Solving the inverse bone remodelling problem using reinforcement learning

A feasibility study

G.B. Schlief

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control, BioMechanical Engineering

Solving the inverse bone remodelling problem using reinforcement learning

A feasibility study

Master of Science Thesis

For the degree's Master of Science in Systems and Control,
and Mechanical Engineering at Delft University of Technology

G.B. Schlief

January 8, 2026

Faculty of Mechanical Engineering (ME) · Delft University of Technology

Abstract

Bone dynamically adapts its strength to its environment in a process called bone remodelling. Mechanical stimuli are the primary predictors of future bone density, and many metrics exist to quantify their relationship with bone density changes. The inverse bone remodelling problem consists of finding the loading conditions that produce a bone density distribution, which is a non-unique, ill-posed problem. Understanding this process is relevant to implant design and osteoporosis prevention. Existing methods, like least-squares approaches, are computationally intensive as they iteratively explore the design space. This thesis explores a novel Reinforcement Learning (RL) framework as a possible alternative because of its effective exploration in inverse ill-posed problems.

Before focusing on the Machine Learning (ML) methods, we developed a dataset of steady-state density-load samples. Existing datasets did not meet the requirements of the RL framework for stable training. The forward model used for data generation was adapted to increase the input space, numerical stability and computational efficiency. The final model generated an extensive (100,000 samples), diverse (five loading types), high-quality dataset that the ML methods could rely on.

We used the dataset to train an Supervised Learning (SL) surrogate and an ensemble model capable of predicting the forward process (Structural Similarity Index Measure (SSIM) = 0.85 and SSIM = 0.87). The RL framework was designed by adapting the inverse problem into a sequential procedure. The surrogate model was used for reward estimation. This avoided learning instabilities created by the non-uniqueness of the problem. To validate the capabilities of the RL framework, we trained the model and a baseline on a simplified, order-reduced version of the problem. It outperformed the SL baseline model (SSIM 0.72 compared to 0.28), demonstrating improved learning stability.

The RL framework, consisting of a sequential decision-making process and a forward estimation surrogate, smooths the reward domain, thereby stabilising learning. Furthermore, the dataset analysis showed the inverse problem relies on identifiable samples, while the forward problem instead needs dataset diversity. Limitations of the approach are the dependency chain between the surrogate and the framework, and the inability to find more than one plausible solution. Future work could explore multi-agent strategies that propose multiple solutions or improve reward discontinuities created by the current metric.

Table of Contents

Preface

This document is a part of my Master of Science graduation thesis. As a double degree student, it was a considerable challenge to find a research direction that connects the fields of Systems and Control and Biomechanical Design, while also being compelling enough to dedicate a full year of work to. I am pleased to say that I have found my niche in this thesis.

During the previous year, the Faculty of Mechanical Engineering decided to stop participating in the Individual Double Degree (IDD) program of the university. As one of the last students in this program, I feel a responsibility to highlight the unique opportunities it provides, and I hope to have demonstrated the additional value and perspectives that arise when looking at a problem through a multidisciplinary lens.

Gijs Schlief, 2025

Acknowledgements

I want to thank my two supervisors, Dr. ir. M.A. Sharifi Kolarijani and Dr.ir. N. Tümer, who have guided me on this journey and provided feedback at critical stages of this thesis. ir. E.W.S. Tay has helped me understand DelftBlue, the supercomputer of Delft University. Furthermore, I would like to thank TU Delft for providing me with a space to study, access to their facilities, and an excellent education. Lastly, special thanks go to Dr.ir. Y.D. Bansod for making his bone remodelling forward model publicly available, without which this thesis would not have been possible.

Delft, University of Technology
January 8, 2026

G.B. Schlief

Chapter 1

Introduction

1-1 Context and Motivation

Despite its apparent rigidity, bone is a dynamic tissue that continually adapts its strength in response to its environment [1]. This process is called bone remodelling and is an important field of study. Mechanical loading is one of the primary predictors of future bone mass [2, 3]. Increased loading stimulates bone deposition, while reduced loading leads to bone loss. Predicting this behaviour correctly is crucial for increased understanding, treatment of conditions like osteoporosis, rehabilitation strategies, and the design of orthopaedic implants.

One commonly used method for predicting this mechanical stimulus-density relation is the Finite Element Method (FEM). The bone domain is discretised into small elements to estimate density adaptation under applied loads and the boundary conditions. This results in a model called the forward FEM bone adaptation model. When its algorithm and mesh are sufficiently refined, it provides an accurate representation of the remodelling process [4]. However, using FEM is computationally demanding, as the density update must be calculated for each element and at each time step of the model. Moreover, estimating realistic load values is challenging, as measuring in vivo is invasive and therefore impractical [5].

Estimating the loading conditions required for realistic forward simulations leads to the inverse bone remodelling problem, determining loading conditions that produce a given final density distribution, as shown in Figure 1-1. Solving the inverse problem enables researchers to refine current load estimations without requiring invasive procedures. However, because the inverse problem involves inferring cause from effect, FEM modelling methods cannot be directly applied. With recent advances in Machine Learning (ML), however, new approaches may be able to capture this complex mapping more efficiently. In particular, Reinforcement Learning (RL), and more specifically Proximal Policy Optimisation (PPO), offers a promising approach to solving the inverse problem by learning the underlying mapping between load and density. Similar inverse problems, like in the field of optics, have been solved using this novel technique [6, 7].

To summarise, this study examines the application of RL as a potential solution to the inverse bone remodelling problem. A framework is proposed that integrates a surrogate model to provide a computationally stable and continuous training environment for RL agents.

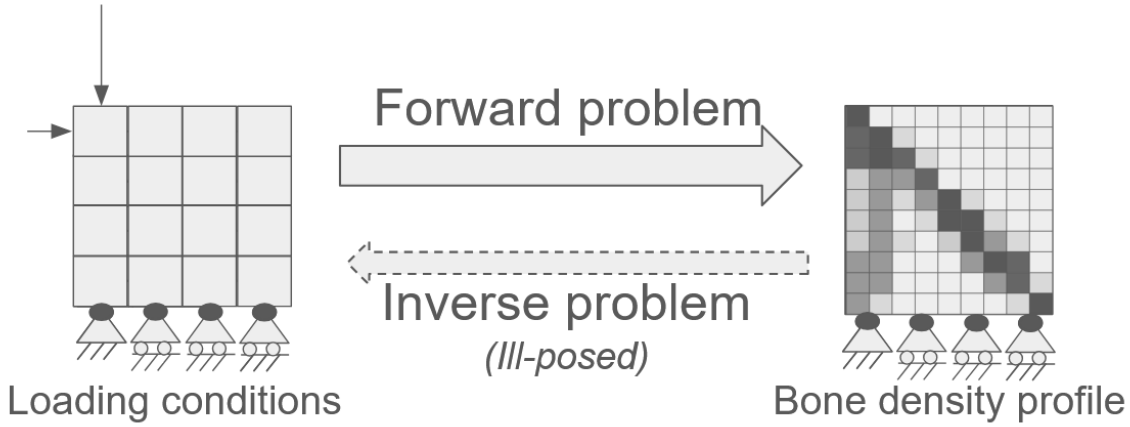


Figure 1-1: Conceptual schematic showing the forward and inverse bone remodelling problem.

1-2 Problem Statement and Rationale

The forward bone remodelling problem, which involves transitioning from load profiles to bone density, is complex, as the mapping between load and density is strongly nonlinear. However, with forward FEM adaptation models, such as those by Weinans et al., the process can be estimated with reasonable accuracy [8]. The inverse problem, mapping density to load profile, has received less attention, as its ill-posed nature and non-uniqueness increase modelling complexity [9]. Forward FEM models like Weinans cannot be reversed, as they depend on changes in density over time that are not directly observable.

Since direct inversion cannot solve the inverse problem, alternative methods are required to estimate the loading condition. A common approach involves testing many candidate load configurations and running the forward model until a matching density is found [5]. The effectiveness of this method, however, is limited due to computational bottlenecks and the combinatorial growth of the search space [10]. Traditional ML methods, such as Supervised Learning (SL), also struggle, as label-based learning relies on a single solution for each input. As the inverse bone remodelling problem is non-unique and admits multiple mechanobiologically consistent loading configurations for a single density distribution, SL models with a single target tend to converge to averaged or unstable solutions. Although stable alternatives exist, they are restricted to low-dimensional problem formulations or simplified loading scenarios [11]. Developing an alternative method that avoids these limitations would help bridge the gap in the field.

To address the non-uniqueness and computational limitations of existing approaches, this study proposes an alternative formulation of the inverse problem based on RL [12]. Rather than directly finding a single solution, an RL agent iteratively explores the design space

by proposing a loading condition, receiving feedback, and refining its proposal until a high-quality match is achieved. The loading condition proposed by the agent is evaluated using the forward model to calculate its corresponding density distribution, instead of being directly compared with the ground-truth load. This density estimate is compared to the original density to provide a reward signal that guides the agent while not penalising alternative mechanobiologically consistent loading configurations. A surrogate neural network further enhances this structure by approximating the forward model, eliminating the need to rerun the forward FEM model during each learning step. Combined, they create a framework that is well-suited to the ill-posed nature of the inverse problem.

1-3 Research Objectives and Questions

This thesis explores the use of an RL framework to solve the inverse bone remodelling problem by addressing the following main question:

Can RL, combined with a neural network surrogate, efficiently and accurately solve the inverse bone remodelling problem?

To answer this question systematically, the following subquestions have been formulated:

1. What defines the inverse bone remodelling problem, and why does it pose unique challenges for computational modelling and ML?
2. How does the forward FEM bone adaptation model work, and how can it be adapted to facilitate dataset generation?
3. How can a sufficiently large and diverse dataset be generated to enable effective learning of bone remodelling behaviour?
4. Can a neural network accurately approximate the forward FEM bone adaptation model?
5. How can RL be used to infer loading conditions from bone density distributions?
6. What are the computational limits and feasibility of the proposed approach?

The thesis is structured into 6 main chapters. Background knowledge regarding the bone remodelling problem and RL is provided in chapter 2, which is followed by explaining the forward model in chapter 3. Chapter 4 builds on this with the dataset generation process, while chapter 5 uses this dataset to train a forward estimation model. The report is finalised with the RL inverse model in chapter 6, and limitations and future improvements are discussed in chapter 7.

Together, these chapters present a novel RL framework for solving the inverse problem, offering an alternative to traditional optimisation-based methods. The following chapter provides the background on bone adaptation, modelling, and ML, creating the foundation for subsequent chapters.

Chapter 2

Background

This chapter establishes the theoretical background on the bone remodelling problem, its inverse counterpart and Machine Learning (ML) applied to bone remodelling problems. A literature study is used to analyse studies on the bone remodelling problem to verify the novelty of the designed Reinforcement Learning (RL) framework. This grounds the thesis in the current state of the art. Furthermore, due to the combined nature of biomechanical and ML topics in this thesis, the chapter also provides the necessary theoretical background knowledge needed to understand both fields.

The chapter begins with a biological overview of the bone remodelling process, followed by an explanation of the forward Finite Element Method (FEM) bone adaptation model used to estimate this remodelling. We then discuss the inverse problem and its additional challenges. Subsequently, highlighting the ML process, explaining Supervised Learning (SL) using a neural network and RL. The chapter ends with the proposed modelling framework and a summary.

2-1 Bone Remodelling and Mechanobiology

As mentioned in the introduction, bone is a constantly adapting organ. This section will ground later-explained bone remodelling models by explaining the biological theory behind bone remodelling. Starting with the cells responsible for remodelling, followed by the mechanobiological process that occurs when bone is under load. Lastly, we underscore its relevance by discussing some clinical applications of the theory.

2-1-1 Cellular Process of Remodelling

Inside the human body, four primary cells are responsible for the bone remodelling process. Together, they maintain bone health and strength by adapting to the environment's demands. The function of these cells is relevant as newer bone remodelling models directly use their concentrations to adapt the bone remodelling law [13].

Osteoclasts are responsible for removing old, unnecessary and damaged bone. These cells break down the calcium matrices and resorb the ions in the bloodstream in a process known as bone resorption. They break down the bone for renewal and to ensure calcium levels within the body stay healthy [2, 14].

The bone builders of the body are osteoblasts. They absorb calcium from the bloodstream and use it to build the bone matrix. Osteoblasts oppose osteoclasts, as they are responsible for creating new bone. A balance between these two cell types is crucial for bone health.

Once osteoblasts become embedded in the bone matrix, they transform into osteocytes. Osteocytes have a distinct function: they provide signals to surrounding cells about the current state of bone health and maintain the integrity of the bone matrix. Inside the bone, they can sense mechanical loading and provide signals to osteoblasts and osteoclasts to form new bone or resorb existing bone.

Finally, osteogenic cells are the stem cells of the bone. They can turn into different types of specialised bone cells and osteoblasts. They are responsible for maintaining a healthy concentration of osteoblasts inside the bone and dividing when necessary.

Figure 2-1 shows a schematic representation of these cell types and their location within the bone. The biological theory is critical because it represents the ground truth behaviour that forward models are trying to estimate. However, cellular processes occur at a microscopic scale that is difficult to use directly in forward bone models. Mechanobiology is the study of directly linking local mechanical loading to bone formation and resorption, bridging this gap in research.

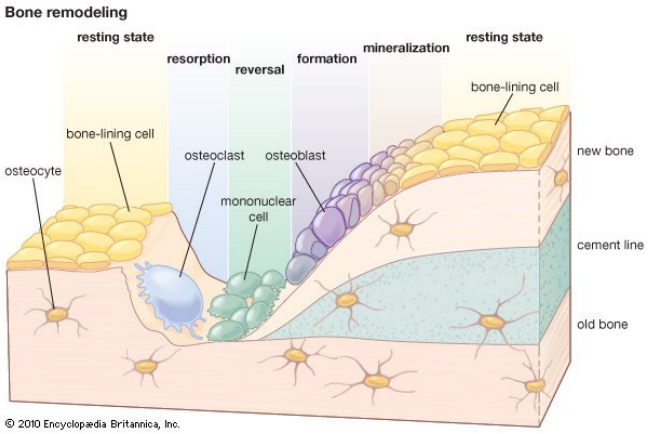


Figure 2-1: This figure shows the bone remodelling process at the cellular level. Source from: Encyclopædia Britannica [15].

2-1-2 Mechanobiology

While researchers did identify the cells responsible for the bone remodelling process in the late 19th century, the mechanisms that regulate their activity remained unclear. Wolff discovered that mechanical loading was the main mechanism responsible for bone resorption and deposition [2]. This mechanism is currently known as Wolff's law.

Frost expanded on this knowledge by creating the mechanostat model [3]. The model divides bone adaptation into three linear zones. In the middle is a zone where bone turnover is zero, and the density of the bone does not change. The low strain on the bone results in bone resorption, whereas overloading results in bone deposition. Combined, these result in a graph plotting bone turnover and strain, as shown in Figure 2-2. Recent studies have questioned the existence of the "lazy zone" proposed by Frost [16], though it remains widely used in bone remodelling models.

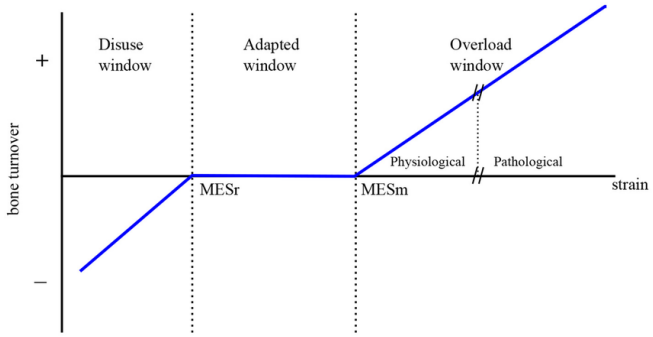


Figure 2-2: This figure illustrates the mechanostat model developed by Frost. The figure is taken from Martinez et al. [17].

Although the literature agrees on the significant role that mechanobiology plays in the remodelling process, the exact mechanical parameter that best represents the load sensed by osteocytes remains a matter of debate [18]. Fluid flow within the bone creates shear forces in the membrane of osteocytes, which in turn release signals regulating osteoblasts and osteoclasts [19]. Stress, strain, strain rate and the Strain Energy Density (SED) are currently the most common predictors. SED represents the energy stored inside the bone, and is sometimes favoured as a predictor because it combines both stress and strain [20]. Understanding these predictors forms the foundation for clinical and engineering applications, as discussed in the following section.

2-1-3 Clinical Relevance

Conditions that disrupt the remodelling process can have a significant impact on a patient's life. Osteoporosis is associated with considerable loss in physical function, social function, and well-being [21]. Furthermore, it affects a large part of the population as it is correlated with old age [22]. Improving bone strength reduces the risk of immobilising fractures, thereby enhancing the well-being and mobility of older people. Astronauts are also susceptible to high bone loss because loads on bone are reduced due to the lack of gravity [23].

A deeper understanding of the bone remodelling process has inspired novel therapeutic and engineering approaches, such as vibration training to stimulate osteocytes and decrease bone resorption [24]. Furthermore, it has improved orthopaedic implant designs, reducing load takeover of the implant to discourage bone resorption around the implant screw caused by stress shielding [25].

These new developments were guided mainly, not by direct biological theory, but by predicting bone behaviour using the forward FEM adaptation model. Such models enable the prediction of bone adaptation based on initial bone density and loading conditions over time.

2-2 Forward FEM bone adaptation models

FEM models are used for solving complex differential problems, making them particularly suitable for predicting bone adaptation under mechanical loads [26]. Forward FEM bone adaptation models are a combination of a FEM model and a forward tissue adaptation model. The FEM model provides the numerical foundation for solving stress and strain within the bone, describing how the material behaves. The forward tissue adaptation model then links the resulting mechanical stimuli to changes in density. The section concludes with a discussion of the limitations of these models, as later chapters utilise and adapt them for dataset generation and building a surrogate model.

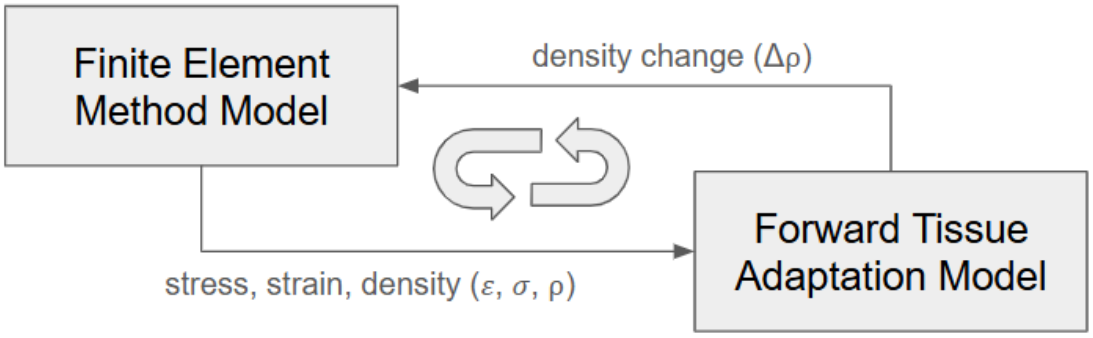


Figure 2-3: Schematic showing the interaction between the FEM model and the forward tissue adaptation model to estimate the bone remodelling process. The FEM model calculates stress, strain and density, and the forward tissue adaptation model uses this to calculate the change in density. This process is repeated many times until some convergence criteria are reached.

2-2-1 FEM Model

FEM is the process of simplifying a complex problem by dividing the geometry into many small elements and solving the problem at each element. The elements solve the problem at a local level by estimating the complex functions involved. Combined, they can solve complex differential equations without having to solve them mathematically. FEM is important as some complex differential functions are impossible to solve algebraically.

The FEM model reaches an equilibrium when the divergence of the stress tensor ($\nabla \cdot \sigma$) is the same as the body force vector (f). This formula is known as the equilibrium equation and shown in Equation 2-1.

$$\nabla \cdot \sigma + f = 0 \tag{2-1}$$

The bone remodelling process is a complex differential equation that elementary functions cannot express, especially for irregular geometry that is common in biological material. The FEM model provides a helpful tool to estimate bone remodelling dynamics even for complex material properties. However, it is the FEM model designer's job to decide how accurately

they want to estimate these complex properties, as this involves a trade-off between model fidelity and computational efficiency.

Forward FEM models rely on the fundamental estimation that the model can ignore microstructure, called the continuum assumption. Under this assumption, the microstructure can be modelled implicitly using average material properties, instead of simulating collagen, pores, and bone cells directly.

The second important property at the heart of these models is the constitutive relation. Denser bone is stiffer, while less dense bone is compliant. Although literature does not agree on the exact parameters that predict this relation, a typical structure is shown in Equation 2-2. This power relationship between Young's modulus and density is widely accepted, but a variety of constants are found based on test setup and bone specimen [27]. In general, the exponent n is between 1.27 and 2.57 for trabecular bone and between 4 and 7.4 for cortical bone [28]. This research suggests that cortical bone stiffness increases more rapidly with increasing density. The reference Young's modulus (E_0) represents the strength of a unit-density material.

$$E = E_0 \rho^n \tag{2-2}$$

To reduce computations, most forward FEM bone adaptation models rely on a few additional assumptions that further simplify the calculation. Firstly, Hooke's law is often used as a relation between stress (σ), stiffness (C) and strain (ε), as shown in Equation 2-3. Because bone is a nonlinear material, this relationship remains valid only under small deformations. However, under normal physiological loading, bone experiences relatively small strains, and Hooke's law is a valid approximation of the behaviour [29].

$$\sigma_{ij} = C_{ijkl} \varepsilon_{kl} \tag{2-3}$$

Furthermore, most forward FEM bone adaptation models assume bone is an isotropic material. Actual bone, especially cortical bone, is anisotropic, where osteocytes act as the mechanosensors that regulate the orientation of bone structure [30, 31].

2-2-2 Forward Tissue Adaptation Model

The forward tissue adaptation model translates the mechanical stress and strain found by the FEM model to a density stimulus. It determines the change in bone density of the body in response to stimuli using Equation 2-4. The function calculates the change in density over time ($\frac{d\rho}{dt}$) based on the current strain, stress and density ($f(\sigma, \varepsilon, \rho)$).

$$\frac{d\rho}{dt} = f(\sigma, \varepsilon, \rho) \tag{2-4}$$

Cowin et al. developed one of the first forward tissue adaptation models [4]. This model modified the elastic stiffness tensor directly based on the previous stiffness and stress on the bone, as shown in Equation 2-5. This model differs from more modern alternatives, which modify the density of the simulation, although they result in the same bone remodelling behaviour.

$$\frac{dC}{dt} = g(\sigma, C) \tag{2-5}$$

Huiskes et al. improved the model of Cowin et al. by estimating the stimulus using the SED instead of just the stress [25]. The SED represents the elastic energy stored per unit volume and can be used to update the density instead of the stiffness tensor. This change enhances computational feasibility, enabling the development of viable, complex models. Weinans et al. further adapted this model, based on Frost's mechanostat theory, by adding a minimal stimulus threshold, referred to as the 'lazy zone' [8]. This adaptation significantly improved the stability of running the FEM simulation. Moreover, the model added a minimum and maximum density, ensuring the model would never result in negative densities or densities outside the bone's feasible range.

Newer models further improved the Weinans model, including cell concentrations directly using differential equations, and created a variable remodelling rate that more accurately reflects reality [13, 32]. Other models focus more on improving morphology, such as those by Du et al., who use a multimaterial model to simulate trabecular bone [33]. Although newer models address certain modelling inaccuracies, they do have their limitations.

2-2-3 Limitations

One of the most significant limitations of the forward FEM bone adaptation models is the computational complexity. Iteratively updating thousands of elements across stress and density, given the nonlinear relation between results, creates long runtimes. Although computing power has grown exponentially over the years, models have kept pace due to increased mesh resolution and model realism.

Secondly, many models assume mechanics is the sole driver of bone adaptation, while other biological factors, such as hormones, vascularisation, and microdamage repair, are ignored. In reality, these factors do influence the remodelling process, and as they are also patient-specific, they reduce the ability of these models to accurately predict remodelling in patients.

As previously discussed, estimating in vivo loading conditions remains a challenge. As a result, the initial conditions of the bone adaptation models are estimates, not accurate force data. These estimated initial conditions limit the accuracy of bone adaptation models.

Furthermore, the forward FEM bone adaptation models cannot work in reverse, as it is impossible to estimate previous stimuli from the final bone density. This limitation makes it especially difficult to predict loading conditions from density distributions, a problem known as the inverse bone remodelling problem.

2-3 Inverse Bone Remodelling Problem

The forward bone remodelling problem updates density distributions based on a given loading condition. In practice, these loading conditions are challenging to measure and are only estimated. This limitation establishes the basis for the inverse problem, which involves determining the initial loading conditions from a given density distribution. Unlike the forward problem, the inverse problem is non-unique, as multiple loading configurations can result in the same density distribution [34]. As a consequence, the problem is ill-posed and thus more complex to solve.

This section reviews traditional least-squares optimisation and recent advances in ML as alternative approaches, highlighting their limitations and motivating the adoption of the new RL structure for solving the inverse bone remodelling problem.

2-3-1 Least-squares Inverse Optimisation Approaches

Early studies addressed a low-dimensional version of the inverse bone remodelling problem using least-squares optimisation. By minimising the difference between the target density (ρ_{obs}) and the new estimate ($\rho_{sim}(F)$), the optimal load configuration is found, as shown in Equation 2-6.

$$\min_{F} ||\rho_{sim}(F) - \rho_{obs}||^2 \tag{2-6}$$

An early application of this method is Fischer et al., who estimate the dominant loading direction of the femur from nine candidate directions [10]. Christen et al. have shown the effectiveness of the least-squares optimisation method by verifying the model with an in vivo dataset on the six basis load conditions (forces and moments in the x,y,z directions) [35].

Modern studies still use the least-squares optimisation method. Synek et al. demonstrated that habitual activities of mammals can be estimated by the optimisation framework by identifying a combination of six predefined loading directions [36]. Bachmann et al. demonstrated how changes in the material model lead to different mean and peak loads by comparing four increasing in complexity material models, including the newest fibre-based model that incorporates the anisotropic properties of bone [5].

As Christen et al. have shown, least-squares optimisation models can estimate the inverse bone remodelling relation for low-dimensional problems. However, these models rely on predefined force locations and angles to keep the search domain small. As a result, these approaches depend heavily on strong prior assumptions about load placement. In scenarios where such prior knowledge is unavailable or incorrect, the optimisation may converge to misleading solutions. Furthermore, the least-squares optimisations rely on iteratively running the computationally intensive forward simulations, which motivated the exploration of ML approaches as an alternative. Neural networks, as universal function approximators, can estimate the forward pass from the least-squares optimisation, slightly reducing physical fidelity but resulting in a much lower computation cost.

2-3-2 ML Approaches

ML methods enabled the creation of surrogate models, which estimate the FEM forward bone remodelling model. Ghosh et al. used this for designing an optimal implant [37], and Pais et al. recently demonstrated that surrogates significantly reduce computations while maintaining model fidelity [38].

Surrogate models accelerate the existing least-squares approach because the neural networks bypass the computationally intensive forward simulation by using an estimation from load to density. Campoli et al., for example, estimate the magnitude and angle of the loads based on density data [39]. Zadpoor et al. underlined the validity of this approach by developing two Artificial Neural Network (ANN) that predict force magnitude using four predefined

locations [11]. Lastly, Garijo et al. demonstrated that neural networks can also be applied at the microstructural level [40].

The addition of ML has improved the effectiveness of traditional inverse optimisation approaches. However, the examples listed above still limit the dimensionality of the design space. Furthermore, the ML methods discussed have limited interpretability. The model identifies the direct input-output mapping without providing insight into the underlying physical relationships. Lastly, least-squares optimisation minimises a fixed objective function, and supervised learning learns a single deterministic mapping. Neither framework explicitly encourages exploration of multiple valid solutions. RL uses a dynamic objective function, incorporating an exploration-exploitation tradeoff, which could improve convergence.

2-4 Reinforcement Learning as a Framework for Solving the Inverse Bone Remodelling Problem

RL provides a framework in which an agent learns by interacting with its environment. The agent moves towards an optimal policy by picking the action that maximises reward. The designer builds the reward and should align it with the problem goal. This iterative nature of solving a problem mirrors the way biological systems adapt. In the context of bone remodelling, the agent can learn loading conditions that minimise the difference between simulated and original density.

This section motivates the use of RL to solve the inverse bone remodelling problem. First, we discuss other ill-posed inverse design problems where RL has been effective. Second, we revisit the inverse bone remodelling problem and analyse if these same advantages apply and could result in an improved inverse optimisation.

2-4-1 Reinforcement Learning in Inverse Design

To the best of our knowledge, RL has not yet been directly applied to the inverse bone-remodelling problem. However, studies have demonstrated the effectiveness of RL in solving ill-posed inverse problems across various engineering applications. For example, Shah et al. used an RL structure to find a metamaterial with optimal acoustic properties by placing sound scattering cylinders [41]. This study demonstrated RL can handle spatially complex, nonlinear mechanical mappings in continuous inverse design tasks. Similarly, Gongora et al. designed an optimal composite 2D structure with a target Young's Modulus [42], highlighting RL 's ability to modify designs toward a mechanical target in a high-dimensional discrete design space. Finally, Shonkwiler et al. designed an optimal laminate structure where they either maximised longitudinal or transverse plate stiffness and compared the results to a genetic algorithm [43]. This research verified RL 's ability to solve this ill-posed inverse design problem, as well as outperform alternatives like genetic algorithms. Together, these studies demonstrate RL 's ability to explore non-unique, ill-posed design problems, characteristics that parallel the inverse bone remodelling problem.

2-4-2 Reinforcement Learning for Bone Adaptation

A key consideration in using RL is the ability to formulate the problem into an Markov Decision Process (MDP). In the case of the bone remodelling problem, this formulation flows naturally. The state represents the current density distribution, and the action consists of a proposed change to the current estimated loading configuration. The reward is then based on the similarity between the current density and the target density. This RL formulation has a clear advantage over previous methods, because intermediate density states provide additional information. The agent receives spatial information about which regions of the intermediate density are close to converging to the target density and which areas remain inaccurate. This results in an iterative design exploration that enables incremental refinement of the loading configuration, which is more easily learned than static or label-based approaches.

Second, the non-uniqueness of the inverse bone remodelling problem aligns naturally with the RL framework, because solution robustness (do small perturbations in the loading conditions result in large deviations in reward?) can be encouraged during model exploration through reward formulation.

Third, RL provides a significant degree of freedom to the designer in choosing a reward and environment. The designer can use this freedom to guide the agent towards an optimal policy more intelligently than a static mean-squared error function. For example, by comparing the original and simulated bone density not only in terms of magnitude but also in terms of morphology, additional information is provided to the agent.

In summary, RL provides a sequential, dynamic and flexible framework for solving the inverse bone remodelling problem. To ground the taken approach, this chapter ends with a more detailed summary of the proposed modelling framework.

2-5 Proposed Modelling Framework and Summary

The knowledge about bone remodelling problems, forward FEM bone adaptation models, surrogate models, and RL is integrated into a framework that solves the inverse bone remodelling problem. The framework consists of three steps, summarised below, and explained in depth in their corresponding chapter.

The first step involves generating a loading-density pair dataset with an adapted forward model. A load-generation model produces realistic force profiles, which we used as input to the adapted forward model. Running this model is a computationally intensive process, so parallelisation and other optimisations are used to minimise computation cost, while maintaining fidelity.

Secondly, a neural network surrogate model trains on the dataset to emulate the forward FEM bone adaptation dynamics. By approximating this complex mapping, the surrogate reduces the computational burden on the RL framework. We repeated this process multiple times and combined these models into a single larger ensemble model that can generalise and provide certainty estimates.

Lastly, the RL structure combines data and surrogate to solve the inverse problem iteratively, by proposing a force profile based on the current density distribution. The surrogate then

used this new force profile estimate to predict the density distribution. We then compared the original density and the surrogate density, and gave a reward based on their similarity. This process repeats until the agent's improvement stagnates.

Figure 2-4 shows an overview of how the three models come together. Both the surrogate and RL agent rely on the dataset's accuracy, which is why we took much care in generating it. Therefore, the following chapter focuses on the adaptations made to the baseline forward model to improve efficiency without compromising realism.

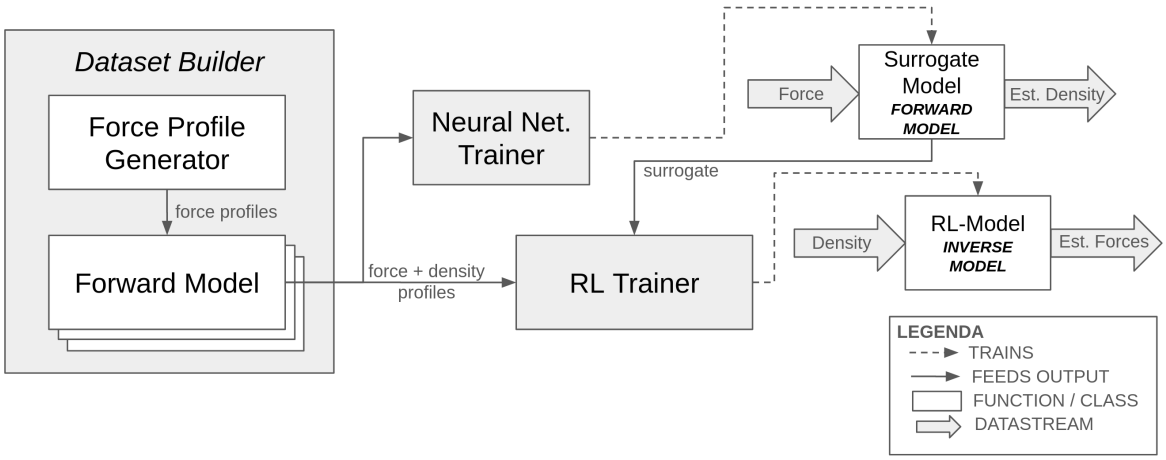


Figure 2-4: Overview of the modelling framework. The dataset builder generates load-density pairs using the forward FEM bone adaptation model. We used this dataset to train a surrogate neural network and combined both in the RL structure, which is capable of solving the inverse bone remodelling task.

Chapter 3

Forward Model

Machine Learning (ML) methods' effectiveness is limited by the quality of their training dataset. We created a dataset that is specifically suitable for Reinforcement Learning (RL) frameworks, as no publicly available datasets suitable for RL were identified. The forward Finite Element Method (FEM) bone remodelling model used for this task must be both fast and sufficiently accurate to preserve the dominant biomechanical load–density relationships.

This chapter adapts the forward model code from Bansod, refactoring it for data generation [44]. The chapter consists of three sections: a description of the baseline forward model, the adaptations made, and, finally, a validation using established literature.

3-1 Baseline Forward Model (Weinans / Bansod)

The choice of a forward FEM bone adaptation model is an important design decision for the final pipeline. The model by Weinans et al. is the most suitable, as it allows for validation after necessary modifications due to its extensive documentation [8]. Furthermore, it is computationally efficient due to its more conceptual focus compared to newer models.

Bansod et al. implemented a modern Python version of the Weinans model to verify the effect of electronic impulses on remodelling and draw conclusions about the relationship between exercise and the density of the human tibia [44]. We used their code as a baseline model due to its modern Python implementation, which is the current standard for machine learning (ML). Using their model resulted in a single language pipeline, which eased implementation downstream.

The baseline forward model provides the primary building block that is expanded and adapted for the ML implementation. The focus of these adaptations and extensions is not perfect physiological realism, but to create a model that generates data suitable for training supervised models and RL agents. To extend the model without compromising biomechanical fidelity, we must understand all components of the baseline, which will be the focus of this section.

The baseline model consists of multiple subfunctions, each responsible for its own part of the computation. A block diagram illustrates each subfunction and its interaction with other subfunctions to form the entire model (Figure 3-1).

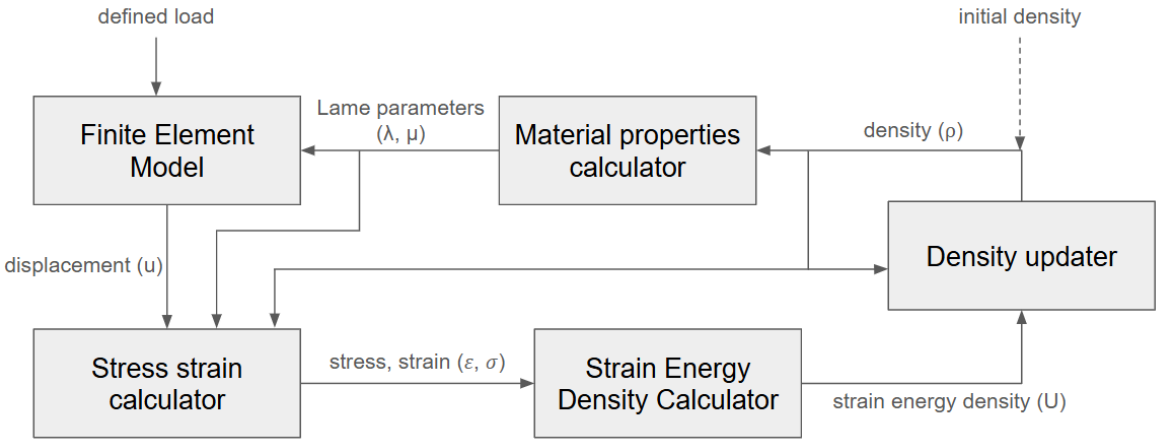


Figure 3-1: Block diagram of the forward modelling process. Each block represents a different part of the code responsible for its own calculations. The arrows represent the flow of variables, pointing from the block that calculates them towards the block where they are needed.

3-1-1 Material Properties Calculator

The first step in the forward FEM bone adaptation model is to calculate the material properties based on the initial density. The material properties are the shear modulus μ and the first Lamé parameter λ, and they define the elastic properties of the material. They are calculated based on Young's modulus E and the Poisson ratio ν using Equation 3-1 and Equation 3-2.

$$\mu = \frac{E}{2 \cdot (1 + \nu)} \tag{3-1}$$

$$\lambda = \frac{E \cdot \nu}{(1 + \nu)(1 - 2\nu)} \tag{3-2}$$

The Poisson ratio is assumed to be a constant of 0.3, which depends weakly on density but is held constant for simplification purposes. The Young's modulus (E) for each element is calculated based on the density and two simulation constants using a power law estimation (Equation 3-3).

$$E = E_0 \cdot \rho^n \tag{3-3}$$

The scaling factor, $E_0 = 3790$ MPa$\cdot(\mathrm{g/cm^3})^{-2}$, and the stiffness ratio, $n = 3$, link the density of bone to its Young's modulus, the same constants as Weinans et al. [8].

The material properties assume a monotonic relationship between density and stiffness, as well as isotropy and linear elasticity, as discussed in chapter 2. The FEM then utilises the material properties to solve the displacement field.

3-1-2 FEM Model

The previously calculated material properties and the element densities are combined in the FEM model to calculate the displacement field.

To discretise the bone domain, a square mesh was created in FEniCS, an open-source Python package used for the FEM. The resolution is 40×40 quadrilateral cells as a compromise between computation time and discrete error. The FEM represents the displacement using first-order Lagrange elements, while the density and material parameters are defined with discontinuous Galerkin.

The FEM assumptions discussed in the background section all apply to this model, and the simulation utilises homogeneous elements in a quasi-static setting. Furthermore, the model uses the standard convergence criteria of FEniCS.

The mesh and material properties together define the weak stiffness form used to compute the displacement field. The weak stiffness form consists of two components: distortional (a_{shear}) and dilatational (a_{vol}), shown in Equation 3-4. The stiffness form depends on the trial displacement field (u), the test displacement field (v) and the previously calculated material parameters. The trial displacement field represents the unknown displacement that the FEM solver computes, while the test function is the admissible virtual displacement which enforces equilibrium. Both are defined on the same mesh and in the same finite element space (V_h) on the domain Ω.

$$a(u, v) = a_{shear}(u, v) + a_{vol}(u, v) \tag{3-4}$$

The shear contribution defines the material's resistance to changes in shape. The contribution integrates the shear parameter (μ) with the inner product of the strain on the trial and test displacement over the domain $\Omega \subset \mathbb{R}^2$, yielding Equation 3-5. The strain tensors ($\epsilon(u), \epsilon(v)$) of the trial and test displacement field are calculated using Equation 3-6.

$$a_{shear}(u, v) = \int_{\Omega} 2\mu \langle \varepsilon(u), \varepsilon(v) \rangle d\Omega \tag{3-5}$$

$$\varepsilon(u) = \tfrac{1}{2}(\nabla u + (\nabla u)^T), \qquad \varepsilon(v) = \tfrac{1}{2}(\nabla v + (\nabla v)^T) \tag{3-6}$$

The volumetric (Lamé) contribution represents the virtual work caused by dilatational deformation. The Lamé parameters are integrated with the divergence of the displacement on the same domain in Equation 3-7. The divergence of the displacement ($\nabla \cdot u$) calculates whether the material is expanding (positive) or contracting (negative), which contributes to the material's resistance to volumetric change.

$$a_{vol}(u, v) = \int_{\Omega} \lambda (\nabla \cdot u)(\nabla \cdot v) d\Omega, \tag{3-7}$$

The bottom nodes are constrained using Dirichlet boundary conditions. They are defined as the set $\Gamma_B \subset \partial\Omega$ with $y = 0$, where $\partial\Omega$ is the boundary of the set Ω. The bottom left node in the set $\Gamma_{B1} \subset \Gamma_B$ with $x = 0$ was fully constrained, disallowing both vertical and horizontal displacement. The other bottom nodes ($\Gamma_{B2} \subset \Gamma_B$ and $\Gamma_{B1} \cap \Gamma_{B2} = \emptyset$) are only constrained in the vertical direction, referred to as roller constraints. The constraints limit the displacement of the nodes directly, which is known as the strong form. The test space that satisfies the Dirichlet boundary conditions is $V_h^0 \subset V_h$.

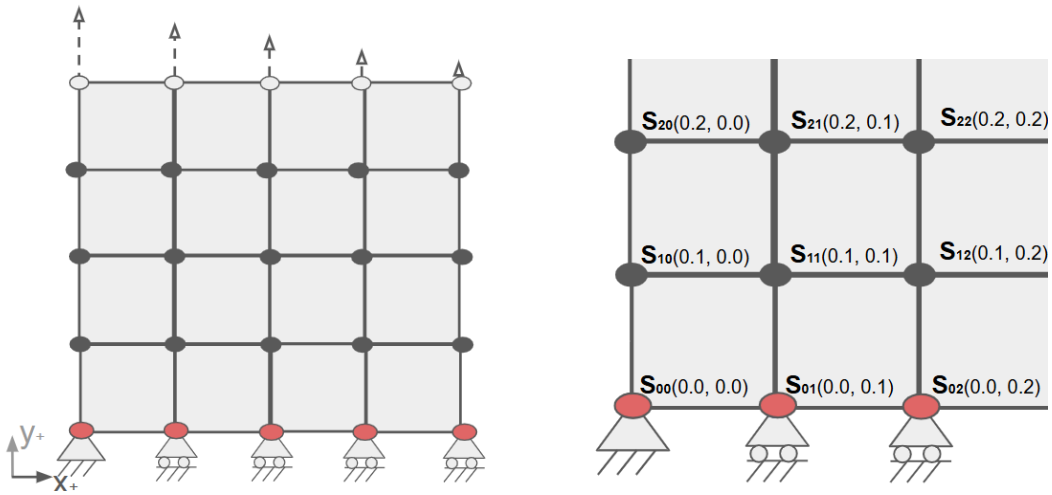
Vertical forces (F_y) were applied along all top nodes ($\Gamma_T \subset \partial\Omega$) using a triangular distribution defined in Equation 3-8. The maximum force of the triangular load is 25 Newton based on Weinans et al. [8]. Here x denotes the x-coordinate along the boundary Γ_T and x_{max} the rightmost x-coordinate. This loading condition results in the weak load form shown in Equation 3-9 with the vertical component of the test function (v_y).

$$F_y(x) = F_{max}(1 - \frac{x}{x_{max}}), \qquad x \in \Gamma_T \tag{3-8}$$

$$L(v) = \int_{\Gamma_T} F_y v_y \, d\Gamma \tag{3-9}$$

The weak stiffness form, the load form and the boundary constraints together define a complete finite element model as shown in Figure 3-2. The FEM solver computes the weak form (Equation 3-10) at each timestep, by setting the weak stiffness form ($a(u_h, v_h)$) equal to the weak load form ($L(v_h)$). Later components then use the resulting final displacement field (u_h) to calculate strains and stresses.

$$a(u_h, v_h) = L(v_h) \qquad \forall v_h \in V_h^0, \tag{3-10}$$



(a) FEM model **(b)** Node naming

Figure 3-2: Baseline forward model with the triangular force profile, square mesh, fixed constraint, and roller constraints.

3-1-3 Forward Tissue Adaptation Model

The forward tissue adaptation model consists of three subcomponents: the stress-strain calculator, the Strain Energy Density (SED) calculator and the density updater. With the displacement field u calculated by the FEM model, Equation 3-11 calculates the strain tensor ε. Hooke's law, shown in Equation 3-12, then calculates the stress tensor σ with the strain tensor, material properties, and displacement field under the assumption that the material is isotropic and linear elastic.

$$\varepsilon = \tfrac{1}{2}(\nabla u + (\nabla u)^T) \tag{3-11}$$

$$\sigma = \lambda\,(\nabla \cdot u)\,I + 2\mu\,\varepsilon \tag{3-12}$$

As discussed in chapter 2, models can use different stimuli to estimate the remodelling process. The baseline uses SED because it combines both stress and strain, as shown in Equation 3-13, and is passed to the last component, the density updater.

$$U = \tfrac{1}{2}\,\langle \sigma, \varepsilon \rangle = \frac{1}{2}\sum_{i,j} \sigma_{ij}\varepsilon_{ij} \tag{3-13}$$

The differential equation shown in Equation 2-4 forms the basis for the last component, the density updater. Formulating the remodelling process using a growth stimulus results in Equation 3-14, which the baseline uses to update the density.

$$\frac{d\rho}{dt} = B(S-k), \quad \rho_{min} < \rho \leq \rho_{max} \tag{3-14}$$

Here, B is a remodelling speed constant, and k is a threshold value representing the minimal stimulus required for bone growth. The mechanical stimulus on the bone (S) is the strain energy density (U) divided by the current density (ρ) as proposed by [25]. Substituting this definition into the remodelling equation yields Equation 3-15.

$$\frac{d\rho}{dt} = B(\frac{U}{\rho}-k), \quad \rho_{min} < \rho \leq \rho_{max} \tag{3-15}$$

Therefore, to update the density at each time step, the model requires the density field (ρ) and the strain energy density (U) from the previous time step. The constants used in the simulation are sourced from the literature, with their exact values and corresponding sources listed in Appendix A. The model employs discrete time using an explicit forward Euler scheme with a normalised time step of one day ($\Delta t = 1$), resulting in Equation 3-16. Note that this equation is only valid for sufficiently small Δt and a linear, threshold-based remodelling relation.

$$\rho^{t+\Delta t} = \rho^t + \Delta t \cdot B(\frac{U}{\rho^t}-k) \tag{3-16}$$

The three elements combined are the forward tissue adaptation model by Weinans. Together with the FEM model, they provide the baseline forward model by Weinans et al. The following section builds on this foundation by providing detailed information about the modifications made to this basic model.

3-2 Adaptations for Robust and Scalable Simulation

The baseline model was adapted to increase robustness and scalability. Noise in datasets is known to significantly impact the predictive accuracy of ML models [45]. The original model's behaviour has some numerical artefacts that hinder robustness, which this section aims to solve. The model furthermore does not reliably generalise across varied input data. This section describes the adaptations made to the remodelling law, the FEM model, and the model output representation.

3-2-1 Remodelling Law Stabilisation and Convergence Control

Running the original forward tissue adaptation model results in dynamically oscillatory steady-state solutions. Small changes in the input have large impacts on the final result due to the unbounded remodelling rate and premature convergence. The steady state solution does not accurately reflect actual bone density, and the ML models are unable to learn the dynamics. In this case steady state is the converged solution where densities do not change above a certain threshold. This section adapts the original remodelling law to improve the steady state solution of the FEM model. The goal is to improve robustness without changing the underlying model dynamics grounded in literature.

First of all, under small loading conditions, cyclical behaviour can occur. The density does not converge towards a single value but has a limit-cycle oscillation. This is a problem because the steady state solution used to train the ML models either receives the high or low value at random. The final result is made deterministic using a clipping function placed on the remodelling function. This results in the update rule shown in Equation 3-17. Other bone remodelling models, like Su et al., have used this cut-off before to prevent oscillations [46]. The cut-off is also biomechanically justifiable as Fahy et al. estimate bone turnover at 7.7% per year in healthy adults [47]. The cut-off is placed at 0.02 as a compromise between convergence speed and removing cyclical behaviour. Lower cut-offs increase the time needed to reach steady state, and higher cut-offs do not remove the cyclicality.

$$\frac{d\rho}{dt} = \text{clip}(B(\frac{U}{\rho} - k),\ -\Delta_{sat},\ \Delta_{sat}), \quad \Delta_{sat} = 0.02 \tag{3-17}$$

To circumvent the premature convergence and reduce sensitivity to the initial conditions of the forward model, two exponential functions dynamically refine convergence criteria. The first exponential equation was used for a dynamic density tolerance ρ_{tol}, as shown in Equation 3-18. The density tolerance converges elements based on the change in density between the current and previous timestep.

$$\rho_{\text{tol}}(t) = \rho_{\text{tol},0}\,\kappa^{\,t} \tag{3-18}$$

To further increase stability, we created a convergence counter. The counter increases when an element falls within the density tolerance and resets to zero when it does not. This way, elements only converge when multiple consecutive steps are below the threshold. Once the convergence passes the threshold coupled to Equation 3-19, the element is considered converged.

$$C_{i,j}(t+1) = \begin{cases} C_{i,j}(t) + 1, & \text{if } \Delta\rho_{i,j}(t) \leq \rho_{\text{tol}}(t), \\ 0, & \text{otherwise.} \end{cases} \tag{3-19}$$

$$\Delta\rho_{i,j}(t) = \left|\rho_{i,j}(t) - \rho_{i,j}(t-1)\right|$$

The required number of consecutive steps C_{tol} also decreases exponentially from ten to one over the course of the simulation, as shown in Equation 3-20. An element converges if $C_{i,j}(t) \geq C_{\text{tol}}(t)$ to ensure that all elements have converged by the end of the simulation.

$$C_{\text{tol}}(t) = C_{\text{tol},0}\,\gamma^{\,t} \tag{3-20}$$

To find the function constants that minimise the error of the steady state solution and the simulation runtime, a parameter sweep is performed. The number of time steps, convergence tolerance decay and the convergence steps decay are varied between contender values from Table 3-1. The 360 configurations were compared to a baseline run for 1000 steps without convergence criteria. The comparison had 100 force samples with batch number 12345. The runtime and relative L2 error of these runs are shown in Figure 3-3.

Table 3-1: Parameter sweep ranges used for robustness analysis. A total of $5 \times 3 \times 4 \times 6 = 360$ configurations were evaluated.

Parameter	Symbol	Values
Density tolerance decay	κ	$\{1.00,\ 1.02,\ 1.04,\ 1.06,\ 1.08\}$
Convergence after steps	$C_{\text{tol},0}$	$\{1,\ 5,\ 10\}$
Convergence steps decay	γ	$\{1.00,\ 0.98,\ 0.96,\ 0.94\}$
Time steps	T	$\{50,\ 100,\ 150,\ 200,\ 250,\ 300\}$

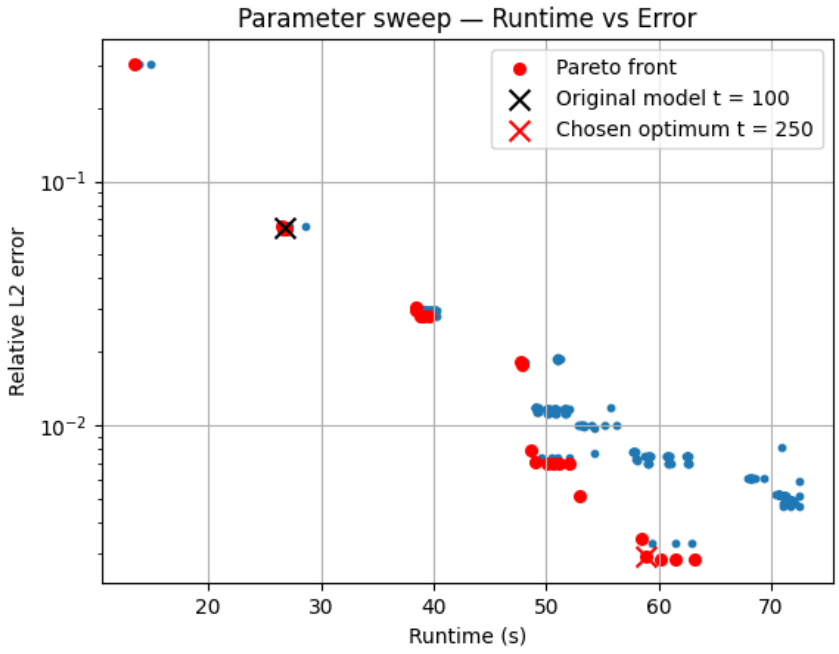


Figure 3-3: Parameter sweep for many different parameters demonstrating the trade-off necessary between steady-state accuracy and runtime. The red points represent the Pareto front of the sweep.

The results of the Pareto sweep are shown in Table 3-2 and demonstrate that exponential decay in the consecutive step convergence does not improve the simulation for longer step simulations. The steps used in the sweep have a strong correlation to both runtime and relative L2 error until 250 steps. The final optimum was chosen based on a strong emphasis on minimising L2 error to reduce noise in the dataset. From the Pareto front models with a similar error, the one with the smallest runtime is chosen as shown in Figure 3-3.

Table 3-2: Pareto-optimal hyperparameter combinations for the convergence controller, showing the trade-off between runtime and steady-state accuracy.

Steps	κ	$C_{\mathbf{tol},0}$	γ	**Runtime (s)**	**L2 error**
100 (Original)	1.00	1	1.00	25.64	0.291
50	1.02	1	0.94	13.50	0.302
100	1.06	1	0.98	26.58	0.0651
150	1.08	5	1.00	38.79	0.0281
200	1.08	5	1.00	48.65	0.00787
250	1.06	5	1.00	58.41	0.00341
250 (Chosen)	**1.06**	**10**	**1.00**	**58.87**	**0.00291**

Importantly, the rate saturation and dynamic convergence do not modify the underlying stimulus-response relationship. The adapted model yields deterministic steady-state solutions that substantially reduce noise. With premature convergence error minimised, the focus is shifted to removing discretisation errors.

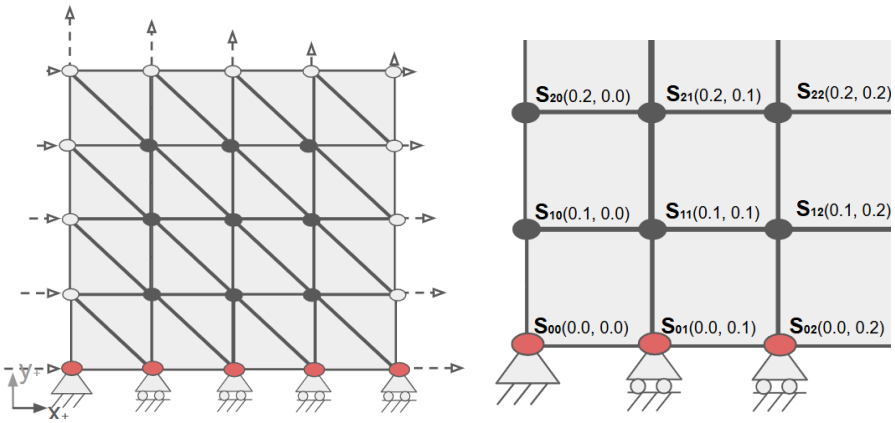
3-2-2 Numerical Discretisation

The previous section modifies the forward model to improve robustness and reduce errors. This section builds on this by minimising the discretisation error created by the FEM model.

As FEM models discretise the bone remodelling domain, discretisation errors become an additional concern. As stated previously, minimising errors increases the chance of successfully training an ML model. First, discussing the change in input from the single rigid force profile to an adaptable input where minimal discrete error is a prime concern. Second, the order of the FEM model is increased to improve bone remodelling fidelity further, and lastly, the output handling is discussed.

The baseline forward model used a prescribed ramp profile along the top row of elements. The adapted version should be able to plot an extensive range of different force profiles. To accommodate this, the new model employs a force profile vector $\mathbf{F}_{seg} \in \mathbb{R}^{I,\{top,\ right,\ left\}}$ where I represents the resolution of the force vector. The vector has a second dimension which represents the top, right, and left columns respectively, as can be seen in Figure 3-4. Loads cannot be placed internally, both to keep the problem space small and increase biomechanical fidelity. This simplification is biomechanically justifiable, as joints and tendons comprise the loads of real bones, which connect to the bone on the boundary. Furthermore, loads cannot be placed along the bottom edge nodes, as the constraints would immediately absorb any forces applied to these nodes.

The FEM model can only work with forces applied directly to node locations. We need a mapping from the discrete force profile to the node forces $\mathbf{f}_{node} \in \mathbb{R}^J$ that works for any com-



(a) Adapted profile **(b)** Node naming

Figure 3-4: Adaptations to the baseline model included the addition of different types of profiles along the top, right, and left edges, perpendicular to their edge.

bination of I and J. The force profile is first split into the three independent side directions, resulting in a force vector on a single boundary $\mathbf{f}_{seg} \in \mathbb{R}^I$. Second, the force piecewise profile values are projected to the node locations. This keeps the discretisation error low and conserves the total applied force. The mathematical representation is shown in Equation 3-21, where the mapping matrix $\mathbf{W} \in \mathbb{R}^{J \times I}$ is defined in Equation 3-22.

$$\mathbf{f}_{\text{node}} = \mathbf{W}\,\mathbf{f}_{\text{seg}} \tag{3-21}$$

The segment-wise total forces are defined on $[s_{i-1}, s_i]$ for $i = 1, \ldots, I$, and x_j is the coordinate of node j on the same boundary. Furthermore, the end points of the boundary are s_0 and s_I and the indices outside $1, \ldots, I$ are omitted. The FEM boundary shape function $N_j(s)$ describes the contribution associated with node j. A schematic representation of this process is shown in Figure 3-5.

$$W_{j,i} = \frac{1}{s_i - s_{i-1}} \int_{s_{i-1}}^{s_i} N_j(s)\,\mathrm{d}s, \tag{3-22}$$

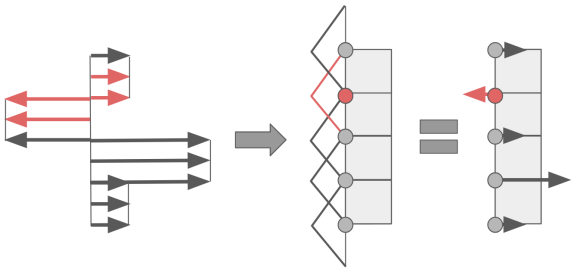
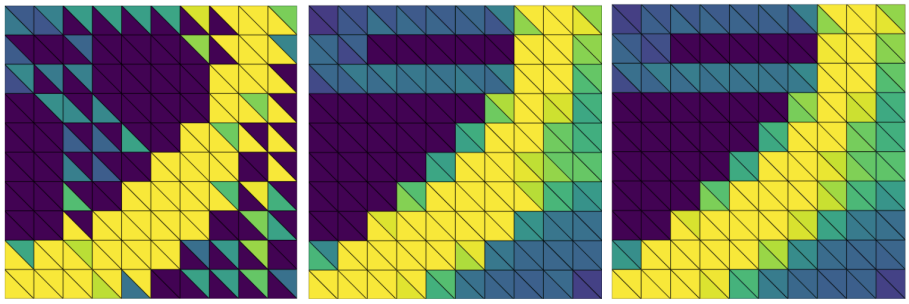


Figure 3-5: Discrete reconstruction of a piecewise-linear boundary traction field from segment-wise forces and its interpolation to FEM boundary nodes. The red forces show which elements in the initial traction field influence one of the final nodes.

To further improve the stability and spatial resolution, the mesh order was increased from first to third-order elements. This increase reduced the numerical error of the mesh by eliminating

inconsistent load transfer, which resulted in skipped elements. These vacancies were a problem in the baseline as they created an inconsistent mesh that was difficult to learn from. Second-order elements significantly reduced this problem, as can be shown in Figure 3-6. Third-order elements further reduced the error, although diminishing returns were observed. However, the computational time between the second and third order was so similar that we chose the latter for the final model (Table 3-3).



(a) First order **(b)** Second order **(c)** Third order

Figure 3-6: The density profile improved by moving from first order to third order elements, because element vacancies disappeared.

Table 3-3: Computation time and element count benchmark (100 samples on a 10×10 grid).

Order	Nodes per element	Computation time
First order	3	117 s
Second order	6	152 s (+30 %)
Third order	10	165 s (+41 %)

The FEM model uses a triangular mesh to represent the data. However, further pipeline elements rely on a square density field representation. To remap the triangular density representation to square one each triangular element $e \in \mathcal{E}$ is mapped to a square grid cell, $\rho_{i,j} \in \mathbb{R}^{n_{rows} \times n_{columns}}$ as shown in Equation 3-23. After this, Equation 3-24 calculates the new values of the square cells $\rho_{i,j}$ by averaging all elements inside a cell, where $m^{-1}(i,j)$ is the set of elements in $\mathcal{E}$ that are mapped to cell (i,j) and $|m^{-1}(i,j)|$ is the size of this set.

$$m : \mathcal{E} \to \{(i,j) | 1 \leq i \leq n_{rows}, 1 \leq j \leq n_{columns}\} \tag{3-23}$$

$$\rho_{i,j} = \frac{1}{|m^{-1}(i,j)|} \sum_{e \in m^{-1}(i,j)} \rho_e \tag{3-24}$$

Although this mapping can reshape any triangular mesh into any size square matrix, our model specifically combines two triangular elements into a single square unit. This alignment of square and triangular mesh results in minimal additional discretisation error relative to the FEM solution, the primary goal of this section.

The baseline model stores its square mesh in the .vtu format. However, as the Supervised Learning (SL) and RL models are unable to work with this file format, they need to be converted to a more suitable format, namely the NumPy array. The NumPy array has wide

package integration, including Torch and Stable Baselines 3, which the pipeline uses later. A JSON file stores the density profile and loads with a serial number using the format shown in Appendix B. In the case of a 10 by 10 matrix with 64-bit elements, each datapoint uses a total of 1.04 kB of storage.

The model is now able to map any force vector input into the FEM model, run a high-order simulation and represent the output into a matrix of size I, J. The last adaptations made to the model focus on computational efficiency and tolerance error minimisation.

3-2-3 Computational Efficiency and Tolerance Errors

The last set of improvements made to the baseline model focused on enhancing computational efficiency and error tolerance minimisation. The computational performance of the forward model directly determines the feasible scale of all downstream elements. As ML typically requires large datasets, the final model is estimated to use approximately 100,000 samples, which means any performance increase will save a substantial amount of time in data generation. To achieve this performance increase, we implemented two key changes: improved initialisation and cross-language communication.

First, the code was refactored from a single function into multiple classes, adhering to the Single Responsibility Principle [48]. This change decreased overhead within the simulation loop, as we moved the modules used for mesh generation and solver initialisation outside of the main simulation loop.

By running cProfile, a profiler that detects the time each section of code takes to run, a significant performance bottleneck was identified. FEniCS uses C++ for executing core FEM operations, while the data handling and control occur in Python. This structure resulted in many cross-language calls that introduced latency. By saving the mesh in a different location in memory, both Python code and C++ code could read and write to the mesh without having to reinitialise it.

To reduce the discrete error, a tolerance hierarchy chain was defined. Three different tolerances are all responsible for part of the model. Density tolerance ($\epsilon_\rho = 10^{-9}$) defines when a cell's density in the mesh has converged, the Krylov solver tolerance ($\epsilon_\kappa = 10^{-10}$) defines when the weak form of the FEM model converges at each timestep, and the boundary tolerance ($\epsilon_B = 10^{-14}$) is used to specify the error allowed in the forces and constraints to ensure conservation of energy. The tolerances are all higher than the machine precision ($\epsilon_{machine}$ $\approx 5 \times 10^{-}16$), and outer model convergence tolerances are intentionally looser than inner solver tolerances. This ensures large components converge due to modelling dynamics, not precision errors.

$$\epsilon_\rho \gg \epsilon_\kappa \gg \epsilon_B \gg \epsilon_{machine} \tag{3-25}$$

In summary, the original model by Weinans et al. was adapted to increase fidelity [8]. Adapting the remodelling law to create a robust model that did not prematurely converge, while minimising discretisation errors using higher order elements. Furthermore, creating a scalable model by adding input-output mappings and improving computational efficiency. To verify if these improvements had the desired effect without changing remodelling dynamics, the forward model was validated.

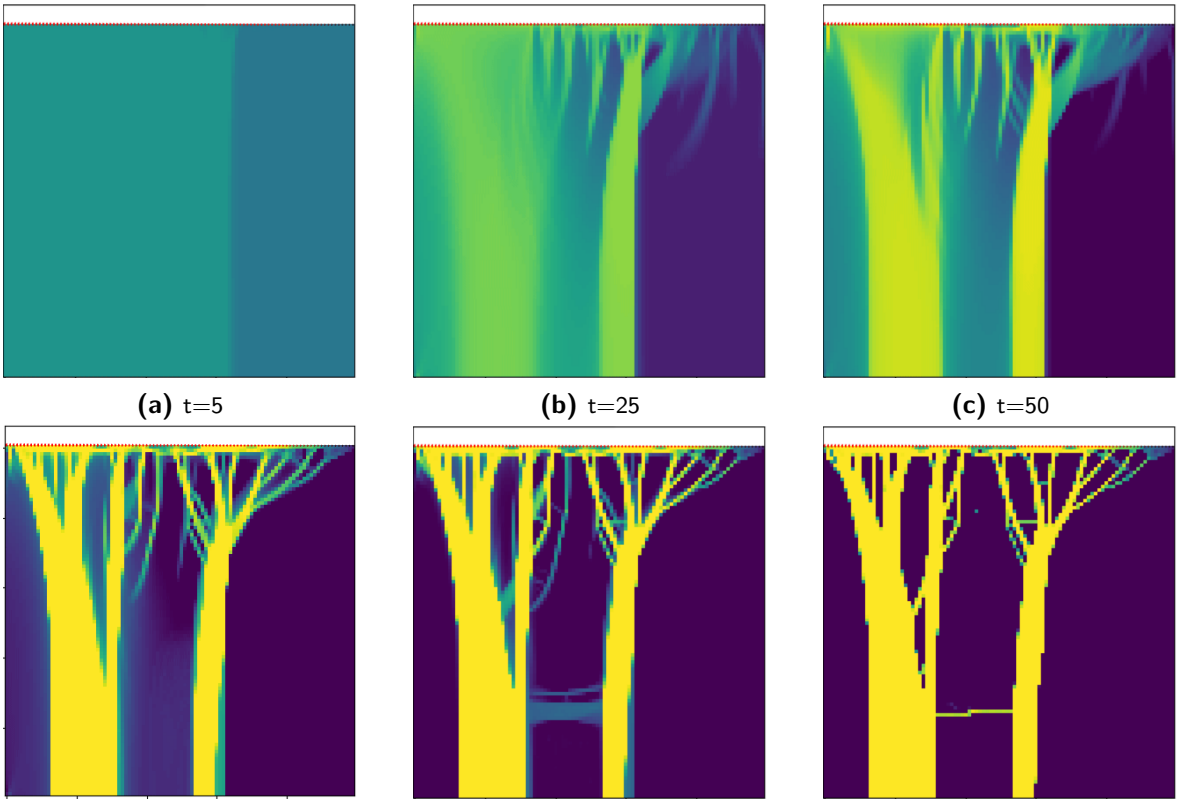
3-3 Forward Model Validation

A systematic validation assessed the cumulative impact of these modifications. This section runs the adapted model and compares the results to both the original paper by Weinans et al., and the baseline forward model from Bansod et al. [8], [44]. We conclude the section by presenting additional test load profiles to validate model robustness and sensitivity.

3-3-1 Time Domain Analysis

To validate the fidelity of the adapted model after modification, we added a plotting and animation feature. Matplotlib and PyVista are two Python packages specifically designed for plotting and visualisation. This tool can make a GIF file displaying the bone density adaptation over time, which is used to analyse convergence behaviour.

The convergence of the elements creates a smooth structure that demonstrates how stresses propagate through the material, as shown in Figure 3-7. The later steps refine the model until most elements have converged to either the maximum or minimum density.



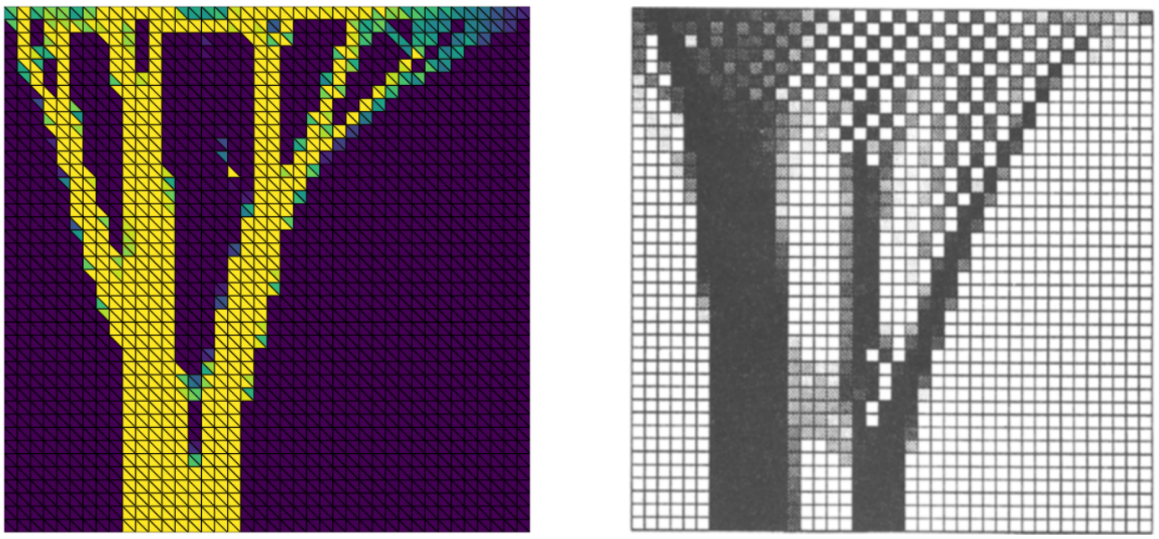
(a) t=5 **(b)** t=25 **(c)** t=50

(d) t=75 **(e)** t=100 **(f)** t=250

Figure 3-7: Density morphology while moving towards the steady-state solution. Density starts smooth, but each element converges to either high (yellow) or low (purple) density.

3-3-2 Comparison to Literature Models

The first validation step consists of comparing the model to established literature using a common loading condition. Weinans et al. use a triangular profile as a loading condition [8]. The load on the left-most element is the highest, while the load on the right-most element is zero. This results in the density profile shown in Figure 3-8.



(a) Adapted Model **(b)** Weinans et al. [8]

Figure 3-8: The 40×40 element density of the adapted forward model and the model by Weinans et al. show similar morphological structure.

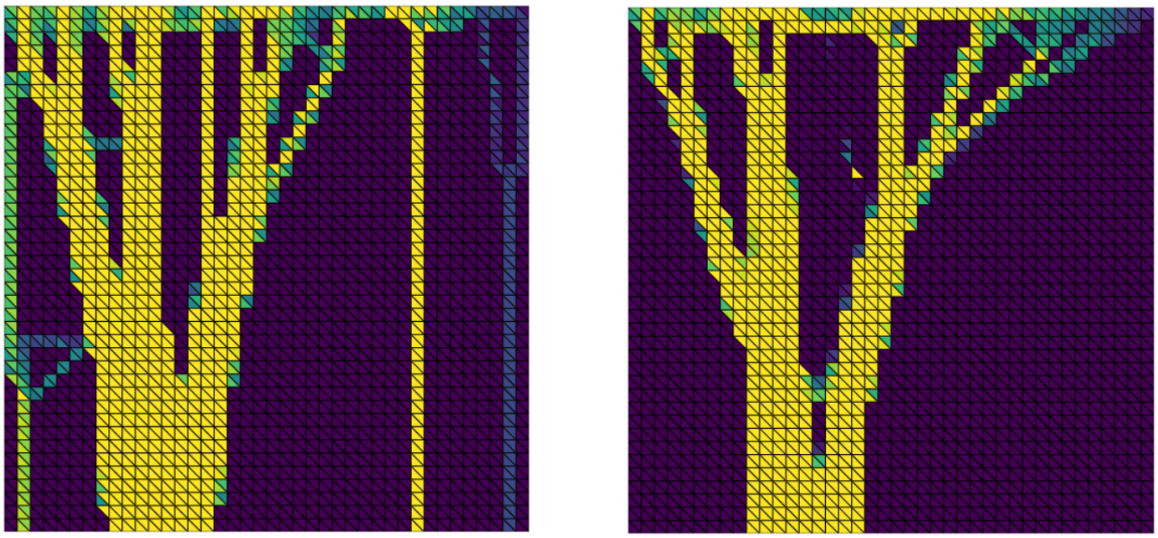
The comparison reveals that the results are morphologically similar. The most obvious difference is the checkerboard pattern that is absent in the adapted model. Furthermore, the supporting pillars are located in slightly different positions and have varying widths. Similar simulations at lower resolution are provided in Appendix C-1 and produce the same differences.

The absence of the checkerboard pattern is attributed to the different types of elements employed. The higher-order elements of the new simulation do not transfer loads to the diagonal elements, rendering the checkerboard pattern non-load-bearing. The location and width of the supporting pillars are related to the forward tissue adaptation model used in the simulation, as discussed in more detail later in this section. The Weinans et al. paper also includes other resolutions, which are also resimulated using the adapted forward model and shown in Appendix C-1.

The model is also compared to the Bansod et al. model shown in Appendix C-2. The conclusions are the same as those in the comparison by Weinans et al. The checker pattern is absent, and the pillar locations differ. This difference is what we expected, as the baseline model also uses the different element order and tissue model.

3-3-3 Impact of Forward Tissue Adaptation Model

The constants used in the adapted model are identical to those used by Bansod et al. However, Rho et al. showed that the literature uses a wider range of constants, most of which employ a higher exponent in the Young's modulus equation [28]. The exponents two and three are both shown in Figure 3-9. Both pillar thickness and bone morphology are changed significantly. This comparison shows that the forward tissue adaptation model has a large impact on the resulting morphology. A higher resolution simulation with this same comparison is shown in Appendix C-4



(a) $n = 2$ **(b)** $n = 3$

Figure 3-9: A small change in the tissue adaptation exponent significantly impacts the structure of the forward FEM bone adaptation model.

After correcting for the element order between the model and baseline, the difference did not entirely disappear. We attribute the remaining difference to changes in the tissue model and the switch from square to triangular elements.

3-3-4 Validation of Load Cases and Boundary Conditions

We also tested the implementation of the modifiable loads and load placement on the side by generating three density profiles. One profile was loaded only on the top, one only on the left and one only on the right. This test yielded clearly distinct outputs, presented in Appendix C-3. They show the implementation is correct by their density morphology.

More complicated loading conditions are also tested and shown in Appendix C-5. Although they cannot be compared to previous literature, due to their non-standard nature. Visually tracking how stresses move through the material from loads to constraints does match bone remodelling theory.

In summary, the modified model appears to produce load-density pairs that align with the literature. Although the matches are not exact, the differences in density are minor. Furthermore, we can explain the differences based on the adaptations from the baseline.

3-4 Discussion and Limitations

The adapted model improves upon the original in terms of data handling, visualisation, robustness, and computational efficiency. However, the model can still be further refined. This section reflects on model quality, current limitations and possible improvements.

3-4-1 Reflection on Model Quality

The primary goal of the forward model adaptations was to enable efficient and reliable data generation. This chapter identified that this is achieved by modifying the code to have a clear input-output pipeline, which results in an ample design space that later ML methods can exploit for learning. The modifications simultaneously improved computational speed and robustness, while maintaining biomechanical fidelity.

Pipeline compatibility is one of the strengths of the adapted model. Firstly, by changing the storage type from the limited FEniCS format to a standard NumPy array, the model can directly communicate with machine learning frameworks. Furthermore, the load case was expanded to a $\mathbb{R}^{I\times\{top,right,left\}}$ array of loading conditions, resulting in a clear input-output relation.

Stability was refined by adapting the convergence scheme using exponential functions, resulting in consistent convergence across all tested load cases. In addition, improving the cross-language communication and initialisation increased the computational speed of the model by more than 50%.

The validation section demonstrated that the adapted model maintains biomechanical fidelity by comparing it to the original model of Weinans et al. [8]. We further improved the fidelity by estimating the displacement field using higher-order elements than the original, which allowed for more accurate local strain estimation and a smoother stress distribution.

To summarise, we achieved the goal of modifying a forward model to facilitate data generation by improving pipeline compatibility, model robustness and computational speed. The remaining limitations, discussed next, primarily stem from simplifying assumptions that limit biological fidelity.

3-4-2 Limitations

As this model is adapted from the model by Bansod et al., the limitations are the same unless specifically addressed in the modifications [44]. General assumptions of forward FEM bone remodelling models include no out-of-plane effects, elastic isotropic material, deterministic modelling, independence of hormones, vascularisation and microdamage repair as discussed in subsection 2-2-3.

One of the most significant model-specific limitations is the small input-output space. Although the model gives more freedom to the user than the original, the loading inputs are still minimal. Especially the limitation towards forces applied perpendicular to the edge overrestricts the model, as real bone loading produces off-axis principal strains which result from tangential components [49]. This simplification reduces the sample space, which could limit dataset variability and thus the learning capacity of the ML model.

The mesh is also limited to a rectangular domain. While this is enough for this proof-of-concept ML pipeline, studies that do full organ simulations adapt the mesh shape to match the respective bone [39]. The boundary constraints are also fixed, further reducing the model's flexibility in addressing different design problems.

Furthermore, loading conditions are time-independent, which limits the ability to simulate long-cycle fatigue modelling. Short-term cycles can be estimated by taking the average loading condition as shown by Weinans et al. [8]. However, long-term fatigue modelling is considered a realistic failure mechanism in bone material [50], which the current model is unable to simulate.

3-4-3 Possible Improvements

Future improvements should strive to solve one of the key limitations of the current model. Key limitations are addressed by improving model fidelity, expanding the load domain, and enhancing mesh and boundary freedom.

First, newer models exist that include the anisotropic properties of bone [51]. Incorporating anisotropic material tensors to capture the directional stiffness of bone would create a more realistic load-density dataset. This more complex dataset could change ML performance significantly.

Second, the loading domain is currently highly constrained, with all forces applied to the edge and perpendicular to it. Expanding the load input beyond the current $\mathbb{R}^{I\times\{top,right,left\}}$ format to all nodes of the mesh at any orientation produces a more diverse sample space. This diversity would strengthen the ability of the downstream ML models to generalise unseen loading conditions. Furthermore, building time-dependent load profiles could be another significant improvement, as they enable the solution of dynamic problems.

Lastly, modern models have extended the original forward FEM bone remodelling models to the third dimension, including out-of-plane effect [33]. This extension increases the output space to three dimensions, allowing researchers to train ML models using accurate bone density data. In turn, improving the models which would have widespread applications in the fields of locomotion, osteoarchaeology, and many more.

Although many improvements are possible, the adapted model meets the primary goal of generating a diverse dataset. The model generates over 100,000 unique load-density pairs, which are explained in detail in chapter 4.

Chapter 4

Dataset Generation

Having established the forward model's inner workings, this chapter motivates the parameter choices and describes the data generation process to ensure reproducibility. This ensures the Machine Learning (ML) models can systematically learn the mapping between loading conditions and steady-state density distributions.

The chapter achieves this by first delving into the design principles of effective datasets for ML, followed by an implementation of these principles in the entire data generation process. Lastly, the characteristics of the resulting dataset are analysed and linked to a discussion about its strengths and limitations.

4-1 Design Principles

Dataset quality has a significant impact on ML model performance [52]. Consistent ML model performance relies on consistent representation, completeness, feature accuracy (fidelity of input variables), and target accuracy (reliability of ground-truth outputs).

Optimising biomechanical fidelity is one way to increase dataset quality. The previous section already went over how this can be achieved for the forward Finite Element Method (FEM) bone remodelling model, while maintaining computational efficiency. However, realistic input data are also an essential aspect of high-fidelity datasets. In the context of bone remodelling, this means designed loading conditions should reflect realistic force profiles, which improves the dataset's realism and reduces the chance that the forward model fails due to unexpected inputs.

The simplest way to give ML models more information is to provide more unique datapoints. The creation of many datapoints relies on both the computational efficiency of the data generation and dense dataset representation. To achieve both, our simulator was optimised as discussed in the previous chapter and each datapoint was represented with a minimal amount of storage (131 floats: 100 density values + 30 force values + 1 serial number).

Dataset completeness, or domain coverage, is defined by how well distributed the data is across the entire space. Coverage of the whole realistic domain, while excluding unrealistic noise, results in a dataset that ML models can generalise more easily. Reinforcement Learning (RL) specifically can profit from a dataset including some simple samples, as it relies on exposure to progressively more complex states for effective policy learning.

With the design goals defined, the focus can shift to implementation, starting with the generation of diverse force profiles, to attempt biomechanically valid loading conditions.

4-2 Force Profile Generation

To achieve a coverage of biomechanically plausible loading conditions, a parameter force profile generator was designed that samples from families of loading distributions. This is an important aspect of the dataset, as input fidelity directly affects output fidelity as well. Furthermore, dataset diversity is a necessity for ML performance as discussed in the previous section. The section first justifies the profile types used, then discusses the combination procedure used to combine them and lastly focuses on scaling the result along the feasible domain.

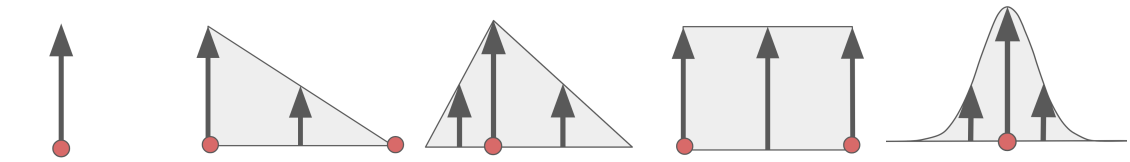
Including loading conditions at every node with arbitrary magnitude M, direction $|\Theta|$ and time dependence T, the input space would grow proportionally with $(N \times M \times |\Theta| \times T)$, leading to high-dimensional datapoints. This does not adhere to the discussed principle of dense datapoint representation. To reduce storage needs, we placed the following restrictions on the loading conditions:

- Forces exist only on the boundary of the mesh.
- Forces are normal to their boundary (zero tangential component).
- Force profiles are time-independent ($\frac{dM}{dt} = 0,\ \frac{d\phi}{dt} = 0$).
- The bottom side is excluded (as it contains the vertical constraints).

The resulting model grows linearly with the force profile resolution I. This offered both computational advantages and a simplified inverse problem that provides a logical starting point for ML.

4-2-1 Profile Types

To generate a wide variety of profiles within these constraints, we designed five different types of standard force profiles in a parameterised format: impact, ramp, triangular, square, and Gaussian (Figure 4-1). Impact force peaks represent concentrated contact forces and test the model's behaviour on highly localised loading. They are associated with high-frequency impact events during locomotion [53]. Ramp and triangular profiles mimic linear stress distributions commonly observed in bending and torsion [54]. Square profiles model uniform pressure along a surface, which can be used in partial bone simulations to approximate cross-sectional interfaces [35]. We based the last profile on a Gaussian distribution, which could represent soft tissue attachment sides, such as bone-tendon connections.



(a) Impact **(b)** Ramp **(c)** Triangle **(d)** Square **(e)** Gaussian

Figure 4-1: The five different force profiles the profile generator can generate.

All these force profile types consist of a small fixed number of parameters. For all force profile types, the profile builder first selects a side from the list $\{top, right, left\}$ to place the profile. It then determines the exact starting and ending locations on the respective side as integers between 0 and 9. Some profile types, such as the single force, only use the starting location, as this is sufficient to define the entire profile. The Gaussian distribution has an additional constant, the variance σ, which is set to one. The magnitude of the force profile is decided by uniformly sampling on the continuous range between 0.1 and 10. Lastly, the loading direction (either compression or tension) is randomly chosen.

Although each force profile generated this way is already distinct, to further increase dataset diversity, different families of loading are combined.

4-2-2 Loading combinations

To further expand the dataset variety, the different force profile types are combined to create a large number of distinct inputs. A bias towards simple loading conditions is added because accurate estimation of these simple loading conditions by the surrogate model is a precondition for the RL framework to work as intended.

The number of times a force is sampled is based on uniformly sampling from a lognormal distribution parameterised by $\mu = 0.5$ and $\sigma = 0.8$ as shown in Equation 4-1. The result is rounded to the nearest positive whole integer. The lognormal distribution is a positively constrained function and has a natural right-skewed distribution, which makes it a logical choice for this procedure. We chose the mean and variance based on a bias toward simplicity while including some intermediate and complex profiles for model generalisation as shown in Table 4-1.

$$p_n = \int_{n-0.5}^{n+0.5} \frac{1}{x\sigma\sqrt{2\pi}} \exp -\frac{(\ln(x) - \mu)^2}{2\sigma^2} \tag{4-1}$$

Table 4-1: The lognormal distribution is parameterised by $\mu = 0.5$ and $\sigma = 0.8$, resulting in a bias towards lower complexity force profiles inside the dataset.

	Simple		Intermediate			Complex
Force Profile Count [-]	1	2	3	4	5	>5
Probability [%]	45.3	24.6	12.8	6.9	3.9	6.6

The force profiles are sampled independently of each other, which means the same family type can exist multiple times inside the dataset. Furthermore, the family of impulse force

profiles can generate more than one force at the same time using the same above-described procedure to increase dataset diversity further.

Figure 4-2 visualised the additive combination of force profiles. In the current generation procedure, no attention is given to the force magnitude in relation to the simulation. The last step of the generation procedure improves this by scaling to realistic magnitude values.

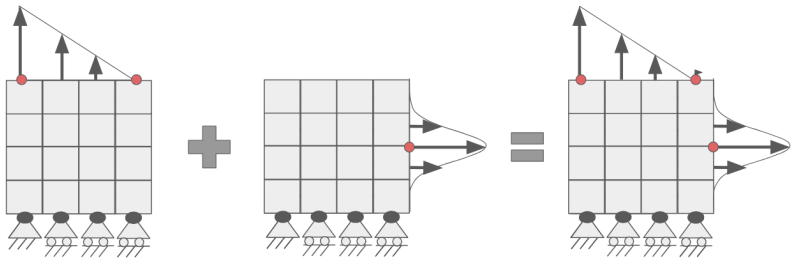


Figure 4-2: The merger combines two force profile types to create a new loading condition.

4-2-3 Magnitude Correction

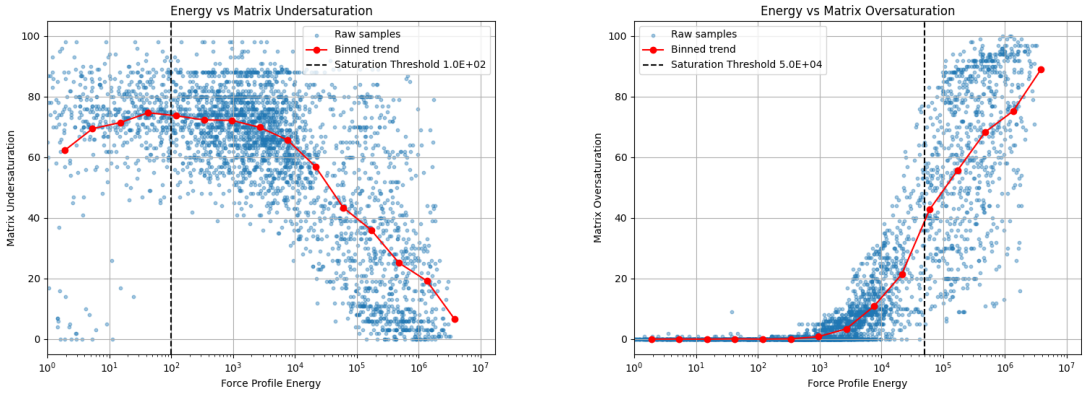
The arbitrary force magnitude loading conditions from the above procedure are a problem because they are not guaranteed to be in the realistic domain of the simulation. Too low force simulation results in undersaturation of the density distribution (most cells converge to the minimum density), while very high force simulations result in oversaturation of the density distribution (most cells converge to the maximum density). Before rescaling the loading conditions, the exact bounds at which saturation occurs first need to be identified.

The bounds of the model were tested by simulating 100,000 samples generated using the above-described procedure and batchseed 42. The samples were randomly scaled to an energy value between 1×10^0 and 5×10^6. Here, 'energy' refers to a quadratic norm of all forces, which is used as a scalar measure of loading intensity, not physical strain energy and is calculated using Equation 4-2. The resulting densities were plotted based on their percentage of over- and undersaturation in Figure 4-3. The red average line clearly demonstrates the over- and under-representations for very low and high energy samples. This resulted in the lower and upper bounds being defined at 1×10^2 and 5×10^4.

$$E = \sum_j F_j^2 \tag{4-2}$$

With the energy bounds established, the loading distributions can be resampled to uniformly cover the feasible domain. A log uniform sampling is used to ensure all orders between this interval are evenly represented in the final data.

Combined, these adaptations result in a large dataset of loads, which spans the feasible bone loading domain.



(a) Undersaturation **(b)** Oversaturation

Figure 4-3: The thresholds of the dataset are decided based on the over- and undersaturation of the element at the maximum or minimum density distribution for a random sampling of the dataset.

4-3 Large-scale Dataset Sampling

Chapter 3 established how the forward FEM bone adaptation model operates. The force profile generator must now pass all loads to the forward model. Running the forward model represents a computational bottleneck in the pipeline, as it involves numerous calculations. This section establishes a pipeline implementation that optimises the dataset generation. We first describe the pipeline structure, followed by an evaluation of the pipeline's data generation performance.

The pipeline is based on maximising forward simulation performance through parallelisation and consists of the model builder, loader, splitter, forward model, data collector, and JSON writer. The way they interact is shown in Figure 4-4.

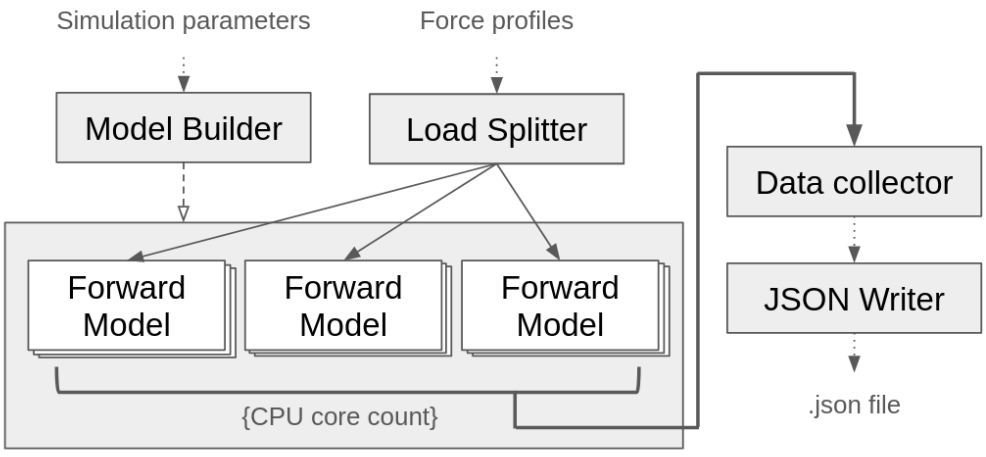


Figure 4-4: Block diagram visualising the information flow of the data generation.

The model builder initialises one forward model for each core of the Central Processing Unit (CPU). It builds all components that do not change between runs to save time on reinitialisation, including the mesh, initial density, constraints, and simulation constants. The initial density remained constant as all uniform initial densities result in similar steady-state solutions [44]. Constraints and simulation constants remained the same as those in the baseline model, while the mesh resolution was reduced from 40×40 to 10×10. This size strikes a balance between computational efficiency and observable bone morphology. This choice enables the Supervised Learning (SL) model and RL agent to identify spatial patterns, while maintaining a realistic timeframe for dataset generation.

After initialisation, the load splitter distributed the force profiles over the different CPU cores. The data collector retrieved the simulation outputs and combined all data points into an extensive list, which it stored in a JSON file. Each entry of the list has a unique serial number, a force profile, and a final output density, as is shown in Appendix B. This structure created a reproducible dataset, as a serial number for randomisation coupled with each unique force profile.

We utilised the data generation pipeline to generate two datasets: one for training the SL model (surrogate) and one for training the RL agent (inverse problem). The RL model was unable to solve the entire inverse problem, which is why we also used a second simplified dataset. We simplified the problem by reducing the loading diversity to a single type, the triangular profile. Exact justification for this simplification is shown in chapter 6. The large dataset, which encompasses all force profile types, comprises 100,000 data points, whereas the smaller dataset contains 16,000 data points.

The computational strategies from the forward model, combined with the dataset generation procedure, significantly sped up the simulation. We simulated the hardware shown in Appendix D. Running a benchmark simulation without the stability improvements yielded a simulation speed of one datapoint per 0.6 seconds, which is around twice as fast as the baseline model ($1.4s$). After the stability improvements were turned on (higher-order elements and loosened convergence), the simulation speed slowed down to 1.6 seconds/sample. We generated the entire dataset of 100.000 datapoints by running three simulation sessions, which had a combined runtime of 45 hours.

However, whether the newly generated samples satisfied the original design goals still needed to be analysed.

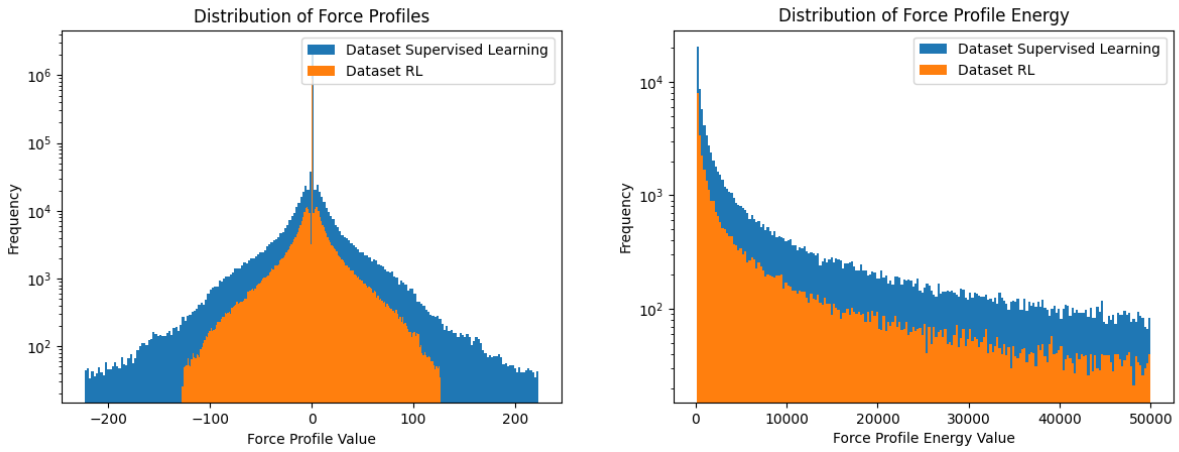
4-4 Dataset Analysis

The datasets are analysed on the original design goals of diversity, coverage, and biomechanical realism. The validation combines dataset distribution figures and key dataset metrics to explain the biases still in the dataset and whether they are expected structural biases inherent to the biomechanical problem or generation biases. The analysis proceeds from input (force) to output (density), distinguishing expected structural biases due to biomechanics from generation biases caused by data synthesis.

4-4-1 Force Profile Analysis

The force profile magnitude is plotted to test for bias in the force profile generation, as shown in Figure 4-5a. Both the SL and the RL datasets exhibit a smooth distribution without peaks. The RL dataset only uses the triangular force profiles, which can be seen in the distribution of force profiles. As all force profiles are loaded along an entire side, the highest energy profiles have lower maximum magnitudes than the SL dataset.

The energy distribution visualisation, shown in Figure 4-5b, demonstrates that the data is indeed redistributed logarithmically between the energy bounds. Both the SL and RL datasets are indeed well spread out over all energies, confirming the dataset's coverage.



(a) Force **(b)** Energy

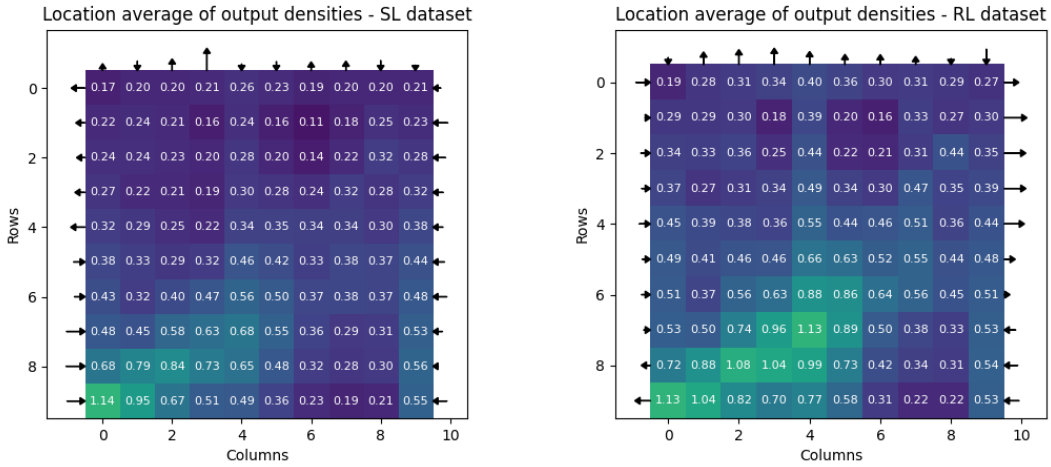
Figure 4-5: Histograms of the force and energy, visualising how the datapoints distribute across the datasets.

The last step consists of analysing any location bias in the input data. This is tested by plotting the mean value at every force location for both datasets, as shown in Figure 4-6. Visually, the forces are scaled to accentuate the bias in the datasets. The randomness in the magnitude and loading direction of the arrows demonstrates that the SL and RL datasets do not have any large bias in the force distribution.

Although the visual plots strongly suggest no bias is present in the dataset, metrics provide an additional check that this is indeed the truth. The coverage fraction and location entropy are calculated to verify this in Table 4-2. The coverage fraction is the number of force locations that have at least one datapoint at which they are not equal to zero. They confirm that both datasets are entirely covered.

The location entropy is calculated using Equation 4-3, where a higher value is a relative measure of a more diverse dataset. The location entropy for the SL dataset is higher as this dataset uses all the profile types, while the RL dataset is limited to the triangular profiles.

$$\mathrm{H}(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i) \tag{4-3}$$



(a) Supervised learning dataset **(b)** RL dataset

Figure 4-6: The average magnitude of the force and density at their respective location. High-density areas (shown in yellow) are close to the fixed constraint in the bottom left corner.

Table 4-2: Summary of key input force profile diversity metrics for both datasets.

	SL dataset	**RL dataset**
Coverage fraction [-]	1.0	1.0
Location entropy [-]	0.989	0.962

In summary, the force profile datasets are highly diverse and cover the entire domain, confirming the success of the force profile generator. No systematic biases were observed within the force profile dataset. Biases that were unobserved by visualisation and metrics are unlikely to materially affect learning negatively, which is a robust foundation for analysing the output diversity in the next section.

4-4-2 Density Distribution Bias

The density distributions contained some bias in both location and magnitude. As steady state density distributions are dependent on simulation initial conditions and model robustness, identifying the origin of these biases can be a challenge. Distinctness in the outputs is important as it simplifies the inverse problem. That is why this section looks at the location and magnitude distribution of density.

Both datasets have location bias as the elements close to the constraints have a higher mean density, shown in Figure 4-6. This bias is consistent with biomechanical theory, as all stress passes through this region, especially the bottom left corner, which contains the only fixed constraint of the simulation. The top left and right corners have the lowest average density as they are furthest away from the constraints, and stresses only pass through this region if forces are placed at the corners. The RL dataset has a higher density in the middle. This makes sense, as this dataset only contains triangular profiles that span an entire side, resulting

in stress trajectories that consistently move through the centre elements. The location bias of the densities cannot be mitigated within the current simulation setup, as they are expected structural biases inherent to the biomechanical problem and not a result of bad domain coverage by the input data.

The smooth distribution of the density magnitude along the feasible bone domain demonstrates that the dataset is relatively unbiased, and is shown in Figure 4-7. Three peaks can be identified, one at the maximum (1.74), one at the minimum (0.01) and one in the centre of the distribution (0.86). The peaks at maximum and minimum density represent the saturated elements on both sides of the dataset. The centre peak is likely a side effect of the mapping from triangular to square elements. We hypothesise that one of the triangular elements has converged to the maximum density while the other converged to the minimum density, resulting in a peak at their average.

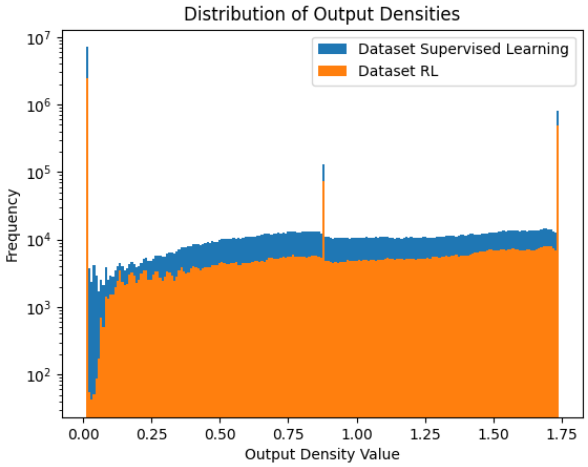


Figure 4-7: Histograms of the density, visualising how the datapoints distribute across the datasets.

To summarise, the bias in the density distribution output can be attributed to the design choices in constraint placement and data discretisation. Combined, the input and output bias are minimal, resulting in a diverse dataset that is effective for ML.

4-5 Discussion and Limitations

The goal of this chapter was to generate a sufficiently large and diverse dataset to enable effective learning of bone remodelling behaviour. We established that this can be achieved by simulating the model on data that is high-fidelity, high-quality and unique. A pipeline able to generate such a dataset was designed, run, and the resulting dataset was validated on diversity, coverage and biomechanical realism. This section will go over the main strengths and limitations of our approach. First, by looking at the representativeness of the dataset, then the implications this has on ML and lastly, possible future improvements.

4-5-1 Dataset Coverage and Representativeness

The large SL dataset has strong coverage of the domain. Five representative load cases that are found in biomechanical simulations are represented, resulting in a high entropy dataset (load location entropy of 0.989 and 0.962).

The dataset does have underrepresented areas in load profiles and density profiles. For example, when loads were placed on all edges of the model, a circular density profile was obtained (low density in the centre, high density close to the boundary). Not many of these datapoints were included in the dataset, which decreased SL performance, shown in detail in chapter 5. The model was also biased to simple profiles ($P_1 + P_2 = 70.1\%$ of the data), although this was by design, it still impacted the variation of the dataset.

4-5-2 Implications for Learning

The underrepresented areas do have implications for the ML models' learning ability. ML models rely on sufficient representation in the dataset to learn datapoint behaviour. The differences between the two datasets and their input-output direction affect the ease of learning directly.

The SL model (surrogate) will likely be easier to train as we directly influence the input set to be highly diverse. However, since 70% of samples consist of simple profile types, the SL model might underperform on complex multiload scenarios. Furthermore, the dataset consists only of static load cases that have reached steady-state. It is unlikely that the models can estimate dynamic bone remodelling behaviour, as it is not contained in the dataset.

While the high input variability is a strength for forward problem estimation, for the inverse problem, it becomes a liability. Estimating a load profile from many different types increases the search space compared to single load identification. The RL dataset thus only uses a single type, the triangular profile. A consequence of this simplification is that the agent can only identify triangular force profiles. However, as this is a feasibility study that only tests RL as an alternative, this simplification is justified.

4-5-3 Future Extensions

The large dataset is already extensive and covers all energy values until over- or undersaturation of the simulation. The dataset was even reduced to a smaller subset to simplify the problem for the RL agent. Future work may prioritise improvements in loading realism over further increases in dataset size.

Furthermore, the dataset is currently limited to steady-state samples. Future work could explore a dataset with three variables, where the loading condition is not mapped to the steady-state density. But instead, the dataset contains input density, load and output density. This way ML models trained learn the true bone remodelling dynamics, not a quasi-static case, which would allow multi-load estimation.

To summarise, the current dataset is more than sufficient for the required ML tasks, and bias is unlikely to have a large impact on model learning. We now use the dataset to estimate the forward bone remodelling process by training the SL model (surrogate) in chapter 5.

Chapter 5

Forward Estimation

The proposed framework for the inverse problem relies on the steady-state density solution in each learning step of the Reinforcement Learning (RL) agent. The forward Finite Element Method (FEM) bone adaptation model can generate this, but its computational costs of 1.6 seconds per sample render it infeasible for RL training, which requires millions of evaluations during training. A neural network-based surrogate model offers a practical alternative as it can approximate the steady-state solution, trading reduced accuracy for computational efficiency.

The chapter first introduces the surrogate architecture, learning process, and evaluation metrics, followed by model training and performance analysis. It concludes with an ensemble model and a discussion of limitations and future improvements.

5-1 Model Design and Training Methodology

To create a clearly defined surrogate with reproducible model performance, this section defines the model structure, the training procedure, and the code implementation.

5-1-1 Neural Network Structure

We explored several different convolutional models to select a neural network structure. The initial structure consisted of two convolutional layers followed by a network of ReLU nodes, which is a common Convolutional Neural Network (CNN) structure. However, this model did not reach the necessary performance. Input-layer kernels underperformed, as the force profile (input) lacks meaningful spatial information. Because the density profile (output) does contain spatial information, the network topology was inverted.

The convolutional layers at the end enable filtering of the predicted density spatially, which enforces local continuity in the final model output. The final structure used for the surrogate model combines three fully connected layers with three convolutional layers, as shown in

Figure 5-1. This combination is chosen because the final density structure has a strong spatial connection to its neighbours, which the convolutional network can capture. Lastly, we added batch normalisation to the convolutional layers. Batch normalisation stabilises intermediate feature distributions, which increases the model's performance on unseen testing data. The resulting CNN structure balanced model complexity and remodelling fidelity, establishing a suitable surrogate for integration within the RL framework.

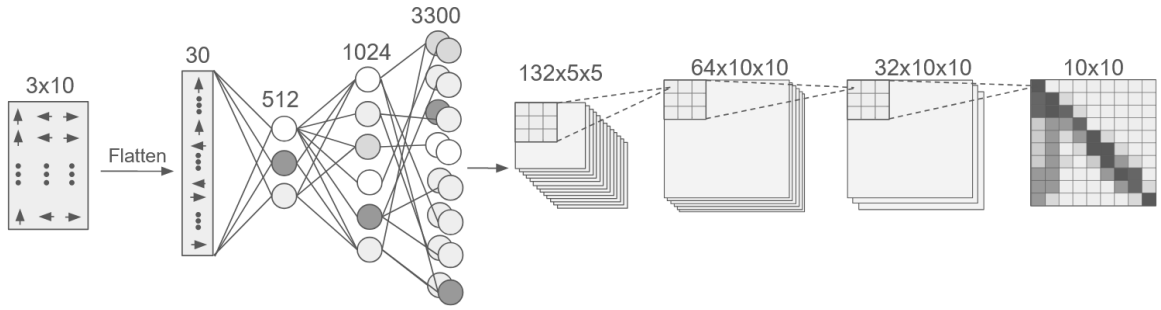


Figure 5-1: The surrogate model information flows from the input (force profile), linear and convolutional layers to the output (density distribution). The first fully connected layers contain a ReLU activation function for each node, with the last layer moving from the 3300 nodes to the spatial structure containing a dropout rate of 30% to promote model generalisation. The last fully connected layer is the same size as the first spatial layer, and one-to-one mapped before training starts. The spatial layers are connected using a 3 by 3 kernel, a stride of 2 and 1 single pixel of padding.

5-1-2 Training Procedure

The 100,000 samples were preprocessed before commencing training to improve the Supervised Learning (SL) model's performance. The dataset is split into training, validation, and test sets $(70\%, 15\%, 15\%)$. The split was performed by assigning samples randomly to one of the sets using seed number one. This ensured hyperparameter tuning and final evaluation remained unbiased.

The dataset was normalised by calculating the mean ($\mu_{x,c}$) and standard deviation ($\sigma_{x,c}$) of the training set for each input and output channel using Equation 5-1. The input set is $x \in \mathbb{R}^{C_x \times N}$ and the output set is $y \in \mathbb{R}^{C_y \times N}$. Each channel is normalised separately to ensure all channels are on a comparable scale and equally represented in the neural network. The validation and test sets are not included in this calculation to avoid data leakage, ensuring the performance evaluation reflects unseen conditions. The mean and standard deviation for each channel are saved because they are needed to preprocess data during forward estimation.

$$\tilde{x}_c = \frac{x_c - \mu_{x,c}}{\sigma_{x,c}}, \qquad \tilde{y}_c = \frac{y_c - \mu_{y,c}}{\sigma_{y,c}}, \tag{5-1}$$

The constants used during training are defined in Table 5-1. A learning rate scheduler improved the training cycle efficiency by gradually lowering the learning rate based on plateau behaviour. Early stopping prevented overfitting and reduced the model's training time. We

set the batch size to 32 with shuffle on to improve model generalisation and used the first-order gradient-based optimiser AdamW, which is standard for deep SL models.

Table 5-1: Training and optimisation hyperparameters used for surrogate model training.

Parameter	Value
Batch size	32
Shuffle	Enabled
Optimisation algorithm	AdamW
Optimizer scheduler	ReduceLROnPlateau
Initial learning rate	1×10^{-4}
Learning rate factor	0.5
Minimum delta	1×10^{-4}
Scheduler patience	25 epochs
Early stopping patience	50 epochs
Maximum epochs	1000

The loss function is the last important step before training, as it defines the difference between the target value and the model's prediction. It can have a large impact on model performance and convergence speed. To define the loss function, we need a metric that can capture the similarity between two bone density distributions. This is analogous to comparing grey-scale images, for which numerous similarity metrics exist.

Several similarity metrics commonly used in the imaging field were considered, including mean squared error, mean absolute error, cosine similarity, Dice coefficient, structural similarity index, and the Wasserstein distance. Mean-error-based metrics fail to capture spatial structure, while more complex distribution-based metrics, such as the Wasserstein distance, are computationally expensive and less stable. A combined loss function of the Mean Squared Error (MSE) and the Structural Similarity Index Measure (SSIM) was used as a compromise between training stability and capturing density morphology, as shown in Equation 5-2.

$$\mathcal{L} = \frac{1}{2}\mathcal{L}_{\text{MSE}} + \frac{1}{2}(1 - \mathcal{L}_{\text{SSIM}}) \tag{5-2}$$

The first component of the loss function is the MSE, as shown in Equation 5-3. The MSE provides more weight to large error differences between the target and estimates. This ensures the model tends away from large errors in the output, which improves training stability. Furthermore, the MSE is the standard in SL models as it naturally emerges from the assumption that errors are normally distributed around the regression target [55].

$$\mathcal{L}_{\text{MSE}} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \tag{5-3}$$

The SSIM accounts for both magnitude and morphological differences, while being simple to implement and relatively low cost. The SSIM quantifies similarity based on frequency information, mimicking aspects of human perception. This is valuable for evaluating performance, because the SSIM can recognise spatially shifted but morphologically consistent data [56].

With the preprocessing steps, training constants and loss function defined, combined with the structure from the previous sections, we moved on to training the model.

5-1-3 Implementation and Runtime

The model was trained in Python using PyTorch, an ecosystem of tools and libraries for Machine Learning (ML) implementation. The hardware and exact environment used are shown in Appendix D (identical to dataset generation). The entire training procedure used is defined in the pseudo-code in Appendix J Algorithm 1.

Finally, the model was trained on the dataset explained in chapter 4. Training produced the loss trajectories shown in Figure 5-2. Training terminated after 198 epochs, with each epoch averaging 27.7 seconds, corresponding to a total training duration of approximately 91 minutes. The model with the best performance on the validation set was kept to avoid overfitting to the training data. The training and validation losses of this best-performing model were 0.0805 and 0.1754 at epoch 148.

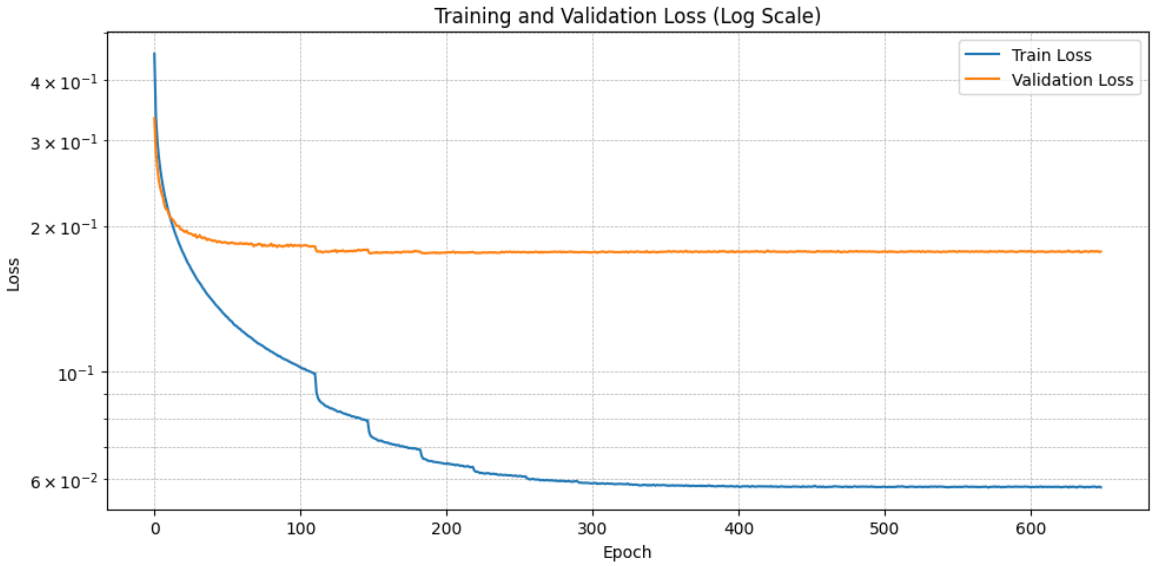


Figure 5-2: The training and validation loss are plotted with a logarithmic y-scale and show a rapid decrease for the first 20 epochs, after which the training and validation loss slow down. After 115 epochs, the surrogate model loss decreased further after the learning rate was halved. The validation loss stopped improving after 148 epochs, and only the training loss decreased further due to model overfitting. The training normally terminates at epoch 198, but this plot shows the entire loss function until training loss also stagnates to exemplify no further improvement.

The model that received the lowest loss value on the validation data is saved and evaluated in the next section using the previously saved test data.

5-2 Forward Estimation Results

The trained models are validated on the test data to analyse their ability to generalise to unseen samples. First, the performance of the single surrogate model is analysed using quantitative metrics and visualised alongside the ground-truth. The section ends with the introduction of an ensemble model that combines multiple surrogates to improve model accuracy.

5-2-1 Single Model

The surrogate model achieved an average SSIM of 0.85 on the test dataset, indicating that the model captures the dominant morphology of most density distributions in the test set. Comparing samples between the original forward data and the surrogate provides a qualitative indication of performance. Figure 5-3 is an example that shows the model can estimate single load profile density distributions with high accuracy in both morphology and absolute error. The surrogate performance on other single distributed profiles like the Gaussian, triangular, ramp and square profiles resulted in similar outcomes and are shown in (Appendix E).

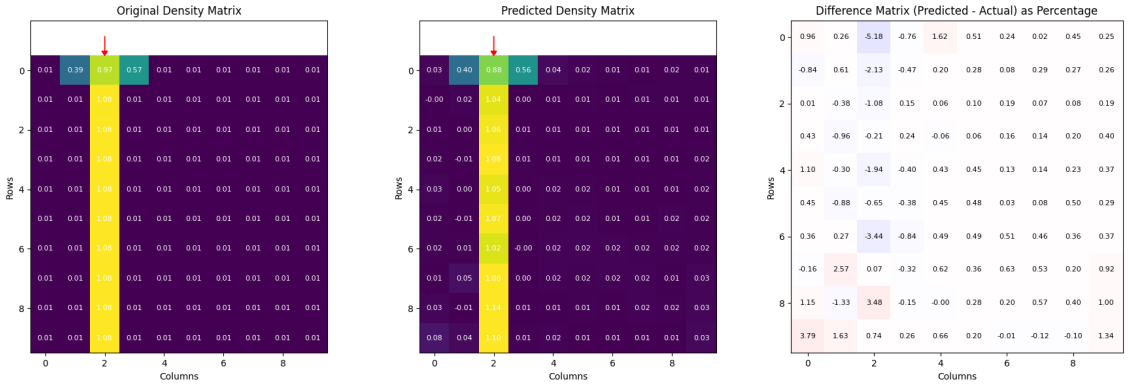


Figure 5-3: A comparison between the original forward bone adaptation model (left) and the surrogate estimation (middle) for a single concentrated load shows that the surrogate closely matches the reference in both magnitude and spatial morphology. This is reflected by an SSIM of 0.98, which is close to the theoretical maximum of 1. The right picture contains a percentage difference plot which shows the maximum density element error is 5.18%.

Samples consisting of a combination of two loading profiles still perform reasonably well, demonstrated by the example in Figure 5-4. Gradient regions (elements on the high-to-low-density boundaries) have higher errors than the single distribution samples.

To identify limitations of the surrogate model, we analysed the worst-performing samples from the test set. The surrogate model fails to generalise to multi-load force profiles in the dataset, as shown in Figure 5-5. This behaviour was expected since these complex cases are underrepresented in the dataset. Expanding the dataset distribution to include more complex load profiles will likely improve the ability of the model to generalise to such cases.

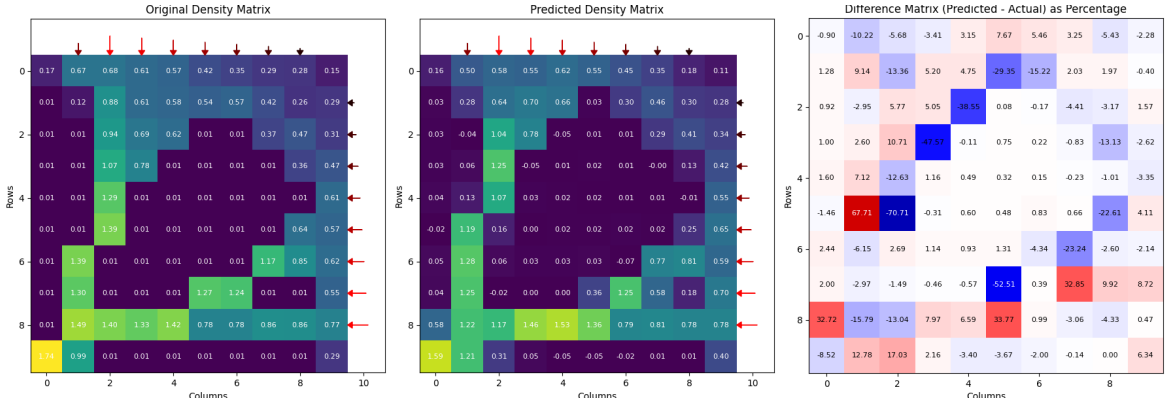


Figure 5-4: Triangular loads along the top and right side show that the surrogate matches the reference at most density locations. The received SSIM is close to the average of the surrogate at 0.67. The right picture contains a percentage difference plot, which shows that some elements in the mesh have large errors compared to the ground truth.

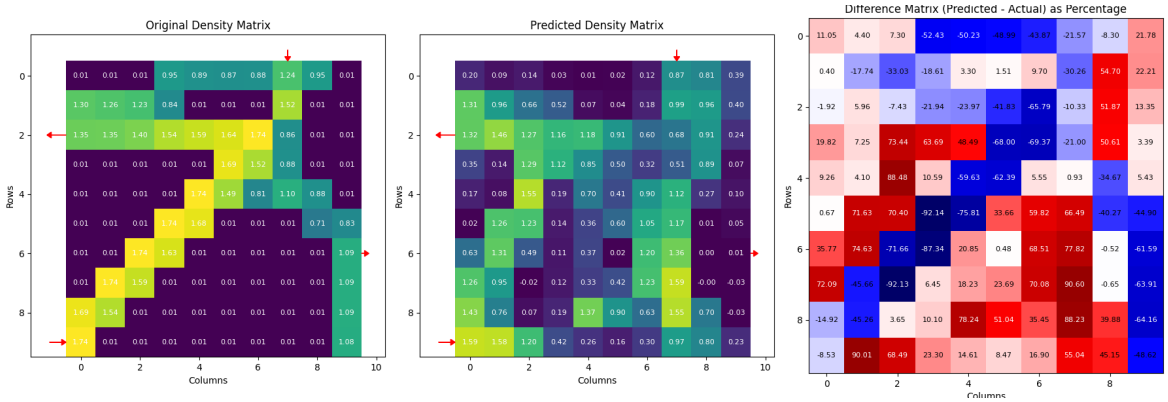


Figure 5-5: The result from the test-set with the lowest SSIM shows the surrogate is unable to simulate complex load conditions. This sample, consisting of four point loads spread out over the sides of the simulation, received an SSIM of -0.086. The difference plot on the right shows elements are both over- and underrepresented with error of more than 50%.

5-2-2 Ensemble Model and Uncertainty

The surrogate model can accurately predict the density distribution for single loading conditions. The performance of the RL framework is limited by the performance of the surrogate model. Any improvement in surrogate accuracy directly increases the likelihood of a successful final model. Furthermore, noise in the surrogate could destabilise the RL framework, motivating the use of ensemble averaging to reduce variance. We implemented a bootstrap aggregating ensemble for these reasons. The surrogate model is trained multiple times on a different subset of the training and validation data, and the combined average is used as the new output.

The ensemble consists of five versions of the base model. We selected an ensemble of five models, as the improvement offered by an additional surrogate diminishes with each model, while five models are enough to obtain an accurate mean and standard deviation. To increase ensemble performance further, each surrogate was initialised with different random starting weights.

The ensemble improved the SSIM from 0.847 to 0.871, as shown in Table 5-2. Although this improvement appears insignificant, at high SSIM any small improvement in the performance metric indicates a large jump in forward estimate performance. Illustrative results of the ensemble model for all different force profile types are shown in Appendix F.

Table 5-2: The surrogate model achieves an over 2000% speed-up compared to the FEM, at the cost of bone density fidelity. The ensemble model improves the accuracy of the surrogate model at the cost of an increased inference time.

	FEM	**Single SL**	**Ensemble SL**
SSIM validation	-	0.950	0.969
SSIM test	-	0.847	0.871
Time per sample [s]	1.6	7.0×10^{-4}	1.0×10^{-2}

A key strength of ensemble models is their ability to estimate uncertainty in their own output. Appendix F contains samples where the standard deviation and difference plots consist of the same morphology. This proxy for the difference can later be exploited in chapter 6 to improve agent performance.

5-3 Discussion and Limitations

The neural network can approximate the forward FEM bone adaptation model accurately for most loading conditions (SSIM = 0.85). An ensemble network further improves this to a robust and reliable predictor of density distributions (SSIM = 0.87). Furthermore, the surrogate achieved a speed-up of over three orders of magnitude compared to the forward FEM model, establishing the foundation for the inverse framework.

The main limitations of the model are in predicting complex loading conditions. The current model does not have the size to fully capture the nonlinearity of the original problem, which limits its ability to generalise to unseen loading types, like the complex profiles underrepresented in the dataset.

From a supervised learning perspective, a few procedures can be used to improve model performance without expanding the dataset. Firstly, hyperparameter tuning was not systematically explored, as model performance was already sufficient for the inverse problem framework. A large performance boost could be achieved by moving from the current handpicked parameters to a more refined hyperparameter search method.

The second improvement strategy is increasing the architecture depth. Larger models can capture more complex behaviour than shallow models. At the cost of increased training and computation time, especially that last one can affect RL runtime negatively.

Overall, the results do confirm the surrogate can estimate the forward model efficiently, providing the foundation for the inverse RL framework explored in the next chapter.

Chapter 6

Inverse Model

Building on the forward estimation framework, this section addresses the inverse bone remodelling problem. In the inverse problem, the force profile is estimated based on the current bone density. The inverse problem is mathematically ill-posed and non-unique because multiple loading conditions can lead to identical density distributions, as shown in chapter 2. This causes gradient descent-based Supervised Learning (SL) methods to converge to averaged solutions (i.e., outputs largely independent of the input) and stagnate in learning the true mapping.

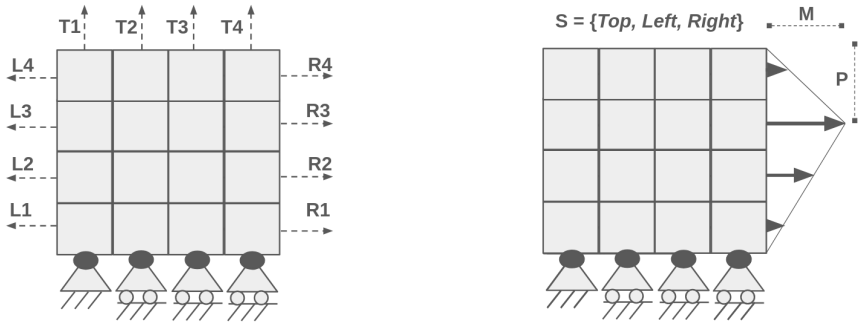
The Reinforcement Learning (RL) framework assigns rewards based on matching the resulting density rather than the exact load, which alleviates some consequences of the non-uniqueness. Furthermore, RL's state-action trajectories do not rely on the differentiability of the bone remodelling mapping. We hypothesise that an RL framework can learn a mapping between bone density and loading conditions that remains stable under this ill-posed problem.

The chapter starts with the full inverse design problem formulation, followed by the order-reduced variation used as a proof of concept. Next, the chapter specifies the implementation and training of the baseline SL model and the RL framework. Lastly, the results of the new framework are shown and discussed.

6-1 Problem Formulation

The large original inverse problem consisted of the entire database used to train the surrogate model. However, both the baseline model and the RL framework were unable to solve this problem. The original problem has a design space of thirty continuous variables, which likely exceeded the diversity in the dataset. This caused the models to converge towards an average solution, resulting in a density profile similar to that shown in Figure 4-6a, after which convergence stagnated. The design space size and density profile similarity caused the Machine Learning (ML) models to fail to identify a meaningful input-output mapping with the current size of the dataset.

To isolate the learning behaviour without the large design space restricting performance, the problem was adapted to a simple variant. The load types in the dataset were reduced to a single type: the triangular profile. The ML model did not have to predict the magnitude at each location; instead, it had to predict the side, location, and magnitude (two discrete and one continuous variable). A visualisation of the original and order-reduced variant of the problem is shown in Figure 6-1.



(a) Original **(b)** Order-reduced

Figure 6-1: The original inverse bone remodelling problem was adapted to an order-reduced variant used as a proof-of-concept for the RL framework.

This change did introduce disadvantages. For example, the dataset size decreased from 100,000 to 16,000 samples. The order-reduced variant is also difficult to upscale as the ML models' output is specifically for placing triangular loading conditions. However, losing model generalisability was a necessary sacrifice to gain a problem able to demonstrate the capabilities of the RL framework. The main differences are summarised in Table 6-1.

Table 6-1: The largest differences between the original problem and the order-reduced variant are the reduction in sample size, loading type and variables.

	Full Problem	**Order-Reduced Variant**
Variables	30 continuous	2 discrete + 1 continuous
Dataset size	100,000	16,000
Load type	5 types	Triangular only

6-2 Baseline Neural Network Model

The performance of the RL model alone does not demonstrate the effectiveness of the developed framework. A baseline model is needed as a comparison to alternative optimisation methods. We decided on an SL model, as it has already proven effective for the forward problem. Furthermore, other literature has shown that SL models can be used to solve some inverse bone remodelling problems [39].

Three different SL output structures were trained, and the best-performing model was used as the baseline. The model input remains a constant 10×10 density profile, and the internal structure is the topological inverse of the previously discussed surrogate model. The loss

function uses the Mean Squared Error (MSE) and the split was kept the same as the forward estimation, with 70% of the data used for model training. The inputs and outputs were all normalised for each channel, and the other training parameters are shown in Table 6-2.

Table 6-2: Parameters used to train the inverse baseline model.

Parameter	Value
Batch size with shuffle	32
Optimisation algorithm	AdamW
Learning rate	1×10^{-4}

The three different output SL models are compared on Structural Similarity Index Measure (SSIM) in Table 6-3. The best performing model used an output vector of 31, of which 30 nodes encoded the exact peak location and the last node was used to encode the magnitude. The node with the highest magnitude was used as the peak location of the triangular distribution.

Table 6-3: The SSIM for three different output baselines. The model that finds the exact peak location and the maximum magnitude outperforms the two alternatives.

Output	Output dimension	SSIM
{left + top + right}	10 + 10 + 10	0.05
{side + peak location + max magnitude}	3	0.14
{exact peak location + max magnitude}	31	0.28

The training and validation loss for the best-performing model is shown in Figure 6-2, and the exact training procedure is shown in Appendix J Algorithm 2. The baseline learns some of the loading behaviour, as can be seen in Appendix G, but fails in picking the correct side, as will be shown in the result section.

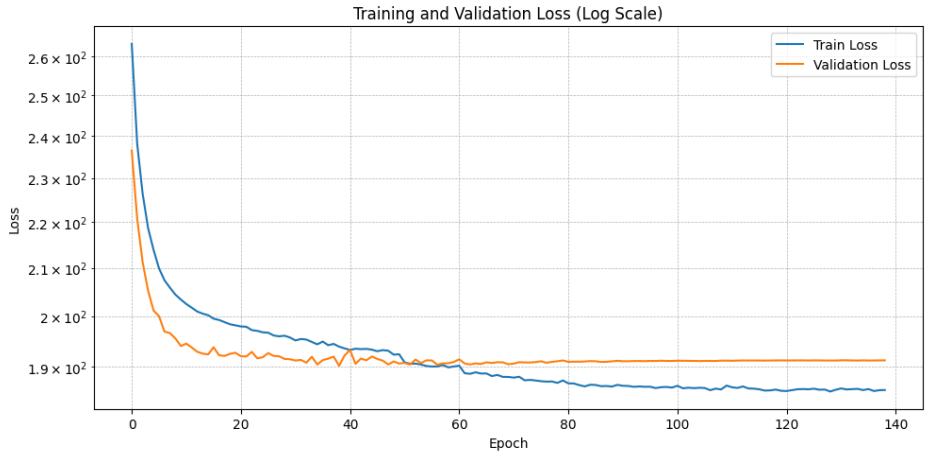


Figure 6-2: The loss of the inverse baseline model per epoch. The best validation loss occurs at epoch 47.

Even though the order of the problem was reduced, the baseline performance clearly shows

that this remains a difficult task, confirming the need for an alternative framework. We now move on towards the structure and implementation of the RL framework.

6-3 Reinforcement Learning Framework Design

This section presents the RL framework developed to address the ill-posed nature of the bone remodelling problem. The following subsections first outline the overall architecture, before going into detail about the different subcomponents.

6-3-1 Overall Architecture

The framework couples an RL agent with the simplified dataset and the surrogate model from chapter 5. The task of the RL agent is to infer a load distribution that best reconstructs the given target density. The surrogate model functions as the environment, transforming the estimated force profile into a density prediction. Combined, they enable convergence towards the original loading condition while addressing the problem of non-uniqueness.

The RL training cycle starts with loading a density profile sample from the dataset. This sample will be continuously compared to the estimated density found by the RL framework. The agent receives the density profile and an empty force profile, and decides its first two actions:

1. Move the force peak either left or right

2. Decrease or increase the force peak magnitude

All actions are relative changes to the current force profile, which improves learning stability and fine-tuned exploration.

The force updater interprets the relative action by updating the force profile to the new height and location and rebuilding the 3×10 matrix. This matrix is the current guess of the RL framework and given to the surrogate model. The surrogate model estimates the forward steady-state density corresponding to the updated force profile.

The original sample is then compared to this estimate, and the agent is given a reward based on their similarity. Furthermore, the estimated density is subtracted from the original density sample, and this difference matrix is given to the agent as an observation, together with the force profile. Each episode spans 25 timesteps because this gives the agent enough steps to adapt to any loading condition inside the dataset from the starting position. In other words, it strikes a balance between sufficient exploration while avoiding over-adaptation to a single sample. The entire process is visualised in Figure 6-3.

With the overall structure defined, the focus is shifted to an in-depth explanation of each subcomponent, starting with the "eyes" of the agent, the state space.

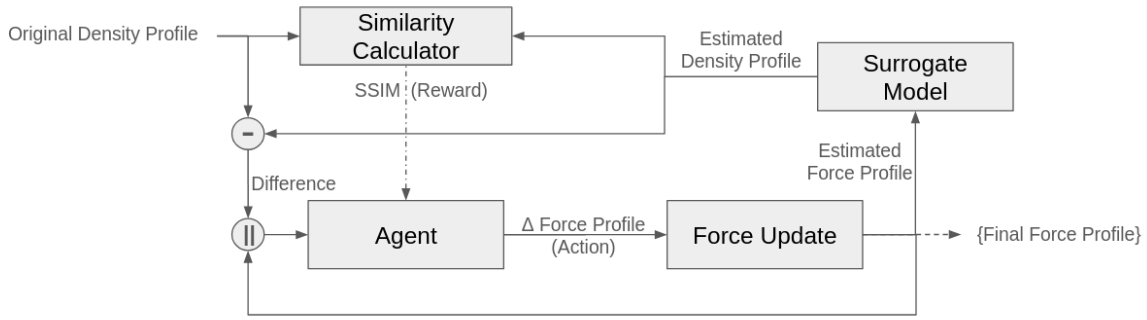


Figure 6-3: Information flow diagram of the RL framework. The original density profile is the input of the model, and the final force profile is the output.

6-3-2 State Space

The state space is an important aspect of an RL model. The agent bases its next action on the current state. Including information the agent cannot use decreases learning speed, while excluding information reduces the agent's performance. Thus, deciding the right state space is a key decision in achieving a high-quality RL model.

The agent is given two matrices: the density difference ($\rho - \hat{\rho}_t$) and the force profile estimate ($\hat{F}$). The density difference consists of the original density sample minus the current estimated density profile. The difference map represents the error between the current estimated solution and the ground truth, and highlights regions where the density should be either increased (positive value) or decreased (negative value). This removes the need for internal comparison in the agent's network and allows the agent to focus on force magnitude and location. Although this alone is enough data for the agent to make an informed decision, initial trial runs showed that including the last estimated force profile improved performance. Combined, they gave a state space of 130 continuous variables in total (100 densities + 30 force values), shown in Equation 6-1.

$$S_t = \begin{bmatrix} \rho - \hat{\rho}_t \\ \hat{F}_{t-1} \end{bmatrix}, \qquad S_0 = \begin{bmatrix} \rho \\ 0 \end{bmatrix}, \qquad \rho, \hat{\rho}_t \in \mathbb{R}^{10\times10}, \quad \hat{F}_t \in \mathbb{R}^{3\times10}, \qquad S_t \in \mathbb{R}^{13\times10}. \tag{6-1}$$

The constraints are based on the maximum and minimal density of the dataset for the top matrix ($\rho - \hat{\rho}_t$), while the bottom elements ($\hat{F}$) are restricted between the minimum and maximum force inside the dataset (Equation 6-2). In the current dataset, this means the force is limited between $\pm 200\ N$, while the density is limited between $\pm 1.73\ \mathrm{g/cm^3}$. The state space was not normalised because the density and force values have comparable magnitudes, preventing scale-related bias.

$$S_{\min} < S_t < S_{\max}, \quad S_{\min} = \begin{bmatrix} \rho_{\min} - \rho_{\max} \\ F_{\min} \end{bmatrix}, \quad S_{\max} = \begin{bmatrix} \rho_{\max} - \rho_{\min} \\ F_{\max} \end{bmatrix} \tag{6-2}$$

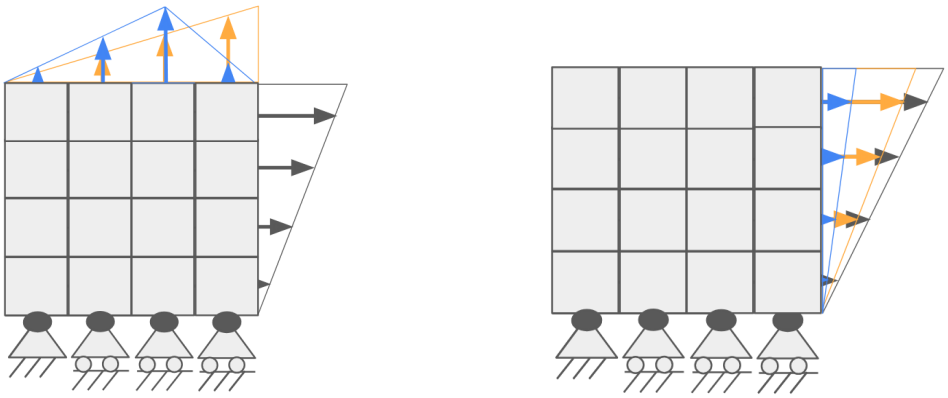
The agent uses this state and the reward to weight the previous action and decide on a new action. The actions that the agent can choose are now discussed in more detail.

6-3-3 Action Space

The action space consists of all actions the RL agent can choose to move to the next state. An intelligent choice of the action space can improve learning. High-dimensional problems are more complex to learn, because the space that has to be explored is larger. Therefore, our largest design consideration was to reduce the order of the action space without modifying the design space.

The order-reduced variant consists of two discrete variables and one continuous variable. There are two common methods to solve this problem: either treat the continuous variable as discrete by dividing it into small discrete steps, or treat the discrete variables as continuous by rounding the output values. We decided to treat all actions as continuous by rounding the agent's discrete action, because this avoids artificial local minima introduced by discretisation.

To reduce the dimensionality of the action space further, the two discrete actions were combined into a single action. The location action $\{0, 1, 2, .., 9\}$ is multiplied with the side action $\{\text{left}, \text{right}, \text{top}\}$ into one discrete state with 30 possibilities. The action space is then reduced further by moving from absolute to relative position. The agent's action does not pick the peak location, but whether to move left or right. When the agent is on the boundary and moves the peak further, it jumps to the other side by design, as is visualised in Figure 6-4.



(a) Location action **(b)** Magnitude action

Figure 6-4: The changes of the force profile over two actions in the same direction, from blue (initial) to orange and then to grey (final). The first figure demonstrates the peak action, while the second figure shows the magnitude action.

The mathematical notation of the resulting action space is shown in Equation 6-3. The first action is moving the location of the peak ($a_t^{(\text{loc})}$), while the second action is adapting the magnitude of the peak $a_t^{(\text{peak})}$. A third action ($a_t^{(\text{stay})}$) is added to ensure the agent can also decide to stay at its current peak location. The environment does not consider the move in the peak location of the first action if this value is positive. This action stabilises the policy towards a single solution. If the agent is unable to stay in the same location, the output would fluctuate between two contender locations. All actions are constrained between ± 1 to

keep relative changes small, which stabilised learning by preventing abrupt state changes.

$$A_t = \begin{bmatrix} a_t^{(\text{loc})} \\ a_t^{(\text{peak})} \\ a_t^{(\text{stay})} \end{bmatrix}, \qquad a_t \in [-1, 1]^3, \quad A_t \in \mathbb{R}^3 \tag{6-3}$$

6-3-4 Reward Function

Reward is a scalar value used by the RL agent to update its policy. The reward function calculates the reward based on the current state and action. Reward has a large influence on model performance, because the policy is updated based on it.

The main objective of the inverse problem is to match a load to the density received as input. As discussed in chapter 2, direct load estimation is ineffective due to the non-uniqueness. This is why we modified the original agent objective to the following alternative: propose a force profile whose surrogate-predicted density matches the target density. In other words, the agent should be given a reward based on the similarity between the original density and the estimated density from the surrogate model. This structure improved learning stability because the agent is not punished for proposing alternative loading conditions that are physically consistent with the given density profile.

A commonly used metric in ML applications is the MSE. The MSE punishes small shifts in model results harshly, which is not suitable due to the error between the true forward model and the surrogate. As discussed in chapter 5, the SSIM is an alternative without this limitation. Its fast computation, combined with the ability to account for morphological similarity, makes it the best for this application. This resulted in the reward function shown in Equation 6-4. Reward normalisation was added to adapt the reward magnitude to the current agent performance. The "Scikit-image" package implementation of the metric was used, which relies only on the two density profiles and their bounds ($\rho_{\min} = 0.01$ and $\rho_{\max} = 1.74$).

$$r_t = \text{SSIM}(\rho, \hat{\rho}_{t+1}) \tag{6-4}$$

The state, action, and reward build up the entire RL environment. A pseudocode implementation of this entire procedure is shown in Appendix J Algorithm 3. This demonstrates the entire environment that is run after each action from the RL agent. To complete the RL framework, the last subsection describes the policy architecture.

6-3-5 RL Algorithm and Policy Architecture

The RL algorithm that was used is Proximal Policy Optimisation (PPO), because it is a modern method that is compatible with continuous action spaces, relative actions and on-policy optimisation [12]. Furthermore, it has a promising track record when it comes to solving other similar inverse problems, shown in chapter 2.

The actor and critic (two neural networks which form the basis of PPO) base their action and estimated reward on a neural network. The network used is an Multi Layer Perceptron (MLP) for both the actor and the critic, because the dimensionality is low for the action space (three

continuous actions) and high for the input space (130 variables). A Convolutional Neural Network (CNN) is not chosen because the observation contains both density and force data, which do not match in a convolutional structure. A limitation of the MLP is that structural data of the input is lost because it is flattened.

The actor and critic both used the same network size, with two hidden layers, each with 64 units and a hyperbolic tangent activation function. Exploration is performed using a Gaussian distribution action sampling. The PPO network outputs a mean and log-standard-deviation for the two actions, and samples from the resulting Gaussian distribution based on the current entropy level.

PPO was implemented using Stable-Baselines3 because it is both computationally efficient and has large design freedom. The hyperparameters were mostly kept at the standard Stable-Baselines3 values, because the model's performance was already satisfactory and shown in Table 6-4 with a short motivation. How these parameters were used to train the PPO model is shown in Appendix J Algorithm 4.

Table 6-4: Hyperparameters used in the RL architecture.

RL Hyperparameters		**Motivation**
On-policy batch length	2048	improves advantage estimation quality
Batch size	64	stable minibatch updates
Entropy coefficient	0.01	maintains exploration
Discount factor (γ)	0.99	rewards long-term alignment
Generalised advantage estimate (λ)	0.95	standard compromise

Overall, the MLP-based PPO policy created a robust architecture for learning the bone remodelling problem.

6-4 Training and Validation

The training of the RL agent was divided into smaller learning episodes, where each episode considered a single sample. Moreover, the agent was iteratively validated using the forward simulation model. Combined, this training and the validation strategy created an environment that maximised the RL framework's performance.

6-4-1 Training environment

The agent used separate sample episodes because the relative actions of the agent need time to converge towards the optimal solution. These episodes were terminated after a maximum number of steps or after the agent produced a density estimate within a threshold of 0.05 g/cm^3 (approximately 3%). The threshold tolerance aligned with the precision of the surrogate model to ensure surrogate error does not hinder the episode convergence. The agent needed 20 steps to reach the furthest force profiles from its starting location with exact pathing. To allow for some suboptimal pathing, the maximum step count was set to 25.

The episode length is important because the discounted reward ($\gamma = 0.99$) is not calculated past the end of the episode, as shown in Equation 6-6. The discounted reward ensures that the agent's planning horizon shortens towards the end of the episode.

$$r_{23} = \text{SSIM}(\rho, \hat{\rho}_{23}) + \gamma \text{SSIM}(\rho, \hat{\rho}_{24}) + \gamma^2 \text{SSIM}(\rho, \hat{\rho}_{25}) \tag{6-5}$$

$$r_{24} = \text{SSIM}(\rho, \hat{\rho}_{24}) + \gamma \text{SSIM}(\rho, \hat{\rho}_{25}) \tag{6-6}$$

$$r_{25} = \text{SSIM}(\rho, \hat{\rho}_{25})$$

The environment state was reset by moving the force profile back to its starting position (top centre with magnitude zero). These consistent episodic starting conditions increased the learning stability. Lastly, a new density sample was loaded to start a new episode.

The model was trained using ten parallel environments, which improved the speed at which samples were collected. We used ten environments because this is the lowest number that saturated the Central Processing Unit (CPU). The surrogate models ran on the Graphical Processing Unit (GPU) to offload work from the CPU and further increase performance.

After these changes, the model had two remaining bottlenecks. The policy updates of the actor and critic were slow because the agent update had to wait until all parallel environments finished. The validation cycle, the second bottleneck, is discussed in the next subsection.

6-4-2 Validation Strategy

The model was validated every 8,000 episodes, corresponding to ≈ 1000 PPO policy updates. The validation cycle had two important tasks: reduce the learning rate after improvement plateaued, and verify that the difference between the forward Finite Element Method (FEM) model and the surrogate did not result in incorrect learning behaviour.

The validation cycle worked by running 100 episodes on a validation dataset. The final SSIM averaged over the 100 episodes resulted in a single validation score. This validation score was then used by a learning rate scheduler to adapt the learning rate of the RL agent.

The learning rate scheduler worked similarly to its SL counterpart, explained in chapter 5. The best validation score and a patience factor kept track of model improvement. The learning rate was reduced by an order of magnitude (factor = 0.1) after the validation score stagnated (patience = 20). This reduced the size of changes made to the agent over time, which resulted in refinement of the policy after learning stagnated. A minimal learning rate of $1 \cdot 10^{-5}$ was added to terminate the entire training process if the RL framework kept stagnating.

The entire procedure described above was used to train two models. Both used the same RL framework, but one was combined with the surrogate model and the other with the ensemble model. This could provide some insight into the effect that surrogate performance has on the final RL framework's performance. The average reward and the validation SSIM of the two models were plotted over time in Figure 6-6 and Figure 6-6.

The implementation of the parallelisation set the groundwork for a fast training process. While the validation cycle created a metric that could be compared to the baseline model, as is shown in the next section, the results.

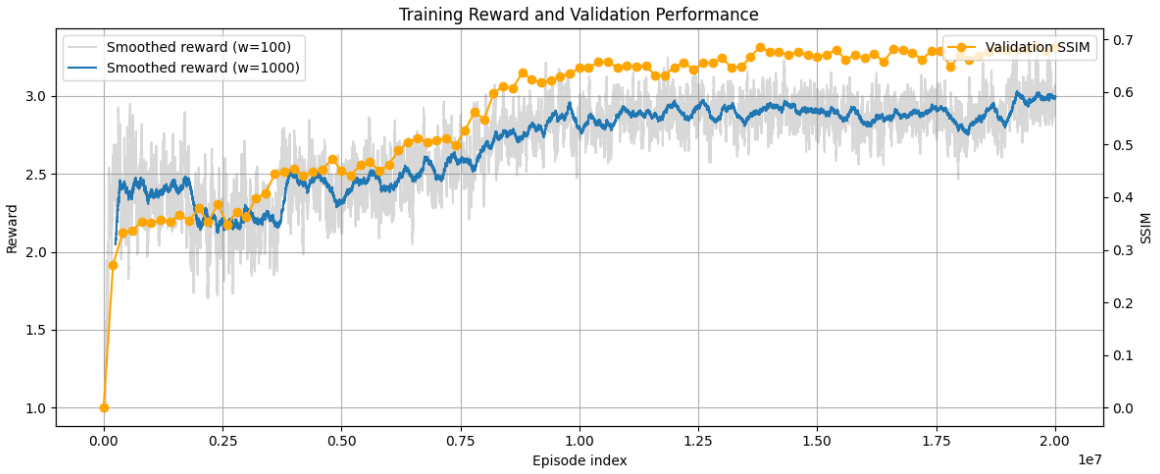


Figure 6-5: Improvement of the RL agent trained with the surrogate model for 20 million samples. The SSIM improves during the training as demonstrated in orange with the SSIM on the right axis. The training reward has a similar trajectory, and its values are shown on the left axis.

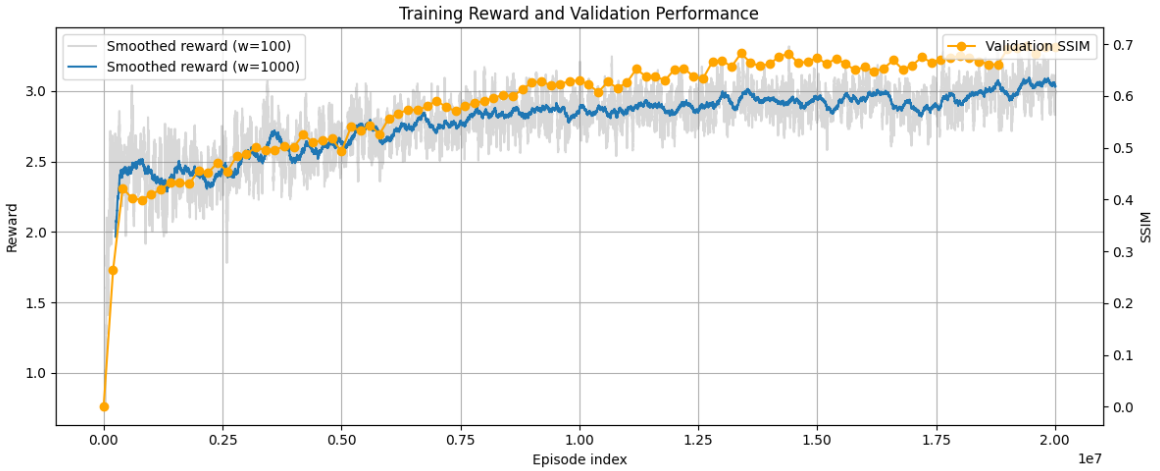


Figure 6-6: Improvement of the RL agent trained with the ensemble model for 20 million samples. The SSIM improves during the training as demonstrated in orange with the SSIM on the right axis. The training reward has a similar trajectory, and its values are shown on the left axis.

6-5 Results

This section analyses the resulting performance of the RL surrogate and ensemble agent. The RL framework is validated by comparing against the baseline SL model in core metrics and analysing the differences between the obtained loading estimations and the ground truths.

6-5-1 Quantitative Results

The quantitative results were obtained by running the RL model on a separate test set of 6000 samples, as shown in Table 6-5. The SSIM and MSE were calculated by placing the final loading distribution found by the agent back into its respective forward model. The side, peak match and magnitude difference metrics were calculated directly from the loading conditions. The peak match percentage requires both the side and peak placement to be exactly correct.

The RL model with the surrogate model reached an SSIM of 0.72, while the model with the ensemble reached an SSIM of 0.73. This represents a substantial improvement over the baseline model, which stagnated at an SSIM of 0.28. The separate metrics clearly identify that the performance difference between the RL models and the baseline stems primarily from the ability to identify the correct loading side. The surrogate model identifies the correct side 99.9% of the time, while the SL baseline is limited to only 58.4%. In the exact matching of the peak and the magnitude differences, the RL models perform similarly to the baseline.

Table 6-5: Performance of the baseline SL model and the two different RL models.

	SL Baseline	**RL Surrogate**	**RL Ensemble**
Average SSIM [-]	0.28	0.72	0.73
Average MSE [-]	0.47	0.21	0.19
Time per sample [s]	0.004	0.052	0.11
Total training time [h]	0.5	16	38
Side Match Percentage [%]	58.40	99.90	97.60
Peak Match Percentage [%]	38.75	31.8	32.70
Peak Magnitude Difference [N]	19.83	19.64	17.45

As an indication of computational efficiency, we timed the total training time and per-sample time for evaluation. The time per sample for the RL ensemble model is significantly higher than that of the RL surrogate because the RL ensemble model needs 5 times as many forward passes. The SL model time per sample is 13 times faster, while the total training time is 32 times faster. This difference is expected due to the number of forward passes required per RL step.

6-5-2 Qualitative Results

The quantitative results demonstrate that both RL frameworks outperform the baseline substantially in accuracy due to the ability to accurately estimate the correct loading side. The qualitative reconstruction figures validate these metrics. The figures of the baseline are shown in Appendix G, while the figures of the RL surrogate and ensemble are shown in Appendix H and Appendix I, respectively.

The RL framework outperforms the baseline model in identifying the correct side to place the force profile consistently, as shown in Figure 6-7 and Figure 6-8. The baseline can perform the other tasks, but creates a completely incorrect morphology due to this mistake.

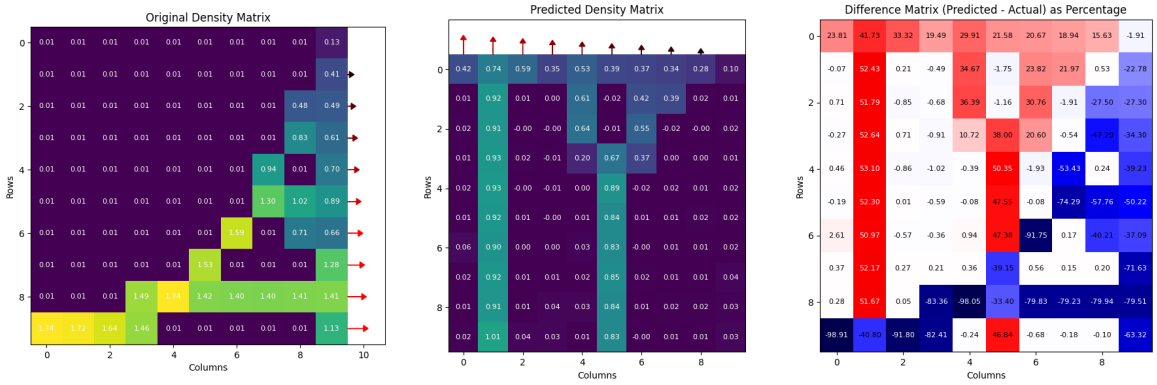


Figure 6-7: An illustrative sample of the baseline model that does not find the correct loading side. The sample received an SSIM of -0.044 as the bone morphology does not match.

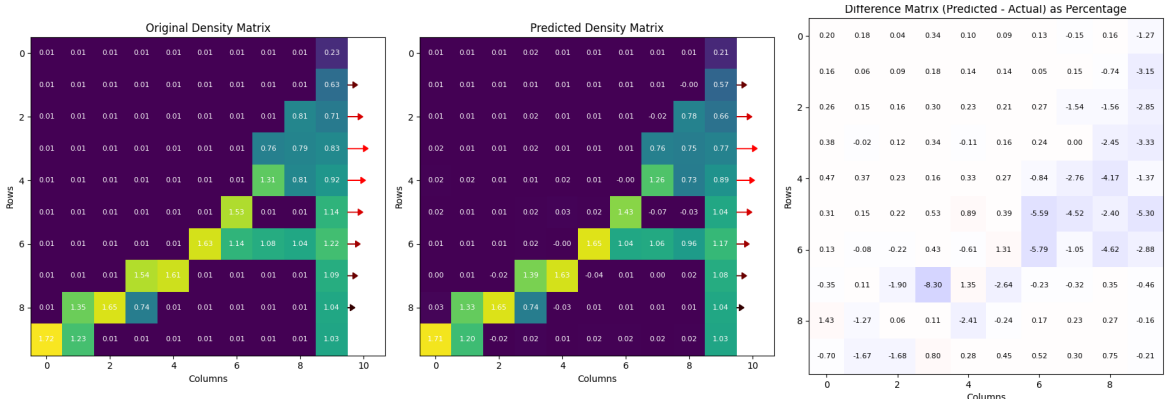


Figure 6-8: An illustrative sample of the RL surrogate model. The correct side and peak are identified, and the samples received an SSIM of 0.997 as bone morphology accurately matches the ground truth.

Although these results show the RL framework can predict triangular force profiles with relatively good accuracy, the underlying reason for the varying performance across variables is discussed in the next section.

6-6 Discussion and Limitations

The results show that the RL models significantly outperform the baseline SL model. This section reflects the results of the RL framework on the original design goal and identifies limitations of the approach. Moreover, future improvements are suggested that expand the model and could overcome current constraints.

6-6-1 Inverse Goal Reflection

As hypothesised, the RL model is better at solving an order-reduced variant of the inverse bone remodelling problem compared to the baseline SL model. However, with the available dataset, the model could not solve the original problem with the five different loading conditions. The performance difference between SL and RL (SSIM more than 2.5 times as high) does show RL is a serious contender for solving the problem.

The framework combined with the ensemble (SSIM = 0.73) outperformed the framework with the single surrogate (SSIM = 0.72) by a single percentage point. We hypothesise that the performance difference between the surrogate and ensemble models is not reflected in the performance of the RL model because the single loading condition problem is solved by both surrogate and ensemble sufficiently.

The RL framework accurately predicted the correct side due to the large morphological differences between left, right and top loading, which dominated the SSIM reward. In contrast, peak magnitude and exact location prediction performance likely suffered from the overrepresentation of this morphology in the reward, because more nuanced changes produced a similar reward.

Overall, these findings indicate that RL can find some inverse mapping between density and load, but the performance is still limited by the reward, computation power and surrogate performance.

6-6-2 Limitations

Although the RL framework demonstrates clear advantages, the model remains a proof-of-concept biomechanical inverse framework. The framework is limited in the problem formulations, reward metric, and computational efficiency.

First, the problem was reduced to triangular force profiles, and the action space was adapted to this simplification. Because the action space was tailored to triangular loads, the agent's MLP could not generalise to other force representations. This restricted the RL framework's ability to generalise to other force representations.

Second, the performance difference between the surrogate and ensemble versions of the framework demonstrates that it is highly sensitive to forward model error. This reliance on the forward estimation is a problem because, at higher resolutions, accurate forward estimation models do not exist. The original FEM simulation also influences the RL framework performance. For example, the inability to differentiate tension and compression is caused by the isotropic assumption in the forward simulation.

Furthermore, incorrect side placement is punished with such a large negative reward that the model has a bias towards the centre of the correct side. An exploration step on the edge of a side could cause it to go over the edge. Although this bias increases the average reward, the agent does not reach the true design goal, exact load placement. To solve this and similar problems, large reward discontinuities should be avoided.

The RL framework training is also around 32 times slower than SL based models, because PPO is on-policy and therefore has low sample efficiency. The RL framework is further slowed by the large number of forward passes included, with one per sample for the surrogate and five per sample for the ensemble. The low computational efficiency limits the scale at which the framework can be used, which decreases the relevance for realistic applications that use large meshes.

Combined, these limitations demonstrate that many conceptual and computational challenges remain before the RL framework can be applied to full-scale biomechanical inverse problems.

6-6-3 Future Improvements

These limitations could be mitigated in future iterations by adapting part of the framework and its training environment.

Future work could improve the model substantially by adapting the reward metric. The SSIM overrepresented coarse morphology in the reward, which hindered learning. A multi-objective reward with a smoother gradient (for example, by combining SSIM with MSE) could improve peak placement and increase magnitude estimation.

A more rigorous alternative is the extension to a multi-agent framework. Each agent focuses on a single design process, reducing the need for reward balancing. It also builds the groundwork for extension to multiple loading conditions, where each specialist agent is responsible for a single force type.

Curriculum learning is the process of slowly increasing the problem's complexity to guide agent learning. The created database contains a natural hierarchy of force profiles that, when ordered from simple to complex, could form the basis for a curriculum strategy.

These extensions could elevate the RL framework from a proof-of-concept to a rigorous inverse bone remodelling problem strategy.

Chapter 7

Concluding Remarks

The inverse bone remodelling problem is fundamentally challenging due to its ill-posedness and the non-uniqueness of load-density mappings. In this thesis, an Reinforcement Learning (RL) framework coupled with a surrogate model addresses this challenge by reframing the inverse task as a density matching problem instead of a direct load reconstruction. This formulation resulted in an RL framework that could learn a constrained single load type variant of the inverse problem, which gradient-based Supervised Learning (SL) models failed to learn stably. This highlights the suitability of RL in the exploration of the non-unique inverse bone remodelling domain. Beyond the bone remodelling context, this demonstrates a general strategy for solving ill-posed inverse problems through sequential decision making.

7-1 Integrated Discussion

The forward model and dataset design defined a force-density mapping that the Machine Learning (ML) models could learn. Although the dataset supported accurate forward learning, structural ambiguity in the inverse direction prevented the RL agent from learning any meaningful mapping. Reducing the load-type diversity removed this ambiguity, which enabled RL training. This revealed a fundamental difference between the forward and inverse problems: forward learning relies on representational coverage of the domain, while inverse learning requires identifiability. The behaviour of the models further reinforces this conclusion: the surrogate model failed to estimate the underrepresented complex loads, while the RL models failed to differentiate between compression and tension.

Furthermore, the ensemble RL framework outperformed the single surrogate RL framework even with an independent validation cycle. This suggests the accuracy of the forward estimation model does not just affect convergence speed, but directly influences the performance of the final model. This is expected because the reward domain used by the agent is directly influenced by the surrogate errors. This underscores the importance of a reliable and robust forward estimation model for the RL framework to converge towards the intended goal of

finding a matching load condition. This improvement in accuracy did increase computational demand, which shows the trade-off made between fidelity and efficiency.

Lastly, though the RL framework stabilised learning in an inverse problem with non-uniqueness in the mapping, it did not resolve the underlying ambiguity. The inability to differentiate between compression and tension demonstrates what happens when non-uniqueness occurs in the dataset. The RL agent converges towards one plausible loading condition, not necessarily the true loading condition. This is an important distinction, because the RL framework has no mechanism for recognising or representing alternative solutions.

7-2 Limitations and Generalisation

Although each chapter already described all key local limitations related to the forward model, database, forward estimation and the RL framework, certain limitations are not restricted to a single component, but overarch the entire pipeline.

First, the pipeline components all rely on the ones before them, resulting in a dependency chain. Dependency chains are dangerous, as a small initial error can lead to a large distortion in the final model. For example, a surrogate model error would distort the reward space for the RL agent. The agent is then trained on the distorted mapping, further reinforcing it through its exploration. This could lead to a framework that does not accurately estimate the inverse bone remodelling process.

Furthermore, the final dataset used to test the RL framework does not have non-uniqueness besides compression and tension similarity. This means that the non-uniqueness capabilities of the RL framework have not been demonstrated by this research. We hypothesise that the current framework will converge towards one of the solutions. The framework can thus not be used to find multiple solutions or even detect when there are multiple solutions. This is a downside because applications that require searching for and comparing multiple contender solutions cannot be performed using this RL framework.

Lastly, we were unable to demonstrate that this framework generalises to data not generated by our forward Finite Element Method (FEM) model, because we did not have data from another model with the same simulation constants.

Some of these constraints on the current result could be mitigated, as discussed below.

7-3 Future Work

The global limitations to the current approach are also the most fundamental to the framework, and thus difficult to overcome. This section proposes some strategies that could be used to avoid or at least reduce the effects of the above constraints.

A first possible extension is aligning the surrogate and the RL agent by using the standard deviation of the ensemble as an uncertainty estimate. Currently, this value is ignored by the agent. A possible extension could be including this uncertainty in the reward calculation. This could be implemented by punishing the agent for building force profiles that the surrogate model cannot estimate with high certainty. This agent-surrogate coupling would reduce

the dependency chain risk by rewarding the agent to take actions that align with surrogate capabilities.

Another possible focus could be quantifying non-uniqueness and identifying all load conditions instead of just one. A possible strategy for this could be a multi-agent framework where each agent is rewarded both for its surrogate density match as well as the uniqueness of its solution. Each agent then explores a different region of the domain, resulting in multiple load solutions.

Lastly, the current models are limited to estimating steady-state solutions. This assumption limits the possible loading conditions that could be predicted. Furthermore, real bone dynamics are never steady-state, as the human body is constantly adapting to load. This estimation also slowed down data generation, as each load condition has to be run until a steady state is reached. In a dynamic version of the RL framework, the forward step could be performed using the original FEM model.

These future extensions demonstrate that adaptations could mitigate some of the current limitations, which shows the method has high potential for further accuracy and fidelity improvements.

7-4 Conclusion

To conclude, this thesis demonstrated that an RL framework can explore the ill-posed design space in the context of the inverse bone remodelling problem. By combining a sequential decision-making process with a forward estimation surrogate model, the framework achieved an Structural Similarity Index Measure (SSIM) of 0.72 under conditions where a standard SL model did not achieve stable learning. The results show that the framework finds a plausible solution to the inverse bone remodelling problem, provided that the identifiability of the dataset remains large enough.

The central contribution of this work lies in reframing the inverse remodelling task as a sequential decision-making process, enabling the use of RL to navigate an inherently non-unique solution space, within the capabilities of the surrogate model. Unlike traditional approaches based on gradient descent optimisation, the RL-based formulation created a stable learning process that finds a consistent loading condition.

Looking past the inverse bone remodelling applications, the proposed pipeline illustrates how an RL agent and SL surrogate can be combined to address ill-posed inverse problems. As forward estimation fidelity and computational efficiency continue to improve, this framework presents a promising foundation for tackling a wider class of ill-posed inverse problems.

Bibliography

[1] J. D. Currey, "The many adaptations of bone," *Journal of Biomechanics*, vol. 36, no. 10, pp. 1487–1495, 2003. doi: 10.1016/S0021-9290(03)00124-6.

[2] J. Wolff, "The Internal Architecture of Normal Bone and Its Mathematical Significance," in *The Law of Bone Remodelling*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 3–22. doi: 10.1007/978-3-642-71031-5_2.

[3] H. M. Frost, "Bone's Mechanostat: A 2003 Update," *The Anatomical Record*, vol. 275, no. 2, pp. 1081–1101, 2003. doi: 10.1002/ar.a.10119.

[4] S. C. Cowin, "Mechanical Modeling of the Stress Adaptation Process in Bone," *Calcified Tissue International*, vol. 36, pp. S98–S103, 1984. doi: 10.1007/BF02406141.

[5] S. Bachmann, D. H. Pahr, and A. Synek, "Hip joint load prediction using inverse bone remodeling with homogenized FE models: Comparison to micro-FE and influence of material modeling strategy," *Computer Methods and Programs in Biomedicine*, vol. 236, 2023. doi: 10.1016/j.cmpb.2023.107549.

[6] H. Wankerl, M. L. Stern, A. Mahdavi, C. Eichler, and E. W. Lang, "Parameterized reinforcement learning for optical system optimization," *Journal of Physics D: Applied Physics*, vol. 54, no. 30, 2021. doi: 10.1088/1361-6463/abfddb.

[7] H. Yan, R. Hao, B. Ye, and S. Jin, "Exploring high-performance photonic crystal slow light waveguides through deep reinforcement learning," *Optics Communications*, vol. 569, 2024. doi: 10.1016/j.optcom.2024.130830.

[8] H. Weinans, R. Huiskes, and H. J. Grootenboer, "The behavior of adaptive bone-remodelling simulation models," *Journal of Biomechanics*, vol. 25, no. 12, pp. 1425–1441, 1992. doi: 10.1016/0021-9290(92)90056-7.

[9] A. A. Zadpoor, "Open forward and inverse problems in theoretical modeling of bone tissue adaptation," *Journal of the Mechanical Behavior of Biomedical Materials*, vol. 27, pp. 249–261, 2013. doi: 10.1016/j.jmbbm.2013.05.017.

[10] K. J. Fischer, C. R. Jacobs, M. E. Levenston, D. D. Cody, and D. R. Carter, "Bone load estimation for the proximal femur using single energy quantitative CT data," *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 1, no. 3, pp. 233–245, 1998. doi: 10.1080/01495739808936704 .

[11] A. A. Zadpoor, G. Campoli, and H. Weinans, "Neural network prediction of load from the morphology of trabecular bone," *Applied Mathematical Modelling*, vol. 37, no. 7, pp. 5260–5276, 2013. doi: 10.1016/j.apm.2012.10.049 .

[12] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv*, 7 2017.

[13] A. C. Rapisarda, A. Della Corte, R. Drobnicki, F. Di Cosmo, and L. Rosa, "A model for bone mechanics and remodeling including cell populations dynamics," *Zeitschrift fur Angewandte Mathematik und Physik*, vol. 70, no. 1, 2019. doi: 10.1007/s00033-018-1055-1 .

[14] A. Marny Mohammed, "An overview of bone cells and their regulating factors of differentiation," *Malaysian Journal of Medical Sciences*, 2008.

[15] R. Curley, "Bone Remodeling," 2010.

[16] P. Christen, K. Ito, R. Ellouz, S. Boutroy, E. Sornay-Rendu, R. D. Chapurlat, and B. Van Rietbergen, "Bone remodelling in humans is load-driven but not lazy," *Nature Communications*, vol. 5, 2014. doi: 10.1038/ncomms5855 .

[17] J. Martínez-Reina, J. L. Calvo-Gallego, and P. Pivonka, "Combined Effects of Exercise and Denosumab Treatment on Local Failure in Post-menopausal Osteoporosis–Insights from Bone Remodelling Simulations Accounting for Mineralisation and Damage," *Frontiers in Bioengineering and Biotechnology*, vol. 9, 2021. doi: 10.3389/fbioe.2021.635056 .

[18] K. Haase and G. Rouhi, "Prediction of stress shielding around an orthopedic screw: Using stress and strain energy density as mechanical stimuli," *Computers in Biology and Medicine*, vol. 43, no. 11, pp. 1748–1757, 2013. doi: 10.1016/j.compbiomed.2013.07.032 .

[19] C. V. B. Gusmão and W. D. Belangero, "How do bone cells sense mechanical loading?" *Revista Brasileira de Ortopedia (English Edition)*, vol. 44, no. 4, pp. 299–305, 2009. doi: 10.1016/s2255-4971(15)30157-9 .

[20] A. Vahdati and G. Rouhi, "A model for mechanical adaptation of trabecular bone incorporating cellular accommodation and effects of microdamage and disuse," *Mechanics Research Communications*, vol. 36, no. 3, pp. 284–293, 2009. doi: 10.1016/j.mechrescom.2008.10.004 .

[21] P. Lips and N. M. van Schoor, "Quality of life in patients with osteoporosis." *Osteoporosis International*, vol. 16, no. 5, pp. 447–455, 2005. doi: 10.1007/s00198-004-1762-7 .

[22] A. E. Smit, O. C. Meijer, and E. M. Winter, "The multi-faceted nature of age-associated osteoporosis," *Bone Reports*, vol. 20, 2024. doi: 10.1016/j.bonr.2024.101750 .

[23] J. D. Sibonga, E. R. Spector, G. Yardley, J. S. Alwood, J. Myers, H. Evans, S. A. Smith, and L. King, "Risk of Bone Fracture due to Spaceflight-induced Changes to Bone," NASA, Tech. Rep., 2024.

[24] A. M. ElDeeb and A. A. Abdel-Aziem, "Effect of Whole-Body Vibration Exercise on Power Profile and Bone Mineral Density in Postmenopausal Women With Osteoporosis: A Randomized Controlled Trial," *Journal of Manipulative and Physiological Therapeutics*, vol. 43, no. 4, pp. 384–393, 2020. doi: 10.1016/j.jmpt.2019.12.003.

[25] R. Huiskes, H. Weinans, H. J. Grootenboer, M. Dalstra, B. Fudala, and T. J. Slooff, "Adaptive bone-remodelling theory applied to prosthetic-design analysis," *Journal of Biomechanics*, vol. 20, no. 11, pp. 1135–1150, 1987. doi: 10.1016/0021-9290(87)90030-3.

[26] S. C. Cowin, *Bone mechanics handbook*, 2001.

[27] S. Nobakhti and S. J. Shefelbine, "On the Relation of Bone Mineral Density and the Elastic Modulus in Healthy and Pathologic Bone," *Current Osteoporosis Reports*, vol. 16, no. 4, pp. 404–410, 2018. doi: 10.1007/s11914-018-0449-5.

[28] J. Y. Rho, R. B. Ashman, and C. H. Turner, "Young's modulus of trabecular and cortical bone material: ultrasonic and microtensile measurements," *Journal of Biomechanics*, vol. 26, no. 2, pp. 11–12, 1993. doi: 10.1016/0021-9290(93)90042-d.

[29] C. T. Rubin and L. E. Lanyon, "Limb Mechanics as A Function of Speed and Gait: A Study of Functional Strains in the Radius and Tibia of Horse and Dog," *Journal of Experimental Biology*, vol. 101, no. 1, 1982. doi: 10.1242/jeb.101.1.187.

[30] H. Cezayirlioglu, E. Bahniuk, D. Davy, and K. Heiple, "Anisotropic yield behavior of bone under combined axial force and torque," *Journal of Biomechanics*, vol. 18, no. 1, pp. 61–69, 1985. doi: 10.1016/0021-9290(85)90045-4.

[31] L. F. Bonewald, "The amazing osteocyte," *Journal of Bone and Mineral Research*, vol. 26, no. 2, pp. 229–238, 2011. doi: 10.1002/jbmr.320.

[32] K. Trachoo, I. Chaiya, and D. Prathumwan, "An improved mathematical modeling of bone remodeling: the role of stability in predicting bone health," *Advances in Continuous and Discrete Models*, vol. 2025, no. 1, 2025. doi: 10.1186/s13662-025-03934-8.

[33] J. Du, S. Li, and V. V. Silberschmidt, "Trabecular bone remodelling: Finite-element simulation," in *Procedia Structural Integrity*, vol. 28. Elsevier B.V., 2020, pp. 577–583. doi: 10.1016/j.prostr.2020.10.067.

[34] K. Fischer, C. Jacobs, M. Levenston, and D. Carter, "Different loads can produce similar bone density distributions," *Bone*, vol. 19, no. 2, pp. 127–135, 1996. doi: 10.1016/8756-3282(96)00140-8.

[35] P. Christen, B. Van Rietbergen, F. M. Lambers, R. Müller, and K. Ito, "Bone morphology allows estimation of loading history in a murine model of bone adaptation," *Biomechanics and Modeling in Mechanobiology*, vol. 11, no. 4, pp. 483–492, 2012. doi: 10.1007/s10237-011-0327-x.

[36] A. Synek, C. J. Dunmore, T. L. Kivell, M. M. Skinner, and D. H. Pahr, "Inverse remodelling algorithm identifies habitual manual activities of primates based on metacarpal bone architecture," *Biomechanics and Modeling in Mechanobiology*, vol. 18, no. 2, pp. 399–410, 2019. doi: 10.1007/s10237-018-1091-y .

[37] R. Ghosh, S. Chanda, and D. Chakraborty, "Qualitative predictions of bone growth over optimally designed macro-textured implant surfaces obtained using NN-GA based machine learning framework," *Medical Engineering and Physics*, vol. 95, pp. 64–75, 2021. doi: 10.1016/j.medengphy.2021.08.002 .

[38] A. Pais, J. L. Alves, and J. Belinha, "A neural network to surrogate computational bone remodelling in the calcaneus," *Knowledge-Based Systems*, vol. 330, 2025. doi: 10.1016/j.knosys.2025.114445 .

[39] G. Campoli, H. Weinans, and A. A. Zadpoor, "Computational load estimation of the femur," *Journal of the Mechanical Behavior of Biomedical Materials*, vol. 10, pp. 108–119, 2012. doi: 10.1016/j.jmbbm.2012.02.011 .

[40] N. Garijo, N. Verdonschot, K. Engelborghs, J. M. García-Aznar, and M. A. Pérez, "Subject-specific musculoskeletal loading of the tibia: Computational load estimation," *Journal of the Mechanical Behavior of Biomedical Materials*, vol. 65, pp. 334–343, 2017. doi: 10.1016/j.jmbbm.2016.08.026 .

[41] T. Shah, L. Zhuo, P. Lai, A. De La Rosa-Moreno, F. Amirkulova, and P. Gerstoft, "Reinforcement learning applied to metamaterial design," *The Journal of the Acoustical Society of America*, vol. 150, no. 1, pp. 321–338, 7 2021. doi: 10.1121/10.0005545 .

[42] A. E. Gongora, S. Mysore, B. Li, W. Shou, W. Matusik, E. F. Morgan, K. A. Brown, and E. Whiting, "Designing composites with target effective young s modulus using reinforcement learning," in *Proceedings - SCF 2021: ACM Symposium on Computational Fabrication*. Association for Computing Machinery, Inc, 10 2021. doi: 10.1145/3485114.3485123 .

[43] S. Shonkwiler, X. Li, R. Fenrich, and S. McMains, "Deep reinforcement learning for stacking sequence optimization of composite laminates," *Manufacturing Letters*, vol. 35, pp. 1203–1213, 2023. doi: 10.1016/j.mfglet.2023.08.133 .

[44] Y. D. Bansod, M. Kebbach, D. Kluess, R. Bader, and U. van Rienen, "Computational Analysis of Bone Remodeling in the Proximal Tibia Under Electrical Stimulation Considering the Piezoelectric Properties," *Frontiers in Bioengineering and Biotechnology*, vol. 9, 2021. doi: 10.3389/fbioe.2021.705199 .

[45] S. Gupta and A. Gupta, "Dealing with noise problem in machine learning data-sets: A systematic review," in *Procedia Computer Science*, vol. 161. Elsevier B.V., 2019, pp. 466–474. doi: 10.1016/j.procs.2019.11.146 .

[46] K. Su, L. Yuan, J. Yang, and J. Du, "Numerical Simulation of Mandible Bone Remodeling under Tooth Loading: A Parametric Study," *Scientific Reports*, vol. 9, no. 1, 12 2019. doi: 10.1038/s41598-019-51429-w .

[47] G. E. Fahy, C. Deter, R. Pitfield, J. J. Miszkiewicz, and P. Mahoney, "Bone deep: Variation in stable isotope ratios and histomorphometric measurements of bone remodelling within adult humans," *Journal of Archaeological Science*, vol. 87, pp. 10–16, 11 2017. doi: 10.1016/j.jas.2017.09.009 .

[48] R. C. Martin, *Agile software development, principles, patterns, and practices.* Pearson, 2014.

[49] C. Milgrom, A. S. Finestone, and A. Voloshin, "Differences in the principal strain angles during activities performed on natural hilly terrain versus engineered surfaces," *Clinical Biomechanics*, vol. 80, 12 2020. doi: 10.1016/j.clinbiomech.2020.105146 .

[50] S. Dendorfer, H. J. Maier, and J. Hammer, "Fatigue damage in cancellous bone: An experimental approach from continuum to micro scale," *Journal of the Mechanical Behavior of Biomedical Materials*, vol. 2, no. 1, pp. 113–119, 2009. doi: 10.1016/j.jmbbm.2008.03.003 .

[51] S. Bachmann, D. H. Pahr, and A. Synek, "A Density-Dependent Target Stimulus for Inverse Bone (Re)modeling with Homogenized Finite Element Models," *Annals of Biomedical Engineering*, vol. 51, no. 5, pp. 925–937, 2023. doi: 10.1007/s10439-022-03104-x .

[52] S. Mohammed, L. Budach, M. Feuerpfeil, N. Ihde, A. Nathansen, N. Noack, H. Patzlaff, F. Naumann, and H. Harmouch, "The effects of data quality on machine learning performance on tabular data," *Information Systems*, vol. 132, 2025. doi: 10.1016/j.is.2025.102549 .

[53] A. A. Zadpoor, A. Asadi Nikooyan, and A. Reza Arshi, "A model-based parametric study of impact force during running," *Journal of Biomechanics*, vol. 40, no. 9, pp. 2012–2021, 2007. doi: 10.1016/j.jbiomech.2006.09.016 .

[54] R. C. Hibbeler and J. H. Lee, *Mechanics of materials.* Pearson Education Limited, 2023.

[55] C. M. Bishop, *Pattern recognition and machine learning.* Springer Science + Business Media, 2009.

[56] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 4 2004. doi: 10.1109/TIP.2003.819861 .

Glossary

List of Acronyms

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CPU	Central Processing Unit
FEM	Finite Element Method
GPU	Graphical Processing Unit
MDP	Markov Decision Process
ML	Machine Learning
MLP	Multi Layer Perceptron
MSE	Mean Squared Error
PPO	Proximal Policy Optimisation
RL	Reinforcement Learning
SED	Strain Energy Density
SL	Supervised Learning
SSIM	Structural Similarity Index Measure

List of Symbols

$_0$	Initial
$_{max}$	Maximum
$_{min}$	Minimum
$_{obs}$	Observed
$_{sim}$	Simulated
$_{tol}$	Tolerance
$_t$	Time
^{n}	Exponential constant
$\hat{P}$	Estimate of P
λ	First Lamé Parameter
$\mathbf{R}$	Real Space
$\mathcal{E}$	Triangular Element Space
$\mathcal{L}$	Loss Function
μ	Shear Modulus
ν	Poisson's Ratio
ρ	Density
σ	Stress Tensor
ε	Strain Tensor
A	Action Space
a	Weak Stiffness Form or Action Component
B	Remodelling Speed Constant
E	Young's Modulus or Energy
F	Force
I	Identity Matrix
k	Minimal Stimulus Threshold
L	Load Form
N	Total Number of Elements
p	Propability
S	State Space
U	Displacement Tensor

Appendix A

Forward Model Constants

This appendix contains a comprehensive table listing all constants used by the forward model, along with a corresponding reference to the justification. Constants were based on the original Weinans model, the parameter sweep, or a tolerance hierarchy. The tolerance hierarchy was defined using non-overlapping orders of magnitude, with a lower limit determined by the machine precision (64 bit $\approx 5 \times 10^{-16}$).

Table A-1: Overview of constants used in the forward bone adaptation model. Justification is based on Weinans [8], the parameter sweep [S] or the tolerance hierarchy [T].

Name	Symbol	Value	Unit	Ref.
Time increment	Δt	1.0	days	[8]
Poisson ratio	ν	0.3	-	[8]
Elastic modulus scale	E_0	3790	$\text{MPa}(\text{g/cm}^3)^{-n}$	[8]
Modulus exponent	n	3.0	-	[8]
Remodelling rate coefficient	B	1.0	$(\text{g} \cdot \text{cm}^{-3})^2/(\text{MPa} \cdot \text{day})$	[8]
Stimulus threshold	k	0.25	-	[8]
Initial density	ρ_0	0.87	g/cm^3	[8]
Minimum density	$\rho_{\min}$	0.010	g/cm^3	[8]
Maximum density	$\rho_{\max}$	1.740	g/cm^3	[8]
Time steps	T	250	days	[S]
Density tolerance decay	κ	1.06	-	[S]
Convergence after steps	$C_{\text{tol},0}$	10	steps	[S]
Convergence steps decay	γ	1	-	[S]
Density tolerance	ϵ_ρ	1×10^{-9}	g/cm^3	[T]
Krylov solver tolerance	ϵ_κ	1×10^{-10}	g/cm^3	[T]
Boundary tolerance	ϵ_B	1×10^{-14}	g/cm^3	[T]

Appendix B

Data Storage

This appendix contains the .json storage format used to save the large dataset of force profile density pairs, which is used by both the neural network and the reinforcement learning agent. It has been moved to the next page for formatting reasons.

STORAGE JSON: This shows the way samples are stored inside the JSON. First with an example, then with a pseudocode implementation.

```
[{"serial_number": 1,
  "force_profile": [
    [0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00],
    [0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00],
    [0.00,0.00,0.00,0.00,0.00,0.00,0.13,1.61,7.20,11.87]
  ],
  "final_output_density": [
    [1.63,1.53,0.88,0.01,0.01,0.01,0.01,0.01,0.01,0.01],
    [1.66,1.74,1.74,1.74,1.31,0.48,0.01,0.01,0.01,0.01],
    [1.07,0.96,0.71,1.23,1.71,1.74,0.88,0.01,0.01,0.01],
    [1.14,0.83,0.64,0.68,0.86,1.35,1.74,0.88,0.01,0.01],
    [1.33,0.70,0.67,0.78,0.82,0.80,1.24,1.74,0.81,0.01],
    [1.49,0.67,0.81,0.92,0.86,0.79,0.80,1.44,1.74,0.01],
    [1.61,0.78,1.05,1.06,0.94,0.83,0.70,0.80,1.73,0.88],
    [1.68,1.14,1.32,1.13,0.91,0.66,0.51,0.63,1.61,1.66],
    [1.74,1.70,1.60,1.49,1.47,1.47,1.40,1.29,1.42,1.74],
    [1.74,1.74,1.74,1.32,0.98,0.77,0.01,0.01,0.93,1.74]
  ]
},{
  "serial_number": "unique number for the datapoint",
  "force_profile": [
    ["top row forces: positive is up, first element is left"],
    ["right row forces: positive is to the right, first element is
       down"],
    ["left row forces: positive is to the right, first element is down
       "]
  ],
  "final_output_density": [
    ["matrix with densities, first element is top left corner"],
    [..., ..., ...],
    ["last element is bottom right corner"]
  ]}]
```

Appendix C

Forward Model Validation

This appendix contains the validation figures for the adapted forward model from chapter 3.

C-1 Comparison to Model of Weinans et al.

The adapted model was compared to the results of Weinans et al. [8] for four different grid resolutions: 5×5, 10×10, 20×20, and 40×40. The corresponding validation figures for the first three resolutions are shown in Figure C-1, C-2 and C-3.

C-2 Comparison to the Models of Weinans and Bansod et al.

At the highest resolution of (40×40), the baseline implementation by Bansod et al. was also added, resulting in Figure C-4 [44].

C-3 Constraint and Load Validation

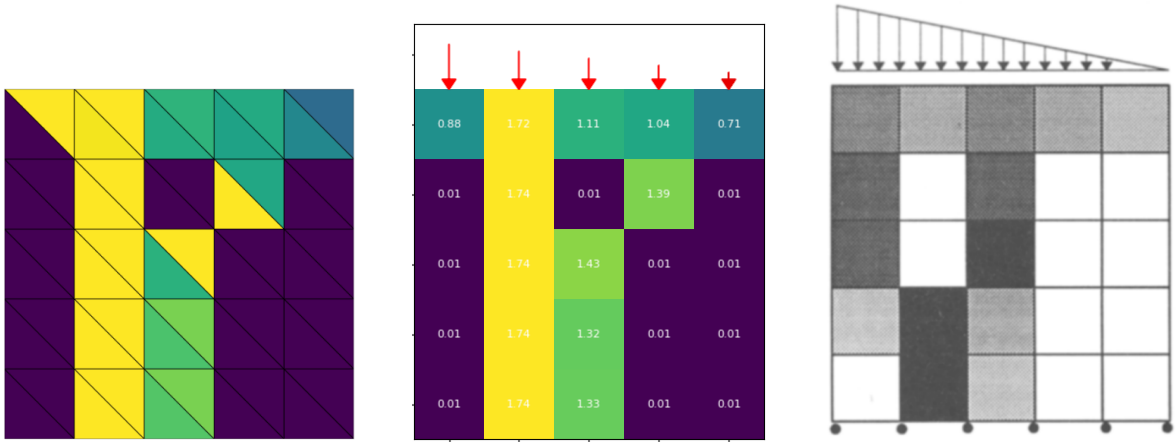
To test the constraints of the model and the modified load inputs, all sides are tested in Figure C-5.

C-4 Parameter Sensitivity Validation

The way the simulations adapt to different constants is tested in Figure C-6.

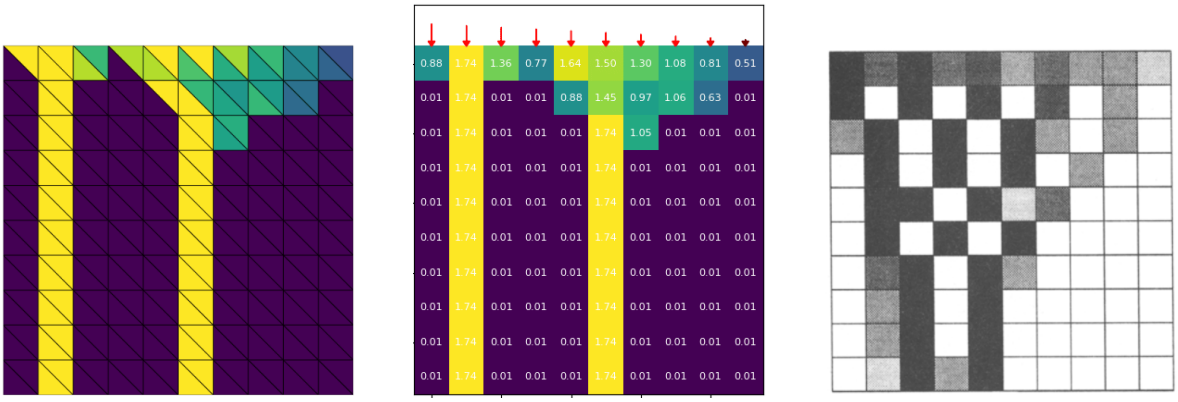
C-5 Complex Loads Validation

Complicated load conditions are tested to demonstrate that the model converges and is reliable, as shown in Figure C-7.



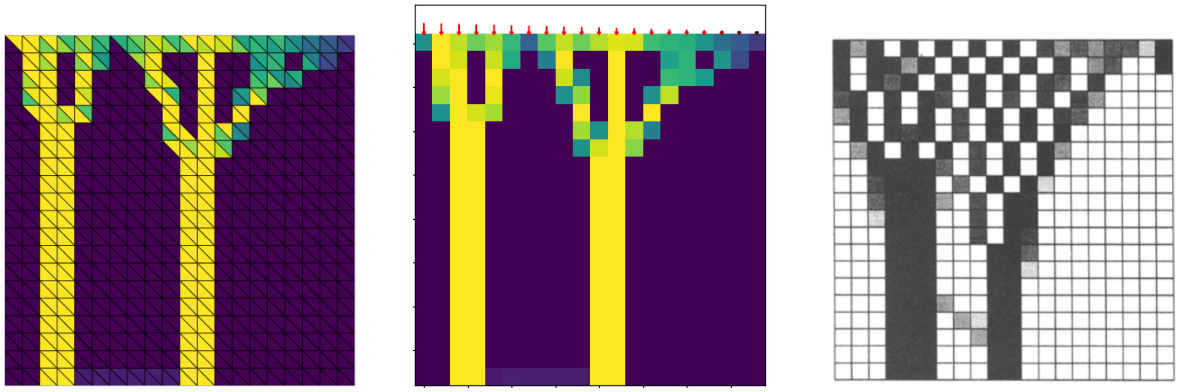
(a) Pyvista **(b)** Matrix **(c)** Weinans et al.

Figure C-1: This figure shows the validation figure used for the 5×5 simulation.



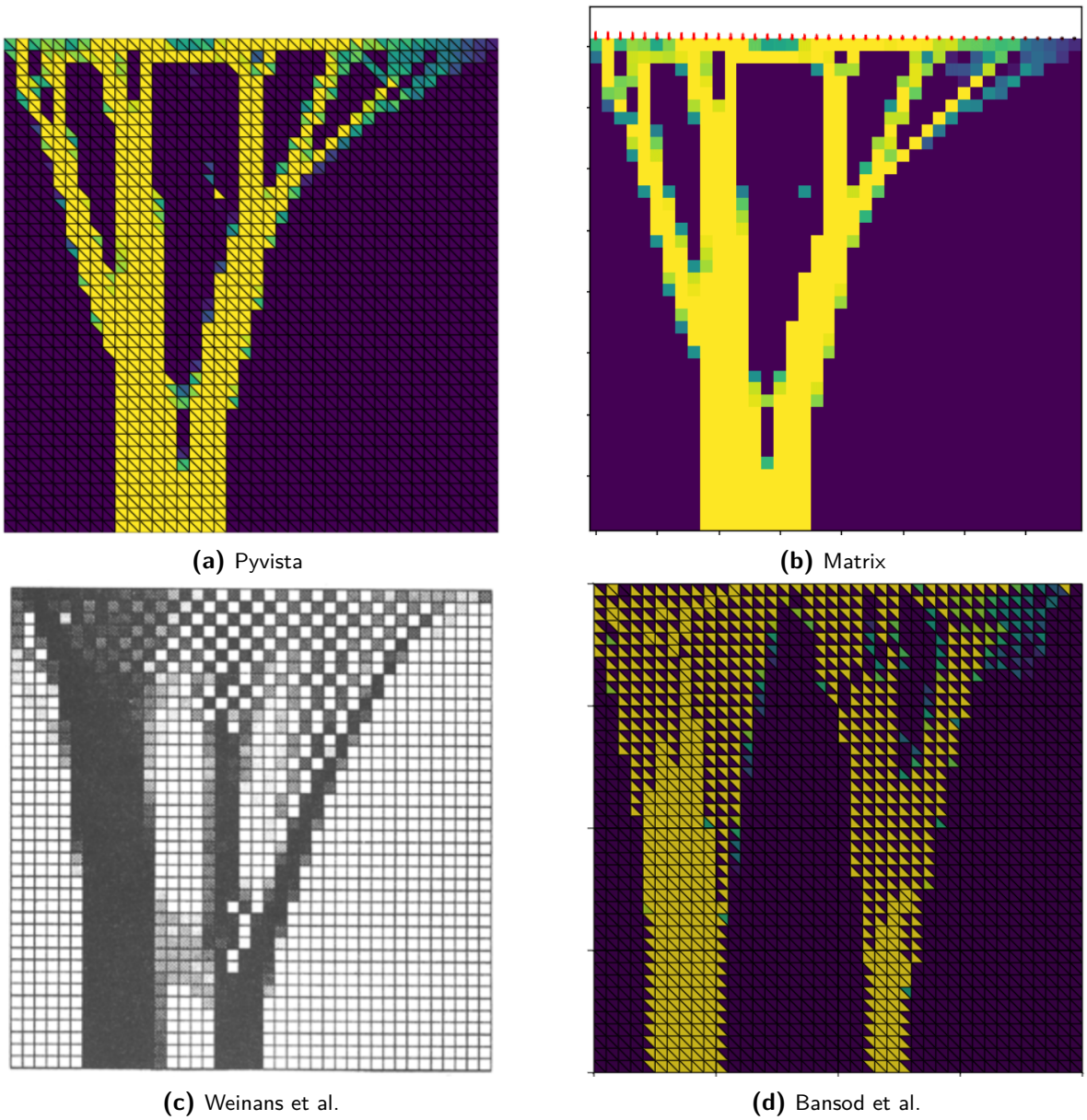
(a) Pyvista **(b)** Matrix **(c)** Weinans et al.

Figure C-2: This figure shows the validation figure used for the 10×10 simulation.



(a) Pyvista **(b)** Matrix **(c)** Weinans et al.

Figure C-3: This figure shows the validation figure used for the 20×20 simulation.



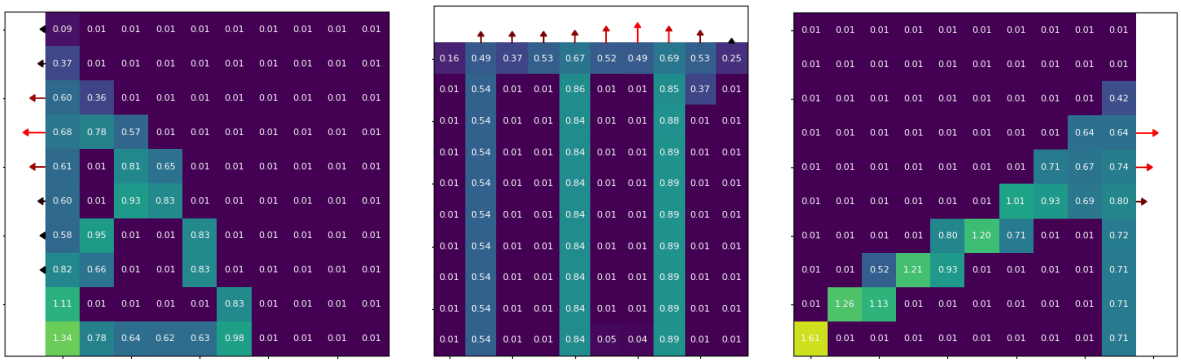
(a) Pyvista

(b) Matrix

(c) Weinans et al.

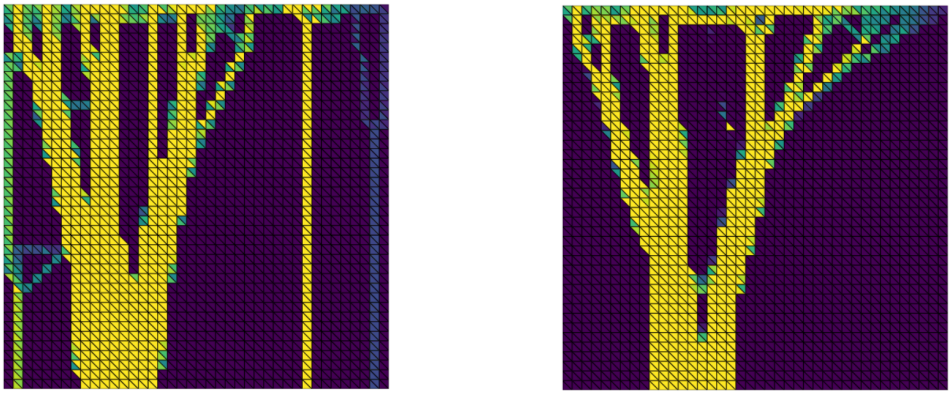
(d) Bansod et al.

Figure C-4: This figure shows the adapted, the Weinans et al. and Bansod et al. model for the 40×40 simulation.



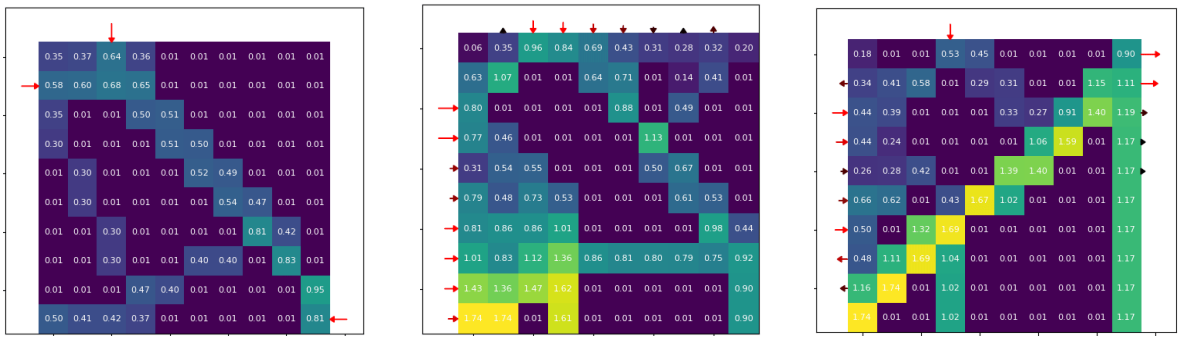
(a) Left **(b)** Top **(c)** Right

Figure C-5: This figure shows the validation figure used for testing the load profiles and constraints.



(a) $n = 2$ **(b)** $n = 3$

Figure C-6: This figure shows the significant impact the tissue remodelling exponent has on the morphology of the bone.



(a) Simple **(b)** Middle **(c)** Complex

Figure C-7: This figure shows the validation used for testing more complex load profiles.

Appendix D

Hardware and Software specifications

All experiments and simulations were performed on a single workstation. All specifications of this workstation are shown in Table D-1. The full source code used to produce the results presented in this thesis is publicly available in a version-controlled repository. The repository contains all simulation scripts, surrogate and ensemble model implementations, reinforcement learning agents, and evaluation pipelines required to reproduce the presented results:

https://github.com/gijsschlief/bone_remodeling

Table D-1: Hardware and software configuration used for all experiments.

Component	**Specification**
Machine	HP ZBOOK Studio x360 G5
CPU	Intel Core i7-8750H (6 cores / 12 threads, 2.20–4.10 GHz)
GPU	NVIDIA Quadro P1000 (4 GB GDDR5, 640 CUDA cores)
CUDA Toolkit	CUDA 12.9 (NVIDIA driver 575.64.03)
RAM	16 GB DDR4 @ 2667 MT/s
Operating System	Ubuntu 24.04.2 LTS
Python Environment	Python 3.11.13, NumPy 2.3.2, PyTorch 2.7.1
FEM Backend	FEniCS 2019.1.0 (PETSc solvers)
Software Tools	Visual Studio Code 1.104.1, Git 2.43.0
Compute Precision	FP64 (double precision)

Appendix E

Surrogate Validation

This appendix contains the validation figures for the surrogate model introduced in chapter 5. An illustrative sample from each load condition has been included in Table E-1.

Table E-1: Overview of surrogate model validation figures by applied load condition. Each figure shows an illustrative surrogate prediction compared to the FEM reference. A difference plot is added to exemplify where they diverge. Note that SSIM can become negative for strongly correlated densities as is the case in the worst prediction.

Load Class	**SSIM**	**Sample**	**Figure (Description)**
Impulse	0.98	8169	Figure E-1 – Concentrated single-point force.
Square	0.83	11029	Figure E-2 – Uniform pressure distribution.
Gaussian	0.85	3604	Figure E-3 – Smooth bell-shaped load profile.
Ramp	0.75	11134	Figure E-4 – Ramp pressure distribution.
Double	0.67	2387	Figure E-5 – Load profile with two distributions.
Triple	0.49	11382	Figure E-6 – Three profile distribution.
Multi	0.60	7719	Figure E-7 – Complex loading combination.
Multi	0.73	13327	Figure E-8 – Complex loading combination.
Worst	-0.086	1024	Figure E-9 – Worst prediction from surrogate.

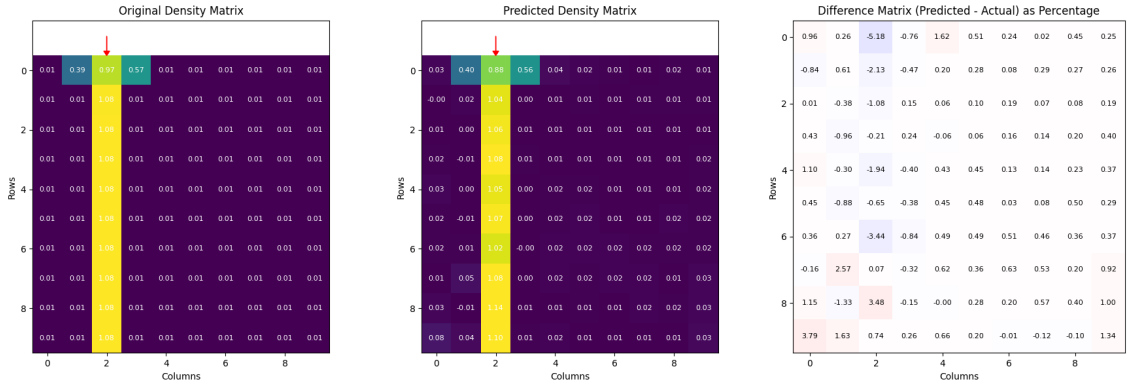


Figure E-1: An impulse loading placed at the top side. The surrogate model is able to predict the exact morphology and errors are minimal.

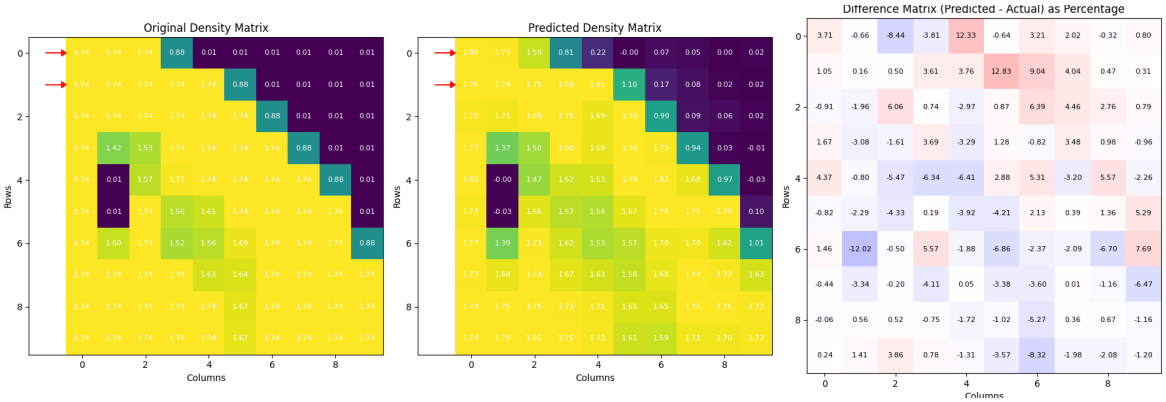


Figure E-2: A square profile of two forces is loaded on the top left side. The surrogate model is able to predict morphology and density with small accuracies in the magnitude.

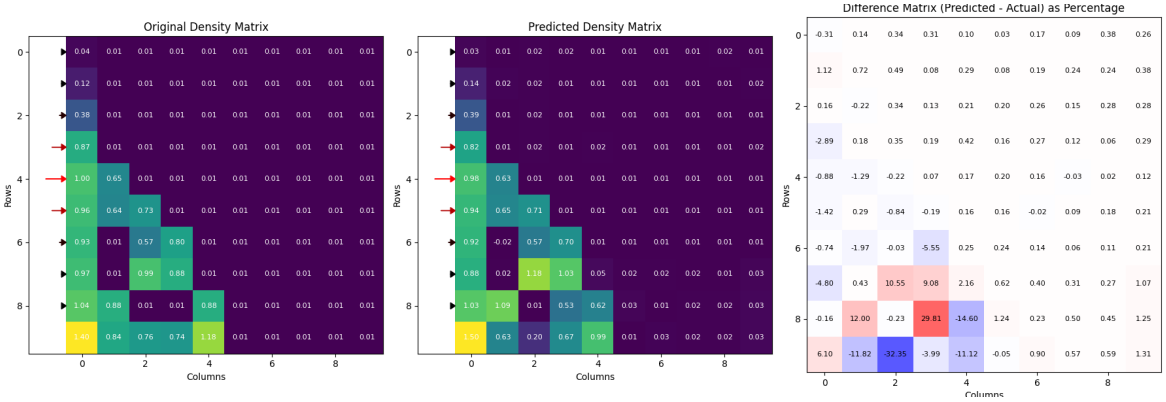


Figure E-3: A gaussian load placed on the left side. The surrogate model is able to predict the morphology, but does not predict all elements exactly in magnitude.

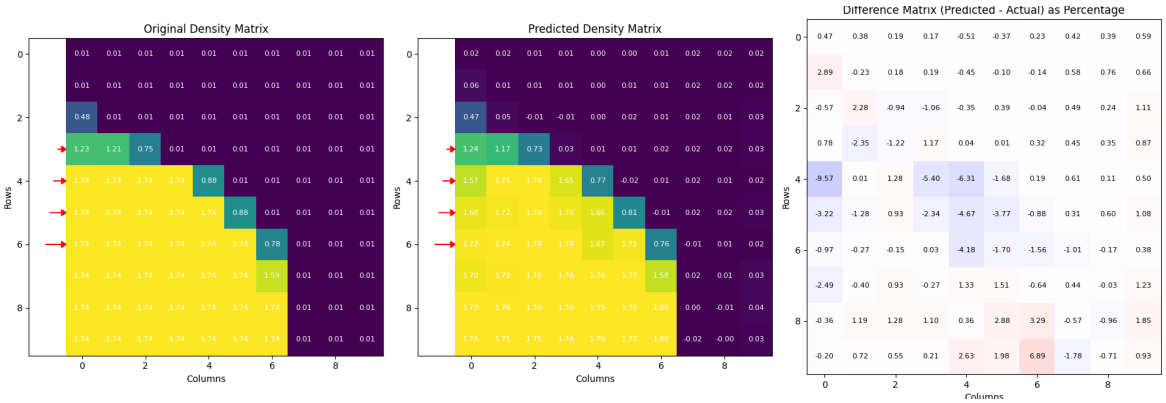


Figure E-4: A single ramp loading condition placed on the left side. The surrogate model is able to accuratly predict density magnitude and bone morphology.

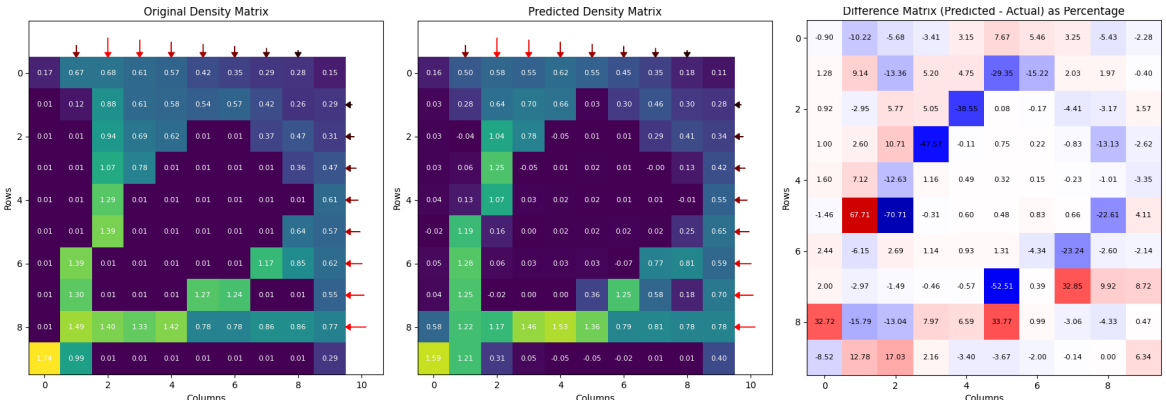


Figure E-5: The top and right side both contain a triangular profile. The surrogate model is able to find the correct morphology but has some inaccurate densities inside the mesh.

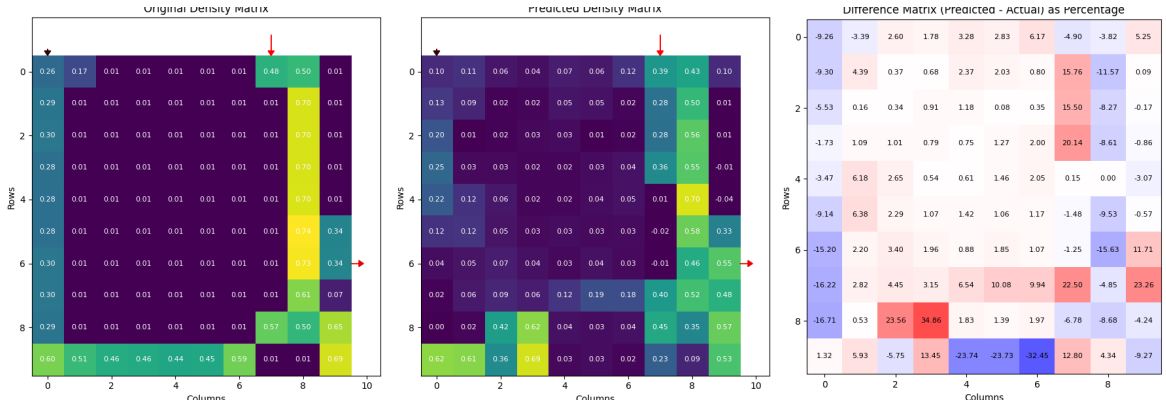


Figure E-6: Three loads placed on the simulation. The surrogate model struggles to find the correct morphology and performance starts to break down.

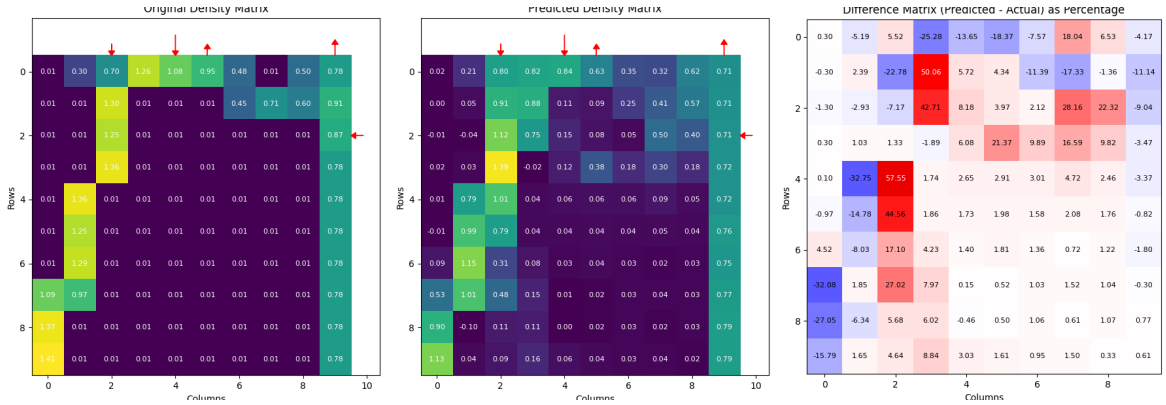


Figure E-7: Simple multi load example of the surrogate model. The surrogate model is able to predict some morphology but fails to find the exact structure and some regions contain large errors.

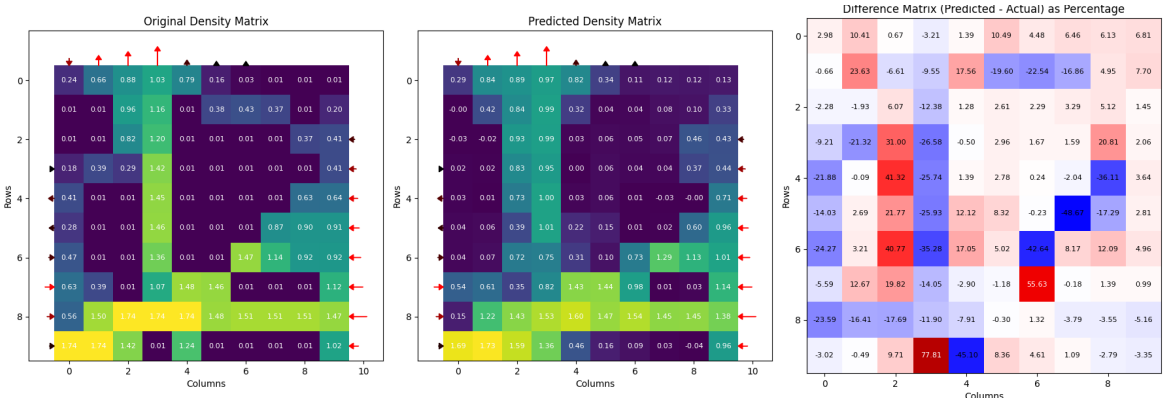


Figure E-8: Multi loading example with different loading types on all three sides. The surrogate model is able to find some structure but unable to find the finer structural details.

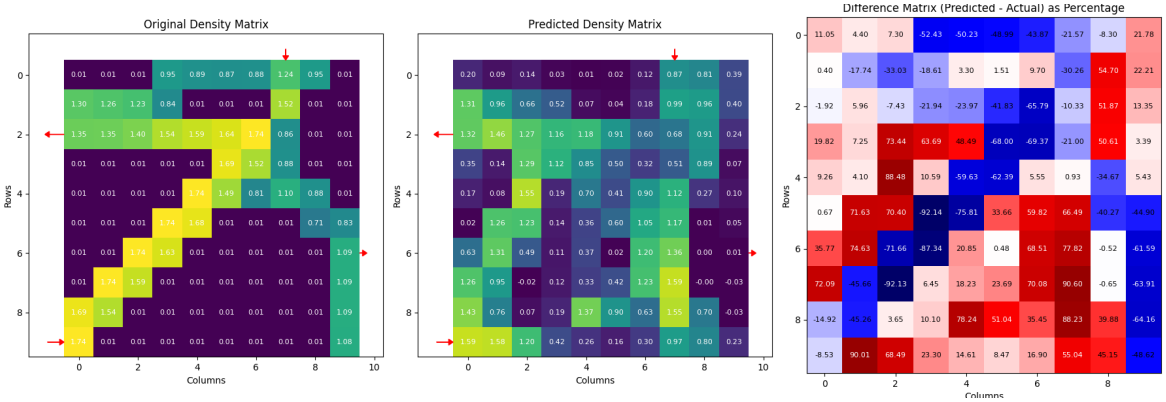


Figure E-9: Worst Structural Similarity Index Measure (SSIM) score from the surrogate model on the test set. The surrogate model is unable to find the correct bone morphology.

Appendix F

Ensemble Validation

This appendix contains the validation figures for the ensemble model introduced in chapter 5. A range of illustrative samples have been included with increasing complexity to demonstrate where performance breaks down Table F-1.

Table F-1: Overview of ensemble model validation figures by applied load condition. Each figure shows an illustrative ensemble prediction (top left) compared to the FEM reference (top right). A difference plot is added to exemplify where they diverge (bottom left). The standard deviation of the models is used to identify the certainty of the ensemble and is shown in the last subfigure (bottom right).

Load Class	**SSIM**	**Sample**	**Figure (Description)**
Impulse	0.69	8052	Figure F-1 – Concentrated single-point force.
Square	0.89	1301	Figure F-2 – Uniform pressure distribution.
Gaussian	0.97	9399	Figure F-3 – Smooth bell-shaped load profile.
Ramp	0.99	12934	Figure F-4 – Ramp pressure distribution.
Double	0.53	8907	Figure F-5 – Load profile with two distributions.
Triple	0.46	13126	Figure F-6 – Three profile distribution.
Multi	0.45	1061	Figure F-7 – Complex loading combination.

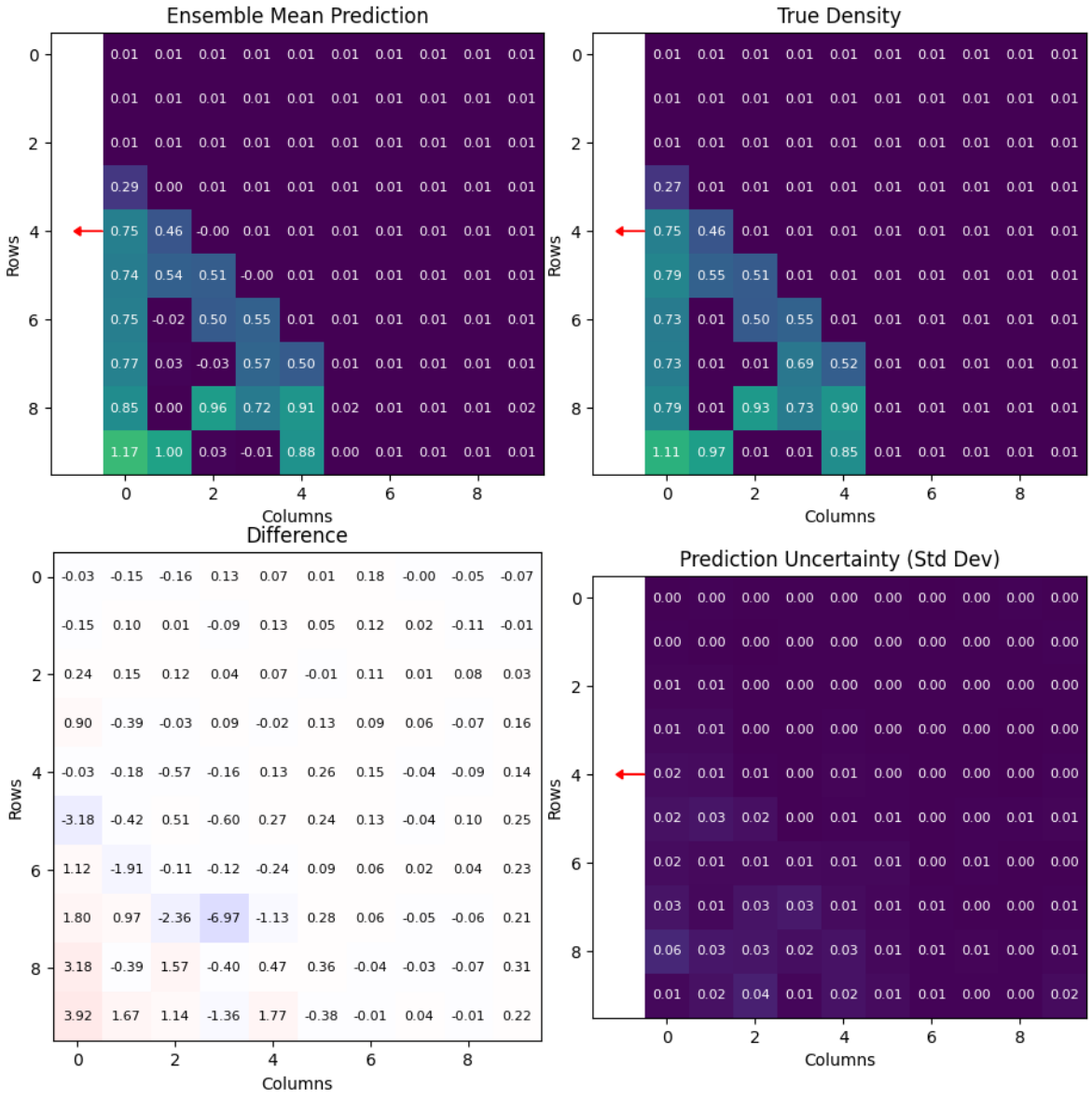


Figure F-1: Impulse load placed on the left side. Ensemble model can accurately represent this loading family.

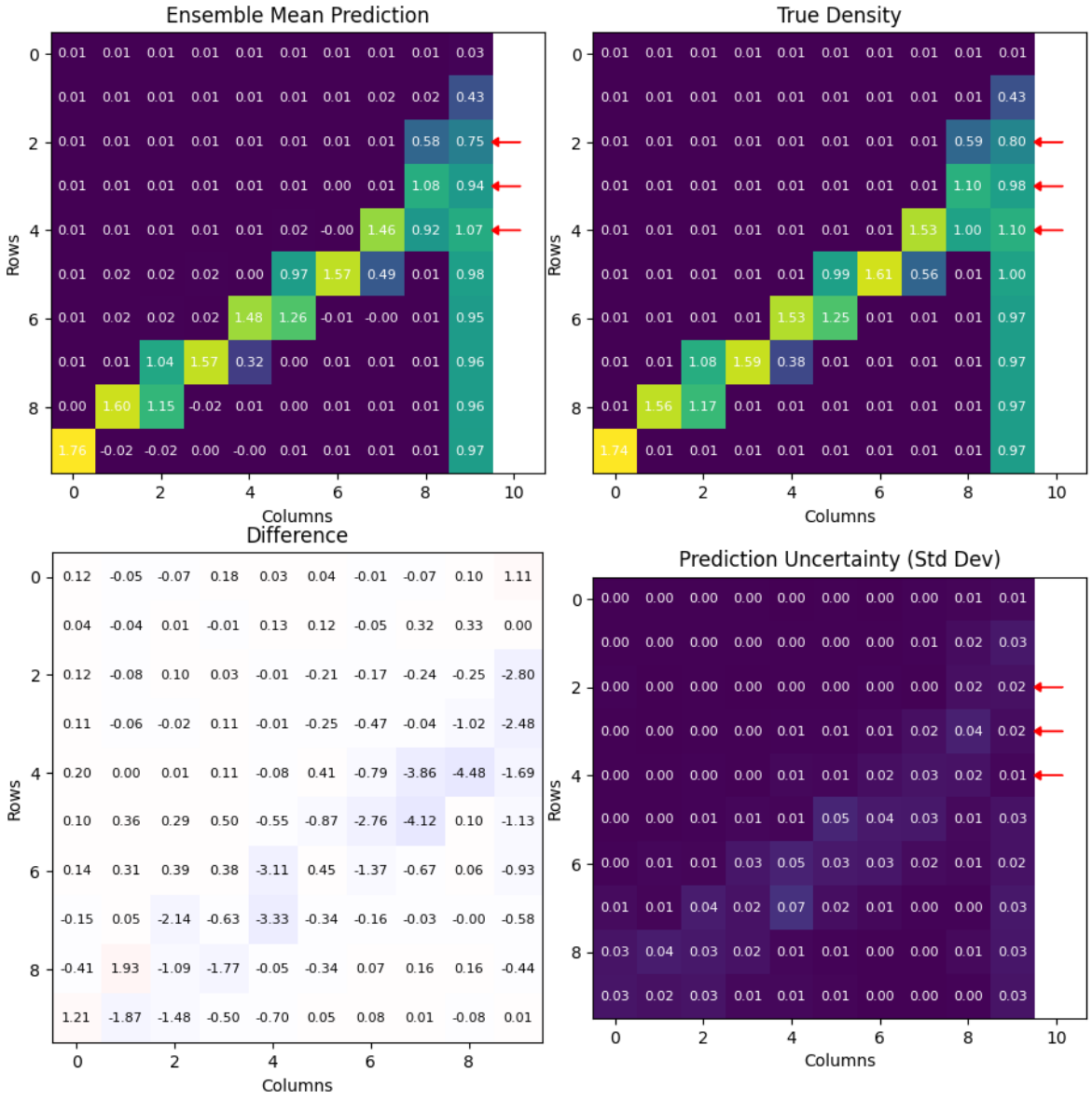


Figure F-2: Square load placed on the right side. Ensemble model can accurately represent this loading family.

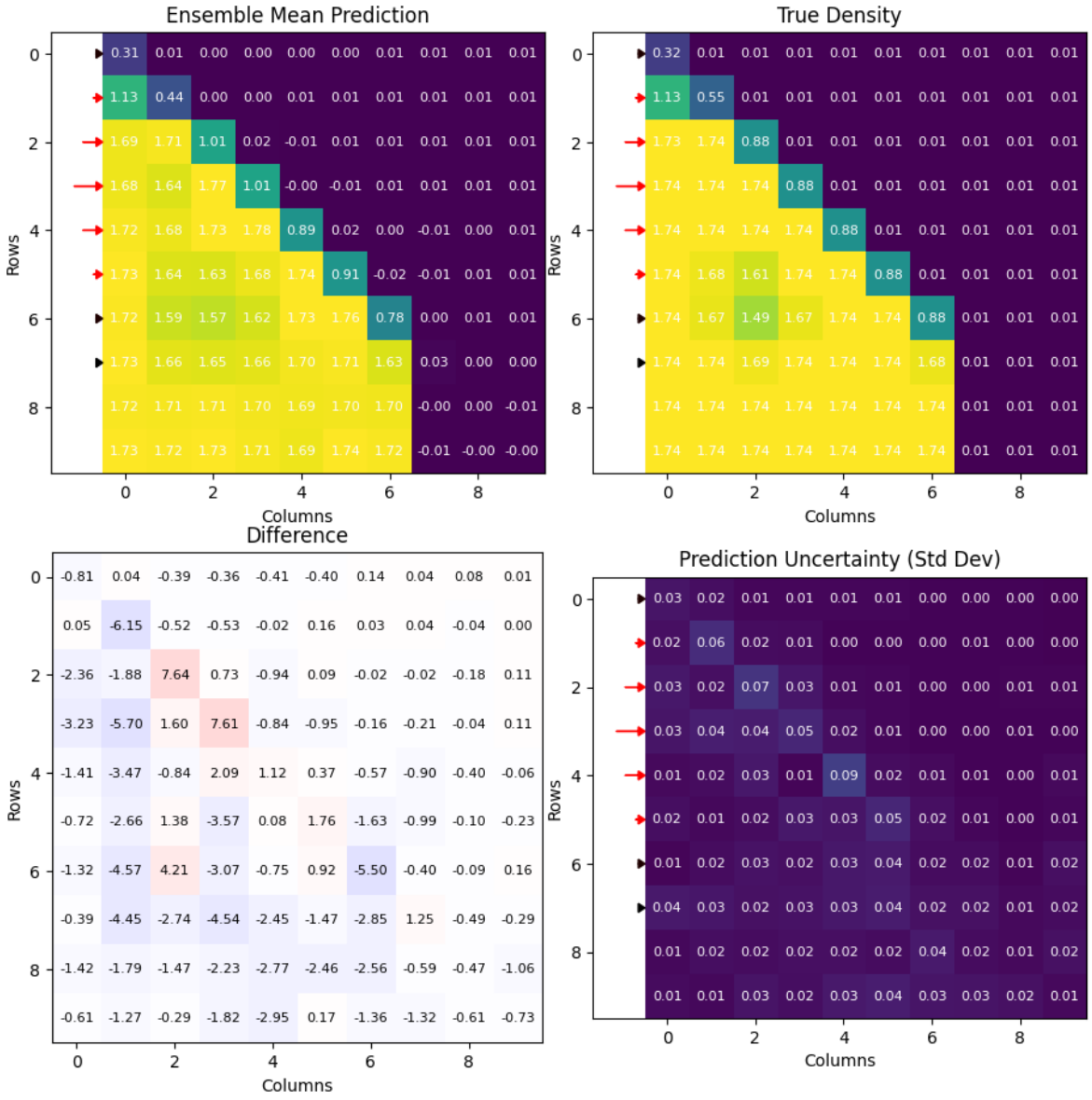


Figure F-3: Gaussian load placed on the left side. Ensemble model can accurately represent this loading family.

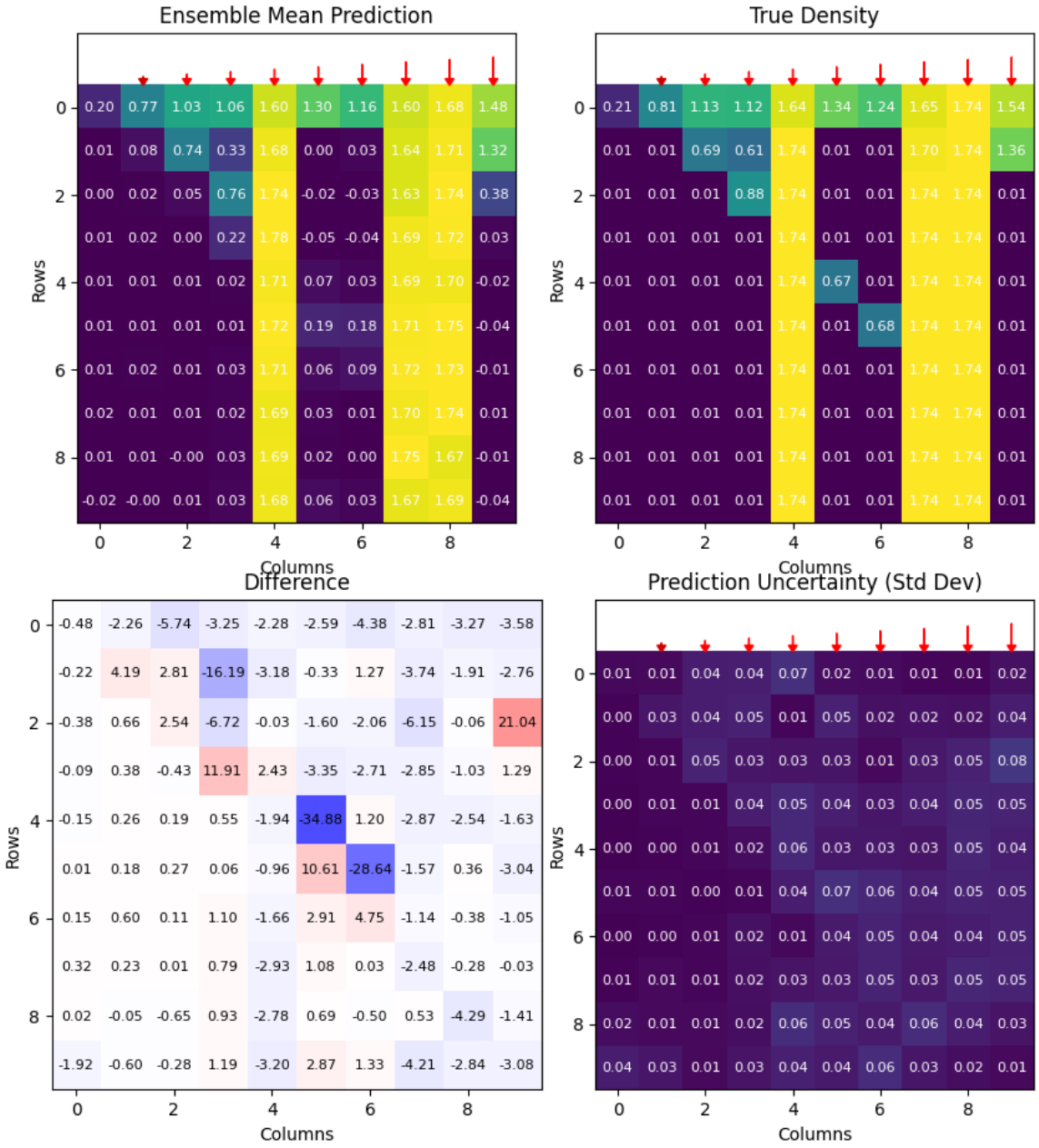


Figure F-4: Ramp load placed on the top. Ensemble model can accurately represent this loading family.

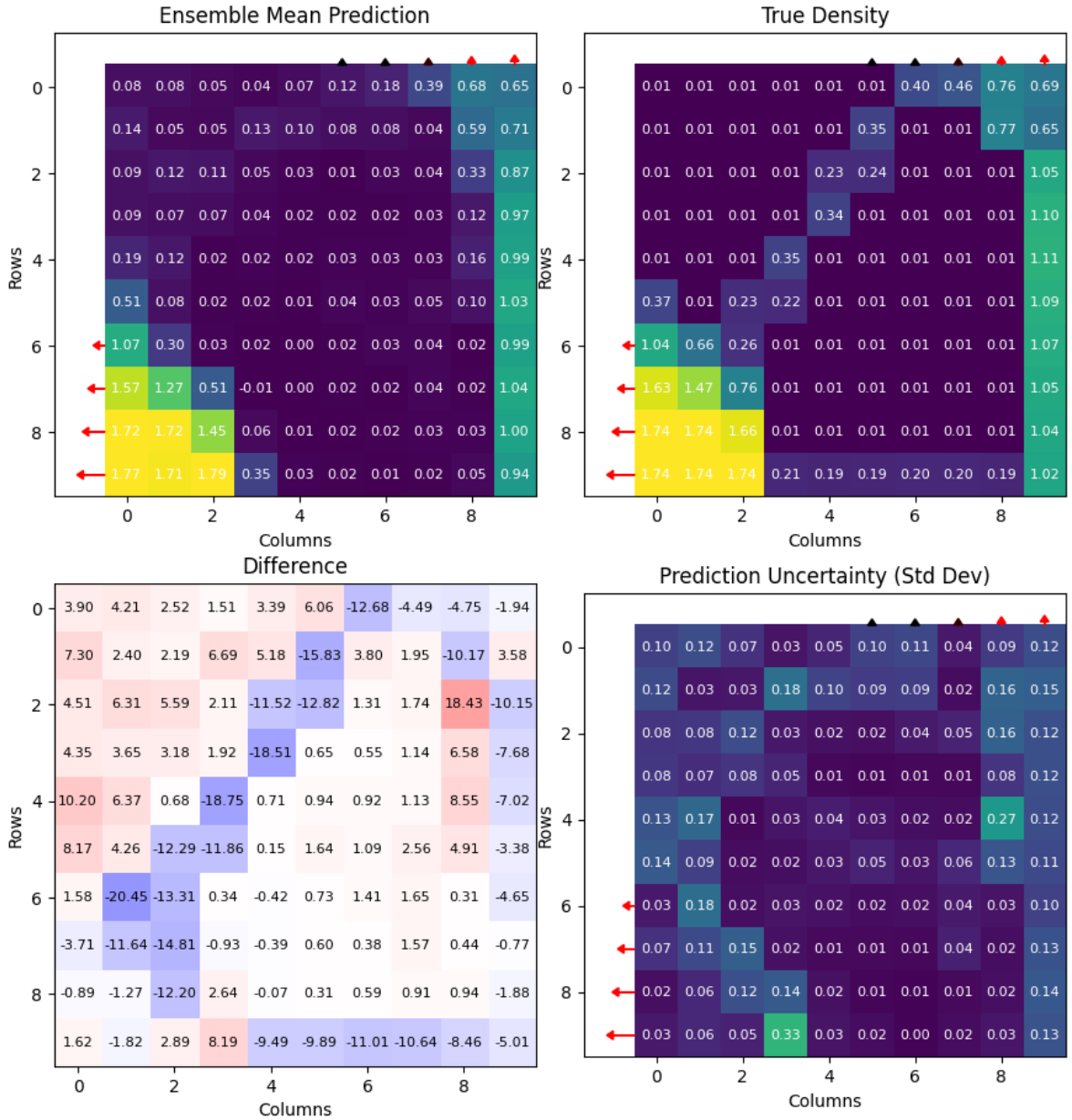


Figure F-5: Two ramp loads placed along the top and the left side. Ensemble model still finds high density areas but error starts increasing in density boundary regions (between high and low density).

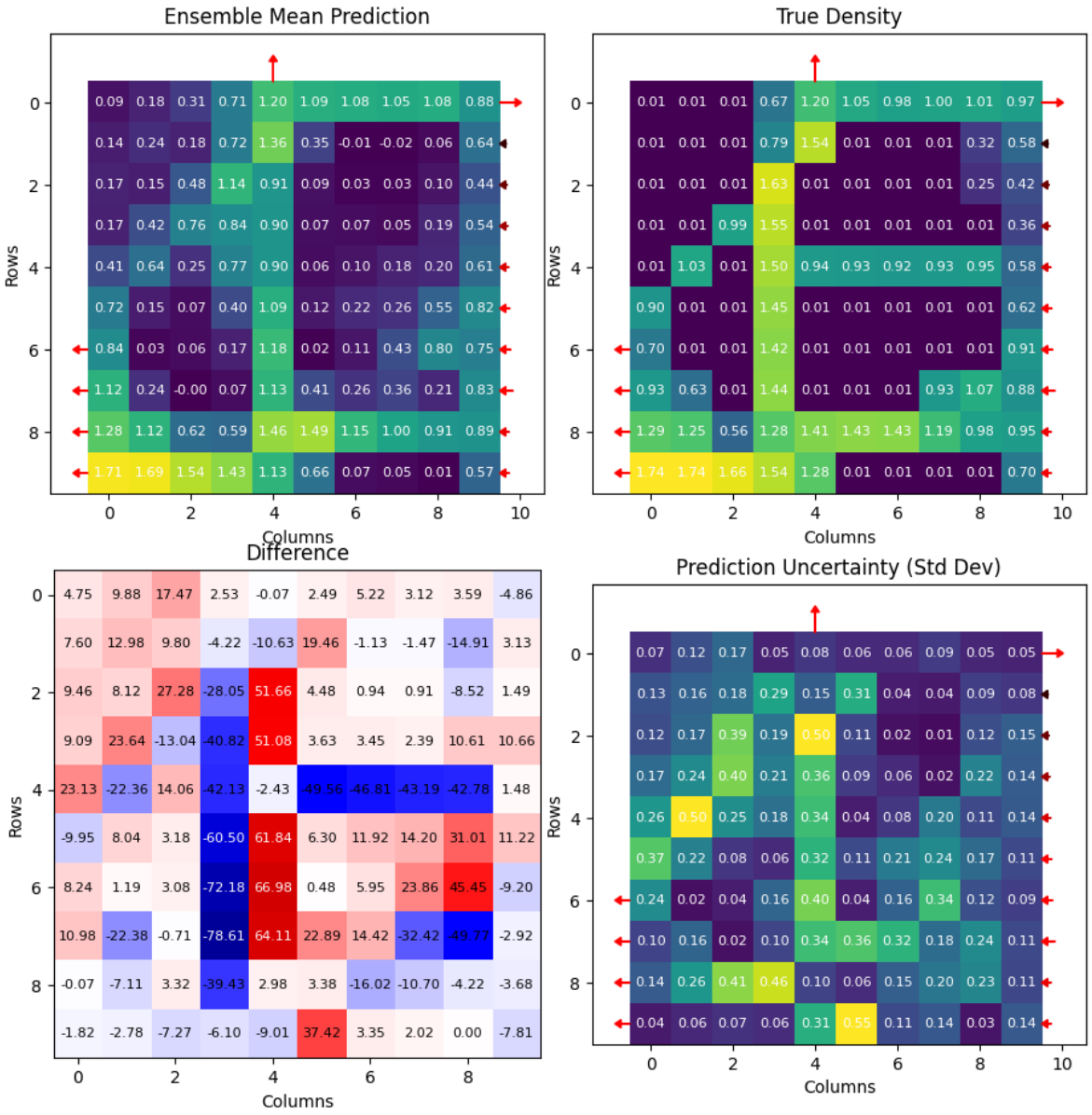


Figure F-6: Three loads placed on all sides of the model. Ensemble model is still able to find bone density morphology but accurate density representation begins to break down.

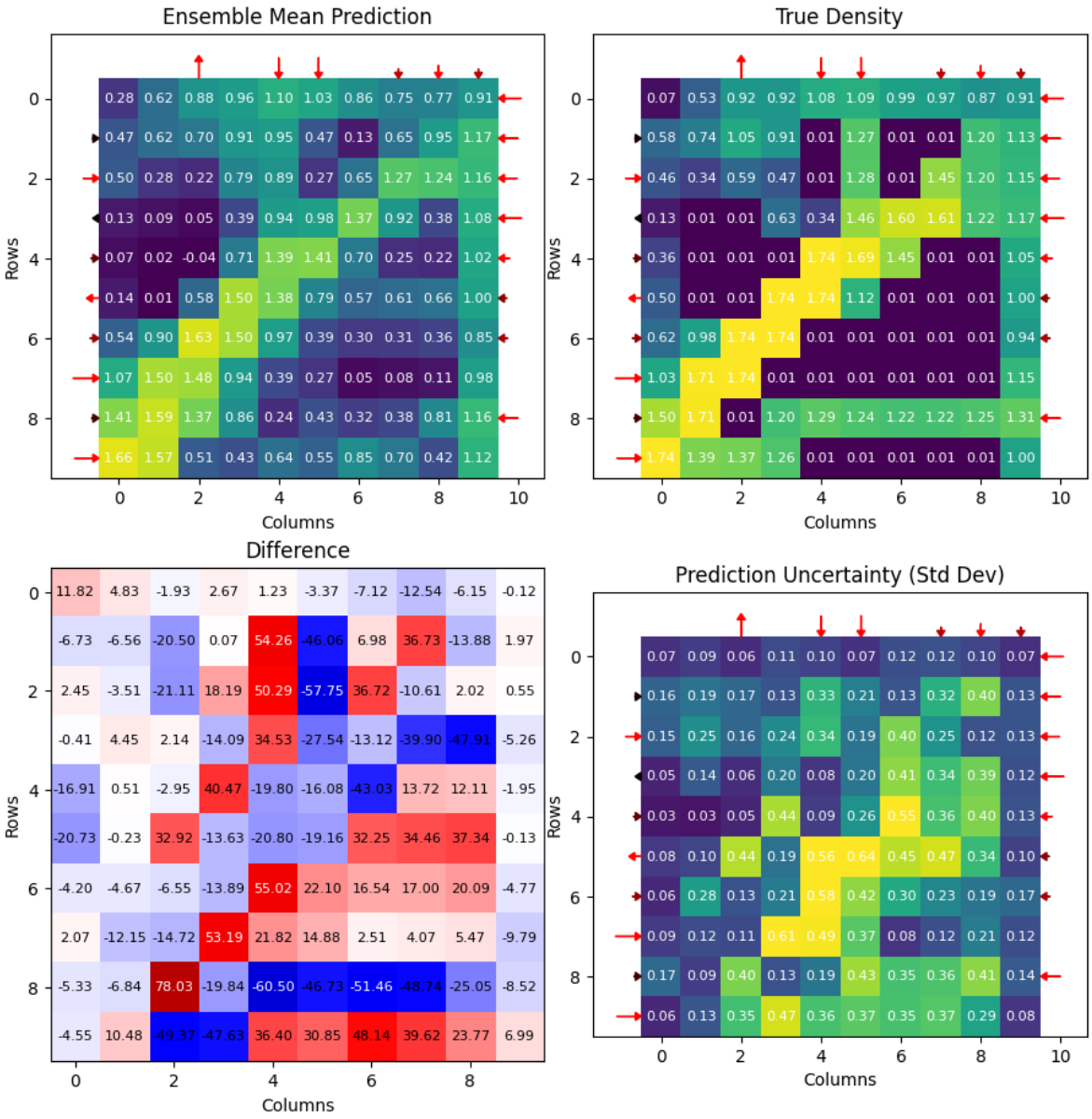


Figure F-7: Eight loading types on all sides of the simulation. Ensemble model performance breaks down and is unable to accurately estimate bone density.

Appendix G

Inverse Baseline Results

This appendix contains the figures for the inverse baseline used to compare the Reinforcement Learning (RL) framework performance introduced in chapter 6. A range of illustrative samples have been included to demonstrate baseline model performance at the simplified task Table G-1. Each figure in the table shows an illustrative force baseline prediction with its surrogate density (in the middle) compared to the original force density reference (on the left). A difference plot is added on the right to exemplify where they diverge.

Table G-1: Overview of baseline inverse model validation figures by side. The side header defines whether the baseline model correctly identified the side the force profile was placed on. The peak header further refines this with the absolute difference in index between the original and estimated force peak for all samples that converged to the correct side. The magnitude header represents the absolute difference between the true and estimated peak where a positive value means overestimation and negative underestimation.

MSE	SSIM	Side?	Peak?	Magnitude?	Sample	Figure
0.34	0.42	Yes	0	-15 N	5798	Figure G-1
0.00084	0.997	Yes	0	-0.51 N	2982	Figure G-2
0.0081	0.26	No	-	-6.2 N	2423	Figure G-3
0.47	0.28	No	-	-4.5 N	439	Figure G-4
0.015	0.93	Yes	0	-3.3 N	5381	Figure G-5
0.47	-0.044	No	-	-7.2 N	1258	Figure G-6
0.12	0.30	Yes	3	-6.3 N	262	Figure G-7
0.33	-0.053	Yes	3	-7.4 N	2688	Figure G-8
0.013	0.93	Yes	0	-4.7 N	1156	Figure G-9

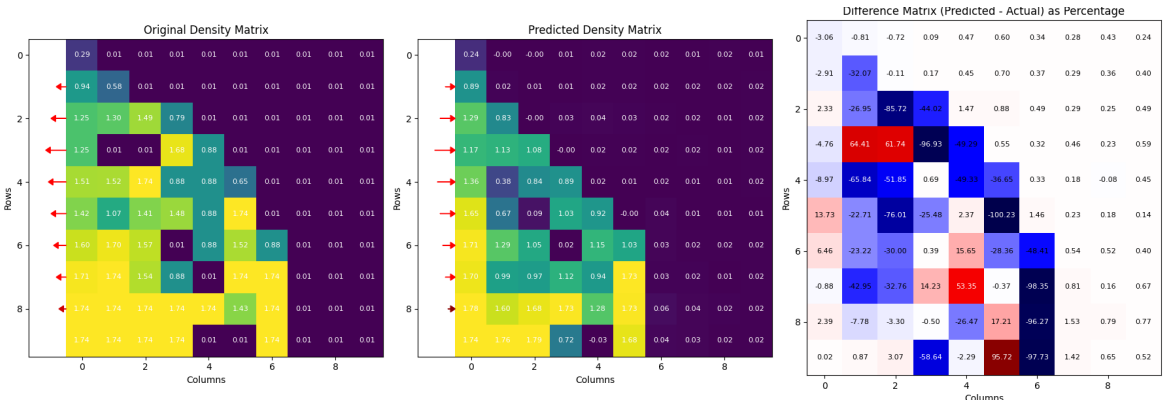


Figure G-1: Inverse baseline prediction for a left-side triangular load.

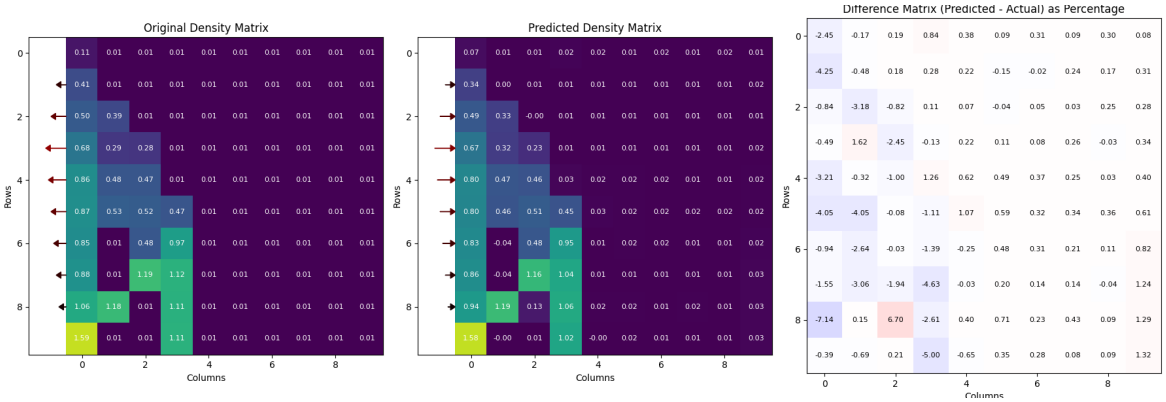


Figure G-2: Inverse baseline prediction for a left-side triangular load.

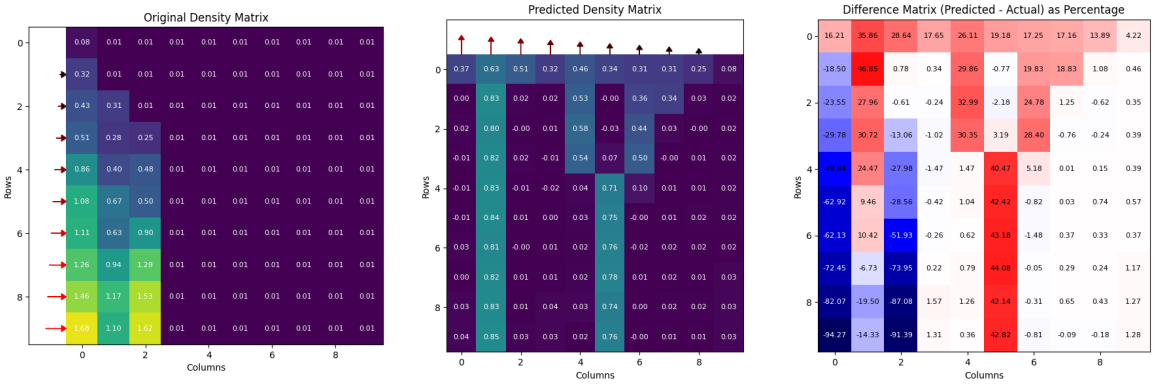


Figure G-3: Inverse baseline prediction for a left-side triangular load.

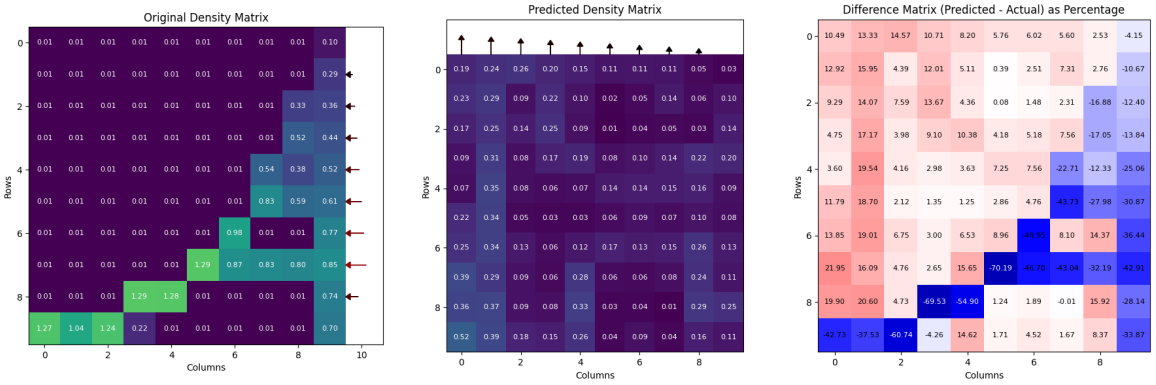


Figure G-4: Inverse baseline prediction for a right-side triangular load.

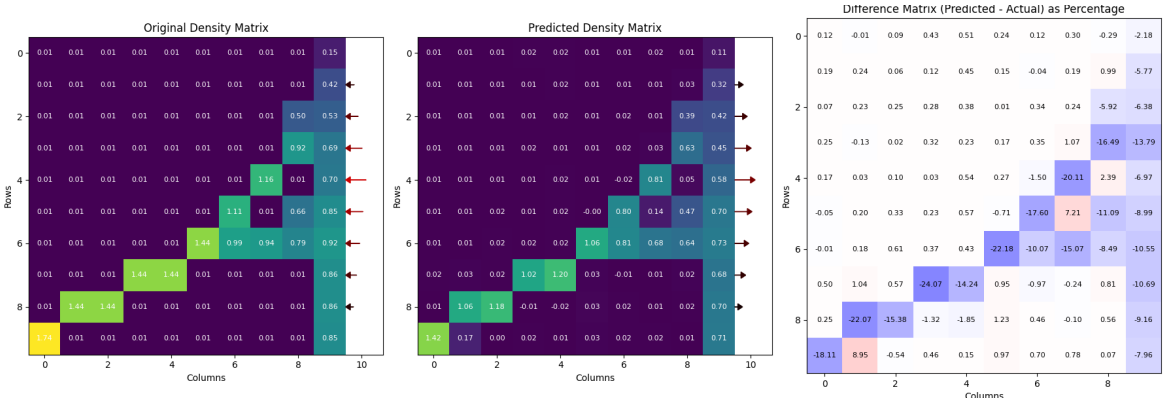


Figure G-5: Inverse baseline prediction for a right-side triangular load.

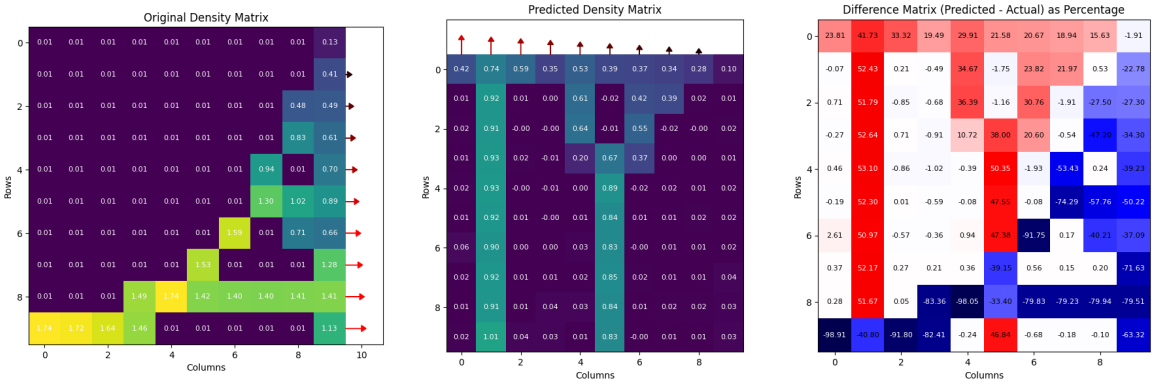


Figure G-6: Inverse baseline prediction for a right-side triangular load.

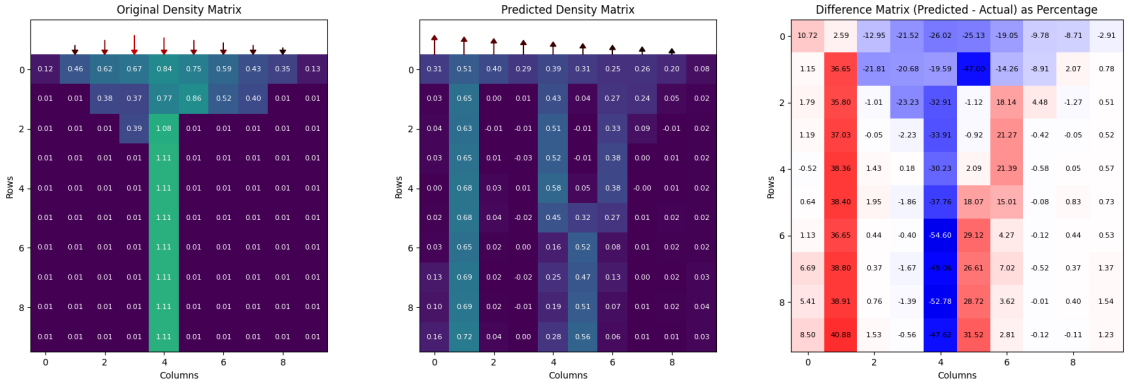


Figure G-7: Inverse baseline prediction for a top-side triangular load.

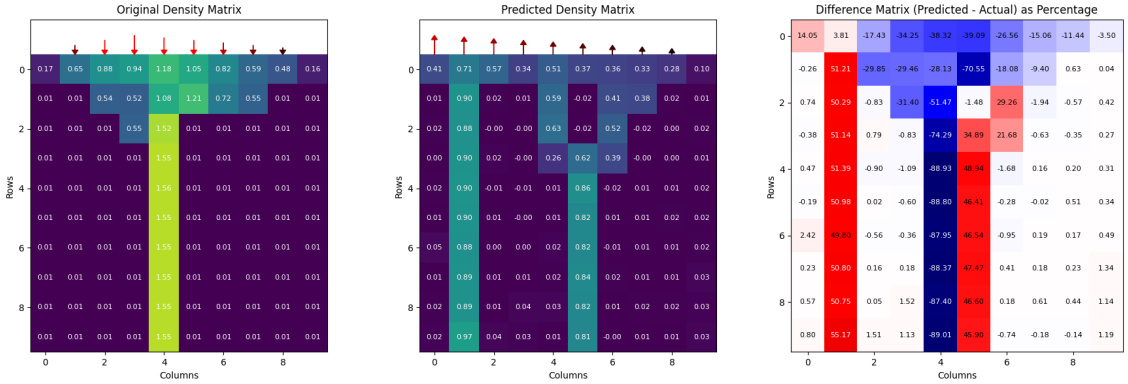


Figure G-8: Inverse baseline prediction for a top-side triangular load.

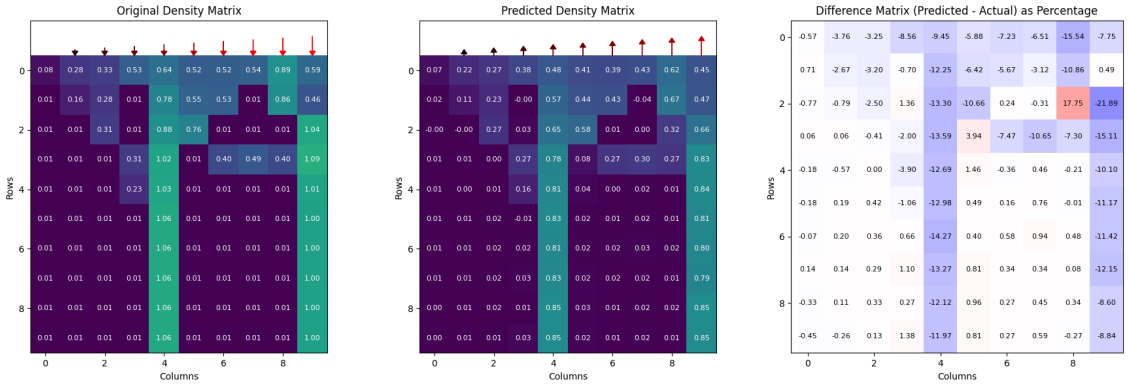


Figure G-9: Inverse baseline prediction for a top-side triangular load.

Appendix H

RL Surrogate Results

This appendix contains the figures for the Reinforcement Learning (RL) agent paired with the surrogate model. A range of illustrative samples have been included to demonstrate RL surrogate model performance at the simplified task Table H-1. Each figure in the table shows an illustrative force profile predicted by the RL agent with its surrogate density (in the middle) compared to the original force density reference (on the left). A difference plot is added on the right to exemplify where they diverge.

Table H-1: Overview of RL surrogate model validation figures by side. As all examples converge to the correct side, this header is unnecessary here but added for ease of comparison to the baseline. The peak header further refines this with the absolute difference in index between the original and estimated force peak. The magnitude header represents the difference between the true and estimated peak where a positive value means overestimation and negative underestimation.

MSE	SSIM	Side?	Peak?	Magnitude?	Sample	Figure
0.20	0.80	Yes	1	-28 N	2834	Figure H-1
0.22	0.64	Yes	2	-19 N	4398	Figure H-2
0.00053	0.998	Yes	0	+0.003 N	4212	Figure H-3
0.71	0.47	Yes	9	-61 N	1210	Figure H-4
0.00098	0.997	Yes	0	-0.18 N	472	Figure H-5
0.23	0.81	Yes	0	-15 N	3423	Figure H-6
0.24	0.72	Yes	5	-37 N	535	Figure H-7
0.087	0.62	Yes	2	+4.3 N	944	Figure H-8
0.029	0.87	Yes	1	+0.17 N	1397	Figure H-9

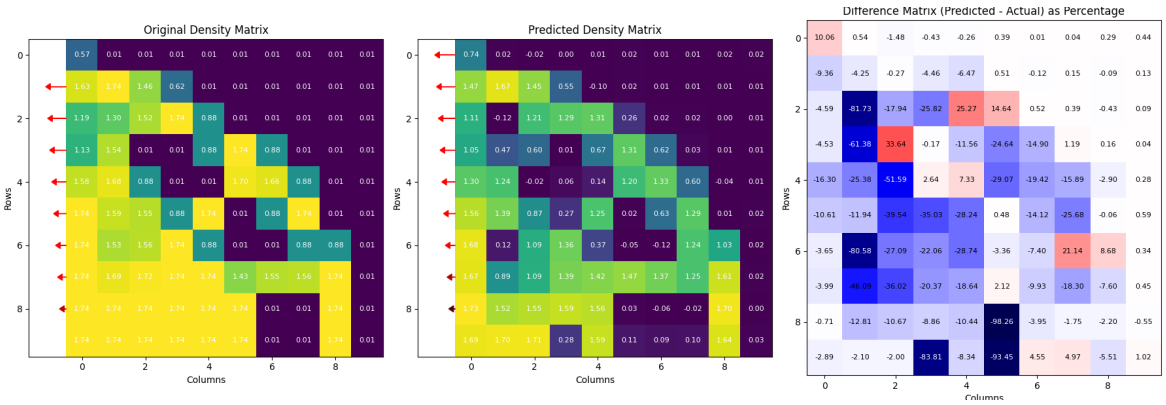


Figure H-1: RL surrogate prediction for a left-side triangular load.

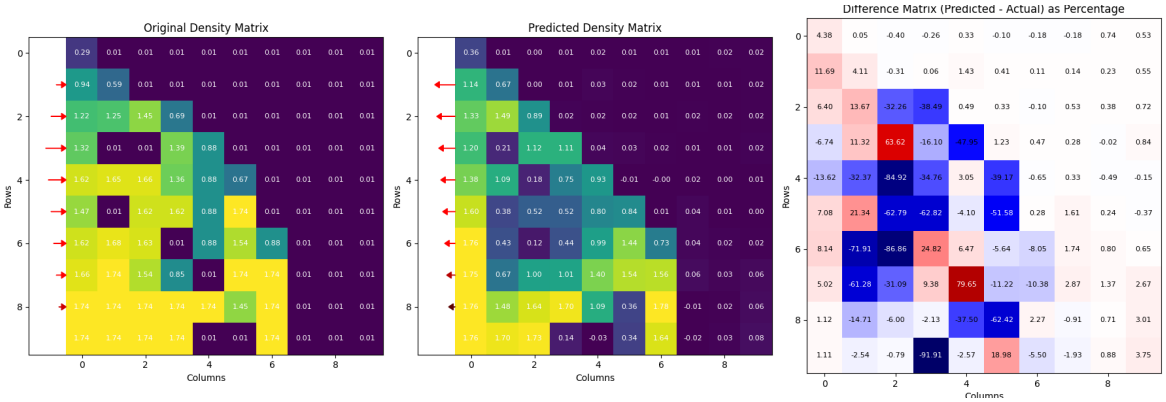


Figure H-2: RL surrogate prediction for a left-side triangular load.

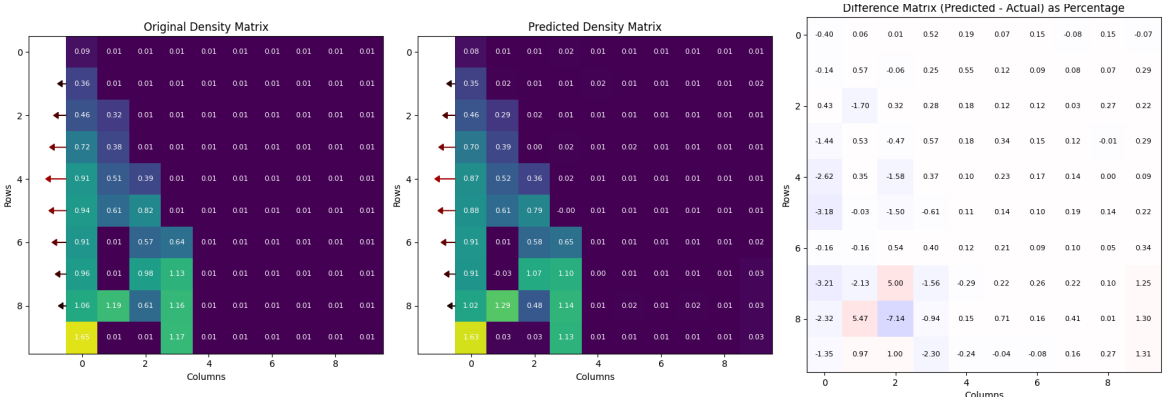


Figure H-3: RL surrogate prediction for a left-side triangular load.

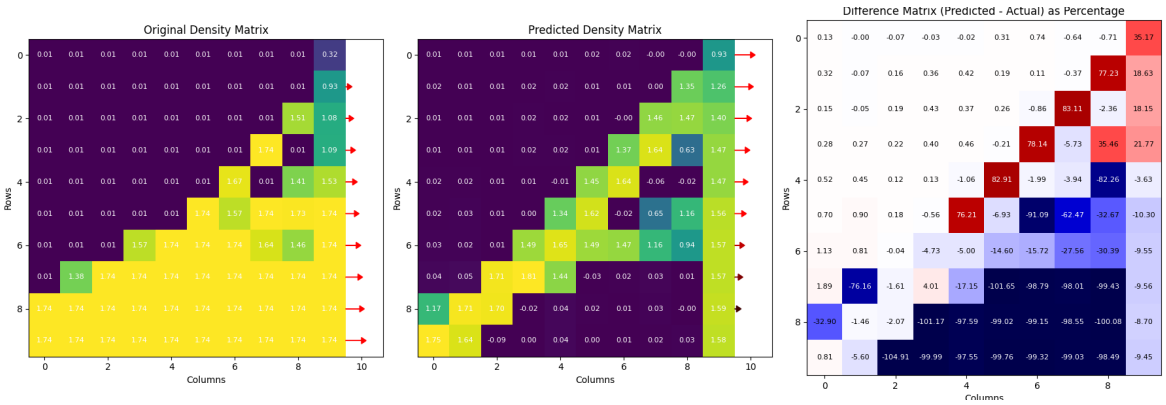


Figure H-4: RL surrogate prediction for a right-side triangular load.

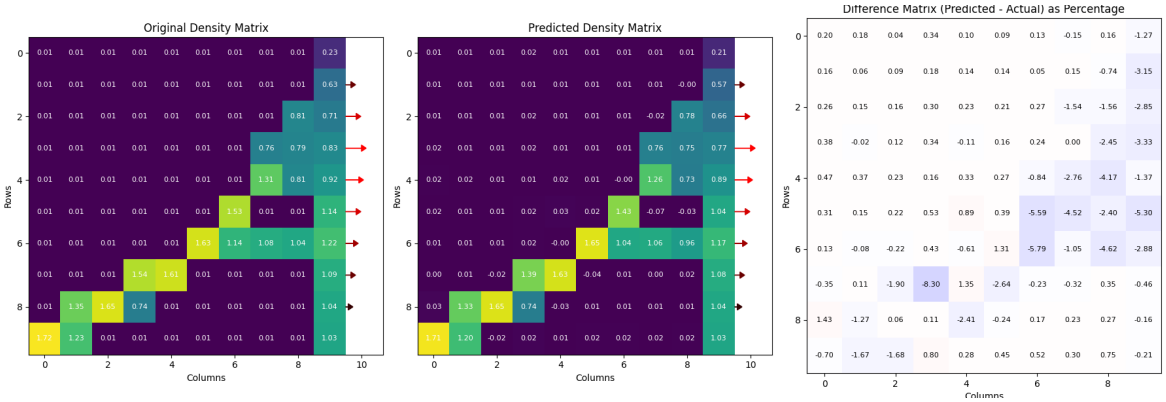


Figure H-5: RL surrogate prediction for a right-side triangular load.

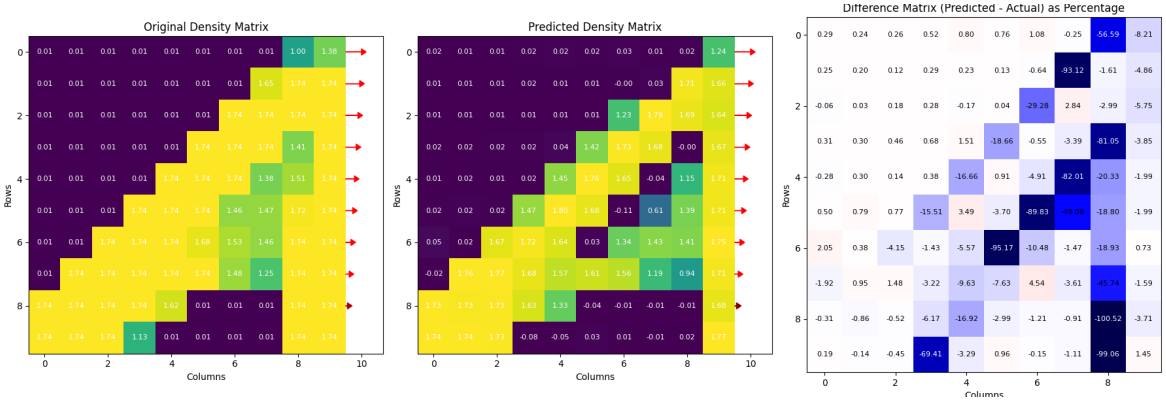


Figure H-6: RL surrogate prediction for a right-side triangular load.

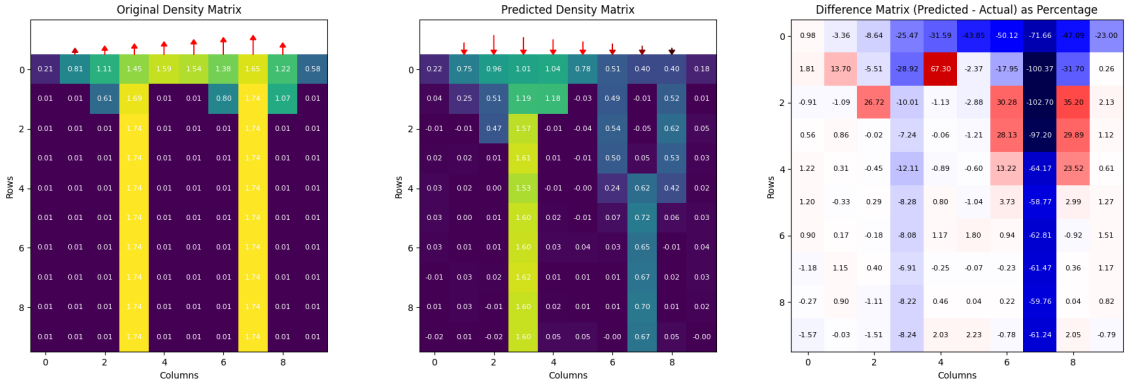


Figure H-7: RL surrogate prediction for a top-side triangular load.

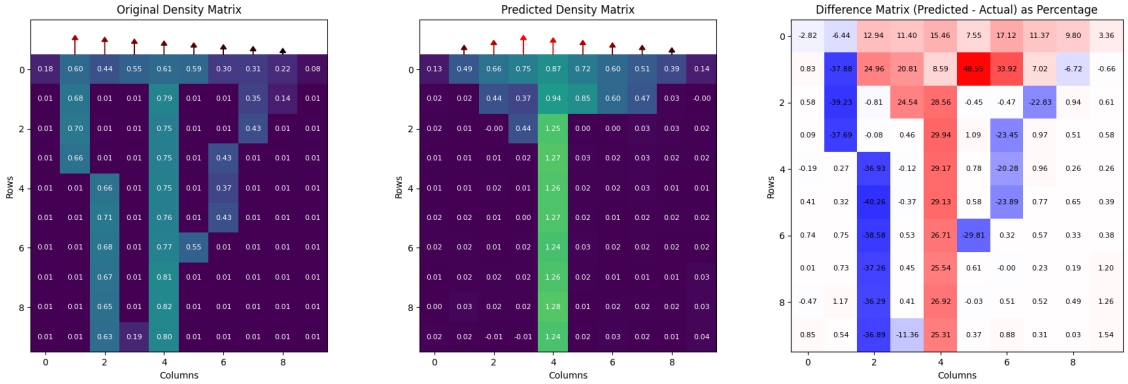


Figure H-8: RL surrogate prediction for a top-side triangular load.

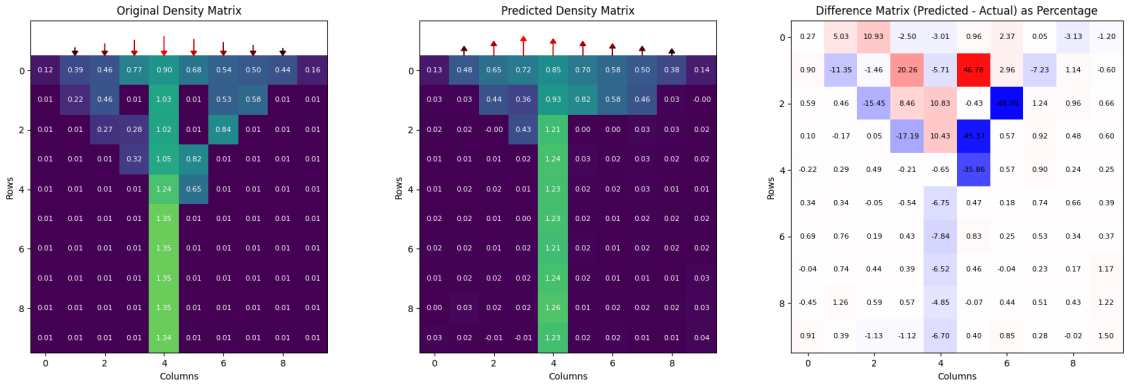


Figure H-9: RL surrogate prediction for a top-side triangular load.

Appendix I

RL Ensemble Results

This appendix contains the figures for the Reinforcement Learning (RL) agent paired with the ensemble model. A range of illustrative samples have been included to demonstrate RL ensemble model performance at the simplified task Table I-1. Each figure in the table shows an illustrative RL ensemble model force prediction with ensemble density (in the middle) compared to the original force density reference (on the left). A difference plot is added on the right to exemplify where they diverge.

Table I-1: Overview of RL ensemble model validation figures by side. As all examples converge to the correct side, this header is unnecessary here but added for ease of comparison to the baseline. The peak header further refines this with the absolute difference in index between the original and estimated force peak. The magnitude header represents the difference between the true and estimated peak where a positive value means overestimation and negative underestimation.

MSE	SSIM	Side?	Peak?	Magnitude?	Sample	Figure
0.00087	0.9963	Yes	0	+0.14 N	1976	Figure I-1
0.034	0.91	Yes	0	-1.1 N	39	Figure I-2
0.024	0.94	Yes	3	-2.3 N	3931	Figure I-3
0.00083	0.9986	Yes	0	-0.21 N	1843	Figure I-4
0.075	0.80	Yes	2	-2.2 N	2683	Figure I-5
0.00076	0.9989	Yes	0	+0.15 N	5637	Figure I-6
0.0090	0.97	Yes	0	+6.4 N	4958	Figure I-7
0.51	0.41	Yes	2	-65 N	3506	Figure I-8
0.063	0.73	Yes	5	+0.57 N	3603	Figure I-9

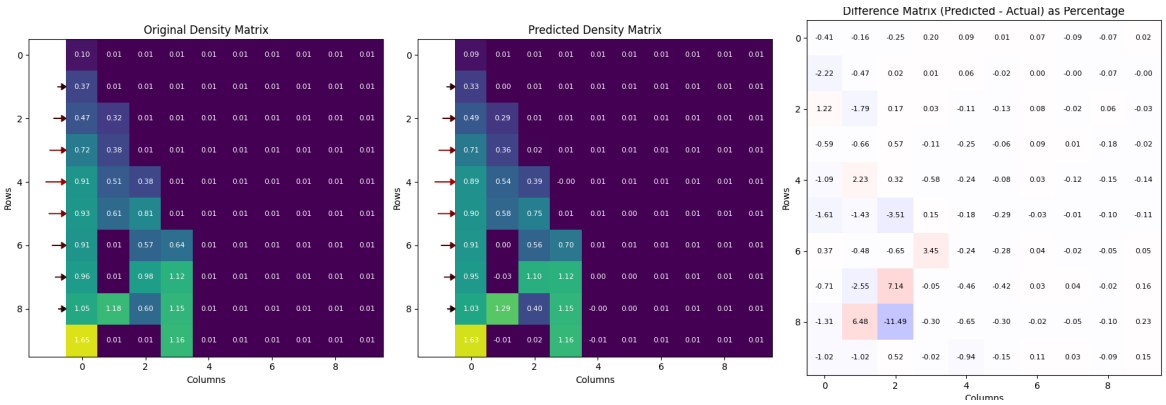


Figure I-1: RL ensemble prediction for a left-side triangular load.

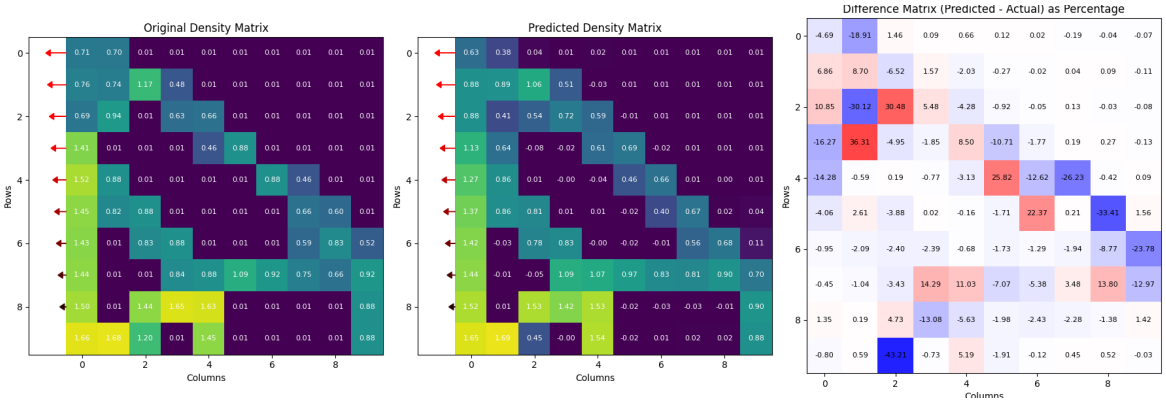


Figure I-2: RL ensemble prediction for a left-side triangular load.

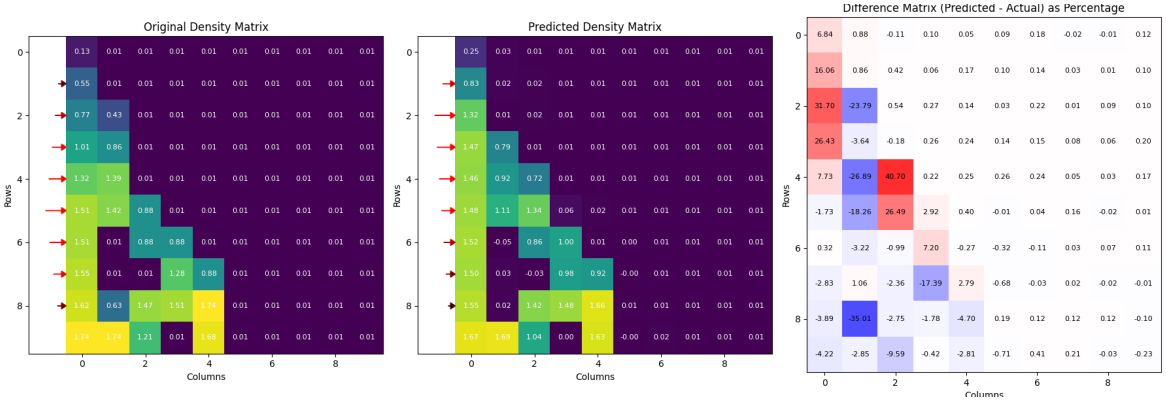


Figure I-3: RL ensemble prediction for a left-side triangular load.

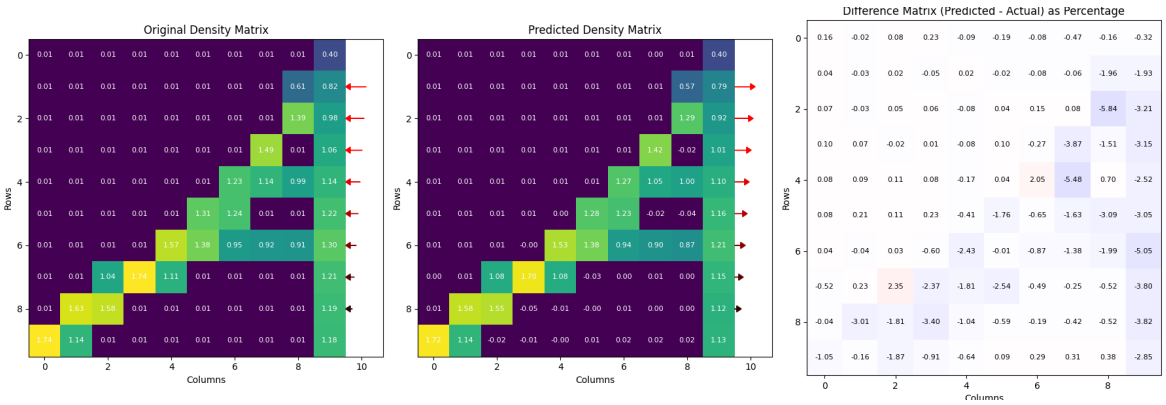


Figure I-4: RL ensemble prediction for a right-side triangular load.

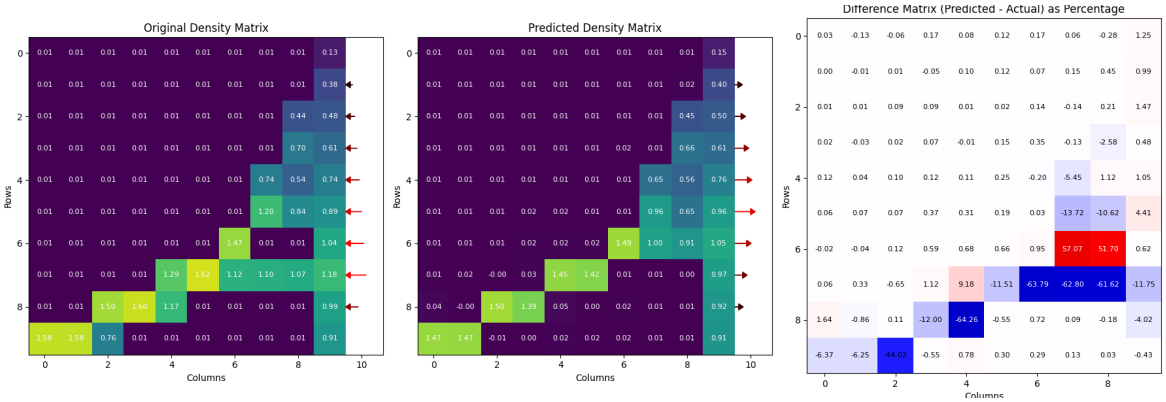


Figure I-5: RL ensemble prediction for a right-side triangular load.

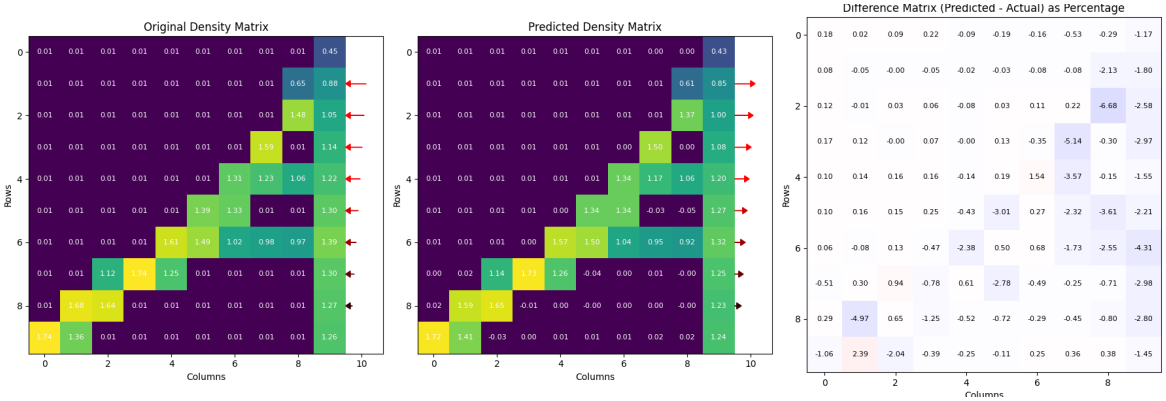


Figure I-6: RL ensemble prediction for a right-side triangular load.

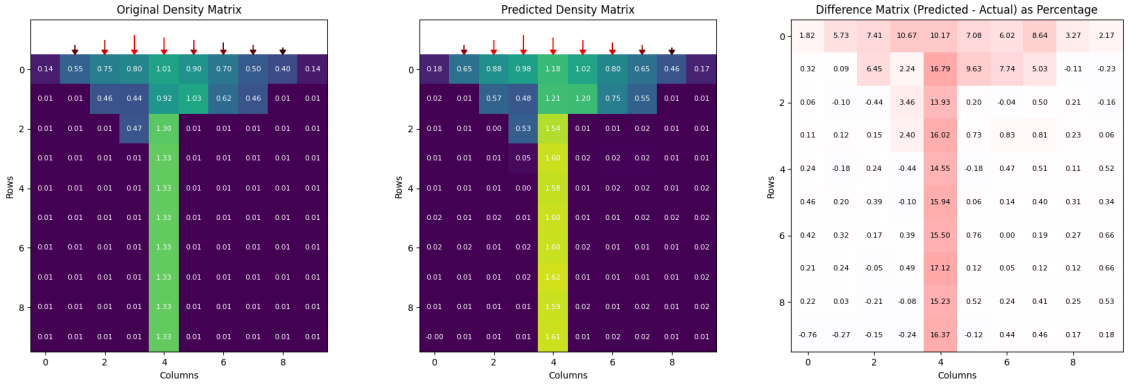


Figure I-7: RL ensemble prediction for a top-side triangular load.

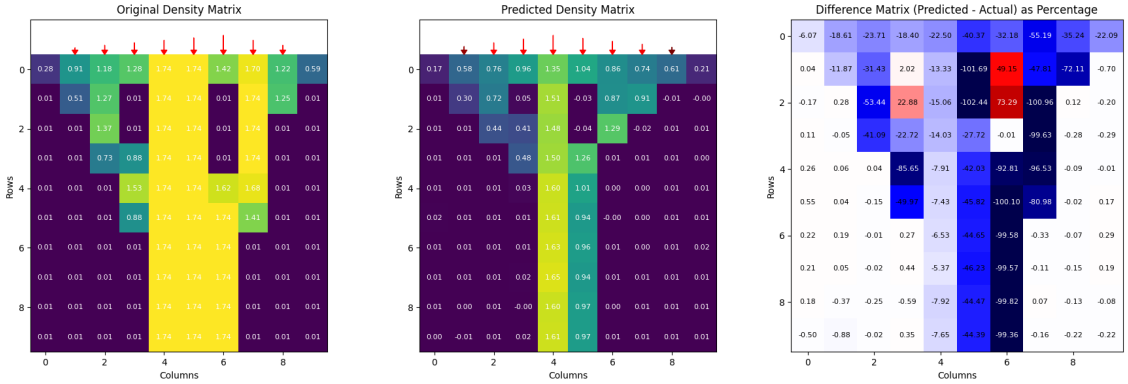


Figure I-8: RL ensemble prediction for a top-side triangular load.

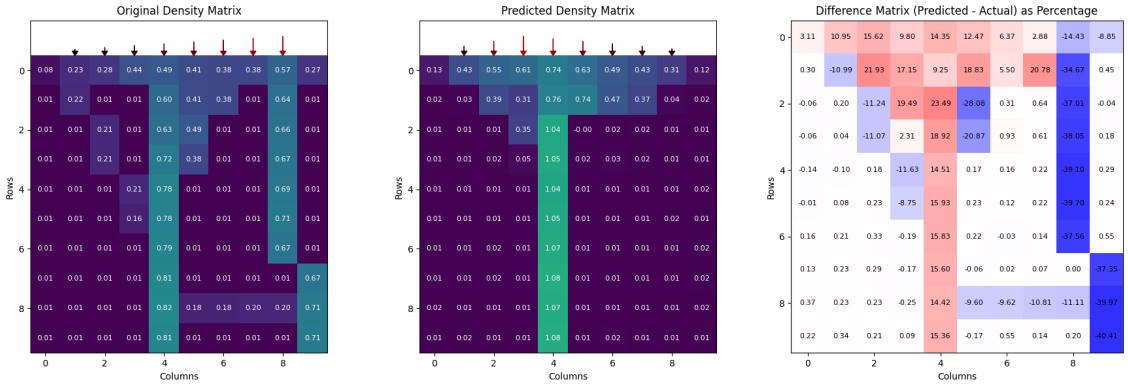


Figure I-9: RL ensemble prediction for a top-side triangular load.

Appendix J

Pseudocode

This appendix provides detailed pseudocode listings that complement the methodological descriptions in the main text and support full reproducibility of the proposed framework.

Algorithm 1 describes the training procedure for the surrogate model. Algorithm 2 details the training procedure for the inverse baseline mode. Algorithm 3 formalises the Reinforcement Learning (RL) environment. Finally, Algorithm 4 presents the Proximal Policy Optimisation (PPO) training procedure used to learn the policy and value networks of the RL framework.

Algorithm 1 Training Procedure for the Surrogate Model

Require: Training samples $(\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}})$, validation samples $(\mathbf{X}_{\text{val}}, \mathbf{Y}_{\text{val}})$, training parameters Θ

Ensure: Trained network parameters θ^*

Compute normalization parameters (μ_x, σ_x) and (μ_y, σ_y) from $\mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}}$
Normalise training and validation data
Initialize neural network f_θ
Initialize optimizer AdamW(θ) and learning-rate scheduler
$L_{\text{best}} \leftarrow \infty$, $p \leftarrow 0$
for epoch $= 1$ to N_{epochs} **do**
    Training phase
    Set model to training mode
    for all mini-batches $(\mathbf{x}, \mathbf{y}) \subset \mathbf{X}_{\text{train}}$ **do**
        $\hat{\mathbf{y}} \leftarrow f_\theta(\mathbf{x})$
        $L \leftarrow \mathcal{L}(\hat{\mathbf{y}}, \mathbf{y})$
        if L is finite **then**
            Backpropagate L and update θ
        end if
    end for
    Validation phase
    Set model to evaluation mode
    $L_{\text{val}} \leftarrow \mathcal{L}(f_\theta(\mathbf{X}_{\text{val}}), \mathbf{Y}_{\text{val}})$
    Update learning-rate scheduler using L_{val}
    Early stopping and checkpointing
    if $L_{\text{val}} + \delta < L_{\text{best}}$ **then**
        $L_{\text{best}} \leftarrow L_{\text{val}}$
        $\theta^* \leftarrow \theta$
        $p \leftarrow 0$
    else
        $p \leftarrow p + 1$
    end if
    if $p \geq P$ **then**
        break
    end if
end for
Load best model parameters θ^*
return θ^*

Algorithm 2 Inverse Baseline Neural Network Training and Evaluation

Require: Dataset of density–force pairs $\{(\rho^{(i)}, F^{(i)})\}$, training parameters Θ, maximum epochs N, early stopping patience P

Split dataset into training, validation, and test sets

▷ Force profile parameterisation

for each force profile $F^{(i)}$ **do**
    Extract peak location $\ell^{(i)}$, peak side $s^{(i)}$, and peak magnitude $m^{(i)}$
    Encode $(\ell^{(i)}, s^{(i)})$ as a 30-dimensional one-hot vector
    Append $m^{(i)}$ to form a 31-dimensional target vector $\mathbf{y}^{(i)}$
end for

▷ Normalisation

Compute output mean μ_y and standard deviation σ_y from training targets
Normalise training, validation, and test targets
Convert densities ρ and targets $\mathbf{y}$ to tensors and move to device
Initialise inverse model g_θ
Initialise AdamW optimiser and learning-rate scheduler
$L_{\text{best}} \leftarrow \infty$, $p \leftarrow 0$
for epoch $= 1$ to N **do**
    Set model to training mode
    for each mini-batch $(\rho, \mathbf{y})$ **do**
        $\hat{\mathbf{y}} \leftarrow g_\theta(\rho)$
        $L \leftarrow \text{MSE}(\hat{\mathbf{y}}, \mathbf{y})$
        if L is finite **then**
            Update θ via backpropagation
        end if
    end for

▷ Validation

    Set model to evaluation mode
    $L_{\text{val}} \leftarrow \text{MSE}(g_\theta(\rho_{\text{val}}), \mathbf{y}_{\text{val}})$
    Update learning-rate scheduler using L_{val}
    if $L_{\text{val}} + \delta < L_{\text{best}}$ **then**
        $L_{\text{best}} \leftarrow L_{\text{val}}$
        Save model parameters $\theta^* \leftarrow \theta$
        $p \leftarrow 0$
    else
        $p \leftarrow p + 1$
    end if
    if $p \geq P$ **then**
        break
    end if
end for
Load best model parameters θ^*

Algorithm 3 Reinforcement Learning Environment for the Inverse Bone Remodelling Problem

Require:
Target density dataset $\mathcal{D}_\rho = \{\rho_{\text{target}}^{(i)}\}_{i=1}^{N}$,
trained surrogate model $f_{\text{sur}}(\cdot)$

Initialise maximum episode length $N_{\max}$, force bounds $F_{\min}, F_{\max}$ and density tolerance ϵ
function RESET
  Reset force estimate $\hat{F}_0 \leftarrow \mathbf{0}$ and predicted density $\hat{\rho}_0 \leftarrow \mathbf{0}$
  Reset predicted density $\hat{\rho}_0 \leftarrow \mathbf{0}$ and timestep $t \leftarrow 0$
  Initialise error map to new sample $e_0 \leftarrow \rho_{\text{target}}$
  return observation $o_0 = [e_0,\ \hat{F}_0]$
end function
function STEP($A_t = (a_t^{(\text{loc})}, a_t^{(\text{peak})}, a_t^{(\text{stay})})$)
  Update peak magnitude:

$$m_{t+1} \leftarrow \text{clip}(m_t + a_t^{(\text{peak})},\ F_{\min},\ F_{\max})$$

  if $a_t^{(\text{stay})} \leq 0$ **then**
    Update peak location:

$$\ell_{t+1} \leftarrow (\ell_t + \text{sign}(a_t^{(\text{loc})})) \bmod (3L)$$

  else
    $\ell_{t+1} \leftarrow \ell_t$
  end if
  Construct force profile $\hat{F}_{t+1} \leftarrow$ TRIANGULARPROFILE(m_{t+1}, ℓ_{t+1})
  Predict density $\hat{\rho}_{t+1} \leftarrow f_{\text{sur}}(\hat{F}_{t+1})$
  Compute reward:

$$r_t \leftarrow \text{SSIM}(\rho_{\text{target}}, \hat{\rho}_{t+1})$$

  Compute density error map:

$$e_{t+1} \leftarrow \rho_{\text{target}} - \hat{\rho}_{t+1}$$

  Increment timestep $t \leftarrow t + 1$
  success $\leftarrow \forall |e_{t+1}| < \epsilon$
  done $\leftarrow$ *success* $\vee\ (t \geq N_{\max})$
  if *success* **then**
    $r_t \leftarrow r_t + (N_{\max} - t)$ ▷ Early convergence bonus
  end if
  return $(o_{t+1} = [e_{t+1}, \hat{F}_{t+1}],\ r_t,\ \textit{done})$
end function

Algorithm 4 PPO-Based Reinforcement Learning for the Inverse Bone Remodelling Problem

Require:

Target density dataset $\mathcal{D}_\rho = \{\rho_{\text{target}}^{(i)}\}_{i=1}^{N}$

Trained surrogate forward model $f_{\text{sur}}(F_t)$

Policy network $\pi_\theta(F_t \mid \rho_t)$ and value network $V_\phi(\rho_t)$

PPO hyperparameters:

On-policy batch length $N_{\text{rollout}} = 2048$

Mini-batch size $N_{\text{batch}} = 64$

Discount factor $\gamma = 0.99$

GAE parameter $\lambda = 0.95$

Entropy coefficient $c_{\text{ent}} = 0.01$

Total training timesteps T

Ensure: Trained policy parameters θ^* and ϕ^*

Initialise policy parameters θ and value parameters ϕ

Initialise parallel environments by sampling $\rho_0 \sim \mathcal{D}_\rho$

Apply reward normalisation

for $t = 1$ to T **do**

  Initialise rollout buffer $\mathcal{B} = \emptyset$

  for $k = 1$ to N_{rollout} **do**

    Observe current state S_k

    Sample action $A_k \sim \pi_\theta(\cdot \mid S_k)$

    Compute estimated loading condition $\hat{F}_k$ from S_k and action A_k

    Predict next density $\hat{\rho}_{k+1} = f_{\text{sur}}(\hat{F}_k)$

    Calculate next state $S_{k+1} = S(\hat{\rho}_{k+1}, \rho, F_k)$

    Compute reward $r_k = R(\hat{\rho}_{k+1}, \rho)$

    Store (S_k, A_k, r_k, S_{k+1}) in $\mathcal{B}$

  end for

  Compute advantages $\hat{\Lambda}_k$ using Generalised Advantage Estimation (GAE)

  Compute returns $\hat{R}_k$

  for each mini-batch of size N_{batch} sampled from $\mathcal{B}$ **do**

    Update policy parameters θ by maximising PPO clipped objective

    Update value parameters ϕ by minimising value loss

    Apply entropy regularisation weighted by c_{ent}

  end for

end for

return Trained policy parameters θ^* and ϕ^*