andres, M. Medricky, H.J.M. Geijselaers, V.T. Meinders, and A.H. van den Boogaard. Plasticity and fracture modeling of the heat-affected zone in resistance spot welded tailor hardened boron steel. *Journal of Materials Processing Technology*, 234:309–322, 2016, doi: 10.1016/j.jmatprotec.2016.03.026.

[14] L. Greve, T.K. Eller, M. Medricky, and M.T. Andres. Hardness-based plasticity and fracture model for quench-hardenable boron steel (22mnb5). volume 1567, 571–574. American Institute of Physics, jan 2013, doi: 10.1063/1.4850038.

[15] M. Rocas, A. García-González, S. Zlotnik, X. Larráyoz, and P. Díez. Non-intrusive uncertainty quantification for automotive crash problems with vps/-pamcrash. *Finite Elements in Analysis and Design*, 193:103556, 2021, doi: 10.1016/j.finel.2021.103556.

[16] H. Contreras. The stochastic finite-element method. *Computers & Structures*, 12(3):341–348, 1980.

[17] J.D. Arregui-Mena, L. Margetts, and P.M. Mummery. Practical application of the stochastic finite element method. *Archives of Computational Methods in Engineering*, 23(1):171–190, 2016.

[18] G. Stefanou. The stochastic finite element method: past, present and future. *Computer methods in applied mechanics and engineering*, 198(9-12):1031–1051, 2009.

[19] M. Loeve. *Probability Theory I.* Graduate Texts in Mathematics. Springer, New York, NY, 1977.

[20] S. Volkwein K. Kunisch. Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics. *SIAM Journal on Numerical Analysis*, 40(2):492–515, 2003, doi: 10.1137/S0036142900382612.

[21] H.M. Park and D.H. Cho. The use of the karhunen-loève decomposition for the modeling of distributed parameter systems. *Chemical Engineering Science*, 51(1):81 – 98, 1996, doi: 10.1016/0009-2509(95)00230-8.

[22] L. Sirovich. Turbulence and the dynamics of coherent structures part i: coherent structures. *Quarterly of Applied Mathematics*, 45(3):561–571, 1987.

[23] I.T. Jolliffe and J. Cadima. Principal component analysis: a review and recent developments. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 374(2065), 2016, doi: 10.1098/rsta.2015.0202.

[24] W.S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. 1943. *Bulletin of mathematical biology*, 52:99–115, 1990, doi: 10.1007/BF02459570.

[25] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989, doi: 10.1007/BF02551274.

[26] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by back-propagating errors. In James A. Anderson and Edward Rosenfeld, editors, *Neurocomputing: Foundations of Research*, 696–699. MIT Press, Cambridge, MA, USA, 1988, doi: 10.1038/323533a0.

[27] A. Botchkarev. A new typology design of performance metrics to measure errors in machine learning regression algorithms. *Interdisciplinary Journal of Information, Knowledge, and Management*, 14:045–076, 2019, doi: 10.28945/4184.

[28] A. Derogar and F. Djavanroodi. Artificial neural network modeling of forming limit diagram. *Materials and Manufacturing Processes*, 26:1415–1422, 2011, doi: 10.1080/10426914.2010.544818.

[29] P.S. Theocaris and P.D. Panagiotopoulos. Neural networks for computing in fracture mechanics. methods and prospects of applications. *Computer Methods in Applied Mechanics and Engineering*, 106(1):213 – 228, 1993, doi: 10.1016/0045-7825(93)90191-Y.

[30] K. Elangovan, C. Sathiya Narayanan, and R. Narayanasamy. Modelling of forming limit diagram of perforated commercial pure aluminium sheets using artificial neural network. *Computational Materials Science*, 47(4):1072 – 1078, 2010, doi: 10.1016/j.commatsci.2009.12.016.

[31] N. Kotkunde, A.D. Deole, and A.K. Gupta. Prediction of forming limit diagram for ti-6al-4v alloy using artificial neural network. *Procedia Materials Science*, 6:341 – 346, 2014, doi: 10.1016/j.mspro.2014.07.043. 3rd International Conference on Materials Processing and Characterisation (ICMPC 2014).

[32] M. Mohamed, S. Elatriby, Z. Shi, and J. Lin. Prediction of forming limit diagram for aa5754 using artificial neural network modelling. *Key Engineering Materials*, 716:770–778, 2016, doi: 10.4028/www.scientific.net/KEM.716.770.

[33] S. Kannadasan, A. Senthil Kumar, P. Chinnaiyan, and C. Sathiya Narayanan. Modelling the forming limit diagram for aluminium alloy sheets using ann and anfis. *Applied Mathematics and Information Sciences*, 11:1435–1442, 2017, doi: 10.18576/amis/110521.

[34] D. Svozil, V. Kvasnicka, and J. Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, 39(1):43–62, 1997, doi: 10.1016/S0169-7439(97)00061-0.

[35] L. Greve, B. Schneider, T.K. Eller, M. Andres, J.D. Martinez, and B.P. van de Weg. Necking-induced fracture prediction using an artificial neural network trained on virtual test data. *Engineering Fracture Mechanics*, 219, 2019, doi: 10.1016/
j.engfracmech.2019.106642.

[36] S. Nasiri, M.R. Khosravani, and K. Weinberg. Fracture mechanics and mechanical fault detection by artificial intelligence methods: A review. *Engineering Failure Analysis*, 81:270 – 293, 2017, doi: 10.1016/j.engfailanal.2017.07.011.

[37] S. Stören and J.R. Rice. Localized necking in thin sheets. *Journal of the Mechanics and Physics of Solids*, 23(6):421 – 441, 1975, doi: 10.1016/0022-5096(75)90004-6.

[38] R. Hill. On discontinuous plastic states, with special reference to localized necking in thin sheets. *Journal of the Mechanics and Physics of Solids*, 1(1):19–30, 1952, doi: 10.1016/0022-5096(52)90003-3.

[39] Z. Marciniak and K. Kuczynski. Limit strains in the processes of stretch-forming sheet metal. *International Journal of Mechanical Sciences*, 9(9):609 – 620, 1967, doi: 10.1016/0020-7403(67)90066-5.

[40] J. Hoole, P. Sartor, J.D. Booker, J.E. Cooper, X. Gogouvitis, and R.K. Schmidt. *Comparison of Surrogate Modeling Methods for Finite Element Analysis of Landing Gear Loads*. American Institute of Aeronautics and Astronautics, 2020, doi: 10.2514/6.2020-0681.

[41] R.C. Heap, A.I. Hepworth, and C. Greg Jensen. Real-time visualization of finite element models using surrogate modeling methods. *Journal of Computing and Information Science in Engineering*, 15(1), 2015, doi: 10.1115/1.4029217.

[42] J.G. Hoffer, B.C. Geiger, P. Ofner, and R. Kern. Mesh-free surrogate models for structural mechanic fem simulation: A comparative study of approaches. *Applied Sciences*, 11(20), 2021, doi: 10.3390/app11209411.

[43] H. Drucker, C.J.C. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. In *Proceedings of the 9th International Conference on Neural Information Processing Systems*, 155–161, Cambridge, MA, USA, 1996. MIT Press.

[44] J.C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers*, 61–74. MIT Press, 1999.

[45] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint*, 2017, doi: 10.48550/ARXIV.1711.10561.

[46] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A Training Algorithm for Optimal Margin Classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, 144–152, New York, NY, USA, 1992. Association for Computing Machinery, doi: 10.1145/130385.130401.

[47] V. Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999.

[48] J. Bohn and M. Feischl. Recurrent neural networks as optimal mesh refinement strategies. Technical Report 33, Karlsruher Institut für Technologie (KIT), 2020, doi: 10.5445/IR/1000125782.

[49] S.K. Mitusch, S.W. Funke, and M. Kuchta. Hybrid FEM-NN models: Combining artificial neural networks with the finite element method. *Journal of Computational Physics*, 446:110651, 2021, doi: 10.1016/j.jcp.2021.110651.

[50] B. Schölkopf, A. Smola, and K.R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998, doi: 10.1162/089976698300017467.

[51] A. García-González, A. Huerta, S. Zlotnik, and P. Díez. A kernel principal component analysis (kpca) digest with a new backward mapping (pre-image reconstruction) strategy. *arXiv preprint*, 2020, doi: 10.48550/ARXIV.2001.01958.

[52] M. Linting, J.J. Meulman, P.J.F. Groenen, and A.J. van der Koojj. Nonlinear principal components analysis: introduction and application. *Psychological methods*, 12(3):336–358, 2007, doi: 10.1037/1082-989X.12.3.336.

[53] A. Ammar, B. Mokdad, F. Chinesta, and R. Keunings. A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids. *Journal of Non-Newtonian Fluid Mechanics*, 139(3):153 – 176, 2006, doi: 10.1016/j.jnnfm.2006.07.007.

[54] A. Ammar, B. Mokdad, F. Chinesta, and R. Keunings. A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modelling of complex fluids: Part ii: Transient simulation using space-time separated representations. *Journal of Non-Newtonian Fluid Mechanics*, 144(2):98 – 121, 2007, doi: 10.1016/j.jnnfm.2007.03.009.

[55] J.O. Ramsay and B.W. Silverman. *Functional data analysis*. Springer series in statistics. Springer, 2 edition, 2005.

[56] H.L. Shang. A survey of functional principal component analysis. *AStA Advances in Statistical Analysis*, 98(2):121–142, 2014, doi: 10.1007/s10182-013-0213-1.

[57] P. Schmid and J. Sesterhenn. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656, 2008, doi: 10.1017/S0022112010001217.

[58] M.A. Caicedo, J.L. Mroginski, S.A. Toro, M. Raschi, A.E. Huespe, and J. Oliver. High performance reduced order modeling techniques based on optimal energy quadrature: Application to geometrically non-linear multiscale inelastic material modeling. *Archives of Computational Methods in Engineering*, 26(1):771–792, 2019, doi: 10.1007/s11831-018-9258-3.

[59] L. Xia and P. Breitkopf. A reduced multiscale model for nonlinear structural topology optimization. *Computer Methods in Applied Mechanics and Engineering*, 280:117–134, 2014, doi: 10.1016/j.cma.2014.07.024.

[60] I.B.C.M. Rocha, F.P. van der Meer, S. Raijmaekers, F. Lahuerta, R.P.L. Nijssen, L.P. Mikkelsen, and L.J. Sluys. A combined experimental/numerical investigation on hygrothermal aging of fiber-reinforced composites. *European Journal of Mechanics - A/Solids*, 73:407–419, 2019, doi: 10.1016/j.euromechsol.2018.10.003.

[61] F. Fritzen and M. Hodapp. The finite element square reduced (fe2r) method with gpu acceleration: towards three-dimensional two-scale simulations. *International Journal for Numerical Methods in Engineering*, 107(10):853–881, 2016, doi: 10.1002/nme.5188.

[62] V. Limousin, X. Delgerie, E. Leroy, R. Ibáñez, C. Argerich, F. Daim, J.L. Duval, and F. Chinesta. Advanced model order reduction and artificial intelligence techniques empowering advanced structural mechanics simulations: application to crash test analyses. *Mechanics & Industry*, 2019.

[63] J.L. Elman. Finding Structure in Time. *Cognitive Science*, 14(2):179–211, 1990, doi: 10.1207/s15516709cog1402_1.

[64] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint*, 2014, doi: 10.48550/ARXIV.1406.1078.

[65] S. Hochreiter and J. Schmidhuber. Long Short-term Memory. *Neural computation*, 9:1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735.

[66] F. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12:2451–2471, 2000, doi: 10.1162/089976600300015015.

[67] A. Sherstinsky. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D: Nonlinear Phenomena*, 404, 2020, doi: 10.1016/j.physd.2019.132306.

[68] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint*, 2014, doi: 10.48550/ARXIV.1412.3555.

[69] D. Keysers, T. Deselaers, H. A. Rowley, L. Wang, and V. Carbune. Multi-language online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1180–1194, 2017, doi: 10.1109/TPAMI.2016.2572693.

[70] L. Greve and B.P. van de Weg. Surrogate modeling of parametrized finite element simulations with varying mesh topology using recurrent neural networks. *Array*, 100–137, 2022, doi: 10.1016/j.array.2022.100137.

[71] F. Ghavamian and A. Simone. Accelerating multiscale finite element simulations of history-dependent materials using a recurrent neural network.

Computer Methods in Applied Mechanics and Engineering, 357, 2019, doi: 10.1016/j.cma.2019.112594.

[72] K. Wang and W.C. Sun. A multiscale multi-permeability poroplasticity model linked by recursive homogenizations and deep learning. *Computer Methods in Applied Mechanics and Engineering*, 334:337 – 380, 2018, doi: 10.1016/j.cma.2018.01.036.

[73] H.P. Wan and W.X. Ren. Parameter selection in finite-element-model updating by global sensitivity analysis using gaussian process metamodel. *Journal of Structural Engineering*, 141(6), 2015, doi: 10.1061/(ASCE)ST.1943-541X.0001108.

[74] R. Ghanem. Stochastic finite elements with multiple random non-gaussian properties. *Journal of Engineering Mechanics*, 125(1):26–40, 1999, doi: 10.1061/(ASCE)0733-9399(1999)125:1(26).

[75] A.A. Basmaji, A. Fau, J.H. Urrea-Quintero, M.M. Dannert, E. Voelsen, and U. Nackenhorst. Anisotropic multi-element polynomial chaos expansion for high-dimensional non-linear structural problems. *Probabilistic Engineering Mechanics*, 70:103366, 2022, doi: 10.1016/j.probengmech.2022.103366.

[76] K. Sepahvand, S. Marburg, and H.J. Hardtke. Stochastic structural modal analysis involving uncertain parameters using generalized polynomial chaos expansion. *International Journal of Applied Mechanics*, 03(03):587–606, 2011, doi: 10.1142/S1758825111001147.

[77] S. Marelli and B. Sudret. An active-learning algorithm that combines sparse polynomial chaos expansions and bootstrap for structural reliability analysis. *Structural Safety*, 75:67–74, 2018, doi: 10.1016/j.strusafe.2018.06.003.

[78] C.M. Tiago and V.M.A. Leitão. Application of radial basis functions to linear and nonlinear structural analysis problems. *Computers & Mathematics with Applications*, 51(8):1311–1334, 2006, doi: 10.1016/j.camwa.2006.04.008. Radial Basis Functions and Related Multivariate Meshfree Approximation Methods: Theory and Applications.

[79] O.P. Le Maître and O.M. Knio, editors. *Spectral Methods for Uncertainty Quantification*. Scientific Computation. Springer Netherlands, Dordrecht, 2010.

[80] X. Dongbin. *Numerical Methods for Stochastic Computations: A Spectral Method Approach*. Princeton university press, 2010, doi: 10.2307/j.ctv7h0skv.

[81] N. Wiener. The Homogeneous Chaos. *American Journal of Mathematics*, 60(4):897–936, 1938.

[82] B. Rosić. Stochastic state estimation via incremental iterative sparse polynomial chaos based bayesian-gauss-newton-markov-kalman filter. *arXiv preprint*, 2019, doi: 10.48550/ARXIV.1909.07209.

[83] B. Rosić, A. Litvinenko, O. Pajonk, and H. Matthies. Direct Bayesian Update of Polynomial Chaos Representations. *TU Braunschweig*, 2011, doi: 10.24355/dbbs.084-201104011413-0.

[84] D.S. Broomhead and D. Lowe. Multivariable Functional Interpolation and Adaptive Networks. *Complex Systems*, 2:321–355, 1988.

[85] P. Hennig, M.A. Osborne, and M. Girolami. Probabilistic numerics and uncertainty in computations. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471(2179), 2015, doi: 10.1098/rspa.2015.0142.

[86] J. Joyce. Bayes' Theorem. In *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2021 edition, 2021.

[87] C.M. Bishop. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, 2006.

[88] H. He, B. Xin, S. Ikehata, and D. Wipf. From Bayesian sparsity to gated recurrent nets. 5554–5564, 2017.

[89] M. Fortunato, C. Blundell, and O. Vinyals. Bayesian recurrent neural networks. *arXiv preprint*, 2017, doi: 10.48550/ARXIV.1704.02798.

[90] H. Zhang, W. Zhang, L. Yu, and G. Bi. Distributed Compressive Sensing via LSTM-Aided Sparse Bayesian Learning. *Signal Processing*, 176, 2020, doi: 10.1016/j.sigpro.2020.107656.

[91] D.J.C. MacKay. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992, doi: 10.1162/neco.1992.4.3.415.

[92] A. Doerr, C. Daniel, M. Schiegg, D. Nguyen-Tuong, S. Schaal, M. Toussaint, and S. Trimpe. Probabilistic recurrent state-space models. *arXiv preprint*, 2018, doi: 10.48550/ARXIV.1801.10395.

[93] N. Nikolaev and P. Tino. Sequential relevance vector machine learning from time series. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks*, 1308–1313, 2005, doi: 10.1109/IJCNN.2005.1556043.

[94] C. Chen, X. Lin, and G. Terejanu. An approximate bayesian long short-term memory algorithm for outlier detection. In *2018 24th International Conference on Pattern Recognition (ICPR)*, 201–206, 2018, doi: 10.1109/ICPR.2018.8545695.

[95] D.T. Mirikitani and N. Nikolaev. Recursive bayesian recurrent neural networks for time-series modeling. *IEEE Transactions on Neural Networks*, 21(2):262–274, 2009, doi: 10.1109/TNN.2009.2036174.

[96] S.P. Chatzis. Sparse bayesian recurrent neural networks. In *Joint european conference on machine learning and knowledge discovery in databases*, 359–372, 2015, doi: 10.1007/978-3-319-23525-7_22.

[97] S. Gulshad, D. Sigmund, and J.H. Kim. Learning to reproduce stochastic time series using stochastic LSTM. In *2017 International Joint Conference on Neural Networks (IJCNN)*, 859–866, 2017, doi: 10.1109/IJCNN.2017.7965942.

[98] S. Kullback and R.A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 1951, doi: 10.1214/aoms/1177729694.

[99] M.E. Tipping. Sparse Bayesian Learning and the Relevance Vector Machine. *J. Mach. Learn. Res.*, 1:211–244, 2001, doi: 10.1162/15324430152748236.

[100] J. Quinonero-Candela and L.K. Hansen. Time series prediction based on the relevance vector machine with adaptive kernels. In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 985–988, 2002, doi: 10.1109/ICASSP.2002.5743959.

[101] F. Liu, H. Song, Q. Qi, and J. Zhou. Time series regression based on relevance vector learning mechanism. In *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*, 1–4. IEEE, 2008, doi: 10.1109/WiCom.2008.2650.

[102] Michael E. Tipping and Anita C. Faul. Fast marginal likelihood maximisation for sparse bayesian models. In Christopher M. Bishop and Brendan J. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*, 276–283. Proceedings of Machine Learning Research, 2003.

[103] Chun R., Haitao M., Tianfei M., and Fangquan W. Efficient structure crash topology optimization strategy using a model order reduction method combined with equivalent static loads. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 234(7):1897–1911, 2020, doi: 10.1177/0954407019893841.

[104] F. Czerwinski. Current trends in automotive lightweighting strategies and materials. *Materials*, 14(21), 2021, doi: 10.3390/ma14216631.

[105] N.N. Hussain, S.P. Regalla, Y.V.D. Rao, T. Dirgantara, L. Gunawan, and A. Jusuf. Drop-weight impact testing for the study of energy absorption in automobile crash boxes made of composite material. *Proceedings of the Institution of Mechanical Engineers, Part L: Journal of Materials: Design and Applications*, 235(1):114–130, 2021, doi: 10.1177/1464420720952813.

[106] M. Tisza and I. Czinege. Comparative study of the application of steels and aluminium in lightweight production of automotive parts. *International Journal of Lightweight Materials and Manufacture*, 1(4):229–238, 2018, doi: 10.1016/j.ijlmm.2018.09.001.

[107] J. Bian, H. Mohrbacher, J.S. Zhang, Y.T. Zhao, H.Z. Lu, and H. Dong. Application potential of high performance steels for weight reduction and efficiency increase in commercial vehicles. *Advances in Manufacturing*, 3(1):27–36, 2015.

[108] T.K. Eller, L. Greve, M.T. Andres, M. Medricky, A. Hatscher, V.T. Meinders, and A.H. van den Boogaard. Plasticity and fracture modeling of quench-hardenable boron steel with tailored properties. *Journal of Materials Processing Technology*, 214(6):1211 – 1227, 2014, doi: 10.1016/j.jmatprotec.2013.12.015.

[109] M. Maikranz-Valentin, U. Weidig, U. Schoof, H. Becker, and K. Steinhoff. Components with optimised properties due to advanced thermo-mechanical process strategies in hot sheet metal forming. *Steel Research International*, 79:92–97, 2008, doi: 10.2374/SRI07SP115-79-2008-92-97.

[110] H. Wang, M. Wan, X. Wu, and Y. Yan. The equivalent plastic strain-dependent yld2000-2d yield function and the experimental verification. *Computational Materials Science*, 47(1):12–22, 2009, doi: 10.1016/j.commatsci.2009.06.008.

[111] F. Barlat, J.C. Brem, J.W. Yoon, K. Chung, R.E. Dick, D.J. Lege, F. Pourboghrat, S-H. Choi, and E. Chu. Plane stress yield function for aluminum alloy sheets part 1: theory. *International Journal of Plasticity*, 19(9):1297 – 1319, 2003, doi: 10.1016/S0749-6419(02)00019-0.

[112] A.V. Hershey. The plasticity of an isotropic aggregate of anisotropic face centered cubic crystals. *Journal of Applied Mechanics*, 21:241–249, 1954, doi: 10.1115/1.4010900.

[113] F. Hosford. A generalized isotropic yield criterion. *Journal of Applied Mechanics*, 39(2):607–609, 1972, doi: 10.1115/1.3422732.

[114] E. Voce. Analysis of stress strain curves. *The Journal of the Royal Aeronautical Society*, 59(534):442, 1955, doi: 10.1017/S0368393100118759.

[115] H.W. Swift. Plastic instability under plane stress. *Journal of the Mechanics and Physics of Solids*, 1(1):1 – 18, 1952, doi: 10.1016/0022-5096(52)90002-1.

[116] S. Ghosh, K. Lee, and P. Raghavan. A multi-level computational model for multi-scale damage analysis in composite and porous materials. *International Journal of Solids and Structures*, 38(14):2335–2385, 2001, doi: 10.1016/S0020-7683(00)00167-0.

[117] Y. Bai. *Effect of loading history on necking and fracture*. PhD thesis, PhD. Massachusetts Institute of Technology, Cambridge, USA, 2008.

[118] M. Rogala, J. Gajewski, and M. Ferdynus. The effect of geometrical nonlinearity on the crashworthiness of thin-walled conical energy-absorbers. *Materials*, 13, 10 2020, doi: 10.3390/ma13214857.

[119] J. Fehr, P. Holzwarth, and P. Eberhard. Interface and model reduction for efficient explicit simulations - a case study with nonlinear vehicle crash models. *Mathematical and Computer Modelling of Dynamical Systems*, 22(4):380–396, 2016, doi: 10.1080/13873954.2016.1198385.

[120] K.J. Bathe, E. Ramm, and E.L. Wilson. Finite element formulations for large deformation dynamic analysis. *International Journal for Numerical Methods in Engineering*, 9(2):353–386, 1975, doi: 10.1002/nme.1620090207.

[121] E. Süli and D.F. Mayers. *An introduction to numerical analysis*. Cambridge university press, 2003, doi: 10.1017/CBO9780511801181.

[122] J.C. Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016, doi: 10.1002/9781119121534.

[123] G. Noh and K.J. Bathe. An explicit time integration scheme for the analysis of wave propagations. *Computers & Structures*, 129:178–193, 2013, doi: 10.1016/j.compstruc.2013.06.007.

[124] N.M. Newmark. A method of computation for structural dynamics. *Journal of the Engineering Mechanics Division*, 85(3):67–94, 1959, doi: 10.1061/JMCEA3.0000098.

[125] T.K. Sarkar, T. Roy, M. Salazar-Palma, and A.R. Djordjevic. Finite-element time domain method. In S.M. RAO, editor, *Time Domain Electromagnetics*, Academic Press Series in Engineering, 279–305. Academic Press, San Diego, 1999, doi: 10.1016/B978-012580190-4/50010-0.

[126] E.S.I. Group. Virtual Performance Solution. https://www.esi-group.com/, 2020. Accessed: 2020.

[127] L. Collatz and P.G. Williams. *The Numerical Treatment of Differential Equations*. Grundlehren der mathematischen Wissenschaften in Einzeldarstellungen mit besonderer Berücksichtigung der Anwendungsgebiete. Springer Berlin Heidelberg, 1966.

[128] G. Lonsdale, J. Clinckemaillie, S. Meliciani, S. Vlachoutsis, and B. Elsner. Parallel and Distributed Crashworthiness Simulation. 1996, doi: 10.5170/CERN-1996-008.173.

[129] B.P. van de Weg, L. Greve, M. Andres, T.K. Eller, and B. Rosić. Neural network-based surrogate model for a bifurcating structural fracture response. *Engineering Fracture Mechanics*, 241, 2021, doi: 10.1016/j.engfracmech.2020.107424.

[130] M. Drieschner, H.G. Matthies, T.V. Hoang, B.V. Rosić, T. Ricken, C. Henning, G.P. Ostermeyer, M. Müller, S. Brumme, T. Srisupattarawanit, K. Weinberg, and T.F. Korzeniowski. Analysis of polymorphic data uncertainties in engineering applications. *GAMM-Mitteilungen*, 42(2):e201900010, 2019, doi: https://doi.org/10.1002/gamm.201900010.

[131] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[132] D. Bashir, G.D. Montanez, S. Sehra, P. Sandoval Segura, and J. Lauw. An information-theoretic perspective on overfitting and underfitting. *arXiv preprint*, 2020, doi: 10.48550/ARXIV.2010.06076.

[133] E.T. Jaynes and G.L. Bretthorst. *Probability Theory: The Logic of Science*. Cambridge University Press, 2003.

[134] B. Rosić. *Variational Formulations and Functional Approximation Algorithms in Stochastic Plasticity of Materials*. PhD thesis, 2012.

[135] M. Mckay, R. Beckman, and W. Conover. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output From a Computer Code. *Technometrics*, 21:239–245, 1979, doi: 10.1080/00401706.1979.10489755.

[136] B.P. van de Weg. *Machine-learning-based necking prediction in sheet metal under complex load paths*. Master's thesis, University of Twente, 2019.

[137] S.E. Davis, S. Cremaschi, and M.R. Eden. Efficient surrogate model development: Impact of sample size and underlying model dimensions. In M.R. Eden, M.G. Ierapetritou, and G.P. Towler, editors, *13th International Symposium on Process Systems Engineering (PSE 2018)*, Computer Aided Chemical Engineering, 979–984. Elsevier, 2018, doi: 10.1016/B978-0-444-64241-7.50158-0.

[138] C. Kamath. Intelligent sampling for surrogate modeling, hyperparameter optimization, and data analysis. Technical report, Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States), 2021, doi: 10.2172/1836193.

[139] S. Das and S. Tesfamariam. State-of-the-art review of design of experiments for physics-informed deep learning. *arXiv preprint*, 2022, doi: 10.48550/ARXIV.2202.06416.

[140] M. Ibrahim, S. Al-Sobhi, R. Mukherjee, and A. AlNouss. Impact of sampling technique on the performance of surrogate models generated with artificial neural network (ann): A case study for a natural gas stabilization unit. *Energies*, 12(10), 2019, doi: 10.3390/en12101906.

[141] S.S. Jin. Accelerating gaussian process surrogate modeling using compositional kernel learning and multi-stage sampling framework. *Applied Soft Computing*, 104:106909, 2021, doi: 10.1016/j.asoc.2020.106909.

[142] N. Metropolis and S. Ulam. The monte carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

[143] W.K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970, doi: 10.1093/biomet/57.1.97.

[144] R.A. Fisher. *The Design of Experiments*. The Design of Experiments. Oliver and Boyd, Edinburgh, 1935.

[145] J.H. Halton. Algorithm 247: Radical-inverse quasi-random point sequence. *Commun. ACM*, 7(12):701–702, 1964, doi: 10.1145/355588.365104.

[146] I.M Sobol. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, 1967, doi: 10.1016/0041-5553(67)90144-9.

[147] H. Niederreiter. Low-discrepancy and low-dispersion sequences. *Journal of Number Theory*, 30(1):51–70, 1988, doi: 10.1016/0022-314X(88)90025-X.

[148] B. Tuffin. On the use of low discrepancy sequences in Monte Carlo methods. *Monte Carlo Methods and Applications*, 2(4):295–320, December 1996, doi: 10.1515/mcma.1996.2.4.295.

[149] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics, 1992, doi: 10.1137/1.9781611970081.

[150] J. Cheng and M.J. Druzdzel. Computational investigation of low-discrepancy sequences in simulation algorithms for bayesian networks. *arXiv preprint*, 2013, doi: 10.48550/ARXIV.1301.3841.

[151] J.G. van der Corput. Verteilungsfunktionen. I. *Proceedings. Akadamie van Wetenschappen Amsterdam*, 38:813–821, 1935.

[152] C. Lemieux. *Monte Carlo and Quasi-Monte Carlo Sampling*. Springer Series in Statistics, 2009, doi: 10.1007/978-0-387-78165-5.

[153] D. Costarelli. *Sigmoidal functions approximation and applications*. PhD thesis, Dottorato di Ricera in Matematica, Departiment di Matematica e Fisica, Sezione di Matematica, Universitada Degli Studi, Roma, 2014.

[154] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K.J. Lang. Phoneme recognition using time-delay neural networks. *IEEE transactions on acoustics, speech, and signal processing*, 37(3):328–339, 1989.

[155] D.J.C. MacKay, D.J.C.M. Kay, and Cambridge University Press. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[156] L. Behera, S. Kumar, and A. Patnaik. On adaptive learning rate that guarantees convergence in feedforward networks. *IEEE transactions on neural networks*, 17(5):1116–1125, 2006.

[157] A. Mohtashami, A. Jaggi, and S. Stich. On avoiding local minima using gradient descent with large learning rates. *arXiv preprint*, 2022, doi: 10.48550/ARXIV.2205.15142.

[158] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint*, 2014, doi: 10.48550/ARXIV.1412.6980.

[159] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

[160] A.T. Mohan and D.V. Gaitonde. A deep learning based approach to reduced order modeling for turbulent flow control using lstm neural networks. *arXiv preprint*, 2018, doi: 10.48550/ARXIV.1804.09269.

[161] A. Yaguchi, T. Suzuki, W. Asano, S. Nitta, Y. Sakata, and A. Tanizawa. Adam induces implicit weight sparsity in rectifier neural networks. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 318–325. IEEE, 2018, doi: 10.1109/ICMLA.2018.00054.

[162] J. Chung, K. Kastner, L. Dinh, K. Goel, A.C. Courville, and Y. Bengio. A recurrent latent variable model for sequential data. *arXiv preprint*, 2015, doi: 10.48550/ARXIV.1506.02216.

[163] P. Gurevich and H. Stuke. Gradient conjugate priors and multi-layer neural networks. *arXiv preprint*, 2018, doi: 10.48550/ARXIV.1802.02643.

[164] R.J. Williams and D. Zipser. *Gradient-Based Learning Algorithms for Recurrent Networks and Their Computational Complexity*, 433–486. L. Erlbaum Associates Inc., 1995, doi: 10.5555/201784.201801.

[165] P.J. Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural networks*, 1(4):339–356, 1988, doi: 10.1016/0893-6080(88)90007-X.

[166] B.P. van de Weg, L. Greve, and B. Rosić. Long short-term relevance learning. *International Journal for Uncertainty Quantification*, 2023, doi: 10.1615/Int.J.UncertaintyQuantification.2023039739.

[167] A.M. Lyapunov. The general problem of the stability of motion. *International Journal of Control*, 55(3):531–534, 1992, doi: 10.1080/00207179208934253.

[168] S. Hochreiter. *Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München.* PhD thesis, 1991.

[169] B. Hammer. On the approximation capability of recurrent neural networks. *Neurocomputing*, 31(1):107–123, 2000, doi: 10.1016/S0925-2312(99)00174-5.

[170] D.P. Flanagan and T. Belytschko. A uniform strain hexahedron and quadrilateral with orthogonal hourglass control. *International Journal for Numerical Methods in Engineering*, 17(5):679–706, 1981, doi: 10.1002/nme.1620170504.

[171] C. Tome, G.R. Canova, U.F. Kocks, N. Christodoulou, and J.J. Jonas. The relation between macroscopic and microscopic strain hardening in f.c.c. polycrystals. *Acta Metallurgica*, 32(10):1637–1653, 1984, doi: 10.1016/0001-6160(84)90222-0.

[172] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, R. Jozefowicz, Y. Jia, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, M. Schuster, R. Monga, S. Moore, D. Murray, C. Olah, J. Shlens,

B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. and Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/, 2015. Accessed: 2019.

[173] E.S. Olivas, J.D.M. Guerrero, M. Martinez-Sober, J.R. Magdalena-Benedito, L. Serrano, et al. *Handbook of research on machine learning applications and trends: Algorithms, methods, and techniques: Algorithms, methods, and techniques*. IGI global, 2009.

[174] Z. Huang, S.M. Siniscalchi, I. Chen, J. Wu, and C.H. Lee. Maximum a posteriori adaptation of network parameters in deep models. *arXiv*, 2015.

[175] A. Kaban. On Bayesian classification with Laplace priors. *Pattern Recognition Letters*, 28(10):1271–1282, 2007, doi: 10.1016/j.patrec.2007.02.010.

[176] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[177] L.P. Kadanoff. More is the same; phase transitions and mean field theories. *Journal of Statistical Physics*, 137:777–797, 2009.

[178] P.M. Williams. Bayesian regularization and pruning using a Laplace prior. *Neural computation*, 7(1):117–143, 1995, doi: 10.1162/neco.1995.7.1.117.

[179] V. Keshavarzzadeh, R.M. Kirby, and A. Narayan. Variational inference for nonlinear inverse problems via neural net kernels: Comparison to bayesian neural networks, application to topology optimization. *Computer Methods in Applied Mechanics and Engineering*, 400, 2022, doi: 10.1016/j.cma.2022.115495.

[180] B.C. Arnold and S.J. Press. Compatible conditional distributions. *Journal of the American Statistical Association*, 84(405):152–156, 1989.

[181] P. Berti, E. Dreassi, and P. Rigo. Compatibility results for conditional distributions. *Journal of Multivariate Analysis*, 125:190–203, 2014, doi: 10.1016/j.jmva.2013.12.009.

[182] M.E. Khan and H. Rue. The bayesian learning rule. *arXiv preprint*, 2021, doi: 10.48550/ARXIV.2107.04562.

[183] J. Plante. A proof of bonnet's version of the mean value theorem by methods of cauchy. *The American Mathematical Monthly*, 124(3):269–273, 2017.

[184] H. Thodberg. Bayesian backprop in action: Pruning, committees, error bars and an application to spectroscopy. In J. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems*. Morgan-Kaufmann, 1993.

[185] M.A. Woodbury. *Inverting Modified Matrices*. Memorandum Report / Statistical Research Group, Princeton. Department of Statistics, Princeton University, 1950.

[186] D. Shutin, T. Buchgraber, S.R. Kulkarni, and H.V. Poor. Fast Variational Sparse Bayesian Learning With Automatic Relevance Determination for Superimposed Signals. *IEEE Transactions on Signal Processing*, 59(12):6257–6261, 2011, doi: 10.1109/TSP.2011.2168217.

[187] D.R. Jones. A Taxonomy of Global Optimization Methods Based on Response Surfaces. *Journal of Global Optimization*, 21(4):345–383, 2001, doi: 10.1023/A:1012771025575.

[188] P.I. Frazier. A tutorial on bayesian optimization. *arXiv preprint*, 2018, doi: 10.48550/ARXIV.1807.02811.

[189] A. Nouy. Proper generalized decompositions and separated representations for the numerical solution of high dimensional stochastic problems. *Archives of Computational Methods in Engineering*, 17(4):403–434, 2010.

[190] E.H. Moore. On the reciprocal of the general algebraic matrix. *Bulletin of the American Mathematical Society*, 26(9):394–395, 1920.

[191] X. Wan and G.E. Karniadakis. Multi-element generalized polynomial chaos for arbitrary probability measures. *SIAM Journal on Scientific Computing*, 28(3):901–928, 2006, doi: 10.1137/050627630.

[192] J.A. Nelder and R. Mead. A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313, 1965, doi: 10.1093/comjnl/7.4.308.

[193] M.J.D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162, 1964, doi: 10.1093/comjnl/7.2.155.

[194] M. Onderwater. Outlier preservation by dimensionality reduction techniques. *International Journal of Data Analysis Techniques and Strategies*, 7:231 – 252, 2015, doi: 10.1504/IJDATS.2015.071365.

[195] I. Sraj, O.P. Le Maître, O.M. Knio, and I. Hoteit. Coordinate transformation and polynomial chaos for the bayesian inference of a gaussian process with parametrized prior covariance function. *Computer Methods in Applied Mechanics and Engineering*, 298:205–228, 2016, doi: 10.1016/j.cma.2015.10.002.

[196] J. Yang, B. Faverjon, H. Peters, and N. Kessissoglou. Application of polynomial chaos expansion and model order reduction for dynamic analysis of structures with uncertainties. *Procedia IUTAM*, 13:63–70, 2015, doi: 10.1016/j.piutam.2015.01.017.

[197] N. El Mocayd, M. Shadi Mohamed, D. Ouazar, and M. Seaid. Stochastic model reduction for polynomial chaos expansion of acoustic waves using proper orthogonal decomposition. *Reliability Engineering & System Safety*, 195, 2020, doi: 10.1016/j.ress.2019.106733.

[198] T. Crestaux, O. Le Maître, and J.M. Martinez. Polynomial chaos expansion for sensitivity analysis. *Reliability Engineering & System Safety*, 94(7):1161–1172, 2009, doi: 10.1016/j.ress.2008.10.008. Special Issue on Sensitivity Analysis.

[199] U. Kjems, L.K. Hansen, and S.C. Strother. Generalizable singular value decomposition for ill-posed datasets. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, 528–534, Cambridge, MA, USA, 2000. MIT Press.

[200] D.W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. Graduate Texts in Mathematics. John Wiley and Sons, New York, 1992, doi: 10.1002/9780470316849.

[201] K. Shridhar, F. Laumann, and M. Liwicki. Uncertainty estimations by softplus normalization in bayesian convolutional neural networks with variational inference. *arXiv preprint*, 2018, doi: 10.48550/ARXIV.1806.05978.

[202] G.E. Hinton and D. van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, 5–13, New York, NY, USA, 1993. Association for Computing Machinery, doi: 10.1145/168304.168306.

List of publications

Journal papers

Author

[1] B.P. van de Weg, L. Greve, and B. Rosić. Long short-term relevance learning. *International Journal for Uncertainty Quantification*, 2023, doi: 10.1615/Int.J.UncertaintyQuantification.2023039739

[2] B.P. van de Weg, L. Greve, M. Andres, T.K. Eller, and B. Rosić. Neural network-based surrogate model for a bifurcating structural fracture response. *Engineering Fracture Mechanics*, 241, 2021, doi: 10.1016/j.engfracmech.2020.107424

Co-author

[3] L. Greve and B.P. van de Weg. Surrogate modeling of parametrized finite element simulations with varying mesh topology using recurrent neural networks. *Array*, 100–137, 2022, doi: 10.1016/j.array.2022.100137

[4] L. Greve, B. Schneider, T.K. Eller, M. Andres, J.D. Martinez, and B.P. van de Weg. Necking-induced fracture prediction using an artificial neural network trained on virtual test data. *Engineering Fracture Mechanics*, 219, 2019, doi: 10.1016/j.engfracmech.2019.106642

Conferences

[5] B. P. van de Weg, L. Greve, B. Rosić, Surrogate modeling for finite element simulations, VDI Simvec, 2022.

[6] B.P. van de Weg, B. Rosić, Auto-adapting multi-element polynomial chaos expansion, Engineering mechanics symposium, 2022.

[7] B.P. van de Weg, B. Rosić, Long short-term relevance learning, Engineering mechanics symposium, 2021.

[8] B.P. van de Weg, L. Greve, B. Rosić, Interactive crash simulation, Engineering mechanics symposium, 2020.

[9] L. Greve, M. Andres, B.P. van de Weg, Necking prediction using neural networks, VDI Conference Automotive CAE, 2019 (Co-Author).

Prizes

1. 3^{rd} best presentation (2022 VDI Simvec conference, Baden-Baden, Germany)
2. Best presentation (2020 Engineering Mechanics symposium)