Bayesian Network Metamodels for Inference-driven design space exploration

Submitted by

Zack Xuereb Conti

Thesis Advisor(s)

Stylianos Dritsas, Shaowei Lin, Sawako Kaijima

Architecture and Sustainable Design

A thesis submitted to the Singapore University of Technology and Design in fulfillment of the requirement for the degree of Doctor of Philosophy

2019

ABSTRACT

This dissertation presents a statistical approach to assist intelligent decision-making in the early stages of design, when coupling engineering simulation with computational design systems.

The coupling of design and analysis facilitates the navigation of a design space with respect to measurable engineering criteria. In practice, this typically involves manipulating the values of the input parameters and observing the simulation output, in a trial and error fashion or automatically, using an optimization algorithm. However, typical design-analysis systems are unidirectional. Thus, when considering many parameters, it becomes very difficult to maintain control over the computational system because a cyclic approach to design and analysis externalises knowledge accumulation to human cognition. In other words, it becomes challenging to keep track of all cause-effect relationships, to generate a comprehensive understanding of the multi-dimensional design space.

Instead, the research presents a statistical approach to generate abstractions of typical coupled systems in the form of probabilistic models, using Bayesian networks (BN). BNs facilitate reasoning about cause and effect relationships over multiple dimensions while their probabilistic representation, facilitates a broader representation of the design space than, with typical coupled systems. In our approach we take advantage of the fact that BNs do not distinguish between inputs and outputs and thus, enable inverse reasoning from effect to cause. The capacity to reason about inverse scenarios within a probabilistic representation enables to fix a target on a response of interest and quickly identify the input ranges that are likely to generate favourable responses within the set target. In other words, we can narrow down a vague understanding of a design space into the meaningful regions of interest. This suggests a shift from decision-making with discrete choices towards a ‘softer’ abstraction of the design space to assist the intuition.

Through a case study we demonstrate how a probabilistic bi-directional mechanism can be useful for both architects and engineers to translate engineering constraints into architectural constraints and to communicate expert feedback to the architecture team in the form of soft knowledge.

ACKNOWLEDGEMENTS

I would like to show my gratitude towards my main Doctoral advisor Sawako Kaijima, whose rigorous mentorship has guided me through a challenging learning curve, particularly during the early phases of my research. I would also like to thank my co-advisors Stylianos Dritsas and Shaowei Lin for their insightful feedback throughout my research journey.

I would like to show my appreciation towards the team at BayesiaUSA, particularly Stefan Conrady, for sponsoring me with a license to their BayesiaLab software, and for their technical assistance.

I remain grateful towards Paul Shepherd, and Paul Richens whose mentorship throughout my MPhil degree in Digital Architectonics at Bath, provided me with a fundamental stepping stone to my Doctoral research.

I would like to thank my colleagues Narasimha Bodetti and Oliver Weeger for their valuable input in mechanical engineering related aspects of the research. Furthermore, I would also like to show my appreciation towards my office colleagues whose multi-disciplinary background, has contributed to my work directly and indirectly, in various ways.

I would like to thank my dear friends, Özgün, Priji, Barnabe, Giacomo, Fredy, Ezgi, Bharath, Sandra and Thommen, whose friendship and support have made Singapore feel like a home away from home.

Last but not least, I would like to show my immense gratitude towards my mother and father, whose incessant love and support throughout my academic journey was instrumental to reaching thus far.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

This dissertation aims to assist intelligent decision-making when coupling engineering simulation with computational design systems. The research presents a statistical approach to generate abstractions of typical coupled systems in the form of probabilistic models. A probabilistic representation facilitates bi-directional reasoning between inputs and outputs over a broader representation of the design space, than with singular outcomes in typical design and analysis systems.

This chapter motivates the research goal by challenging the typical notion of coupled design-analysis systems for informing design choices. We begin by arguing that the deterministic format of typical computational design systems, does not promote comprehensive knowledge elicitation of a design space, particularly in high dimensional input scenarios.

1.1 Design-analysis systems

'Computational design system' is a general term used to describe different approaches of using a computer's numerical abilities to assist a design process. These include parametric, algorithmic, and generative approaches. While it is not the focus of this dissertation to distinguish between these approaches, we can define a generic computational design system as a set of interrelated components that operate collectively to assist with accomplishing a goal.

Faster computers together with better numerical algorithms have extended the availability of engineering simulation to domains outside of engineering-centric fields, such as building design. The increased availability of simulation tools together with the emergent systematic approach of describing design attributes through parameters have facilitated the coupling of the two, to inform decisions about engineering concerns from the early stages of design (Shea, Aish, & Gourtovaia, 2005). A coupled system allows the navigation of a design space with respect to measurable engineering criteria. In typical practice, this involves manipulating the values of the parameters in the computational design system and observing the quantitative output of the simulation, in a trial and error fashion or iteratively, using an optimization algorithm.

In today's practices, the integration of engineering simulation tools with computational schemas has become a standard approach within both architects' and expert engineers' workflows. The coupling of the two offers advantages for both, automation and discovery; automation of tedious analogue processes and a tool for discovery of new solutions. Furthermore, the availability and ease with which such coupled systems can be integrated in design workflows, has facilitated engineering tasks in the early stages, to the design team. Of course, this has not replaced the role of consultant engineers in a project; on the contrary, this has improved the communication between the two through a more integrated approach to building design.

However, in this dissertation we highlight that the way in which we assemble and operate computational design systems in general, has not changed since their inception, despite the emergence of the computational domain. More particularly, we highlight that progress in computational design methods focuses predominantly on techniques for generating design alternatives, and much less on supporting intelligent inferences. By 'intelligence' we intend, the ability to comprehend the relationships between cause and effect and thus, facilitate the ability to control and manipulate design, through a comprehensive understanding of the engineering constraints.

In the following sections we attribute this main challenge to the fact that typical computational design systems have a deterministic representation, meaning that they assume as forward system between design and analysis, which can only process and compute information about one scenario at a time, thus, renders design exploration as a one-directional cyclic loop. This argument motivates the need for a bi-directional approach to computational design-analysis systems.

1.2 Computational systems are deterministic

A deterministic system is one in which, outcomes are precisely determined through an explicit description of relationships, implying no room for variation. A deterministic model is analogous to an analogue clock mechanism, where the geometric relationships between gears and cogs are known and defined explicitly and thus, can be drawn very precisely to work together and subsequently rotate the clock hands, leaving no room for variation.

In this context, computational design systems and simulation software used for building design are considered to be deterministic because outcomes are precisely determined through an explicit description of rules and relationships. More specifically, typical computational design systems can only handle discrete scenarios because the formal rules tying the components together are defined explicitly. Additionally, simulation software typically used for building design is also

considered to be deterministic because, the mathematical expression underlying the hood of a numerical analysis model is defined very precisely, based on very well defined laws of physics. Therefore, overall, the deterministic representation of typical coupled systems intake scalar inputs and generate a singular outcome thus, can only generate and analyse only one design scenario at a time (Figure 1.1).

As a result, we argue that the deterministic representation of typical coupled systems limits intelligent decision-making, because the singularity of the output does not provide any insight about the causal relationships between the design parameters and the simulation response. Furthermore, it becomes difficult to consolidate a comprehensive interpretation of the design space from the design and analysis of singular instances, especially in the presence of multiple parameters where correlations between the simulation inputs might be influencing the simulation response.

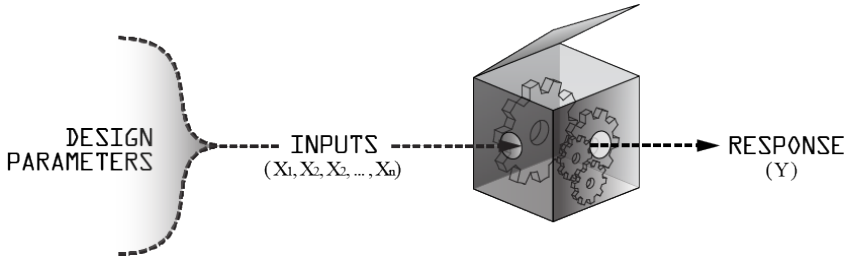


Figure 1.1: Exact inputs and outputs of a deterministic design-analysis systems.

When coupling simulation with a computational design system, we are intrinsically setting up a forward system. In other words, a system that computes only in one direction; input parameters→ simulation response (Figure 1.1). We assert that the one-directionality of the coupled system, together with the explicitness of the simulation inputs and outputs, transform so-called 'design exploration' into a cyclic feedback loop between design and analysis, which becomes difficult to guide intuitively, when navigating high dimensional design spaces. In other words, the coupled system becomes a black box whose exploratory scope becomes limited to cyclic *searches* of singular points in a design space (Figure 1.2), either manually by trial and error, or automatically using a stochastic search algorithm.

Random and automatic search-based approaches can at times result in surprising outcomes or ones that inspire further outcomes. However, in general we claim that search-based approaches are not suitable for eliciting a comprehensive understanding of high dimensional design spaces and thus, do not support intelligent control.

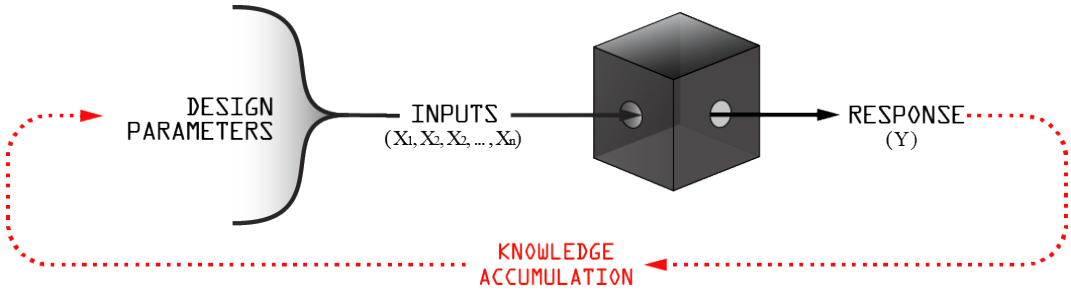


Figure 1.2: Externalisaiton of knowledge accumulation in typical unidirectional cyclic design exploration.

We argue that the cyclic representation of the coupled system externalises knowledge-accumulation hence, becomes dependant on human cognition to keep track of cause and effect relationships between parameters and response. As a consequence, it becomes very challenging to make intelligent-decisions because a shallow understanding of the design space, does not facilitate architects and engineers with control over the system, especially when considering nonlinear relationships due to correlations between parameters. In other words, the system becomes difficult to keep track of, implying a difficulty to control the engineering behaviour.

In typical building design, it is desirable for designers to maintain an intuitive control over the design aspects of the project. In other words, if a designer is not able to control a design-analysis system, then the role of a computational approach for informing early stage decisions becomes useful only with automatic methods. Some might argue that the use of automatic stochastic a search approach would be suitable for such cases. We acknowledge that automatic design space search methods may reveal interesting or spontaneous outcomes however, the lack of insight about the causality of such outcomes provides little power to manipulate intelligently and develop further the outcome, thus renders it as an independent 'outlier'. In practice, designers are likely to abandon tools and methods, even trendy ones, when their intuition is bypassed and thus, when their hand over control is compromised.

Instead, the intuition of a designer in the early stages of a project, desires guidance and flexibility– guidance for navigating high dimensional design space scenarios, and the facility to make informed flexible choices, without committing early on. A good human-computer relationship is one that assists the intuition, not bypasses it. In this context, we argue the use of design-analysis systems for early stage design should provide assistance with drawing a comprehensive understanding of the design space to guide intelligent navigation, even in high dimensions. Furthermore, the coupled system must communicate a broader representation of the inputs and the outputs to facilitate a richer insight into cause and effect, the outcome of which can result in flexible choice-making.

Overall efforts should focus on guiding the intuition (not substitute it). Eliminating memory load from the cognition to keep track of relationships, frees up the capacity for the intuition to take central control of creative manipulation and of the overall convergence of the final design outcome, instead of some stochastic algorithm.

1.3 Research questions and research goals

1.3.1 Questions

The questions we need to ask are the following:

1) How can we represent design-analysis systems to assist architects and engineers with making intelligent decisions when considering multiple parameters?

2) How can we represent design-analysis systems to provide the capacity for broader choices in the early stages of design?

1.3.2 Goals

The research questions suggest the need for a representation of design-analysis systems that can:

1) keep track of input-output relationships to eliminate dependence on cognition yet facilitate a comprehensive understanding of the design space in high dimensional scenarios, and
2) provide a broader representation of the inputs and outputs, to provide the capacity for richer choices in the early stages of design.

Therefore, we set 1 and 2 as the research goals for this dissertation. To summarise further, the goals pertain tasks of *knowledge-elicitation* and *knowledge-representation*. These are mutually dependant and thus require an approach that addresses both criteria simultaneously.

The combined goal of this research aims to overcome the one-directionality limitations of typical design-analysis systems by generating system abstractions that facilitate comprehensive decision-making over a broader representation of the design space.

1.4 A probabilistic approach

Notable work by Kilian (2006), initiated the conversation about introducing bi-directionality in design exploration, as a means to overcome the limitations posed by the one-directionality of typical design evaluation digital systems. Kilian highlights how very few research work has addressed the challenge of reverse mapping.

> *"...very few research projects allow the reversal of the process. This has to do with the non-deterministic nature of reverse mapping, as there are infinitely many possible solutions even for the simplest conditions."*

This statement defines very clearly, the basis of the approach developed in this dissertation and highlights how not much development has occurred on this front, since.

In this dissertation we achieve 'non-deterministic mapping' through a probabilistic approach. We draw parallels with a well studied problem in the statistics community; that of 'reasoning over uncertainty'. In general, reasoning over uncertainty refers to the difficulty with informing decisions based on information that is not certain, hence the term, 'uncertain'. In this research, we find that the way in which uncertain information is represented and reasoned with in the statistics domain, is analogous with our research scope, to reason bi-directionally over a broader representation of the design space.

Typical applications in applied statistics represent uncertainty scenarios as a probabilistic model, where such a representation does not compute singular information, but instead, computes uncertain inputs in the form of probability distributions of likely values. This way, analysts are able to reason about causal insight while keeping values 'broadly defined', hence, without making any compromising assumptions about fixed (certain) input and output values.

In this research, we take advantage of such a probabilistic representation to wrap coupled design-analysis systems into a broad representation where input parameters and simulation outputs are represented by probability distributions in a probabilistic model representation. In our approach we derive the probability distributions directly from simulation data; in other words, by generating input configurations in the design space and evaluating the response of each.

Subsequently, a probabilistic model representation of the design space facilitates architects an engineers to reason about cause and effect relationships while maintaining a broad overview of

the design space through their probabilistic representation. Furthermore, decision making through a probabilistic reasoning approach generates a broader range of outcomes. We argue that reasoning about broader choices, is a desirable attribute of a designer's intuition in the early stages of a design project.

Therefore, our approach suggests a shift, from deterministic decision-making based on singular points in a design space, to soft decision-making based on reasoning about meaningful regions in the design space that concern questions of interest, the outcome of which result in a gradation of choices.

In this research we adopt Bayesian networks, which are a popular type of probabilistic model for reasoning over uncertainty in multi-dimensional scenarios. Bayesian networks combine techniques from classic probability theory and machine learning to consolidate intricate networks of relationships underlying challenging problems. In other words, Bayesian networks are useful because they unload the demand to keep cognitive track of relationships in high-dimensional scenarios. Their applications vary from applications in the US military, to stock market prediction.

In this context, our application of Bayesian networks in this research offers a twofold approach to 1) consolidate and keep track of intricate cause-effect relationships in high dimensions, and whose 2) probabilistic representation enables reasoning over a broad representation of the design space thus, facilitates flexible decision-making.

It is important to clarify that there exist two interpretations of probability in the statistics community; frequentist and Bayesian. A frequentist interpretation views probabilities as counts of outcomes over repeated experiments, whereas a Bayesian interpretation views probabilities as a quantification of a 'guess' or 'belief' based on knowledge. We emphasise that the use of the term 'probability' in this dissertation does not imply a frequentist interpretation because the input and output probability distributions are not derived from a repeated experiment (e.g. flipping a coin). Rather, the probability distributions are derived from a numerical simulation model which is typically deterministic.

Therefore, our use of the term 'probabilistic' in this work, leans towards a Bayesian perspective where probabilistic reasoning with the model implies reasoning with beliefs or guesses about the possible input or response values. These beliefs are computed based on the limited information that the Bayesian network has about the numerical simulations, and is analogous to a human expert gaining intuition about a system after observing its complex behaviour at a few data points. The algorithms used in Bayesian networks are based on Bayesian methods in statistics - methods

which interpret probabilities as beliefs rather than expected frequencies in repeated experiments. Bayesian networks are explained in more detail in Chapter 3 and Chapter 4.

The notion of representing a simulation model into a compact statistical model, is common practice in fields of aerospace and automotive engineering. Such a representation is commonly referred to a *simulation metamodel*. In this dissertation we adopt Bayesian networks as a metamodel representation ergo the term, Bayesian network metamodel (BNM).

1.5 Bayesian network metamodel

The search for a relationship between a set of inputs $(x_1, \dots, x_n)$ and an output (y) naturally leads us to the statistical task of 'model approximation'. In fields like mechanical engineering, statistical techniques such as regression are borrowed to substitute complex numerical simulation code with a *metamodel* (or surrogate model), which is a simpler and more computationally efficient model to compute.

In this context, we adopt a statistical representation of the coupled design-analysis system such that we can capture relationships between input parameters and response variables directly from *generated* input-output simulation data, and *represent* the relationships in a statistical model that can be exploited for decision-making. In other words, we bypass the need to look inside the domain-expertise of the 'black-box', by approximating relationships between its inputs and its outputs.

Let us first briefly introduce traditional simulation metamodels to provide background and further justification for the notion of a probabilistic metamodel using Bayesian networks.

1.5.1 Typical metamodels

A typical metamodel can be described as a 'model of a model' (J. P. C. Kleijnen, 1986), typically expressed as Eqn. (1), where y is the simulation response, f is the simulation model and $\hat{f}$ is the approximated model.

$$y = f(x) \approx \hat{f}(x) \tag{1}$$

Quicker prediction of y is therefore obtained using $\hat{f}$. The most common metamodeling techniques for approximating simulation models include response surfaces and polynomial

regression (J. P. Kleijnen, 2008), (J. P. Kleijnen & Sargent, 2000), Kriging (Ankenman, Nelson, & Staum, 2010), and neural networks. Section 2, will elaborate further each of these methods and their applications in literature pertaining to building-design and surrounding fields.

We acknowledge that metamodels are a suitable approach for consolidating relationships into a compact representation. However, we find that the representation of traditional metamodel approaches render the relationships inaccessible for insight-gain, when the number of inputs are numerous. Furthermore, the representation of typical metamodels are mostly deterministic, thus not resolving the pitfalls discussed earlier.

In more detail, we argue that the functional representation $\hat{f}(x)$, by which typical metamodels are expressed renders the relationships inaccessible for reasoning, because the inputs and the outputs of a typical metamodel concern exact scalar values. Subsequently, when introduced into the computational design system environment, the metamodel re-encounters the one-directional feedback loop scenario discussed previously. In fact, the predominant goal of metamodel applications in literature pertain to faster prediction for increasing the number of design-analysis-optimisation cycles rather than for explanatory purposes.

Having said this, the few applications of typical metamodels for explanatory purposes in literature, demonstrate to provide meaningful insight. Predominantly, these applications take advantage of the faster computation achieved by the metamodel to generate significantly more data and subsequently adopt further statistical methods that focus on knowledge-discovery. These statistical methods include classic methods such as correlation analysis, methods from design of experiments such as factorial designs, methods from sensitivity analysis, and also advanced methods in descriptive statistics to plot and illustrate complex information.

Despite, knowledge-discovery becoming a two step process (metamodel building followed by knowledge-discovery), we acknowledge their usefulness. However, the cognitive challenge remains; the format of the insight outputted by these applications is deterministic and hence, does not provide the capacity of making flexible choices. In other words, when dealing with multiple inputs, the cognitive load to keep track of disconnected information such as correlation values, total, primary and higher order sensitivity values, or simply illustrative plots, remains.

1.5.2 Probabilistic metamodel

Instead, we opt for a probabilistic approach, inspired from the problem of ‘reasoning over uncertainty’ in the statistics community. We present a probabilistic representation of the design-

analysis system as a probabilistic model, with which we can make intelligent decisions over a high-resolution representation of the design space.

In more detail, the main advantage of a probabilistic representation is *probabilistic inference*. It is probabilistic inference that handles probabilistic reasoning over uncertain information.

In general, 'inference' can be defined as the process of drawing conclusions based on data. More specifically, we look at 'Bayesian' inference, which is a type of inference technique to manipulate one or multiple probability distributions of interest and observe how remaining probability distributions in the probabilistic model change, based on their causal relationships. In other words, we see Bayesian inference as a powerful mechanism to navigate relationships between soft representation of input parameters and engineering responses.

Relationships between probability distributions, also referred to as probabilistic relationships, differ from functional relationships in typical metamodels of the form $\hat{f}(x)$, in that they do not compress/approximate data into hard relationships defined by some constant value. Instead, relationships between probability distributions are observed as a joint probability distribution (JPD). A JPD can be defined as a representation of the probability of every possible combination of values of each variable (Binder, Koller, Russell, and Kanazawa (1997) (Figure 1.3).

Chapter 3 will introduce the notion of a JPD more formally and provide a deeper understanding of what they mean, how they operate and how we take advantage of their representation to achieve the projected research goal.

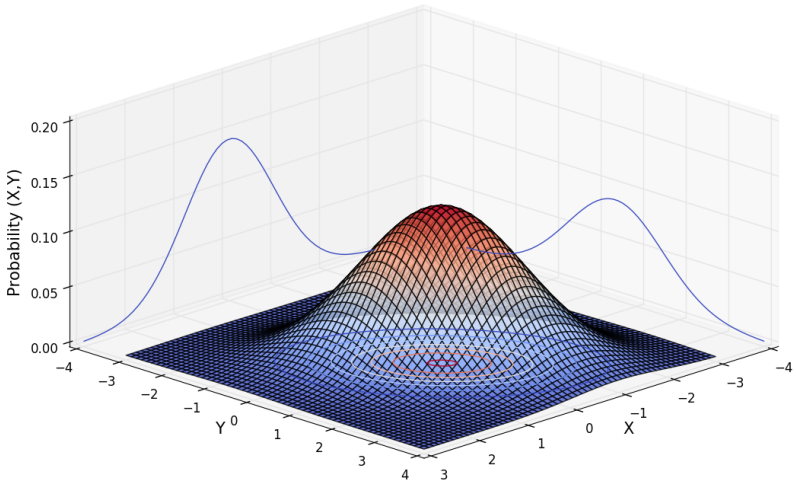


Figure 1.3: Joint probability distribution.

The main justification of a JPD representation in this research, is that it does not distinguish between the 'right-hand side' (inputs) and 'left-hand side' (output) of an equation. In our metamodel application, we take advantage of this mathematical indifference between inputs and outputs in two significant ways:

1) To achieve bi-directional inference. We treat the the typical 'forward' input→output design-analysis problem in the form of bi-directional representation that also allows the reasoning about the inverse problem (input←output). The bi-directionality can provide insight into the causality of design variables on the engineering simulation response.

2) To allow the consideration of multiple inputs and outputs. The lack of distinction between input and outputs, allows the consideration of multiple design parameters and engineering responses in the same metamodel.

1.5.3 Probabilistic Graphical Models (PGM)

In this dissertation, we adopt Probabilistic Graphical Models (PGM) to enable probabilistic inference over multiple inputs and outputs. A PGM is a type of statistical model that can represent a high dimensional JPD very efficiently. More specifically, we adopt Bayesian networks, which are a type of PGM. Our application of Bayesian networks as metamodels suggests the term 'Bayesian network metamodel (BNM).

Bayesian networks are capable of extracting causal relationships between variables, directly from data using machine learning techniques, and enable reasoning over high-dimensional uncertainty, by means of inference techniques from probability theory. Therefore, we may conclude that introducing BNMs at the architecture-engineering interface provides the capacity to make decisions based on a richer representation of the relationships between design parameters and engineering response variables, the outcomes of which result in a gradation of outcomes instead of a singular scenario.

Chapter 3 introduces Bayesian networks formally, explains how to construct a BNM and how it can be used for decision-making. BNMs are fairly novel in the metamodeling community. Section 3.5 provides a review of the few BNM applications that exist.

1.6 Contribution

This dissertation presents a knowledge-driven approach to engineering simulation for architectural design decisions.

> *"The role of science or engineering is not just to repeat the experiment, but surely to produce some causal model, some explanation. If we can build this kind of building engineering causal model into our design tools, then these can be directly accessed and inform the design process. We can actually build the prediction of the experience into the design. That is what completes the loop because essentially we want design with feedback."*
>
> *(Aish (2005) in Kolarevic & Malkawi, 2005)*

With the proposed Bayesian network metamodel approach, we urge to rethink the way in which we utilise simulation in computational design systems to learn about engineering behaviour. In this context, this dissertation proposes a shift from utilising engineering simulation as an iterative feedback loop between design and analysis (Figure 1.4, left), towards encapsulating engineering knowledge, into a probabilistic representation—namely a Bayesian network metamodel, that enables bi-directional reasoning between design and analysis (Figure 1.4, right).

In other words, we shift from a notion of exploring points in a design space, towards navigating relationships and manoeuvring constraints, to provide the broader capacity for creative inferences while taking all input-output relationships into account.

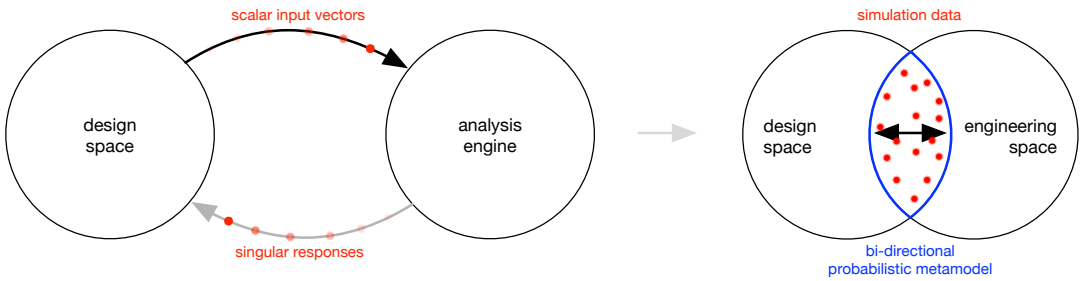


Figure 1.4: Shift from feedback loop to bi-directional metamodel.

1.6.1 Bi-directional translational mechanism

Let us analogise that the metamodel inputs represent the architecture domain and the metamodel output represents the engineering domain. We argue that the bi-directionality of the Bayesian network metamodel (BNM) together with its probabilistic representation, can be taken advantage of as a translational mechanism between the two domains.

1.6.1.1 Interpreting engineering feedback

It can be difficult for architectural designers to interpret the simulation output without expertise in the underlying engineering phenomenon. This can lead to trusting numerical results blindly, without a chance of explaining the causal framework of the numerical output.

On the other hand, it can be difficult for engineers to communicate the results of their expertise back to architectural designers, in qualitative terms because they may not always fully understand the design intent.

In this context, we argue that a BNM can act as a bi-directional translation intermediary between the two domains. Where, the use of bi-directional inference can help engineers to communicate engineering feedback in terms of design parameters, based on the relationships between the two. This way, engineers only deal with imposing constraints on the output distributions of the BNM, based on their expertise and experience while, architects only deal with the input distributions to interpret engineering constrains in terms of design constraints. The 'translation' with a BNM may lead to richer decisions because probabilistic inference can take into account the entire network of relationships between the inputs and the outputs when computing information.

Translating physical phenomena into qualitative terms is not new. In fact, we draw inspiration from the field of 'qualitative physics', which is an area in artificial intelligence concerned with "representing and reasoning about the physical world". For example, areas of qualitative physics focus on addressing "the problem of how to represent differential equations qualitatively, and how to organize such knowledge in a usable form" (Forbus, 1988). The aim of qualitative physics resonates with the scope of this dissertation.

1.6.1.2 Flexible choices in the early stages

A BNM can be used to improve communication between engineers and architects in two ways: flexibility and translation.

Consulting engineers typically require a discrete number design options for them to provide feedback on. This implies, (1) architects need to converge flexible thinking into discrete set of options for evaluation, or (2) domain experts can only be consulted at later stages when design goals are clearer.

The probabilistic representation together with the bi-directionality of a BNM, can be utilised to maintain design freedom when consulting expert engineers on a project. More specifically, a BNM approach enables architects to hand over a flexible parametric model to the engineers, and in turn, engineers may communicate engineering constraints in the form of feasible input ranges.

As a result, the architects are presented with a parametric model that is informed with feasible ranges on the parameters that satisfy engineering criteria, imposed by the engineers. Thus, expert feedback is not limited to discrete scenarios anymore but embedded in the form of soft feedback within the computational design system.

1.6.1.3 "Soft" optimisation

A BNM allows to specify a target of interest from the output probability distribution, and use bi-directional inference to predict what the input distributions should be, based on the causal relationships between the two.

In other words, the bi-directional inference can be used to instantly identify input distributions that satisfy some target objective, for example minimise or maximise response values. Furthermore, the capability for multiple outputs also enables to identify the input distributions that satisfy multiple target response objectives. The latter functionality can allow exploration of trade-offs between response targets.

This results in a set of "soft" input ranges, the combination of which, are likely to yield the desired target objectives.

1.6.2 Contribution to metamodeling community

The notion of utilising Bayesian networks as metamodels for simulation data is a novel contribution in itself. There is very little literature adopting Bayesian networks as metamodels in both the Bayesian network community and the metamodeling community. A literature review of metamodeling applications using applications can be found in Chapter 4.

1.7 Chapter breakdown

Chapter 2 introduces simulation metamodels, explains how simulation metamodels are constructed and provides a review on typical methods and applications as a means to provide justification for adopting probabilistic graphical models.

Chapter 3 introduces probabilistic models and probabilistic inference, as an alternative approach to typical metamodels.

Chapter 4 provides a step by step description on how to build a Bayesian network metamodel.

Chapter 5 describes two software packages developed for sharing and making Bayesian network metamodels accessible. One is a python library package for technical-inclined users and one is built as a plug-in within Rhino3D/ Grasshopper aimed at the intuition-oriented user.

Chapter 6 illustrates a case study application from the 2016 Serpentine Pavilion in London, using the BNM metamodel.

Chapter 7 provides a concluding arguments and describes future work.

Chapter 2

SIMULATION METAMODELS

2.1 What is a simulation metamodel?

By definition, a metamodel is an approximation of the input-output function underlying the numerical simulation model. Metamodels are used widely in engineering fields to substitute computationally demanding simulation with a mathematical approximation, that is much quicker to compute.

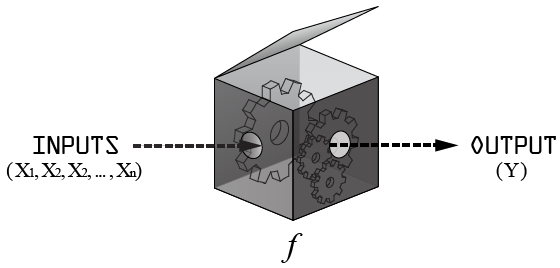


Figure 2.1: Inputs and outputs of the simulation model.

2.1.1 Theoretical background

The term metamodel was popularised by Jack Kleijnen (J. P. C. Kleijnen, 1986), who expresses a metamodel as a "model of a model". Early works referred to metamodels as "regression" or "experimental designs" (Burdick & Naylor, 1966; Hunter & Naylor, 1970; Walsh, 1963). In the field of statistics, a 'regression problem' is equivalent to the task of 'model approximation'. That is the search for a relationship between *inputs* (X) and *outputs* (Y). In the metamodeling community, the study of a computer simulation is referred to as a 'computer experiment'. In computer simulation experiments, analysts use regression to: (1) substitute the computationally expensive simulation model with a regressed mathematical model that is much more efficient to compute (Law, 2015; Simpson, Peplinski, Koch, & Allen, 1997) and (2) yield some insight into the relationship between the inputs and the observed outputs of the simulation analysis. The former is the most popular scope for their use in literature. The term 'metamodeling' is used to describe a process of constructing a metamodel. Metamodeling techniques for the purpose of approximating simulation models include: polynomial regression models (J. P. Kleijnen, 2008),

spline models (R. R. Barton, 1998), neural networks (Fonseca, Navaresse, & Moynihan, 2003), Kriging models (Ankenman et al., 2010), response surfaces (J. P. Kleijnen & Sargent, 2000), and game theoretic models (Poropudas & Virtanen, 2010a, 2010b).

In general, we can describe a standard metamodel as:

$$y = f(x) \approx \hat{f}(x) \tag{2}$$

where y is the simulation response, f denotes the mathematical function implicitly defined by numerical algorithm implementing the simulation model, $\hat{f}$ is the approximated model (metamodel). Predicted values (y) are therefore obtained using $\hat{f}$.

2.1.2 Terminology

In the metamodeling community, simulation inputs or parameters may be referred to as 'factors'. For this reason, any reference to factors implies inputs or input parameters. Furthermore, the output of a simulation may at times be referenced to as a 'response'. The term 'parameters' may at times refer to regression parameters not to be confused with design parameters. A clear distinction will be made when this is the case.

2.2 How do we build a simulation metamodel?

Formulating a metamodel typically involves the following four steps (Figure 2.2): (1) generate configurations of input values, (2) run the simulations to generate response data, (3) use the input-output data to 'fit' a statistical model to data points, and finally (4) validate the robustness of the model. Steps 1 and 2 deal with the design of a computer experiment, while steps 3 and 4 deal with the building of a reliable statistical approximation. The following subsections will explain each of these steps in further detail.

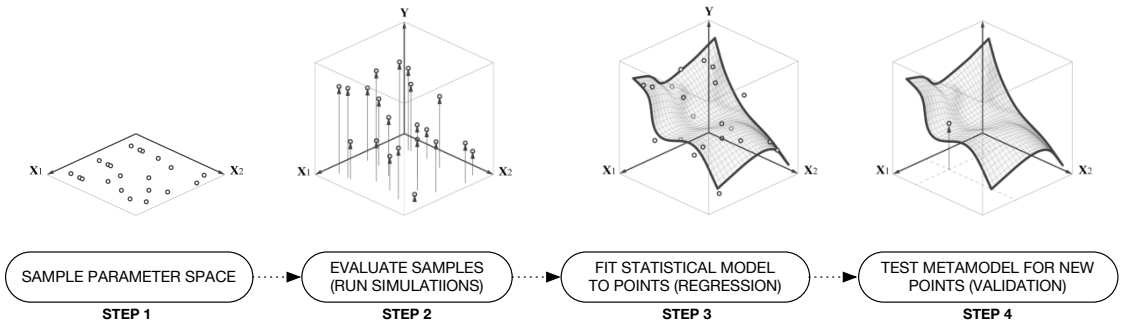


Figure 2.2: Typical metamodeling workflow.

2.2.1 Sample the parameter space (step 1)

In order to generate data, we first need to set up a computer experiment. This involves running a number of simulations at various input configurations (Jerome Sacks, William J. Welch, Toby J. Mitchell, & Henry P. Wynn, 1989a). A typical question that arises before running a computer experiment is that of how to determine these input configurations. It is well established that the sampling strategy of a design space has a direct on the quality of the metamodel (K.-T. Fang, Li, & Sudjianto, 2005). In other words, how do we determine which points in a parameter space to evaluate?

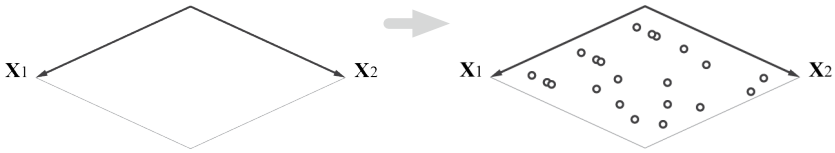


Figure 2.3: Samples in parameter space

The configurations of input values constitute an "experimental design". The region corresponding to the values of the inputs over which we wish to study or model the response is the experimental region. In this context, a point in the experimental region corresponds to a vector composed of simulation input values. Therefore, an experimental design is a configuration of runs, for which we would like to compute a simulation response.

Here, we discuss two typical sampling approaches used for formulating metamodels; *classic factorial designs*, and *space-filling designs*.

2.2.1.1 Factorial designs

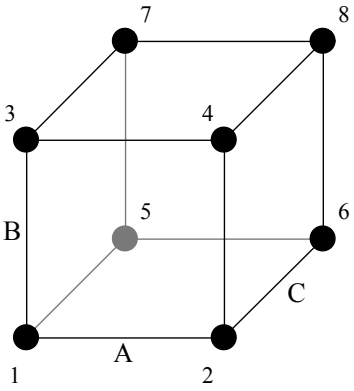
Factorial designs stem from the theory of Design of Experiments (DOE). DOE was originally developed by Fisher (1935) for the strategic planning of agricultural land in the 1930s. Since then, classic DOE methods such as factorial designs have been used extensively for carefully planning physical experiments with the scope of revealing insight with a minimum number of experiment runs. More recently, classic DOE methods were extended for planning of computer simulation experiments.

In general factorial designs involve selecting a combination of variables, referred to as *factors* and for each, a discrete number of values, referred to as *levels*. These are represented in a *design*

matrix as a sequence of experiments to be performed, whose rows and columns denote experiment runs and factor levels, respectively. These levels correspond to values that can be inputted into each factor. The strategic sequence of levels will reveal useful information about the relationship between the variables and observed physical behaviour when comparing and contrasting the results (Saltelli, Chan, & Scott, 2000). Furthermore, the information can be used to construct a polynomial regression metamodel.

Main types of factorial designs used for computer experiments include *full factorial deigns* and *fractional factorial designs.* In further detail:

(i) A full factorial design (FFD), requires m^k simulation runs, where each of the k factors are investigated at m levels. The most common designs are the 2^k (for evaluating main effects and interactions) and 3^k (for evaluating main effects, quadratic effects and interactions) for k factors at 2 and 3 levels respectively. They are common because they only need 2 levels for each factor. These can be low and high, often -1 and +1 (or – and +). Conceptually, a 2^k factorial designs sample at the corners of a hypercube defined by the factors' low and high settings (Table 1). A hypercube is a geometric description of the design space.



Run	Levels			Results
	Factor A	Factor B	Factor C	y
1	-	-	-	y_1
2	+	-	-	y_2
3	-	+	-	y_3
4	+	+	-	y_4
5	-	-	+	y_5
6	+	-	+	y_6
7	-	+	+	y_7
8	+	+	+	y_8

Figure 2.4: Representation of sampled hypercube for a 2^3 design (left).

Table 1: Sequence of '+' and '-' values represented in a design matrix for a 2^3 design (right).

The size of full FDs increases exponentially with k which may lead to an inefficient number of simulation runs when computational time for one run is costly. For example, 5 factors with 2 levels each (2^5) requires 32 runs while 10 factors with 2 levels each require 1024 runs. In this case, fractional factorial designs are adopted.

(ii) Fractional factorial designs (FFD) are suitable when experiments are costly and the number of design points are large (Box, Hunter, & Hunter, 1978). A fractional FD consists of a fraction of a full FD. Some high order interactions (interactions between a large group of parameters) generally have insignificant influence on the response (Saltelli et al., 2000; Sanchez, 2005). With this assumption in mind, the number of required runs for a factorial experiment is reduced dramatically. This can be illustrated using the 2^3experiment shown above (Table 1), but the same holds for other designs. For the sake of the example, let it be assumed that there are zero interactions between factors in the 2^3 design in Table 1, then we could introduce a new factor D in a new column and investigate the four factors in $2^3 = 8$ runs rather than four factors in 16 runs.

Graphically speaking, fractional FDs sample at a carefully-chosen fraction of the corner points on the hypercube. Figure 2.5 is a representation of the sampling for a 2^{3-1}design, where three factors are observed, each at two levels, in only $2^{3-1} = 4$ runs (Sanchez, 2005). For a more detailed review of factorial designs, see Raymond H Myers, Montgomery, and Anderson-Cook (1995).

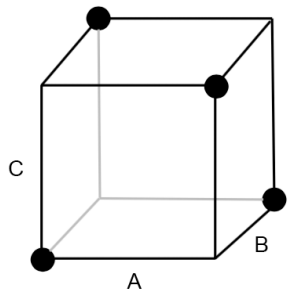


Figure 2.5: Geometrical representation of a 2^{3-1}factorial.

However, when dealing with multiple parameters, it is difficult to make assumptions about the functional form relating the inputs and outputs of the numerical model underlying the simulation because interesting characteristics of the function such as maxima and minima are likely to be anywhere in the design space. In this context, factorial designs become less preferred for generating input samples, because considering only a few levels within the ranges of each parameter, is not enough to capture such characteristics. Furthermore, the number of simulation runs increases exponentially with the number of parameters.

In this context, the need for more efficient designs when considering many factors gave rise to a group of techniques called *space-filling designs*.

2.2.1.2 Space-filling deigns

Space-filling designs focus on achieving a more uniform coverage of the entire parameter space instead of relying only on the extremes of the parameter space, to cover all regions with similar probability.

In most literature on computer experiments, the term 'space-filling' is intended as a synonym for 'evenly spread' (Pronzato & Müller, 2012). However, in a more technical sense, it is also an algorithm for generating samples for any number of variables n, such that as n increases, the algorithm produces samples that are increasingly dense. On the other hand, we want to avoid a distribution that is too evenly spread but more pseudo-randomly spread, to avoid systematic correlations between the parameters. The technical term for 'goodness of a spread' is a *discrepancy* and this varies with the type of space-filling method used.

Here, we discuss two main types of space-filling designs used in metamodeling: *Latin hypercube sampling (LHS)*, and *Sobol sequences* methods.

(i) *Latin hypercube sampling (LHS)* (McKay, Beckman, & Conover, 1979) was developed as an improved version of pseudo-random sampling. This method ensures that each of the input variables has all portions of its range represented by dividing the range of each interval [0,1] into n subintervals of equal length. In other words, the design space is divided into a grid and values within each of the intervals are then randomly selected. LHS is computationally cheap to generate and can cope with many input variables however, since they are generated randomly, they are not the most robust.

(ii) *Sobol' sequences* (Sobol', 1990) belong to the family of quasi-random sequences, which are designed to generate values as evenly spread as possible, even when the problem dimensions are high (n>2). When compared with LHS, the main difference is that the values are chosen under the consideration of the previously sampled points and thus avoid the occurrence of clusters and gaps. Sampling with Sobol' sequences are said to be more efficient because they generate samples with low discrepancy and therefore less density is required to 'fill' the design space evenly (Krykova, 2003).

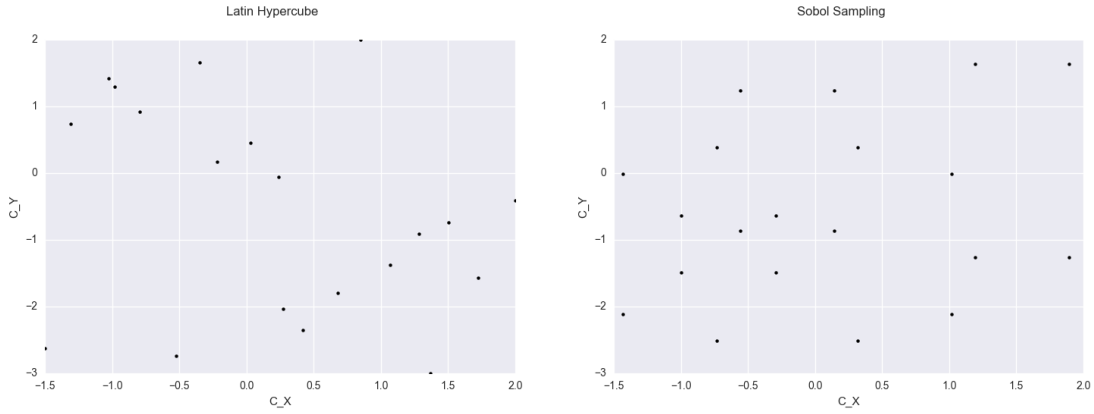


Figure 2.6: Comparative 2D scatter plots of pseudo-random (left) and quasi-random sampling (right) for a sample size of 20.

Figure 2.6 illustrates scatter plots of two of the six dimensions in the design problem under study in this experiment (variables C_X and C_Y). The plots suggest that the samples (points) generated by Sobol' sequences are more evenly distributed than those generated by LHS. For the sake of this example, samples of only 20 points were generated as it becomes difficult to identify the difference in discrepancies at higher densities such as those studied in this experiment. Python libraries 'pyDOE' (Lee, 2014) and 'SALib' (Herman, 2014) were used to generate the LHS and Sobol' samples in Figure 2.6, respectively.

While sampling with Sobol' might seem like the preferred choice, the higher discrepancy in LHS can be beneficial because it induces more randomness in the sample. Randomness in an input sample is important as it reduces the chance of correlation between input variables prior to the experiment. Theoretically, and as in physical experiments, variables under study should always be independent.

Unlike with factorial methods, there is no standard equation to dictate the number of points required in space-filling methods, i.e., the number of simulation runs. Some literature suggest rules of thumb based on the number of parameters considered. For example, Chapman, Welch, Bowman, Sacks, and Walsh (1994) and Jones, Schonlau, and Welch (1998) suggest using a sample size of 10d, where d is the number of parameters. While a rule of thumb might provide some form of guidance, the number of points required to secure a robust metamodel is very much dependant on characteristics of the problem such as nonlinearity, which are typically unknown beforehand, especially when dealing with numerous parameters.

2.2.2 Run simulations to generate data (step 2)

Once the input samples are generated, they are plugged into the simulation to generate the simulation response data, required to build the statistical model. Each sample point in the list of samples can be interpreted as a vector of scalar input values. Each vector is inputted into the numerical simulation to generate a response. The batch simulation process is generally automated through a script.

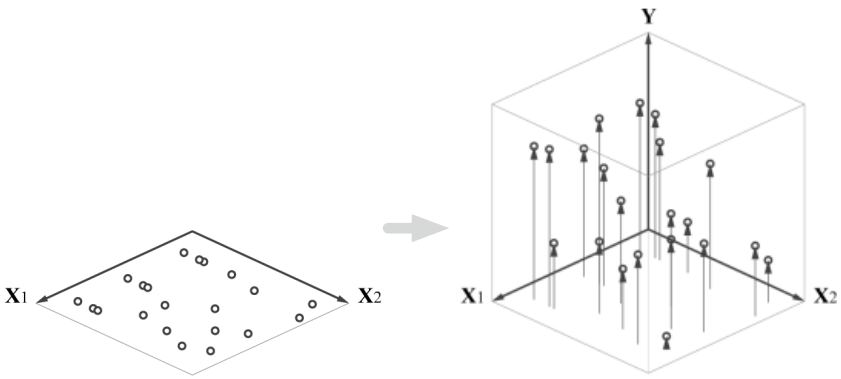


Figure 2.7: Evaluation of points in parameter space (running simulations).

2.2.3 Build the statistical model (step 3)

This step is where the actual statistical model is built.

Once the appropriate experimental design is selected and the necessary simulation runs are complete, the response data is aggregated together with the input samples to form an input-output dataset. With this dataset in hand, the next step is to build the actual metamodel. In general metamodeling is a task of regression. In non-statistics terminology, regression is a process of formulating a mathematical expression that best describes the true relationship between inputs and outputs that produced the data points (function type) (Figure 2.8).

In more detail, a regression task involves (1) selecting an appropriate statistical approximation model that best describes the global characteristics of the data points, and subsequently (2) selecting a fitting method to fine-tune the selected approximation model such that it fits tightly to the local characteristics of the data points.

Typical regression models used for metamodeling include polynomial regression models (J. P. Kleijnen, 2008), spline models (R. R. Barton, 1998), neural networks (Fonseca et al., 2003), Kriging models (Ankenman et al., 2010), response surfaces (J. P. Kleijnen & Sargent, 2000), and game-theoretic models (Poropudas & Virtanen, 2010a, 2010b). In this section, we focus on the

most widely implemented approximation models and fitting methods: *response surfaces*, *Kriging*, and *neural networks*.

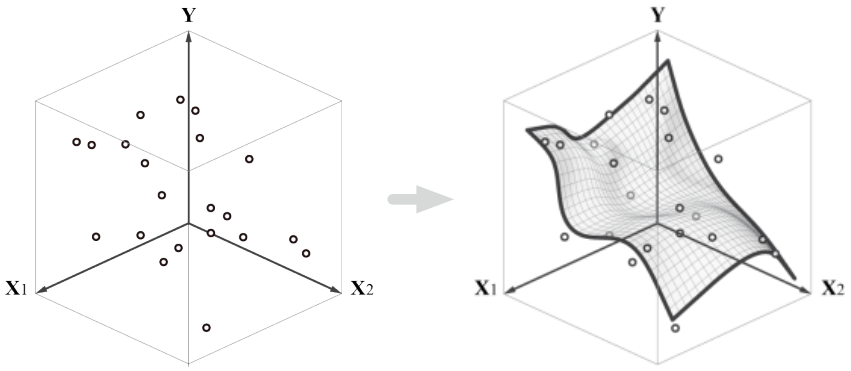


Figure 2.8: Fitting of statistical model.

In general, the relationship between a response y and a vector of independent factors x that influence y, is given by Eqn. (3).

$$y = f(x) + \varepsilon \tag{3}$$

where, ε denotes the random error, as a means to take into account the inaccuracy of simulation model f, in representing the real-world phenomenon. This error is typically assumed to have a normal distribution with mean zero.

2.2.3.1 Response surface metamodels

Response surfaces are the most typical and straightforward approach to metamodeling, where an approximate response surface $\hat{f}$ is created to approximate the true (yet, unknown) response surface f. A response surface can be easily thought of as a landscape with hills, where the x, y coordinates represent the input variables and the height of the hills represents the model's response.

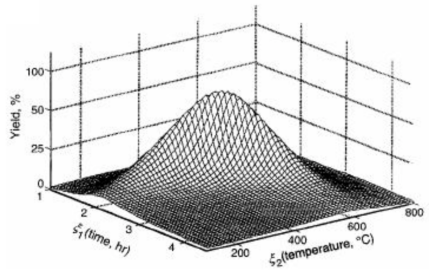


Figure 2.9: A theoretical response surface for a chemical engineering problem (Raymond H. Myers, Anderson-Cook, & Montgomery, 2014).

The most used response surface approximation functions are *low-order polynomial functions*, whose mathematical expression is generally described as a linear combination of basis functions from a polynomial parametric family (other 'families' include sine functions, piecewise polynomials, etc.). Polynomial basis functions are separate polynomial functions, adapted from traditional polynomial regression in statistics, that are combined together to form the polynomial model. The number of basis functions increases rapidly with the number of input factors and the degree of the polynomial.

In this context, a first-order polynomial is typically used when considering few factors and are in fact the most common choice for metamodeling of simulation models (K.-T. Fang et al., 2005). First-order polynomials take only into account the independent influence of the input factors on the response, as follows:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k + \varepsilon \tag{4}$$

$$\hat{y} = \beta_0 + \sum_{i=1}^{k} \beta_{\mathrm{i}} x_{\mathrm{i}} + \varepsilon \tag{5}$$

where, $\hat{y}$ denotes the predicted response, $x_1, \dots, x_{\mathrm{k}}$ denote the input factors, and $\beta_0, \dots, \beta_{\mathrm{k}}$ denote the polynomial coefficients, which are required to fine-tune the polynomial function to fit the data more tightly. In statistics, $\beta_0, \dots, \beta_{\mathrm{k}}$ are referred to as 'parameters' (not to be confused with simulation input parameters). We can think of parameters as the local slope/gradient at that point. When normalised, the parameter values can be used as an indicator of the significance of the term it is assigned to, in influencing the response $\hat{y}$. In other words, the value of a parameters serves as a sensitivity indicator. In general, parameters are estimated from the simulation data generated in step 2. In the case of polynomial functions, parameters are estimated using *least square regression.*

Therefore, once the model approximation model is selected, and the coefficient values of the parameters are estimated then, the response surface metamodel can be used to predict the response $\hat{y}$.

When numerous factors are considered, and significant curvature appears to exist (due to nonlinear relationships) when plotting the data, it is recommended to introduce second-order terms such as ${x_{\mathrm{i}}}^2$ in the polynomial model, and interaction terms such as $x_{\mathrm{i}} x_{\mathrm{j}}$, that account for

the interaction between two factors. More specifically, a second order polynomial regression takes the following form:

$$\hat{y} = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \beta_{ii} {x_i}^2 + \sum_{i}^{k} \sum_{j}^{k} \beta_{ij} x_i x_j + \varepsilon \quad (6)$$

Higher-order polynomials can also be included, however it is not advised in metamodeling community due to instabilities (R. R. Barton, 1992). Furthermore, higher orders require significantly more simulation data to estimate all the coefficients in the polynomial equation, particularly in large dimensions (Jin, Chen, & Simpson, 2001).

2.2.3.2 Kriging metamodels

Kriging (Krige, 1951) is another type of regression technique used for metamodeling of simulation models (Booker et al., 1999; Sacks et al., 1989a; Stein, 1987; Van Beers & Kleijnen, 2004). Typically, Kriging models are fitted to data that are obtained for larger global design spaces, rather than small local regions of a problem space, such as those used in low-order polynomial regression metamodels, in Eqn. (4).

Kriging works by interpolating a response surface exactly through the generated data points in the input-output dataset (Figure 2.10). In fact, Kriging is considered an 'exact' interpolator, which means that the function prediction for input values already observed in the dataset, will be exactly equal to the simulated output values. Subsequently, response values for new input values (not in the dataset), are estimated based on the value and proximity to known points on the response surface.

In more detail, a typical Kriging metamodel is composed of a combination of two models: a so called 'trend' function $P(x)$ and a function $Z(x)$ that models the local 'departures' from the trend function (Martin & Simpson, 2005; Jerome Sacks, William J Welch, Toby J Mitchell, & Henry P Wynn, 1989b; Sasena, 2002; Simpson, Mauery, Korte, & Mistree, 2001). When combined linearly, these models are used to predict a response $\hat{y}$, as follows:

$$\hat{y} = P(x) + Z(x) \quad (7)$$

The trend function $P(x)$ in Eqn. (7) is defined as a typical regression function as follows:

$$P(x) = \sum_{i=1}^{k} \beta_i f_i(x) \tag{8}$$

where, β are regression parameters (coefficients), and $f_i(x)$ is a function whose basis is typically polynomial (constant, linear, quadratic, etc.), that approximates the overall (global) trend of the data points, as illustrated in Figure 2.10. On the other hand, the local departures model $Z(x)$ is a stochastic (probabilistic) component that generates deviations from the trend model, such that the Kriging metamodel interpolates through the sampled response data as illustrated in Figure 2.10. The local departures model $Z(x)$ typically makes use of radial basis functions (RBF).

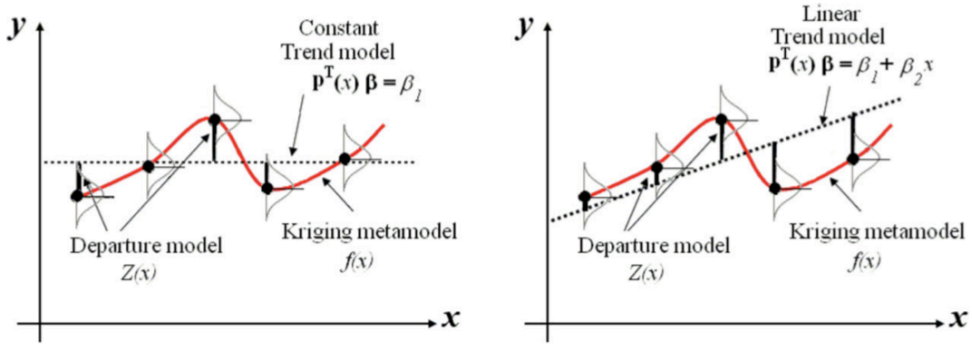


Figure 2.10: Kriging is a combination of two models (Acar, 2013).

Typically, Kriging models are fitted to data from large experimental areas as opposed to smaller areas in low-order polynomial regression; i.e. Kriging models are global rather than local. These models are also common for prediction and ultimately sensitivity analysis and optimisation (J. P. Kleijnen, 2009). In literature Kriging is described as an accurate metamodeling technique as it will always interpolate exactly with the known design points (Biles, Kleijnen, Van Beers, & Van Nieuwenhuyse, 2007; Simpson, Mauery, et al., 2001; Van Beers & Kleijnen, 2004; Zakerifar, Biles, & Evans, 2009).

2.2.3.3 Neural network metamodels

A neural network (also referred to as an artificial neural network), is a type of model form machine learning (a subset of statistics) that can be used to predict multiple target outputs from a set of inputs. A neural network is as an assembly of so called 'neurons', connected by edges. A single neuron represents a regression model, referred to as a 'transfer function' while, edges represent weight values (Figure 2.11). In this framework, each neuron computes a sum η, by summing the weighted inputs (see Eqn. (9)) and subsequently applying η to a transfer function to compute a response $\hat{y}$. Typical transfer functions include step-functions or sigmoid functions

such as in Eqn. (10). The latter are the most typical as they capture nonlinear relationships very well.

$$\eta = \sum_{i=0}^{k} w_i x_i + \beta \tag{9}$$

$$\hat{y} = \frac{1}{1 + e^{-\eta}} \tag{10}$$

Subsequently, a neural network is created by assembling the neurons into an architecture. The typical architecture is called a forward feed architecture as illustrated in Figure 2.11, right. Neurons in the middle layer between inputs and outputs, also referred to as the 'hidden' layer, capture the intrinsic characteristics of the relationships that map the inputs to the target outputs. Increasing the number of hidden layers increases the chance of capturing deeper characteristics of the input-output mapping and thus, improves the generalizability of the metamodel. On the other hand, 'too many' layers may lead to issues such as 'model over-fitting'.

Building a neural network metamodel involves two steps: 1) indicating the architecture of the network (number of layers, number of inputs, number of outputs, etc.), and 2) training the neural network to 'learn' the characteristics of the mapping from simulation data thus, learn how to predict target responses for new input values. In comparison with response surfaces, these two steps are equivalent to 1) selecting an approximation model, and 2) estimating the coefficients (parameters) from the generated simulation data. When assembled, a neural network can be thought of as a matrix of multiple regression models. This arrangement enables multi-input, multi-output metamodels, which can be powerful in a building design context when taking multiple engineering considerations into account.

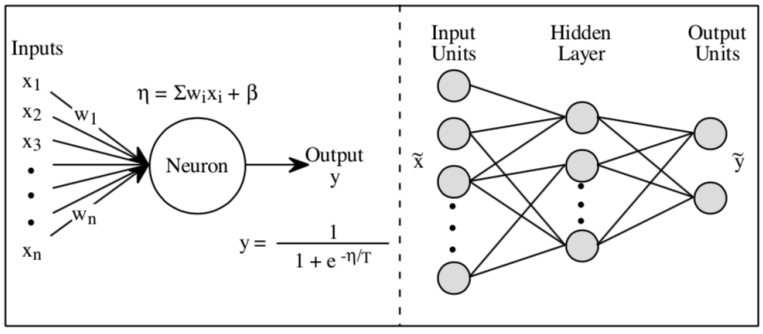


Figure 2.11: Anatomy of a single neuron (left), and a neural network architecture (right).

The advantage of neural networks as metamodels is that they are not limited to the number of degrees of orders. In other words, they can accommodation flexible types of relationships,

varying from linear to very nonlinear relationships typical with high dimensional problems (Grossberg, 1988). In fact, they are considered powerful enough to build 'universal approximators', if their architecture is large enough (Funahashi, 1989; Hornik, Stinchcombe, & White, 1989). Neural networks are also considered a global metamodeling method.

2.2.3.1 Other methods

Other types of models used for metamodels include Radial Basis Functions (RBF) (Costa & Nannicini, 2018; Dyn, Levin, & Rippa, 1986; H. Fang & Horstemeyer, 2006), Least Interpolating Polynomials (De Boor & Ron, 1990), and Multivariate Adaptive Regression Splines (Friedman, 1991). Furthermore, Varadarajan, CHEN*, and Pelka (2000) present an approach that combines polynomial functions and artificial neural networks. A number of publications in the metamodeling community provide abundant insight by comparing and contrasting these methods however, there is no consensus as to which metamodeling approach is superior (Chen, Tsui, Barton, & Meckesheimer, 2006; Giunta & Watson, 1998; Jin et al., 2001; Koch, Simpson, Allen, & Mistree, 1999; Simpson, Mauery, et al., 2001; Simpson, Poplinski, Koch, & Allen, 2001). Viana, Simpson, Balabanov, and Toropov (2014) argue that these studies affirm the suspicion that the quality of a metamodel depends on both the nature of the problem and the design of experiments selected for generating the input samples combinations.

Despite the multitude of methods, J. P. Kleijnen (2009) claims that low-order polynomial regression models remain the most popular type of metamodel, across different fields. This is likely the case because the technique is the easiest to use, provided that the user is aware of the pitfalls outlines in Simpson, Poplinski, et al. (2001).

2.2.4 Validating the statistical model (step 4)

Once a metamodel is built, it is necessary to validate the quality of the statistical model formulated in step 3, to secure that the formulated input-output mapping captures the true input-output relationships underlying the simulation model, as well as possible. In other words, it is necessary to test the generalizability of the metamodel for predicting new input values; i.e. input values that were not included in the dataset used to build the metamodel in the first place.

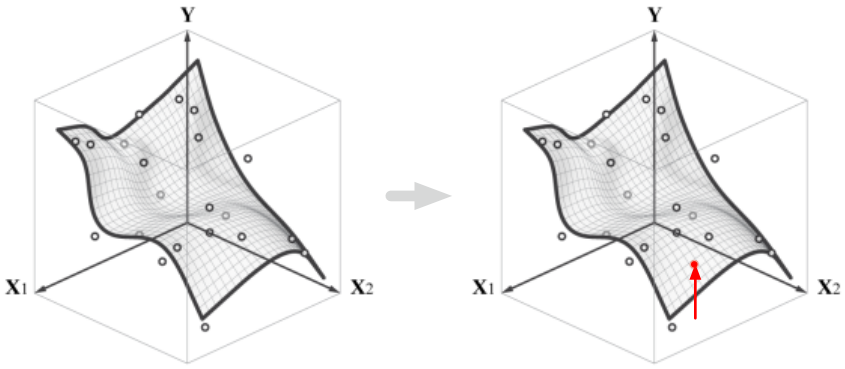


Figure 2.12: Predict for new input values, to validate the prediction accuracy of the metamodel.

The most typical approach to metamodel validation is through a method called cross-validation, where the input-output dataset generated in step 2, is split into a so-called 'training set' and a 'testing-set', before building the statistical model in step 3. In other words, the statistical model is built using only the 'training-set' from the complete input-output dataset. The percentage split between training and testing data can vary, depending on the quality of the dataset.

Once the metamodel is built on the training set (step 3), the metamodel is used to predict response values for the input values in the testing-set. Subsequently, the predicted values are compared with the actual simulation response values in the testing-set, as a means to quantify how well can the metamodel predict a response when compared the original simulation response. The prediction accuracy of the metamodel can be estimated statistically, by means of the Root Mean Square Error (RMSE). The RMSE is a widely accepted method to quantify the robustness of metamodels in general. This is defined as:

$$\text{RMSE} = \sqrt{\frac{\sum(\hat{y}_t - y_t)^2}{n}} \tag{11}$$

where, $\hat{y}_t$ is the predicted value, y_t is the actual value generated from simulation and n is the number of data points in the testing dataset. The RMSE is a value given in original units however in order to compare the accuracy between different metamodels, it is necessary to normalise the RMSE value hence, normalised room mean square error (NRMSE).

A comprehensive review of the validation of metamodels is available in J. P. Kleijnen and Sargent (2000).

2.3 Metamodel applications

In literature, most metamodel applications focus mostly around aerospace, automotive and civil engineering applications. Subsequently a large body of technical publications was produced over the last few decades thus, paving the way for metamodel applications in other fields, including building-related simulation.

2.3.1 Early applications

Early examples include the metamodeling of a water system simulation model (Hufschmidt & Fiering, 1966), and the metamodeling for faster simulation of an aircraft jet engine inlet design and subsequently optimisation of the aerofoil and trapezoidal ducts (John C, Marius, Serhat, & Anthony T, 1995). Before the 1990s response surface models and neural networks were amongst the most popular methods, especially for aerospace engineering applications. A review of these methods and applications is given by Barthelemy and Haftka (1993); Sobieszczanski-Sobieski and Haftka (1997). Subsequently, interest in simulation approximation grew significantly during the mid 1990s, with emphasis placed on response surface models. The surge of interest resulted from research in relation to wind analysis for High Speed Civil Transport. Overall, simulation approximation models enabled faster computation of a response and thus, opened up a number of attractive abilities which were previously cumbersome to compute such as higher-order sensitivity analysis, and optimisation searches of larger design spaces.

2.3.2 Building-design related applications

In much more recent times, the popularity of simulation metamodels was noticed by the building-design related disciplines such as energy performance simulation, structural engineering, etc. The predominant focus of these applications lies with faster design space exploration and optimisation.

For instance, Capozzoli, Mechri, and Corrado (2009) utilise regression analysis to perform a simple polynomial approximation for heating energy needs and for cooling energy needs, which can then be used by architects as a faster and simpler replacement of more complex equations involved in energy calculations. Hygh, DeCarolis, Hill, and Ranjithan (2012) also present a regression model to act as a decision support tool instead of computationally demanding energy simulation models during the early stages of design. In addition, they present regression coefficients to quantify the sensitivity of heating, cooling and total energy loads to building design parameters. Hygh et al. (2012) claim that these standardized regression coefficients can

be used directly by designers to identify the building design parameters that drive energy performance, for early stages of design. Similarly, Ritter, Schubert, Geyer, Borrmann, and Petzold (2014) present an adaption of response surface methodology, which they claim makes it possible to generate metamodels automatically without having to deal with the complex mathematical structure; thus, making it easier for designers to make use of energy calculation during early design stages. Besides rapid feedback in early design stages, approximate models are also beneficial for quicker simulation-based optimisation such as: energy optimisation for urban buildings (Panão, Gonçalves, & Ferrão, 2008), optimization for redundant building cooling heating and power system (Jiangjiang Wang, Zhai, Jing, & Zhang, 2010), multi-objective optimisation for solar low energy buildings (Peippo, Lund, & Vartiainen, 1999). Costa, Nannicini, Schroepfer, and Wortmann (2015) present an RBF based metamodel as a means to achieve faster optimisation cycles of Useful Daylight Illuminance (UDI) for a façade design.

Metamodels have also been used to substitute Finite Element based simulation such as Computational Fluid Dynamics (CFD), which is known to be very computationally demanding: Klemm, Marks, and Klemm (2000) present a metamodel-based optimisation method to derive objective functions for optimisation by applying a polynomial regression methods on CFD simulation results. Tresidder, Zhang, and Forrester (2012) use Kriging metamodels to optimise CO2 emissions and construction costs of buildings. Similarly Gengembre, Ladevie, Fudym, and Thuillier (2012) minimise 20-year life cycle cost of a building model also using a Kriging metamodel. They discuss that the accuracy of the metamodel is acceptable and that such approximate models can help designers explore the design space with cheaper simulation.

2.4 Discussion

To reiterate, the research goal set out in Chapter 1 aims to overcome the limitations of typical design-analysis systems discussed earlier. Instead of utilising engineering simulation as a black-box to generate a response for singular scenarios or to search a design space for the best performing scenario, we aim to capture the simulation input-output relationships and represent them such that they can be reasoned with, and such that the flexibility of the input parameters is maintained.

In this section 2.3 we introduced the notion of metamodels, as an approach to encapsulate the input-output relationships through statistical approximation models. The above literature demonstrates clearly the popularity of metamodels, as good approximators of numerical simulation models, for faster computation of approximated response $\hat{y}$. In most applications, metamodels are used as a means to an end; the final goals may be validation and verification of

the simulation model, sensitivity or what-if analysis of that model, and optimization (J. P. Kleijnen, 1998; J. P. Kleijnen & Sargent, 2000; Law, 2015).
However, in this dissertation we argue, that the representation of typical metamodels such as response surface polynomial models, Kriging models and neural networks, limits their capabilities for guiding early stage design, when introduced to the environment of a computational design system. More specifically, we argue that:

1) the explicit representation of relationships in functional models still limits inputs and outputs to scalar values thus, singular scenarios, and

2) typical metamodels tend to become a black box when dealing with multiple input parameters in the design-analysis system. It becomes challenging to consolidate a global understanding of the cause-effect relationships for decision making because of the algebraic-intricacy of the function.

For example, let us consider the response surface metamodel with a first-order polynomial regression model described in Eqn. (4). Here, the relationships between the input x_i and the output y is defined by a parameter β (model parameter). In other words, the relationship between x and y is defined *exactly* by a scalar coefficient value, thus, yielding a scalar response. Furthermore, as the number of simulation inputs $x_i \dots x_n$ increase, a first order polynomial might not suffice to capture potential nonlinearities due to multiple interacting variables. The consideration of higher order terms required to capture the input-output relationships such as in Eqn. (6), can render the β values difficult to interpret due to the algebraic fragmentation of multiple terms such $\beta_i x_i$ and $\beta_{ii} {x_i}^2$ to describe the behaviour of one variable, let alone multiple. Overall, the increased number of functional terms pose a challenge to form a global picture of the cause-effect relationships and hence, a challenge to draw a conclusion to guide design decisions. A similar challenge is posed with other types of popular functional metamodels such as Kriging metamodels, whose trend function is typically based on a polynomial function such as in Eqn. (8).

As for neural networks, they may not seem to be function-based however, since the weights on each neuron depend on the choice of a preselected activation function (Figure 2.11), they can be though of as a summation of a series of functions. Neural networks are by nature a black box, as it is very difficult to interpret input-output relationships given the number of neurons in the middle layers and the number of layers themselves. Having said that, a very promising subfield of research in machine learning has emerged, that focus on developing methods to interpret the features which the network has learnt. For example Olah et al. (2018) provide a very illustrative study on visually interpreting what the middle layers of an image-based convolutional neural

network (a type of neural network) actually mean (Figure 2.14). This approach can be extremely powerful to understand the latent variables driving a response phenomenon. However, the state of this research is limited to image-based inputs as of yet.

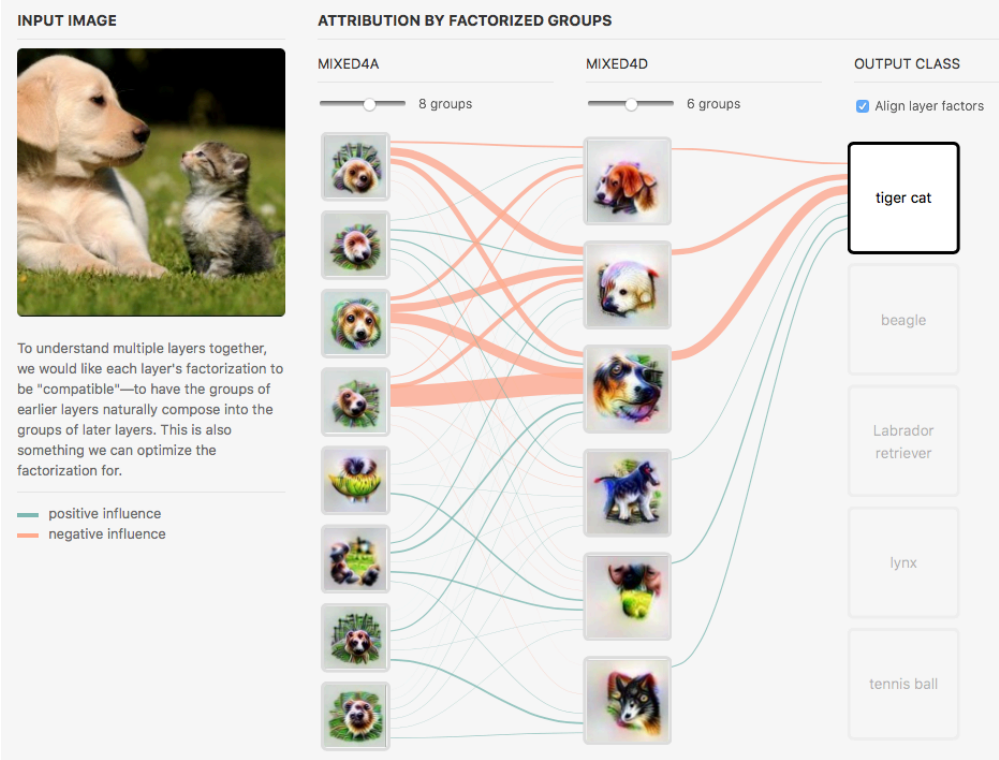


Figure 2.13: Interactive applet for hidden-feature interpretation of a neural network (Olah et al., 2018).

One way of gaining further insight into the input-output relationships using typical metamodels is by taking advantage of their computation speed to generate significantly more data and subsequently adopt further statistical methods that focus on knowledge-discovery. These statistical methods include classic methods such as correlation analysis, methods from design of experiments such as factorial designs, methods from sensitivity analysis, and also advanced methods in descriptive statistics to plot and illustrate complex information. These methods can provide very useful insight, despite the two-step process. However, we argue the cognitive challenge to consolidate the information from these methods due to the curse of dimensionality, remains. The format of the knowledge outputted by these methods is mostly static and hence, not explorable as a consolidated model. In other words, when dealing with multiple variables, it remains very difficult to make intelligent inferences whilst taking into account all fragmented bits of insight such as correlation values, sensitivity values, or simply illustrative plots.

We may conclude that as a result of the above challenges 1) and 2), function-based metamodels in design systems become a black box (Figure 2.14). In other words, the metamodel can be used as a black box that is good for predicting an exact response output when presented with new input values, but are not so accessible for reasoning during the flexible stages of design, and for drawing consolidated inferences when presented with multiple parameters in the system.

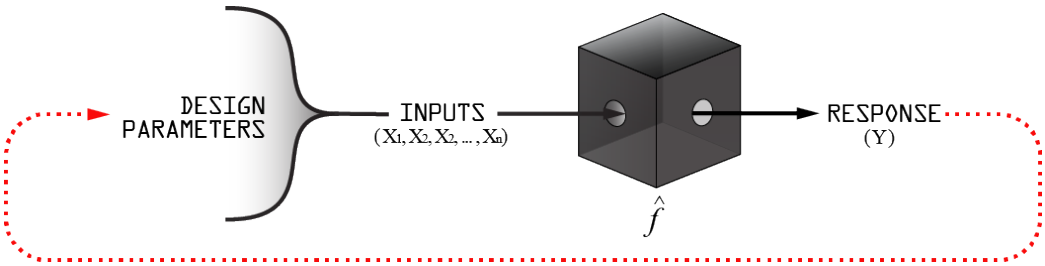


Figure 2.14: Black-box metamodel.

At this point, we can summarise the challenge into the following question: Can we build a metamodel capable of drawing consolidated inferences over multidimensional inputs, while providing a broader representation of the design space?

Our answer to this question lies with building a metamodel whose underlying statistical approach represents relationships in a non-deterministic way. In this context, we take on a probabilistic model approach.

2.4.1 Probabilistic metamodel

In this dissertation we assimilate the challenge of 'decision making over a high resolution representation of the design space', to 'reasoning over uncertainty', which is a well studied task in the field of probability. This analogy sets the core idea behind the metamodel approach presented in this dissertation.

In general, the challenge with reasoning over uncertainty refers to the difficulty of understanding a problem, when the available information about the problem is not fixed, or not certain hence, uncertain. In statistics, uncertainty is addressed through a representation of the problem as a probabilistic model. A probabilistic model does not compress relationships into a functional from, preselected from a family of possible functions. Instead, relationships are represented probabilistically, in a joint probability distribution. Subsequently, a probabilistic model enables reasoning over uncertain information using inference, even in multiple dimensions.

Therefore, in this research we adopt a probabilistic approach to metamodeling to enable reasoning over multiple dimensions using probabilistic inference, and to provide a broader representation of the design space such that all sampled data points are considered, instead of 'fitting' a function between points.

The next chapter will introduce the notion of a probabilistic metamodel in more detail.

Chapter 3

A PROBABILISTIC METAMODEL

In the previous chapter, we argued how typical metamodels may not always be useful for early stages of design because their predominantly deterministic format 1) still limits metamodel inputs and outputs to scalar values and 2) is difficult to infer comprehensive insight about the cause-effect relationships over multiple dimensions. More specifically, the functional representation of typical metamodels compresses relationships between input and output variables into explicit functions thus compressing the resolution of the design space into an inaccessible function, while rendering it challenging to interpret the relationships when dealing with multiple parameters.

Instead, in this thesis we take on a probabilistic metamodeling approach. We approximate the design space as a probabilistic model, where instead of compressing relationships into a functional model, we consider the joint probability distribution between the inputs and the outputs bounding the design space, thus providing a broader representation of the inputs and outputs bounding the design space. Furthermore, a JPD does not distinguish between inputs and outputs and thus, enables bi-directionality between inputs and outputs, while allows for multiple inputs and output metamodels. Subsequently, we introduce Bayesian networks, which are a type of probabilistic model that combine the power of machine learning techniques and classic probability theorems for reasoning over high-dimensional JPDs, efficiently.

With the above in context, this chapter introduces the idea of a bi-directional metamodel for navigating architectural design spaces through knowledge discovery and reasoning about ‘soft’ inputs and outputs. Section 3.1 illustrates this concept by viewing the architectural design space through a probabilistic lens and making reference to basic yet fundamental concepts from probability theory. Section 3.2 introduces probabilistic graphical models, which are type of probabilistic models to handle inference with many variables. Furthermore, section 3.3 introduces Bayesian networks, which are a specific type of PGMs, as a statistical method of choice. Finally, section 3.4 provides a literature review on related applications of Bayesian networks as metamodels in other fields.

3.1 Probabilistic representation of the design space

In typical metamodeling procedures, once simulated data is generated, typically a function is selected and fit to the generated data points in the design space. In this thesis, we take on a probabilistic approach, where instead of compressing input-output relationships into an explicit function, we consider all generated data points as a probabilistic model. More specifically, we introduce the input parameters and simulation response output as random variables bounding a probabilistic approximation of the design space (Figure 3.2). The careful use of the term 'representation' is key here; we adopt a probabilistic 'wrapper' to data achieved from a deterministic source.

3.1.1 Probabilistic inputs and outputs

A random variable is a quantity that can take multiple possible values, and is represented as a probability distribution of possible values. Typically, a random variable is used to describe a quantity whose outcome does not remain fixed. For example, in physical experiments measured outcomes of an experiment may not always repeat themselves due to uncontrollable factors, and hence, the response is represented as a probability distribution of possible outcomes.

In this research we adopt the notion of random variables, not to represent uncertainty of some experimental outcome, but to represent and maintain flexibility in the input parameters of a design system. More specifically, in the metamodel context, we consider inputs and outputs as random variables (Figure 3.1) where, inputs are represented as a frequency distribution of sampled input values from the design space, while outputs are represented as a frequency distribution of the generated simulation responses. A frequency distribution can be illustrated as a classic histogram.

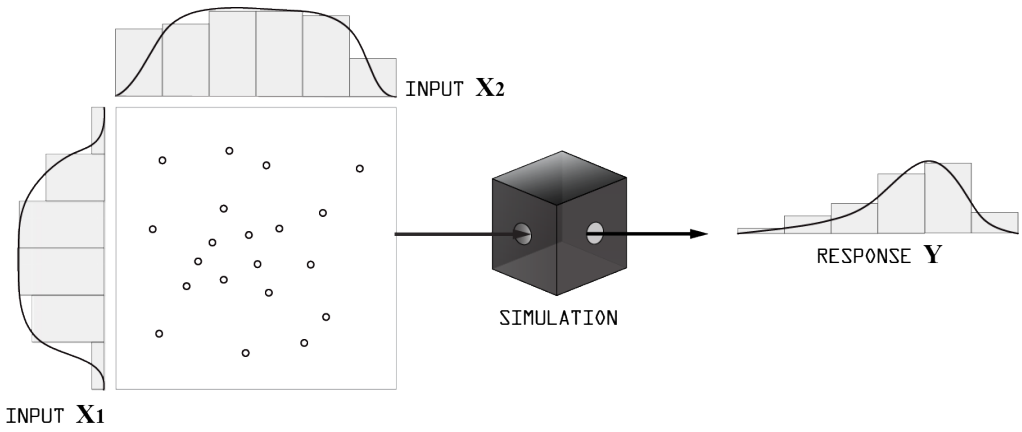


Figure 3.1: Input and output derived as frequency distributions of samples and response data.

When assembled, the three random variables X1, X2, and Y in Figure 3.1, bound a design space illustrated in Figure 3.2. A design space bound by random variables differs from the typical design space representation bound by scalar variables, in the way that the input-output relationships are represented.

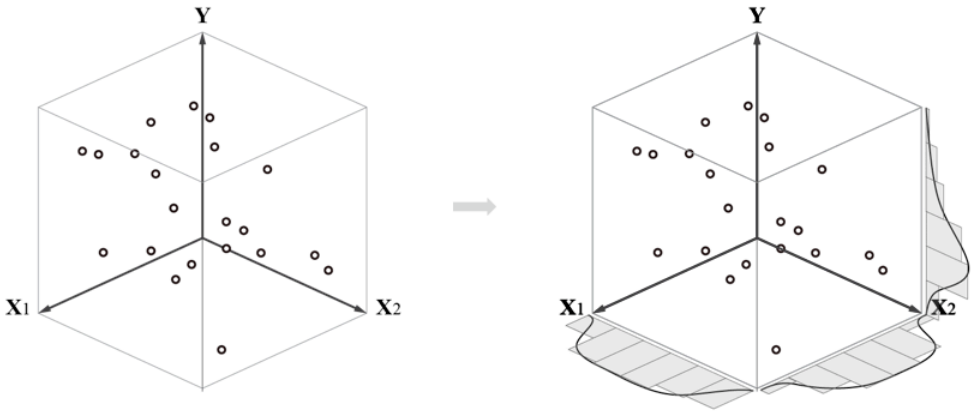


Figure 3.2: Design space bound by random variables (right) instead of scalar variables (left).

Relationships between all random variables, i.e. all probability distributions, are encompassed in the *joint* probability distribution of X1, X2 and Y, which can be written as $P(X1, X2, Y)$. In this notation, the "," stands for the English word "and", meaning that $P(X1, X2, Y)$ can be read as the probability of X1, X2 and Y, values occurring simultaneously. In this context, a JPD can be defined as a multi-dimensional probability distribution that describes the probabilities of combinations of X1, X2 and Y values, referred to as states (Binder et al., 1997). A state is typically a discrete instance of a random variable. A joint probability distribution is in fact typically specified as a table of combinations of X1, X2 and Y states. Each row contains a different combination of states and is then assigned with a probability, as illustrated in Table 2. The total number of entries is equivalent to k^N parameters (N variables, k states).

X1	X2	Y	*P*(X1, X2, Y)
0	0	0	0.018
1	0	0	0.002
0	1	0	0.072
1	1	0	0.008
0	0	1	0.252
1	0	1	0.378
0	1	1	0.108
1	1	1	0.62

Table 2: A hypothetical joint probability table with 3 parameters and 2 states each, implying 8 possible entries.

Therefore, instead of *fitting* an exact function to a set of data points, in a probabilistic metamodel approach we do not assume a form but, consider all the generated data points in a JPD (Figure 3.3). In other words, we choose to approximate the design space as a JPD because it allows us to encompass relationships between inputs and outputs as probabilistic relationships thus, maintain the intended flexibility of the original computational design system.

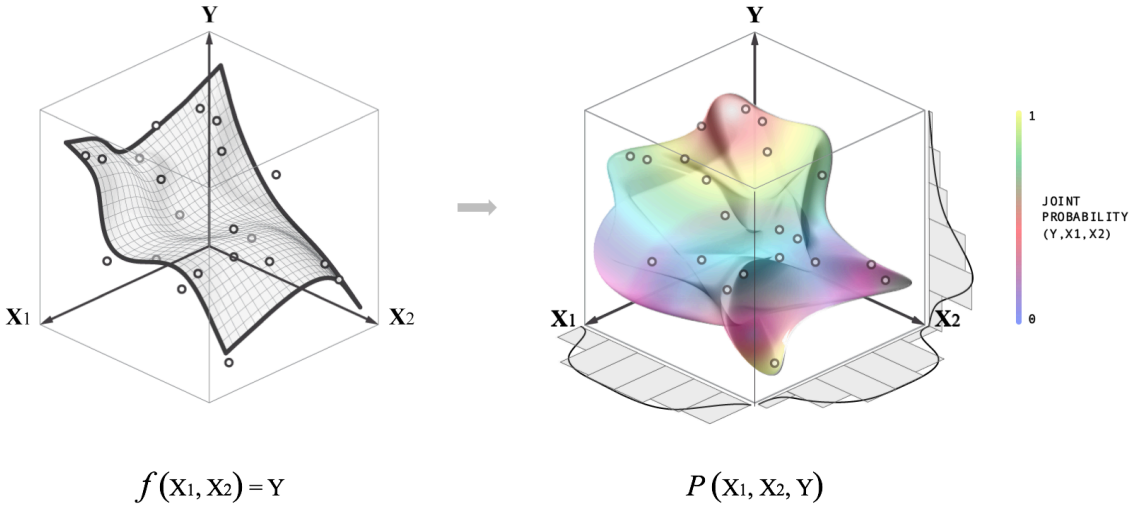


Figure 3.3: Shift from fitting a function to a set of points, to considering all points in a JPD.

3.1.2 Shifting from a forward to a bi-directional metamodel

Traditionally, once a metamodel is built, it can be used to predict response Y, efficiently and gain insight into the input-output relationships. Let us visualise the act of using the metamodel to predict Y, as an act of 'slicing' the design space at specific input values of interest. In typical metamodels such as those discussed in Chapter 2, slicing the exact function at specific input values of X1 and X2, would result in a scalar singular point; in other words, an exact scalar response Y. The exact response tells us nothing about how it was generated; in other words, it does not give any insight about the causality of Y, in terms of X1, X2. This challenge is emphasised further when the number of inputs considered are numerous.

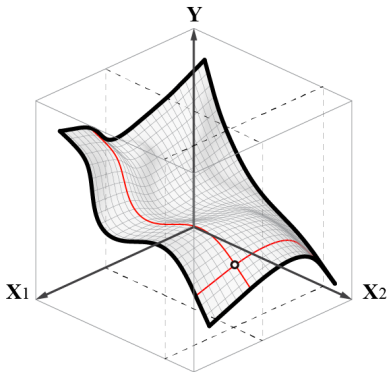


Figure 3.4: The outcome of slicing a typical metamodel is a scalar point.

On the other hand, slicing a probabilistic metamodel implies slicing a JPD. In this context, slicing the JPD at values of X1 and X2 reveals a probability distribution of Y, instead an exact value of Y, since we do not fit an exact 'surface' to the points. Note that, in the three dimensional context illustrated in Figure 3.5, the resulting Y distribution is an intersection between respective slices at X1 and X2. Therefore, we can think of the JPD as a cloud of combinations of inputs and output values; slicing the cloud reveals a probability distribution of input and output occurrences.

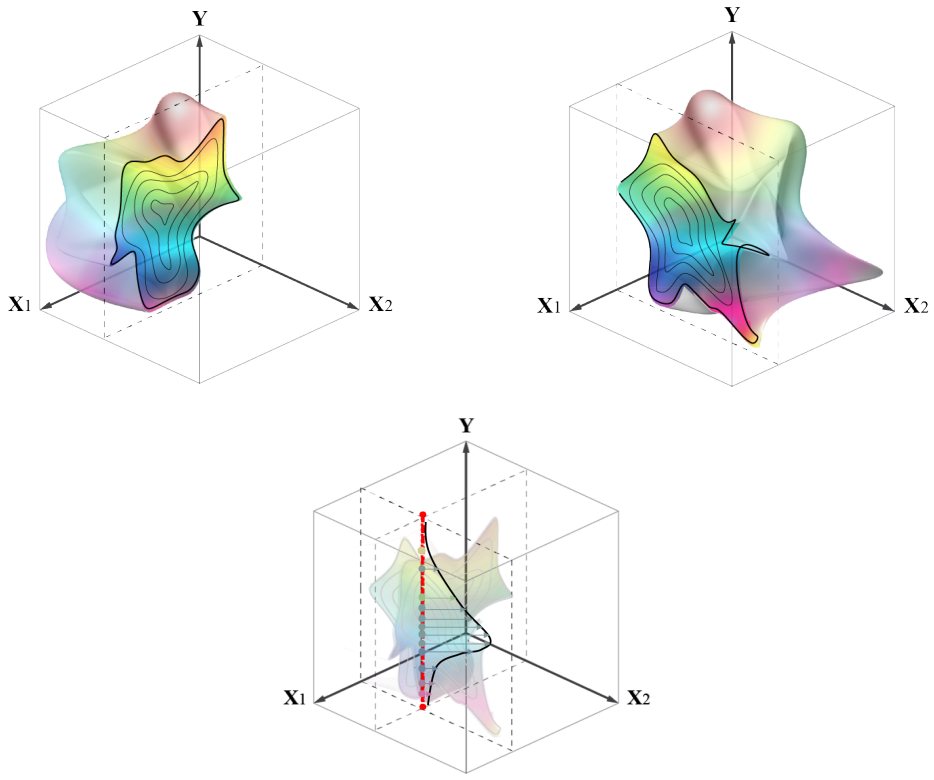


Figure 3.5: 'Slicing' the joint probability distribution to predict Y; X1=x, X2=x → Y

Now, the formulation of a JPD does not distinguish between the 'right-hand side' and 'left-hand side' of the model, unlike a typical functional model. This can be immediately noticed by glancing at the algebraic difference between functional *f* and probabilistic *P* expressions in Figure 3.3. This mathematical indifference implies that we may slice a JPD in *any* direction. For example, we can slice the JPD at an output at a value y of interest, and immediately reveal the likely joint distribution of X1, and X2 to cause Y. This suggests a bi-directionality between inputs and outputs.

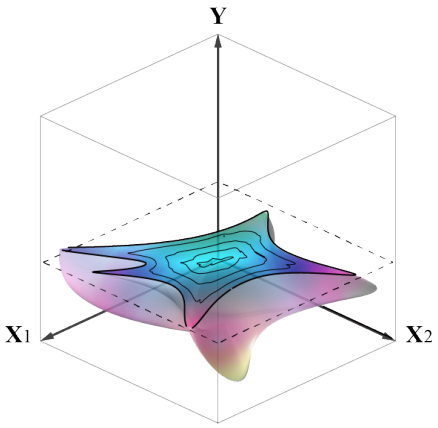


Figure 3.6: 'Slicing' the joint probability distribution to predict X1 and X2; X1, X2 ←Y=y

Therefore, in our metamodel application, we take advantage of the mathematical indifference between inputs and outputs to achieve bi-directionality in the metamodel. The indifference between inputs and outputs means that we can use the probabilistic model to compute inputs→outputs and also outputs→inputs. A bi-directional metamodel can provide insight into the cause-effect relationships between the input parameters and the simulation responses.

3.2 Probability theory: inference

In probability theory, the figurative act of "slicing" a JPD, is a task of probabilistic inference, where slicing the JPD at different intervals, reveals probabilistic relationships between the random variables. More formally, these probabilistic relationships are referred to as *conditional* relationships and are described by conditional probability.

3.2.1 Conditional probability

To illustrate the notion of conditional probability, let us consider a simple two-dimensional example. Consider the JPD of the depth of a cantilever beam (x-axis) and its maximum displacement (y-axis), illustrated in Figure 3.7. In this figure, the JPD is visualised as a kernel density estimate (KDE) contour plot of the parametrically generated design points, for illustration's sake only. A KDE is a type of mathematical nonparametric smoothing algorithm.

Slicing the JPD at beam depth values of for example 1.25mm, 1.75mm and 2.25mm, reveals probability distributions of maximum displacement as an effect of fixing the beam depth variable at the respective values. These slices are the conditional probability distributions of the maximum displacement of the beam at each of the beam depths. The term 'conditional' is motivated by the fact that each outcome of maximum displacement is conditioned by its mechanical relationship with the depth of the beam. In this context, 'conditional' can be thought of as a synonym for

'related' in probability theory. In general, conditional probability of a variable X given that Y is written as $P(X|Y = y)$ where, the vertical line "|" is read as "given". Therefore, in this example, the conditional probability of maximum displacement given that beam depth is 1.25mm is expressed as, $P\ (\text{max displacement} \mid \text{depth} = 1.25)$

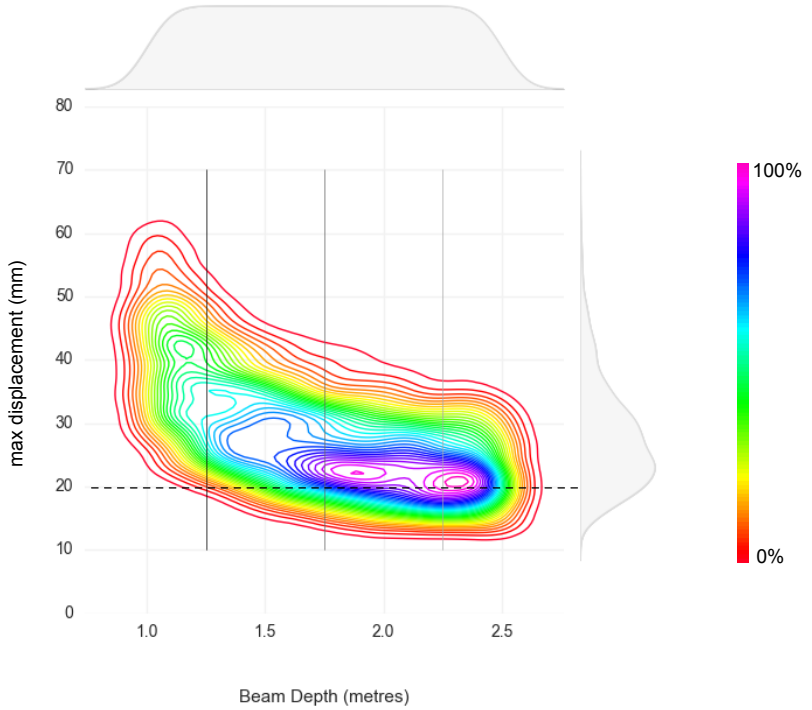


Figure 3.7: JPD (kde) of beam depth and max displacement.

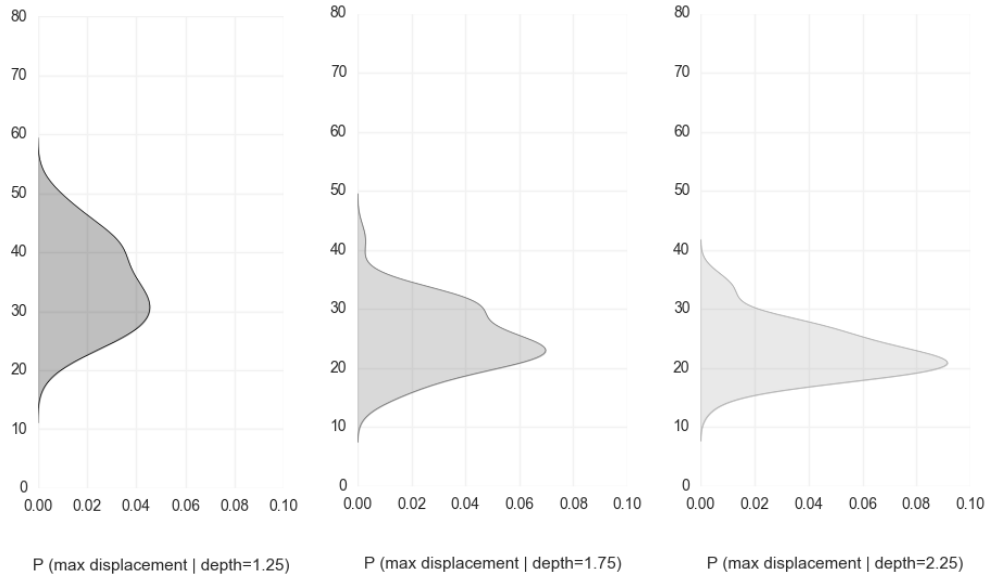


Figure 3.8: Conditional probability distributions of max displacement at depths: 1.25mm, 1.75mm and 2.25mm.

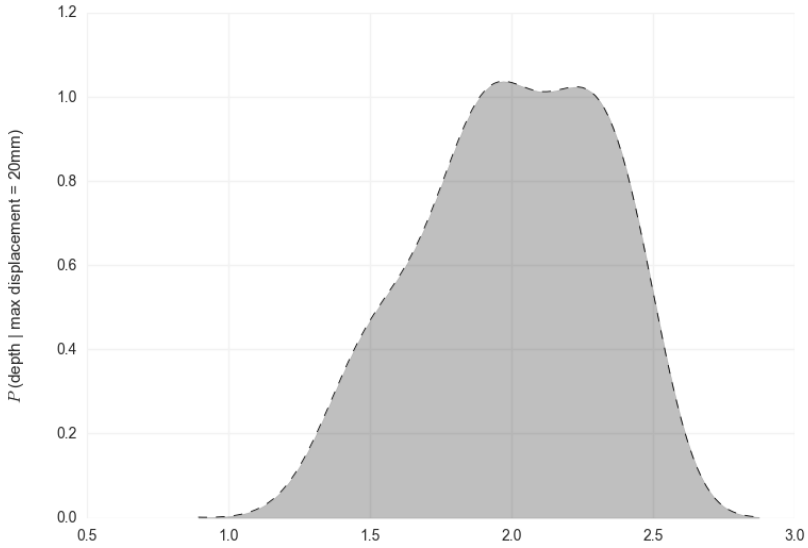


Figure 3.9: Conditional probability distribution of beam deth for max displacement at 20mm.

Figure 3.8 and Figure 3.9demonstrate clearly the anatomy of a JPD; a composition of a series of conditional probability distributions (slices). The next section discusses how to determine conditional probability distributions using inference techniques from probability theory. More particularly, we introduce Bayesian inference.

3.2.2 Bayesian Inference

Bayesian inference is a method of probabilistic inference in which Bayes' theorem is used to "update" the probability distributions of a random variable under study when values of other variables are kept fixed. In other words, Bayes' theorem is used to compute the conditional probabilities from a given JPD.

Bayes' Theorem, is named after its creator, Rev. Bayes (1763), who was a famous mathematician. Bayes' theorem is given by Eqn. (12).

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{12}$$

where,

$P(A)$ and $P(B)$ are the *marginal* probability distributions of A and B; i.e. the probability distribution of each being true without regard to each other. The marginal distributions can be recovered any time from the JPD. In Bayes' theorem $P(A)$ is also referred to as the *prior*.

$P(A|B)$ is the conditional probability of observing event A given that we know the value of B. In Bayes' theorem the term on the left hand side is referred to as the posterior.

$P(B|A)$ is the conditional probability of observing event B given that we know the value of A. In Bayes' theorem the term $P(B|A)$ is the *likelihood* of B given that we know A.

In this research, we adopt Bays' theorem as a bi-directional inference mechanism to predict the probability distribution of simulation response Y for input values X of interest. More specifically,

$$P(Y|X = x^*) = \frac{\mathrm{P}(X|Y)(\mathrm{P}(Y)}{\mathrm{P}(X)} = \frac{\mathrm{P}(\mathrm{Y} \cap X)}{\mathrm{P}(X)} \tag{13}$$

Further, in this research we take advantage of the indifference between inputs and outputs in a JPD, to use Bayes' theorem to compute the inverse problem in Eqn. (14), i.e. to predict the probability distribution of the inputs X for simulation response of interest. In practice, Eqn. (14) implies that we can identify the joint probability distribution of X1 and X2 for specific values of Y, and subsequently use marginalisation to derive P(X1) and P(X2).

$$P(X|Y = y^*) = \frac{\mathrm{P}(Y|X)(\mathrm{P}(X)}{\mathrm{P}(Y)} = \frac{\mathrm{P}(X \cap \mathrm{Y})}{\mathrm{P}(Y)} \tag{14}$$

As can be observed from Eqn. (13) and Eqn. (14), the respective computation of $P(\mathrm{X}|Y = y^*)$ and $P(Y|\mathrm{X} = x^*)$ are identical. The mathematical indifference suggests that a probabilistic metamodel can be used as a bi-directional inference mechanism to compute both input→outputs and vice versa, outputs→inputs.

3.2.2.1 Bayesian inference in the design space

To illustrate further the statistical concept of Bayesian inference, we can think of the Cartesian design space (bound by a set of inputs X and simulation response Y), as a mathematical set S

composed of all possible combinations of sampled X values and yielded Y values, which together make up a probability of 1, hence S=1. While relationships in typical regression models are defined by deterministic functions, in probability, conditional relationships are expressed in terms of the relative proportion of 'overlap' between probabilities (Figure 3.10). In probability, the overlap represents the probabilities of two events occurring simultaneously, however in the simulation metamodel scenario, the overlap represents the effect of one variable on the other. Therefore, using Bayes' Theorem in Eqn. (13) to predict the simulation output y, corresponds to the ratio of X, containing overlap with Y $[P(Y \cap X)/\ P(X)]$. Similarly using Bayes' theorem in Eqn. (14) to infer the values of input variables X, corresponds to the proportion of Y containing overlap with X $[P(X \cap Y)/\ P(Y)]$.

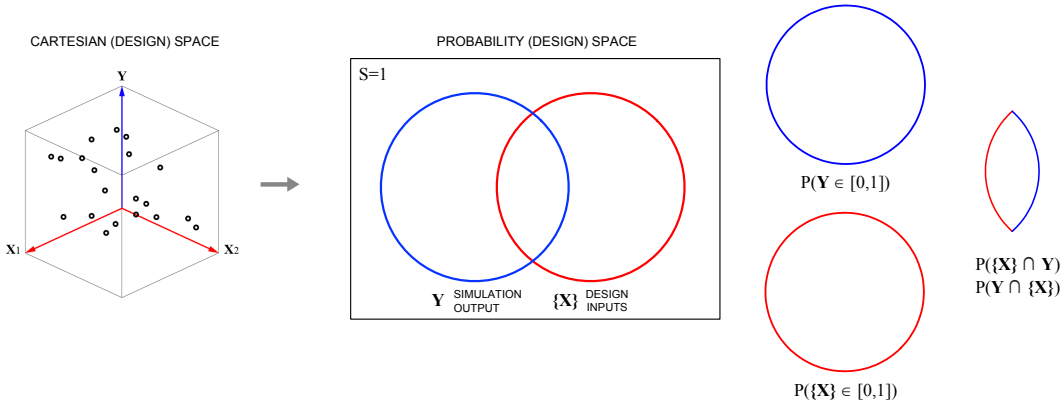


Figure 3.10: Illustrative breakdown of the design space represented as a space of probabilities.

With reference to the Venn diagram illustration, when performing inference, we are therefore reducing the space of all possible outcomes to the probability space concerned only with the outcome of interest. When translating this notion in terms of the design space, the operation of $P(X|Y)$ (which can also be denoted as $Y \rightarrow X$) implies that inference can reveal how the design variables (X) are effecting the physical behaviour (Y), suggesting that we can immediately identify which ranges of input values are most likely to reach the performance goal of interest. In other words, we can reduce a vague design space of all possible solutions, to a more meaningful design space because it would concern only solutions that produce a feasible response within the range of interest (Figure 3.11).

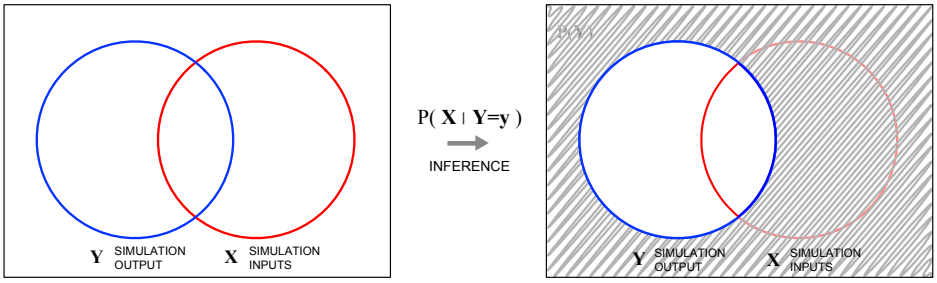


Figure 3.11: Reducing the space of all probable input configurations to the ones that concern the output (Y=y) being queried.

3.2.3 The curse of dimensionality

In general, a full JPD table needs k^N parameters (N variables, k states). This implies that as the number of variables or number of considered states increase, it becomes increasingly challenging to specify the full joint probability distribution (JPD) because the fully specified JPD can become too large to store computationally, require large amounts of data to represent, and subsequently the probabilities in the JPD become vanishingly small and no longer meaningful (Karkera, 2014).

For these reasons, in this research we adopt Probabilistic Graphical Models (PGM), which are a type of probabilistic model.

3.3 Probabilistic Graphical Models (PGM)

PGMs deal with JPDs involving multiple random variables because they avoid having to specify the full joint distribution by using the concept of *independent parameters*. This implies that PGMs can be useful to handle probabilistic inference over many random variables.

3.3.1 Graphs

A graphical model is a graph-based representation of the dependencies between multiple variables. By definition, a *graph* is a representation of a mathematical structure used to model relations between objects, in the form of nodes and edges. A classic example is the graph representation of the historically notable mathematics problem of the seven bridges of Konigsberg where the problem was to devise a walk through the city that would cross each of those bridges once and only once. Euler (1741) abstract representation of the problem as a graph proved that there was no solution to this problem.

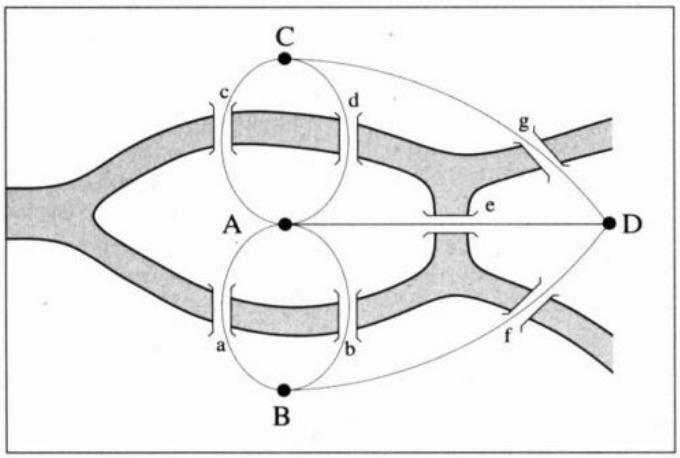


Figure 3.12: Seven bridges of Koningsberg can be solved as a mathematical graph (Kadesch, 1997).

In PGMs, the nodes represent random variables while the edges correspond to direct probabilistic dependencies between the connected nodes (Murphy, 2001). In other words, a PGM is a representation of the JPD via a graph structure. In various domains, the visual aspect of PGMs convey the important structural information of a complex problem in a natural visual manner (W. Buntine, 1996a). PGMs can be categorised into two types: directed and undirected.

3.3.2 Directed and undirected PGMs

Directed graphical models (DGM) are also referred to as Bayesian Networks or Belief Networks (BNs) while, undirected graphical models (UGM) are also referred to as Markov Random Fields (MRFs) or Markov Networks (Jordan, 2004; Murphy, 2001). 'Directed' refers to a specified direction of the edge (generally indicated by an arrow), indicating a causality between two nodes (Figure 3.13, left). For example, a directed edge going from node X to node Y indicates that random variable X is the parent causing an effect on random variable Y. On the other hand, the edges of UDGs have no direction (Figure 3.13, right), as they do not encode any information about causality. Nonetheless, edges of UDGs still represent a direct relationship (probabilistic dependency) between the nodes, the same as in Bayesian Networks (Koller, Friedman, Getoor, & Taskar, 2007). In some cases it is possible to use a mixture of both DGMs and UDGs; these are called chain graphs (W. L. Buntine, 1995; Lauritzen & Wermuth, 1989).

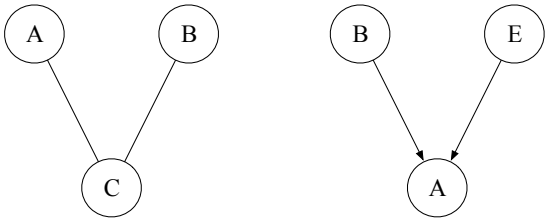


Figure 3.13: Undirected graph (left), directed graph (right).

In general, graphical models are widely adopted for problem solving because they can exploit the dependence properties that exist in the most real-world problems including the fields of science and engineering (Koller et al., 2007). Typically, undirected graphical models are more popular with physical and vision communities, whereas directed models are more popular with the AI and statistics communities (Murphy, 2001).

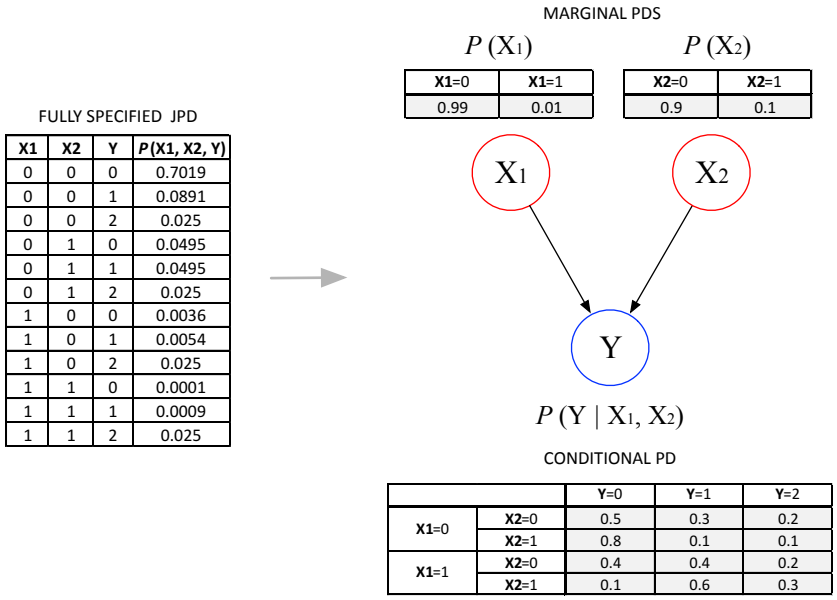
In this thesis we adopt directed graphs, namely Bayesian networks as a form of probabilistic metamodel for representing causal relationships between multiple parameters and multiple simulation responses in such a way that can be explored bi-directionally.

3.4 Bayesian Networks (BN)

A Bayesian Network is a directed type of PGM where directed edges represent causal relationships between nodes. More specifically, a directed edge represents conditional dependence between two random variables. Therefore, when assembled together a BN is a network of conditional dependences which together efficiently describe a JPD without having to fully describe all the parameters. This compaction is achieved by converting the global distribution (JPD) to local conditional distributions at each node, in terms of the parent nodes connected to that node (Pearl & Russell, 1998) (Figure 3.14). These local probability distributions can be either marginal, for nodes without parents ($P(X1)$ $P(X2)$, or conditional, for nodes with parents ($P\ (Y|X1, X2)$). In the latter case, the dependencies are described by a conditional probability table (CPT) for each node given its parents in the graph (Figure 3.14) .

Let's go through each probability table in the BN in Figure 3.14. Random variables X1 and X2 take two values each and therefore need only require independent parameter each. The Y table has 12 (3x4) parameters. However, each row sums up to 1, and therefore, we only need two independent parameters per row. The whole table, therefore, needs 8 (2x4) independent parameters. Therefore, the total number of parameters for this network is 1 (X1) + 1 (X2) + 8 (Y)

= 10, which is less than 12 parameters required by a fully specified JPD. This fragmentation of the JPD is referred to as factorisation. Two less parameters might not seem like great savings in this example but as the number of parameters and states increase, the savings become more effective because the number of parameters in a CPT grow linearly while in a fully specified JPD they increase exponentially. In the case study example in section 4.3.2, we will demonstrate how we can drastically reduce the number of parameters by reversing the causal edge directions.



X1	X2	Y	P(X1, X2, Y)
0	0	0	0.7019
0	0	1	0.0891
0	0	2	0.025
0	1	0	0.0495
0	1	1	0.0495
0	1	2	0.025
1	0	0	0.0036
1	0	1	0.0054
1	0	2	0.025
1	1	0	0.0001
1	1	1	0.0009
1	1	2	0.025

X1=0	X1=1
0.99	0.01

X2=0	X2=1
0.9	0.1

		Y=0	Y=1	Y=2
X1=0	X2=0	0.5	0.3	0.2
	X2=1	0.8	0.1	0.1
X1=1	X2=0	0.4	0.4	0.2
	X2=1	0.1	0.6	0.3

Figure 3.14: Bayesian network representation of a JPD.

In probability, we define the compact representation of the JPD in product form, using the chain rule. For example the JPD of the example in Figure 3.14 is:

$$P(X1, X2, Y) = P(Y|X1, X2) * P(X1) * P(X2) \tag{15}$$

With the above in context, we can therefore, introduce simulation inputs and outputs as nodes in a Bayesian network, whose directed edge structure reflects the causal direction of computation between simulation inputs and output, respectively.

Bayesian networks take on only discrete probability distributions, therefore, all continuous input and output values must be discretized respectively. In this context, we can think of a BNM as a discretised n-dimensional design space where each hyper-cell (hyper-parameter) is assigned with a probability (Figure 3.15). In other words, each n-dimensional pixel corresponds to a cell in the n-dimensional JPD table.

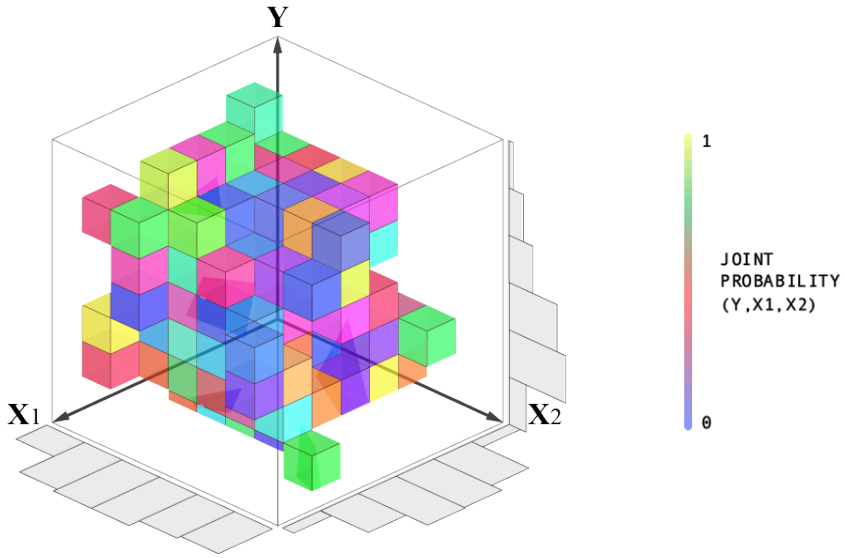


Figure 3.15: Illustration of a discretised JPD as a discrete approximation of the design space.

3.4.1 Learning Bayesian networks

From the previous section we can summarise that learning a Bayesian network involves two mains steps: *structure learning* (graph topology) and *parameter learning* (CPTs). Both can either be specified manually or learned automatically from data. More specifically, if knowledge about the problem domain is available, the CPTs can be specified manually, otherwise they can be learned automatically from data using supervised learning algorithms. Supervised learning is a machine learning task to discover the parameters of a predictive model by mining data. In Bayesian networks the parameters can be learned directly from data, either using expectation-maximization (EM) or maximum-likelihood estimation (MLE), i.e. the relative frequencies of the observed combinations of the values. Both are supervised learning algorithms. Further supervised learning algorithms are discussed by Lucas, van der Gaag, and Abu-Hanna (2004). For detailed description regarding automatic learning of conditional probabilities see and Lucas et al. (2004); Spiegelhalter (1998).

The graph topology of a BN can be also either be encoded manually or learned automatically from data. There exist two main structure learning methods: Bayesian and constraint-based approaches. The Bayesian method (Heckerman, Geiger, & Chickering, 1995) involves a combination of constructing the arcs manually based on expert knowledge and automatic learning from data to find the most likely structure. However, this can be rather computationally demanding (Steck & Tresp, 1999). On the other hand, the constraint-based approach searches for conditional dependence and independence in the data, and then find a model structure that best

explains these dependencies and independencies. These are computationally lighter hence why they are more common. For further reading on automatic structure learning, see Steck and Tresp (1999).

The major advantage of automatic learning of the parameters and structure is that they do not require expert knowledge of the underlying domain. While the automatic approach is very advantageous, the learning process is more efficient if prior knowledge is available. When this is the case, BNs are advantageous as they make it easy to combine different sources of knowledge; expert knowledge can be combined with data (Marcot, Holthausen, Raphael, Rowland, & Wisdom, 2001). Besides different sources, data measured with different accuracies can also be combined.

Nowadays, various applications are available for building Bayesian networks. These are embedded with a variety of algorithms for automatically learning structure, parameters and various other machine learning techniques for data mining. An example of such software is BayesiaLab ® (Conrady & Jouffe, 2010). Alternatively, open-source options are available in the Python programming language; such as *libpgm* (CyberPoint International, 2012) and *pybbn* (Vang, 2017).

3.5 Review of BNs as metamodels

3.5.1 Other disciplines

The application of BNs to simulation metamodeling is rather novel but is slowly gaining popularity in numerous research domains. D. N. Barton et al. (2012); Uusitalo (2007) review pros and cons of BN applications for modelling uncertain and complex domains such as ecosystems and environmental management. Poropudas and Virtanen (2011) present a dynamic BN for tracking the evolution of a simulation metamodel over time. This is particularly useful for time dependent problems such as queues analysis and air combat. Pousi, Poropudas, and Virtanen (2013) adopt BNs as an exploratory metamodeling tool for supporting simulation studies conducted with stochastic simulation models containing multiple inputs and outputs. They take advantage of the various analyses allowed by BNs such as input uncertainty and inverse reasoning, as a means to guide and aid utilisation and interpretation of the simulation model under consideration. Liang and Mahadevan (2015) adopt BNs as a simulation surrogate model (metamodel) to incorporate different uncertainty sources for multi-objective optimisation purposes.

3.5.2 Building design

In the architectural field, past work on the applications of BNs, seem to deal with very specialised engineering problems such as fatigue crack growth analysis in structures (Sankararaman, Ling, & Mahadevan, 2015). However, there seems to be very little attention given to engineering applications of BNs in an architecture-engineering discourse: Naticchia (1999) and De Grassi and Naticchia (2001) present a general approach for modelling physical behaviour by means of BNs for preliminary architectural design stages, with specific focus on the ventilation-design of halls. Their scope is to guide students or architects with no understanding of fluid-dynamics, through the analysis process of numerical data and to suggest improvements to the design. This was compiled into a software tool, which relies on a probabilistic causal representation, to qualitatively express knowledge of fluid-dynamics needed to explain the ventilation behaviour of a space. Furthermore Naticchia et al. argue that while the results of the CFD simulation seem to give all the needed information of interest (both numerically and visually rendered), they become intractable when one then questions on how to use this information to improve the design for better ventilation, leading to a trial and error approach of simulation. They claim that performing inference over a BN can answer these questions and guide the non-skilled user. Their application of BNs, involved expert-based assumptions about the selection of variables and the structure of the network. We note that the attempt by Naticchia et al. demonstrates the untapped advantages that metamodeling with BNs can have compared to other metamodeling techniques reviewed earlier.

It is important to note that despite the advantages of metamodeling with BNs, typical methods such as polynomial regression and Kriging can be more computationally efficient to build. However, they lack the explanatory power, especially when dealing with many inputs and outputs, as discussed in Chapter 2.

Chapter 4

BAYESIAN NETWORK METAMODEL

This chapter describes the steps involved to build a Bayesian network metamodel (BNM) in the context of coupled-design analysis systems.

Typical simulation metamodels such as those described in Chapter 2 involve three overall steps: 1) setting up a computer experiment which involves running multiple simulation runs with different configurations of input values to generate a dataset, 2) constructing the metamodel from the generated dataset, and finally 3) verifying how well metamodel can predict for new inputs. The same steps apply to Bayesian network metamodels (BNM) except for two additional sub-steps in step 2, and as highlighted in Figure 4.1.

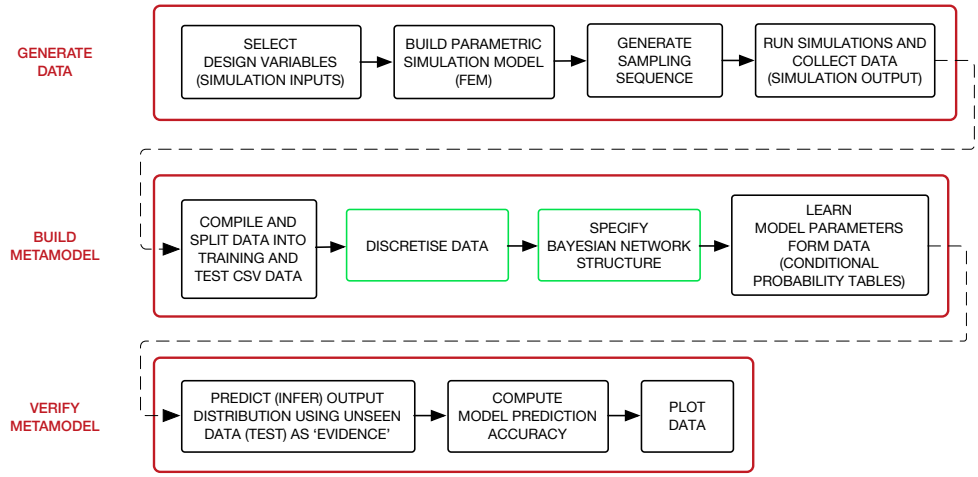


Figure 4.1: Overall Bayesian network metamodel workflow

The following sections describe each of these steps in further detail by making reference to a case-study interpretation of an existing structure.

4.1 Illustrative example

For explanatory purposes, we select a case study from structural engineering. Inspiration was drawn from one of the cantilevered roof structures designed by famous Italian architect and structural engineer, Pierre Luigi Nervi (1891-1979). This structure is found in the municipal stadium of Florence and was built between 1930 and 1932. Nervi's design integrates the cantilevered roof within the seating grandstand resulting in a cost and structurally efficient solution. This project is an example of a good synthesis between architectural design and structural engineering.

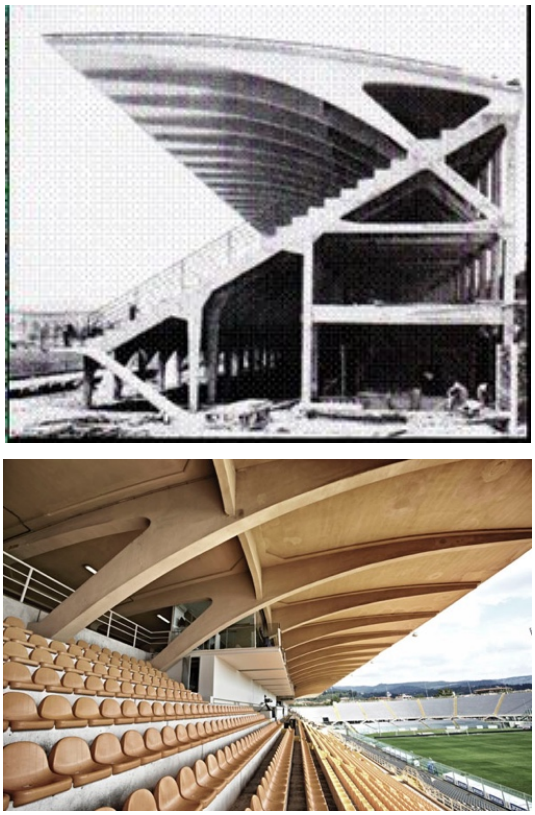


Figure 4.2: Top: side elevation of the integrated roof grand stand structure (book Ingg. Nervi & Bartoli, societa per azioni). Bottom: recent photograph of the seating roofed over by the cantilever.

The roof structure has a total length of 100 metres and consists of a thin reinforced concrete shell spanning continuously between 24 curved reinforced concrete cantilever beams (Figure 4.2, right). The roof structure region between A and B is referred to as the 'back span' and the part between the interior column B and the roof tip C is referred to as the 'cantilever' (Figure 4.3). The cantilever measures ~14.5 metres while the back-span measures ~ 7.4 metres. The depth of the cantilevering part of the beam varies between 0.4 metres at C and 2.6 metres at B, before it forks into two beams with a depth of 0.55 metres each (Figure 4.3). The shell thickness and the

roof beam were assumed to be 0.10 metres and 0.40 metres wide respectively. These dimensions were scaled from Nervi's original drawings as presented in a recent study (Adriaenssens & Billington, 2013).

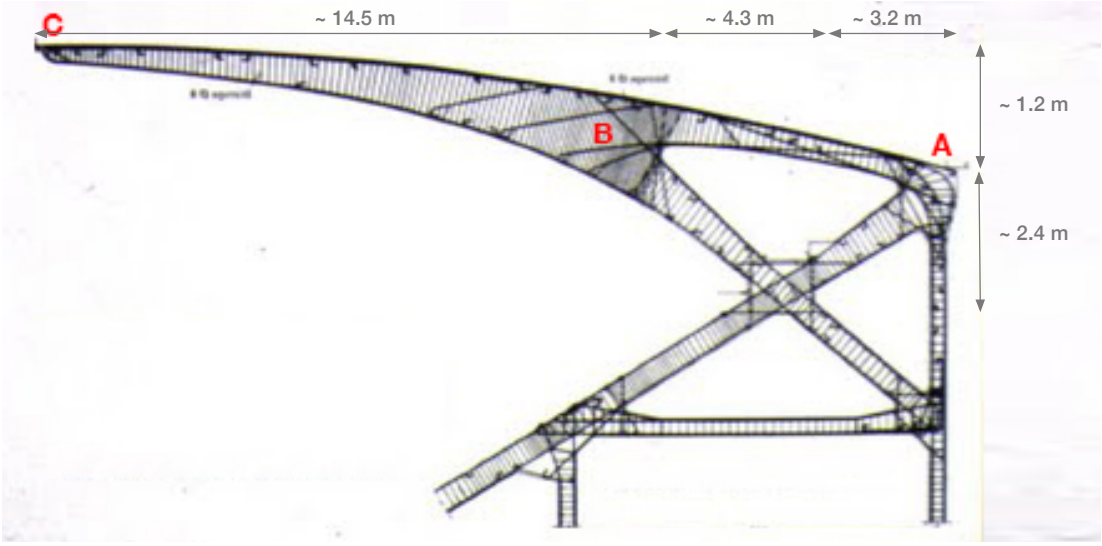


Figure 4.3: Reinforcement drawing of one of the cantilevering beams supporting the thin concrete shell (Adriaenssens and Billington 2013).

4.1.1 Parametric geometry

For this case study, we opt for a 2D frame analysis (Figure 4.4) of the cantilevering beam illustrated in Figure 3 to estimate the maximum deflection of the structure. We opt for a simple 2D analysis problem to avoid any potential issues typically associated with 3D solid modelling for simulation and rather, focus the efforts on validating the statistical aspects.

A parametric 2D frame of the geometry was modelled in Grasshopper ® (Rutten, 2012) and used as a frame model for 'Millipede' (Michalatos & Kaijima, 2014), which is a Grasshopper plugin for structural analysis. For this frame model, we extract six geometric variables of interest from Nervi's original geometry: X position of vertex C (C_X), Y position of vertex C (C_Y), amplitude of the curve CA (B_pos), X position of E (E_X), depth of beam profile at C (C_depth), and depth of beam profile at B (B_depth). The varying depth in Nervi's original design plays a critical role in his structural concept as it is dictated by the law governing the variation of moments in the structure, and was therefore important to consider when simplifying the geometric model. However, since frame models are not compatible with representing plane

strain, we opted for a work-around by discretising the frames whose depth varies, particularly BC, BA, and BE. The frame between B and C was discretised into varying heights by interpolating values between variables C_depth and B_depth respectively to mitigate the varying depth as in the original structure. The same was applied to the frame between B and A, where the depth at A was fixed to 0.55 metres. All variables were allowed to vary within their respective ranges as tabulated in Table 3. All remaining members were fixed at dimensions as per Nervi's built structure and as indicated in Figure 4.4.

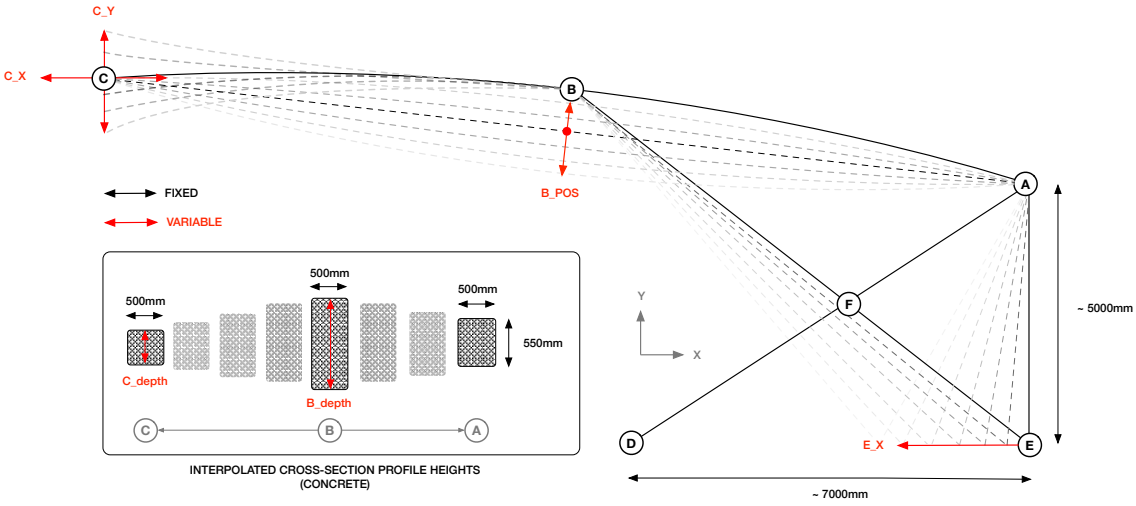


Figure 4.4: Parametricsied geometry of cross-sectional beam supporting the shell roof. The solid black frames indicate the fixed geometry as per Nervi's original structure.

	Variable	*Notation*	*Max*	*Min*
Inputs	X coordinate of C	C_X	-1.5m	2m
	Y coordinate of C	C_Y	-3m	2m
	Amplitude	B_pos	-1.5m	2m
	Beam depth at C	C_depth	0.3m	1m
	Beam depth at B	B_depth	1m	2.5m
	X coordinate of E	E_X	0m	5m
Outputs	Deflection	Deflection		

Table 3: Selected input and output variables, and the respective input ranges.

4.1.2 Parametric finite element model

All frame members' cross-sectional profiles were assumed to be constructed from solid concrete with a density of 2400 kg/m3. Given, the volume of the shell and density of concrete we assumed a uniform load of 10819 N/m is carried by one beam. This was applied uniformly to frames BC and BA. This value was estimated with the assumption that the total load carried by one beam is

equivalent to two halves of the shell spanning laterally between two beams (two half spans, along each side of the beam). D and E were fixed in translation and rotation for all directions.

As a default case and verification of the frame model, the geometry corresponding to that of Nervi's design, produced a maximum deflection of 37mm, at point C (Figure 4.5). This value compares well with Nervi's calculation of 38mm at the cantilever tip, before construction (Nervi, 1955). Adriaenssens and Billington (2013) also indicate a deflection of 32mm from their analysis.

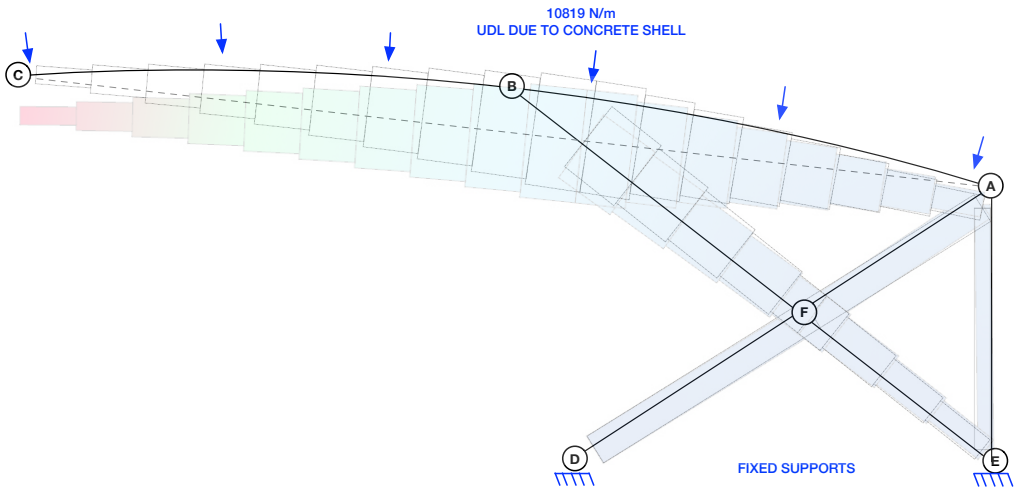


Figure 4.5: Loading and support conditions considered during the simulation.

4.2 Generate data

This step involves setting up a computer experiment using engineering simulation. This involves, selecting the design parameters, setting up a parametric simulation model, generating a sampling sequence and finally generating and collecting the simulation response data.

4.2.1 Selecting the metamodel inputs

The number of inputs impacts directly the number of data points required for building the metamodel. When one simulation run is time consuming, a large number of data points might be cumbersome to compute. For these reasons, the selection of inputs must be done carefully such that: inputs that will demonstrate most insight should be selected. In cases where the inputs are restricted, then other methods of dimension reduction may be employed as a means to identify the inputs that are most sensitive on the response to be predicted. Dimension reduction methods include sensitivity analysis and PCA.

For this example, we make use of the geometric parameters as the metamodel inputs.

4.2.2 Sample the input space

Once the metamodel inputs were selected, the next step is to sample the input space. In other words, to generate a configuration of simulation input to run. We assume that we know little to nothing about the functional form relating the inputs and outputs of the numerical model underlying the simulation, meaning that interesting characteristics of the function such as maxima and minima are likely to be anywhere in the design space.

We make use of space-filling design namely, quasi-random sequences.

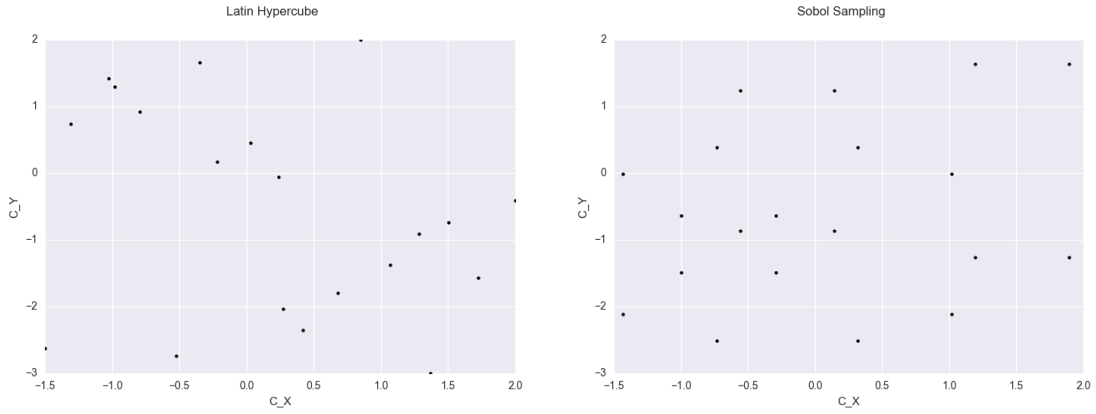


Figure 4.6: Comparative 2D scatter plots of pseudo-random (left) and quasi-random sampling (right) for a sample size of 20.

Figure 2.6 illustrates scatter plots of two of the six dimensions in the design problem under study in this case study (variables C_X and C_Y). The plots suggest that the samples (points) generated by Sobol' sequences are more evenly distributed than those generated by LHS. For the sake of this example, samples of only 20 points were generated as it becomes difficult to identify the difference in discrepancies at higher densities such as those studied in this experiment. Python libraries 'pyDOE' (Lee, 2014) and 'SALib' (Herman, 2014) were used to generate the LHS and Sobol' samples in Figure 2.6, respectively.

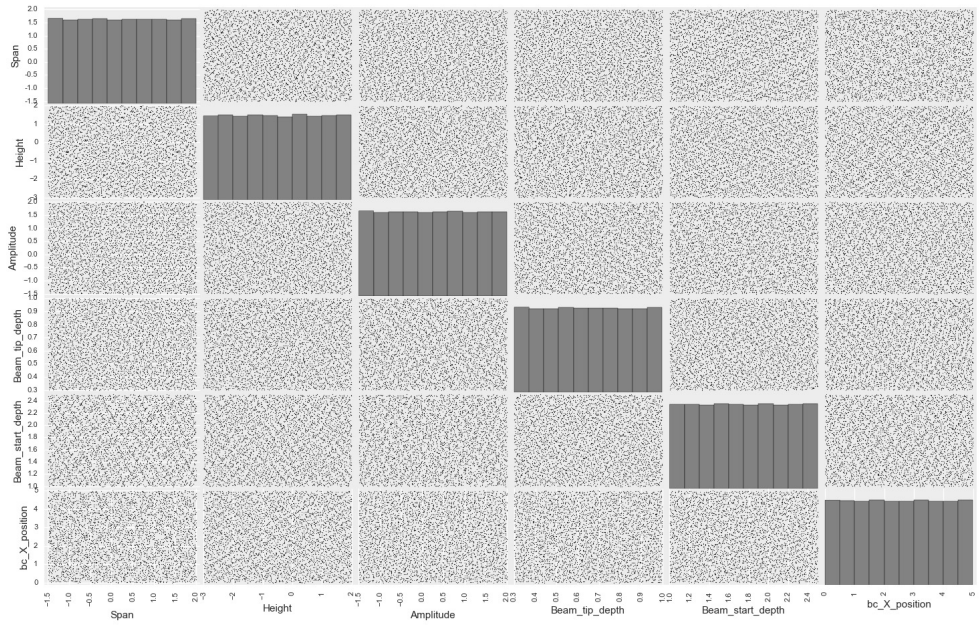


Figure 4.7: Matrix plot of the entire input space.

When runs of computer simulation are expensive or time-consuming, what is a reasonable size of samples to ensure a reasonable prediction accuracy? There is a large body of literature on this topic (Rai & Campbell, 2006; G. G. Wang & Shan, 2007), however the answer depends largely on the type of problem under study, in other words, on the complexity of the functional form of the relationship between the inputs and the outputs of the simulation model. Furthermore, this complexity is dictated by the number of variables in the problem. These factors have an impact on the quality of the metamodel and therefore on the accuracy of its prediction.

For the above reasons, we studied six different sample sizes (500, 1000, 4000, 10000, 12000, 14000) for both LHS sampling and Sobol based sampling, meaning a total of 12 metamodels. The scope was to explore the sensitivity of the number of samples and the type of sampling method on the predictive accuracy of the metamodel. The scope is to advise the formulation of future metamodels of similar design problems with this information. For the case study, we also utilised 'pyDOE' (Lee, 2014) and 'SALib' (Herman, 2014) libraries within Python environment to generate the sample sequences.

Notes: (i) LHS samples are generated as normalised values (0,1) and were mapped onto the respective ranges specified in Table 3. (ii) We also generate samples using a Factorial Design of Experiments (DOE) method as a third sampling method for comparison's sake.

4.2.3 Evaluate the samples

Once the input sequences are generated, we assigned them to the parametric frame model in Grasshopper as slider input values. For each simulation run, the maximum deflection of the structure was recorded from the simulation output and stored to a CSV file.

4.2.4 Split data into training/testing

The input sequences and recorded output values were tabulated in the form of a CSV dataset, such that each row consisted of the sequence of input values and the resulting deflection value. Each row corresponded to one simulation run therefore, the total number of rows equated to the total number of simulation runs.

In general, when constructing a predictive model, it is standard practice to split the generated dataset into a training set and a testing set. The former is used to build the actual statistical model while the latter is used to verify its prediction accuracy by presenting it with input values it has not 'seen' before and then assessing the difference between the predicted and the simulated value. A popular rule of thumb splits the ratio between training and test data at 80%/20%, respectively. In the field of machine learning, this ratio is often referred to as the Pareto principle. The validity of the accuracy estimates is related to the number of data points kept aside for the test set (Hamad, Al-Zaben, & Owies, 2014). However, if the overall data set (training and test) is large, then any division (within a reasonable range, i.e. excluding extreme ratios such as 5/95) may work. On the other hand, if the data set is small, then the split has a crucial role in the prediction accuracy. For this case study, we consider three splitting ratios: 80/20, 50/50, 20/80 to gain an understanding of the sensitivity of the split ratio on the prediction accuracy.

Other statistical model validation approaches such as cross-validation require different ways of splitting the dataset. These will be investigated in future research.

4.3 Build Bayesian network

In Bayesian networks, inputs and outputs are presented as discrete random variables, represented by discrete probability distributions. After generating the input-output dataset, each variable needs to be discretised prior to building the Bayesian network. Subsequently, the structure of the network is specified, and lastly, the conditional probabilities of the network are learned based on the data.

For this example, we use Python programming language as a platform for executing the following steps. The workflow was written into a Python library and eventually as a plug-in for Grasshopper in Rhino6. These are discussed in more detail in Chapter 5.

4.3.1 Discretisation

Bayesian Networks take only discrete distributions. Therefore, any continuous distribution needs to be discretised into a fixed number of intervals. For example, all input distributions generated using space-filling approaches such as LHS or Sobol' sequences are sampled from continuous ranges and thus, need to be discretised.

The discretisation of a variable involves dividing the interval of the variable into a fixed number of in-between ranges, referred to as 'bins' and then, sorting the variable values according to their corresponding range. The sorting task is called 'binning'. The way in which the continuous variables are discretised must be thought about carefully because it has a direct impact on the prediction accuracy of the Bayesian network. There are a variety of discretisation algorithms however, in this thesis we focus on 'equal distance', and 'percentile' binning. The former divides the extreme range into a number of equal-sized bins as illustrated in Figure 4.8, while the latter approach divides the continuous range into dynamically-sized intervals such that each bin contains an equal frequency of observations, as illustrated in Figure 4.9. The red vertical ticks in both figures represent the observations. Percentile binning is necessary to avoid scenarios where outlying observations due to extreme responses, result in bins containing only a few observations, and other bins containing the majority of the observations, as illustrated in Figure 4.8. Consequently, the model will erroneously predict the "correct" bin given that most observations lie there.

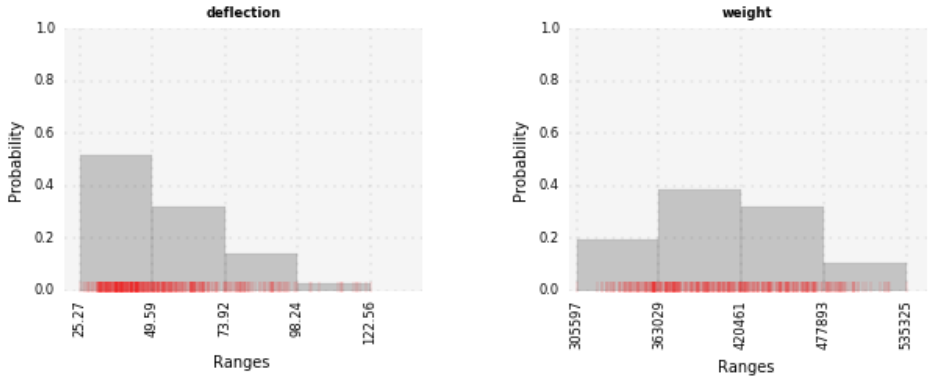


Figure 4.8: Discretisation of continous ranges into equidistant bins.

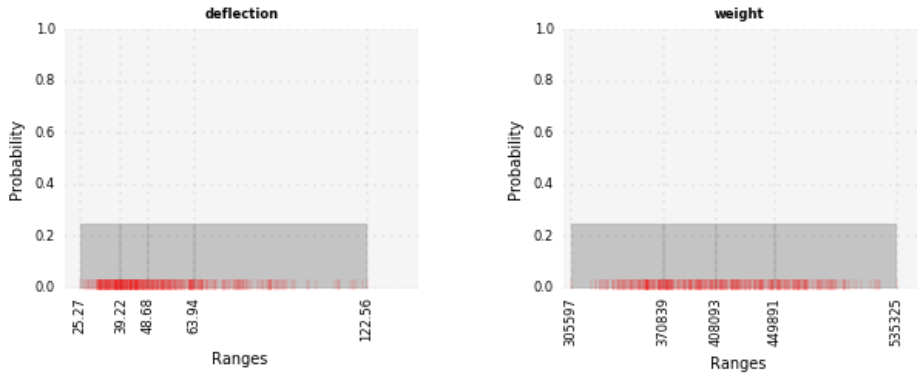


Figure 4.9: Discretisation of continous ranges into percentile bins.

As a rule of thumb, in the presented metamodeling framework, we specify that, an input whose distributions are driven from space-filling sampling algorithms, are to be discretised using equal binning given their pseudo-uniform distribution. On the other hand, simulation responses are to be discretised using percentile binning.

From a prediction robustness perspective, it is desirable for each bin to capture a fair amount of data. The term ‘fair’ is used vaguely because the robustness of Bayesian networks is a matter of trial and error to some extent. There is no obvious method or rule for deriving the number of bins. The number of bins corresponds to the number of states of a random variable and therefore, the number of parameters that are required to learn the network. Therefore, increasing the number of model parameters implies an exponential increase in computational demand when computing inference. On the other hand, too few bins might not provide enough states for revealing desired insight about the inputs and outputs. Therefore, the trade-off here lies with selecting a number of intervals and their bounds, desirable by both the the user’s interest and the robustness of the Bayesian network.

In this research, we also looked into discretisation methods that are more autonomous such that the number of bins and the ranges of the intervals are selected automatically based on the parameters and relationships underlying the model. One such ‘automatic discretisation’ method is the Minimum Description Length algorithm, which discretizes the continuous attributes of data matrix using entropy criterion with the Minimum Description Length as a stopping rule. In future work, we intend to develop this further and embed it into the metamodeling workflow. There exists a dedicated body of work on such methods; see Levine (2011); Nonchev (2015).

For this example, we discretise each random variable into 6 bins equidistantly for the inputs and by percentile for the outputs, as illustrated in Figure 4.10.

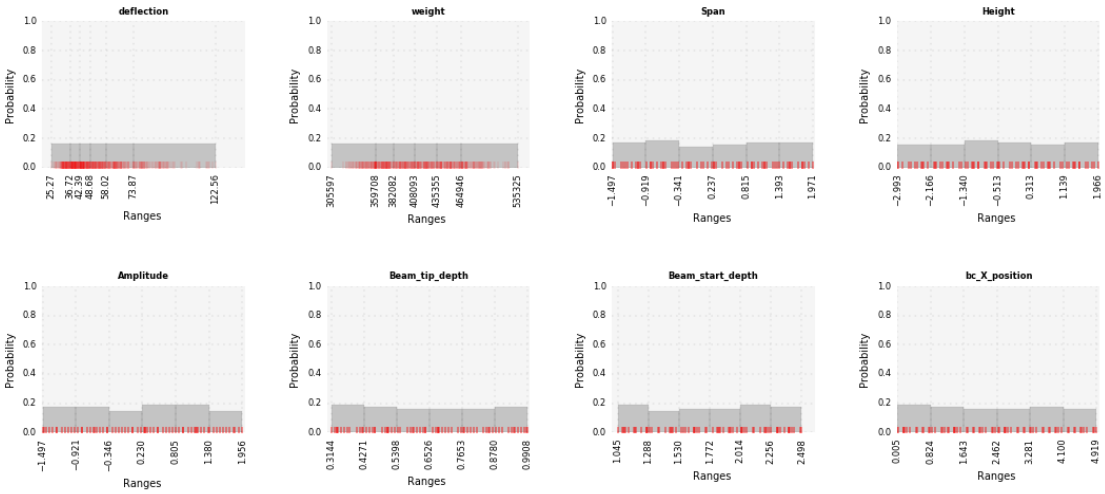


Figure 4.10: All discretised distributions.

4.3.2 Bayesian network topology

The network topology of a probabilistic metamodel does not change from one problem to another. The structure of the network is always interpreted as a set of edges between input nodes and output nodes, and whose direction flows from inputs to output, as illustrated in Figure 4.12.

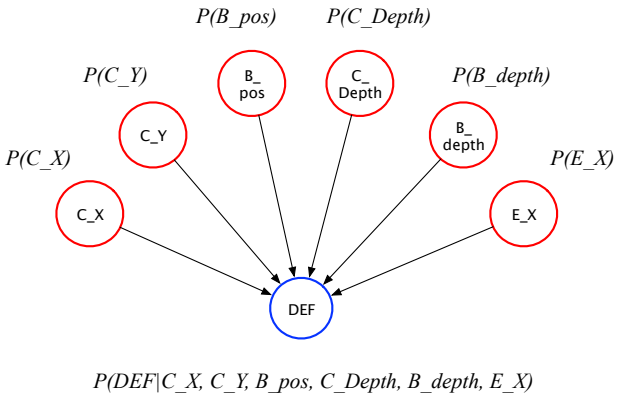


Figure 4.11: Input/output model Bayesian Network.

In general, when a node in a BN receives a large number of directed edges, the CPT can become so large that the number of cells in the table exceeds the number of samples. As a consequence, the conditional probabilities in the table become very sparse and meaningless. To avoid such a scenario, we can reverse the direction of the edges by taking advantage of the argument that causal relationships cannot be identified from observational data, meaning that the direction of the edge does not represent a causal relationship when the conditional probability distributions are learned from data and therefore have no negative effect on the prediction accuracy (Ellis & Wong, 2008). Causality is a topic of debate in the probability community (Pearl & Mackenzie,

2018). However, in the context of Bayesian Networks, many agree with the above statement, claiming that causal inference is only possible when casual information is available utilizing a controlled experiment and or from domain theory (Ellis & Wong, 2008). Thus, in our case, the direction of the edges can be specified arbitrarily since we are dealing with pseudo-randomly sampled data from a simulation model, whose underlying numerical model we assume little to no knowledge about.

Effectively, reversing the edges drastically reduces the total number of parameters required to learn the network. For example, in this example, reversing the edges drastically reduces the total number of probabilities to be computed from 233,315 to 185 parameters. The reversal implies that the probability distribution of deflection (DEF) becomes marginal $P(DEF)$ (6-1=5 parameters) while the probability distributions of the simulation inputs now only receive one parent edge each, implying that they are now conditioned by only deflection ($P\ (DEF|C\_X)$, ($P\ (DEF|C\_Y)$, ($P\ (DEF|B\_pos)$, etc.) (6-1*6*6=180 parameters).

Therefore, as a rule of thumb, when the number of inputs is larger than the number of outputs, the directions of the edges should flow from outputs to inputs.

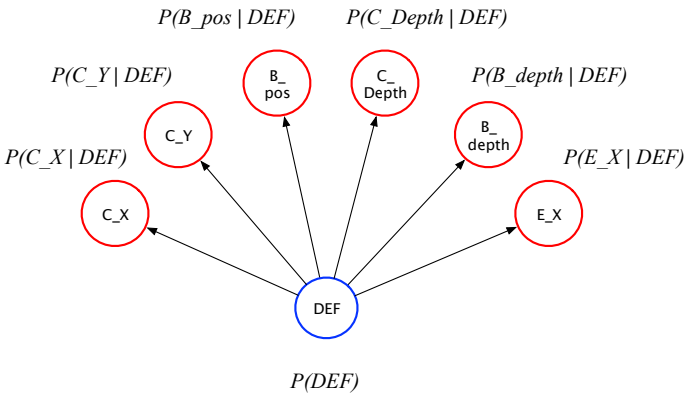


Figure 4.12: Reversed edges to reduce CPT size of total parameters required to learn BN.

A Bayesian network metamodel is not restricted to one output; we can add multiple outputs to the network. For example, in this case study we include the weight of the structure as a second output node in the metamodel (Figure 4.13). The consideration of multiple outputs implies that we can perform bi-directional inference between multiple inputs and outputs. Such a multi-input multi-output scenario is demonstrated further in a case study application described in Chapter 6.

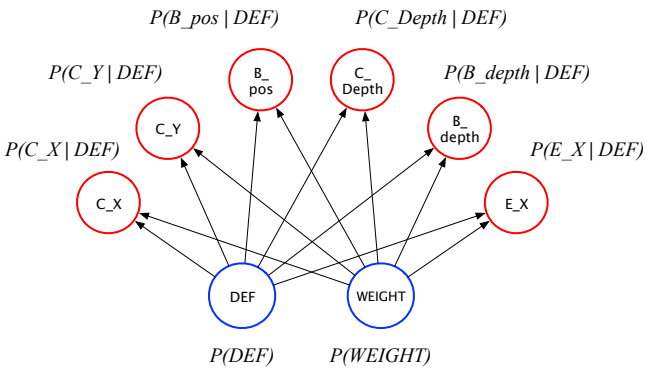


Figure 4.13: Multi-input, multi-output BN metamodel.

4.3.3 Learn conditional probability tables

The parameters in the CPT tables for each node were learnt directly from the input-output simulation data generated in 4.2, using the expectation-maximization (EM) algorithm. It is worth to note that a La Place smoothing algorithm was implemented as a means to avoid zero division error when learning parameters, caused by empty bins. In simple words, the algorithm adds a 'dummy' count to the empty bins.

4.4 Validate the Bayesian network metamodel

Once the BN metamodel is constructed, we want to estimate how well it can predict the output in comparison to the actual simulation model, when presented with the test data, which we put aside previously. In this section we restrict the definition of 'validation' to model accuracy. The case study in chapter 6 will serves as a validation of the BNM in a real project application.

4.4.1 K-fold cross-validation

A standard approach as part of validating a metamodel is to employ a cross-validation approach technique. Particularly, we adopt a k-fold cross-validation approach. K-fold cross-validation is a resampling procedure that involves splitting the input-output dataset into training/testing, in *k* different ways. In other words, the metamodel is built and tested *k* times, on *k* different training and testing. The factor *k* refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. This approach aims to reduce bias due to training/testing splitting of the dataset. In this research we adopt a *k* value of 10, which is often assumed as a default value, that has been found through experimentation to generally result in a model skill estimate with low bias a modest variance.

4.4.2 Measuring and illustrating prediction accuracy

Assessing the predictive performance of a model is an independent topic of research in the statistics and machine learning community; for example, see Bennett et al. (2013). The most typical method is the Root Mean Square Error (RMSE). This is defined by Eqn. (16), where $\hat{y}_t$ is the predicted value, y_t is the actual value generated from simulation and n is the number of samples in the test dataset.

$$RMSE = \sqrt{\frac{\sum(\hat{y}_t - y_t)^2}{n}} \tag{16}$$

However, since the output of a BNM is non-scalar, RMSE can only be used if a mean of the probability distribution is taken. Instead, we adopt a custom approach: i) using ***d1***—the difference between the mean of the predicted bin and the simulated value as shown in Figure 4.14 left, or (ii) ***d2*** — the difference between the mean of the predicted distribution bin and mean of the bin containing the simulated value as shown in Figure 4.14 right. ***d1*** can be considered as a 'stricter' validation, given that BNs essentially deal with a classification rather than prediction of a scalar value.

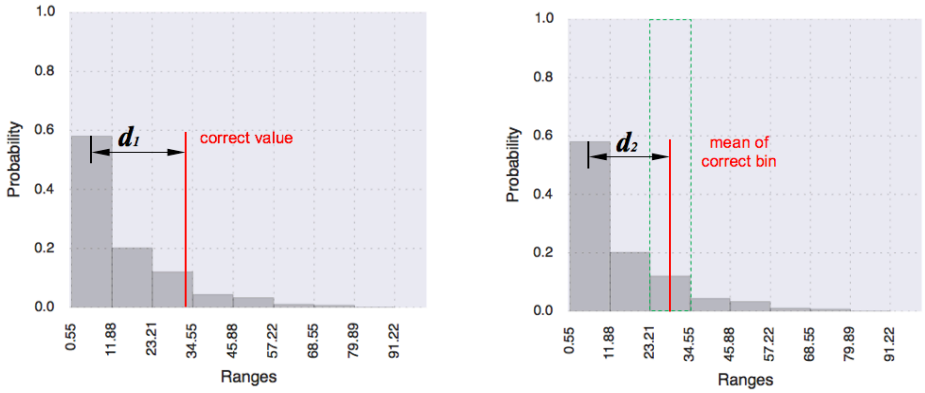


Figure 4.14: Left: difference ***d1*** between simulated value and the mean of the predicted bin. Right: difference ***d2*** between means of bin containing simulated value and predicted bin.

The resulting validation plot distributions in Figure 4.15 and Figure 4.16 provide an intuitive indication of how well the model predicts the numerical responses. ***d1*** plots for the remaining 9 folds of max deflection target and weight target, can be found in Appendix 7.3.4A.1. Overall, when observing the distribution of the d1 plots for all 10 folds, it can be noted that the BNM built on 4000 data points, seems to predict the correct response with an average 85% accuracy, for both targets.

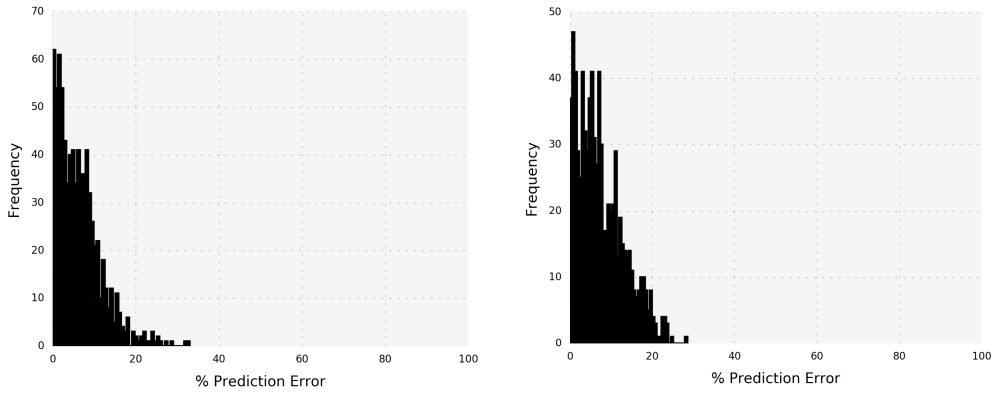


Figure 4.15: Histogram of ***d1*** values, from first the fold of the max deflection target (left) and weight target (right).

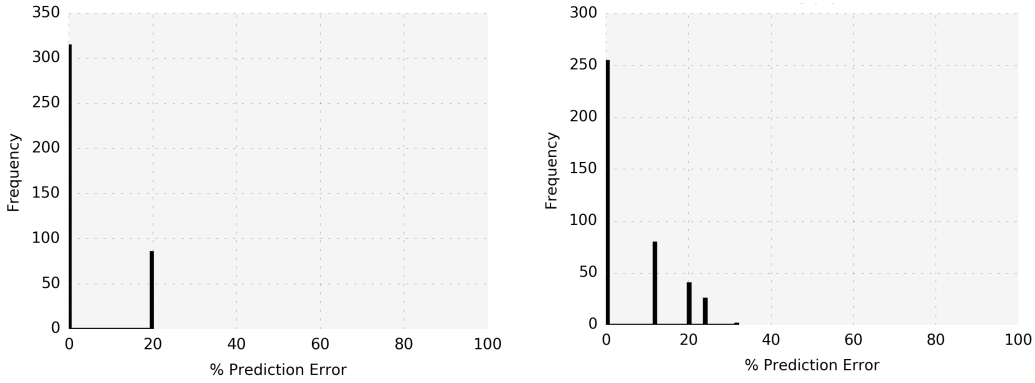


Figure 4.16: Histogram of ***d2*** values, from the first fold of the max deflection target (left) and weight target (right).

4.4.3 Effect of testing/training split, sampling method and sampling size

For this study we validate the prediction quality of the Bayesian network metamodel by building a series of BNMs for i) three sampling methods and ii) for six different sample sizes. The purpose is to understand the effect of i and ii on the quality of the model, and thus, identify a suitable sampling method and size. We consider the RMSE as an overall prediction accuracy measure. The RMSE is a value given in original units however since our scope was to compare the accuracy between different data sizes, it was important to normalise the RMSE value (NRMSE). Furthermore, since the RMSE is a scalar value, we assume the expected value of the probability distribution of the predicted maximum deflection as a weighted mean value by (17).

$$\hat{y}_t = E(DEF) \tag{17}$$

The NRMSE was estimated (i) for six different sample sizes, (ii) three different sampling methods, and (iii) three different training/test data split ratios. Figures 11, 12 and 13 visualise (i) and (ii) for three splitting ratios (70/30, 50/50, 30/70).

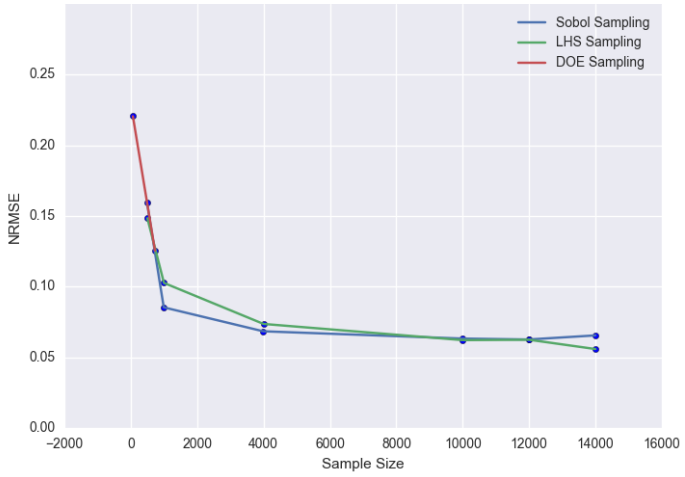


Figure 4.17: NRMSE vs. number of total data samples (training+test data). Split ratio 30/70.

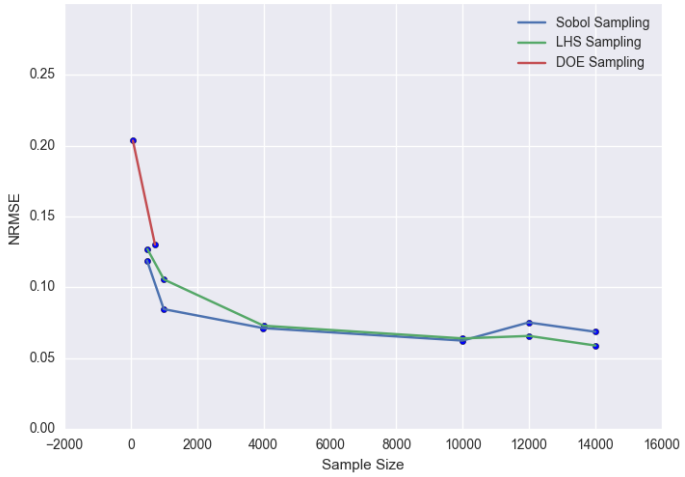


Figure 4.18: Normalised accuracy (NRMSE) vs number of total data samples (training+test data). Split ratio 50/50.

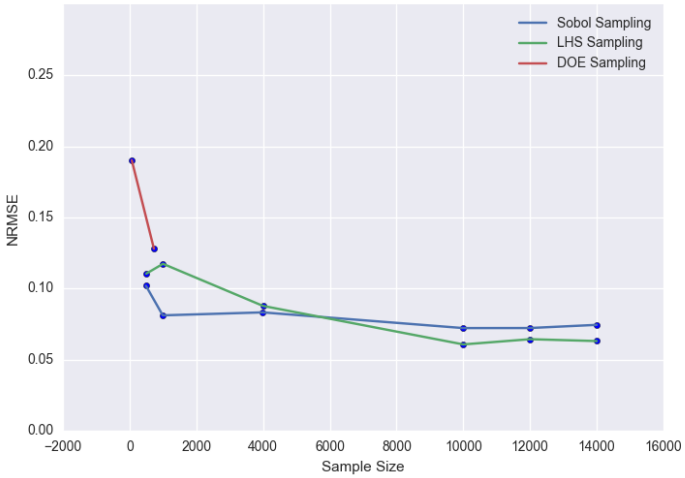


Figure 4.19: Normalised accuracy (NRMSE) vs number of total data samples (training+test data). Split ratio 70/30.

The following overall observations were inferred from the resulting plots in Figure 4.17, Figure 4.18 and Figure 4.19:

a) **Sample sizes**: The NRMSE decreases as the number of samples increases, meaning that a larger training sample, implies a more accurate prediction. The NMRSE seems to converge to around 90% ~ 95% when the sample size is 4000, out of the six sample sizes. There is no reliable rule of thumb to determine the sample size.

   The sample size is directly related to the number of input parameters being considered. Therefore, gauging the amount of data required based on the number if input parameters being considered might lead to the right direction. Furthermore, any knowledge about the linearity/nonlinearity of the problem, if available, indicates whether more data might be required to capture important characteristics. In future work (section 7.3) we intend to make use of dimension reduction methods to reduce the number of input parameters and thus, the number of samples required.

b) **Sampling method**: The sampling method does not seem to have a significant effect on the prediction accuracy of the BN. There does not seem to be a preferential trend between Sobol and LHS sampling as their overall trend seems to be similar, that is, NRMSE error is reduced as the sample size increases. This is partly due to the fact that BNs fit a joint probability distribution of all points rather than a function within a set of points. As for DOE,

c) **Split ratios**: The split ratio between training and test data seems to have an overall effect on the smaller sample sizes because the training data set becomes very small at 30/70. 70/30 split seems to be a better ratio for this problem.

The trade-off lies between simulation data intensity and the prediction quality of the Bayesian network metamodel. From the above analyses, we conclude that for such a problem, a sample size of 4000 using Sobol method to select input values secures a reliable prediction accuracy. Of course, the statistical quality of the model is highly problem-dependant and thus, may vary with each scenario. It is therefore recommended to gain some preliminary insight into the overall trend of the design space, prior to any further data-intensive steps.

Figure 4.20 illustrates a step by step algorithm to build and cross-validate a Bayesian network metamodel (BNM).

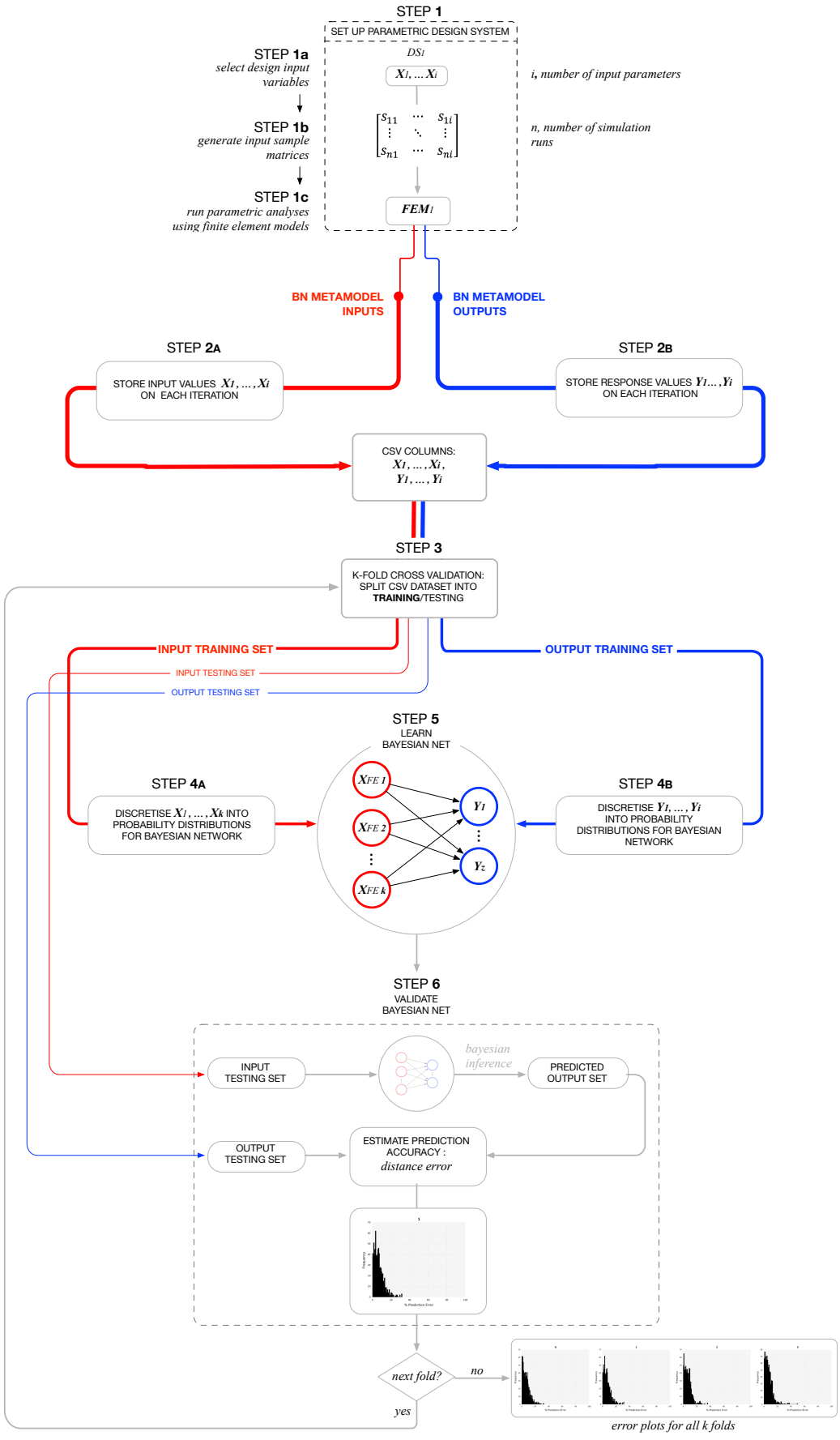


Figure 4.20: Detailed BNM build and validation algorithm.

4.5 Probabilistic Inference

Once built, the Bayesian network is a representation of the JPD between the input geometrical variables, the maximum deflection of the cantilever structure and its weight. We can now manipulate the probability distributions using inference, to metaphorically dissect the probabilistic relationships composing the JPD. The probabilistic representation allows us to use the metamodel as both a forward and inverse model thus, gain insight into the cause and effect relationships between the inputs and the outputs. Therefore, in the cantilever roof structure example, we can gain insight into what drives the maximum deflection of the structure and thus, how can we improve the geometry to minimise the deflection and the total weight of the structure.

4.5.1 Algorithms

There are two types of algorithms for computing Bayesian inference; exact inference and approximate inference. The former is used when the size of the conditional probability tables (model parameters) are manageable. The latter is an approximate method to compute inference over a large number of model parameters. Approximate inference works by sampling the joint probability distribution using a method such as Gibbs sampling algorithm (Geman & Geman, 1987). In this dissertation, we make use of exact inference given the manageable number of variables dealt with. However, in future work we intend to implement approximate inference into our BNM workflow.

4.5.2 Setting evidence

Performing inference with the metamodel is as simple as setting hard probabilities to the bin or bins of the random variable or variables of interest, and subsequently, observing how the other distributions in the network are updated based on the network of relationships. In the Bayesian networks community, the act of setting hard values to a probability distribution is referred to as "setting evidence". In the following plots, the bins highlighted in green indicate the evidence bins, the updated bins are indicated in red, and the dark grey bins in the background indicate the marginal probability distribution before any evidence or updates occurred. A comparison between the red and dark grey bins provides an intuitive measure of sensitivity pertaining to the variable.

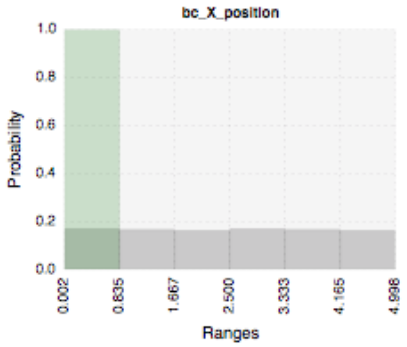


Figure 4.21: Example of setting evidence on the probability distribution of a random variable.

By default, we assume evidence to be in the form of 100% probability. However, it is also possible to specify 'soft' evidence where instead of setting 100% to one bin, multiple bins can be assigned with different probabilities as evidence. For instance, in Figure 15, we assign bins 1, 2 and 3 with 59%, 20% and 24% probability, respectively.

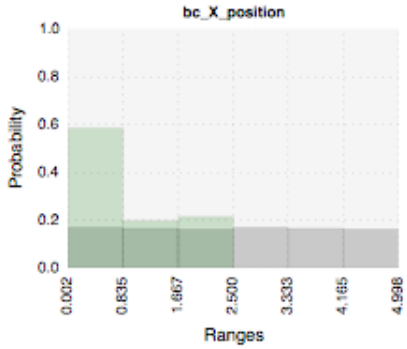


Figure 4.22: Example of setting soft evidence on the probability distribution of a random variable.

4.5.3 Bi-directional inference

Figure 14 illustrates the Bayesian network metamodel used in a forward scenario (inputs→output), where the metamodel is used to predict the response distributions of maximum deflection and weight (highlighted in red), for the ranges of the selected bins (highlighted in green). The response distributions can either be interpreted as a probability distribution of likely response values or as a mean value.

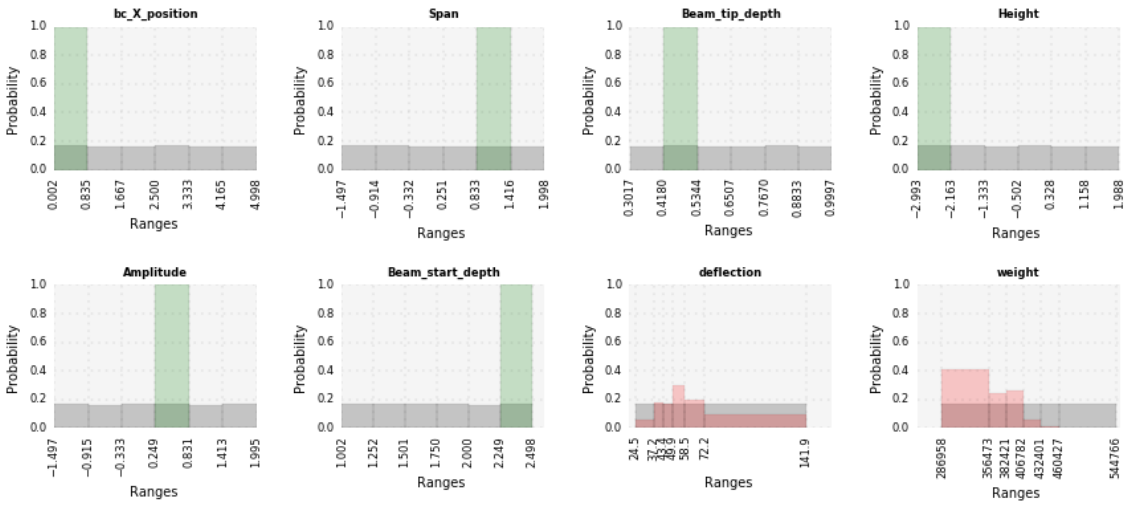


Figure 4.23: Resulting posterior output distributions (red) for the selected input bins (green).

The bi-directionality of a BNM also allows the inverse (inputs→output), that is, computing the input probability distributions for a specified output distribution. For example, in Figure 4.24 we set 100% evidence to the bin with the lowest maximum deflection values. Through Bayesian inference, the input probability distributions are updated based on their cause-effect relationship with the output. From the resulting probability distributions, we can note two forms of insight: 1) immediately note the range of input values to achieve structural geometries whose maximum deflection is guaranteed to fall between the range of the first bin, and 2) the sensitivity of the inputs to minimise the output. The latter is noted by observing the shape of the updated distribution, such as that of the *Span* and *beam_start_depth* input variables. As for the other inputs, their shape is more uniform, implying that the values along their ranges have a similar probability of achieving maximum deflection within the first bin – this translates into a sensitivity.

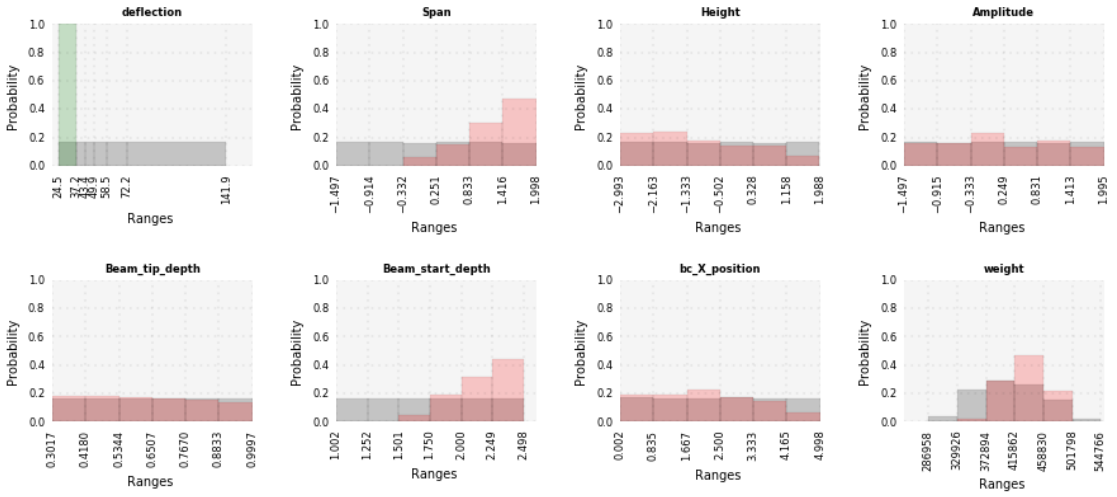


Figure 4.24: Resulting posterior input distributions (red) for the selected output bins (green).

Furthermore, we can also set evidence on multiple outputs; for example, to infer the input probability distributions that maintain a maximum deflection and weight within their first bins. The updated probability distributions indicate structural configurations likely to yield the queried values of weight and maximum deflection.

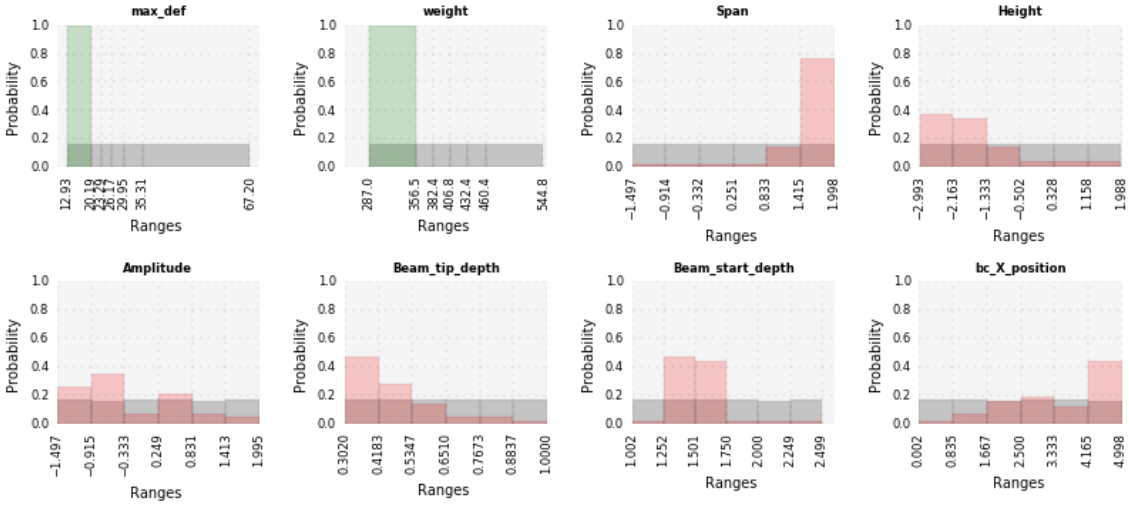


Figure 4.25: Resulting posterior input distributions (red) for the selected output bins (green).

Through the probabilistic representation, knowledge-insight about the cause-effect relationships is gained while the flexibility of the design parameters is maintained.

The notion of using probabilistic inference with a Bayesian network metamodel for knowledge-inference is illustrated by means of a realistic case study project in Chapter 6.

Chapter 5

SOFTWARE PACKAGE

The Bayesian network Metamodeling (BNM) workflow was developed by the author into a Python library package and a Rhino Grasshopper plug-in. This chapter explains how each of these packages were built and breaks-down the functionality of each of their components. In some instances, the following sections will make use of the Nervi cantilever mini case study discussed in the previous chapter for illustrative purposes.

5.1 Python library package: `bnmetamodel`

`bnmetamodel` is a Python library package that facilitates the building of Bayesian network metamodels (BNM), directly from input-output simulation data. The library is dependant on two existing Python libraries for building Bayesian Belief Networks: `libpgm` (CyberPoint International, 2012) and `PyBBN` (Vang, 2017). These libraries are widely used by machine learning experts and exist as open-source. The `bnmetamodel` package was written in an object-oriented format and is composed of two main classes: one that handles the data, and one that builds the actual Bayesian network. A third class, acts an overall wrapper to facilitate easy-building of BNMs requiring only two arguments (discussed in 5.1.3).

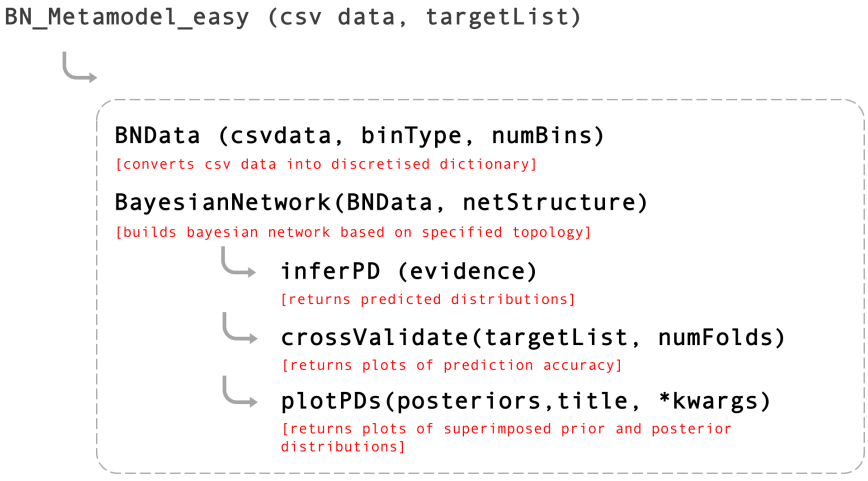


Figure 5.1: Object-oriented hierarchy of the Python package.

5.1.1 Data handling

Class name: `BNdata`

Arguments required:

Argument	Description
`csvdata`	A string indicating the file path to input-output simulation data. Can also be supplied as a Pandas DataFrame[1].
`binType`	A dictionary whose values specify the discretisation type for each variable. If equal binning specify '*e*' and if by percentile, specify '*s*'.
`numBins`	A dictionary whose values specify the number of bins specifically for each variable

Table 4: Arguments required for `BNdata`.

This class takes the input-output CSV file and discretises all variables as required for building the Bayesian network. An instance of `BNdata` returns a dictionary of 'bin counts' corresponding to the discretised inputs and outputs. The discretisation algorithm discretises each variable based on the discretisation type, specified in `binType` dictionary, and on the number of bins specified in `numBins` dictionary.

5.1.2 Bayesian network building

Class name: `BayesianNetwork`

Arguments required:

Argument	Description
`BNdata`	File path to input-output simulation data. Can also be supplied as a dataframe.
`netStructure`	A dictionary specifying the directed-edge connectivity between the variables (nodes). File path to a stored dictionary may also be supplied.

Table 5: Arguments required for `BayesianNetwork`.

This class builds a Bayesian network whose input-output topology is specified by `netStructure`, while the conditional probability tables of each node (model parameter) are

[1] A *dataframe* is a popular data matrix type within *pandas*, which is a popular library in the Python programming language.

learned directly from the discretised data returned by `BNdata`. If `netStructure` is not specified, a default network topology is specified automatically based on the ratio between inputs and outputs. A number of functions can be accessed via `BayesianNetwork`, mainly: to validate the metamodel, for inference, and subsequently to plot the outcomes. Table 6 describes each of these functions and their required arguments.

Class functions:

Function	**Arguments required**	**Description**
`crossValidate`	This function cross-validates the Bayesian network in *k* number of folds.	
	`targetList`	An array with target response names.
	`numFolds`	Number of *k*.
`inferPD`	This function calculates posterior probability distributions for supplied evidence.	
	`query`	A dictionary with query variables names.
	`evidence`	A dictionary with evidence variables names.
`plotPDs`	This function plots the input and output probability prior and inferred posterior distributions.	
	`posteriors`	Output from `inferPD`.
	`title`	Main title of plot.
	`kwargs*`	

Table 6: Functions available to `BayesianNetwork`.

Once the Bayesian network is built, we can use the `crossValidate` function to perform a *k*-fold cross-validation of the BNM, which will return a set of plots, for each fold. The `crossValidate` function uses the `kFold` splitting algorithms from `sklearn` Python library to generate the *k* number of training/testing splits. Subsequently, we can use `inferPD` function to reason and make queries with the network, by setting evidence and updating the probabilities. Currently, `inferPD` adopts exact inference either using a variable elimination algorithm from the `libpgm` library, or the junction-tree algorithm from the `PyBBN` library. In subsequent versions we intend to implement approximate inference algorithms, to handle inference over larger numbers of model parameters. The `plotPD` function returns a series of

plots illustrating superimposed prior and posteriors probability distributions as illustrated in the implementation example in 5.1.4.1. The superimposition helps with comparing how a distribution has changed from its marginal state to its posterior state when using Bayesian inference to update the network. The evidence bins are shaded in green, while the posterior bins are shaded in red.

5.1.3 BNM wrapper

Class name: `BN_Metamodel_easy`

Arguments required:

Argument	Description
`csvdata`	file path to input-output simulation data. Can also be supplied as a dataframe.
`targetList`	a dictionary whose values specify the discretisation type for each variable. If equal binning specify '`e`' and if by percentile, specify '`s`'.

Table 7: Arguments required for `BN_Metamodel_easy`.

This class builds a BNM by supplying simply a file path to the simulation data and specifying the targets outputs. By default, the class assumes 6 number of bins for each variable and assumes the inputs to be discretised using equal distance binning while the outputs using percentile binning. The structure is specified automatically by an algorithm that determines the edge direction based on the number of inputs vs. outputs, as a means to reduce the number of model parameters, as discussed previously in section 4.3.2.

5.1.4 Implementation example

The following implementation of the above-detailed steps makes use of the Nervi structural engineering example introduced in Chapter 4 as a means to motivate this implementation example. Section 5.1.4.1 demonstrates a typical implementation of `bnmetamodel`, while section 5.1.4.2 illustrates the implementation of a `bnmetmodel` using the `BNMetamodel_easy` wrapper.

5.1.4.1 Typical implementation

The following code excerpt demonstrates how to implement the explained Python package explained to build a BNM for the mini case study introduced in section 4.1. Figure 5.2 illustrates the resulting plot when using the PlotPD function.

```
import bnmetamodel
from bnmetamodel import BNdata
from bnmetamodel import BayesianNetwork

### specify csv file path
csvfilepath = '/Users/Desktop/metamodel/Nervi_CSV_4000.csv'

### specify net toplogy file path
bn_structure_filepath = '/Users/Desktop/metamodel/Nervi_structure.txt'

### specify type of discretisation method per variable
binTypeDict = {'deflection': 'p' ,'weight': 'p' , 'Height':'e',
'Amplitude':'e', 'Beam_tip_depth':'e', 'Beam_start_depth':'e',
'bc_X_position':'e', 'Span':'e'}

### specify number of bins per variable
numBinsDict = {'deflection': 6, 'weight': 6, 'Height':6,
'Amplitude':6, 'Beam_tip_depth':6, 'Beam_start_depth':6,
'bc_X_position':6, 'Span':6}

### specify which variables are response targets
targets = ['deflection', 'weight']

### discretise data using BNdata
data = BNdata.BNdata(csvdata=csvfilepath, targetlist=targets,
binTypeDict=binTypeDict,numBinsDict=numBinsDict)

### build Bayesian Network
bn = BayesianNetwork.BayesianNetwork(data, bn_structure_filepath)

### cross-validate Bayesian network metamodel
bn.crossValidate(targets ,10)

### specify evidence of interest
evidence = {'deflection':[1.0, 0.0, 0.0, 0.0, 0.0, 0.0],
'weight':[1.0, 0.0, 0.0, 0.0, 0.0, 0.0] }

### update all probability distributions using inference
based on specified evidence
posteriors = bn.inferPD(evidence)

### plot pds
bn.plotPDs(xlabel='Ranges ', ylabel='Probability',maintitle='Posterior
Distributions',displayplt=True,posteriorPD=posteriors,
evidence=evidence.keys())
```

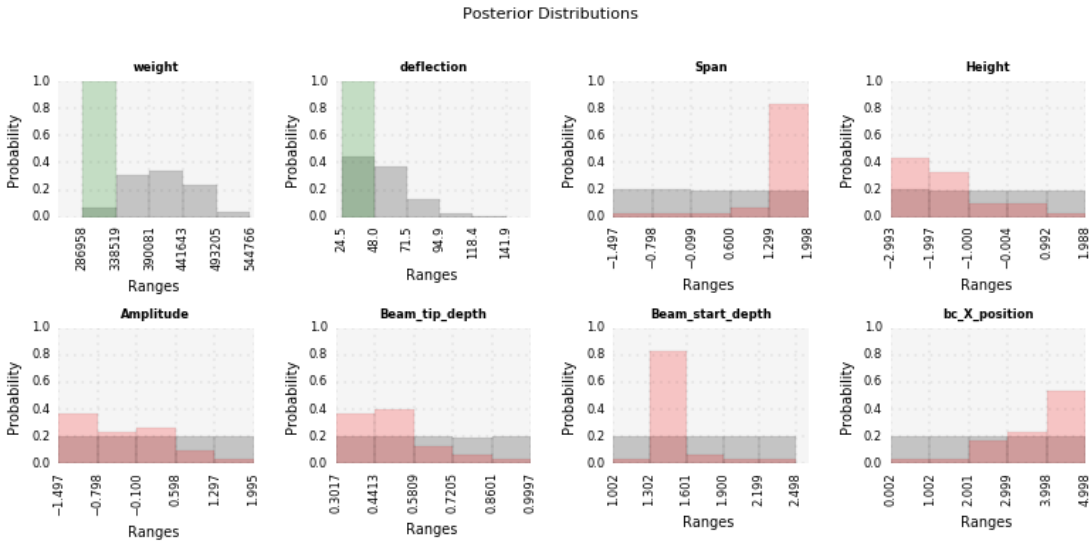


Figure 5.2: Automatic plot of the prior and posteriors distributions returned by the plotPD function.

5.1.4.2 Easy implementation

Alternatively, the following facilitates an easier implementation by making use of the `BN_Metamodel_easy` class to build a Bayesian network metamodel requiring only the CSV data and the list of response targets. The last line of code returns the same plot as the one illustrated in Figure 5.2. The wrapper makes the following assumptions: all inputs and output are discretised into 6 bins; inputs are discretised equidistantly, while outputs are discretised using percentile binning.

```
### Point to csvfilepath
csvfilepath = '/Users/Desktop/metamodel/Nervi_CSV_4000.csv'

### specify which variables are response targets
targets = ['deflection', 'weight']

### Instantiate a bnmetamodel wrapper – specify targets
b = BN_Metamodel_easy.BN_Metamodel_easy(csvfilepath, targets)

### Construct the bnmetamodel
bn = b.generate()

### Specify scenario of interest
evidence = {'deflection':[1.0, 0.0, 0.0, 0.0, 0.0], 'weight':[1.0, 0.0, 0.0,
0.0, 0.0] }

### Perform inference to 'update' probability distribution (Bayesian
inference)
posteriors = bn.inferPD(evidence)

### Plot posterior probability distributions
bn.plotPDs(xlabel='Ranges ', ylabel='Probability',maintitle='Posterior
Distributions',displayplt=True,
posteriorPD=posteriors,evidence=evidence.keys())
```

5.2 Rhino Grasshopper plug-in

The Rhino Grasshopper plug-in serves as a user-interface to the Python package discussed in 5.1. Figure 5.3 illustrates an overview of the components built into the plug-in. These include PSlider, POutput, DataGenerator, and the ModelBuilder. The following sections discuss each in further detail and finally, we demonstrate how the components work together to build a BNM in a parametric workflow, and subsequently how to use the BNM to exercise inference.

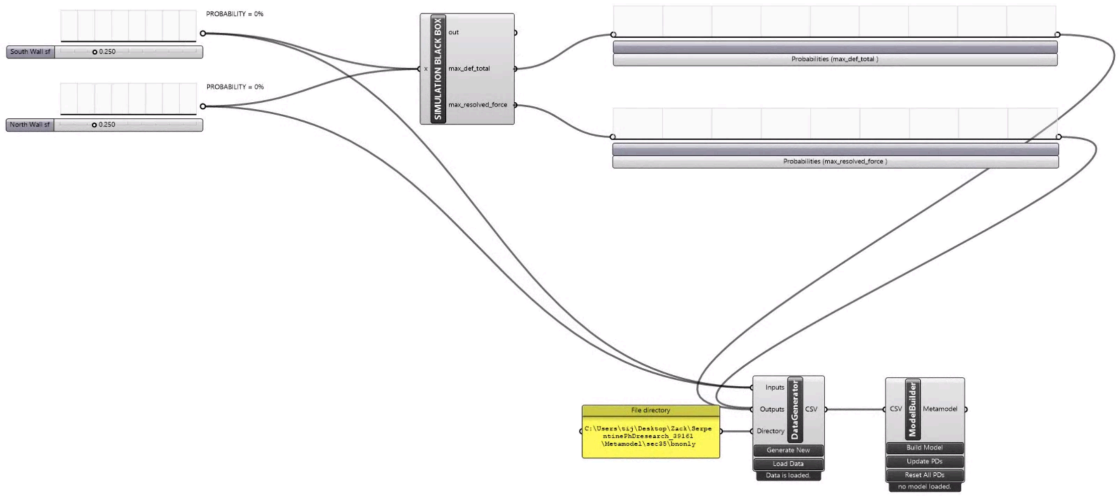


Figure 5.3: Overall view of BNM setup in a parametric environment.

5.2.1 Probabilistic input component: PSlider

The probabilistic input component, aka 'PSlider' component, represents an input random variable. The component takes the form of a typical Grasshopper number slider, most users are familiar with, and is extended to illustrate a probabilistic distribution along the range of the slider. In other words, the intention here is to illustrate an input random variable using the language most users are already familiar with. The probability distribution is populated remotely, via the 'ModelBuilder' component, once data is generated. The probability of the value set by the slider grip is displayed on the upper right corner of the component. In other words, the probability displays the probability of the corresponding bin above the slider knob position. Thus, as one slides the knob, the probability is displayed in real-time. The 'PSlider' has the same functionality as a normal slider and can be used in a typical fashion however it is a required input to the 'DataGenerator' component discussed in 5.2.3.

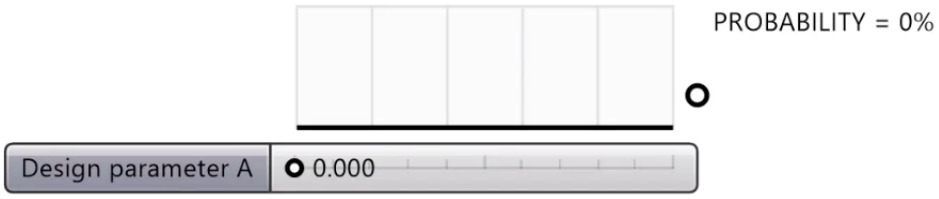


Figure 5.4: PSlider component.

The probability distributions bins are interactive such that a user may double click to set a bin to 100% inference. This is used during inference as demonstrated in 5.2.5.

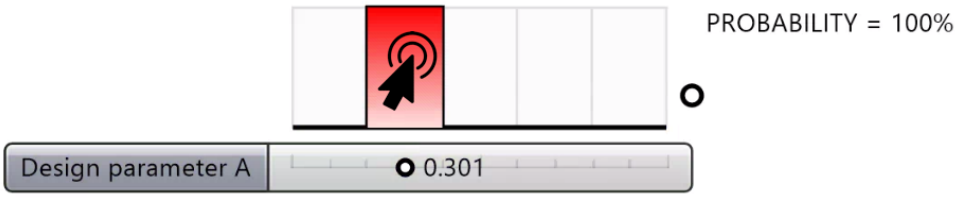


Figure 5.5: PSlider component: interactive bins to set evidence.

Alternatively, a custom "soft" probability distribution can be entered by right-clicking the component and entering a custom probability distribution in the 'customPD' sub-menu. This is useful in situations where the input of interest is highly uncertain.

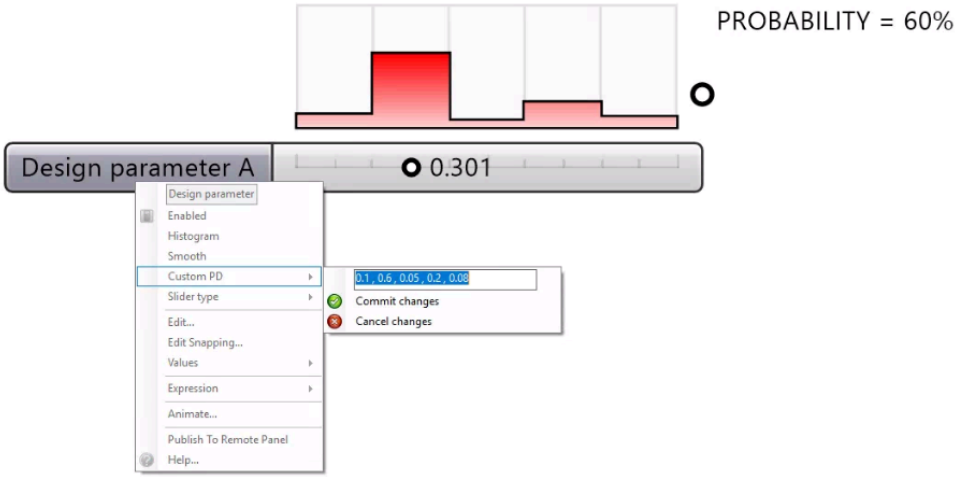


Figure 5.6: PSlider component: setting custom PD.

5.2.2 Probabilistic output component: POutput

The probabilistic simulation output component, nicknamed as the 'POutput' component takes the form of a typical Grasshopper parameter component, and extended to illustrate a probabilistic distribution. In other words, it is a way of illustrating an output random variable. The probability distribution is populated remotely, via the 'ModelBuilder' component, once data is generated. The interactivity of the bins is the same as described in the case of the 'PSlider'. The 'POutput' component requires a parameter input; that being the response output of the simulation module. The 'POutput' component automatically takes the name of the simulation output parameter plugged into it, as illustrated in Figure 5.7.

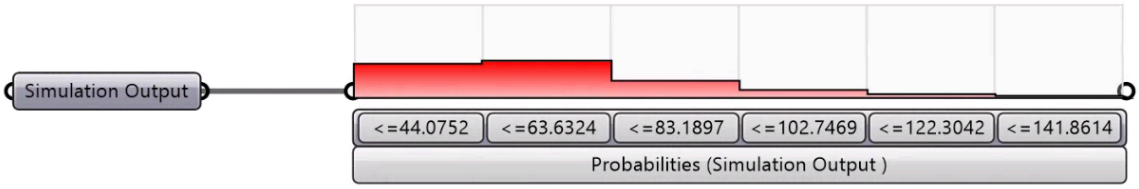


Figure 5.7: POutput component.

5.2.3 Data generator component: DataGenerator

The 'PSlider' and 'POutput' components are plugged into the 'DataGenerator' to generate the simulation data required for building the Bayesian network metamodel. The sampling algorithm and number of simulation runs may be specified within the component'.

If data is not available, the user may use the 'generate' button to generate new data. The component generates input samples within the ranges specified by each of the input 'PSliders', then runs the simulations for each combination of input samples and subsequently stores an input-output csv file to the specified directory.

Alternatively, if data is already available, a user may upload the CSV data file directly via the load button.

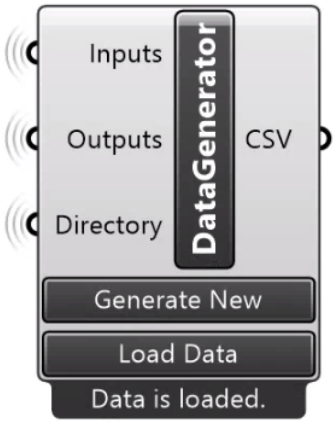


Figure 5.8: DataGenerator component.

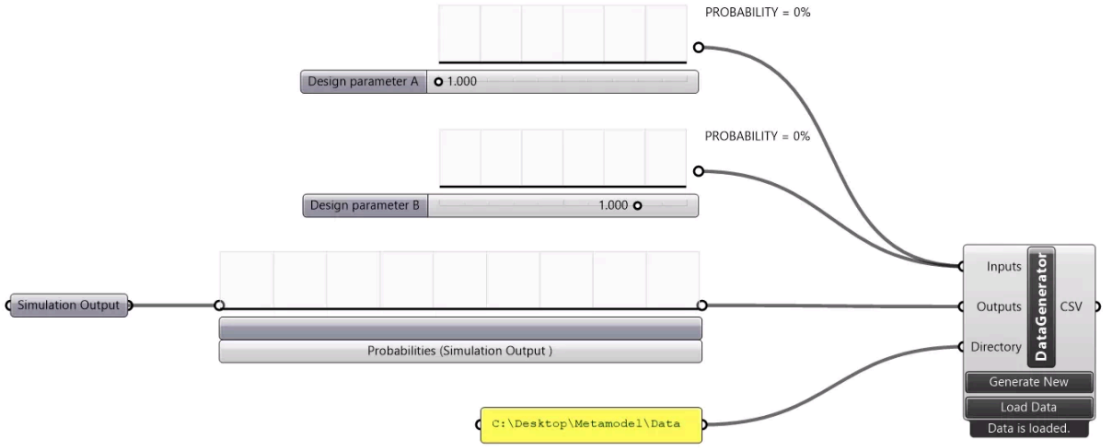


Figure 5.9: Inputs required by the DataGenerator component.

5.2.4 Model builder component: ModelBuilder

The ‘ModeBuilder’ component inputs the CSV data from the ‘DataGenerator’ component and subsequently handles the Bayesian network building process based on this CSV data. The component has three buttons: ‘Build Model’, ‘Update PDs’ and ‘Reset All PDs’.

‘Build Model’ builds the Bayesian network metamodel. ‘Update PDs’ searches the ‘PSliders’ and ‘POutputs’ plugged into the ‘DataGenerator’ for evidence and subsequently calls the inference function to update the probability distributions, remotely. The ‘Reset All PDs’ button resets all ‘PSlider’ and ‘POutput’ bins to their marginal probability values.

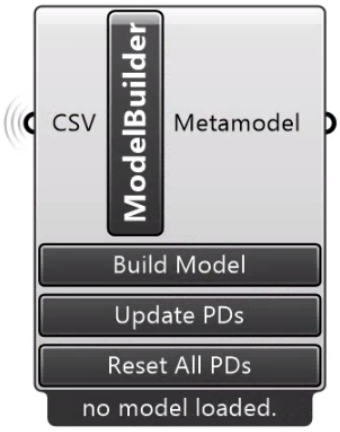


Figure 5.10: ModelBuilder component.

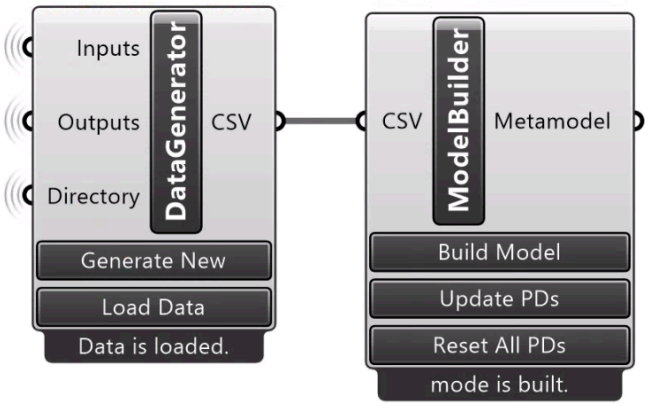


Figure 5.11: CSV input required by ModelBuilder component.

5.2.5 Use example

The following illustrates an implementation of the plug-in to build a Bayesian network metamodel for the case study example introduced in the earlier Chapter. Figure 5.12 illustrates the built metamodel, where 'PSliders' and 'POutputs' whose bins are populated with their marginal probability distributions, as frequency distributions of the input-output CSV data. More specifically, the input PSlider components illustrate the frequency distributions of samples inputs for each input parameter, while, the POutput components illustrate the frequency distributions of response values recorded on each simulation run.

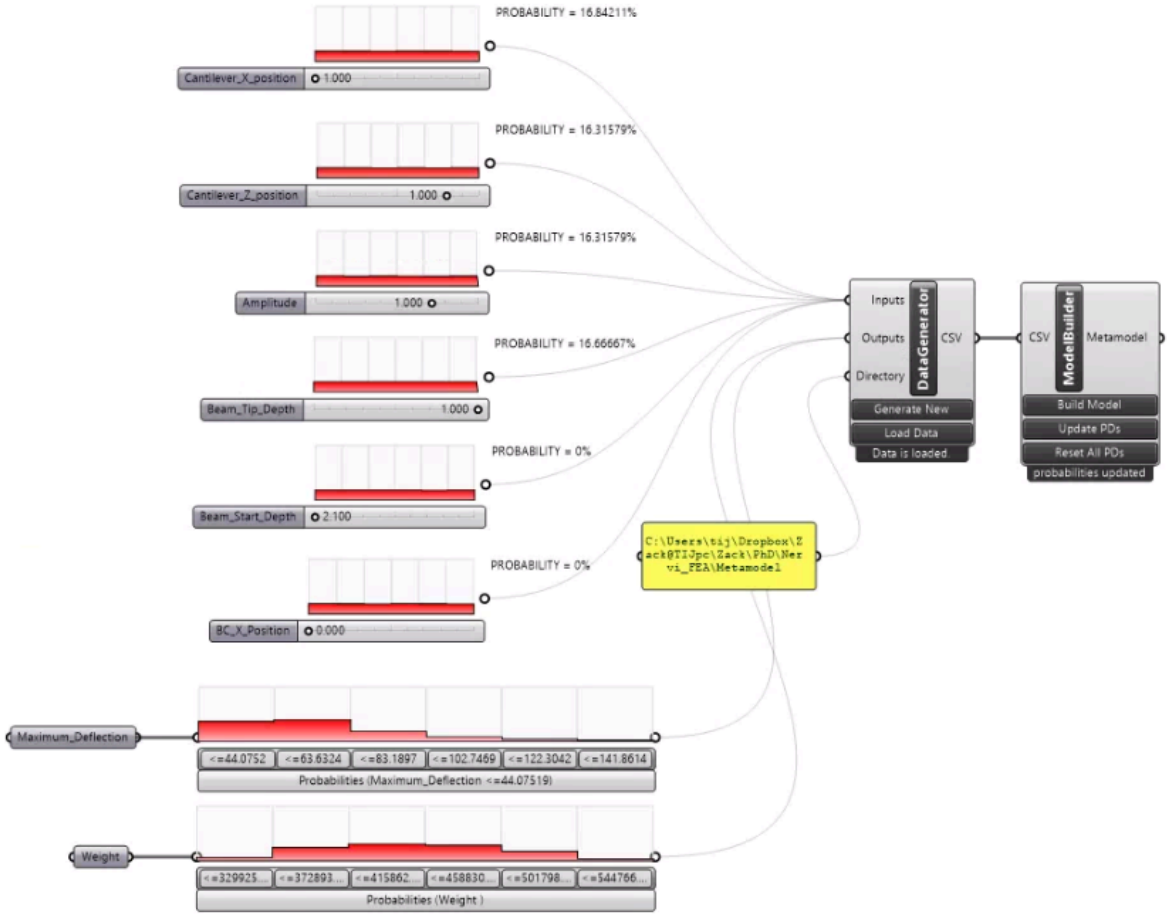


Figure 5.12: Overall implementation of plug-in components.

Figure 5.13 illustrates how to set evidence by double-clicking on a bin, for example, the bin with the lowest value in Maximum Deflection 'POutput'. Once evidence is set, clicking the 'UpdatePDs' button, updates all other 'PSlider' and POutput' probability distributions, via inference in the background. The inference mechanism occurs in the background via the Python package described in section 5.1.

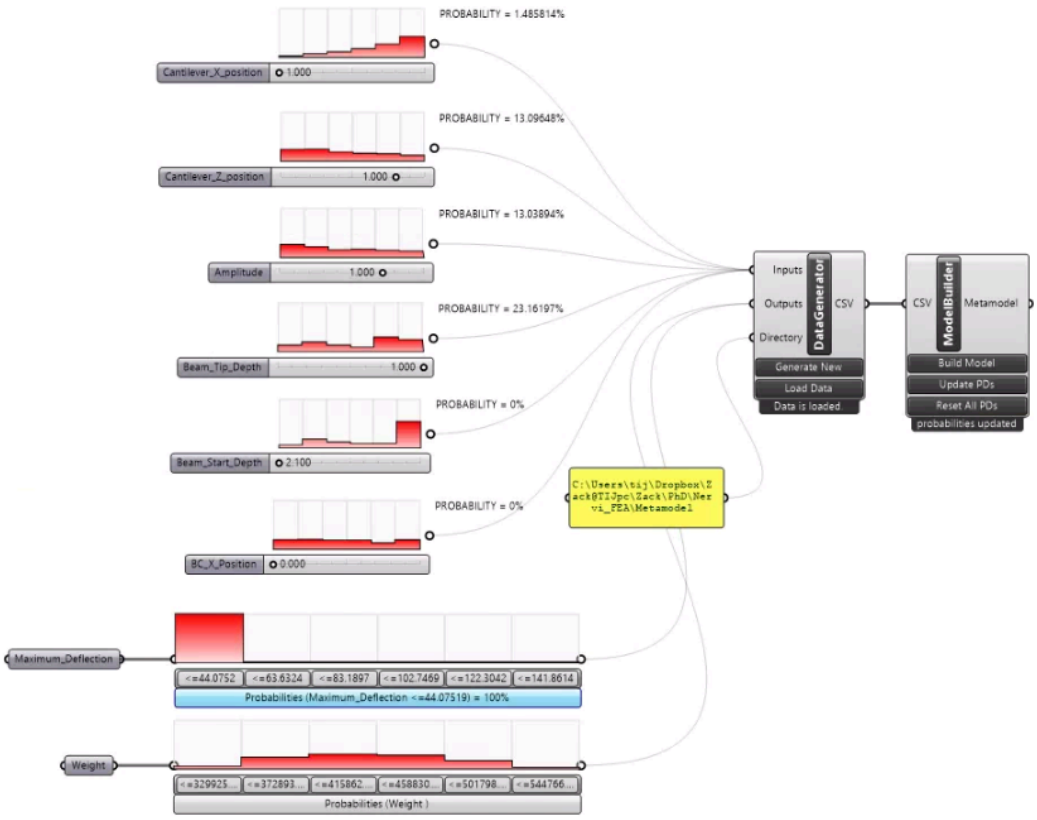


Figure 5.13: Setting evidence in the POutput and using inference to update the remaining probability distributions.

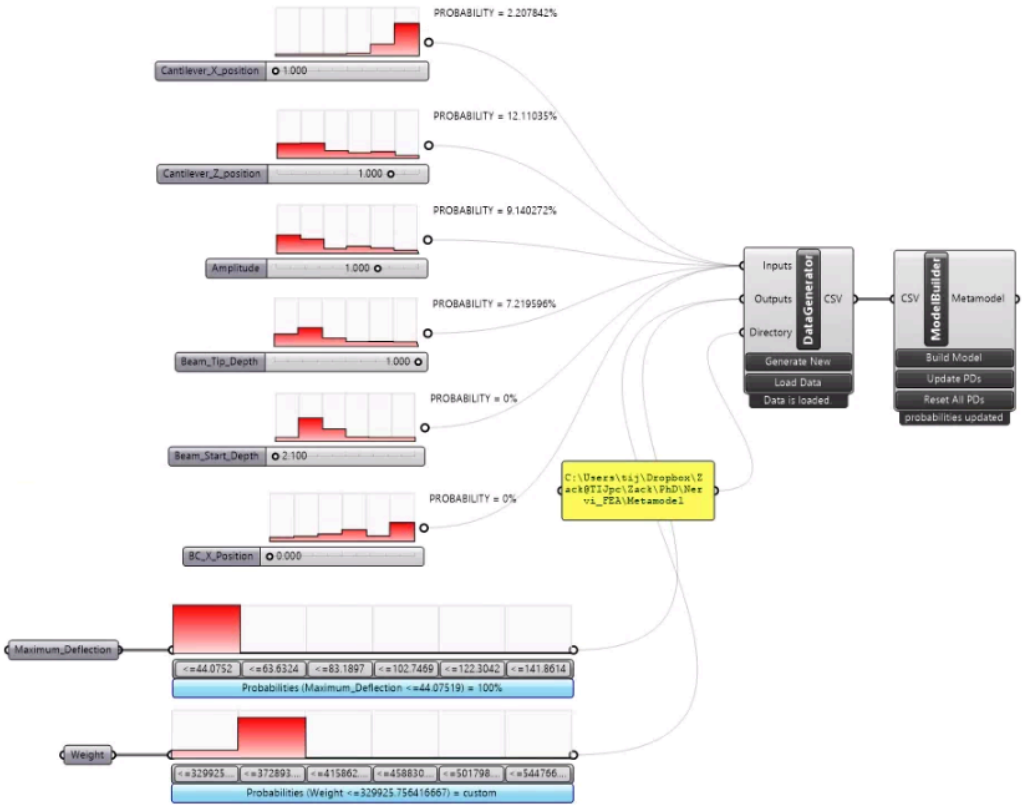


Figure 5.14: Setting hard and soft evidence on multiple POutputs.

Chapter 6

CASE STUDY APPLICATION

6.1 Scope of study

This chapter aims to test the validity of our research (BNM) by means of a retrospective study of a built case study project. A realistic project can reveal the usefulness of the proposed approach whilst highlight challenges that need further work. In this context, we have selected the 2016 Serpentine Pavilion in London as a project of choice because it provides an exemplary scenario of a real-world project where good communication at the architecture-engineering interface was critical to manage strict time constraints.

Figure 6.1: Render of the 2016 serpentine pavilion in London Design render © Bjarke Ingels Group (BIG).

6.2 Background

Each year, since 2000, the serpentine Gallery in London commissions a renowned architect to design a pavilion to be built and displayed during the summer months. The 2016 pavilion was designed by Bjarke Ingels Group (BIG) and engineered by AKT-II (Adams Kara Taylor).

6.2.1 Architectural composition

The pavilion concept consists mainly of two 30-metre-long sinusoidal walls; one concave, and the other convex. The two walls undulate towards one another starting at the bottom until they merge into each other vertically, at the top. Each wall is composed of overlapping, open-ended boxes that vary in length depending on the sinusoidal geometry. The box pattern on either wall is set in an inverse checkerboard pattern to its neighbour thus, allowing the cuboids from each wall to merge into a seamless grid at the top part of the pavilion. The stepping and staggering of these boxes at the ground level create "a 'pixelated' external landscape open to climbing and sitting, while inside BIG has taken the opportunity to sculpt a series of differently scaled spaces intended for seating, a bar and a live performances" (Kingman, Dudley, & Baptista, 2017). The open-ended boxes allow a certain 'play with opacity' of the pavilion's walls, when viewed from different angles; when viewed longitudinally, the boxes appear solid whereas when viewed from the perspective of a visitor walking through, they seem to "dematerialise" into a slender grid of lines, allowing views into the surrounding park.

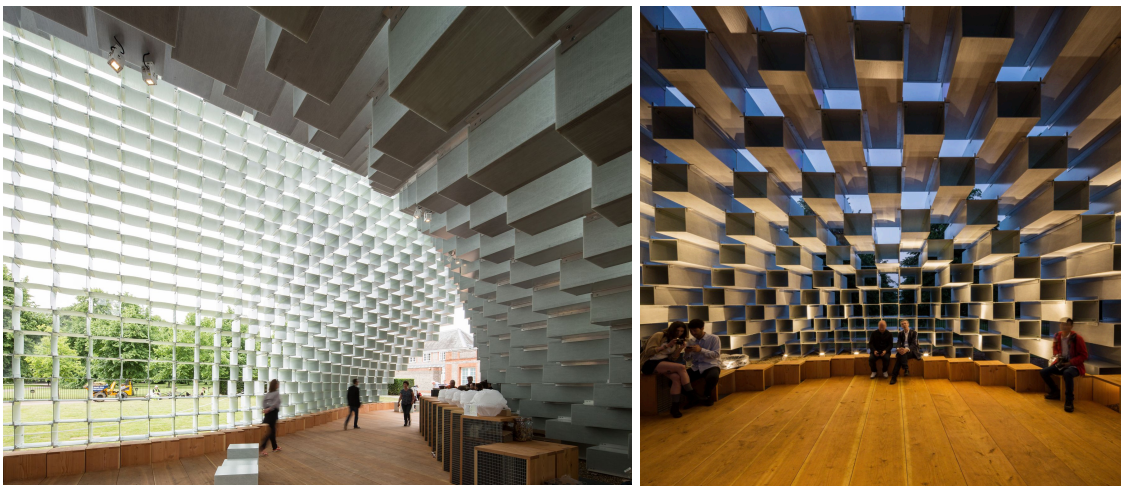


Figure 6.2: Internal shots of the pavilion in London (source: radu malasincu)

6.2.2 Material and structural considerations

Each box making up the geometry of the pavilion is composed of four glass fibre-reinforced plastic (GFRP) plates with GFRP angles glued in each corner to increase lateral stability and vertical loading-bearing capacity. From the start of the project, BIG emphasised they wanted to experiment with this material. GFRP is a composite material formed of a glass fibre encapsulated within a plastic resin matrix that typically has strength comparable to that of steel but with only

around a quarter of the weight. The material can be produced on a large scale with a high degree of consistency and at a low cost.

Subsequently, neighbouring GFRP boxes are connected by 10-mm thick cruciform-shaped aluminium that provided the necessary weight to strength ratio. Finally, a bespoke flat-headed bolt-and-sleeve was used to fix the GFRP angles, the GFRP plate and the aluminium cruciform together.

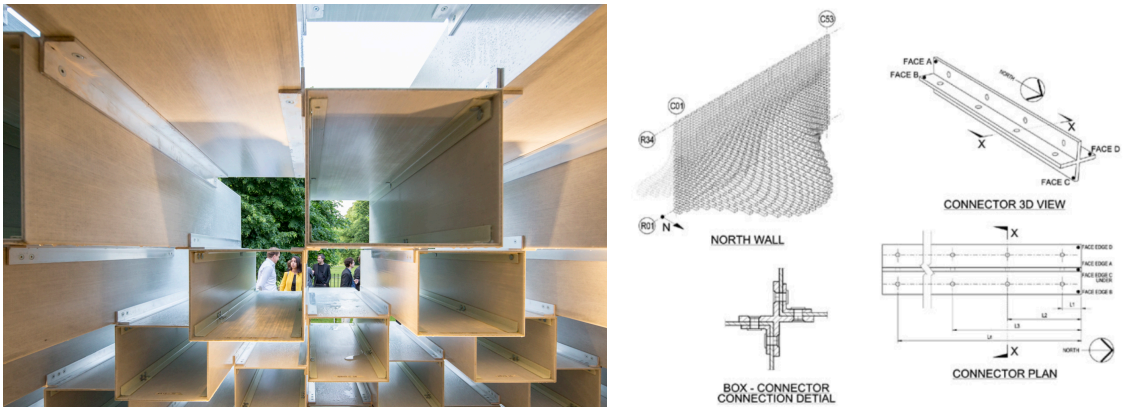


Figure 6.3: Connection detail (source: Laurian Ghinitoiu, AKT-II)

When assembled, the arrangement of the GFRP boxes making up the architectural language of the pavilion, also had to act as the primary load-bearing elements of the structure. In other words, any imposed loads had to be transferred to the ground via the boxes themselves and the connections and fixings between them. For this reason, observing the local behaviour of the boxes and their connections was critical.

AKT-II's role involved analysing the behaviours of the GFRP boxes at both local and global scales, under various load scenarios. This included high fidelity non-linear finite element analysis (FEA) mesh models of individual boxes, FEA of small clusters containing up to a dozen boxes. Furthermore, the engineering team engaged the material supplier to undertake a series of physical tests, since "GFRP is not widely recognised in its application to primary load-bearing structures, outside of highly specific and specialised applications" (Kingman, Dudley, Baptista, et al., 2017). The results were used to verify the digital models.

The outcomes of the digital and physical tests were used to calibrate the global 2D and 3D parametric frame model.

6.3 Parametric workflow

The time budget from concept design to coordinated production information limited to only three months. The strict time budget together with the structural intricacy of the concept required the engagement of engineering consultants from the very start of the project. For these reasons, both design teams at BIG and AKT-II decided it was necessary to adopt parametric approaches to describe the entire geometry.

6.3.1 Architecture: BIG

In more detail, the parametric developed by BIG focused mostly on form generation. This included: generating sinusoidal surfaces, projecting a grid onto it, and extruding the checkerboard pattern into the boxes at desired lengths, based on internal space configuration, etc. The overall sinewave geometry of either of the pavilion walls was dictated by the design team at BIG, based on which gave the necessary enclosed floor area underneath, while ensuring the maximum clear head-height spaces, etc. Once the sine wave form was fixed, the grid was then overlaid on this surface and boxes were extruded (both by BIG), and total lengths were generated to test with the manufacturer (see above). The size of the grid was fixed early on, based on the most cost-effective sheet widths available from the GFRP manufacturer. With these dimensions known, this size of each cell in the grid cell was fixed (but not the number of columns or rows in the grid).

6.3.2 Engineering: AKT-II

On the other hand, the parametric workflow developed by the engineering team at AKT was used to take in the geometry generated by the design team, and process it for analysis. In more detail, the Grasshopper definition takes in an assembly of 3D cuboid surfaces in Rhino as inputs (provided by the design team at BIG), finds the overlaps between the boxes, and calculates the appropriate connector typology (whether one, two or three pairs of bolts). Furthermore, the parametric model assigns the necessary GFRP sheet thickness (whether 10mm, 6mm or 3mm) based on local stiffness requirements. The automatic assignment of bolt fixings and sheet thickness was calibrated based on in-depth studies of local behaviour. This included high-fidelity analyses of assembled boxes and also physical testing of GFRP material as a sheet and when assembled. Once the connections and thickness are assigned, the definition then generates a series of 2D vertical sectional frame models along the longitudinal axis of the pavilion; one at each column of cells on the grid. These sectional frames are subsequently used as 2D frame analysis models. The models are analysed using finite element analysis with Grasshopper software plug-in Karamba (Preisinger & Heimrath, 2014) to generate a structural response. The response data

for each box in each of the sections was then used to generate false-colour dot distributions, showing where forces were too high (bottom left figure in Figure 6.4). These images were then handed back to the architecture team at BIG to indicate which box lengths required adjustment to achieve stability, based on the false-colour image.

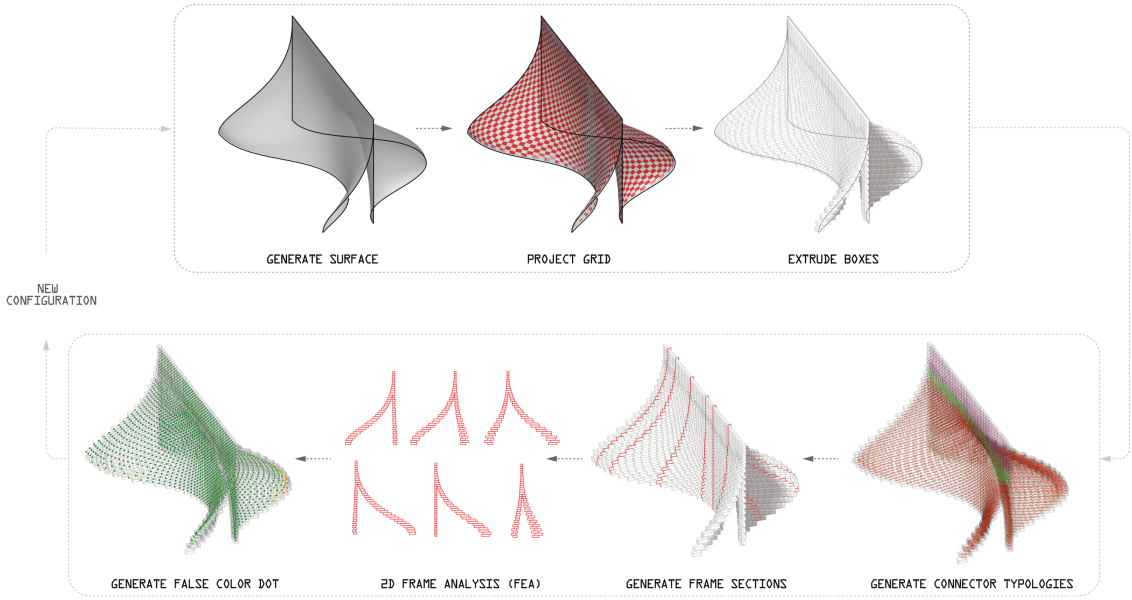


Figure 6.4: Overal parametric workflow between BIG and AKT-II.

For this case study we acquired the parametric model Grasshopper from AKT-II utilised during the project development and described above (6.3.2) to regenerate simulation data as required for constructing a Bayesian network metamodel.

6.4 Challenge addressed

Throughout the iterative design and analysis process, AKT's engineering role was to reduce global areas of high forces. On each design iteration, AKT would 'plug' BIG's generated design options into their parametric workflow, to generate cross-sectional frame analysis models. Each cross-section was analysed under different loading conditions. Ultimately a false colour image that was generated to communicate to the architecture team which areas need tweaking to reduce extreme axial and shear forces. Throughout this process, it was important for BIG to maintain control over the design intent and thus kept a manual hand between design and analysis.

In this case study we argue that instead of communicating discrete feedback about single design scenarios to the architecture team, the engineering team may use a BNM approach to suggest

constraints on the parameters thus, maintain the flexibility of the architecture team's parametric workflow. More specifically, instead of analysing one design scenario at a time, the engineering team could utilise a BNM to capture relationships between geometry and behaviour into a bi-directional model which could then be used to rapidly translate engineering constraints into architectural constraints on the parameters. In other words, identify input parameter ranges that are likely to generate forms that maintain the structural behaviour within a structurally feasible range. Subsequently, the engineering team would then suggest these parameter ranges to the architecture team thus, maintain the flexibility of the parametric model with the scope of reducing the number of back and forth iterations between the two teams.

In section 6.5 we demonstrate a retrospective study to show how engineers may build several "local" BNMs to understand the influence of the architectural parameters controlling the global geometry defined by architects, and local behaviour at different cross-sections along the pavilion. We then build a "global" BNM that takes into account the relationships between the global geometry and the behaviour of multiple local responses, simultaneously. Subsequently, in section 6.6, we describe how to use the BNMs built in section 6.5 to translate local engineering constraints into architectural constraints.

In the following sections, we illustrate this study by utilising the Grasshopper plug-in developed in this research (Chapter 5), to build BNMs within AKT's parametric model. This will also serve as a means to test and illustrate how engineers may visually communicate engineering feedback and constraints back to the architecture design team in qualitative terms of the design parameters, thus prolong the flexibility of the architecture team's workflow.

6.5 Bayesian network metamodel (BNM)

Let us assume that the architecture team hands over a parametric definition of the north and south pavilion wall geometry, over to the engineering team. In this retrospective scenario, we describe how the engineering team may use their existing parametric workflow to strategically generate simulation response data and use this data to build metamodels for learning about physical behaviour at different scales and for making inferences over multiple variables, and over flexible parameters (probability distributions), which would otherwise be very challenging to compute cognitively.

More specifically we will demonstrate how to build a multi-input multi-output BNM to capture relationships between geometry and multiple local responses and subsequently take advantage of

the bi-directionality to translate engineering thresholds imposed on the local responses, into constraints on the ranges of the architectural parameters controlling the north and south wall geometry.

For explanation's sake, in this case study we focus only on 6 cross-sections out of the 51 possible cross-sections in the grid.

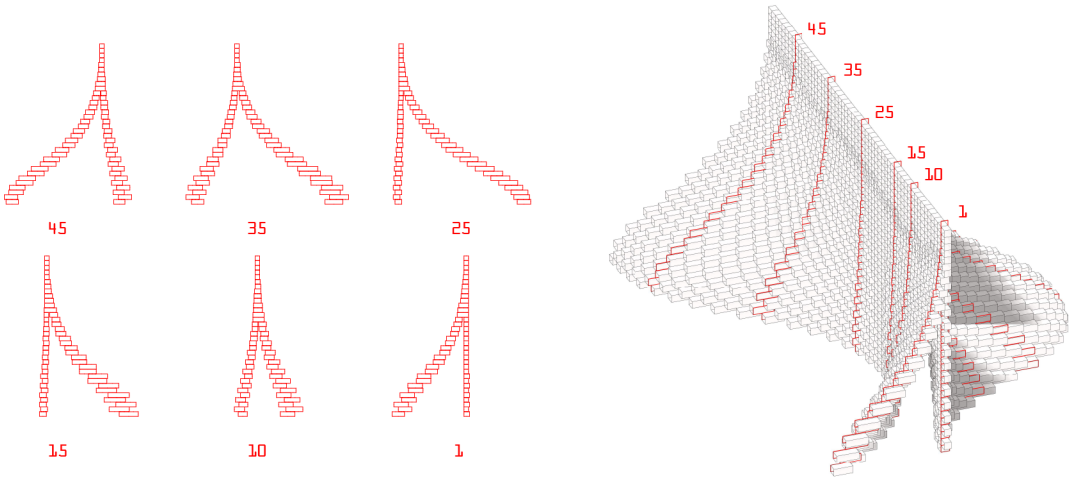


Figure 6.5: Selected cross-sections.

The following describes the three main steps involved with building the Bayesian network metamodels: Generate data (step 1), build Bayesian networks, and (step 2) validate the quality of each metamodel (step 3).

6.5.1 Data generation (step 1)

6.5.1.1 Selected parameters

Most parameters in BIG's workflow were fixed early on due to a combination of strict time budget for design exploration, and material volume budget limitations. These include the height and width of boxes, overall pavilion height and length. However, parameters controlling the bulginess of each wall were identified by AKT as critical factors influencing the forces in the boxes, and thus, were continually tweaked by the architecture team based on AKT's post-analysis feedback, until reasonable forces in the bolts connecting the boxes were achieved. It was important for BIG to maintain the design intent and thus, kept manual control over the tweaking while keeping in mind any extreme behaviour as indicated by AKT's false-colour dot images.

For this case study we describe the sinusoidal form of the walls by two parameters controlling the sinusoidal proportions of each respective wall. From here on, these parameters are labelled as North wall scale factor and South wall scale factor. A value 1 implies the as-built geometry proportions (as indicated in Figure 6.6). Table 8lists the minimum and maximum values bounding the ranges of each parameter.

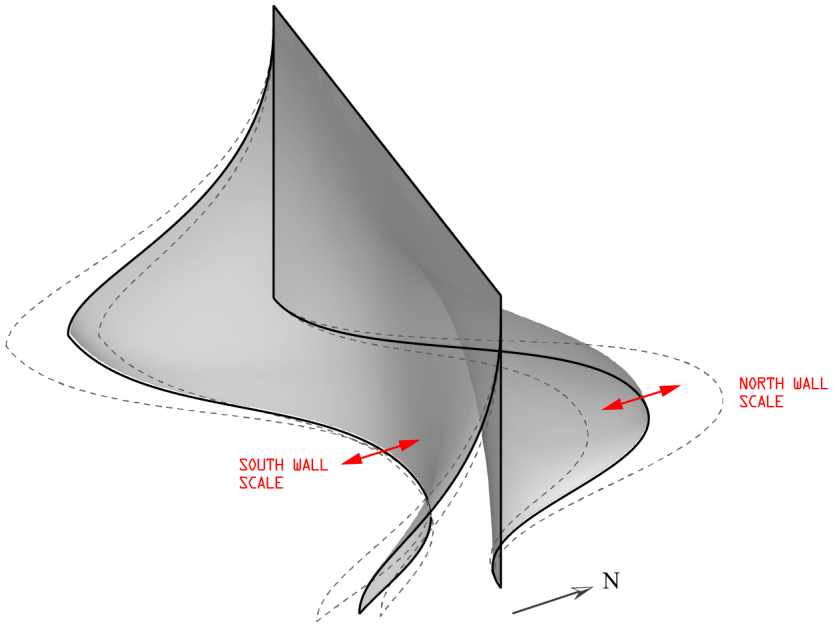


Figure 6.6: Parametric model. Illustration of parameters under study

Parameter	Notation	Max	Min
North Wall scale factor	*north_wall_sf*	0.75	1.3
South Wall scale factor	*south_wall_sf*	0.85	1.3

Table 8: Selected parameters for study and their ranges.

6.5.1.2 Selected experimental design method (sampling input space)

We assume we know nothing about the design space. Therefore, we intend to sample as many regions in the parameter space as possible. The scope is for the metamodel learning algorithm to generate a varied-enough response data set that captures as much information as possible about the relationships between the parameters and the responses. For this study, we generate 1000 input samples from the input parameter space using Sobol' sequences (Sobol', 1990), to generate quasi-random sequences of parameter values (Figure 6.7). Quasi-random sequences ensure that discrepancy between the sample points is kept somewhat low. In other words, the points are

distributed as homogenously as possible while retaining some level of randomness to avoid any systematic correlations between inputs. Each sample point implies a single simulation run. Thus, a total of 1000 simulation runs were computed with each cross-sectional analysis model.

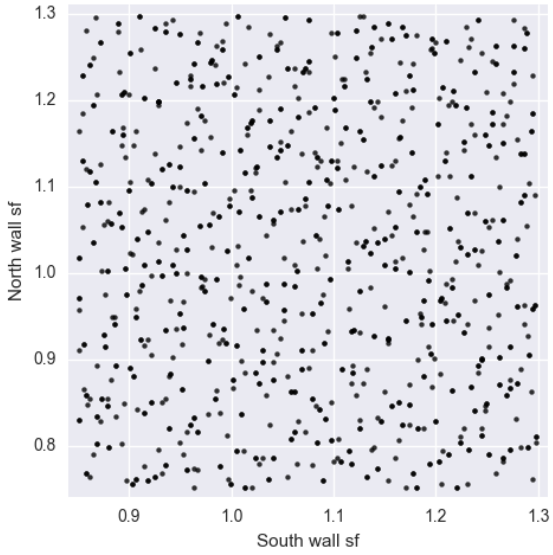


Figure 6.7: 1000 quasi-random samples from the input space using Sobol Sequences.

Input samples were generated by plugging the input parameters and output responses directly into the *DataGenerator* component (Figure 6.8). Once the input sample set was generated, the component automatically runs a simulation analysis for each row in the input file and records the yielded output responses to the specified directory.

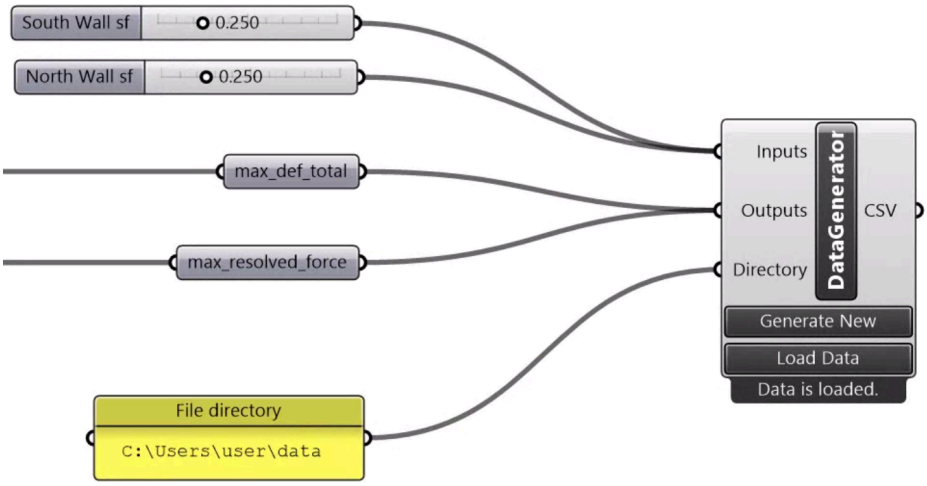


Figure 6.8: Grasshopper implementation using the Data generator to generate input samples.

6.5.1.3 Evaluation (simulation runs)

Once combinations of input values for different combinations of North Wall scale and South wall scale factor values were generated, the Grasshopper component in Figure 6.8 automatically batch runs all simulation analyses and records the input-output data to the specified directory. This process was repeated to generate response data for each of the 2D cross-sectional FEMs, selected for this study. It is important to note that the same set of input samples generated in 6.5.1.2 was used for each of the 2D cross-sectional analyses. This was important since our ultimate scope is to assemble the data into one input-output dataset to build a BNM with the cross-sectional responses for multiple cross-sectional as outputs.

The 2D analysis models were modelled as a series of shell elements representing each box section. The overlapping 'box shells' were connected vertically by linear elastic beam elements, whose spring cross-section, represented the bolt connections between the boxes. The assigned shell thickness was 1cm. For each 2D analysis, the first row of boxes at the base of each cross-sectional FEM, were assumed to be fixed in x, y, and z directions, and from any rotation in the x direction (Figure 6.9). Furthermore, GFRP material properties were assumed for each of the analyses. The characteristic material properties of the GFRP boxes were derived through material-characterisation test data provided to AKT, by the material supplier, Fiberline AG. These properties are recorded in Table 1. Note that the aluminium stiffening profiles were not taken into account in this study.

Material	Engineering constants	Value
GFRP	Tensile Modulus, 0deg	23000 MPa
	Shear modulus	9200
	Density	18 kN/m^3
	Coefficient of thermal expansion (1/°C)	0.00001
	Yield strength	235 MPa

Table 9: Material properties assumed in FEA.

In total, four loading scenarios were considered. These include LC1: gravity loading due to the self-weight of the material, LC2: lateral loads due to wind pressure from the prevailing North direction, LC3: area loads on the bottom rows of boxes caused by public activity and finally, LC4: vertical on all boxes of the pavilion, as a worst-case scenario. Note that all loads were applied as point loads at the points of connection between the overlapping boxes (Figure 6.9).

The values assumed for each load case are specified in Table 10. These four loading scenarios are the same scenarios assumed by AKT during the actual project development.

Case	Load	Value
LC1	Gravity (self-weight)	9.8 KN/m2
LC2	Wind load (North)	0.3 KN/m2
LC3	Vertical loading (bottom rows)	1 KN/box
LC4	Vertical loading (all boxes)	0.75 KN/m2

Table 10: Load quantities considered for each scenario.

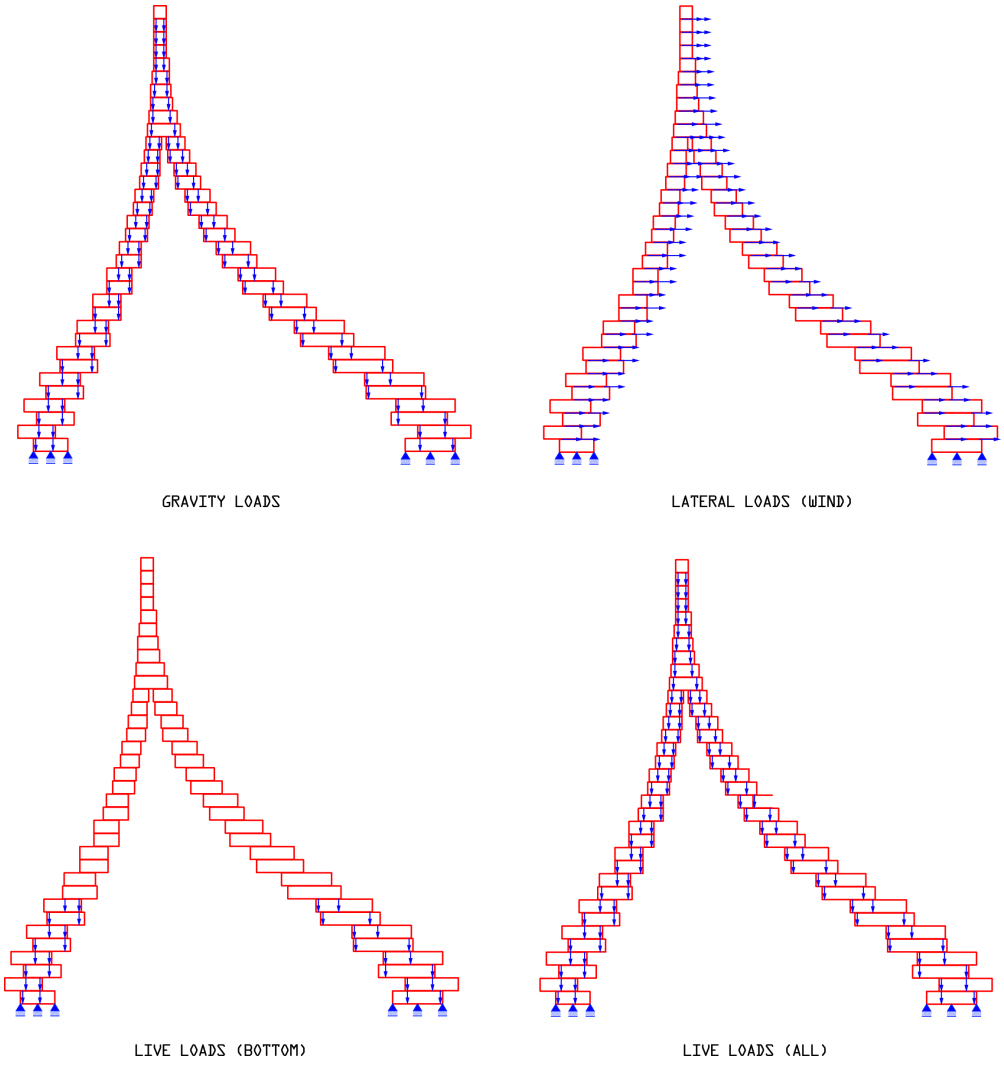


Figure 6.9: Loading scenarios considered for each cross-sectional finite element model.

For each cross-sectional frame analysis, the following FEA responses listed in Table 11 were recorded.

Response	Notation	Units
total maximum deflection (sum of all load cases)	*max_def*	mm
maximum resolved axial and shear forces in the bolts (all load cases)	*max_force*	kN

Table 11: Considered FEA responses.

In total, the case study involved, 6 simulation experiments, one for each of the selected cross-sections illustrated in Figure 6.5, resulting in 6 input-output CSV datasets.

6.5.2 Building the Bayesian network (step 3)

This section discusses how: 1) metamodels were built for each of the 6 input-output datasets and 2) how these datasets were combined into two main datasets to build two 'global' metamodels with 6 outputs each; one global metamodel for each response.

Once the simulation response data for all six frames were generated and recorded, we progressed onto building the actual Bayesian networks. As discussed, we built two types of metamodels. One for gaining insight into the relationship between global geometry and local behaviour at the cross-sectional scale. Subsequently, we assemble multiple local responses into one multi-input, multi-output metamodel and take advantage of the bi-directionality to help engineers translate local engineering constraints into architectural constraints on the global geometry, based on the cause-effect influence between the architectural parameters and the physical behaviour. The ultimate scope is to communicate behavioural feedback to the architecture team in the form of ranges, to maintain flexibility in the architect's parameters of interest.

In all BNMs, we represent the input parameters and response variables as discrete probability distributions, as required by Bayesian networks. The BNM input and output distributions were derived directly from the input-output dataset generated in the previous step (6.5.1.3). More specifically, the input probability distributions were derived as frequency distributions of sampled input values in the dataset, while the response probability distributions for maximum deflection of each loading case, were derived as frequency distributions of simulation output values in the data set. Subsequently, the input distributions were discretised into equally spaced

bins given the (pseudo) uniformity of the distribution they were sampled from in the first place. Further, the response probability distributions were discretised using a percentile binning approach to ensure that all bins contain the same frequency of data points and thus, avoid model bias issues due to extreme values, as discussed previously in 4.3.1

6.5.2.1 Local BNM

Figure 6.10 illustrates the BNM for exploring local behaviour at different cross-sections of the pavilion. For this scenario, the *north_wall_sf* parameter and *south_wall_sf* parameters were introduced as BNM input distributions, while the FEA response outputs: total maximum deflection due to all four load cases, and maximum resolved axial and shear forces in the bolts due to all load cases, were respectively introduced as BNM output distributions.

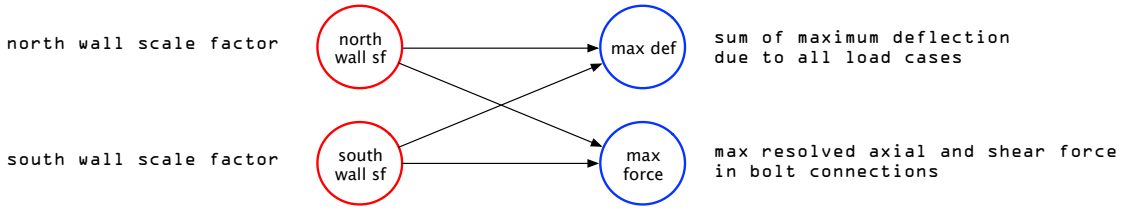


Figure 6.10: BNM used for each cross-section frame.

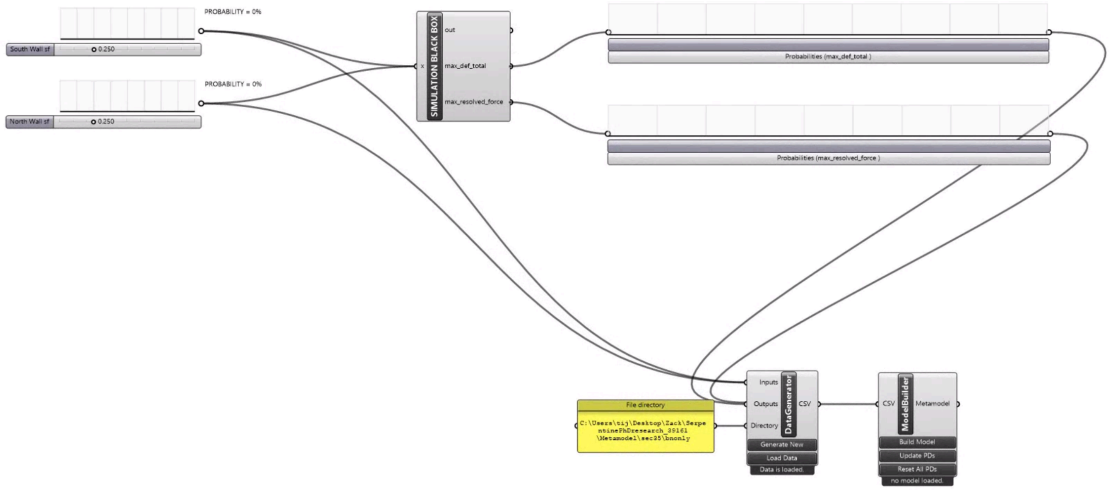


Figure 6.11: Grasshopper implementation of the local BNM.

6.5.2.2 Global BNM

In the second part of the study, we aim to adopt the inverse inference capabilities, to translate local physical constraints based on required engineering criteria to maintain feasibility, into constraints on the global geometry, which can then be communicated back to the architecture team in the form of constrained parameters. The focus of this exercise is to translate structural feasibility constraints on the local response of multiple cross-sections simultaneously into constraints on the global geometry. For this reason, we build two separate global BNMs; one focusing on global maximum deflection, and one focusing on maximum resolved axial and shear force.

More specifically, in both global metamodels, the same parameters controling the global geometry (*north wall* sf and *south wall sf)*, are maintained as metamodel inputs. As for the outputs, in the first global BNM we introduce the maximum resolved axial and shear force at each of the selected sections as multiple metamodel outputs (illustrated in Figure 6.12) while, in the second global BNM we introduce the maximum deflection at each of the selected sections (illustrated in Figure 6.13) as multiple metamodel outputs.

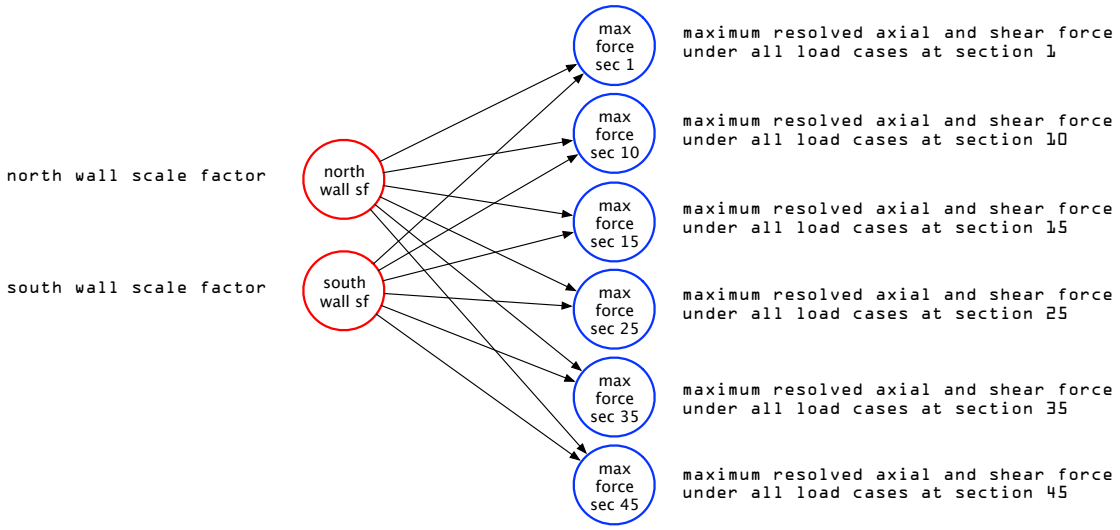


Figure 6.12: Bayesian network metamodel with multiple cross-sectional max force outputs.

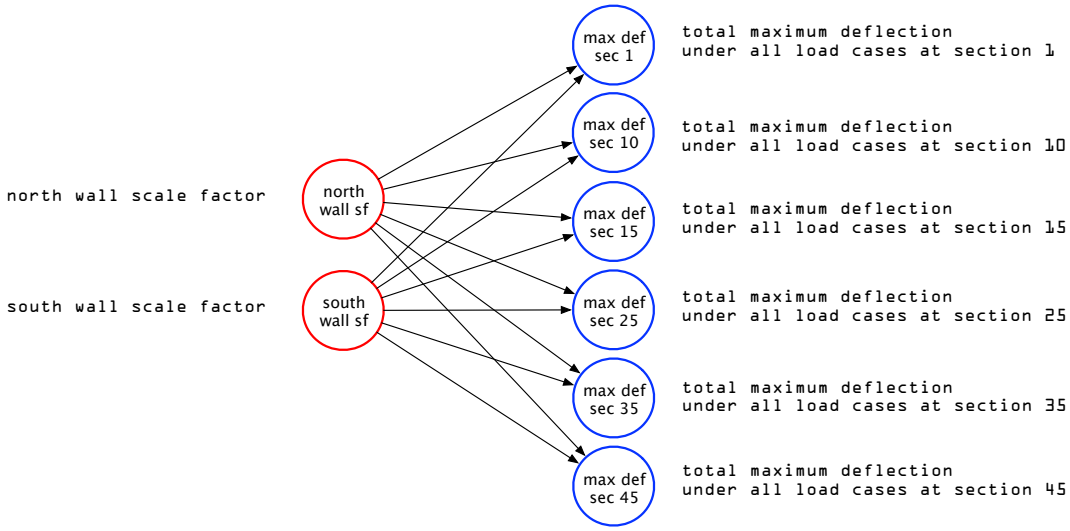


Figure 6.13: Bayesian network metamodel with multiple cross-sectional max deflection outputs.

6.5.3 Validation (step 4)

For this study, we validate the statistical robustness of both global BNMs by assessing how well they can predict simulation responses for each target output, for new input values that were excluded from the dataset used to build the metamodel in the first place. We make use of the *k*-fold cross-validation method discussed in section 4.4.1. We consider a *k* value of 5, implying 5 folds where, the dataset is split into 5 different ways, for each target, for each BNM. Thus, a total of 50 BNMs were built for validation purposes (5 folds*5 targets*2 BNMs). Appendix A illustrates the ***d1*** error histograms for each fold, for each target. More specifically, section A.2 illustrates error plots for predicting the maximum total deflection output for each cross-section in the global BNM in Figure 6.13, while section A.3 illustrates error plots for predicting the maximum resolved force output for each section in the global BNM in Figure 6.12. Note that the cross-validation for each BNM was done using the Python package developed and discussed in Section 5 because the validation functionality was not yet built into the Grasshopper plug-in at the time of writing.

Overall, in the deflection BNM (section A.2), the tallest error peaks seem to concentrate within a 20% error-band. While 20% might seem like a large error-band, the probabilistic aspect of such models must be taken into account where the model is classifying a bin rather than predicting a scalar value. In fact, if we were to plot the ***d2*** histograms (section A.4), we can note the tallest error peaks concentrate within 5%, implying that the BNM is predicting the correct bins in which the simulated values are actually contained, most of the time.

Additionally, we note that for both ***d1*** and ***d2***, it seems that the distribution of errors for predicting both force and deflection also spreads beyond the 20% mark, for cross-sectional cases 1, 25 and 45 (see Figures A-4, A-7, A-9, A-10, A-13, and A-15). However, the probability of such errors is significantly low. Furthermore, the error plots reveal how the spread occurs in sparse clusters. A simple scatter plot of the input-output data (Figure 6.14), reveals how the sparse symptoms occur because the response data is grouped into sparse clusters, in the first place. This is a direct result of the parametric model setup where the logic causes discrete perturbations in the response data. In general scenarios, this may either be a sign of an issue with the parametric setup, not enough input parameters or simply the nature of the problem.

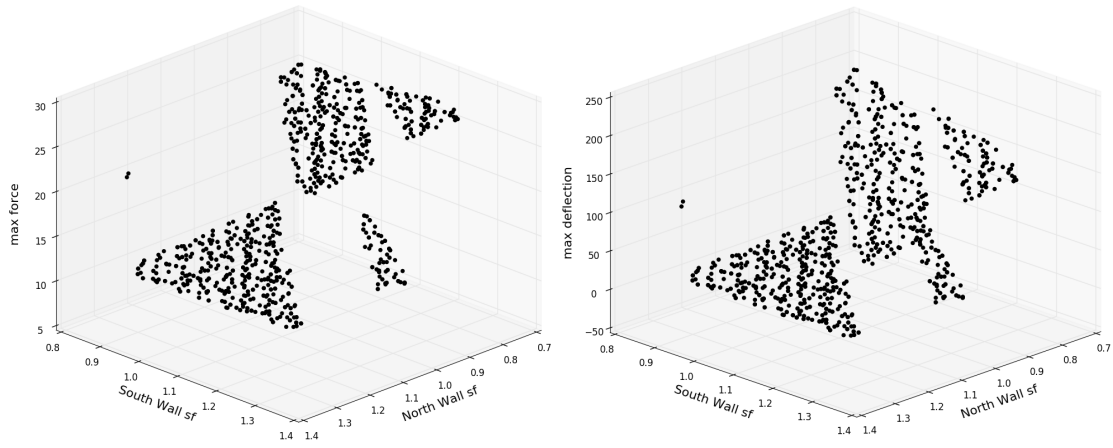


Figure 6.14: Scatter plots of example input-output data for cross-section 1: max force (left), max deflection (right).

Irrespective of the cause, it is good to note that in general, Bayesian networks are known to suffer some limitation when it comes to inference with models whose model parameters (conditional probability tables) were learned from sparse or missing data. Literature such as Daly, Shen, and Aitken (2011) and Liao and Ji (2009), indicate that this tends to occur because some of the probabilities can be undefined if the case does not show up (missing) in the dataset. An immediate resolution to 'smoothen' out the gaps between the data clusters, is to include more input parameters in the problem setup that can explain the 'missing' scenarios (data points). Alternatively, work from the statistics and machine learning community such as in Heckerman et al. (1995) and W. Buntine (1996b), suggest to assume some form of prior distribution on the response variables, which is subsequently updated from data. We intend to consider such remedies in future work. For the sake of this case study, we will maintain the intended parametric set up with North and South wall scale factors as inputs, to test the limits of the approach. Having foreseen the above limitations, the outcomes of this case study still provide insight.

6.6 Multi-input multi-output bi-directional inference with BNM

Our ultimate goal in this case study is to communicate high-resolution feedback about structural behaviour to guide and support the architect's intuition. More specifically, instead of analysing only one scenario at a time, in this section we illustrate how the engineering team may utilise the bi-directional inference capabilities of the BNMs built in 6.5 to: 1) gain an understanding of the relationships between the architectural parameters controlling the global geometry and the local cross-sectional behaviour of the pavilion, and 2) utilise the knowledge of these relationships to communicate constraints on the ranges of the architect's parameters to the architecture team, instead of communicating discrete analysis output. This way, flexibility of choice is maintained, by guiding the architects towards the structurally feasible regions of the design space.

6.6.1 Local BNMs: understanding sectional behaviour

For brevity's sake, in this section, we will only illustrate BNMs for section 1 and section 25, out of the 6 selected sections.

Once each of the sectional BNMs is built, we can manipulate their input and output probability distributions to explore the cause-effect relationships between inputs and outputs, bi-directionally. More specifically, the relationship between north wall scale factor (*north wall sf*), south wall scale factor (*south wall sf*), and cross-sectional maximum deflection and maximum forces in the bolt connections. For each scenario we keep in mind that the engineering goal is to avoid areas of extreme global forces by 1) maintaining max axial and shear forces in the connection bolts <= ~10kN, and 2) keeping the maximum deflection at each section at a minimum.

6.6.1.1 Section 1

Figure 6.15 illustrates the BNM for cross-section 1 (see Figure 6.16) where inputs and outputs are represented as discrete probability distributions.

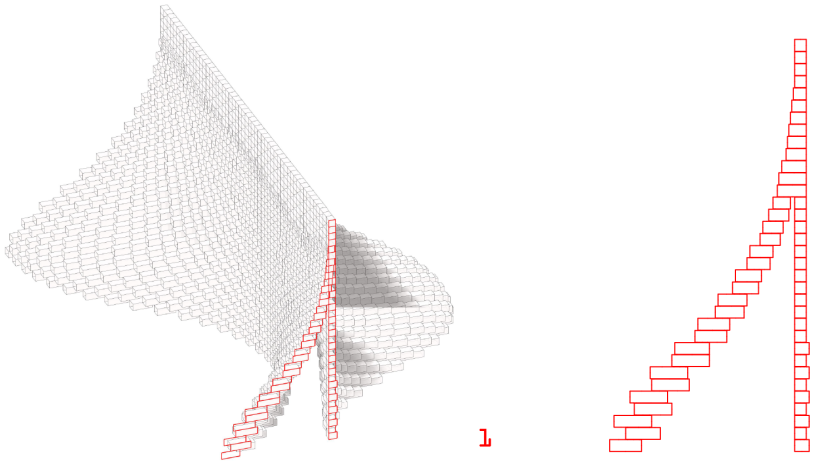


Figure 6.15: 2D frame at cross-section 1.

We begin this exercise by attempting to ask, which input settings are likely to maintain maximum axial and shear forces in the connections <= ~10kN? We attempt to answer this question by setting hard evidence on the *max_resolved_force* probability distribution. We do this by setting the bins, whose range contains or falls <= ~10kN, to 100% probability (see Figure 6.17). The underlying Bayesian network automatically "updates" all other distributions in the network, based on the relationship between the *north wall sf* and *south wall sf* wall parameters and the the *max_resolved_force*. The use of the word "updating" implies the computation of Bayesian inference. When observing the updated input probability distributions in Figure 6.17, one can immediately note a relationship trend, where decreasing the *south wall sf* while increasing the *north wall sf* can produce global geometries that maintain the maximum axial and shear force in the bolt connections of cross-section 1, below 10.9421kN. In other words, the *north wall sf, and south wall sf* distributions in Figure 6.17 illustrate *how* to set geometric parameters to maintain axial and shear forces below 10.9421kN.

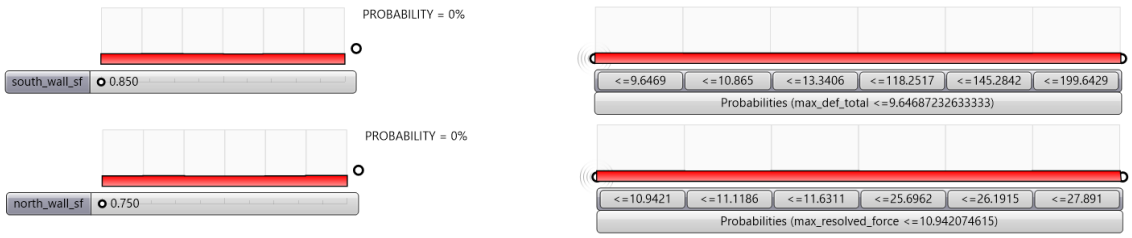


Figure 6.16: Marginal probability distributions.

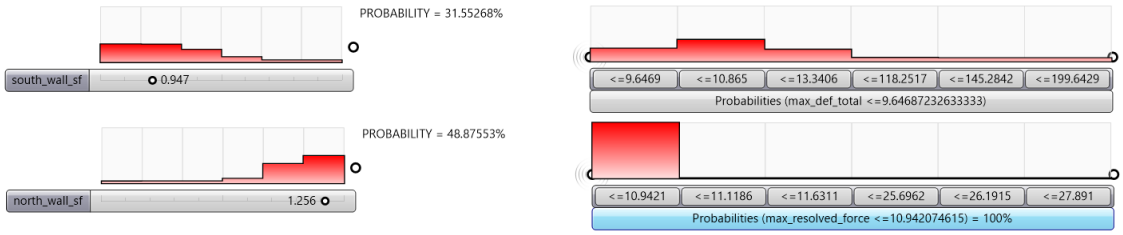


Figure 6.17: Probability (max_resolved_force <= 10.5421 kN)=100%

We can also note that after "updating" in Figure 6.17, the probability distributions of the other response was also updated because they share a common relationship with the inputs. In theoretical terms, though the responses are marginally independent of each other, they remain conditionally dependent through their mutual relationship with the input parameters, even though they are not directly linked by causal edges in the Bayesian network (because they are not a direct cause of each other). Implicitly, these updated response distributions provide us with information about the relationships between the responses; reducing the maximum force seems to have an almost complementary relationship with reducing the maximum deflection at cross-section 1.

For example, if we were to reset the distributions back to their marginal state (see Figure 6.16), and this time set the smallest bin on the *max_def_total* distribution to 100% probability (Figure 6.18), we observe that updated *north_wall_sf* distribution illustrates a similar trend as the previous scenario in Figure 6.17, while *south_wall_sf* shows an almost reverse scenario.

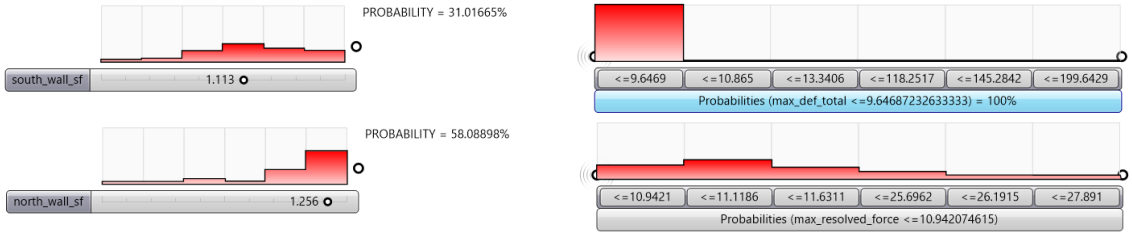


Figure 6.18: Probability (max_def_total<= 9.65mm)=100%

The two goals of maintaining *max_resolved_force* <= ~10kN, and maintaining a low maximum deflection, seem to have a complementary relationship at section 1. Therefore, the natural next step would be to set the bins with the lowest bins in both *max_resolved_force* and *max_def_total* to 100% probability and to observe the updated input distributions (see Figure 6.19). The resulting updates, immediately distribute the probability towards the regions of the input space that are likely to satisfy the 'constraints' placed on the outputs. As expected, the probability distribution of the *north_wall_sf* is mostly distributed towards the region of higher input values,

while *south_wall_sf* probability distribution concentrates higher probability towards the overlapping bins between the probability distributions in Figure 6.17 and Figure 6.18.

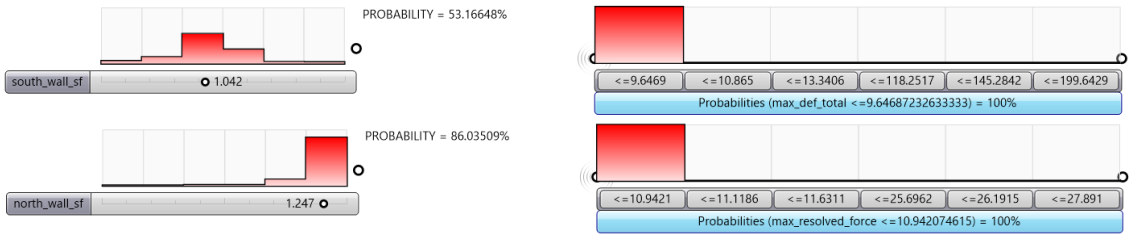


Figure 6.19: Probability (max_def_total <= 9.65mm)=100%,
Probability (max_resolved_force <= 10.94kN)=100%.

6.6.1.2 Section 25

Now, let us repeat the same exercise with the BNM built for cross-section 25 (Figure 6.20).

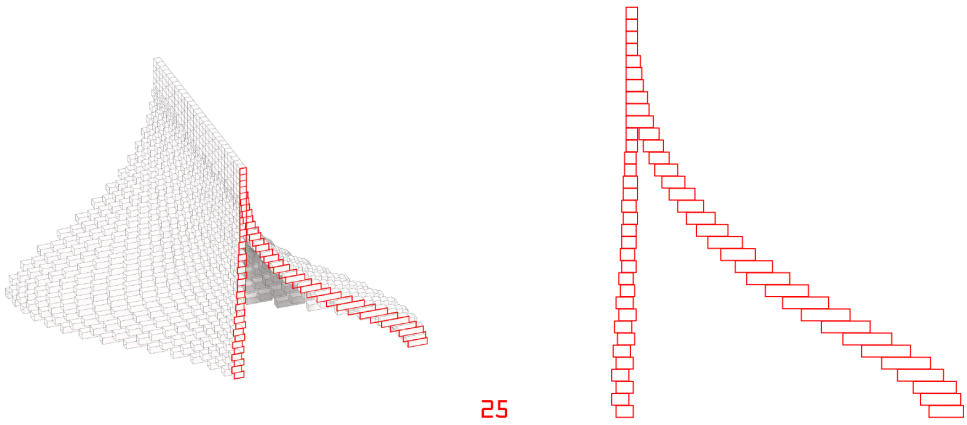


Figure 6.20: 2D frame at cross-section 25.

Figure 6.21 illustrates the discretised inputs and outputs and their marginal probability distributions. We can start with asking the same question as in the previous cross-sectional scenario; what are the input distributions that are required to maintain a maximum resolved axial and shear force <=~10kN? We attempt to answer this question by setting the *max_resolved_force* bins, whose ranges contain or are <= ~10kN, to 100% probability. Figure 6.22 illustrates that the maximum bound on the first bin is 11.68kN and thus, only the first bin is set to 100%. On updating all probability distributions in the network, we can immediately notice how the input probability distributions change in Figure 6.22. More specifically, we can note that increasing

south_wall_sf has a minimising effect on the forces in the bolt connections, whereas *north_wall_sf* values between 0.75 and 0.937 guarantee maximum forces under 11.68kN.

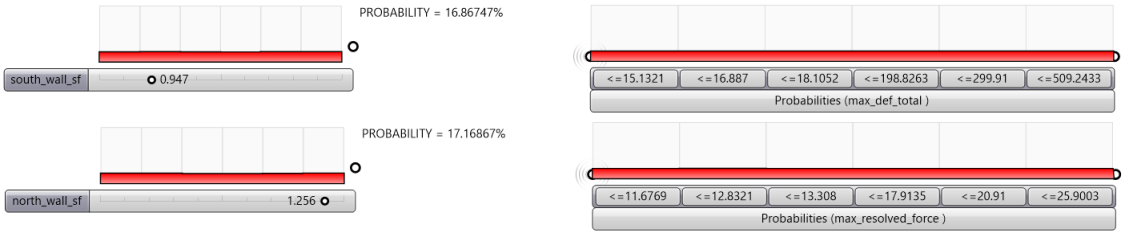


Figure 6.21: Marginal probability distributions.

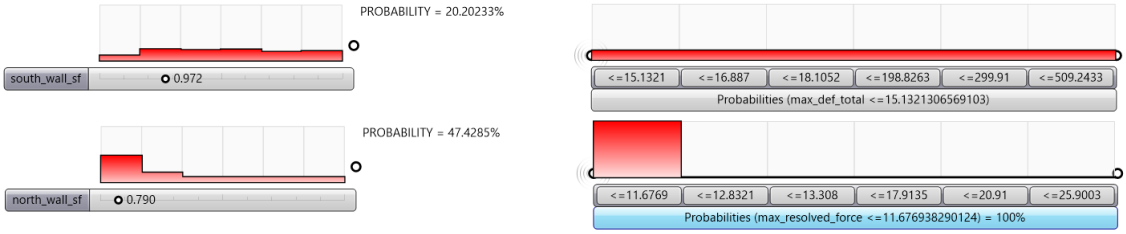


Figure 6.22: Probability (max_resolved_force <= 11.68kN)=100%

The updated *max_def_total* probability distribution also shows how minimising the forces also minimises maximum deflection in the cross-section. In other words, setting the bins with the lowest values of both *max_resolved_force*, and *max_def_total*, confines the input distributions to the ranges indicated in Figure 6.23. Also, note how it is the south wall that contributes mostly to the maximum deflection in the cross-section 25. This is justified by the verticality of the south wall (Figure 6.20). The sensitivity of *south_wall_sf* was noted by the indifference in shape between the *north_wall_sf* probability distribution in Figure 25 and 26.

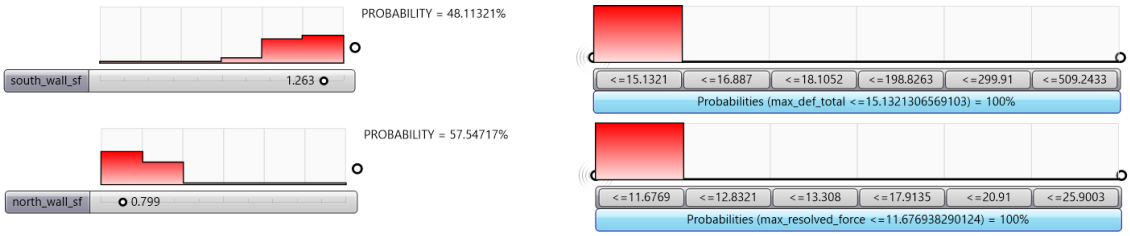


Figure 6.23: Probability (max_def_total <= 15.13mm)=100%, Probability (max_resolved_force <= 11.68kN)=100%.

On comparison, the two inference exercises for sections 1 and 25 illustrate how the response outputs *max_resolved_force* and *max_def_total* have the opposite relationship with the inputs,

particularly the *north_wall_sf*. This inverse relationship is evident when comparing *north_wall_sf* probability distribution of section 1 in Figure 6.19 to section 25 in Figure 6.23.

In the next section we attempt to assemble *max_resolved_force* and *max_def_total* responses from each cross-section (1, 10, 15, 25, 35, 45) into one BNM.

6.6.2 Global BNM: translating sectional constraints into global design parameter constraints

To reiterate, our goal of assembling a global BNM is to translate local constraints on multiple cross-sectional responses, into feasible input ranges. We use the bi-directionality of the Bayesian network to inverse infer the *north_wall_sf* and *south_wall_sf* probability distributions that would satisfy the desired response goals. In other words, we aim to identify the feasible input ranges that yield *max_resolved_force* <= ~10kN and keep *max_def_total* at a minimum, at all cross-sections, simultaneously. The ultimate scope is to subsequently communicate these global ranges back to the architecture team.

As introduced earlier, in this exercise we split the responses into two global BNMs; one focusing on *max_resolved_force* and the other on *max_def_total*.

6.6.2.1 Global BNM: reducing maximum forces

Figure 26 illustrates the assembled global BNM where *max_resolved_force* response from each of the selected cross-sections are introduced as multi-outputs. Once assembled, we can manipulate the probability distributions to infer an understanding of the cause-effect relationships, similar to the previous exercise in 6.6.1.

We start by attempting to infer the input distributions that would yield a minimum *max_resolved_force* for cross-section 1. We achieve this by assigning 100% probability to the bin containing the lowest max force values (<= ~10kN), as illustrated in Figure 26, and subsequently updating all other probability distributions via inference. Figure 27 illustrates the updated input and output probability distributions. Immediately, we can observe two changes: in the inputs and the outputs. The updated input probability distributions indicate that a combination trend of decreasing *south_wall_sf* and increasing *north_wall_sf* can achieve *max_resolved_force* <= ~10.9kN at cross-section 1. As for the remaining updated response probability distributions, these indicate an almost inverse relationship between minimising the maximum forces in cross-section 1 and the minimising the distributions of maximum force in the other cross-sections.

From this overview, we can already tell that there are no input configurations that will satisfy maximum forces <= ~10kN, for all cross-sections.

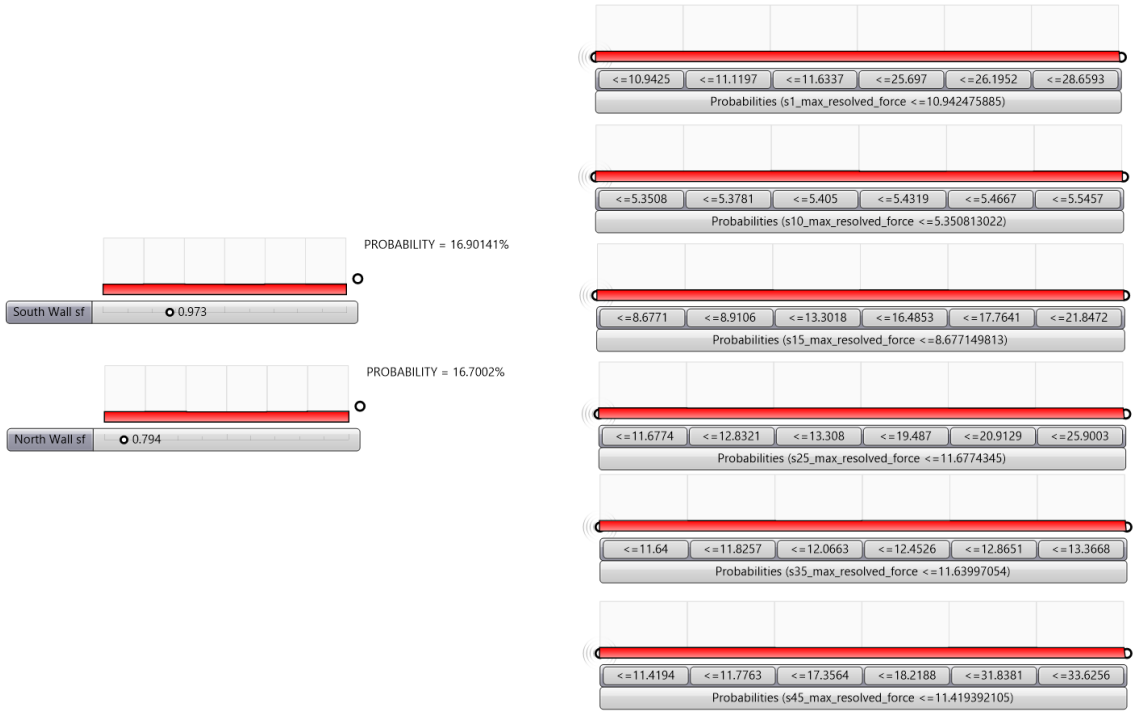


Figure 6.24: Marginal probability distributions.

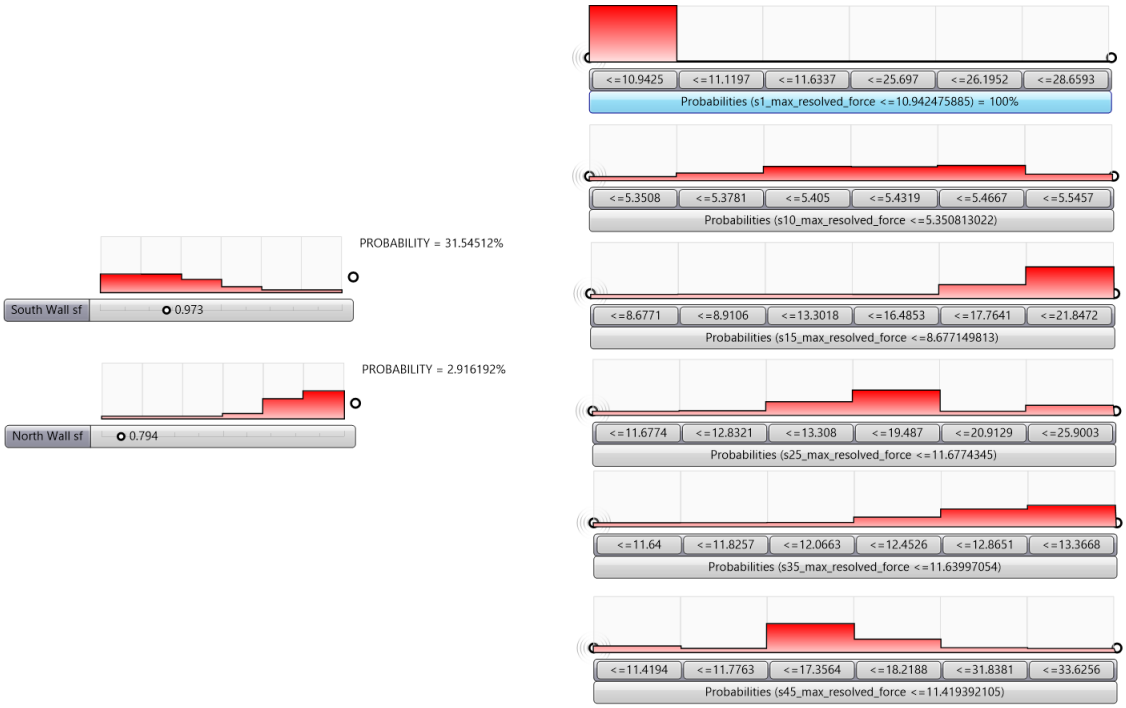


Figure 6.25: Probability (s1_max_resolved_force <= 10.94kN)=100%.

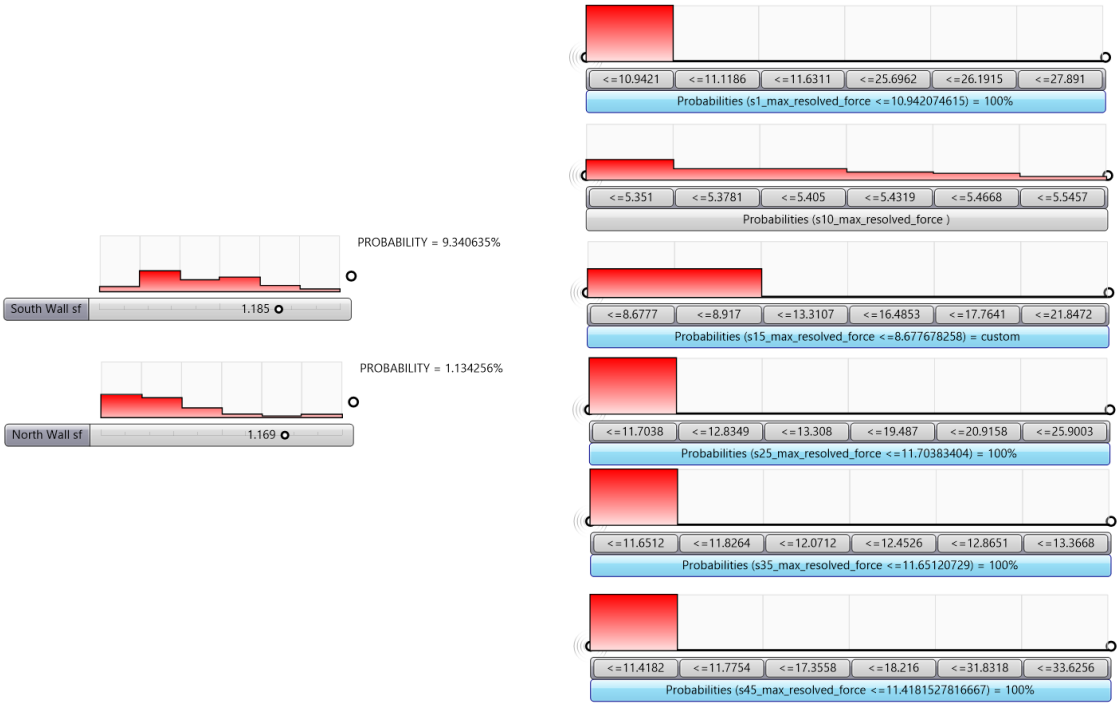


Figure 6.26: Probability (s1, s25, s35, s45_max_resolved_force <= ~10kN)=100%.

It is tempting to distribute the 100% probability to the response bins whose values fall below 10kN, and observe what the input distributions would yield. However, we suspect that the shape of the updated input probability distributions will result in 'wide and shallow distributions' because these criteria cannot be satisfied due to the inverse relationships observed in Figure 6.25. In fact, the updated input distributions in Figure 6.26 confirm our suspicion.

Therefore, the above initial exercise calls for a compromise solution by trading-off the response distributions. We can continue to utilise the interactivity of the BNM to 'slice' the joint probability distribution between all inputs and outputs, at different response bins, to explore these trade-offs.

In this context, we proceed by setting the third cross-section force response bin to 100%, as a means to observe the updated response distributions. We select the third bin as a middle ground between the first bin of the cross-section 1 response distribution, and the distributions of cross-sections 15, 25, and 35 in Figure 6.26. Therefore, slicing at the middle cross-section 1 is a form of compromise between the extremes distributions. Once updated in Figure 6.27, we observe the uniformity of the other response distributions. We take this as a sign of 'stable bin for fixing' for cross-section 1.

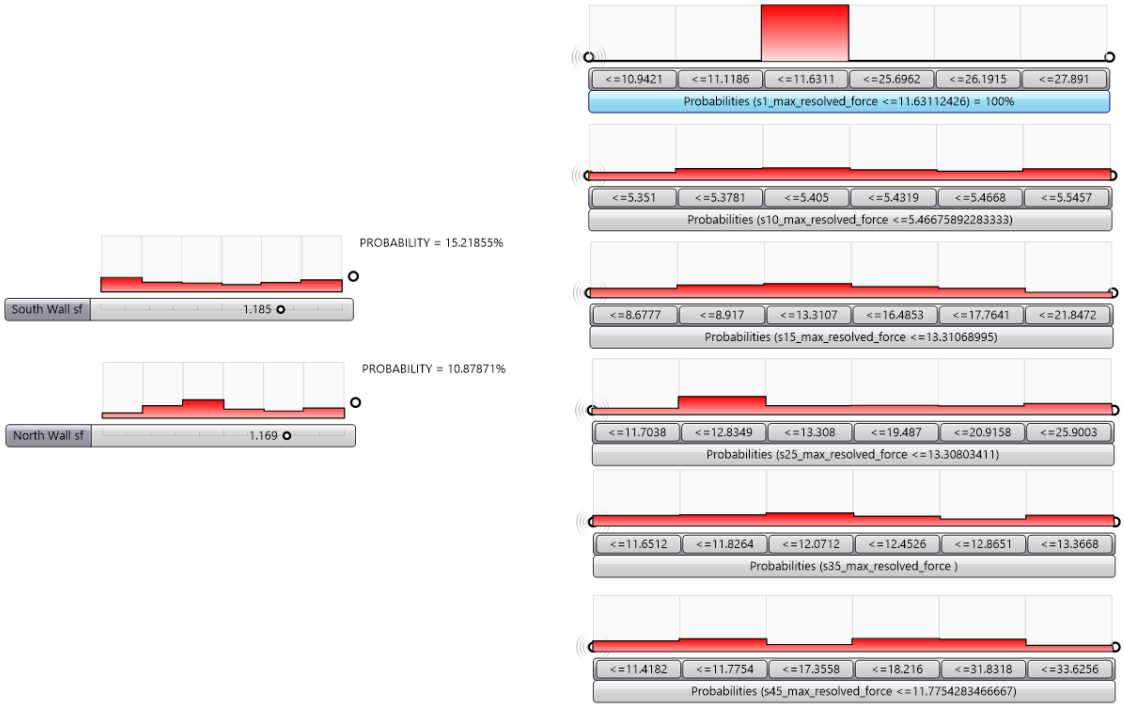


Figure 6.27: Probability (s1_max_resolved_force <= 11.63kN)=100%).

Further on, we proceed to set the third bin on cross-section 15 to 100% probability. We can skip cross-section 10 and leave its probability distribution unconstrained, given all values lie below 5.55kN (<10kN). Once updated, the distribution peaks of the remaining response distributions (cross-sections 25, 35 and 45), are suggestive that it is likely to find a compromise around 13kN.

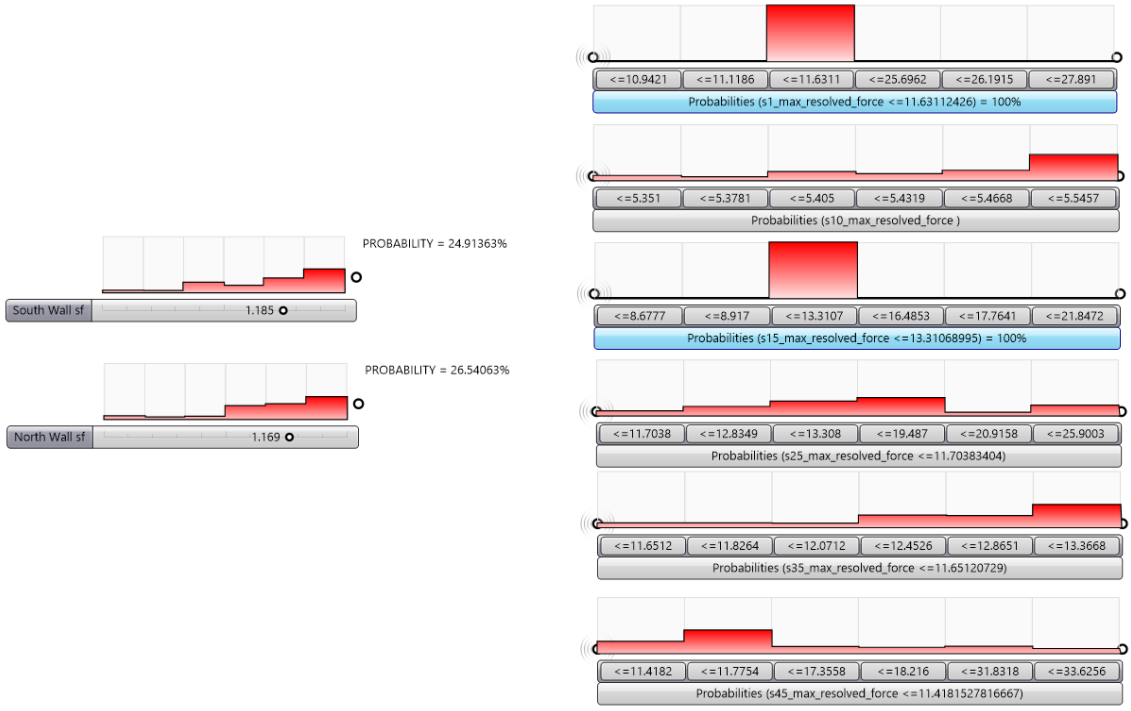


Figure 6.28: Probability (s1, s15_max_resolved_force <= ~13. 3kN)=100%.

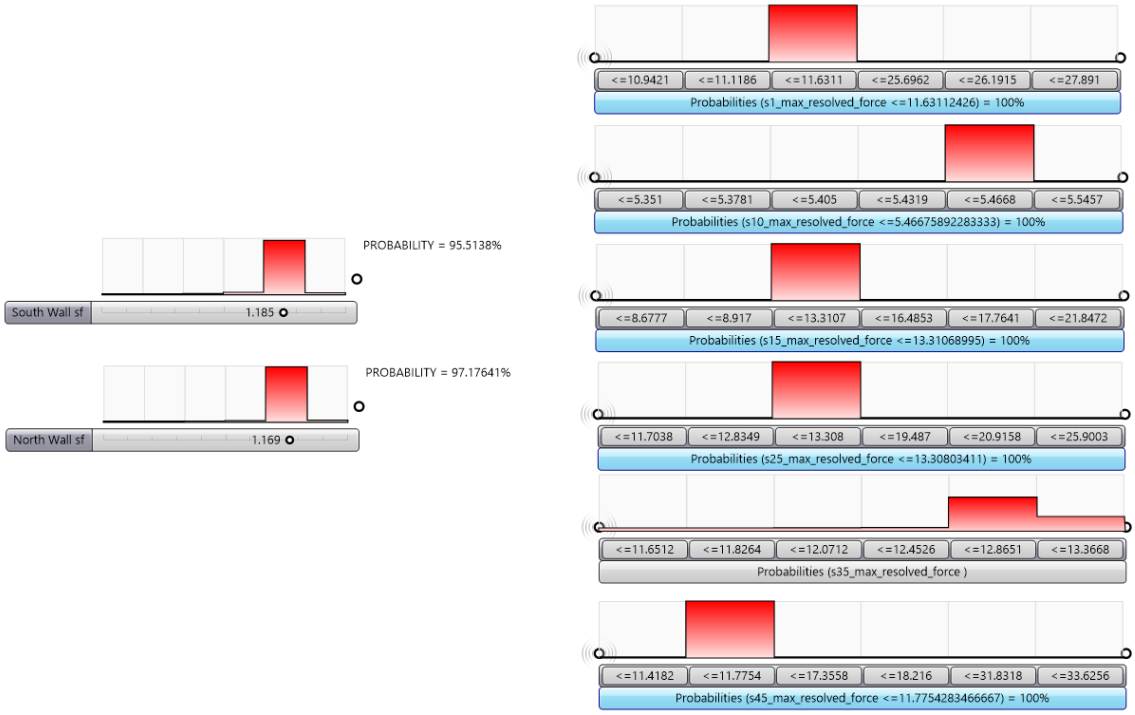


Figure 6.29: Probability (s1, s10, s15, s25, s45_max_resolved_force <= ~13. 32kN)=100%.

We take on a similar approach for the subsequent response distributions until a compromise that yields narrower input probability distributions is reached. It is good to observe, as the response distributions are concentrated to the feasible response ranges, the convergence of the probability distribution increases as can be seen when following through Figure 6.27Figure 6.29.

6.6.2.2 Global BNM: maximum deflection

Similarly, we assemble another global BNM where *max_def_total* response from each of the selected cross-sections are introduced as multi-outputs, to explore the influence of the pavilion geometry on the global deflection of each cross-section. Figure 6.30 illustrates the discretised marginal probability distributions of the BNM inputs and outputs.

Our scope is to find input ranges that reduce both high forces and high deflection at each cross-section. For this reason, we begin this exercise with utilising the input probability distributions inferred from the previous exercise, and observe whether the inferred *max_def_total* distributions fall towards the bins with the lower deflection values. On a side note, this exercise illustrates the bi-directional capability of the BNM to compute both inputs→outputs and outputs→inputs.

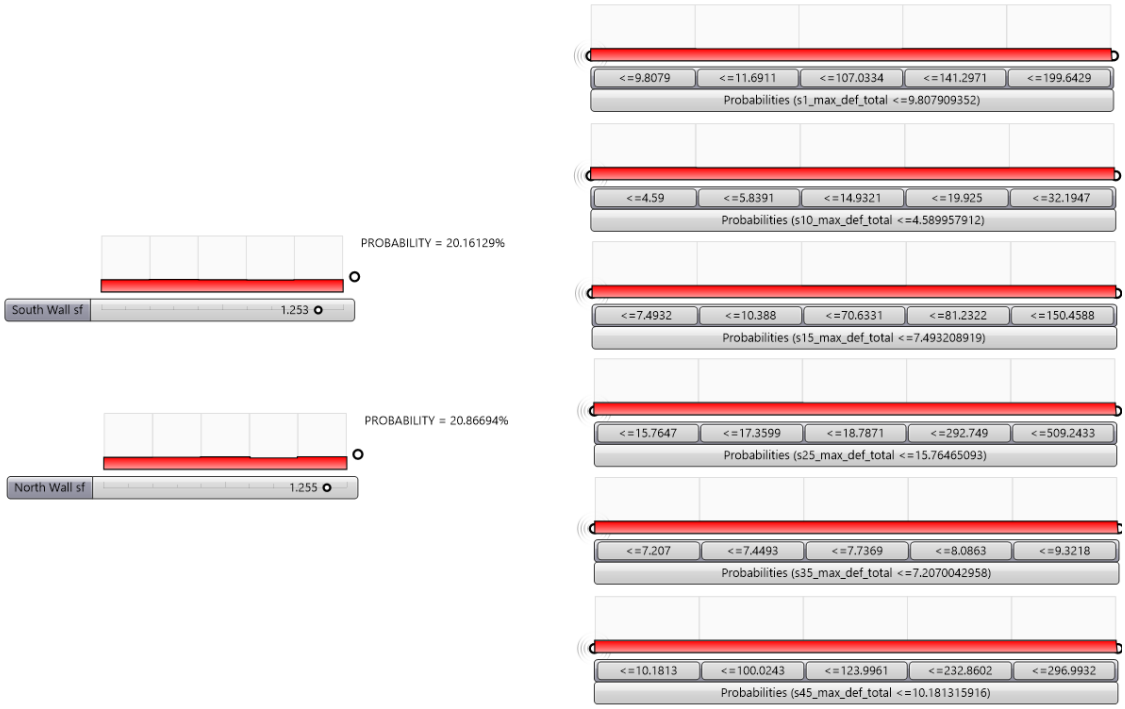


Figure 6.30: Maginal probability distributions.

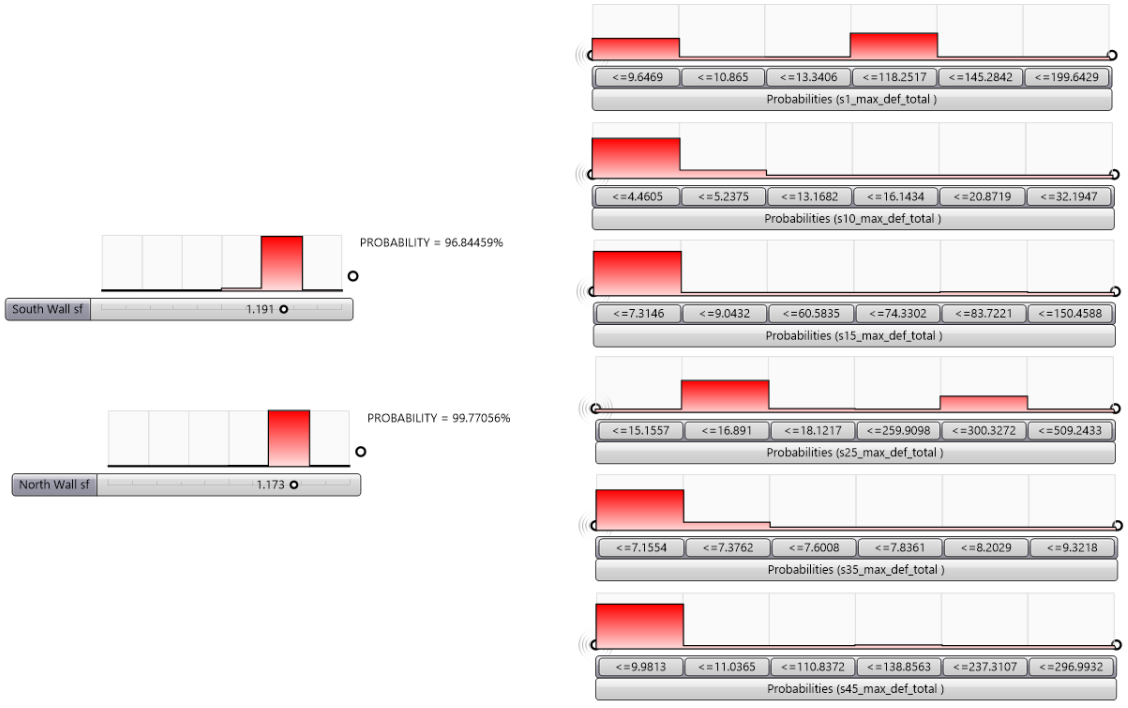


Figure 6.31: Setting the input distributions derived from the ‘force response scenario’ as inputs.

From Figure 6.31, we can observe that the compromise found in the previous exercise in section 6.6.2.1 yields distributions whose probability is distributed to specific bins, mostly towards the

lower values. Subsequently, we redistribute the response probability to the updated bins containing lower values and re-infer the input probability distributions.

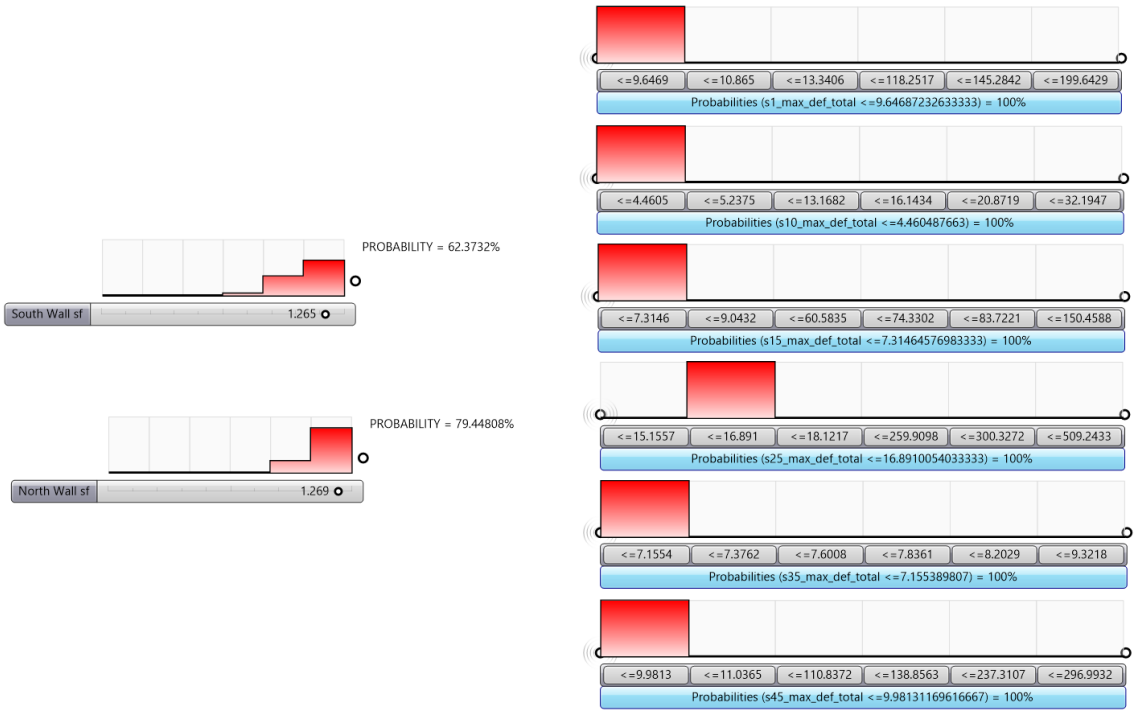


Figure 6.32: Further adjustments to the output distributions to secure lowest feasible forces.

As a final check, we input the inferred input distribution in Figure 32, as input distributions to the force BNM.

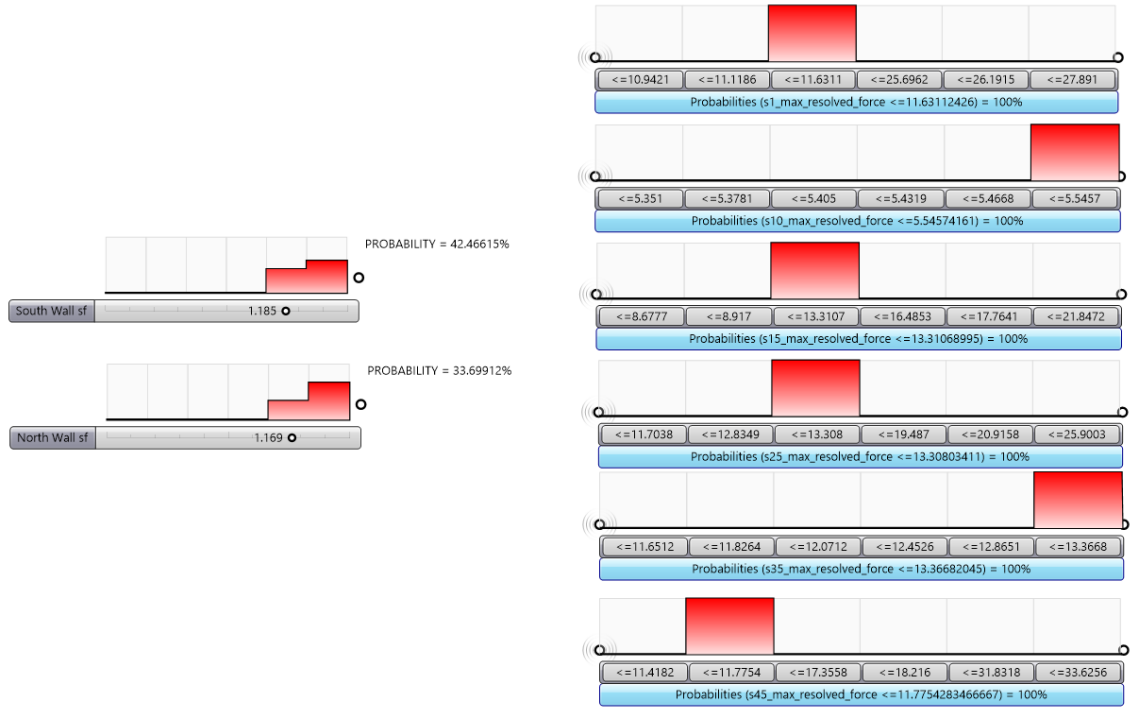


Figure 6.33: Final suggested input ranges and compromised feasible output responses.

6.7 Outcomes

The outcomes from both exercises suggest the input ranges tabulated in Table 12. These ranges guarantee axial and shear forces in the bolt connections below 13.4kN and maximum deflection values below 17mm. Note that the study did not take into account the lateral stability of the structure between cross-sections. Lateral stiffening would contribute to reducing the maximum deflection further.

Parameter	**Ranges**
South wall scale factor	[1.1487, 1.3]
North wall scale factor	[1.115, 1.3]

Table 12: Suggested input ranges.

The comparative study in Table 13 and Figure 6.34 illustrates a suggested geometry outcome selected randomly from the suggested input ranges. The suggested input values in Table 12, taken from within the suggested range in Table 13, show a slight overall improvement in the cross-sectional force responses from the as-built geometry. This slight improvement can be seen from the fewer 'red' dots in Figure 6.34 (right) from left.

Parameter		**As-built input value**	**Suggested input value**
South_wall_sf		1	1.191
North_wall_sf		1	1.290
Max Force [kN]	**s1**	11.21kN	11.14kN
	s10	5.437kN	5.47kN
	s15	8.87kN	9.13kN
	s25	21.52kN	13.15kN
	s35	11.92kN	12.8kN
	s45	18.47kN	11.48kN

Table 13: Actual and suggested input ranges and response results.

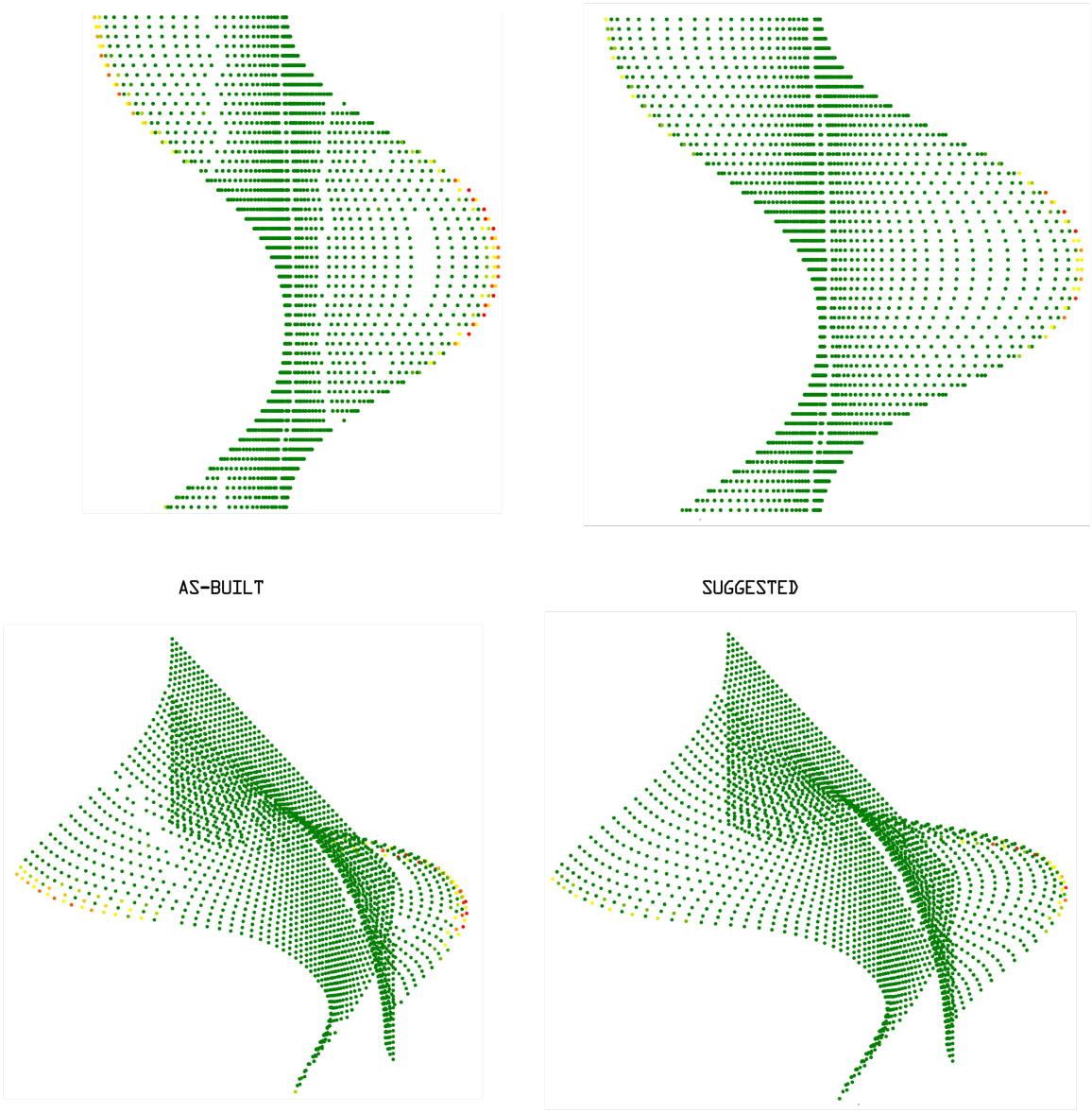


Figure 6.34: False-color dot plot comparison between as-built and suggested outcome.

6.8 Discussion

The outcome of the global inference exercises in 6.6.1.1and 6.6.1.2, helped to reveal a cross-section of the relationships between the global geometry of the north and south wall of the pavilion, and the localised and global behaviour of a number of cross-sections along the pavilion spine.

Both local and global inference exercises revealed conflicting relationships between inputs and outputs but also amongst the outputs themselves. In general, stretching a wall implies stretching

the length of the box elements thus, axial forces increase due to larger bending moments on the bolt connections. On the flip side, it seems longer boxes, seem to stabilise the global deflection of the structure, probably because of increased total weight. Therefore, the overall conflict is one between local behaviour and global behaviour.

As a result of the conflict, it was not possible to find input distributions the satisfy the goals set out at the start of this case study. That is: to maintain maximum resolved axial and shear force in the bolt connections <= ~10kN, while keeping the maximum deflection at a minimum, for all cross-sections simultaneously. Instead, we took advantage of the multi-response capabilities of the BNM together with probabilistic inference, to explore the trade-offs between the responses, with the scope of converging to a compromise. More specifically, the trade-off occurred between slightly higher bolt capacity requirements and slightly higher global deflection. The compromise was achieved by observing the level of certainty of the input probability distributions, on each step of the exploration. Therefore, the exploration took the form of a sequential process of setting evidence on the responses and observing the updated response distributions as a means to guide the convergence. The approach can be interpreted as a rule of thumb and was flagged through the application of this case study.

From the study it was clear that the selection of input parameters had an impact on the quality on the outcomes of the insight gained in two aspects: 1) model robustness, and 2) conflicting constraints.

1) Conflicting objectives demand more localised control over the geometry. This would increase the chances of satisfying the intended engineering limits (force < 10kN and minimise max-deflection). To a certain extent, the inferred outcomes in the local exercises in 6.6.1, can be considered as suggestive local scale ranges for each cross-section.
2) The sparseness of the response data flagged in 6.5.3, is directly implied by an insufficient number of input parameters to describe localised behaviour, that seems to be controlling the overall behaviour of the structure. This makes sense, given that the loads are carried directly by the local box elements and the local bolt connections.

Lastly, with input ranges in hand, the engineering team may communicate these ranges in the form of constraints on the architectural geometry. This way, the architecture team may retain a capacity to control design creatively, within feasible ranges. Furthermore, the availability of software package such as the Grasshopper plug-in presented in Chapter 5, can be used as a translational mechanism for engineers to not only communicate ranges on the parameters, but also communicate an ‘expert-tuned’ version of the Bayesian network metamodel as an assistive tool

for architects to navigate difficult design spaces, uninteresting design activity through its flexible representation of knowledge.

6.9 Challenges

The application of this case study project helped to flag certain challenges and limitations of the BNM approach presented in this thesis, which may arise at the architecture-engineering interface of real project scenarios. The main challenges highlighted pertain to discretisation of the inputs and output probability distributions, amount of data required, and field response outputs (in order of severity).

6.9.1 Trading-off multiple responses

From this exercise, it became clear, that when considering multiple responses in a BNM, it is not easy to reach a compromise, especially when the number of inputs is small, and when the relationships between inputs and outputs is complicated due to nonlinearities. From this case study, we understood the importance of observing the 'shape' of the updated input probability distributions, as an indicative measure of how reliable the inferred input ranges are. In other words, a wide and shallow distribution, implies a not-so-reliable range, i.e. a low probability that the predicted input ranges will correctly yield the selected response bins. On the other hand, an input distribution shape whose probability is more concentrated towards a narrower range/s is more likely to yield the requested solutions (combined desirable responses).

So, when considering numerous responses, how do we reach input probability distributions whose shape is more certain? When exploring trade-offs between responses by setting hard distributions, we suggest taking note of how the other response distributions update. In other words, the shape of the updated response probability distributions is suggestive as to what is a likely trade-off and what is not likely at all.

The lesson learned here is the need for a benchmark, below which an inferred input distribution can be considered as an unreliable source of information. A suggestion is to compute the estimate of the joint probability of the 'tallest' input bins, as a global measure of inferred input certainty. The joint probability is achieved by simply multiplying the probabilities of each bin. This is allowed by the law of probability when two random variables are marginally independent. Such a measure requires future research.

The BNM is not an automatic approach. Making intelligent inferences with the BNM requires and encourages human intervention.

6.9.2 Discretisation

By default, Bayesian networks require for continuous variables to be discretised into bins. In Chapter 4, we discussed how the discretisation of the variables in the network, has a direct influence on the predictive quality of the metamodel. For this reason, in the previous chapter, we established a rule of thumb for selecting the discretisation method based on the form of the distribution. More specifically, when the probability distribution of an input or output variable looks uniformly distributed, we can opt for equally spaced ranges (typically the case for the input distributions). On the other hand, when the data takes a less uniform distribution (typically outputs), we opt for percentile binning as a way of securing bins containing equal frequencies of data. A percentile approach reduces bias in the model by avoiding very wide ranges containing few data points, due to extreme values.

However, the discretisation of a variable into bins is dictated by the discretisation method only. The resulting bin ranges may not be helpful for the specific query of interest. Thus, may become limiting when trying to make specific queries. In the future research, we intend to develop a hybrid approach, that combines preferred discretisation ranges by a user, and automatic approach. As for the latter, we intend to look further into approaches such as minimum description length (MDL), which is an optimisation algorithm that aims to suggest the best combination of number of bins and ranges, that maximise the predictive robustness of the model. Applications of this approach for Bayesian networks can be found in (Levine, 2011).

6.9.3 Data concerns

In principle, Bayesian networks require a thorough amount of data to build the conditional probability tables that capture the relationship between the variables. As with any other type of metamodels, there is no explicitly defined rule to determine the number of data points required to build a Bayesian networks metamodel. In general, the number of data points required should increase with the number of input parameters considered to increase the chances of a reliable model that captures important underlying relationships between parameters and responses, which would otherwise not be captured with fewer data. However, when the time it takes to run one simulation is lengthy, generating enough data to build a reliable model can become daunting. For example, in this case, the finite element analysis of each frame was not particularly time demanding, however, the logic involved in processing the surface, generating the grid and slicing

the frames, was demanding and thus, as a result, each simulation iteration took an average of 30 seconds. When multiplied by 1000 data points this resulted in approximately 7 hours on a regular machine.

In this case study, the limitation of generating data for each section limited the number of responses considered in the global metamodel. The original intent was to build a BNM with 51 responses, one for the maximum deflection at each of the 51 sections of the pavilion.

We justify the insight gained to be worth the computational expense to generate the data and which may perhaps be overcome with increasingly available powerful machines. However, despite the focus of this thesis not concerning data efficiency per se, we discuss potential measures to address data concerns for metamodels in general, as part of our future work, in Chapter 7.

Chapter 7

CONCLUSION AND FUTURE WORK

7.1 Intentions and outcomes

This scope of this dissertation was two-fold:

1) To shift from using engineering simulation as a black box for analysing case by case scenarios or stochastic search algorithms, towards learning about the cause-effect relationships between multiple design parameters and simulation responses, to assist intelligent decisions in the early stages of design.
2) To provide a broader representation of the design space to facilitate the capacity for flexible decisions in the early stages of design.

Together, 1) and 2) aim to improve decision-making through richer knowledge-gain, while facilitate flexible decision-making in the early stages of design. We argued that a comprehensive understanding of the design space combined with the capacity to make soft decisions, provide the ingredients to assist yet allow, a designer's intuition to control the design convergence in the early stages.

Consequently, we asked the following research questions: How can we represent design-analysis systems to assist architects and engineers with making intelligent decisions when considering multiple parameters? How can we represent design-analysis systems to provide the capacity for flexible choices in the early stages of design?

In a holistic response to address 1) and 2), we presented a probabilistic metamodeling approach. We argued that the task of 'decision-making over high-dimensional design spaces while maintaining the flexibility of the parameters', is analogous to problem of 'reasoning over uncertainty', which is a well-studied task in the field of applied probability.

The task of decision-making is a cognitive task involving reasoning with insights and information. Therefore, in this analogous context, we viewed the probabilistic representation of uncertainty in the field of statistics, as a suitable host for a broader representation of a design space. Now, in probability, reasoning over uncertainty is approached by means of a probabilistic model, which

is achieved by approximating the inputs and outputs into a joint probability distribution (JPD). Subsequently, a probabilistic representation of the problem facilitates the use of probabilistic inference techniques such as Bayes' theorem, which are useful to dissect the JPD and learn about the cause and effect dependencies between probability distributions (i.e. probabilistic relationships).

In this context, we presented the notion of a probabilistic metamodel that uses a Bayesian network to approximate a JPD between simulation inputs and outputs, as opposed to the typical functional metamodels. Hence, the term Bayesian network metamodel (BNM). The notion of a BNM is novel to the field of building design and somewhat very recent to the metamodeling community.

Bayesian networks, are good at representing large join probability distributions efficiently and subsequently good at computing probabilistic inference over multiple variables. JPD does not distinguish between inputs and outputs and thus, allows bi-directional inference between the two. We took advantage of the indifference: 1) to reason about cause and effect, bi-directionally by computing inputs→ outputs and output→ input, and 2) to enable the consideration of multiple outputs in the same metamodel.

Through the case studies in Chapters 4 and 6, we demonstrated how the introduction of a BNM in the environment of a computational design system, can enable reasoning over multiple input and output probability distributions via bi-directional inference; while maintaining a capacity for flexible choices through the probabilistic representation of the inputs and outputs. Furthermore, we illustrated how the notion of inference in a BNM implies better decision making — more specifically, we intend that a consolidated representation of the dependencies enables practitioners to take into account the entire network of relationships between all inputs and outputs, while making specific inferences, which would otherwise prove to be very challenging through cognitive means. This offers a significant advantage over typical use of simulation in computational design systems, and also over other metamodeling techniques.

It is not intended for the presented BNM approach to be discussed as an alternative to optimisation. On the contrary, in this research we foresee the knowledge-driven approach to form part of a family of tools that can be used in a sequential workflow throughout the building-design process. For example, the inverse probabilistic inference capability of a BNM, could be useful to narrow down a vague design space into the meaningful regions of interest and subsequently, employ optimisation search to search for the best solution or set of solutions within these regions.

7.2 Challenges

The case study application in Chapter 6 successfully illustrated the versatility of BNM applications; to investigate important relationships driving physical behaviour, to explore trade-offs between responses, to translate constraints on engineering behaviour into constraints on architectural parameters, and finally a mechanism to communicate soft engineering feedback to the architecture team instead of singular outcomes.

On the other hand, the case study flagged a number of important concerns when utilising a BNM approach in a realistic project scenario; mainly, challenges with discretisation, the quality of selection of parameters. Each of the challenges and potential approaches were discussed at the end of Chapter 6.

These challenges are subject to further research, as outlined in more detail, in Section Figure 7.3.

7.3 Envisaged goal

The BNM approach presented and discussed in this dissertation serves as a proof of concept in the context of a greater scheme which aims to generate and communicate intelligent abstractions of computational design systems. The ultimate goal is to achieve a robust workflow between data generation, statistical modelling and interactive visualisation.

We foresee three key challenges that need to be addressed to increase the relevance of the presented BNM approach for real-world applications: A) time compression to build a BNM, B) prediction of field-output, and C) communication of statistical insight through visualisation in the geometric environment.

A, B and C are intended to contribute to a robust BNM schema. Having said that, in future work we intend to address these challenges as three independent research questions, where each has potential for novel contributions. In more detail:

A. **Time compression: how can we compress the time to build a robust BNM**
    a. eliminate redundant inputs using sensitivity analysis to reduce amount of data required hence, compress time to build metamodel, or
    b. build a reusable metamodel by selecting engineering features that are independent of the input design parameters. This way simulation data is accumulated from multiple

analyses by carrying forward the same BNM throughout different iterations of a project, instead of constructing independent BNMs for each scenario.

B. **Field-output: can we adapt the BNM to handle bi-directional inference with field-outputs?**
    a. apply dimension reduction techniques on the field output

C. **Visualisation: how can we communicate statistical insight intuitively?**
    a. visualising the joint probability distribution in the form of a spatial probability field (using uncertainty visualisation techniques)

The following sections provide a more detailed description of research questions A, B and C in the form of preliminary studies and hypotheses. In future research we intend to investigate ways of addressing B and C simultaneously, where nodal elements of the response field can be represented as vectors in a probability spatial field.

7.3.1 Time compression

Bayesian network metamodels require significant amount data to ensure a robust metamodel. It may become a cumbersome task to build a BNM when one numerical simulation run is computationally demanding to compute. Despite strides made in computing power and technology, we propose two potential ways to improve time compression due to simulation data generation: by reducing the input dimensions or by developing a novel approach to build a metamodel that can recycle simulation data from one analysis scenario to the next.

7.3.1.1 Input dimension-reduction (data)

In general, the amount of data required to build a reliable metamodel is directly related with the type and number of metamodel input parameters being considered. The same applies to Bayesian network metamodels. One way of reducing the number of data points required, is by reducing the number of metamodel inputs, based on their sensitivity in influencing the metamodel response.

The quality of a metamodel to capture physical behaviour well, is directly dependant on the selected metamodel inputs. However, not all inputs are significantly important for explaining some output behaviour. In other words, the response of some physical or numerical system, may be sensitive to certain inputs and less or not sensitive to others. This may be the case in when

implementing Bayesian network metamodels for design; the thought process for rationalising architectural geometry into a selection of parameters, might not take into account the inputs required to describe the physical behaviour of the artefact. In such cases, certain inputs may become redundant when introducing them as metamodel inputs.

We can identify these redundant variables by means of sensitivity analysis. The hypothesis is therefore, to introduce sensitivity analysis as a design parameter screening method, prior to introducing them as BNM inputs. The challenge however is that the reliability of thorough sensitivity analysis methods such as 'variance decomposition' (Sobol', 1990), rely on vast amount of data, which contradicts the scope of employing the screening method in the first place.

In the field of experimental design methods, there exist SA methods focusing on screening of inputs with sparse data. These include Factorial or Fractional Designs (Fisher, 1935), and Morris method (Morris, 1991). However, these methods focus on first order sensitivities, where interactions between inputs are not taken into account. This can become unreliable in cases where nonlinear input-output relationships exist. For these reasons in preliminary research, we adopted a method by the name of Multiplicative Dimension Reduction Method (MDRM), which enables the higher-order sensitivity analysis with significantly low number of data points. The MDRM is a recently developed method (Zhang, 2013) which generates input samples based on Gaussian weights as initial values, while holding the other variables constant. The method relies on a limited number of response data points from which not only are statistical moments such as mean and standard deviation calculated, but also the first order and global sensitivities estimated. For example, for a 5 variable problem, only 50 data points are required. Figure 7.1 illustrates a pre-elementary study of a structural analysis problem involving five geometric parameters. The study demonstrates the potential of the MDRM SA method where total sensitivities (first +higher order sensitivities) achieved using the MDRM method, compared closely to sensitivities achieved with significantly more data. More specifically, the MDRM method used 240 times less the amount of data when compared with a typical amount required by the Sobol variance decomposition method. Further details and applications of the MDRM method can be found at (Balomenos, Genikomsou, Polak, & Pandey, 2015; Raimbault, 2016; Zhang, 2013)

Type	num samples	ST_X1	ST_X2	ST_X3	ST_X4	ST_X5
Sobol	12000	0.61530717	0.06223576	0.37830496	0.06890139	0.07153294
MDRM	50	0.62248939	0.03806256	0.26614296	0.00962445	0.00962445

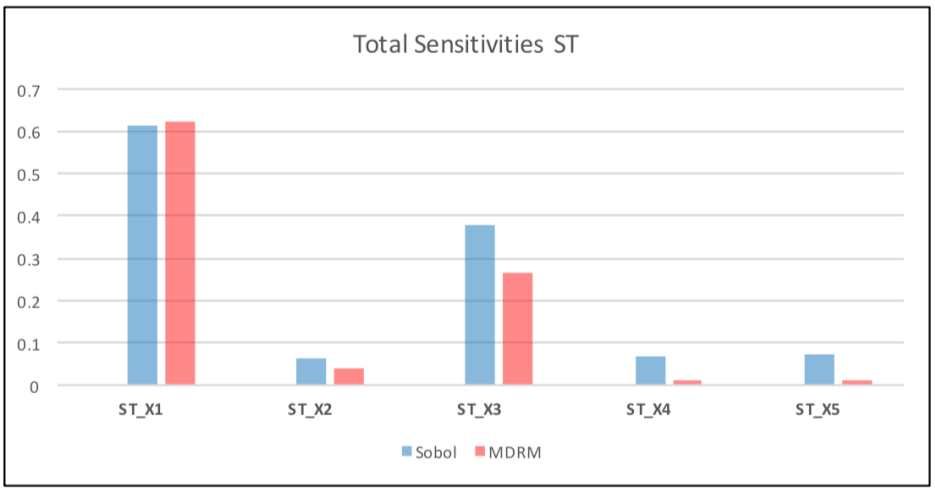


Figure 7.1: Preeliminary comparative study between Sobol SA (a variance decomposition method) and MDRM.

In future research we intend to investigate the robustness of this method further and implement it seamlessly into the BNM workflow presented in this research, as an automatic underlying method of reducing the number of input dimensions.

A further research question that needs to be tackled is how to deal with dimension reduction using MDRM SA for multiple response outputs? This might require doing separate SA analyses for each response, respectively.

7.3.1.2 Flexible metamodel inputs (data)

Another approach for addressing limitations due to amount of data required when building metamodels, is the notion of 'recycling' data from one metamodel to the next. In a preliminary study, we developed an approach to build a 'flexible simulation metamodel' whose inputs can generalise for different design problems. The hypothesis aims to carry forward the same metamodel from one design space to the next thus, build up on previous data.

In general, each time new design variables are introduced to a design-analysis system, a completely new metamodel has to be built from fresh simulation data. Thus, information learned about a design space is not carried forward from one metamodel to the next. Consequently, from a data perspective, useful data generated in previous simulation experiments is put to waste.

In response, we propose to build a flexible metamodel whose inputs can generalise for new design problems. We argue that since the domain of any mathematical model is bound by the inputs and output/s that characterise it, the ability for a metamodel to generalise for new problems is directly related to how generalised the selected inputs are. If we were to look underneath the hood of any engineering simulation tool, we would observe that inputs to the numerical analysis are not design problem dependent, but are generalised such that any simulation analysis model is described by one set of fundamental variables that are derived from domain-related theory and that are critical to calculating the response. Furthermore, these variables are computationally inexpensive to extract from any simulation analysis model. For example, in the finite element analysis (FEA) of different parametric truss models, the moment of inertia, mass, centre of gravity, axial and bending member stiffness, are crucial for calculating the stiffness matrix, while independent of the parametric description of each truss. While, it is not our scope to delve into the math underlying the numerical model, we hypothesise that if we can identify the set of variables that are critical to the math itself through domain expert help, and introduce them as input variables into the metamodel, we can build a flexible metamodel that can describe physical behaviour, independent of the variables that describe the design problem, thus can be carried forward from one design space to the next. Theoretically, the approach can be adopted with any metamodel.

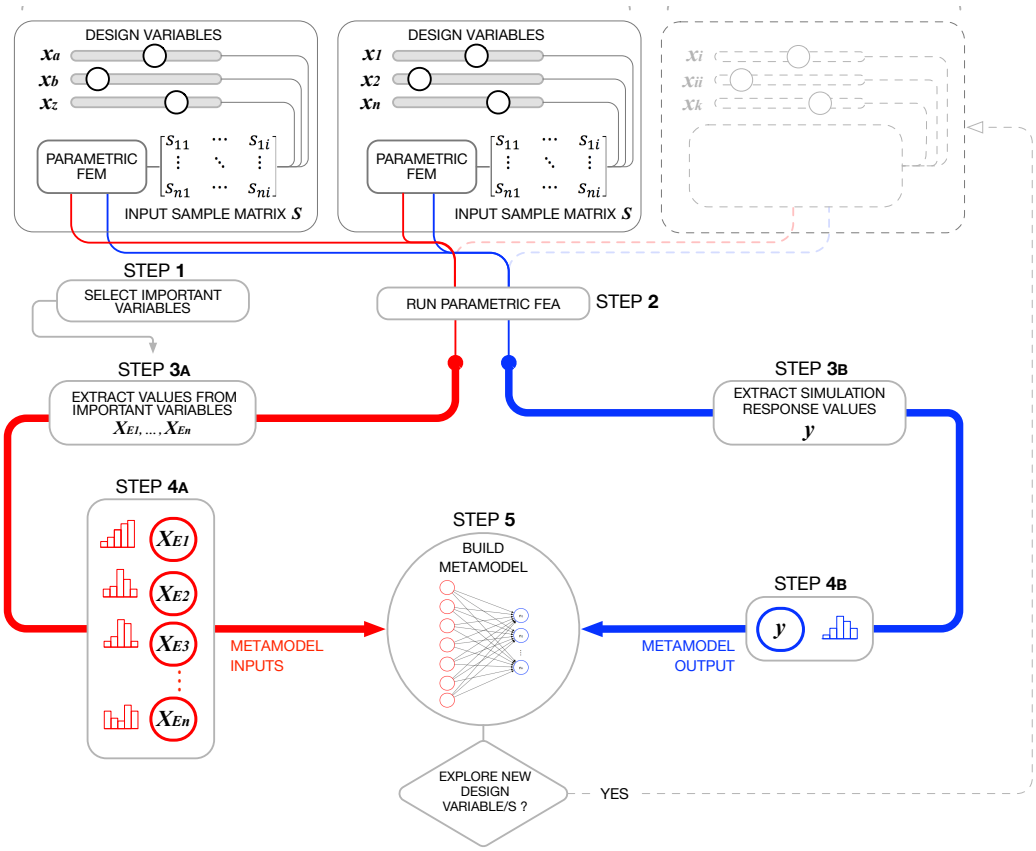


Figure 7.2: Workflow to build a flexible BNM, that can generalise for new design inputs.

In (Conti & Kaijima, 2018) we demonstrate this approach using a hypothetical case study scenario where three different parametric 2D cantilever truss designs are considered. The aim of the study is to first build the base metamodel from data generated with problem A. Subsequently, we test the generalizability of the base BNM by carrying forward the same metamodel to predict response and infer inputs, for two new truss design problems: B and C.

The core difference between building metamodels with this flexible approach, and the method presented in this dissertation, is the selection of the fundamental engineering variables as inputs instead of using the design parameters as inputs. In a recent publication (Conti & Kaijima, 2018), we published a preliminary case study example demonstrating the potential of this approach. More specifically, we identified $(EA/L)\cos\theta$, $(EA/L)\sin\theta$, $(EI/L)\cos\theta$, and $(EI/L)\sin\theta$ as fundamental variables and introduced them as inputs of the flexible metamodel. These variables are suitable because they are (i) critical to the calculation of the simulation response, and (ii) are independent of the design variables. Our scope is not to assemble the global stiffness matrix, but to select metamodel inputs. Therefore, we treat the values of each important variable as an accumulation of all the elements. Furthermore, we decide to also include additional variables; the *total mass* of the assembled structure and the center of gravity in x and y directions (*cogx* and *cogy*, respectively). Even though they are already implicitly considered in the bending and axial stiffness calculations, we argue that additional independent metamodel inputs might benefit the model to capture information that is being 'compressed' when accumulating stiffness values.

The values of $(EA/L)\cos\theta$, $(EA/L)\sin\theta$, $(EI/L)\cos\theta$, $(EI/L)\sin\theta$, *total mass*, *cogx* and *cogy*, are stored on each simulation run, together with the typical responses. Each time, new design variables are explored, the new generated data is concatenated with the previous dataset and the metamodel is then rebuilt, at no significant computational expense. Subsequently, all past and new continuous data is discretised (as required by Bayesian Networks), and introduced as probability distributions (nodes) in a Bayesian Network. Once the base metamodel is built, design variables are then mapped onto the metamodel inputs in the form of a secondary Bayesian Network such that the metamodel can then be used to predict response, and/or infer the design input distributions for a target response values of interest (Figure 7.3). The subsequent design problems are mapped onto the base model in the same way.

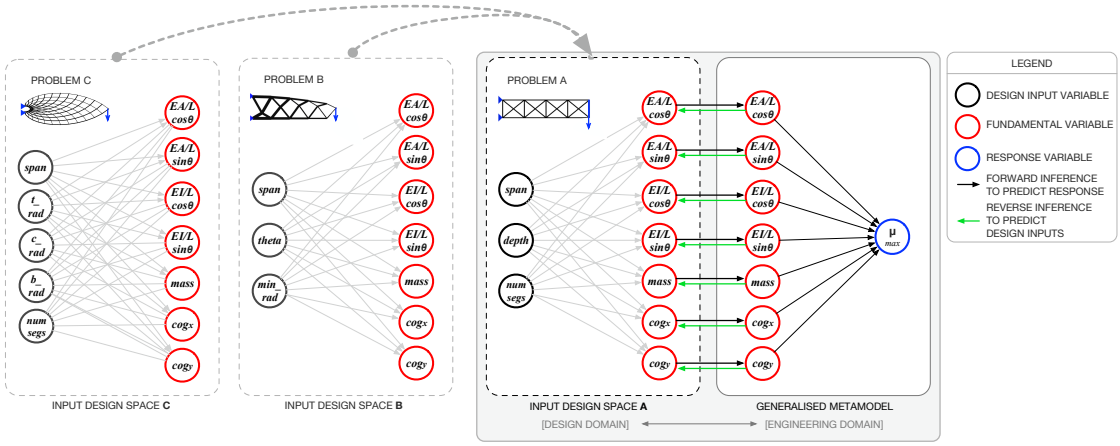


Figure 7.3: Workflow to map new design inputs onto the generalsied BNM (using the case study as an example).

In the case study we demonstrated that we can accumulate data from one problem to then next with the same metamodel, which in turn has the potential to reduce the amount of simulation data for subsequent design problems. The latter depends on the quality of the selected metamodel inputs however, this sub-hypothesis requires further investigation. In typical design and engineering offices, generated data is put away once a project is concluded and thus, lost. In this context, a framework for 'recycling' data urges the need to keep structured track of data in typical offices.

We argue that the same approach discussed for elastic FEA in the case study, should work for other FE-based methods given that they all involve the computation of the K matrix. Thus, we can identify these generalized variables for any type of FE-based analysis methods.

$$[K^{e_n}]\{u^{e_n}\} = \{F^{e_n}\}$$

	Property $[K]$	Behaviour $\{u\}$	Action $\{F\}$
Elastic	stiffness	displacement	force

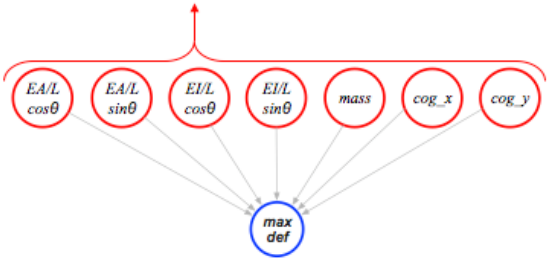


Figure 7.4: Identified generalizable variables fundamental to elastic FEA.

$$[K^{e_n}]\{u^{e_n}\} = \{F^{e_n}\}$$

	Property $[K]$	Behaviour $\{u\}$	Action $\{F\}$
Elastic	stiffness	displacement	force
Thermal	conductivity	temperature	heat source
Fluid	viscosity	velocity	body force

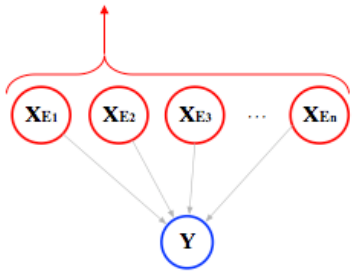


Figure 7.5: Generalisability of apprach for other FE-based analysis methods.

In future work, we intend to study the robustness of this approach to building metamodels by introducing more challenging problems. Furthermore, we would like to push the generalizability of the metamodel further such that it can generalise for different boundary, loading, and material scenarios. Furthermore, we would like to focus on smarter sampling strategies, such that we can predict more specifically where in the design space the samples should be mostly concentrated to avoid redundant sampling that is already captured form data of previous problems.

7.3.2 Finite element response field as metamodel outputs

The response of a numerical simulation model is typically generated in the form of a so-called 'field-output'. A field-output is an array of 'elements' or a field of responses. We are often interested in understanding local behaviour and thus, response field prove useful. However, metamodels in general are limited to a single output (except for neural networks), a single response is typically picked from the field of elemental responses based on a specific interest, for example the 'maximum value'. This may not provide any insight about local behaviour.

Despite the ability of BNMs to handle multiple outputs, it is computationally impractical to introduce all elemental responses from an analysis model as BNM outputs. This is especially the case with solid finite element analysis, whose field-outputs typically consist of thousands if not millions of elements. This challenge provokes a research question; how can we adapt BNM metamodels to handle and predict field-outputs?

A field of outputs can be thought of as a field of variables, where the output of each element in the field, can be considered as an independent response variable. We hypothesise about applying Dimension Reduction (DR) techniques on the field of variables to compress the variables into a significantly smaller, and more manageable, set of response variables. We intend to consider specifically two DR methods: Principal component analysis (PCA), and Auto-encoders.

PCA is a classic DR method that compresses (projects) multiple variables into a linear combination of principal components that are driving the output. In simpler terms, it reduces the redundancy of variables that do not contribute to an output. Labelling these outputs is in fact, very useful to understand the driving force behind a phenomenon. A significant advantage of PCA is that the original set of variables can be projected back from the compressed principal components. This approach is inspired from an application of PCA in car crash simulation (Schwarz, Ackert, & Mauermann, 2018). However, the linearity of PCA might not be suitable when dealing with nonlinearities in the field behaviour and thus miss out on the interesting regions of the underlying model. Furthermore, PCA does not handle random variables.

Alternatively, Auto-encoders are a type of artificial neural network that are self-supervised such that they can learn a compressed representation of input data. The multi-layered architecture of Auto-encoders follows that of neural networks, thus inherits the advantages of neural networks for capturing nonlinear behaviour. An example application of Auto-encoders to the prediction of structural behaviour in physical experiments can be found in Pathirage et al. (2018), while for numerical data from finite element analysis can be found in Roewer-Despres, Khan, and Stavness (2018).

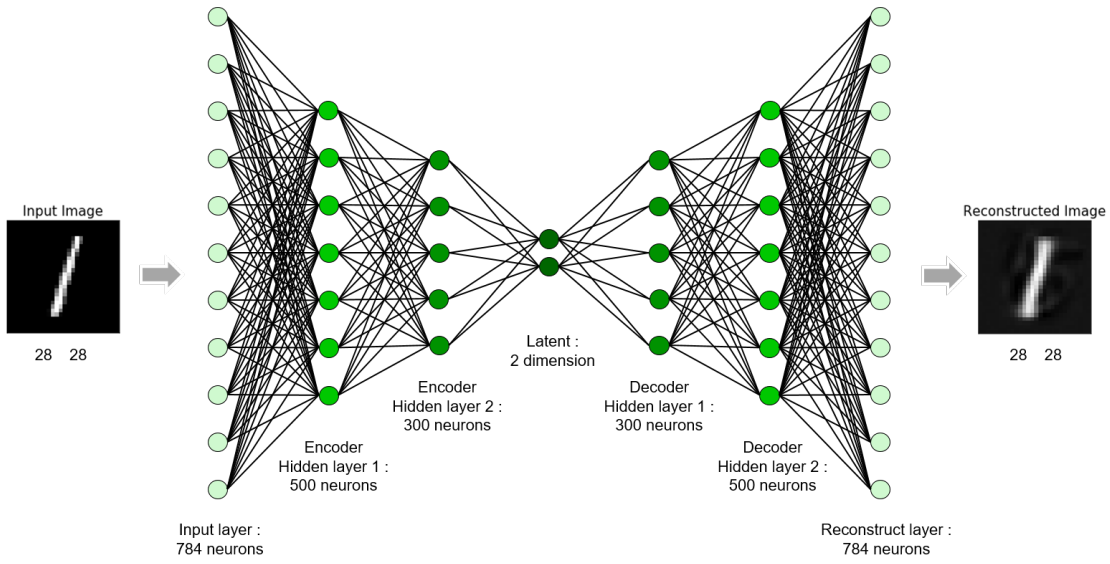


Figure 7.6: Example of an Auto-encoder to reconstruct hand-written numbers.
(source: Prof. Seungchul Lee, iSysems Design Lab)

In the field of machine learning, Auto-encoders are popular with image reconstruction applications. More specifically, an Auto-encoder can be fed with an image input and the network is able to learn *what* characterises the image. For example, Figure 7.6 illustrates the popular use of an Auto-encoder to learn how to reconstruct handwritten digits. The middle layer, also referred to as the bottle neck, represents a compressed representation of the full inputs (pixels in the case of images).

The bottleneck captures the latent characteristics that are necessary to construct a handwritten digit. Building a predictive model with these latent characteristics, implies that the model can interpolate for new digits.

Auto-encoders however, deal with singular outputs. Therefore, how do we adapt auto-encoders to BNMs, whose inputs and outputs are probabilistic? Recent variants of auto-encoders, namely, *variational auto-encoders* (VAE), allow probabilistic inputs and outputs. The application of VAEs to BNMs are subject to future research.

In future work we intend to analogise between the pixels of an image and the elements in the field output of a finite element mesh. We foresee a direct mapping between the two which opens up application possibilities of a vast wealth of documented expert knowledge in machine learning methods for building intelligent simulation metamodels.

7.3.3 Visualisation

An effective visualisation strategy plays a significant role in the validity of the BNM approach. In other words, it is necessary to communicate the insight gained from the inference exercises as intuitively as possible. Our initial attempt to display the input and output distributions directly in the parametric model serve as a stepping stone to a more sophisticated approach to effectively communicate insight as intuitively as possible.

More specifically, in future research we envision a workflow to map the inferred probability distributions as a probabilistic tensor field, where FEA mesh nodes correspond to vectors in the spatial field. This will entail collaboration with domains of ‘uncertainty visualisation’, which is dedicated field of research. We are inspired by uncertainty visualisation methods (Figure 7.7) such as ‘spatial probability fields’, used in typical applications such as meteorology (Figure 7.8).

Notable research in this area include (Pöthkow, 2015; Pöthkow & Hege, 2013; Junpeng Wang, Hazarika, Li, & Shen, 2018). Furthermore, these methods go beyond illustrative mapping of

probabilistic fields. A field representation enables geometric operations that can suggest new geometric outcomes; for example, extracting a mean isosurface from the field.

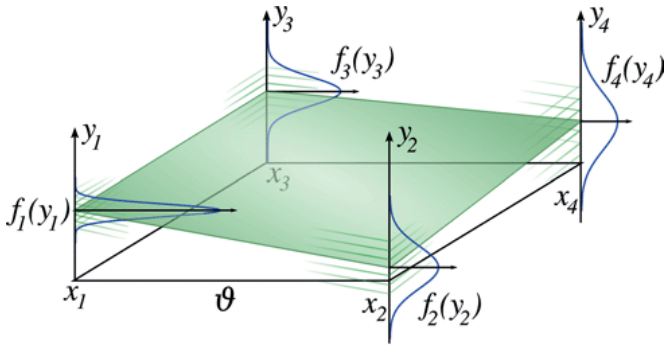


Figure 7.7: Example of a probability field in 2D(Pöthkow, Weber, & Hege, 2011).

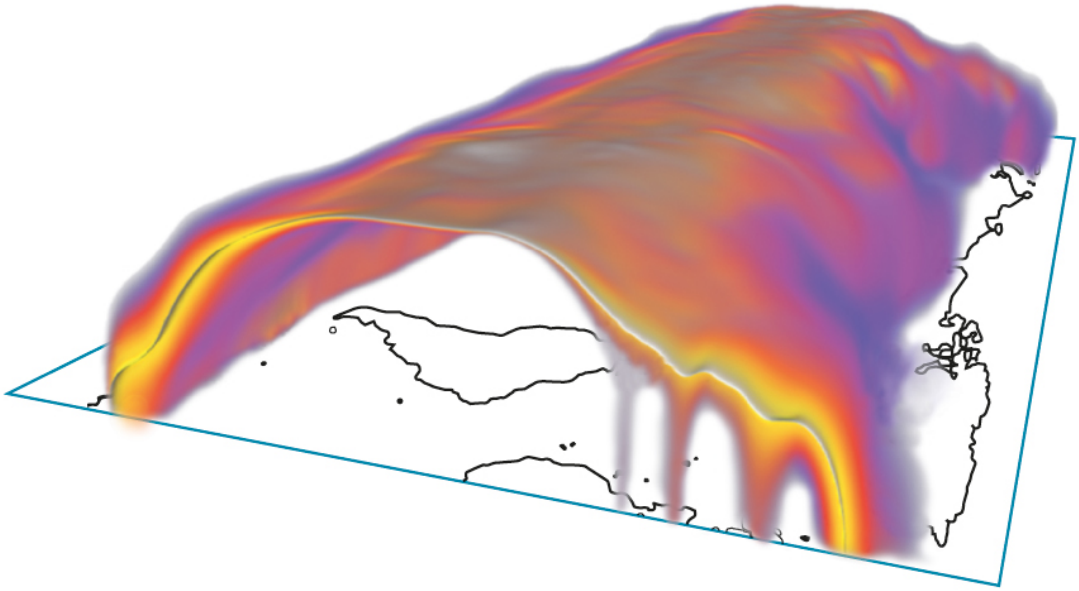


Figure 7.8: Color-coded spatial probabilities of the 0°C isotherm in the earth's atmosphere at some specific date. The white surface represents the isotherm of the ensemble average (Pöthkow et al., 2011).

7.3.4 Envisioned workflow

We envision that further development on recyclable simulation metamodels, field output prediction, and visualisation, can come together into a robust workflow (Figure 7.9) that can change and advance the way we utilise engineering tools for the design of buildings both in practice and in education.

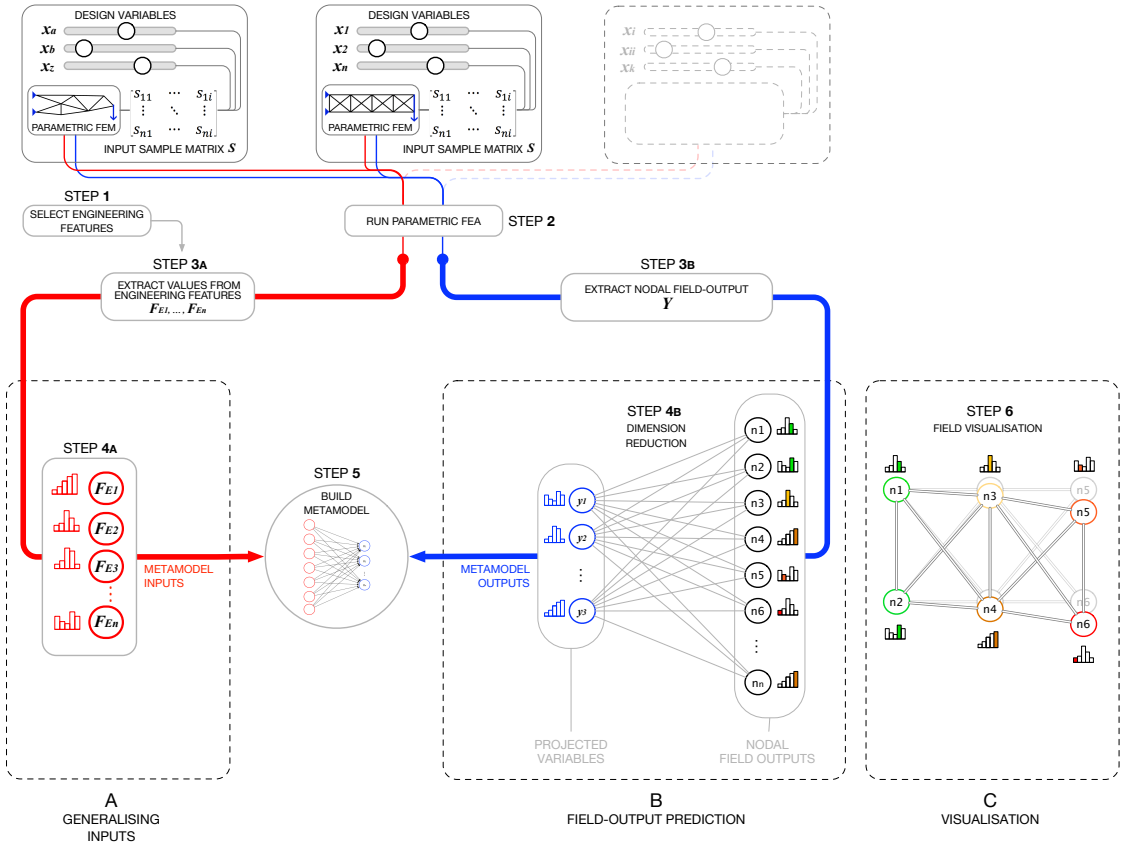


Figure 7.9: Envisaged workflow

BIBLIOGRAPHY

Acar, E. (2013). Effects of the correlation model, the trend model, and the number of training points on the accuracy of K riging metamodels. *Expert Systems, 30*(5), 418-428.

Adriaenssens, S., & Billington, D. P. (2013). Nervi's cantilevering stadium roofs: discipline of economy leads to inspiration. *Journal of International Association of Shell and Spatial Structures, 54*, 169-178.

Ankenman, B., Nelson, B. L., & Staum, J. (2010). Stochastic kriging for simulation metamodeling. *Operations research, 58*(2), 371-382.

Balomenos, G. P., Genikomsou, A. S., Polak, M. A., & Pandey, M. D. (2015). Efficient method for probabilistic finite element analysis with application to reinforced concrete slabs. *Engineering Structures, 103*, 85-101.

Barthelemy, J.-F., & Haftka, R. T. (1993). Approximation concepts for optimum structural design—a review. *Structural optimization, 5*(3), 129-144.

Barton, D. N., Kuikka, S., Varis, O., Uusitalo, L., Henriksen, H. J., Borsuk, M., . . . Linnell, J. D. C. (2012). Bayesian networks in environmental and resource management. *Integrated Environmental Assessment and Management, 8*(3), 418-429. doi:10.1002/ieam.1327

Barton, R. R. (1992). *Metamodels for simulation input-output relations*. Paper presented at the Proceedings of the 24th conference on Winter simulation.

Barton, R. R. (1998). *Simulation metamodels*. Paper presented at the Proceedings of the 30th conference on Winter simulation.

Bayes, T. (1763). A letter from the late Reverend Mr. Thomas Bayes, FRS to John Canton, MA and FRS. *Philosophical Transactions (1683-1775), 53*, 269-271.

Bennett, N. D., Croke, B. F., Guariso, G., Guillaume, J. H., Hamilton, S. H., Jakeman, A. J., . . . Perrin, C. (2013). Characterising performance of environmental models. *Environmental Modelling & Software, 40*, 1-20.

Biles, W. E., Kleijnen, J. P., Van Beers, W., & Van Nieuwenhuyse, I. (2007). *Kriging metamodeling in constrained simulation optimization: an explorative study*. Paper presented at the Simulation Conference, 2007 Winter.

Binder, J., Koller, D., Russell, S., & Kanazawa, K. (1997). Adaptive probabilistic networks with hidden variables. *Machine Learning, 29*(2-3), 213-244.

Booker, A. J., Dennis Jr, J., Frank, P. D., Serafini, D. B., Torczon, V., & Trosset, M. W. (1999). A rigorous framework for optimization of expensive functions by surrogates. *Structural optimization, 17*(1), 1-13.

Box, G. E., Hunter, W. G., & Hunter, J. S. (1978). Statistics for experimenters.

Buntine, W. (1996a). Graphical models for discovering knowledge. *Advances in knowledge discovery and data mining*, 59-82.

Buntine, W. (1996b). A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on knowledge and data engineering, 8*(2), 195-210.

Buntine, W. L. (1995). *Chain graphs for learning*. Paper presented at the Proceedings of the Eleventh conference on Uncertainty in artificial intelligence.

Burdick, D. S., & Naylor, T. H. (1966). Design of computer simulation experiments for industrial systems. *Communications of the ACM, 9*(5), 329-339.

Capozzoli, A., Mechri, H. E., & Corrado, V. (2009). Impacts of architectural design choices on building energy performance applications of uncertainty and sensitivity techniques. *International Building Performance Simulation Association*.

Chapman, W. L., Welch, W. J., Bowman, K. P., Sacks, J., & Walsh, J. E. (1994). Arctic sea ice variability: Model sensitivities and a multidecadal simulation. *Journal of Geophysical Research: Oceans, 99*(C1), 919-935.

Chen, V. C., Tsui, K.-L., Barton, R. R., & Meckesheimer, M. (2006). A review on design, modeling and applications of computer experiments. *IIE transactions, 38*(4), 273-291.

Conrady, S., & Jouffe, L. (2010). BayesiaLab. USA: Bayesia USA.

Conti, Z. X., & Kaijima, S. (2018). A Flexible Simulation Metamodel for Exploring Multiple Design Spaces.

Costa, A., & Nannicini, G. (2018). RBFOpt: an open-source library for black-box optimization with costly function evaluations. *Mathematical Programming Computation, 10*(4), 597-629.

Costa, A., Nannicini, G., Schroepfer, T., & Wortmann, T. (2015). Black-box optimization of lighting simulation in architectural design *Complex Systems Design & Management Asia* (pp. 27-39): Springer.

CyberPoint International, L. (2012). libpgm - Python library for Probabalistic Graphical Models.

Daly, R., Shen, Q., & Aitken, S. (2011). Learning Bayesian networks: approaches and issues. *The knowledge engineering review, 26*(2), 99-157.

De Boor, C., & Ron, A. (1990). On multivariate polynomial interpolation. *Constructive Approximation, 6*(3), 287-302.

De Grassi, M., & Naticchia, B. (2001). Modelling environmental complexity for sustainable design practice *Towards Sustainable Building* (pp. 135-160): Springer.

Dyn, N., Levin, D., & Rippa, S. (1986). Numerical procedures for surface fitting of scattered data by radial functions. *SIAM Journal on Scientific and Statistical Computing, 7*(2), 639-659.

Ellis, B., & Wong, W. H. (2008). Learning causal Bayesian network structures from experimental data. *Journal of the American Statistical Association, 103*(482), 778-789.

Euler, L. (1741). Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, 128-140.

Fang, H., & Horstemeyer, M. F. (2006). Global response approximation with radial basis functions. *Engineering optimization, 38*(04), 407-424.

Fang, K.-T., Li, R., & Sudjianto, A. (2005). *Design and modeling for computer experiments*: CRC Press.

Fisher, R. A. (1935). The Design of Experiments.

Fonseca, D., Navaresse, D., & Moynihan, G. (2003). Simulation metamodeling through artificial neural networks. *Engineering Applications of Artificial Intelligence, 16*(3), 177-183.

Forbus, K. D. (1988). Qualitative physics: Past, present, and future *Exploring artificial intelligence* (pp. 239-296): Elsevier.

Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics, 19*(1), 1-67.

Funahashi, K.-I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks, 2*(3), 183-192. doi:https://doi.org/10.1016/0893-6080(89)90003-8

Geman, S., & Geman, D. (1987). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images *Readings in computer vision* (pp. 564-584): Elsevier.

Gengembre, E., Ladevie, B., Fudym, O., & Thuillier, A. (2012). A Kriging constrained efficient global optimization approach applied to low-energy building design problems. *Inverse Problems in Science and Engineering, 20*(7), 1101-1114.

Giunta, A., & Watson, L. (1998). *A comparison of approximation modeling techniques-Polynomial versus interpolating models.* Paper presented at the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization.

Grossberg, S. (1988). Nonlinear neural networks: Principles, mechanisms, and architectures. *Neural Networks, 1*(1), 17-61.

Hamad, H., Al-Zaben, A., & Owies, R. (2014). Interpretability and variability of metamodel validation statistics in engineering system design optimization: a practical study. *International Journal for Simulation and Multidisciplinary Design Optimization, 5*, A05.

Heckerman, D., Geiger, D., & Chickering, D. M. (1995). Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning, 20*(3), 197-243. doi:10.1023/a:1022623210503

Herman, J. (2014). Sensitivity Analysis Library (Salib). Retrieved from http://jdherman.github.io/SALib/

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks, 2*(5), 359-366. doi:https://doi.org/10.1016/0893-6080(89)90020-8

Hufschmidt, M. M., & Fiering, M. B. (1966). Simulation techniques for design of water-resource systems.

Hunter, J., & Naylor, T. H. (1970). Experimental designs for computer simulation experiments. *Management Science, 16*(7), 422-434.

Hygh, J. S., DeCarolis, J. F., Hill, D. B., & Ranjithan, S. R. (2012). Multivariate regression as an energy assessment tool in early building design. *Building and Environment, 57*, 165-175.

Jin, R., Chen, W., & Simpson, T. W. (2001). Comparative studies of metamodelling techniques under multiple modelling criteria. *Structural and Multidisciplinary Optimization, 23*(1), 1-13.

John C, O., Marius, P., Serhat, Y., & Anthony T, P. (1995). Computer-simulation surrogates for optimization: Application to trapezoidal ducts and axisymmetric bodies.

Jones, D. R., Schonlau, M., & Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization, 13*(4), 455-492.

Jordan, M. I. (2004). Graphical models. *Statistical Science*, 140-155.

Kadesch, R. R. (1997). *Problem Solving Across the Disciplines*: Prentice Hall.

Karkera, K. R. (2014). *Building Probabilistic Graphical Models with Python*: Packt Publishing.

Kilian, A. (2006). Design exploration through bidirectional modeling of constraints.

Kingman, J., Dudley, J. E. G., & Baptista, R. (2017). THE 2016 SERPENTINE PAVILION

A CASE STUDY IN LARGE-SCALE GFRP STRUCTURAL DESIGN AND ASSEMBLY. In A. Menges, B. O. B. Sheil, R. Glynn, & M. Skavara (Eds.), *Fabricate 2017* (pp. 138-145): UCL Press.

Kingman, J., Dudley, J. E. G., Baptista, R., Menges, A., Sheil, B. O. B., Glynn, R., & Skavara, M. (2017). THE 2016 SERPENTINE PAVILION

A CASE STUDY IN LARGE-SCALE GFRP STRUCTURAL DESIGN AND ASSEMBLY *Fabricate 2017* (pp. 138-145): UCL Press.

Kleijnen, J. P. (1998). Experimental design for sensitivity analysis, optimization, and validation of simulation models. *Handbook of simulation: Principles, methodology, advances, applications, and practice*, 173-223.

Kleijnen, J. P. (2008). *Design and analysis of simulation experiments* (Vol. 20): Springer.

Kleijnen, J. P. (2009). Kriging metamodeling in simulation: A review. *European Journal of Operational Research, 192*(3), 707-716.

Kleijnen, J. P., & Sargent, R. G. (2000). A methodology for fitting and validating metamodels in simulation. *European Journal of Operational Research, 120*(1), 14-29.

Kleijnen, J. P. C. (1986). *Statistical tools for simulation practitioners*: Marcel Dekker, Inc.

Klemm, K., Marks, W., & Klemm, A. J. (2000). Multicriteria optimisation of the building arrangement with application of numerical simulation. *Building and Environment, 35*(6), 537-544.

Koch, P. N., Simpson, T. W., Allen, J. K., & Mistree, F. (1999). Statistical approximations for multidisciplinary design optimization: the problem of size. *Journal of Aircraft, 36*(1), 275-286.

Kolarevic, B., & Malkawi, A. (2005). *Peformative Architecture*: Routledge.

Koller, D., Friedman, N., Getoor, L., & Taskar, B. (2007). 2 Graphical Models in a Nutshell.

Krige, D. G. (1951). *A statistical approach to some mine valuation and allied problems on the Witwatersrand.*

Krykova, I. (2003). *Evaluating of path-dependent securities with low discrepancy methods.* Worcester Polytechnic Institute.

Lauritzen, S. L., & Wermuth, N. (1989). Graphical models for associations between variables, some of which are qualitative and some quantitative. *The Annals of Statistics*, 31-57.

Law, A. M. (2015). *Simulation Modeling and Analysis*: McGraw-Hill.

Lee, A. (2014). pyDOE [computer program]. Retrieved from http://pythonhosted.org/pyDOE/

Levine, N. D. (2011). Using Minimum Description Length for Discretization Classification of Data Modeled by Bayesian Networks.

Liang, C., & Mahadevan, S. (2015). Reliability-based Multi-objective Optimization under Uncertainty.

Liao, W., & Ji, Q. (2009). Learning Bayesian network parameters under incomplete data with domain knowledge. *Pattern Recognition, 42*(11), 3046-3056.

Lucas, P. J., van der Gaag, L. C., & Abu-Hanna, A. (2004). Bayesian networks in biomedicine and health-care. *Artificial intelligence in medicine, 30*(3), 201-214.

Marcot, B. G., Holthausen, R. S., Raphael, M. G., Rowland, M. M., & Wisdom, M. J. (2001). Using Bayesian belief networks to evaluate fish and wildlife population viability under land management alternatives from an environmental impact statement. *Forest ecology and management, 153*(1), 29-42.

Martin, J. D., & Simpson, T. W. (2005). Use of kriging models to approximate deterministic computer models. *AIAA journal, 43*(4), 853-863.

McKay, M. D., Beckman, R. J., & Conover, W. J. (1979). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics, 42*(1), 55-61.

Michalatos, P., & Kaijima, S. (2014). Millipide (Grasshopper plugin for structural analysis).

Morris, M. D. (1991). Factorial Sampling Plans for Preliminary Computational Experiments. *Technometrics, 33*(2), 161-174. doi:10.1080/00401706.1991.10484804

Murphy, K. (2001). An introduction to graphical models.

Myers, R. H., Anderson-Cook, C. M., & Montgomery, D. C. (2014). *Wiley Series in Probability and Statistics : Response Surface Methodology : Process and Product Optimization Using Designed Experiments (3rd Edition)*. Somerset, NJ, USA: Wiley.

Myers, R. H., Montgomery, D. C., & Anderson-Cook, C. M. (1995). Response Surface Methodology. Process and Product Optimization Using Designed Experiments", John Willey & Sons. *Inc., New York, NY*, 134-174.

Naticchia, B. (1999). *Physical Knowledge in Patterns: Bayesian Network Models for Preliminary Design*. Paper presented at the Architectural Computing from Turing to 2000: 17th eCAADe Conference Proceedings, University of Liverpool, UK.

Nervi, P. L. (1955). *Structures*: Mc-Graw Hill.

Nonchev, B. S. (2015). Model Selection for Data Analysis Based on the MDL Principle.

Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K., & Mordvintsev, A. (2018). The building blocks of interpretability. *Distill, 3*(3), e10.

Panão, M. J. O., Gonçalves, H. J., & Ferrão, P. M. (2008). Optimization of the urban building efficiency potential for mid-latitude climates using a genetic algorithm approach. *Renewable Energy, 33*(5), 887-896.

Pathirage, C. S. N., Li, J., Li, L., Hao, H., Liu, W., & Ni, P. (2018). Structural damage identification based on autoencoder neural networks and deep learning. *Engineering Structures, 172*, 13-28. doi:https://doi.org/10.1016/j.engstruct.2018.05.109

Pearl, J., & Mackenzie, D. (2018). *The book of why: the new science of cause and effect*: Basic Books.

Pearl, J., & Russell, S. (1998). *Bayesian networks*: Computer Science Department, University of California.

Peippo, K., Lund, P., & Vartiainen, E. (1999). Multivariate optimization of design trade-offs for solar low energy buildings. *Energy and Buildings, 29*(2), 189-205.

Poropudas, J., & Virtanen, K. (2010a). Game-theoretic validation and analysis of air combat simulation models. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 40*(5), 1057-1070.

Poropudas, J., & Virtanen, K. (2010b). *Simulation metamodeling in continuous time using dynamic Bayesian networks.* Paper presented at the Proceedings of the Winter Simulation Conference.

Poropudas, J., & Virtanen, K. (2011). Simulation metamodeling with dynamic Bayesian networks. *European Journal of Operational Research, 214*(3), 644-655. doi:http://dx.doi.org/10.1016/j.ejor.2011.05.007

Pöthkow, K. (2015). *Modeling, Quantification and Visualization of Probabilistic Features in Fields with Uncertainties.* Retrieved from https://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/5790

http://www.diss.fu-berlin.de/diss/receive/FUDISS_thesis_000000099462?lang=en

Pöthkow, K., & Hege, H. C. (2013). *Nonparametric models for uncertainty visualization.* Paper presented at the Computer Graphics Forum.

Pöthkow, K., Weber, B., & Hege, H. C. (2011). *Probabilistic marching cubes.* Paper presented at the Computer Graphics Forum.

Pousi, J., Poropudas, J., & Virtanen, K. (2013). Simulation metamodelling with Bayesian networks. *Journal of Simulation, 7*(4), 297-311.

Preisinger, C., & Heimrath, M. (2014). Karamba—A toolkit for parametric structural design. *Structural Engineering International, 24*(2), 217-221.

Pronzato, L., & Müller, W. G. (2012). Design of computer experiments: space filling and beyond. *Statistics and Computing, 22*(3), 681-701.

Rai, R., & Campbell, M. I. (2006). *Qualitative and quantitative sequential sampling.* Paper presented at the ASME 2006 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference.

Raimbault, J. (2016). *Modelling Fatigue Deterioration and Retrofitting in Bridge Management Systems.* University of Waterloo.

Ritter, F., Schubert, G., Geyer, P., Borrmann, A., & Petzold, F. (2014). *Design Decision Support–Real-time Energy Simulation in the Early Design Stages.* Paper presented at the Computing in Civil and Building Engineering (2014).

Roewer-Despres, F., Khan, N., & Stavness, I. (2018). *Towards finite element simulation using deep learning.* Paper presented at the 15th International Symposium on Computer Methods in Biomechanics and Biomedical Engineering.

Rutten, D. (2012). Grasshopper: generative modeling for Rhino. *Computer software, Retrieved April, 29*, 2012.

Sacks, J., Welch, W. J., Mitchell, T. J., & Wynn, H. P. (1989a). Design and Analysis of Computer Experiments. *Statistical Science, 4*(4), 409-423.

Sacks, J., Welch, W. J., Mitchell, T. J., & Wynn, H. P. (1989b). Design and analysis of computer experiments. *Statistical Science*, 409-423.

Saltelli, A., Chan, K., & Scott, E. M. (2000). *Sensitivity analysis* (Vol. 1): Wiley New York.

Sanchez, S. M. (2005). *Work smarter, not harder: guidelines for designing simulation experiments.* Paper presented at the Proceedings of the 37th conference on Winter simulation.

Sankararaman, S., Ling, Y., & Mahadevan, S. (2015). Fatigue Crack Growth Analysis and Damage Prognosis in Structures. *Emerging Design Solutions in Structural Health Monitoring Systems*, 207.

Sasena, M. J. (2002). *Flexibility and efficiency enhancements for constrained global design optimization with kriging approximations.* University of Michigan Ann Arbor.

Schwarz, C., Ackert, P., & Mauermann, R. (2018). Principal component analysis and singular value decomposition used for a numerical sensitivity analysis of a complex drawn part. *The International Journal of Advanced Manufacturing Technology, 94*(5), 2255-2265. doi:10.1007/s00170-017-0980-z

Shea, K., Aish, R., & Gourtovaia, M. (2005). Towards integrated performance-driven generative design tools. *Automation in Construction, 14*(2), 253-264.

Simpson, T. W., Mauery, T. M., Korte, J. J., & Mistree, F. (2001). Kriging models for global approximation in simulation-based multidisciplinary design optimization. *AIAA journal, 39*(12), 2233-2241.

Simpson, T. W., Peplinski, J., Koch, P. N., & Allen, J. K. (1997). On the use of statistics in design and the implications for deterministic computer experiments. *Design Theory and Methodology-DTM'97*, 14-17.

Simpson, T. W., Poplinski, J., Koch, P. N., & Allen, J. K. (2001). Metamodels for computer-based engineering design: survey and recommendations. *Engineering with Computers, 17*(2), 129-150.

Sobieszczanski-Sobieski, J., & Haftka, R. T. (1997). Multidisciplinary aerospace design optimization: survey of recent developments. *Structural optimization, 14*(1), 1-23.

Sobol', I. y. M. (1990). On sensitivity estimation for nonlinear mathematical models. *Matematicheskoe Modelirovanie, 2*(1), 112-118.

Spiegelhalter, D. J. (1998). Bayesian graphical modelling: a case-study in monitoring health outcomes. *Journal of the Royal Statistical Society: Series C (Applied Statistics), 47*(1), 115-133.

Steck, H., & Tresp, V. (1999). *Bayesian belief networks for data mining.* Paper presented at the Proceedings of the 2. Workshop on Data Mining und Data Warehousing als Grundlage moderner entscheidungsunterstützender Systeme.

Stein, M. (1987). Large sample properties of simulations using Latin hypercube sampling. *Technometrics, 29*(2), 143-151.

Tresidder, E., Zhang, Y., & Forrester, A. I. (2012). Acceleration of building design optimisation through the use of kriging surrogate models. *Proceedings of building simulation and optimization*, 1-8.

Uusitalo, L. (2007). Advantages and challenges of Bayesian networks in environmental modelling. *Ecological modelling, 203*(3), 312-318.

Van Beers, W., & Kleijnen, J. P. (2004). *Kriging interpolation in simulation: a survey.* Paper presented at the Simulation Conference, 2004. Proceedings of the 2004 Winter.

Vang, J. (2017). PyBBN. *GitHub*.

Varadarajan, S., CHEN∗, W., & Pelka, C. J. (2000). Robust concept exploration of propulsion systems with enhanced model approximation capabilities. *Engineering Optimization+ A35, 32*(3), 309-334.

Viana, F. A., Simpson, T. W., Balabanov, V., & Toropov, V. (2014). Special section on multidisciplinary design optimization: metamodeling in multidisciplinary design optimization: how far have we really come? *AIAA journal, 52*(4), 670-690.

Walsh, J. E. (1963). Use of linearized nonlinear regression for simulations involving Monte Carlo. *Operations research, 11*(2), 228-235.

Wang, G. G., & Shan, S. (2007). Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design, 129*(4), 370-380.

Wang, J., Hazarika, S., Li, C., & Shen, H.-W. (2018). Visualization and visual analysis of ensemble data: A survey. *IEEE transactions on visualization and computer graphics*.

Wang, J., Zhai, Z. J., Jing, Y., & Zhang, C. (2010). Particle swarm optimization for redundant building cooling heating and power system. *Applied Energy, 87*(12), 3668-3679.

Zakerifar, M., Biles, W. E., & Evans, G. W. (2009). *Kriging metamodeling in multi-objective simulation optimization.* Paper presented at the Simulation Conference (WSC), Proceedings of the 2009 Winter.

Zhang, X. (2013). Efficient computational methods for structural reliability and global sensitivity analyses.

APPENDIX A

A.1 *d1* prediction validation of BNM in chapter 4

The following illustrate prediction error plots from a five-fold cross-validation for each target in the BNM: *maximum deflection*, and *weight*.

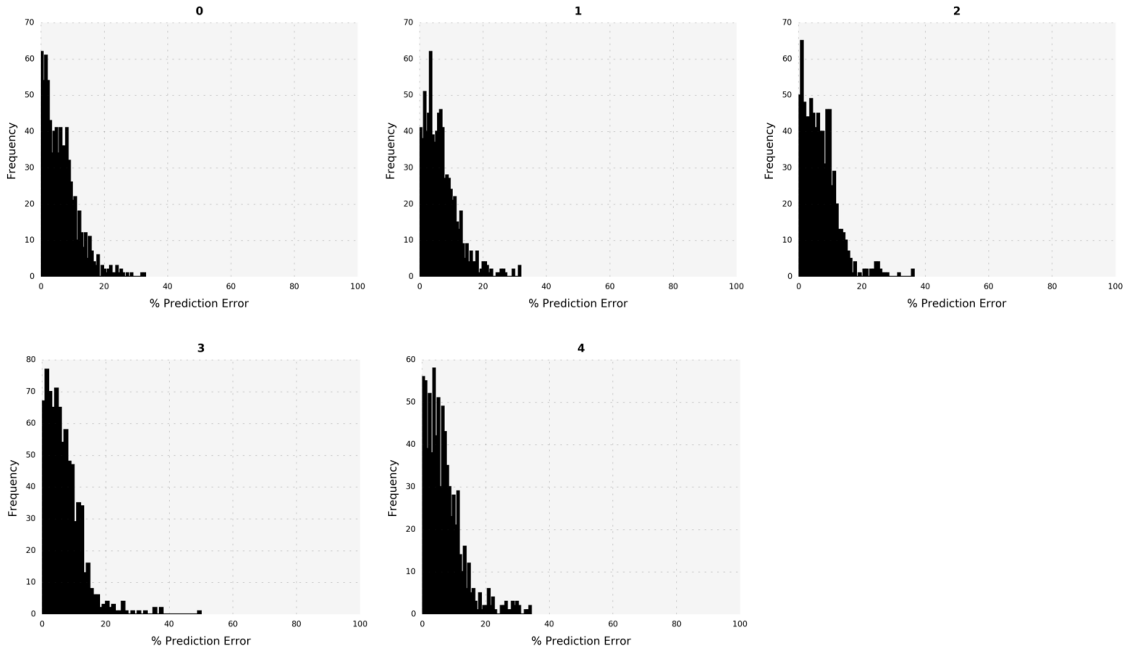


Figure A-1: Validation plots for five folds. TARGET: *maximum deflection*.

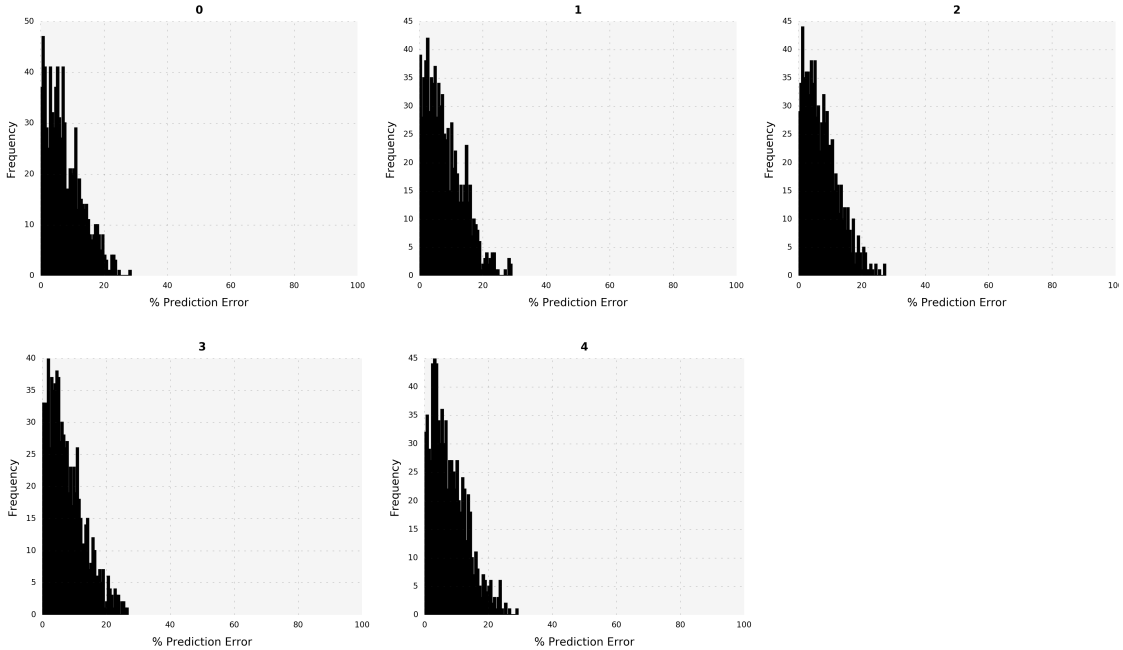


Figure A-2: Validation plots for five folds. TARGET: *weight*.

A.2 *d1* prediction validation of global BNM in Chapter 6, target: deflection

The following illustrate prediction error plots from a five-fold cross-validation for each deflection target corresponding to each pavilion cross-section: *s*[1/10/15/25/35/45]*_max_def_total*.

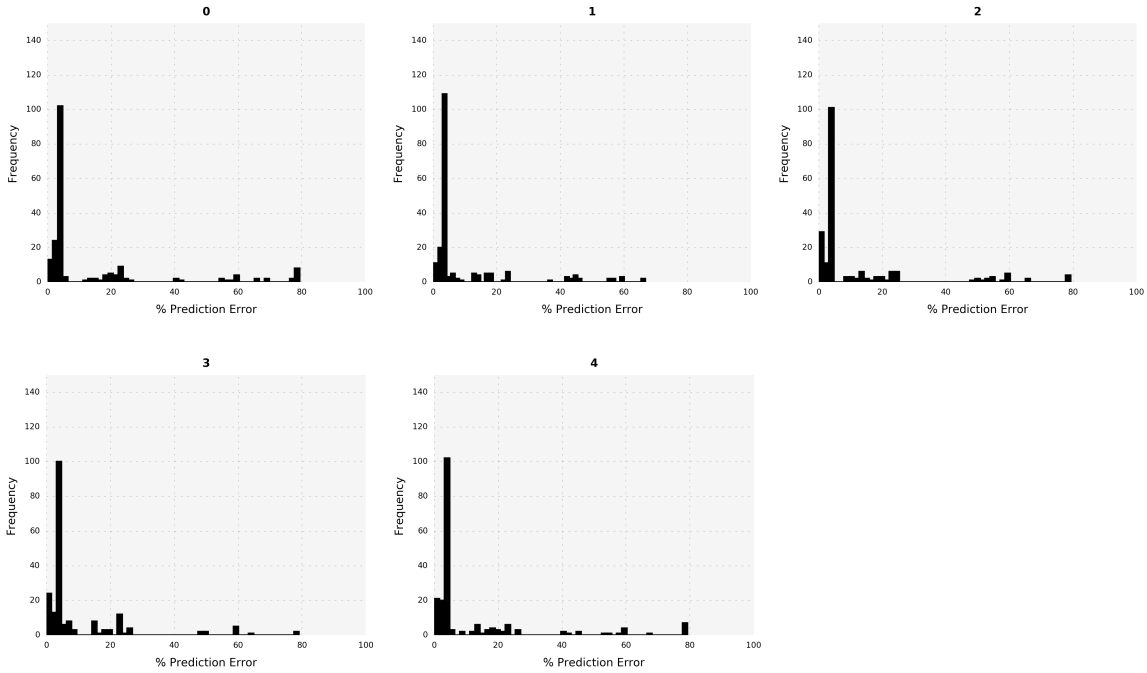


Figure A-3: Validation plots for five folds. TARGET: s1_max_def_total.

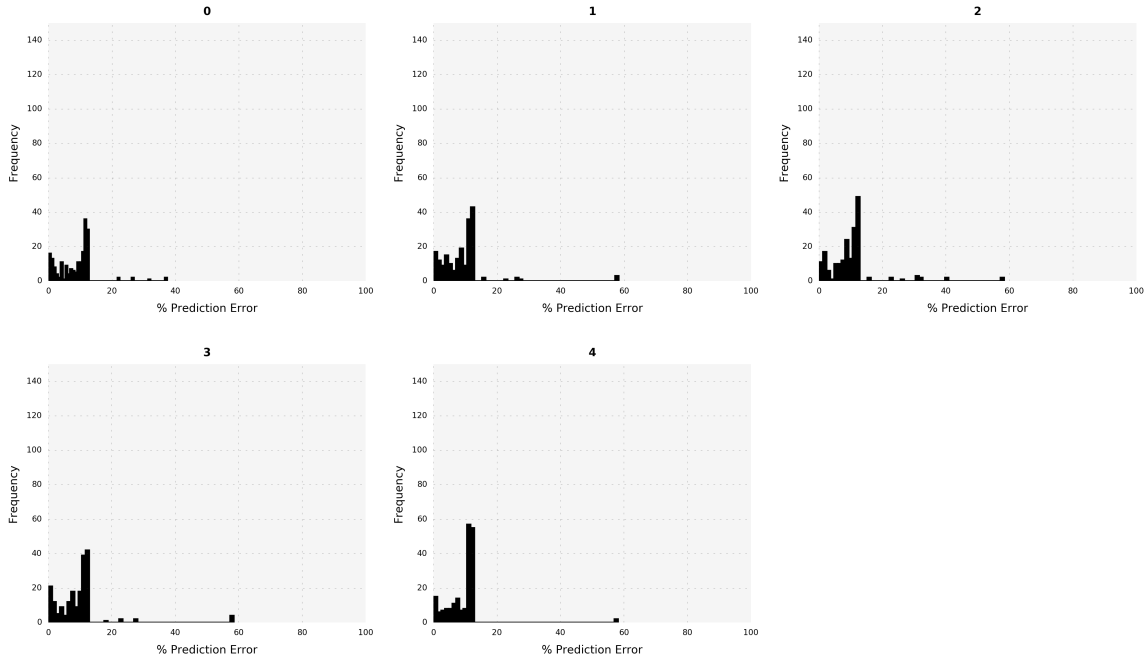


Figure A-4: Validation plots for five folds. TARGET: s10_max_def_total.

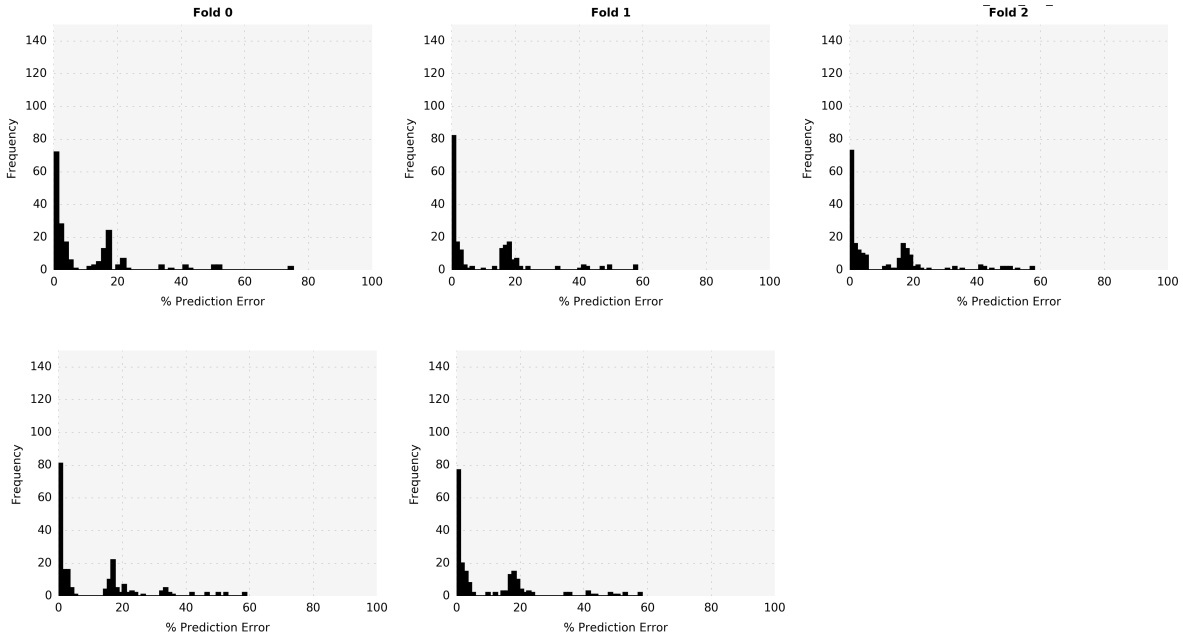


Figure A-5: Validation plots for five folds. TARGET: s15_max_def_total.

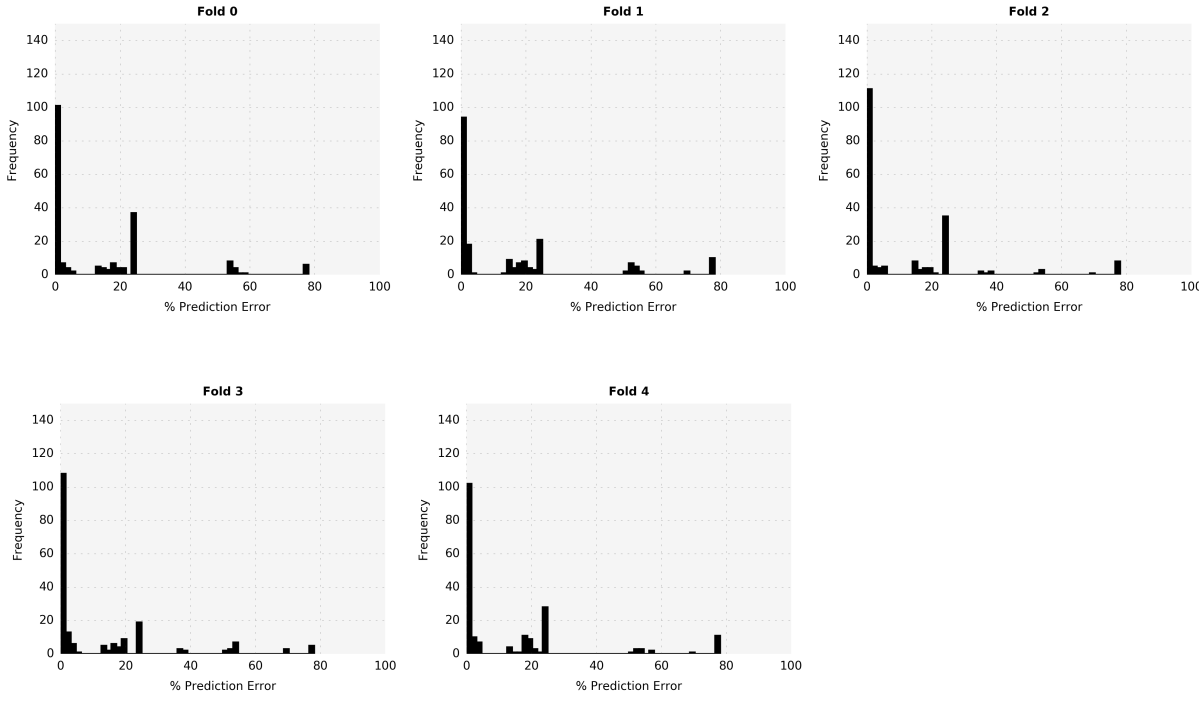


Figure A-6: Validation plots for five folds. TARGET: s25_max_def_total.

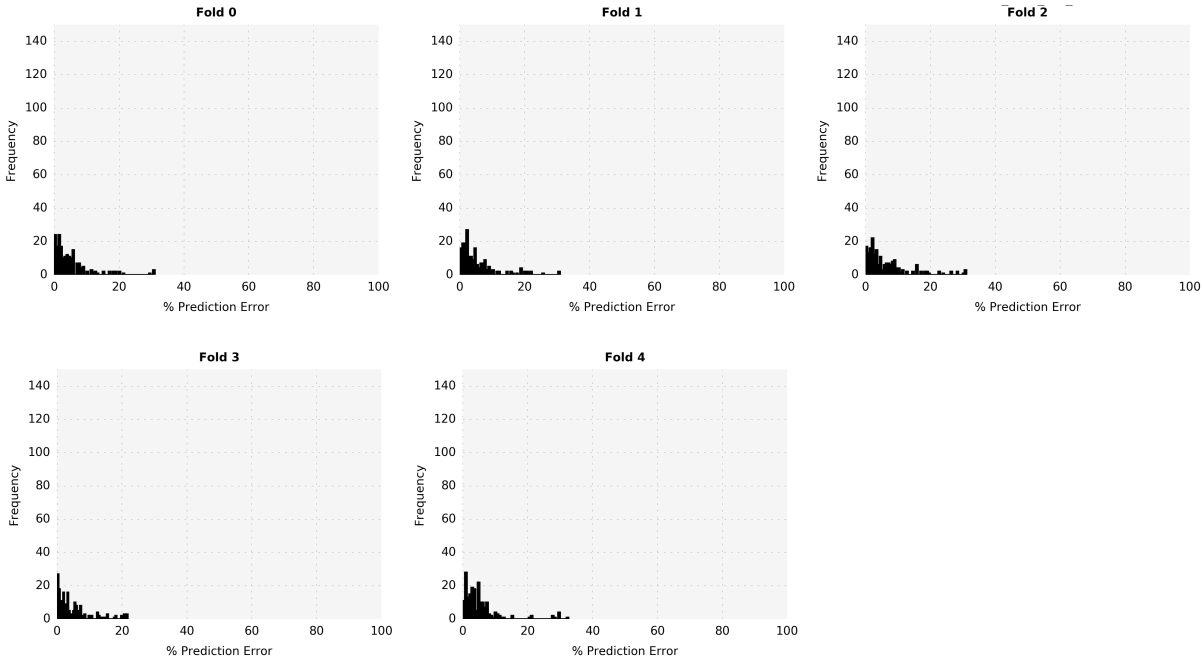


Figure A-7: Validation plots for five folds. TARGET: s35_max_def_total.

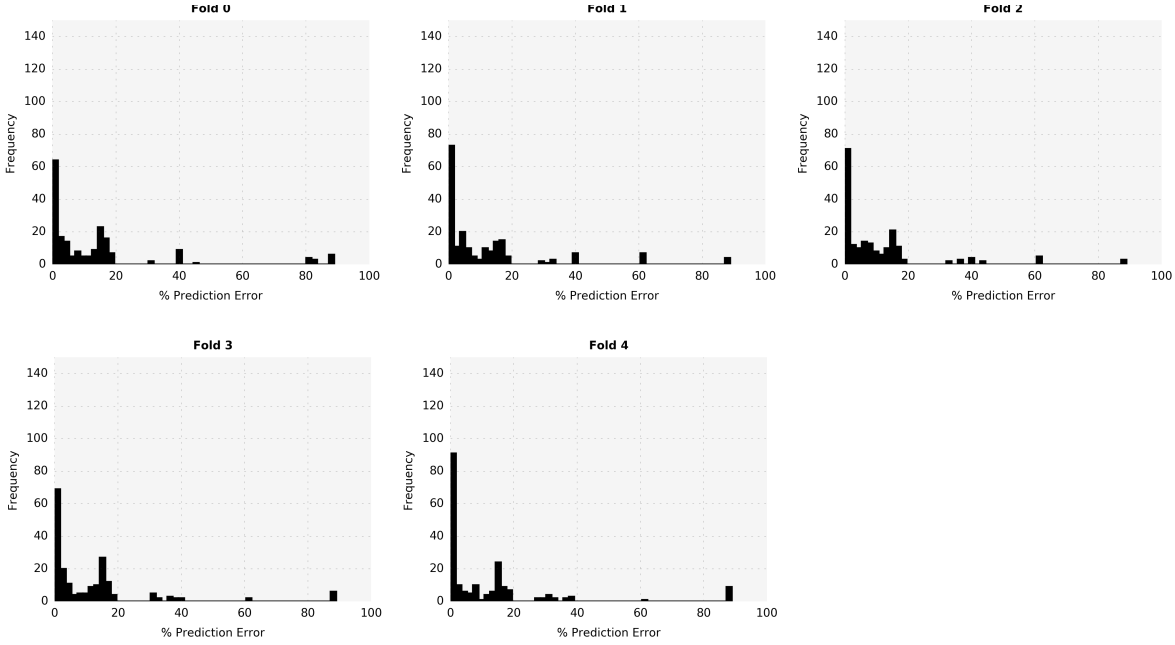


Figure A-8: Validation plots for five folds. TARGET: s45_max_def_total.

A.3 *d1* prediction validation of global BNM in Chapter 6, target: force

The following illustrate prediction error plots from a five-fold cross-validation for each force target corresponding to each pavilion cross-section: *s*[1/10/15/25/35/45]*_max_resolved_force.*

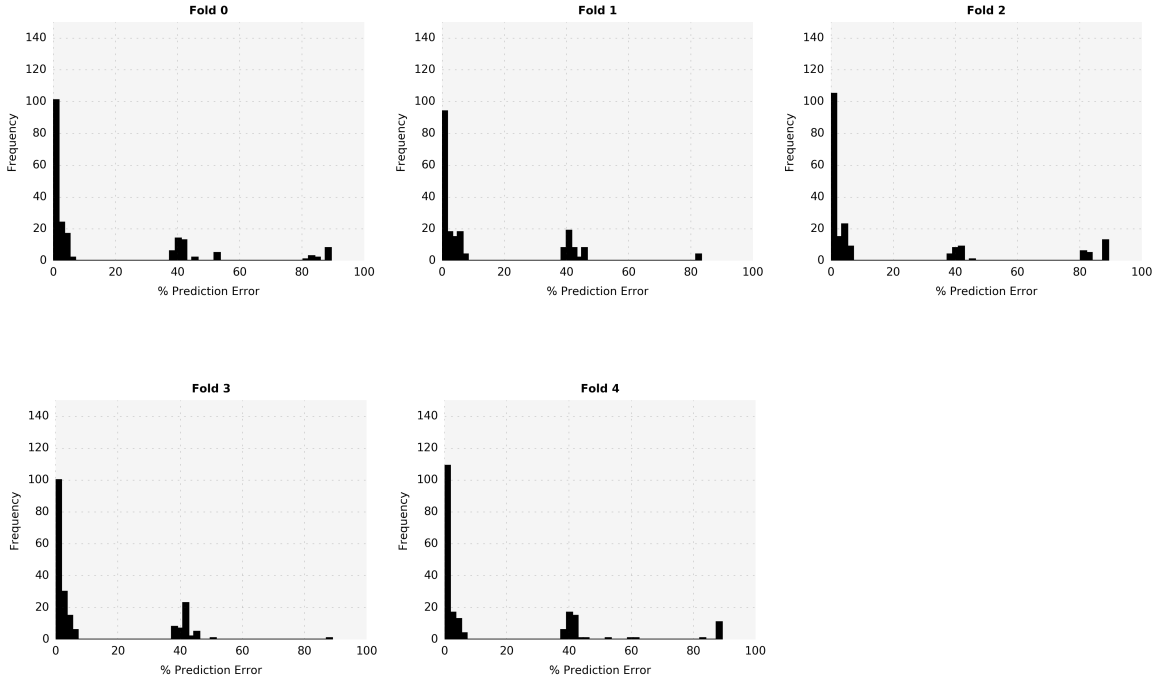


Figure A-9: Validation plots for five folds. TARGET: s1_max_resolved_force.

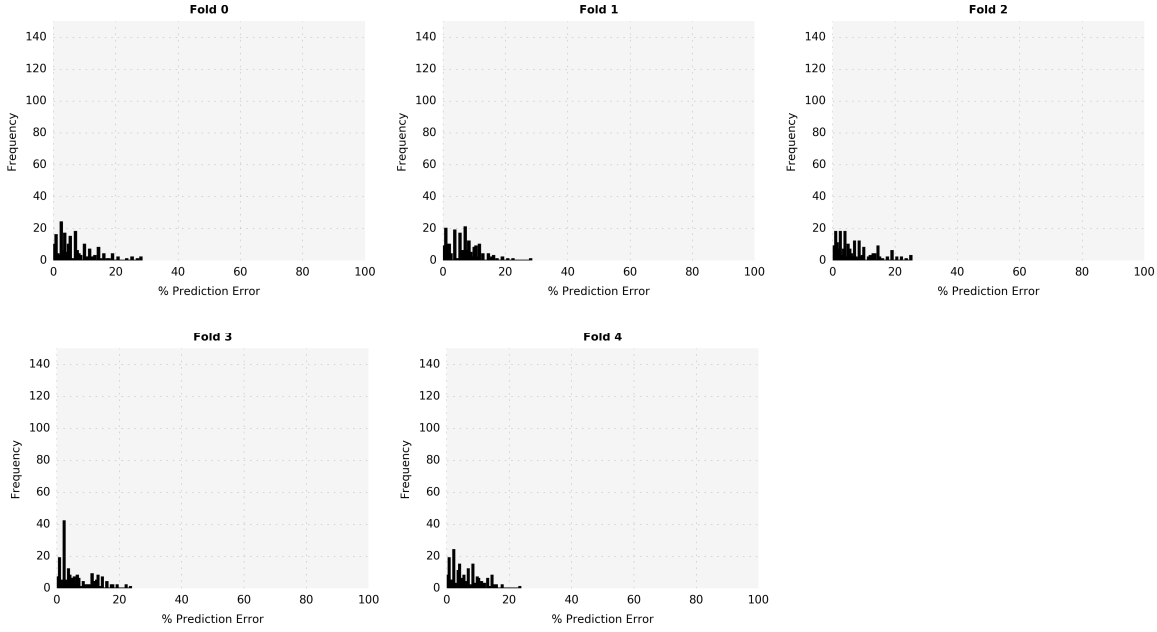


Figure A-10: Validation plots for five folds. TARGET: s10_max_resolved_force.

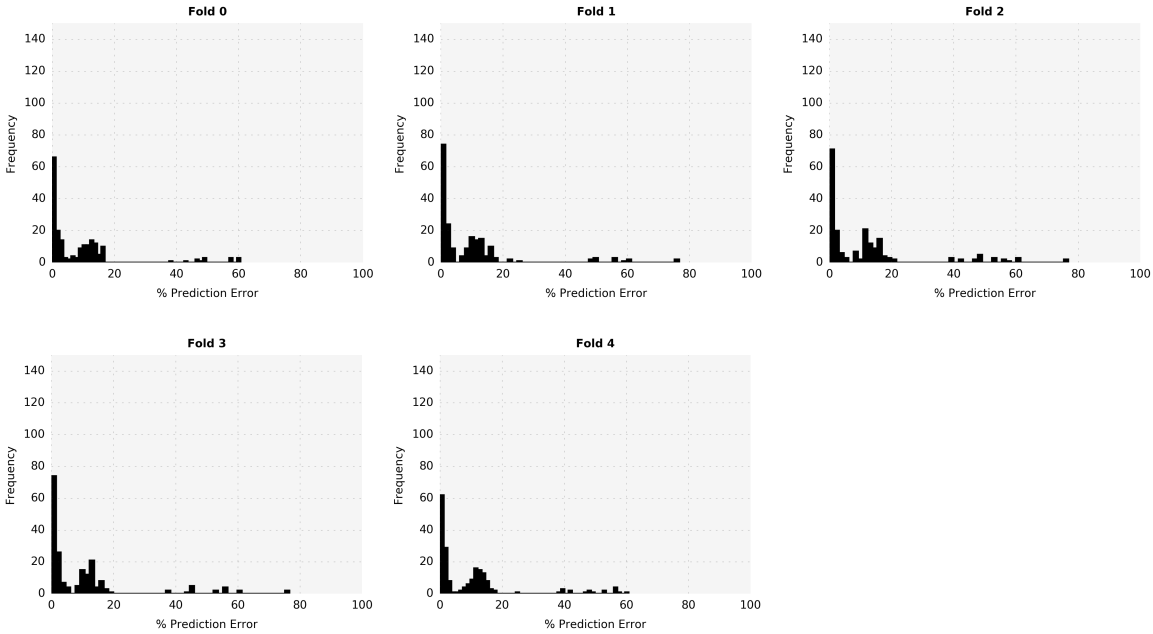


Figure A-11: Validation plots for five folds. TARGET: s15_max_resolved_force.

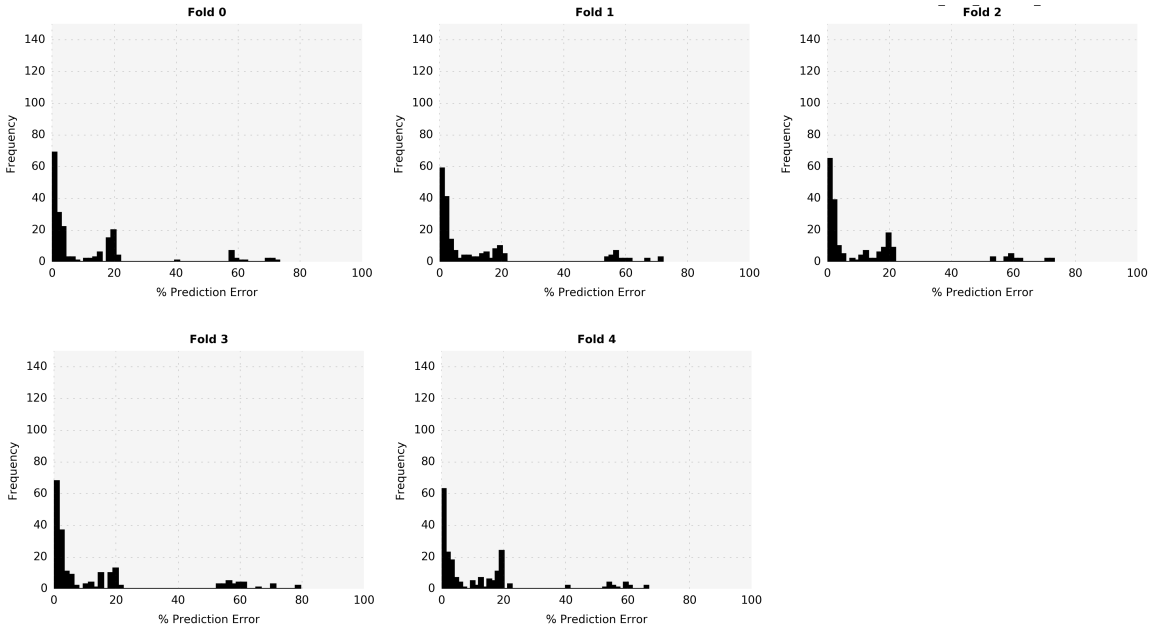


Figure A-12: Validation plots for five folds. TARGET: s25_max_resolved_force.

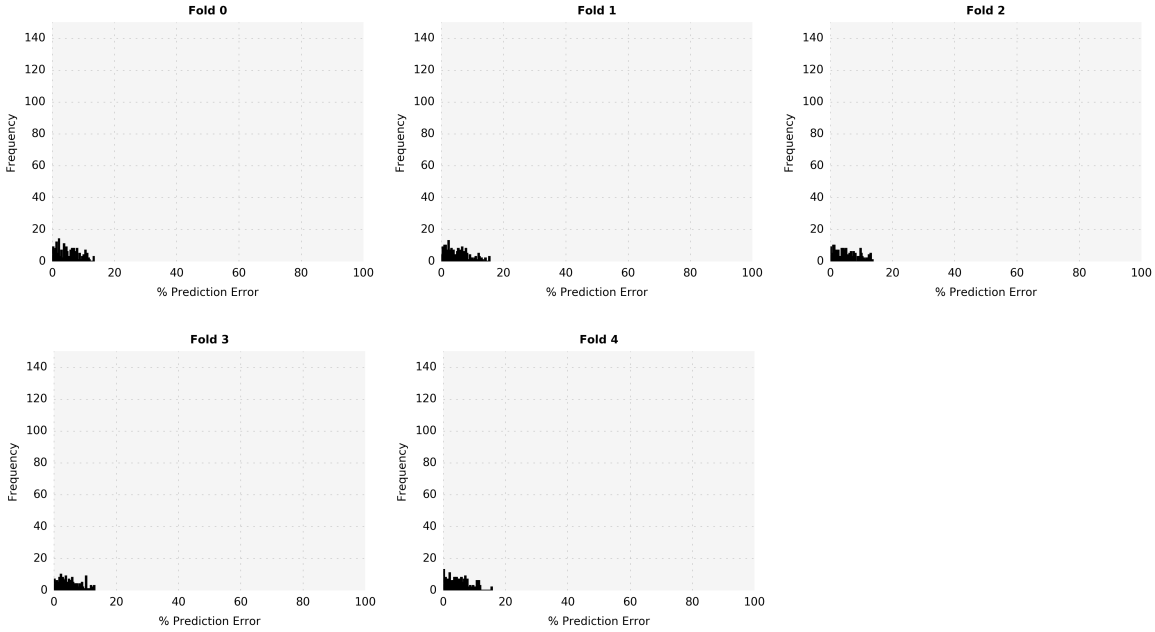


Figure A-13: Validation plots for five folds. TARGET: s35_max_resolved_force.

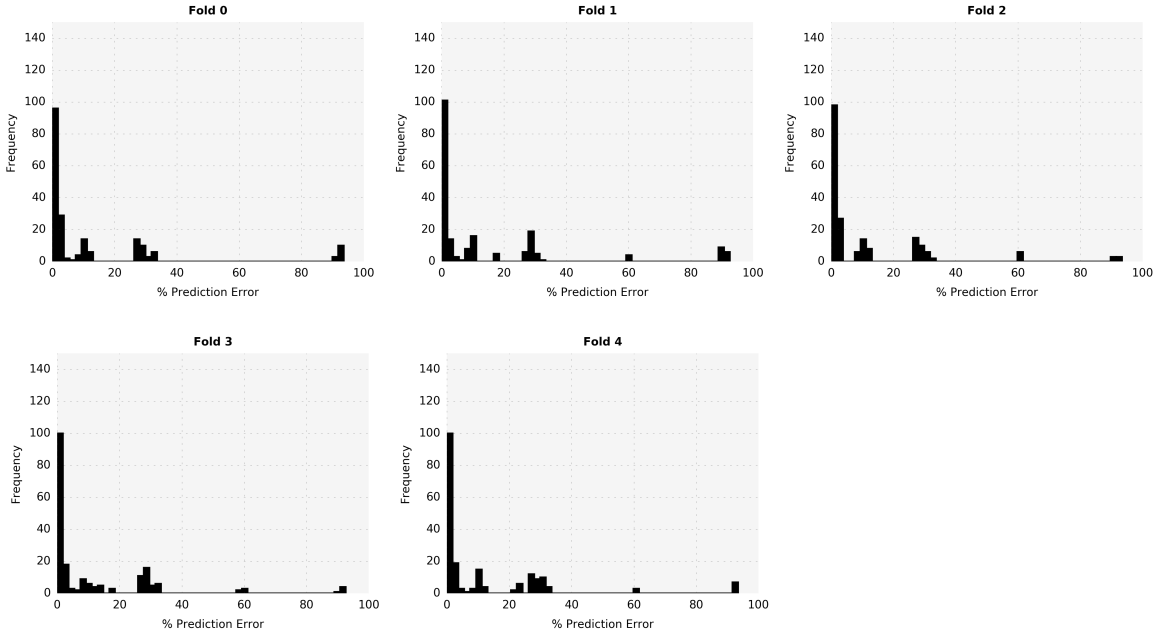


Figure A-14: Validation plots for five folds. TARGET: s45_max_resolved_force.

A.4 *d2* prediction validation of global BNM in Chapter 6, target: deflection

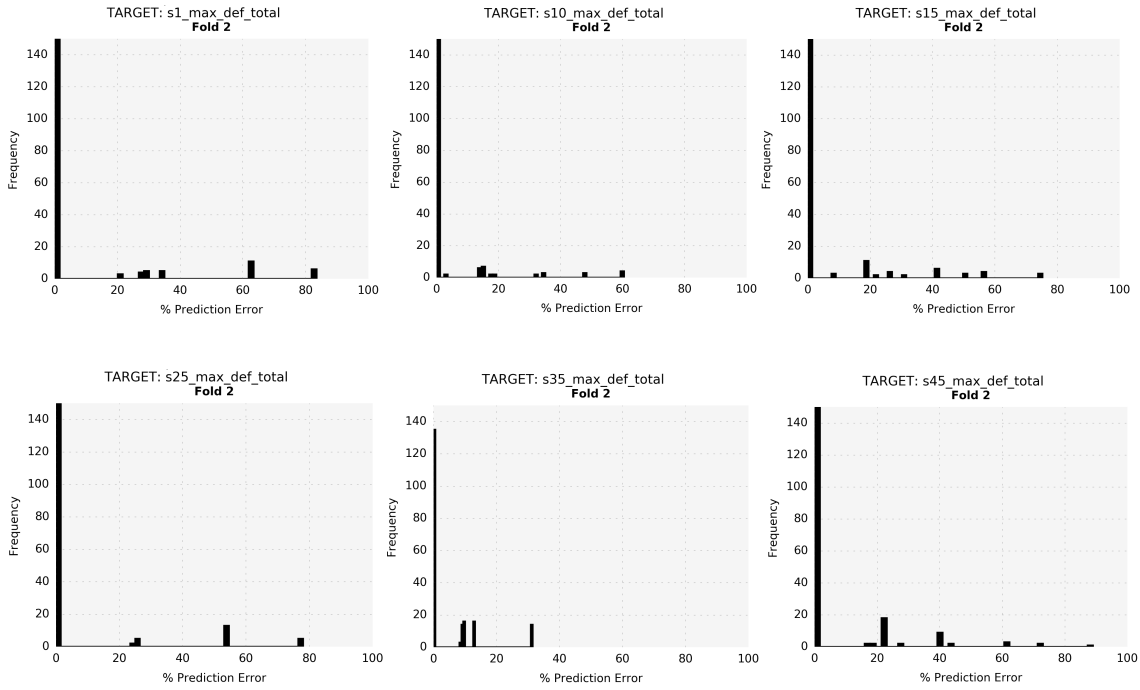


Figure A-15: ***d2*** validation plots for one fold, for each deflection target.

A.5 *d2* prediction validation of global BNM in Chapter 6, target: force

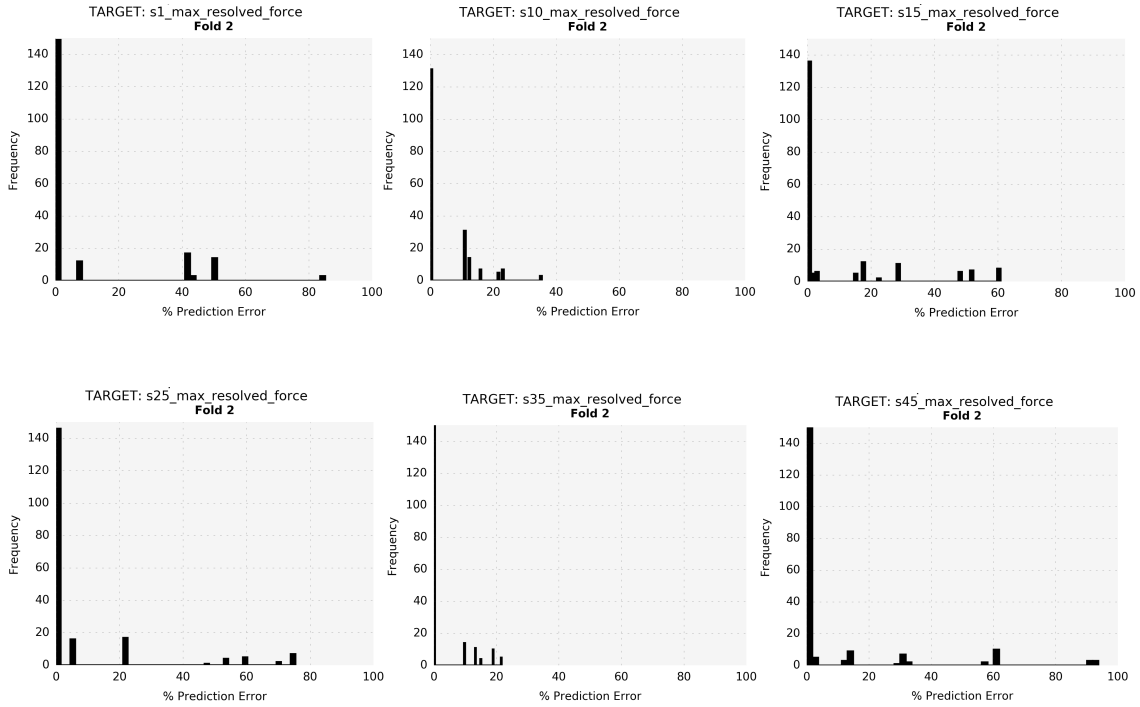


Figure A-16: ***d2*** validation plots for one fold, for each force target.