The Application of Neural Networks to Predict Skin Evolution After Burn Trauma

Literature Study

Selma Husanovic

Supervisors:
Alexander Heinlein
Fred Vermolen

November 19, 2023

Contents

1 Introduction

Post-burn wounds present a significant challenge for individuals, affecting a considerable number of people worldwide. Beyond the immediate pain and discomfort they cause, these wounds, particularly those proximal to joints, can lead to functional impairments and immobility. Understanding the evolution of wounds following burn trauma is paramount for clinicians to devise effective treatment strategies. While numerical models have proven effective in providing accurate approximations of wound evolution, they come with limitations. These models are often computationally expensive, requiring substantial time for execution, and altering parameters necessitates rerunning the entire model.

Recent research has shifted towards exploring the potential of machine learning, specifically neural networks, in predicting skin evolution post-burn trauma. Advantages of using neural network surrogates include high accuracies and fast evaluation. This study serves as an introduction to the application of neural networks in this domain. Additionally, it formulates a new approach for contributing novel insights to the field of wound evolution prediction.

The study is organised in the following way. Chapter 2 provides a foundational understanding of the biology of dermal wound healing, setting the stage for subsequent discussions. In Chapter 3, the classical theory of elastic deformation is explored, as this theory forms the basis for the mathematical model describing wound evolution, which incorporates the concept of morphoelasticity. Chapter 4 delves into the specifics of morphoelasticity. Chapter 5 introduces the morphoelastic mathematical model for burn injuries, bridging the gap between biological principles and mathematical formulations. Chapter 6 provides an overview of machine learning, particularly focusing on neural networks. Chapter 7 surveys existing applications of neural networks to predicting skin evolution after burn trauma, providing a comprehensive understanding of the current state of the field. Finally, chapter 8 outlines our plans for a new research approach, building upon the insights gained from prior chapters and contributing to the advancement of predictive modelling in post-burn wound evolution.

2 The Biology of Dermal Wound Healing

This chapters serves to give a comprehensive description of the current biological understanding of dermal wound healing. Section 2.1 gives a brief introduction into skin anatomy and physiology. Section 2.2 highlights the extracellular matrix, a vital component of the skin that needs to be rebuilt during wound healing. Section 2.3 delves into the four phases of wound healing and explains the biological and mechanical processes at work. Lastly, section 2.4 focuses on skin injuries due to severe burns. The emphasis is on complications due to an abnormal healing response.

2.1 The Human Skin

The human skin is a complex, multilayered organ, consisting of heterogeneous cell types and extracellular components [15]. It is one of the largest organs in the human body, having a surface area of approximately 2m^2 and making up 16% of the total body weight [38]. Amongst its many functions, the most important ones are preventing the organism from dehydrating, while protecting it from its environment. Skin is dynamic, able to heal itself and responsive to the external environment, ensuring human survival [16]. As schematically depicted in Figure 1, the skin consists of three layers: the *epidermis*, the *dermis* and the *subcutis*.

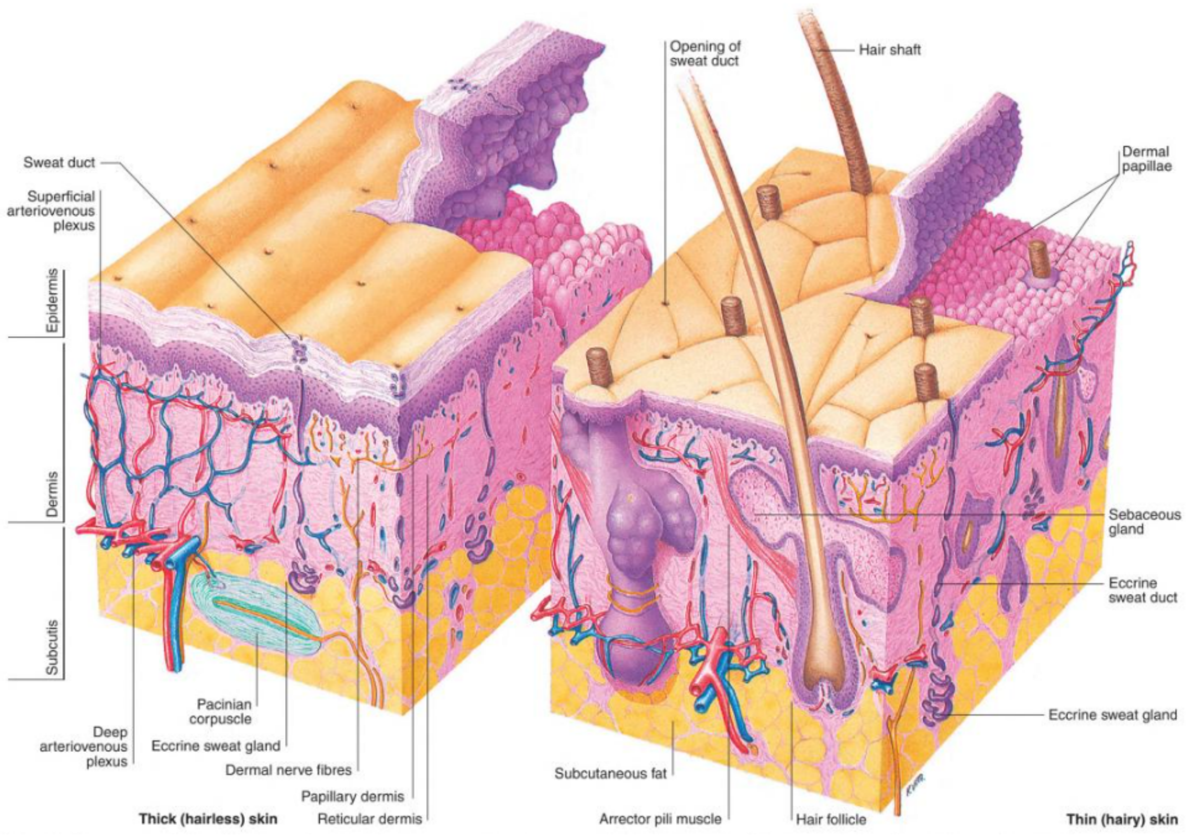


Figure 1: Schematic structure of the skin. Comparison of thick, hairless skin (left) and thin, hairy skin (right). The three skin layers are clearly visible. Taken from [16].

The epidermis is the outermost layer and is approximately 0.1mm thick, although thickness can vary depending on the location [16]. It primarily functions as a protective barrier. The main cell type in the epidermis is the *keratinocyte*, which constitutes 90% of epidermal cells [36]. Keratinocytes differentiate upwards through the epidermis. Their maturation stages can be divided into four physical layers. It takes approximately 35 days for the whole epidermal layer to be replaced by new cells [38].

Immediately below and connected to the epidermis lies the dermis. Its thickness varies between 0.3mm (e.g., on the eyelids) and 0.6mm (e.g., on the back and soles) [37]. The dermis serves numerous valuable purposes: it provides firmness, flexibility and tensile strength to the skin. Moreover, it binds water, regulates the temperature and contains receptors of sensory stimuli [15]. The dermis is less cellular than the epidermis, consisting primarily of fibrous *extracellular matrix* (ECM), that surrounds the dermal cells and other constituents (e.g. neurovascular network, sensory receptors). The ECM plays an important role in wound healing and will be treated in more detail in the subsequent section. The most abundant cell type in the dermis is the *fibroblast*, that migrates through the tissue and is responsible for producing and maintaining the ECM [16]. Other skin cells present are immune cells that protect from pathogens, e.g., *macrophages*; mast cells that are involved in allergic reactions, blood vessel, and nerve cells [37].

Directly below the dermis lies the subcutis (sometimes called the hypodermis), a layer of loose connective tissue and fat. Subcutaneous tissue varies in thickness across the body, additionally depending on the sex of the individual [38]. It insulates the body, serves as a reserve energy supply and cushions the skin [15].

2.2 The Extracellular Matrix

The extracellular matrix is often referred to as the “ground substance” of the dermis. It is an intricate structure of different molecules that fills the space between skin cells and provides structural integrity, elasticity and mechanical strength to the tissue. Figure 2 provides a schematic visualisation of the ECM in healthy human skin. The following subsections aim to give a detailed overview of ECM anatomy and physiology, with special emphasis on ECM remodelling.

2.2.1 Composition of the ECM

The major constituent of the ECM is *collagen*, a protein making up 70% of the dermis [16]. The periodically banded collagen fibers form a network that provides tensile strength and structural support [39]. Different types of collagen are present, the most abundant ones being type I, III, and V [37].

Another central component of the ECM is *elastin*, a protein that provides elasticity to the skin. Elastic fibres return the skin to its normal configuration after being stretched or deformed [15]. Elastin and collagen together maintain skin’s firmness and flexibility.

Additionally, the ECM contains *proteoglycans* (PGs) and *glycosaminoglycans* (GAGs). These are large molecules that are interspersed between and stick to collagen fibers in the ECM [39]. An example of a GAG is hyaluronic acid. PGs and GAGs are able to bind water molecules, providing a gel-like milieu and thus regulating the hydration of the skin. They effectively ensure skin’s plumpness and smoothness [37].

Glycoproteins like *fibronectin* and *laminin* are adhesive molecules that promote cell attachment to the ECM. One of the functions of the ECM is to act as a scaffold for the skin cells, enabling them to adhere and migrate through the matrix. Fibronectin and laminin provide these anchor points.

The ECM is a reservoir for *signalling molecules*, such as *growth factors*, *cytokines*, and *hormones*, that are involved in cell communication. Skin cells have receptors on their surface that are able to recognise specific signalling molecules. When one of the latter binds, it triggers a certain response from the cell (e.g., migration or the secretion of a specific molecule) [2]. Growth factors, a specific kind of signalling molecules, regulate the growth, proliferation, and differentiation of cells [2].

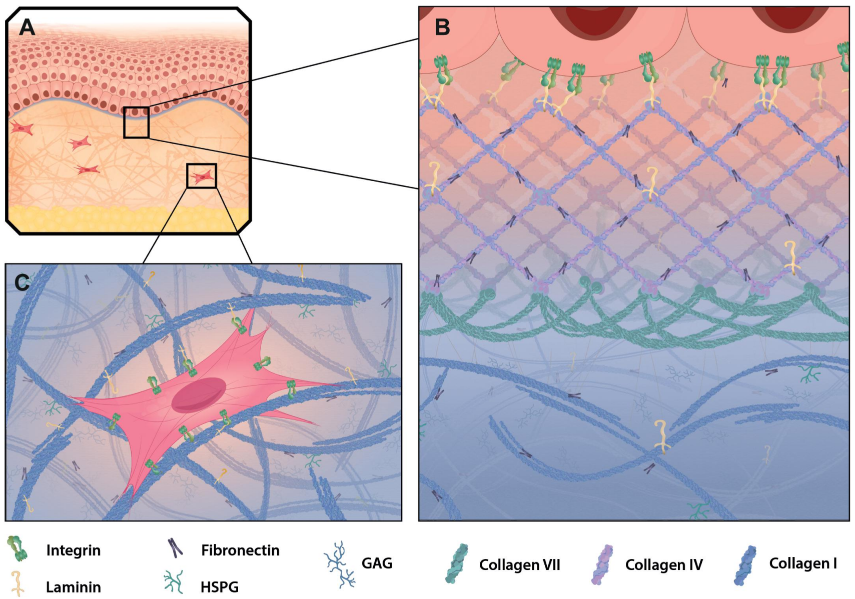


Figure 2: ECM in healthy human skin. The three skin layers are clearly visible (A). The ECM is a complex network of molecules that surrounds and supports the cells (B). It has attachment sites for the skin cells, allowing for cell adhesion and migration through the matrix (C). Taken from [39].

2.2.2 ECM Remodelling

The ECM is a highly dynamic structure that is constantly being remodelled. This entails a process in which ECM components are synthesised, deposited, degraded, and reorganised [34]. *Fibroblasts* and *myofibroblasts* are key cellular players in ECM remodelling.

As discussed in section 2.1, fibroblasts are the most common type of cells in the dermis. They are responsible for synthesising and depositing the components that constitute the extracellular matrix, as detailed in subsection 2.2.1 [16]. Fibroblasts can differentiate into myofibroblasts, resulting into a specialised skin cell that has contractile properties, resembling both fibroblasts and smooth muscle cells [28]. Myofibroblasts are also involved in producing constituents of the ECM.

An important instance in which ECM remodelling occurs is during wound healing and tissue repair. In the case of a dermal wound, part of the ECM is completely destroyed and requires reconstruction [14]. Here, fibroblast-to-myofibroblast differentiation occurs. Myofibroblasts play an important role during wound healing. Through their contractile properties, they exert mechanical forces on the ECM, contributing to its realignment and restructuring [28]. This additionally leads to wound *contraction*: the edges of the wound are drawn together, facilitating the healing process [28]. The subsequent section will delve more into the complex process of wound healing.

2.3 The Phases of Wound Healing

Wounds to the skin can be categorised in a number of different ways. One distinction is between epidermal and dermal wounds. In this work, the focus is on the latter, where we assume that the wound is as deep as to affect the dermis, where the ECM is damaged or destroyed.

Dermal wound healing is a complex sequence of overlapping events which are often described separately but in reality form a continuum referred to as *the healing cascade* [7]. For the ease of explanation, we shall also make the distinction of four separate phases: *hemostasis*, *inflammation*, *proliferation* and *remodelling*. The following subsections serve to explain the current biological understanding of each healing phase in more detail. Please refer to Figure 3 for a schematic overview.

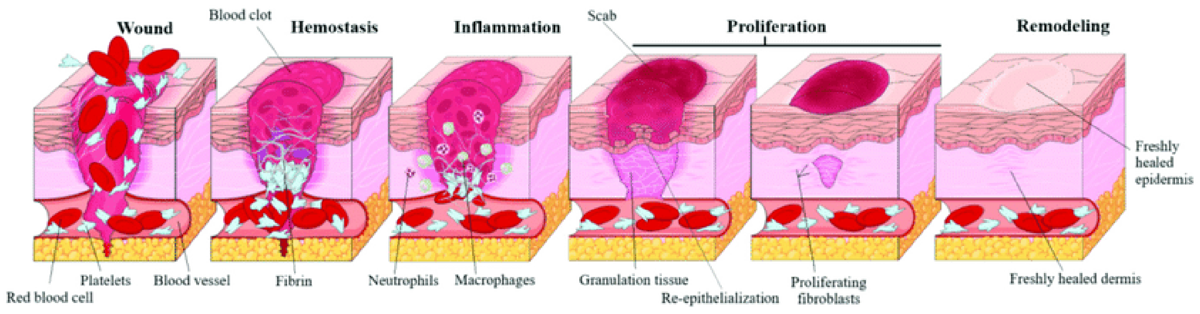


Figure 3: Healing cascade phases of dermal wounds. Taken from [18].

2.3.1 Hemostasis

The moment healthy tissue is injured, the healing response commences. Blood vessels in the area constrict to reduce blood flow [17]. As the blood components spill into the site of the injury, the platelets come into contact with exposed collagen and other elements of the ECM [7]. This triggers the platelets to release clotting factors. They aggregate at the site of injury, forming a blood clot to stop the bleeding. The fibrin clot acts as a temporary barrier and releases signalling molecules to initiate healing [5]. It also serves as a temporary matrix through which cells can migrate [17].

2.3.2 Inflammation

Hemostasis is followed by the inflammatory phase, which is characterised by the removal of pathogens. Now, the blood vessels near the injury dilate and become more permeable. This promotes the transportation of leukocytes (*neutrophils* and *monocytes*), which adhere to the blood vessel walls and migrate out of the bloodstream, into the tissue [24].

Neutrophils are the first to arrive at the site of the injury to combat pathogens. The major function of the neutrophil is to remove foreign material, bacteria, non-functional host cells, and damaged matrix components that may be present in the wound site [7]. Neutrophils have a relatively short lifespan.

Monocytes differentiate into *macrophages*, which are very potent immune cells that engulf and kill pathogens [36]. They are also responsible for removing nonfunctional host cells, bacteria-filled neutrophils, damaged ECM, foreign debris, and any remaining bacteria from the wound site [7].

Macrophages release certain signalling molecules, such as growth factors and cytokines, to recruit even more immune cells to the site of injury [17]. The inflammatory response causes redness, swelling and wamth in the wounded area. The presence of wound macrophages is a marker that the inflammatory phase is nearing an end and that the proliferative phase is beginning [7].

2.3.3 Proliferation

Once the wound site is cleaned out, fibroblasts from the surrounding undamaged skin migrate in and proliferate. They synthesise and deposit new ECM components, including collagen, to rebuild the tissue framework [17].

Another predominant cell type proliferating during this phase is the *endothelial cell* (i.e., cells that form the walls of blood vessels). New blood vessels are formed through a process called *angiogenesis*, supplying oxygen and nutrients to support healing [7].

Initially, granulation tissue is formed, consisting of proliferating fibroblasts, newly formed blood vessels and loose ECM. This is temporary tissue that is later substituted for the real ECM [39]. The granulation tissue fills the wound from the base up, gradually filling the wound defect [14].

Signalling molecules induce fibroblasts already located in the wound site to transform into myofibroblasts [17]. They exhibit less proliferation compared to the fibroblasts coming in from the wound periphery. Myofibroblasts are also responsible for producing constituents of the ECM [28]. As myofibroblasts are able to exert large contractile forces, their presence effectively turns the granulation tissue into a temporary contractile organ [35]. This pulls the wound edges toward the center, which results in a gradual reduction of the wound area [31].

During the proliferative phase, *epithelisation* occurs: epithelial cells (keratinocytes) from the wound edges start to migrate and proliferate across the wound periphery. They gradually cover the granulation tissue and in doing so effectively close the wound [31].

2.3.4 Remodelling

The remodelling phase is the longest healing phase and can take up to a year [31]. Clinically, this is perhaps the most important phase, as granulation tissue becomes mature scar tissue over this time [15]. It is characterised by collagen remodelling and further contraction [31].

In the healing wound, granulation tissue is initially comprised of large amounts of type III collagen. During this phase, fibroblasts gradually change the type III collagen to collagen type I [15]. This leads to increased tensile strength of the scar. However, scar tissue will always remain weaker, as the final tensile strength is about 80% of that of surrounding healthy skin [24].

The collagen fibres additionally undergo some reorganisation. The collagen that is initially laid down is thinner than that in uninjured skin and is orientated parallel to the skin (instead of the basket weave pattern seen in uninjured skin) [17]. Over time, the collagen fibers are reabsorbed and deposited thicker, rearranged and cross-linked, such that they align along mechanical tension lines [31]. The latter contrasts with the random alignment of collagen fibres in healthy ECM [24].

As granulation tissue matures into scar tissue, the cell densities decrease. Many of the cells undergo *apoptosis* (i.e. programmed cell death). This leaves a relatively acellular and avascular, flat and thin scar of gradually increasing strength [30]. The extracellular matrix has been successfully restored, but with slightly different properties than the pre-injured ECM.

2.4 Burn Injuries to the Skin

A burn is an injury to the skin usually caused by heat. Since burns are a type of wound, they heal in a similar way as described in section 2.3. However, one distinction between general wound healing is that in the case of a burn, hemostasis is often bypassed. The reason for this is that intense heat can cause coagulation and destruction of blood vessels, leading to reduced or almost immediate cessation of bleeding [19]. As a result, the hemostasis phase is not as prominent or may not occur at all in many burn injuries. That is why, when a burn occurs, the body's response is to initiate the inflammatory phase of wound healing directly [11].

Severe burns can lead to a significant decrease in mobility in the affected area over the long term, primarily due to the development of *contractures* and *hypertrophic scarring.* As detailed in section 2.3, contraction is a natural response of the body, facilitating healing. However, when excessive, contractions may become pathological [7]. If long-term reduced mobility occurs, it is commonly named a contracture

[11]. The degree of contracture severity is influenced by factors such as the size of the wound, its location on the body and the extent of the skin tightening [44]. Contractures can inflict significant pain and discomfort on the patient and may lead to lifelong disabilities that can profoundly impact their future.

Another complication often arising in severe burn injuries is hypertrophic scarring. This pathological condition is characterised by a stiff, raised and uneven-textured scar, that does not extend far from the edges of the original wound [24]. Hypertrophic scarring is due to excessive healing, where an excess of ECM is produced and deposited [7]. In wound healing that leads to pathological scars, the inflammatory response is often greater and continues for an unusually long period of time. During the proliferative phase, fibroblasts and myofibroblasts continue to proliferate and synthesise ECM components much longer than usual, possibly due to this prolonged inflammation [24]. This leads to hypertrophic scarring, which can inflict pain, discomfort, and itch on the patient. Hypertrophic scars can also restrict movement if they are located close to a joint [14].

3 The Classical Theory of Elastic Deformation

This chapter introduces the well-established and widely understood principles of classical elasticity. It provides a solid foundation for understanding the fundamental concepts of deformation, stress, and strain. This background knowledge will serve as a framework for grasping the concept of *morphoelasticity*, introduced in the subsequent chapter 4. This chapter is largely based on the books 'An Introduction to Continuum Mechanics' by J.N.Reddy [42] and 'Continuum Mechanics' by A. J. M. Spencer [46].

Elasticity is the material property of a solid body to undergo deformation when subjected to an external force and to return to its original shape once the force is removed. This is in contrast to *plasticity*, in which the body is unable to withstand temporary changes to its shape and undergoes permanent deformation. Elasticity theory is concerned with the deformation of bodies under the influence of applied forces and, in particular, with the stresses and strains which result from deformations.

3.1 Descriptions of Motion

We consider a body $\mathcal{B}$ that occupies the region of space $\mathcal{R}_0 \subseteq \mathbb{R}^3$ at $t = 0$. $\mathcal{R}_0$ is called the *reference configuration*. We may view $\mathcal{B}$ as a set of particles, having a continuous distribution of matter in space and time. This is why we refer to $\mathcal{B}$ as a *continuous body*. After a time t, $\mathcal{B}$ has moved and deformed, such that it now occupies the region $\mathcal{R}_t \subseteq \mathbb{R}^3$, called the *current* or *deformed configuration*. Figure 4 displays a schematic visualisation.

The motion of $\mathcal{B}$ may be described by a mapping of the form

$$\mathbf{f} : (\mathcal{R}_0, \mathbb{R}_+) \to \mathcal{R}_t.$$

Thus, given a particle in $\mathcal{B}$ that is initially at the point with position vector $\mathbf{X}$, we can find the position vector $\mathbf{x}$ at time t that corresponds to the same particle by applying $\mathbf{f}$:

$$\mathbf{x} = \mathbf{f}(\mathbf{X}, t). \tag{1}$$

Equation (1) is referred to as the *material* or *Lagrangian description*. We assume that the deformation of $\mathcal{B}$ is continuous, so that $\mathbf{f}$ is a continuous and bijective function for any t. Hence, the inverse function exists, and we may also write

$$\mathbf{X} = \mathbf{f}^{-1}(\mathbf{x}, t). \tag{2}$$

The relationship in equation (2) is called the *spatial* or *Eulerian description*.

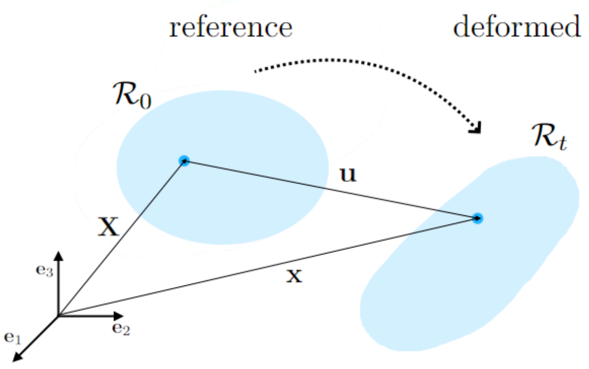


Figure 4: Reference and current configurations of the body $\mathcal{B}$.

The *displacement vector* $\mathbf{u}$ of a particle from its position X in the reference configuration to its position vector x at time t is given by

$$\mathbf{u} = \mathbf{x} - \mathbf{X}.$$

In Lagrangian description, $\mathbf{u}$ is considered a function of $\mathbf{X}$ and t, so that

$$\mathbf{u}(\mathbf{X}, t) = \mathbf{f}(\mathbf{X}, t) - \mathbf{X}.$$

In Eulerian description, however, $\mathbf{u}$ is a function of $\mathbf{x}$ and t:

$$\mathbf{u}(\mathbf{x}, t) = \mathbf{x} - \mathbf{f}^{-1}(\mathbf{x}, t).$$

A *rigid-body motion* is one in which all material particles of the body $\mathcal{B}$ undergo the same linear and angular displacements. However, a deformable body is one in which the material particles can move relative to each other. In the former case, the deformation may be determined only by considering the change of distance between any two arbitrary but infinitesimally close points of the continuum.

3.2 Deformation Gradient Tensor

The *deformation gradient tensor* $\mathbf{F}$ is the fundamental measure of deformation in continuum mechanics. It is the second-order tensor which maps infinitesimal line elements in the reference configuration to line elements in the current configuration, consisting of the same material particles. Figure 5 provides a schematic visualisation. It is defined by the following relation:

$$d\mathbf{x} = \mathbf{F}\, d\mathbf{X}, \tag{3}$$

where $\mathbf{F} = \nabla \mathbf{f}(\mathbf{X}, t) = \frac{\partial \mathbf{x}}{\partial \mathbf{X}}$ is the deformation gradient tensor. For continuous deformations, $\mathbf{F}$ is nonsingular, such that we may also write:

$$d\mathbf{X} = \mathbf{F}^{-1}\, d\mathbf{x},$$

where $\mathbf{F}^{-1} = \nabla \mathbf{f}^{-1}(\mathbf{x}, t) = \frac{\partial \mathbf{X}}{\partial \mathbf{x}}$ is the inverse deformation tensor.

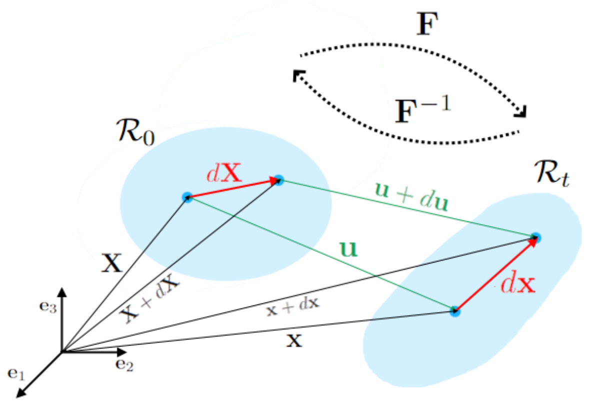


Figure 5: The deformation gradient tensor acting on a line segment.

The deformation gradient tensor $\mathbf{F}$ can be expressed in terms of the displacement vector $\mathbf{u}$ in the following manner:

$$\mathbf{F} = \nabla \mathbf{u} + \mathbf{I}. \tag{4}$$

Similarly, for $\mathbf{F}^{-1}$ we have the following relation:

$$\mathbf{F}^{-1} = \mathbf{I} - \nabla \mathbf{u}.$$

According to the polar decomposition theorem, the invertible second-order tensor $\mathbf{F}$ may be uniquely decomposed into a product of two second-order tensors:

$$\mathbf{F} = \mathbf{R}\,\mathbf{U} = \mathbf{V}\,\mathbf{R}. \tag{5}$$

Here $\mathbf{R}$ is proper orthogonal (i.e. $\mathbf{R}^T = \mathbf{R}^{-1}$ and $\det \mathbf{R} = 1$) and represents a rigid rotation. $\mathbf{U}$ and $\mathbf{V}$ are called the *right* and *left stretch tensor*, respectively. Their names are partly an indication of their position relative to the *rotation tensor* $\mathbf{R}$. The tensors $\mathbf{U}$ and $\mathbf{V}$ are symmetric positive definite.

Combining Equations (3) and (5) results into the following expression:

$$d\mathbf{x} = (\mathbf{RU})\, d\mathbf{X} = (\mathbf{VR})\, d\mathbf{X}.$$

The polar decomposition implies that the deformation of a line element $d\mathbf{X}$ in the reference configuration onto $d\mathbf{x}$ in the deformed configuration can be decomposed in two ways. Either a pure stretch deformation is applied (i.e. $d\mathbf{x}' = U d\mathbf{X}$) followed by a rotation (i.e. $d\mathbf{x} = \mathbf{R} d\mathbf{x}'$) or equivalently, by applying a rigid rotation first (i.e. $d\mathbf{x}' = \mathbf{R} d\mathbf{X}$), followed by a stretching (i.e. $\mathbf{dx} = \mathbf{V} d\mathbf{x}'$). Please refer to Figure 6 for a more visual representation.

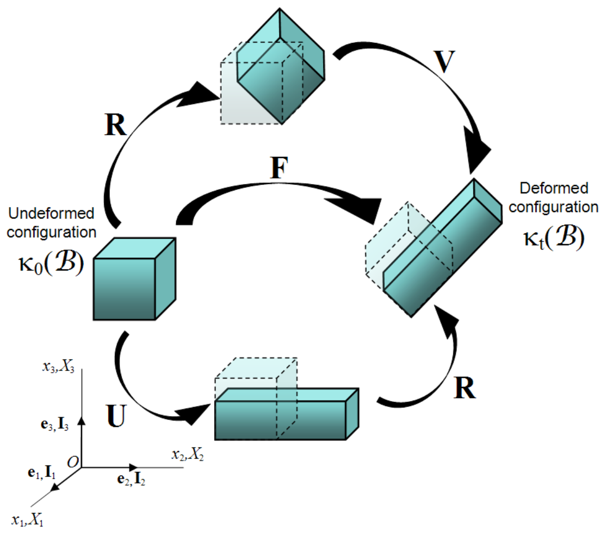


Figure 6: Representation of the polar decomposition of the deformation gradient. Taken from [49].

[I need to make a similar picture as above but then in the same style as the others I used and made before.]

3.3 Cauchy-Green Deformation Tensors

The deformation gradient tensor is not rotation-independent, as it accounts for both stretching and shearing effects, as well as rotations. In continuum mechanics, it is often convenient to use rotation-independent measures of deformation. As a rotation followed by its inverse rotation leads to no change ($\mathbf{R}\mathbf{R}^T = \mathbf{R}^T\mathbf{R} = \mathbf{I}$) we can exclude the rotation by multiplying $\mathbf{F}$ by its transpose:

$$\mathbf{C} = \mathbf{F}^T\,\mathbf{F}. \tag{6}$$

This leads to the rotation-independent, symmetric second-order tensor $\mathbf{C}$, called the *right Cauchy–Green deformation tensor*.

Let $dS = ||d\mathbf{X}||$ be the initial *distance* between two material particles of the body $\mathcal{B}$ and let $ds = ||d\mathbf{s}||$ be the distance between the same particles at time t. Then there exists the following relation for the right Cauchy-Green tensor:

$$ds^2 = d\mathbf{X}^T\,\mathbf{C}\,d\mathbf{X}. \tag{7}$$

We thus observe that $\mathbf{C}$ can be interpreted as a tensorial description of the relationship between distance in the current configuration and displacement in the initial configuration.

Reversing the order of multiplication in Equation (6) leads to the *left Cauchy–Green deformation tensor*, which is defined as

$$\mathbf{B} = \mathbf{F}\,\mathbf{F}^T.$$

Like the right Cauchy-Green tensor, $\mathbf{B}$ is symmetric. The relation with distance is the following:

$$dS^2 = d\mathbf{x}^T\,\mathbf{B}^{-1}\,d\mathbf{x}. \tag{8}$$

This means that, given an infinitesimal vector $d\mathbf{x}$ between neighbouring particles in the current configuration, $\mathbf{B}^{-1}$ may be used to determine dS, the original distance between those particles.

In a rigid motion, the deformation gradient tensor $\mathbf{F}$ is proper orthogonal (i.e., $\mathbf{F}^T = \mathbf{F}^{-1}$). In this case, $\mathbf{B}$ and $\mathbf{C}$ are both equal to the identity tensor. This is in accordance with the physical interpretation of the Cauchy-Green tensors, as distances between any two material points remain unchanged by a rigid motion.

The left and right Cauchy-Green tensors are closely related to the left and right stretch tensors defined in Equation (5). Using this equation, we find that

$$\mathbf{C} = \mathbf{F}^T\,\mathbf{F} = \mathbf{U}^T\,\mathbf{R}^T\,\mathbf{R}\,\mathbf{U} = \mathbf{U}^2.$$

Similarly we have

$$\mathbf{B} = \mathbf{F}\,\mathbf{F}^T = \mathbf{V}\,\mathbf{R}\,\mathbf{R}^T\,\mathbf{V}^T = \mathbf{V}^2.$$

Since the stretch tensors are symmetric positive definite, it follows that the Cauchy-Green tensors are as well.

3.4 Strain

A useful measure of deformation is one with the property that a rigid motion leads to a deformation of zero. This leads to the concept of *strain*, used to evaluate how much a given displacement differs locally from a rigid body displacement. The *Lagrangian strain tensor* $\mathbf{E}^L$ is defined in the following manner:

$$(ds)^2 - (dS)^2 = 2\, d\mathbf{X}^T\, \mathbf{E}^L\, d\mathbf{X}. \tag{9}$$

Let us first consider the physical interpretation. Given two particles in the body $\mathcal{B}$, such that $d\mathbf{X}$ is the vector separating them in Lagrangian coordinates, the Lagrangian strain tensor allows us to calculate how much the line segment is stretched in the current configuration, relative to the initial configuration (as measured by $(ds)^2 - (dS)^2$). This implies that the strain tensors enables the degree of stretching to be evaluated, for any direction of the vector $d\mathbf{X}$.

Similarly, the *Eulerian strain tensor* $\mathbf{E}^E$ is defined as

$$(ds)^2 - (dS)^2 = 2\, d\mathbf{x}^T\, \mathbf{E}^E\, d\mathbf{x}. \tag{10}$$

Importantly, the strain tensors are equal to the zero tensor at a material point if and only if the body is completely unstretched in the neighbourhood of that point ($ds = dS$).

Equations (9) and (10) can respectively be combined with the definitions of the Cauchy-Green tensors given in Equations (7) and (8), to obtain more formal definitions for the strain tensors. This results into the following relations:

$$\mathbf{E}^L = \frac{1}{2}\left(\mathbf{C} - \mathbf{I}\right), \tag{11}$$

$$\mathbf{E}^E = \frac{1}{2}\left(\mathbf{I} - \mathbf{B}^{-1}\right). \tag{12}$$

The tensors $\mathbf{E}^L$ and $\mathbf{E}^E$ are relatively difficult to calculate and manipulate. Luckily, many solid bodies experience only small changes of shape when under the influence of forces of reasonable magnitudes. The consequence is that we can simplify Equations (11) and (12) in the case when all displacements are small. We then assume that the displacement gradient is small (i.e. $||\nabla\mathbf{u}|| << 1$). This enables us to perform a linearisation of the strain tensors, where higher order terms can be neglected. It turns out that the linearisations of the two strain tensors are the same, which implies that there is little difference in the material and spatial coordinates of a given material point in the continuum (under our assumption). This gives rise to the following expressions:

$$\mathbf{E}^L \simeq \mathbf{E}^E \simeq \boldsymbol{\varepsilon} = \frac{1}{2}\left[\nabla\mathbf{u} + (\nabla\mathbf{u})^T\right].$$

Here $\boldsymbol{\varepsilon}$ is called the *infinitesimal strain tensor.* Using Equation (4), $\boldsymbol{\varepsilon}$ can also be expressed in terms of the deformation gradient tensor $\mathbf{F}$:

$$\boldsymbol{\varepsilon} = \frac{1}{2}\left(\mathbf{F} + \mathbf{F}^T\right) - \mathbf{I}.$$

We must keep in mind that the infinitesimal strain tensor is only a valid representation of strain when all deformations are small. On top of that, the infinitesimal strain tensor is sensitive to rotations. It is therefore not a true measure of deformation, since it is not invariant to all rigid motion.

3.5 Stress

In continuum mechanics, *stress* is a physical quantity that describes forces that are present during the deformation of a body $\mathcal{B}$. It expresses the internal forces that neighbouring material particles exert on each other. These macroscopic forces are in truth the net result of a very large number of intermolecular forces and collisions between the particles in these molecules.

Stress is effectively a measure of the internal forces exerted over an arbitrary internal area element and has units of force per area. If δA is an ifinitesimal area element of the body $\mathcal{B}$ with normal vector $\hat{\mathbf{n}}$ and $\mathbf{F}$ is the total force acting over that surface element, than the *stress vector* $\mathbf{T}(\hat{\mathbf{n}})$ satisfied the following relation:

$$\mathbf{T}(\hat{\mathbf{n}}) = \lim_{\delta A \to 0} \frac{\delta\mathbf{F}(\hat{\mathbf{n}})}{\delta A} = \frac{d\mathbf{F}}{dA}.$$

The stress vector depends on its location in the body, as well as the orientation of the plane on which it is acting, as defined by its normal vector $\hat{\mathbf{n}}$. The component of $\mathbf{T}$ that is in the direction of $\hat{\mathbf{n}}$ is called *normal stress*, whereas the component that is normal to $\hat{\mathbf{n}}$ is called *shear stress.*

If we are interested in the state of stress at a point in $\mathcal{B}$, we must consider all stress vectors $\mathbf{T}(\hat{\mathbf{n}})$ associated with all planes that pass through that point. However, according to *Cauchy's stress theorem*, it is enough to know the stress vectors on three mutually perpendicular planes. The stress vector on any other plane passing through that point can then be found through a coordinate transformation.

The Cauchy's stress theorem states that there exists a second order tensor $\boldsymbol{\sigma}$, independent of $\hat{\mathbf{n}}$, such that the following relationship exists:

$$\mathbf{T}(\hat{\mathbf{n}}) = \hat{\mathbf{n}} \cdot \boldsymbol{\sigma}. \tag{13}$$

$\boldsymbol{\sigma}$ is known as the *Chauchy stress tensor*. Equation (13) implies that the stress vector at any point in $\mathcal{B}$ can be expressed in terms of the components of $\boldsymbol{\sigma}$: the stress vectors on the planes perpendicular to the coordinate axes, see Figure 7. This completely defines the state of stress at this point.

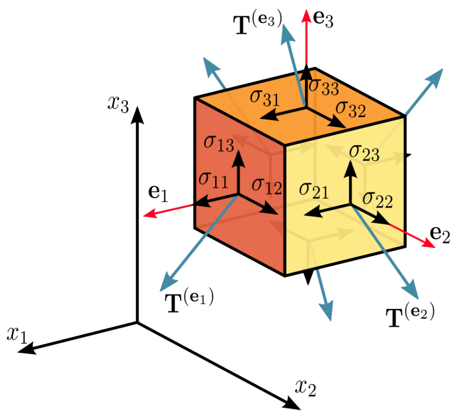


Figure 7: Components of stress in three dimensions. Taken from [48].

3.6 Constitutive Equations

In an elastic solid, the presence of a stress will generally lead to a strain. This means that the body will become stretched or decompressed in response to the internal forces it experiences. The relationship between stress and strain, dependent on the material taken into consideration, is defined by the *constitutive law* for the material.

The simplest example of a constitutive law is *Hooke's law for isotropic solids*. This relation is given (in component form) by

$$\sigma_{ij} = \lambda \delta_{ij} \sum_k \varepsilon_{kk} + 2\mu \varepsilon_{ij}. \tag{14}$$

Here $\boldsymbol{\sigma}$ is the stress tensor, $\boldsymbol{\varepsilon}$ is the strain tensor, $\boldsymbol{\delta}$ is the Kronecker delta, and λ and μ are the *Lamé coefficients*. The latter are characteristic of the material considered.

An *isotropic* material is one for which every material property at a point is the same in all directions. In other words, the material possesses no preferred orientation. Example of isotropic solids are metals in their usual polycrystalline form. Equation (14) is appropriate for describing the deformation of a simple, isotropic, perfectly elastic solid (e.g., steel under small deformations). However, more complex material requires more complex constitutive equations.

Anisotropic materials have mechanical properties that vary with direction (e.g., wood). There is an extension of Hooke's Law, which relates stress and strain in a more general form, taking into account anisotropic materials. The constitutive equation is given by

$$\boldsymbol{\sigma} = \mathbf{C}\boldsymbol{\varepsilon}, \tag{15}$$

and is known by the *generalised Hooke's law*. Here $\mathbf{C}$ is a fourth-order tensor known as the *elasticity* or *stiffness tensor*, containing all the material's elastic constants, which describe the relationship between stress and strain in different directions. In an isotropic material, the stiffness tensor simplifies because the material's properties are the same in all directions, and Equation (15) can be reduced to equation (14).

Note that for Hooke's law, the stress-strain relationship is assumed to be linear. This is just one of many possible relationships that can exist between stress and strain in materials. For example, many elastic materials undergo plastic deformations when they are stressed beyond a certain limit, known as the *yield stress*. Once the yield stress is reached, the change in strain does not correspond anymore to the change in stress as described by Hooke's law. Plastic deformation does not return to the original shape upon unloading. This means that the stress-strain relationship has changed.

Biological materials in particular display complicated plastic behaviour. Not only can plasticity occur in response to mechanical stress, the action of cells can directly modify the structure of a tissue. This is why we need a more complicated constitutive law for tissue elasticity.

4 Morphoelasticity and the Zero Stress State

Growth is the process by which a body increases in size through the addition of mass. In biological systems, growth may occur in many different forms. In skin healing, soft tissue growth occurs, changing the fundamental mechanical structure of tissue. Continuum mechanics and nonlinear elasticity provide a natural framework to study growth. The basic idea is that in a biological growth process, the deformation of a body can be due to both a change of mass, and an elastic response. This concept is known as *morphoelasticity* and it forms the backbone of the mathematical model for burn injuries, presented in chapter 5.

This chapter serves to explain the concept of morphoelasticity. Section 4.1 firstly explains the importance of taking residual stresses into account when considering tissue growth. Section 4.2 then explains the concept of the zero stress state and lastly, section 4.3 connects this to the concept of morphoelasticity.

4.1 Residual Stresses

Additionally to exhibiting plastic behaviour, many biological materials experience *residual stress* [24]. Residual stresses remain in a solid material after the original cause of the stresses has been removed. It occurs when a body is bound to itself in such a way that it cannot deform back to relieve its internal stresses.

The classical example of residual stress in biology was first described by Chuong and Fung [6], where they show that cylindrical arteries experience residual stress. After a longitudinal cut, the artery deforms out of its cylindrical shape and the residual stresses are relieved.

When determining how a body responds to an applied force, residual stresses must be taken into consideration as well. Soft tissue growth and dermal wound healing are examples of biological processes that involve changes to the fundamental mechanical structure of tissue, which possibly leads to residual stresses [24]. In order to account for that, Rodriguez *et al.* [43] proposed a theory for describing soft tissue growth, based on the concept of a *zero stress state*. The subsequent section considers this concept into more detail.

4.2 The Zero Stress State

The zero stress state is a locally-defined representation of the deformation of a solid body, required to relieve all elastic stresses at a given point on the interior. It is a hypothetical configuration in which the body "would like to be". Mathematically, the zero stress state corresponds to a multiplicative decomposition of the deformation gradient tensor $\mathbf{F}$ [43]. Given a particle in $\mathcal{B}$, if $\mathbf{X}$, $\mathbf{x}$, and $\boldsymbol{\chi}$ are position vectors in the initial, current and zero stress state configurations, respectively, then this gives the following decomposition:

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \mathbf{X}} = \frac{\partial \mathbf{x}}{\partial \boldsymbol{\chi}} \frac{\partial \boldsymbol{\chi}}{\partial \mathbf{X}} = \mathbf{F}_e \mathbf{F}_g. \tag{16}$$

Here $\mathbf{F}_g$ is a second-order tensor representing growth (or shrinkage). It essentially maps infinitesimal vectors in the initial configuration of the body, to infinitesimal vectors in a local zero stress configuration. The elastic component $\mathbf{F}_e$ is a second-order tensor that maps infinitesimal vectors in the zero stress state configuration, to infinitesimal vectors in the current configuration. $\mathbf{F}_e$ represents deformation due to mechanical forces.

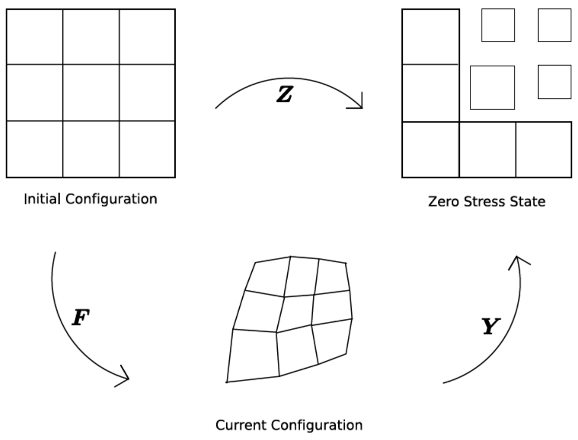


Figure 8: Rodriguez *et al.* [43] proposed that residual stress can be accounted for by decomposing the deformation from the initial to the current configuration as the product of an elastic and a growth deformation. Taken from [24]. Note that here $\mathbf{F}_g = \mathbf{Z}$ and $\mathbf{F}_e^{-1} = \mathbf{Y}$.

4.3 Morphoelasticity

The concept of *morphoelasticity* was first introduced by Goriely *et al.* [22], detailing the process of coupling an elastic model to a continually evolving zero stress state. The first theory of biological morphoelasticity was presented by Rodriguez *et al.* [43] and was later modified and expanded by Hall [24].

The fundamental assumption of morphoelasticity is the decoupling of the deformation gradient tensor as given in Equation (16). Physically, this decomposition can be interpreted as follows [13]. Biological growth is a map from a stress-free initial configuration to another (hypothetical) zero stress configuration. Due to the grown material, the latter will not fit in Euclidean space, as it will overlap or tear (have gaps). This means that $\mathbf{F_g}$ may break the continuity of the tissue. The purpose of the elastic part of the deformation $\mathbf{F}_e$ is then to restore the continuity by moving apart the overlapping regions (at the expense of introducing compressive stress) or filling the gaps (by introducing tensile stress). This means that the current configuration is no longer stress-free, even when it is unloaded, and will thus experiences residual stress. Figure 8 provides a schematic visualisation.

In his PhD thesis, Hall [24] expanded upon the concept of morphoelasticity, formulating how the zero stress state changes in time, in response to tissue growth and remodelling. Hall [24] derived several related evolution equations that mathematically describe the change of *effective strain* over time. In his work, effective strain is defined as a local measure for the difference between the current configuration of the tissue and a hypothetical configuration where the tissue is mechanically relaxed. Assuming that the effective strains are small, Hall additionally formulated an evolution equation that describes the dynamic change of the *infinitesimal effective strain* over time. His derivations are not straight-forward and rather lengthy, containing multiply subtleties. That is why we will not go into detail here and refer the interested reader to the PhD thesis of Hall [24]. The final derived equation that describes how the infinitesimal effective strain changes over time will be stated in the subsequent chapter, in Equation (19).

5 Morphoelastic Model for Burn Injuries

This chapter presents the mathematical model for skin evolution after burn trauma. The general mathematical framework is presented in section 5.1. A more precise description for the relevant biological constituents is formulated in sections 5.2 - 5.5. Section 5.6 treats the mechanical components present in the model. Lastly, the complete system is summarised in section 5.7.

5.1 Mathematical Framework

The general morphoelastic model for post-burn wound contraction was developed by Koppenol [30], who used the theory of morphoelasticity developed by Hall [24] to incorporate the formation of long term deformations (contraction) into the dermal layer of the skin.

The model considers four biological constituents and three mechanical components as the primary variables. The biological constituents are the fibroblasts (N), the myofibroblasts (M), a generic signalling molecule (c), and collagen (ρ). The mechanical components are the dermal layer displacement ($\mathbf{u}$), the the dermal layer displacement velocity ($\mathbf{v}$), and the effective strain ($\boldsymbol{\varepsilon}$). The following system of partial differential equations is used as a basis for the model:

$$\frac{\mathrm{D}z_i}{\mathrm{D}t} + z_i(\nabla\cdot\mathbf{v}) = -\nabla\cdot\mathbf{J}_i + R_i, \tag{17}$$

$$\rho_t\left(\frac{\mathrm{D}\mathbf{v}}{\mathrm{D}t} + \mathbf{v}(\nabla\cdot\mathbf{v})\right) = \nabla\cdot\boldsymbol{\sigma} + \mathbf{f}, \tag{18}$$

$$\frac{\mathrm{D}\boldsymbol{\varepsilon}}{\mathrm{D}t} + \boldsymbol{\varepsilon}\,\mathrm{skw}(\nabla\mathbf{v}) - \mathrm{skw}(\nabla\mathbf{v})\,\boldsymbol{\varepsilon} + (\mathrm{tr}\,(\boldsymbol{\varepsilon}) - 1)\,\mathrm{sym}(\nabla\mathbf{v}) = -\mathbf{G}. \tag{19}$$

Let us first explain some notation. We note that the operator $\frac{\mathrm{D}}{\mathrm{D}t}$ stands for the *material derivative*:

$$\frac{\mathrm{D}}{\mathrm{D}t} \equiv \frac{\partial}{\partial t} + \mathbf{v}\cdot\nabla.$$

If the material derivative is applied to a second-second tensor, then it is applied to each of the scalar elements of this tensor separately.

Secondly, any second-order tensor $\mathbf{L}$ may be written as the sum of a symmetric and skew-symmetric part in the following manner:

$$\mathbf{L} = \frac{1}{2}\left(\mathbf{L} + \mathbf{L}^T\right) + \frac{1}{2}\left(\mathbf{L} - \mathbf{L}^T\right) = \mathrm{sym}(\mathbf{L}) + \mathrm{skew}(\mathbf{L}).$$

Here, $\mathrm{sym}(\mathbf{L})$ is a second-order tensor with the property $\mathrm{sym}(\mathbf{L}) = \mathrm{sym}(\mathbf{L})^T$ and $\mathrm{skew}(\mathbf{L})$ is a second-order tensor with the property $\mathrm{skew}(\mathbf{L}) = -\mathrm{skew}(\mathbf{L})^T$.

Equation (17) is the conservation equation for the cell density / concentration for each of the four biological constituents. Here z_i represents the concentration, $\mathbf{J}_i$ is the flux per unit area, and R_i is a reaction term representing the kinetics of constituent i, for $i \in \{N, M, c, \rho\}$. A more precise expression for $\mathbf{J}_i$ and R_i for each of the constituents will be given in sections 5.2 to 5.5.

Equation (18) is the conservation equation for linear momentum. Here ρ_t represents the total mass density of the dermal tissue, $\boldsymbol{\sigma}$ is the stress tensor, and $\mathbf{f}$ is the total body force working on the dermal layer. Note that $\mathbf{v} = \frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t}$. The constitutive relation for $\boldsymbol{\sigma}$, as well as a more precise expression for $\mathbf{f}$ will be given in section 5.6. We note that Equation (18) actually gives rise to multiple equations, one for each component of the velocity vector $\mathbf{v}$. For example, in 3D this would result in three equations.

Lastly, Equation (19) is the evolution equation that describes how the infinitesimal effective strain ($\boldsymbol{\varepsilon}$) changes over time. It is this equation that captures the morphoelasticity of the dermal layer, taking into account permanent deformation (in this case contraction) and residual stresses. It was formulated by Hall [24] and is based on his extensive theory on the zero stress state and morphoelasticity. The second-order tensor $\mathbf{G}$ is a growth tensor that describes the rate of active change of the effective strain. It will be formulated in section 5.6. We note that Equation (19) actually gives rise to multiple equations, one for each component of the strain tensor $\boldsymbol{\varepsilon}$. For example, in 3D this would result in nine equations.

5.2 The Fibroblast Population

In order to simplify notation, we from now on replace z_i by i. Hence, z_N becomes N, the cell density of the fibroblasts in the dermis. Let us first describe the appropriate flux-term $\mathbf{J}_N$, that incorporates both random movement of fibroblasts through the dermal layer and the directed movement of fibroblasts up the gradient of signalling molecule c, if present. The former is modelled by a cell density dependent

Fickian diffusion, and the latter process is modelled using a simple model for chemotaxis [26]. Taken together this gives

$$\mathbf{J}_N = -D_F F \nabla N + \chi_F N \nabla c, \tag{20}$$

where $F = N + M$. Here D_F is the (myo)fibroblast diffusion parameter and χ_F is the chemotactic parameter.

Equation (17) also contains a reaction term R_N describing the kinetics of the fibroblasts. Three things are taken into consideration: proliferation, differentiation into myofibroblasts and apoptosis. The first is modelled using an adjusted logistic growth model. The presence of a signalling molecule c is assumed to enhance both proliferation and cell differentiation. We obtain the following expression:

$$R_N = r_F \left(1 + \frac{r_F^{\max} c}{a_c^I + c}\right)(1 - \kappa_F F) N^{1+q} - k_F c N - \delta_N N. \tag{21}$$

Here, the parameter r_F is the cell division rate, $r_F^{\max}$ is the maximum factor with which the cell division rate can be enhanced due to the presence of the signalling molecule, a_c^I is the concentration of the signalling molecule that causes the half-maximum enhancement of the cell division rate. $\kappa_F F$ represents the reduction in the cell division rate due to crowding, q is a fixed constant, k_F is the signalling molecule-dependent cell differentiation rate of fibroblasts into myofibroblasts, and δ_N is the apoptosis rate of fibroblasts.

5.3 The Myofibroblast Population

For the myofibroblasts, the flux-term in Equation (17) in very similar to the one for the fibroblasts. In the same way it accounts for random movement of myofibroblasts through the dermal layer and the directed movement of myofibroblasts up the gradient of signalling molecule c. We obtain:

$$\mathbf{J}_M = -D_F F \nabla M + \chi_F M \nabla c. \tag{22}$$

The reaction term describing the kinetics of myofibroblasts is also very similar. Almost the same adjusted logistic growth model as used for the fibroblast population is used, the only difference being the assumption that myofibroblasts solely divide when the generic signalling molecule is present. This gives us the following:

$$R_M = r_F \left(\frac{(1 + r_F^{\max}) c}{a_c^I + c}\right)(1 - \kappa_F F) M^{1+q} - k_F c M - \delta_M M, \tag{23}$$

where δ_M is the apoptosis rate of myofibroblasts.

5.4 The Generic Signalling Molecule

We assume that the signalling molecules diffuse through the dermis according to linear Fickian diffusion. This gives the following flux-term:

$$\mathbf{J}_c = -D_c \nabla c, \tag{24}$$

where D_c is the diffusion coefficient of the generic signalling molecule.

Furthermore, we assume that both fibroblasts and myofibroblasts release and consume the signalling molecules. Additionally, signalling molecules are removed from the dermis through proteolytic breakdown (breakdown of proteins into smaller components). The reaction term becomes the following:

$$R_c = \frac{k_c (N + \eta^I M) c}{a_c^{II} + c} - \delta_c \, g(N, M, c, \rho) c. \tag{25}$$

Here, k_c is the maximum net secretion rate of the signalling molecule, η^I is the ratio of myofibroblasts to fibroblasts in the maximum net secretion rate of the signalling molecules and the collagen molecules, a_c^{II} is the concentration of the signalling molecule that causes the half-maximum net secretion rate of the signalling molecule, and δ_c is the proteolytic breakdown rate of the signalling molecule.

The function $g(N, M, c, \rho)$ represents the concentration of a generic *metalloproteinase* (MMP). This enzyme is assumed to remove the signalling molecules through a proteolytic breakdown. In this study, we take the following relationship:

$$g(N, M, c, \rho) = \frac{(N + \eta^{II} M) \rho}{1 + a_c^{III} c}. \tag{26}$$

The parameter η^{II} is the ratio of myofibroblasts to fibroblasts in the secretion rate of the MMPs and the $1/(1 + a_c^{III} c)$ term represents the inhibition of the secretion of the MMPs due to the presence of the signalling molecule.

5.5 The Collagen Molecules

For collagen, we assume that there is no active transport in the dermis, as secreted collagen molecules are attached to the ECM instantly. This means that the flux-term in Equation (17) is zero:

$$\mathbf{J}_\rho = \mathbf{0}. \tag{27}$$

For the reaction term, three things are incorporated: collagen molecules are produced by both fibroblasts and myofibroblasts, the secretion rate is enhanced in the presence of the signalling molecule, and there is a proteolytic collagen breakdown analogous to the removal of the signalling molecules. This collectively results into

$$R_\rho = k_\rho \left(1 + \frac{k_\rho^{\max} c}{a_c^{IV} + c}\right)(N + \eta M) - \delta_\rho\, g(N, M, c, \rho)\rho. \tag{28}$$

Here, k_ρ is the collagen molecule secretion rate, $k_\rho^{\max}$ is the maximum factor with which the secretion rate can be enhanced due to the presence of the signalling molecule, a_c^{IV} is the concentration of the signalling molecule that causes the half-maximum enhancement of the secretion rate, and δ_ρ is the degradation rate of the collagen molecules.

5.6 The Mechanical Components

In Equation (18), a visco-elastic constitutive relation is used for the stress-strain relation in the dermal layer. The visco-elastic relation for the dermal stress is:

$$\boldsymbol{\sigma} = \mu_1 \mathrm{sym}(\nabla \mathbf{v}) + \mu_2[\mathrm{tr}(\mathrm{sym}(\nabla \mathbf{v}))\mathbf{I}] + \frac{E\sqrt{\rho}}{1+\nu}\left(\boldsymbol{\varepsilon} + \mathrm{tr}(\boldsymbol{\varepsilon})\frac{\nu}{1-2\nu}\mathbf{I}\right). \tag{29}$$

Here μ_1 and μ_2 are the shear and bulk viscosity, respectively, and ν is the Poisson's ratio. $E\sqrt{\rho}$ represents Young's modulus (stiffness), which we assume to be dependent on the concentration of the collagen molecules.

Additionally, the total body force $\mathbf{f}$ in Equation (18) needs a more precise description. We assume that the myofibroblasts generate an isotropic stress, due to their pulling on the ECM, which is proportional to the product of the cell density of the myofibroblasts and a simple function of the concentration of the collagen molecules:

$$\mathbf{f} = \nabla \cdot \boldsymbol{\psi}, \tag{30}$$

$$\boldsymbol{\psi} = \xi M \left(\frac{\rho}{R^2 + \rho^2}\right)\mathbf{I}. \tag{31}$$

$\boldsymbol{\psi}$ is a second-order tensor representing the total generated stress by the myofibroblast population, the parameter ξ is the generated stress per unit cell density and the inverse of the unit collagen concentration, and R is a fixed constant.

Lastly, we consider the growth contribution tensor $\mathbf{G}$ in Equation (19). We assume that the rate of active change of the effective strain is proportional to four things: the product of the amount of effective strain, the local concentration of the MMPs, the local concentration of the signalling molecule, and the inverse of the local concentration of the collagen molecules. Taken collectively, this results into the following symmetric tensor:

$$\mathbf{G} = \zeta\left(\frac{g(N, M, c, \rho)c}{\rho}\right)\boldsymbol{\varepsilon} = \xi\left(\frac{(N + \eta^{II}M)c}{1 + a_c^{III}c}\right)\boldsymbol{\varepsilon}, \tag{32}$$

where the parameter ξ is the rate of morphoelastic change.

5.7 The complete system of equations

Combining Equations (17) - (32) gives us the complete mathematical model that describes post-burn evolution of the dermal layer:

$$\begin{aligned}
\frac{\mathrm{D}N}{\mathrm{D}t} + N(\nabla \cdot \mathbf{v}) &= -\nabla \cdot (-D_F F \nabla N + \chi_F N \nabla c) + r_F\left(1 + \frac{r_F^{\max} c}{a_c^I + c}\right)(1 - \kappa_F F)N^{1+q} - k_F c N - \delta_N N,\\
\frac{\mathrm{D}M}{\mathrm{D}t} + M(\nabla \cdot \mathbf{v}) &= -\nabla \cdot (-D_F F \nabla M + \chi_F M \nabla c) + r_F\left(\frac{(1 + r_F^{\max})c}{a_c^I + c}\right)(1 - \kappa_F F)M^{1+q} - k_F c M - \delta_M M,\\
\frac{\mathrm{D}c}{\mathrm{D}t} + c(\nabla \cdot \mathbf{v}) &= -\nabla \cdot (-D_c \nabla c) + \frac{k_c(N + \eta^I M)c}{a_c^{II} + c} - \delta_c \frac{(N + \eta^{II}M)\rho}{1 + a_c^{III}c}c,\\
\frac{\mathrm{D}\rho}{\mathrm{D}t} + \rho(\nabla \cdot \mathbf{v}) &= k_\rho\left(1 + \frac{k_\rho^{\max} c}{a_c^{IV} + c}\right)(N + \eta M) - \delta_\rho \frac{(N + \eta^{II}M)\rho}{1 + a_c^{III}c}\rho,
\end{aligned} \tag{33}$$

$$\rho_t\left(\frac{\mathrm{D}\mathbf{v}}{\mathrm{D}t}+\mathbf{v}(\nabla\cdot\mathbf{v})\right)=\nabla\cdot\left(\mu_1\mathrm{sym}(\nabla\mathbf{v})+\mu_2[\mathrm{tr}(\mathrm{sym}(\nabla\mathbf{v}))\mathbf{I}]+\frac{E\sqrt{\rho}}{1+\nu}\left(\boldsymbol{\varepsilon}+\mathrm{tr}(\boldsymbol{\varepsilon})\frac{\nu}{1-2\nu}\mathbf{I}\right)\right)+\nabla\cdot\left(\xi M\left(\frac{\rho}{R^2+\rho^2}\right)\mathbf{I}\right),$$

$$\frac{\mathrm{D}\boldsymbol{\varepsilon}}{\mathrm{D}t}+\boldsymbol{\varepsilon}\,\mathrm{skw}(\nabla\mathbf{v})-\mathrm{skw}(\nabla\mathbf{v})\,\boldsymbol{\varepsilon}+(\mathrm{tr}\,(\boldsymbol{\varepsilon})-1)\,\mathrm{sym}(\nabla\mathbf{v})=-\xi\left(\frac{(N+\eta^{II}M)c}{1+a_c^{III}c}\right)\boldsymbol{\varepsilon}.$$

6 Machine Learning Background

The function of this chapter is twofold: firstly, it serves to give an introduction into machine learning and specifically *artificial neural networks*. Section 6.1 gives a comprehensive overview of neural networks, covering all relevant concepts and terms. This section is based on the following sources: [21], [3], and [25]. Additionally, this chapter introduces specific neural network types that are of interest in this study. Section 6.2 gives a description of *Physics-Informed Neural Networks* (PINNs), section 6.3 introduces *Deep Operator Networks* (DeepONets) and the combination of the two, *physics-informed DeepONets* is discussed is section 6.4.

6.1 A Brief Overview of Neural Networks

This section serves to give an introduction into neural networks. Subsection 6.1.1 gives a general description and considers the general architecture. Subsection 6.1.2 covers activation functions used to introduce non-linearity in the network. Training of the network is explained in subsection 6.1.3. This is followed by the explanation of forward propagation and backpropagation in subsections 6.1.4 and 6.1.5, respectively. Different optimisation algorithms are considered in subsection 6.1.6. Lastly, the testing and validation of a network is discussed in subsection 6.1.7.

6.1.1 General Architecture and Description

Neural networks are computational models designed to mimic the brain's learning process. They consist of interconnected nodes, or *neurons*, each performing simple computations [21]. The neurons are organised in layers, including an *input layer*, one or more *hidden layers*, and an *output layer*. A hidden layer is called *dense* or *fully-connected* if each neuron in the layer is connected to all neurons in the subsequent layer. The number of layers in the network is called the *depth*. That is why the term *deep learning* or *deep neural network* refers to networks with multiple hidden layers. The *width* of the network is defined as the number of neurons in the hidden layers. Figure 9 provides a visualisation of a fully-connected, deep neural network.

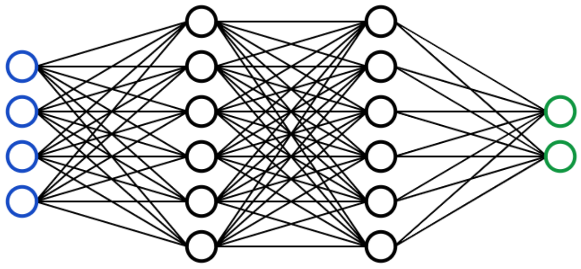


Figure 9: A visualisation of a fully-connected, deep neural network. Input layer in blue, two hidden layers in black and output layer in green. Taken from [50].

A neural network is used to approximate a mapping $\mathbf{f}$ that takes as input a vector $\mathbf{x} \in \mathbb{R}^n$ and outputs a vector $\mathbf{y} \in \mathbb{R}^m$, such that $\mathbf{y} = \mathbf{f}(\mathbf{x})$. It has been shown that neural networks are universal function approximators. This means that, with enough hidden layers, a neural network is able to approximate any (nonlinear) continuous function $\mathbf{f}$ arbitrarily well [21]. A neural network is essentially a parametrised mapping $\hat{\mathbf{f}}$ that takes as input $\mathbf{x} \in \mathbb{R}^n$ and outputs $\hat{\mathbf{y}} \in \mathbb{R}^m$. It tries to learn the values of certain parameters $\boldsymbol{\theta}$, such that $\hat{\mathbf{f}}(\mathbf{x};\boldsymbol{\theta}) \approx \mathbf{f}(\mathbf{x})$, i.e., $\hat{\mathbf{y}} \approx \mathbf{y}$.

In order to approximate both linear and nonlinear mappings, a neural network uses a combination of an affine transformation and a nonlinear *activation function*. Let us assume we have a fully-connected network with L hidden layers. Each neuron in the first hidden layer takes as input the whole vector $\mathbf{x}$ and applies the affine transformation $\mathbf{g}(\mathbf{x},\boldsymbol{\theta}) = \mathbf{g}(\mathbf{x},\mathbf{w},b) = \mathbf{w}^T\mathbf{x} + b$. Here $\mathbf{w} \in \mathbb{R}^n$ is a vector of the *weights* of the neuron and $b \in \mathbb{R}$ is the *bias*. If we combine these weights and biases in a matrix $W_1 \in \mathbb{R}^{n_1 \times n}$ and $\mathbf{b}_1 \in \mathbb{R}^{n_1}$ respectively for all n_1 neurons in this hidden layer, the transformed vector $\hat{\mathbf{x}}$ can be computed as

$$\hat{\mathbf{x}} = W_1\mathbf{x} + \mathbf{b}_1.$$

The transformed vector is subsequently passed through a fixed nonlinear activation function α to obtain the hidden values of the first layer:

$$\mathbf{h}_1 = \alpha(\hat{\mathbf{x}}) = \alpha(W_1\mathbf{x} + \mathbf{b}_1).$$

Note that the activation function is applied element-wise to the vector $\hat{\mathbf{x}}$, but itself is still a scalar function. There are many choices for an activation function. They will be discussed in greater detail in subsection 6.1.2.

The output $\mathbf{h}_1 \in \mathbb{R}^{n_1}$ of the first hidden layer is taken as input of each node in the second hidden layer and the same process is repeated. This continues through all layers of the network. To summarise, a neural network with L hidden layers can be written as

$$\mathbf{h}_1 = \alpha(W_1\mathbf{x} + \mathbf{b}_1), \tag{34}$$
$$\mathbf{h}_{i+1} = \alpha(W_{i+1}\mathbf{h}_i + \mathbf{b}_{i+1}), \quad \text{for } i = 1, \dots, L-1 \tag{35}$$
$$\hat{\mathbf{y}} = \beta(W_{L+1}\mathbf{h}_L). \tag{36}$$

Here $\hat{\mathbf{y}} \in \mathbb{R}^m$ is the output of the neural network. Note that for the output layer there is often a different activation function used, denoted by β. One may also add a bias vector to the last layer. The vectors $\mathbf{h}_i \in \mathbb{R}^{n_i}$ for $i = 1, \dots L$ are the respective outputs of the hidden layers, which can be thought of as the intermediate states of the network. The matrices $W_i \in \mathbb{R}^{n_i \times n_{i-1}}$ and vectors $\mathbf{b}_i$, for $i = 1, \dots L+1$ contain the weights and biases, respectively. These are collectively called the *learnable parameters* of the network, which are learned through a process called *training*. Subsection 6.1.3 will cover the training of learnable parameters.

If we define

$$\begin{aligned}
\mathbf{F}_1 &:= \alpha(W_1\mathbf{x} + \mathbf{b}_1),\\
\mathbf{F}_i &:= \alpha(W_{i+1}\mathbf{h}_i + \mathbf{b}_{i+1}), \quad \text{for } i = 1, \dots, L-1\\
\mathbf{F}_L &:= \beta(W_{L+1}\mathbf{h}_L),
\end{aligned}$$

we can formulate the neural network as the composition of parameter-dependent functions:

$$\hat{\mathbf{f}}(\mathbf{x}; \boldsymbol{\theta}) = \mathbf{F}_L \circ \cdots \circ \mathbf{F}_1(\mathbf{x}). \tag{37}$$

Here $\boldsymbol{\theta}$ denotes all learnable parameters.

6.1.2 Activation Functions

Activation functions introduce non-linearity into neural networks, enabling them to approximate nonlinear functions and thereby capture intricate relationships within data [21]. Without activation function, the output of the neural network would be a linear combination of the inputs, which greatly restrict the usability. Common activation functions include the *sigmoid*, *tanh* and *Rectified Linear Unit* (ReLU) functions, which are visualised in Figure 10.

The ReLU function is defined by

$$\alpha(x) = \max(0, x).$$

The advantage of the ReLU function is that it is computationally efficient. A disadvantage is that it can cause nodes to "die", since the gradient for $x < 0$ is always zero. This implies that backpropagation is no longer possible and learning can terminate. More on backpropagation can be found in subsection 6.1.5. There exist adaptations of the ReLU activation function, designed to prevent the problem described above. One of them is the *Leaky ReLU*, defined in the following manner:

$$\alpha(x) = \max(0.1x, x).$$

The sigmoid function is defined by

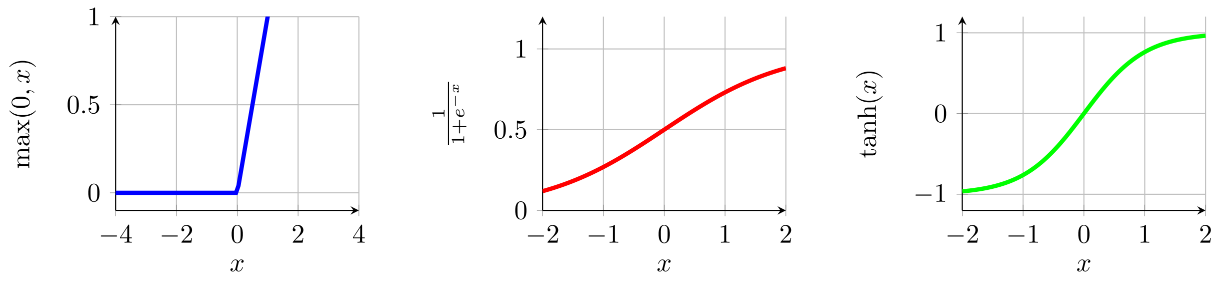
$$\alpha(x) = \frac{1}{1 + e^{-x}}.$$

Perks of the sigmoid function are that it normalises the outputs of the hidden layers and it has a smooth gradient. A large drawback is that it suffers from vanishing gradients if the values of $|x|$ are large. This can slow down or even terminate learning. For this reason, the sigmoid function is not often used as activation function in hidden layers. However, it is useful in the output layer of classification models, where it returns a probability (a score between 0 and 1).

The hyperbolic tangent can be written as

$$\alpha(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

The tanh has the same properties as the sigmoid function, except that it is zero-centered, making it more suitable for inputs with strongly negative, neutral, and strongly positive values.



(a) ReLU (b) Sigmoid (c) tanh

Figure 10: Three common activation functions.

It is clear that each activation function has its advantages and disadvantages. When training a neural network, one should try different functions to find which one performs best for the specific task at hand.

6.1.3 Loss Functions and Training

Let I and O be the input and output sets, respectively:

$$\begin{aligned} I &= \{\mathbf{x}^1, \dots, \mathbf{x}^N\}, \\ O &= \{\mathbf{y}^1, \dots, \mathbf{y}^N\}, \end{aligned} \tag{38}$$

where $\mathbf{x}^i \in \mathbb{R}^n$ and $\mathbf{y}^i \in \mathbb{R}^m$, for $i = 1, \dots, N$. Subsection 6.1.1 described how one can think of a neural network as a function $\hat{\mathbf{f}}$ that takes as input a vector $\mathbf{x}^i \in I$ and outputs a corresponding $\hat{\mathbf{y}}^i \in \mathbb{R}^m$. In Equation 37 we have seen that it is actually a composition of parameterised functions, so that we can write $\hat{\mathbf{f}}(\mathbf{x}^i;\boldsymbol{\theta}) = \hat{\mathbf{y}}^i$. The aim of the neural network is to approximate the possibly non-linear, but continuous function $\mathbf{f}$ that has the same inputs $\mathbf{x}^i \in I$ but outputs the corresponding $\mathbf{y}^i \in O$. The neural network performs well if $\hat{\mathbf{y}}^i \approx \mathbf{y}^i$, for all $i = 1, \dots, N$

Its objective is to find the values of the weights and biases, collectively denoted by $\boldsymbol{\theta}$, such that deviations of the network output from the desired output are penalised. Training a neural network means solving the following general optimisation problem:

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \mathcal{L}\left(\hat{\mathbf{f}}(\mathbf{x}^i;\boldsymbol{\theta}), \mathbf{y}^i\right). \tag{39}$$

Here $\mathcal{L}$ is a general *loss function*, which defines a distance metric between the network output and the desired output $\mathbf{y}$. There are many choices for $\mathcal{L}$. One example is the *mean squared error* (MSE) defined by

$$\mathcal{L}^{MSE}\left(\hat{\mathbf{f}}(\mathbf{x}^i;\boldsymbol{\theta}), \mathbf{y}^i\right) = \frac{1}{N}||\hat{\mathbf{f}}(\mathbf{x}^i;\boldsymbol{\theta}) - \mathbf{y}^i||_2^2.$$

Other well-known loss functions are the *root mean squared error* (RMSE) and *mean absolute error* (MAE):

$$\begin{aligned} \mathcal{L}^{RMSE}(\left(\hat{\mathbf{f}}(\mathbf{x}^i;\boldsymbol{\theta}), \mathbf{y}^i\right)) &= \frac{1}{\sqrt{N}}||\hat{\mathbf{f}}(\mathbf{x}^i;\boldsymbol{\theta}) - \mathbf{y}^i||_2, \\ \mathcal{L}^{MAE}(\left(\hat{\mathbf{f}}(\mathbf{x}^i;\boldsymbol{\theta}), \mathbf{y}^i\right)) &= \frac{1}{N}||\hat{\mathbf{f}}(\mathbf{x}^i;\boldsymbol{\theta}) - \mathbf{y}^i||_1. \end{aligned}$$

The set I defined in (38) is often called the *training set*, containing the *training data*. One element of the training set is called a *training sample*. During training, the training data is passed through the network multiple times to adapt or train the learnable parameters $\boldsymbol{\theta}$. Training consists of three phases: *forward propagation*, *backpropagation*, and *optimisation*. The former two will be treated in subsection 6.1.4 and 6.1.5, respectively. The latter refers to solving Equation (39), which usually means solving a complex, higly non-convex optimisation problem. This is realised using a gradient-based optimisation algorithm. Section 6.1.6 will delve into this subject in greater detail.

6.1.4 Initialisation and Forward Propagation

The first phase in training is the computation of the *predictions* $\hat{\mathbf{y}}^i = \hat{\mathbf{f}}(\mathbf{x}^i,\boldsymbol{\theta})$ for each sample in the training set. This is called forward propagation. For a given training sample, the prediction can be calculated using the scheme detailed in Equations (34) - (36): the input values are fed to the first layer of the network, multiplied by its weights and added to its bias, and then passed through a nonlinear activation function before being passed to the next layer. This process is repeated until the output layer is reached, giving the prediction based on the current set of weights and biases in the network.

These weights and biases need to be initialised in order to compute the first predictions. This is done once at the beginning of training and then they are shared across all training samples during forward propagation. Initialisation can be done in multiple ways and often depends on the network architecture considered. One common technique is *Xavier initialisation* or *normalised initialisation* [20]. Here the weights are drawn from the uniform distribution:

$$W \sim U\left[\frac{-\sqrt{6}}{\sqrt{n_j + n_{j+1}}}, \frac{\sqrt{6}}{\sqrt{n_j + n_{j+1}}}\right],$$

where n_j is the number of inputs of the j-th layer and n_{j+1} is the number of outputs of the $(j+1)$-th layer.

After forward propagation, the loss function $\mathcal{L}\left(\hat{\mathbf{f}}(\mathbf{x}^i;\boldsymbol{\theta}), \mathbf{y}^i\right)$ between the predictions and actual values can be evaluated for each training sample. By adding all terms we can now formulate the minimisation problem that needs to be solved, as given in Equation (39).

6.1.5 The Computational Graph and Backpropagation

To solve Equation (39), the gradient of the sum of loss functions with respect to the learnable parameters $\boldsymbol{\theta}$ needs to be determined. Due to linearity, the gradients of the separate terms in the objective function can be calculated individually and then summed. This means that the network needs to determine

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}\left(\hat{\mathbf{f}}(\mathbf{x}^i;\boldsymbol{\theta}), \mathbf{y}^i\right),$$

for $i = 1, \dots N$. The learnable parameters $\boldsymbol{\theta}$ are actually the weight matrices W_j and bias vectors $\mathbf{b}_j$, for $j = 1, \dots, L+1$ (for each hidden layer in the network and the output layer).

When training a neural network, the numerical evaluation of a gradient expression needs to determined many times and this can be computationally expensive. The *backpropagation algorithm* provides an easy and computationally cheap solution, making efficient use of the chain rule for differentiation. It works by recursively applying the chain rule and computing the gradient of the loss with respect to the output of each layer, and then propagating these gradients backward through the network to compute the gradients with respect to the parameters in each layer.

For illustrative purposes, let $\mathcal{L}(\hat{\mathbf{y}}) \in \mathbb{R}$, $\hat{\mathbf{y}} \in \mathbb{R}^m$, $\boldsymbol{\theta} \in \mathbb{R}^n$, and $\hat{\mathbf{y}} = g(\boldsymbol{\theta})$. The chain rule for differentiation is then given by

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \sum_{j=1}^{m} \frac{\partial \mathcal{L}}{\partial y_j}\frac{\partial y_j}{\partial \theta_i}.$$

Hence, the gradient of $\mathcal{L}$ with respect to $\boldsymbol{\theta}$ can be written as

$$\nabla_{\boldsymbol{\theta}}\mathcal{L} = \left(\frac{\partial \hat{\mathbf{y}}}{\partial \boldsymbol{\theta}}\right)^T \nabla_{\hat{\mathbf{y}}}\mathcal{L},$$

where $\frac{\partial \hat{\mathbf{y}}}{\partial \boldsymbol{\theta}}$ denotes the Jacobian matrix of function g. Now assume we have $\mathcal{L}(\hat{\mathbf{y}}) \in \mathbb{R}$, $\hat{\mathbf{y}} \in \mathbb{R}^m$, $\mathbf{h} \in \mathbb{R}^k$, $\boldsymbol{\theta} \in \mathbb{R}^n$, $\hat{\mathbf{y}} = g(\mathbf{h})$, and $\mathbf{h} = l(\boldsymbol{\theta})$. The gradient of $\mathcal{L}$ with respect to $\boldsymbol{\theta}$ now becomes

$$\nabla_{\boldsymbol{\theta}}\mathcal{L} = \left(\frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}}\right)^T \left(\frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{h}}\right)^T \nabla_{\hat{\mathbf{y}}}\mathcal{L}.$$

Here $\frac{\partial \hat{\mathbf{y}}}{\partial \mathbf{h}}$ and $\frac{\partial \mathbf{h}}{\partial \boldsymbol{\theta}}$ denote the Jacobian matrices of function g and l, respectively. This principle of obtaining the gradient of the loss function with respect to the network's learnable parameters as a product of its gradient and Jacobian matrices at each network layer, is at the backbone of the backpropagation algorithm.

One way to conceptualise this better is by considering the *computational graph* of the neural network. Here each variable and operation (addition, multiplication, the application of an activation function) is denoted by a node in the graph. The graph keeps track of the order in which these operations are performed. Let us consider a simple example of a network with one input $x \in \mathbb{R}$, one hidden layer with one node and one output $\hat{y} \in \mathbb{R}$. We consider only one sample of the training set. The output of the first hidden layer can be determined by

$$h = \alpha(w_1 x + b). \tag{40}$$

Assuming that the network output is calculated using no activation function and no bias, we have

$$\hat{y} = w_2 h. \tag{41}$$

This gives rise to a loss function $\mathcal{L}(\hat{y}, y)$, where $y \in \mathbb{R}$ is the desired output. The computational graph of this simple neural network is depicted in Figure 11.

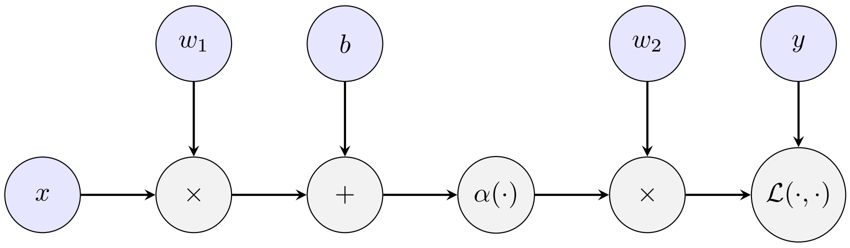


Figure 11: The computational graph of a simple neural network with one input $x \in \mathbb{R}$, one hidden layer with one neuron and one output $\hat{y} \in \mathbb{R}$. The grey nodes in the computational graph denote operations, whereas the purple nodes denote variables.

Sweeping through the computational graph from left to right gives the sequential order of operations in which we can break down Equations (40), (41) and the calculation of the loss function $\mathcal{L}(\hat{y}, y)$. For each intermediate step, we can determine the partial derivatives of the output with respect to the inputs:

$$\begin{aligned}
f &= w_1 x & &\implies \frac{\partial f}{\partial x} = w_1, & &\frac{\partial f}{\partial w_1} = x,\\
g &= f + b & &\implies \frac{\partial g}{\partial f} = 1, & &\frac{\partial g}{\partial b} = 1,\\
h &= \alpha(g) & &\implies \frac{\partial h}{\partial g} = \alpha'(g), & &\\
i &= w_2 h & &\implies \frac{\partial i}{\partial h} = w_2, & &\frac{\partial i}{\partial w_2} = h,\\
j &= \mathcal{L}(i, y) & &\implies \frac{\partial \mathcal{L}}{\partial i}, & &\frac{\partial \mathcal{L}}{\partial y}.
\end{aligned}$$

Subsequently, determining the derivative of the loss $\mathcal{L}$ with respect to the learnable parameters (in this case w_1, w_2, and b) is done by traversing the computational graph in a reverse direction. This process involves multiplying the corresponding partial derivatives, effectively applying the chain rule multiple times:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_1} &= \frac{\partial \mathcal{L}}{\partial i}\frac{\partial i}{\partial h}\frac{\partial h}{\partial g}\frac{\partial g}{\partial f}\frac{\partial f}{\partial w_1},\\
\frac{\partial \mathcal{L}}{\partial w_2} &= \frac{\partial \mathcal{L}}{\partial i}\frac{\partial i}{\partial w_2},\\
\frac{\partial \mathcal{L}}{\partial b} &= \frac{\partial \mathcal{L}}{\partial i}\frac{\partial i}{\partial h}\frac{\partial h}{\partial g}\frac{\partial g}{\partial b}.
\end{aligned}$$

6.1.6 Optimisation Algorithms

After both forward propagation and backpropagation, the gradients with respect to each learnable parameter of the objective function that the network aims to minimise are known. These gradients can then be used in a gradient-based algorithm to find the updates of the learnable parameters.

Let us first introduce the standard *gradient descent algorithm* for finding the updated weights and biases of the network. This is also called *batch gradient descent*. It is in line with what we have considered so far: solving Equation (39) by determining $\nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\hat{\mathbf{f}}(\mathbf{x}^i; \boldsymbol{\theta}), \mathbf{y}^i\right)$ for $i = 1, \ldots, N$. That is: for all samples in the training set. The idea is to utilise the fact that the gradient shows the direction of steepest ascent of the objective function. This implies that minus the gradients points into the direction of reaching a minimum. That is why the gradient descent algorithm calculates the updates of the learnable parameters as follows:

$$\boldsymbol{\theta}_{updated} = \boldsymbol{\theta} - \lambda * \nabla_{\boldsymbol{\theta}} \left[\sum_{i=1}^{N} \mathcal{L}\left(\hat{\mathbf{f}}(\mathbf{x}^i; \boldsymbol{\theta}), \mathbf{y}^i\right)\right]. \tag{42}$$

Here λ is the *learning rate*. This is a very important hyperparameter of the network and should be initialised by the researcher. Choosing a learning rate that is too large may result in jumping over the minimum of the objective function, whereas choosing it too small can result in a very long computation time or never reaching a minimum.

After the learnable parameters are updated, the learning process recommences: forward propagation is applied to the whole training set, this time using the updated weights and biases. The new loss function that needs to be minimised is calculated and backpropagation is used do determine the gradient with respect to the updated parameters. Subsequently, the gradient descent algorithm updates the parameters once again, using the scheme in Equation (42). This process is repeated until a predefined stopping criterion is met. Once the network has finished training, the weights and biases of the last iteration are frozen.

In standard gradient descent, the entire training dataset is used to compute the gradient of the objective function with respect to the model parameters in each training step. This can become very computationally expensive, especially when training set is large. That is why in practice *mini-batch gradient descent* is utilised. Here, the training set is randomly divided into disjoint sets of a fixed size. Such a set is called a *mini-batch* and the number of samples in a batch is a hyperparameter often denoted by the *batch size*. The idea is then to approximate the gradient of the objective function (which is separable, as it consists of summed terms) with the the gradient of only those terms that correspond to samples from one mini-batch. Each mini-batch contributes to one update of the model's parameters. We say that one *epoch* is completed after the model has iterated through all the mini-batches in the training dataset exactly once.

For illustrative purposes, let us assume we subdivide the training set into K disjoint sets $S_1, \dots, S_K$ with batch size k. We then approximate

$$\nabla_{\boldsymbol{\theta}} \left[\sum_{i=1}^{N} \mathcal{L}\left(\hat{\mathbf{f}}(\mathbf{x}^i; \boldsymbol{\theta}), \mathbf{y}^i \right) \right] \approx \nabla_{\boldsymbol{\theta}} \left[\sum_{i \in S_j} \mathcal{L}\left(\hat{\mathbf{f}}(\mathbf{x}^i; \boldsymbol{\theta}), \mathbf{y}^i \right) \right] = \sum_{i \in S_j} \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\hat{\mathbf{f}}(\mathbf{x}^i; \boldsymbol{\theta}), \mathbf{y}^i \right).$$

The mini-batch gradient descent algorithm calculates the updates of the learnable parameters as follows:

$$\boldsymbol{\theta}_{updated} = \boldsymbol{\theta} - \lambda * \nabla_{\boldsymbol{\theta}} \left[\sum_{i \in S_j} \mathcal{L}\left(\hat{\mathbf{f}}(\mathbf{x}^i; \boldsymbol{\theta}), \mathbf{y}^i \right) \right]. \tag{43}$$

The network iterates over the mini-batches, each time performing forward propagation and calculating the loss. It then performs backpropagation to compute the gradients of the loss with respect to the learnable parameters. Subsequently, the learnable parameters are updated using the scheme in Equation (43). This process is repeated for the desired number of epochs. For each new epoch, the data is randomly shuffled to ensure that the model doesn't learn patterns that might be specific to the order of the data in the training set, and new groups of mini-batches are created.

There exist many optimisation algorithms that are based on mini-batch stochastic gradient descent, but utilise a adaptive learning rate λ. Popular examples include *AdaGRAD* [8], *RMSprop* [27] and *Adam* [29]. While mini-batch gradient descent uses a fixed learning rate for all parameters, Adagrad, RMSprop, and Adam dynamically adjust the learning rates based on the historical behavior of each parameter during training. This approach is particularly useful in dealing with different scales and is shown to be more robust [45].

6.1.7 Testing and Validation

After the training phase, the neural network is ready to be evaluated on unseen data to assess its generalisation capabilities. This phase is crucial for determining how well the model performs on data it has never seen during training. The evaluation process involves freezing the weights and biases obtained from the final iteration of the training phase and subjecting the neural network to a *test dataset.*

To ensure a fair evaluation, a separate dataset for *testing* is used that the neural network has not been exposed to during training. Typically, the original dataset is divided into two subsets: the training set and the test set. The training set is employed for updating the model's parameters, while the test set, held out during training, serves as an unbiased measure of the model's performance. The division ratio can be chosen by the researcher and often depends on the amount of data available. Some examples are: 90%, 80%, 70% (training) vs 10%, 20%, 30% (testing), respectively.

There exist several metrics to gauge the performance of a neural network on the test set. Common performance measures include *accuracy*, *precision*, *recall*, and *F1 score*. These metrics offer insights into different aspects of the model's behaviour. We will only define accuracy here: the proportion of samples for which the model produces the correct output. Hence, an accuracy of 1.00 represents a perfect algorithm.

Optimal performance often requires fine-tuning of hyperparameters. Hyperparameters are configuration settings that are not learned during training but significantly influence the model's performance. Examples are the learning rate, the number of hidden layers, and the batch size. Techniques such as *grid search* or *random search* can be used to systematically explore the hyperparameter space and identify the combination that maximises the model's performance on the validation set [25].

6.2 Physics-Informed Neural Networks

Physics-Informed Neural Networks (PINNs) combine the power of neural networks with physical principles to solve partial differential equations (PDEs). The first to introduce this paradigm were Raissi *et al.* [41], who drew inspiration from the early contributions of Psichogios and Ungar [40], and Lagaris *et al.* [32].

Raissi *et al.* observe that when training a deep learning algorithm to accurately identify a nonlinear function in the context of physics or biology, there is often prior information available in the form of governing equations (system of PDEs) or some other empirically validated rules. This information can act as a regularisation agent, such that it constrains the space of admissible solutions of the neural network. When the information content of the data the neural network sees is in this way amplified, the algorithm can steer itself quickly to the right solution and generalise well in cases when limited training samples are available.

Let us describe the general form of a nonlinear partial differential equation:

$$u_t + \mathcal{N}[u;\lambda] = 0,\;\; x \in \Omega, t \in [0,T]. \tag{44}$$

Here, $u(t,x)$ denotes the latent solution, $\mathcal{N}[\cdot\,;\lambda]$ is a nonlinear operator parametrised by λ, and $\Omega \subseteq \mathbb{R}^D$. A wide variety of problems in mathematical physics and biology may be written in the form of Equation (44), such as conservation laws, diffusion processes, advection-diffusion-reaction systems, and kinetic equations. We note that the system of equations describing the post-burn evolution of the dermal layer as given in section 5.7 can also be written in this form.

Given (noisy) data of the system (44), there are two distinct problems we can apply a PINN to. The first one is what Raissi *et al.* call the *data-driven solution* of the PDE: given fixed model parameters λ, we want to approximate the unknown solution $u(t,x)$ of the system. To this end, we define $f(t,x)$ to be the left-hand side of Equation (44):

$$f := u_t + \mathcal{N}[u;\lambda]. \tag{45}$$

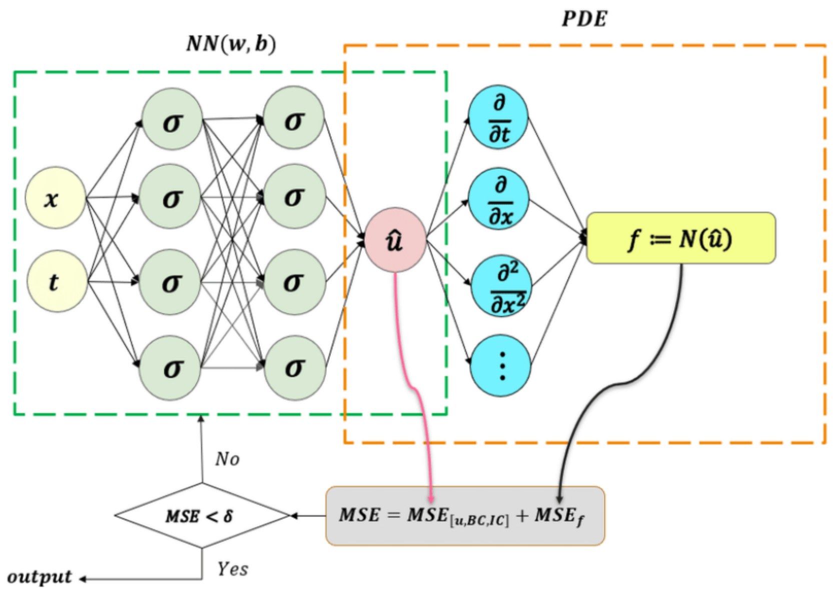


Figure 12: The schematic of PINNs for solving PDEs. Taken from [23].

We then proceed by approximating $u(t,x)$ by a deep neural network. Together with Equation (45), this results into a physics informed neural network $f(t,x)$. The network $f(t,x)$ can be derived using automatic differentiation [1], which applies the chain rule for differentiating compositions of functions. The PINN has the same parameters as the network representing $u(t,x)$, but with different activation functions due to the action of the differential operator $\mathcal{N}$. The shared parameters between the neural networks $u(t,x)$ and $f(t,x)$ can be learned by minimising the mean squared error loss

$$MSE = MSE_u + MSE_f,$$

where

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2, \tag{46}$$

and

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2. \tag{47}$$

Equation (46) corresponds to the initial and boundary data $\{t_u^i, x_u^i, u^i\}_{i=1}^{N_u}$ on $u(t,x)$. Equation (47) enforces our PDE at a finite set of *collocation points* $\{t_f^i, x_f^i\}_{i=1}^{N_f}$. MSE_f penalises the PDE not being enforced in these collocation points and encourages the PINN to learn the structural information expressed by the PDE during the training process. Figure 12 gives a schematic overview of the PINN architecture.

6.3 Operator Learning

In section 6.2 we have introduced neural networks as function approximators. Another and perhaps more powerful result is that a neural network with a single layer can accurately approximate any nonlinear *operator* (i.e., a mapping from a function space into another function space) [4]. This leads to a different application named *operator learning*. The focus in this study will be on the recently introduced deep operator network (*DeepONet* [33]) for learning operators accurately and efficiently from a relatively small dataset.

The setup in the paper by Lu *et al.* [33] that introduces DeepONets is as general as possible. We consider an operator G that takes an input function u and then $G(u)$ is the corresponding output function. For any point y in the domain of $G(u)$, the output $G(u)(y) \in \mathbb{R}$. The network takes two inputs: u and y, and outputs $G(u)(y)$. The function u is evaluated at finitely many locations $\{x_1, \ldots, x_m\}$, which Lu *et al.* call *sensors*. Part A of Figure 13 gives a visualisation of the general DeepONet architecture.

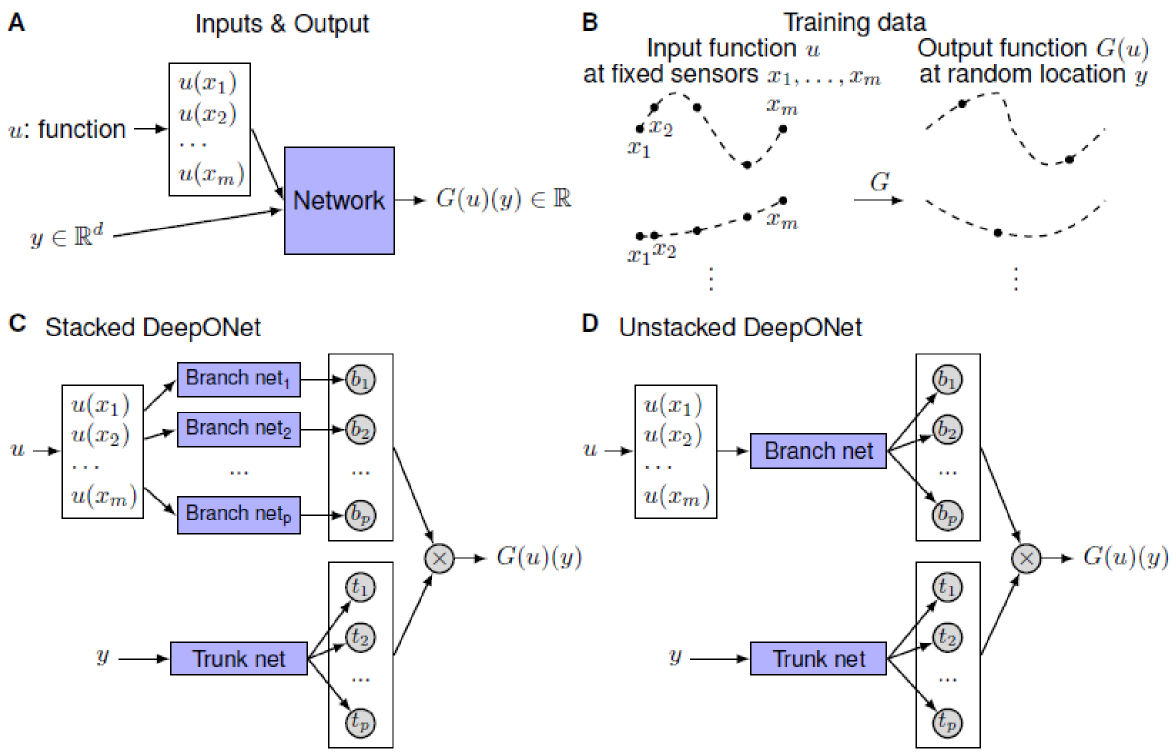


Figure 13: General DeepONet architecture (A). Sensor points are taken at fixed locations (B). The stacked DeepONet has one trunk network and p stacked branch networks (C). The unstacked variant has one trunk network and one branch network (D). Taken from [33].

During training, the DeepONet takes many functions u, which are usually sampled from a chosen function space. Possible examples include Gaussian random field and orthogonal (Chebyshev) polynomials. The only condition required is that the sensor locations are the same for all input functions u, see part B of Figure 13.

Lu *et al.* propose two versions of DeepONet. Both architectures consist of a *trunk network* that takes y as the input and outputs a vector $[t_1, \dots, t_p]^T \in \mathbb{R}^p$. The *stacked* DeepONet additionally has p *branch networks* that take as input $[u(x_1), \dots, u(x_m)]^T$ and ouput a scalar $b_k \in \mathbb{R}$, for $k = 1, \dots, p$. The *unstacked* DeepONet merges all branch networks into one single branch network (refer to part B of Figure 13). The output of the network is in both cases the dot product

$$G(u)(y) \approx \sum_{k=1}^{p} b_k t_k.$$

One way of conceptualising the branch and trunk network is by considering that the branch network extracts latent representations of input functions and the trunk network extracts latent representations of input coordinates at which the output functions are evaluated.

In the context of (a system of) partial differential equations, a DeepONet may be utilised to learn the solution operator of the system. Consider a system of PDEs where the solution is denoted by $s(x, t)$. Furthermore, there is a parameter $u(x)$ that the solution depends upon. Note that this notation differs from the notation used in section 6.2, where u denoted the solution to the PDE. In our current context, the PDE is parameterised by u, and s stands for the solution. There exist many possibilities for u. Examples include but are not limited to u representing a forcing term, a source term, an initial condition, some other variable parameter in the system, or the domain geometry. We can use a DeepONet to approximate the solution operator to the PDE:

$$G : u(x) \mapsto s(x, t).$$

That is: given any $u(x)$, the DeepONet predicts the solution $s(x, t)$ over the whole domain and for all time. The training set is generated by randomly sampling u from a chosen function space. For each u^i, the solution $s(x, t)$ is sampled at P random locations $\{(x_1, t_1)^i, \dots, (x_P, t_P)^i\}$. The training set for the DeepONet is then a triplet $[\mathbf{u}, \mathbf{y}, G(\mathbf{u})(\mathbf{y})]$, where

$$[\mathbf{u}, \mathbf{y}, G(\mathbf{u})(\mathbf{y})] = \left[\begin{bmatrix} \vdots \\ u^i(\hat{x}_1), \dots, u^i(\hat{x}_m) \\ u^i(\hat{x}_1), \dots, u^i(\hat{x}_m) \\ \vdots \\ u^i(\hat{x}_1), \dots, u^i(\hat{x}_m) \\ \vdots \end{bmatrix}, \begin{bmatrix} \vdots \\ (x_1, t_1)^i \\ (x_2, t_2)^i \\ \vdots \\ (x_P, t_P)^i \\ \vdots \end{bmatrix}, \begin{bmatrix} \vdots \\ s^i(x_1, t_1) \\ s^i(x_2, t_2) \\ \vdots \\ s^i(x_P, t_P) \\ \vdots \end{bmatrix} \right].$$

If we assume an unstacked DeepONet, the branch network takes inputs from the first vector u, the trunk network takes inputs from the second vector $\mathbf{y}$ and the desired output to which we compare the network's output are elements from the third vector $G(\mathbf{u})(\mathbf{y})$.

6.4 Physics Informed DeepONets

The PINN setup detailed in section 6.2 can be combined with DeepONets introduced in section 6.3, to obtain *physics-informed DeepONets.* This can be viewed as an extension of the DeepONet framework and was first proposed by Wang *et al.* [47].

The authors observe that although DeepONets have demonstrated great promise, their application for solving parametric PDEs faces two fundamental challenges. Firstly, they require large amounts of paired input-output observations for training, which can be very computationally expensive to acquire. Ideally, one would wish to be able to train such models without any observed data at all (i.e., given only knowledge of the PDE and its corresponding IBCs). Secondly, DeepONets predict the solution operator of a PDE, but the solution thus obtain does not necessarily adhere to the underlying PDE. To address these challenges, Wang *et al.* propose the simple, yet remarkably effective idea of combining DeepONets with the PINN setup. The general architecture of a physics-informed DeepONet is depicted in Figure 14.

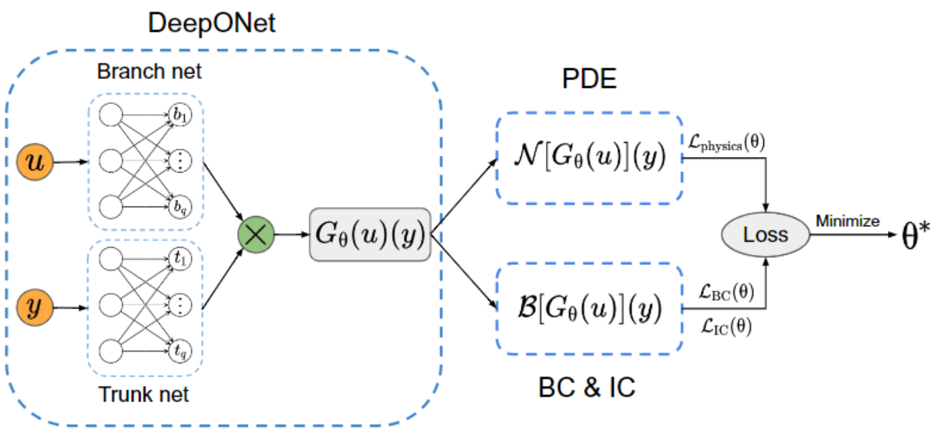

DeepONet
Branch net
Trunk net
u
y
b_1
b_q
t_1
t_q
$G_\theta(u)(y)$
PDE
$\mathcal{N}[G_\theta(u)](y)$
$\mathcal{L}_{\text{physics}}(\theta)$
Loss
Minimize
θ^*
$\mathcal{B}[G_\theta(u)](y)$
$\mathcal{L}_{\text{BC}}(\theta)$
$\mathcal{L}_{\text{IC}}(\theta)$
BC & IC

Figure 14: Physics-informed DeepONet architecture. Taken from [47].

The motivation is that the outputs of a DeepONet are differentiable with respect to their input coordinates. This allows for the use of automatic differentiation [1] to construct a mechanism for biasing the target output function to satisfy the underlying PDE constraint, much in the same way as described in section 6.2. This results into a procedure for training physics-informed DeepONet models in the case of very little training data. It is even applicable in the absence of any training data for the output function, except for the corresponding boundary and initial conditions of the system.

To illustrate the above points, we consider a simple example corresponding to an initial value problem described by the following ordinary differential equation:

$$\frac{ds(x)}{dx} = u(x), \; x \in [0,1], \qquad s(0) = 0. \tag{48}$$

The goal is to learn the solution operator G, which in this case corresponds to the anti-derivative operator

$$G : u(x) \mapsto s(x) = s(0) + \int_0^x u(t)dt,$$

for $x \in [0,1]$. To this end, we randomly sample N functions $\{u^i, \dots, u^N\}$ from a chosen function space. For each u^i, we sample the solution s^i at $P = 1$ random locations $y^i \in [0,1]$. This means that the solution in these points, given the corresponding function u, should be known to us. We choose m sensors $\{x_1, \dots x_m\}$ to evaluate the functions u^i in. Our training set now consists of the triplet

$$[\mathbf{u}, \mathbf{y}, G(\mathbf{u})(\mathbf{y})] = \left[\begin{bmatrix} u^1(x_1), \dots, u^1(x_m) \\ \vdots \\ u^N(x_1), \dots, u^N(x_m) \end{bmatrix}, \begin{bmatrix} y^1 \\ \vdots \\ y^N \end{bmatrix}, \begin{bmatrix} s^1(y^1) \\ \vdots \\ s^N(y^N) \end{bmatrix} \right].$$

Assuming an unstacked DeepONet, the elements of the first vector are inputs to the branch network and the elements of the second vector are corresponding inputs to the trunk network. We will denote the output of the DeepONet by $G_{\boldsymbol{\theta}}(\mathbf{u})(y)$. Choosing the MSE as loss function, the DeepONet needs to solve the following optimisation problem:

$$\min_{\boldsymbol{\theta}} \mathcal{L}_{\text{operator}}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} \left|\left| G_{\boldsymbol{\theta}}(\mathbf{u}^i)(y^i) - s^i(y^i) \right|\right|_2^2. \tag{49}$$

Here $\mathbf{u}^i = [u^i(x_1), \dots, u^i(x_m)]$ represents the input function. We note that the output of the DeepONet is a function of input coordinates $\mathbf{x}$. We can utilise automatic differentiation to calculate the derivative of the output with respect to the input and penalise deviations of the right-hand side of our differential equation in (48). This gives the additional minimisation problem

$$\min_{\boldsymbol{\theta}} \mathcal{L}_{\text{physics}}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}} \frac{1}{Nm} \sum_{i=1}^{N} \sum_{j=1}^{m} \left|\left| \frac{dG_{\boldsymbol{\theta}}(\mathbf{u}^i)(y^i)}{dy}\bigg|_{y=x_j} - u^i(x_j) \right|\right|_2^2. \tag{50}$$

This means that the physics-informed DeepONet needs to minimise the composite loss function given by the sum of Equations (49) and (50):

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_{\text{operator}}(\boldsymbol{\theta}) + \mathcal{L}_{\text{physics}}(\boldsymbol{\theta}).$$

The difference between a physics-informed DeepONet and a regular PINN is that the goal of the latter is to learn the solution of a parametrised PDE, for the case where the parameters are given and remain fixed during model training. This implies that a trained PINN cannot generalise to other input parameters, unless it is re-trained. As a contrast, a physics-informed DeepONet aims to learn the parametrised solution operator that maps different input parameters to the associated PDE solutions. As a consequence, the physics-informed DeepONet can quickly infer PDE solutions corresponding to different input parameters, simply by using model evaluation. Re-training the complete model is not necessary.

7 Literature on Machine Learning for Wound Healing

This chapter gives an overview of the existing landscape of neural network applications in wound healing modelling. Section 7.1 considers the existing numerical models that approximate the solution to the morphoelastic problem, used to obtain training and evaluation data. Sections 7.2 and 7.3 delve into previous research that has applied neural networks to the 1D and 2D problem, respectively.

7.1 Numerical Models

All applications of neural networks to predicting the skin evolution after burn trauma utilise existing numerical models for training data generation. The 1D numerical model [11] solves the system given in (33) on a computational domain $\Omega = [-L_1, L_1]$ with appropriate boundary and initial conditions. The initial wounded area is $\Omega^w = [-L_2, L_2]$, where $L_2 < L_1$. The model makes use of the finite element method with linear basis functions. For the time integration, the backward Euler method is applied, using a monolithic approach with inner Picard iterations. The full derivation can be found in [11].

The numerical model computes the four constituents (N, M, ρ, c) and three mechanical values (u, v, ε). Additionally, the model outputs the *relative surface area* of the wound (RSAW) and the *total strain energy* (TSE). The former can be obtained from a post-processing step, using the displacement field over the domain. The formula for the RSAW at time t is derived by adding the displacement of the wound edge at time t to the initial wound size L_2:

$$RSAW(t) = \frac{L_2 + u(t, x_b)}{L_2}.$$

The TSE is defined by the integral over the strain energy density (per unit volume) and can be seen as a measure for the post-burn discomfort a patient might experience.

In addition to the 1D model, Egberts *et al.* [9] have solved the 2D morphoelastic model for skin evolution using finite uniform bilinear elements. Here, they let the xy-plane run parallel to the surface of the skin and neglect the effects of the depth of the skin. The authors argue that such a configuration can be used to approximate the kinetics of a wound on a non-curved body part, such as a patient's chest or back. The computational domain is $\Omega = [-L_1, L_1] \times [-L_1, L_1]$ and the initial wounded area is the subset

$$\Omega^w = \left\{(x, y) : \left|\frac{x}{a}\right| + \left|\frac{y}{a}\right| \le 1\right\}.$$

Since the rhombus shape of the wound is symmetrical (see Figure 15), the code computes on one quarter of the burn domain, using the vertical and horizontal symmetry axes. The output of the 2D numerical model is the same as in the 1D case: it solves for the chemical and mechanical values, the RSAW, and the TSE.

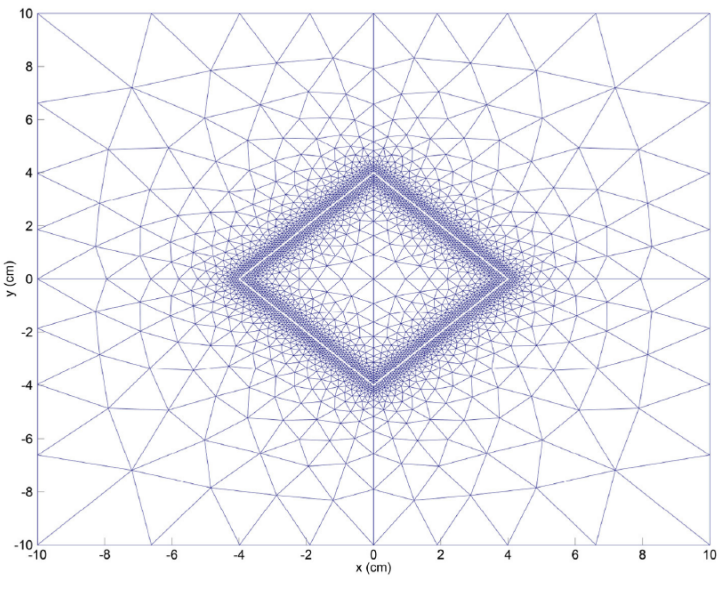


Figure 15: Initial mesh for the 2D finite element simulation. The initial wound boundary is white. Taken from [12].

7.2 One-Dimensional Problem

The first step of applying machine learning to the morphoelastic model describing post-burn skin evolution was taken by Schaaphok [44]. In her thesis, Schaaphok investigated neural network surrogates to accelerate the computations of finite element models. Here it was assumed that the simulations from the numerical models are the true values and inaccuracies in these solutions are not taken into account.

Schaaphok trained a simple feedfoward, fully connected neural network with 2 hidden layers and 100 nodes in each layer. The input to the network were the 25 parameters of the mathematical model that can vary over the domain or between patients. All other constant input values were ignored, as the neural network is able to learn these implicitly. Schaaphok argued that the relative surface area and the strain energy are the important outputs of the numerical model, as these values provide direct information on skin contraction. That is why she chose the output of the neural network to be the RSAW and the TSE. The dataset for training and evaluation is generated from the one-dimensional morphoelastic model [11].

Schaaphok showed that the network thus trained can achieve high accuracy. Furthermore, the neural network needs only 0.008 seconds to compute the predictions for 480 samples in the validation set. The computation of the original samples takes approximately 1.5 minutes per sample, which shows the significant acceleration neural networks can achieve. Schaaphok argues that a disadvantage of the surrogate method with respect to the morphoelastic model is that some of the flexibility and the physical interpretation of the model are lost. Another disadvantage is that the output format is fixed in the sense the network always predicts from day 0 to day 365.

Schaaphok published her results in a paper in collaboration with Egberts *et al.* [10].

7.3 Two-Dimensional Problem

Schaaphok considered the two-dimensional morphoelastic problem as well. For the generation of the training and evaluation datasets, the same approach was used as for the one-dimensional case. As the simulations of the two-dimensional model are computationally more expensive, fewer simulations were run and each simulation was computed until 100 days instead of 365 days. This preliminary study of the 2D model showed that a neural network can provide a significant acceleration as well, although the results were less accurate than for the one-dimensional model.

A more in-depth study of the application of neural networks to the 2D model was done by Egberts *et al.* [12]. Here, the authors consider a similar fully connected neural network with 2 hidden layers, each containing 100 nodes, and the ReLU activation function. The training and validation data (50.000 samples) is generated using the 2D numerical model. The simulation time was fixed on 365 days and the initial wound shape was the rotated square depicted in Figure 15. The inputs to the neural network were again the 25 parameters that appear in the mathematical model that describes the skin evolution. The output of the network was taken to be either RSA, the TSE, or the shape of the wound/scar boundary.

The results show high performances for all three networks considered. Furthermore, Egberts *et al.* show that neural networks trained in the above manner give a tremendous speedup of 1815000X compared to the numerical simulations. Since machine learning computations prove themselves to provide such a speedup, the authors expect this to be the way to integrate simulations into medical practice. However, they do acknowledge that there are still more extensions to explore. The most important ones include adapting the mathematical model to better represent reality, consider different, more advanced, neural network approaches and use real patient-specific data.

8 Direction of New Research

We now shift our focus away from the existing landscape of machine learning applications in wound healing modelling. This chapter outlines the direction of new research we want to take in order to contribute to this field. Section 8.1 formulates our objective and main research question. The methodology we propose to apply to answer our question is described in section 8.2. Lastly, section 8.3 will describe how the research will commence with a more simplified problem as a proof of concept.

8.1 Objective

In chapter 7 we have found that the first steps towards the application of neural networks to reproduce the finite element simulations of the wound healing model have already been taken. It was found that a simple, feedforward neural network can serve as a surrogate for the one-dimensional morphoelastic numerical model for the prediction of skin contraction. The same was found for the two-dimensional problem, albeit with a slightly less performance. The researchers who have worked on these problems acknowledge that there are still areas for improvement. The most important ones include improving the mathematical model, using real patient-specific data and utilising more advanced neural network architectures. Our area of interest is the latter.

We would like to investigate whether the application of a more sophisticated neural network architecture to the two-dimensional morphoelastic problem will prove to be fruitful. In particular, we would like to apply the DeepONets framework introduced in chapter 6, as we feel that operator learning has promising potentials. The central research question we would like to answer is the following:

Can a neural network be trained to predict the entire time evolution of skin contraction, given only the initial geometry of the wound as input?

This inquiry stems from the practical scenario faced by clinicians dealing with burn trauma patients who seek accurate predictions of wound progression for effective and timely intervention. Initially, only the wound shape and size of a patient are known. Ideally, clinicians would like to get insight into the whole spatio-temporal evolution of skin contraction. We believe operator learning to be a promising avenue, as these networks can be trained to really learn the whole solution to the corresponding system of PDEs, for all time.

8.2 Methodology

Our proposed methodology for answering the research question is to train a DeepONet to approximate the whole wound displacement field for all time t, given different initial wound geometries. Since the system (33) of PDEs that describes the evolution of wound contraction in 2D is known to us, we will incorporate this in the loss function the network needs to minimise, to bias the network to satisfy the underlying PDE. This means that we essentially aim to train a physics-informed DeepONet.

The two-dimensional finite element model will be used to create a training and test set. To this end, the numerical model will be evaluated for many different initial wound geometries, while keeping the values of the other parameters fixed. We will utilise an unstacked DeepONet, as they have fewer number of parameters than stacked DeepONets and thus can be trained faster, using much less memory.

The input to the trunk net will be different spatio-temporal locations for which we want to know the displacement of the dermal layer. The input to the branch net will be different realisations of the initial wound geometry. We will experiment with two ways of feeding the input to the branch net: either as a continuous function or as a discrete representation of the initial wound geometry. The latter could be a pixel-valued image of the wound. The difference in terms of performance and speed shall be investigated.

We will experiment with different activation functions, optimizers and hyperparameters to find the best-performing settings.

8.3 Progression of Complexity

We expect the proposed research approach to be challenging on the machine learning side. Therefore, the research will commence by considering a single material parameter within the 2D morphoelastic model, effectively reducing the problem to a more tractable form. This initial stage will serve as a proof of concept, assessing the feasibility and efficacy of the proposed setup within the morphoelastic context. Subsequently, as the model demonstrates proficiency in predicting wound displacement at fixed time t, the complexity will be incrementally increased. The introduction of a second material parameter within the framework will be a logical next step, with a gradual increase to more comprehensive models capturing additional nuances of the wound healing process.

References

[1] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43, 2018.

[2] R. A. Bradshaw and E. A. Dennis. *Handbook of cell signaling.* Academic press, 2009.

[3] A. L. Caterini and D. E. Chang. *Deep neural networks in a mathematical framework.* Springer, 2018.

[4] T. Chen and H. Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE transactions on neural networks*, 6(4):911–917, 1995.

[5] G. A. Chin, R. F. Diegelmann, and G. S. Schultz. Cellular and molecular regulation of wound healing. *Basic and Clinical Dermatology*, 33:17, 2005.

[6] C.-J. Chuong and Y.-C. Fung. Residual stress in arteries. In *Frontiers in biomechanics*, pages 117–129. Springer, 1986.

[7] R. F. Diegelmann, M. C. Evans, et al. Wound healing: an overview of acute, fibrotic and delayed healing. *Front biosci*, 9(1):283–289, 2004.

[8] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[9] G. Egberts, A. Desmoulière, F. Vermolen, and P. van Zuijlen. Sensitivity of a two-dimensional biomorphoelastic model for post-burn contraction. *Biomechanics and Modeling in Mechanobiology*, 22(1):105–121, 2023.

[10] G. Egberts, M. Schaaphok, F. Vermolen, and P. v. Zuijlen. A bayesian finite-element trained machine learning approach for predicting post-burn contraction. *Neural Computing and Applications*, 34(11):8635–8642, 2022.

[11] G. Egberts, F. Vermolen, and P. Van Zuijlen. A one-dimensional morphoelastic model for burn injuries: stability analysis, numerical validation and biological interpretation. *arXiv preprint arXiv:2010.12897*, 2020.

[12] G. Egberts, F. Vermolen, and P. van Zuijlen. High-speed predictions of post-burn contraction using a neural network trained on 2d-finite element simulations. *Frontiers in Applied Mathematics and Statistics*, 9:1098242, 2023.

[13] A. Erlich, T. Lessinnes, D. Moulton, and A. Goriely. A short introduction to morphoelasticity: the mechanics of growing elastic tissues. *Extremely Deformable Structures*, pages 269–297, 2015.

[14] M. Flanagan. *Wound healing and skin integrity: principles and practice.* John Wiley & Sons, 2013.

[15] R. K. Freinkel and D. T. Woodley. *The biology of the skin.* CRC Press, 2001.

[16] D. Gawkrodger. *Dermatology e-book: an illustrated colour text.* Elsevier Health Sciences, 2016.

[17] I. George Broughton, J. E. Janis, and C. E. Attinger. The basic science of wound healing. *Plastic and reconstructive surgery*, 117(7S):12S–34S, 2006.

[18] R. Ghiulai, O. J. Roşca, D. S. Antal, M. Mioc, A. Mioc, R. Racoviceanu, I. Macaşoi, T. Olariu, C. Dehelean, O. M. Creţu, et al. Tetracyclic and pentacyclic triterpenes with high therapeutic efficiency in wound healing approaches. *Molecules*, 25(23):5557, 2020.

[19] G. Glas, M. Levi, and M. Schultz. Coagulopathy and its management in patients with severe burns. *Journal of Thrombosis and Haemostasis*, 14(5):865–874, 2016.

[20] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[21] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning.* MIT press, 2016.

[22] A. Goriely, M. Robertson-Tessi, M. Tabor, and R. Vandiver. Elastic growth models. *Mathematical modelling of biosystems*, pages 1–44, 2008.

[23] Y. Guo, X. Cao, B. Liu, and M. Gao. Solving partial differential equations using deep learning and physical constraints. *Applied Sciences*, 10(17):5917, 2020.

[24] C. L. Hall. *Modelling of some biological materials using continuum mechanics.* PhD thesis, Queensland University of Technology, 2008.

[25] A. Heinlein and K. Postek. *Linear Algebra and Optimization for Machine Learning.* TU Delft, 6 2022.

[26] T. Hillen and K. J. Painter. A user's guide to pde models for chemotaxis. *Journal of mathematical biology*, 58(1-2):183–217, 2009.

[27] G. Hinton, N. Srivastava, and K. Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2, 2012.

[28] B. Hinz. Formation and function of the myofibroblast during tissue repair. *Journal of Investigative Dermatology*, 127(3):526–537, 2007.

[29] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[30] D. C. Koppenol, F. J. Vermolen, F. B. Niessen, P. P. van Zuijlen, and K. Vuik. A mathematical model for the simulation of the formation and the subsequent regression of hypertrophic scar tissue after dermal wounding. *Biomechanics and modeling in mechanobiology*, 16:15–32, 2017.

[31] S. S. Kordestani. Chapter 3 - wound healing process. In S. S. Kordestani, editor, *Atlas of Wound Healing*, pages 11–22. Elsevier, 2019.

[32] I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9(5):987–1000, 1998.

[33] L. Lu, P. Jin, and G. E. Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

[34] P. Lu, K. Takai, V. M. Weaver, and Z. Werb. Extracellular matrix degradation and remodeling in development and disease. *Cold Spring Harbor perspectives in biology*, 3(12):a005058, 2011.

[35] G. Majno and I. Joris. *Cells, tissues, and disease: principles of general pathology.* Oxford University Press, 2004.

[36] J. McGrath, R. Eady, and F. Pope. Anatomy and organization of human skin. *Rook's textbook of dermatology*, 1:3–2, 2004.

[37] W. Montagna. *The structure and function of skin.* Elsevier, 2012.

[38] R. Penzer and S. Ersser. *Principles of skin care: a guide for nurses and health care practitioners.* John Wiley & Sons, 2010.

[39] K. Pfisterer, L. E. Shaw, D. Symmank, and W. Weninger. The extracellular matrix in skin inflammation and infection. *Frontiers in cell and developmental biology*, 9:682414, 2021.

[40] D. C. Psichogios and L. H. Ungar. A hybrid neural network-first principles approach to process modeling. *AIChE Journal*, 38(10):1499–1511, 1992.

[41] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[42] J. N. Reddy. *An introduction to continuum mechanics.* Cambridge university press, 2013.

[43] E. K. Rodriguez, A. Hoger, and A. D. McCulloch. Stress-dependent finite growth in soft elastic tissues. *Journal of biomechanics*, 27(4):455–467, 1994.

[44] M. Schaaphok. A fast neural network-based computational framework for the prediction of skin contraction. Unpublished thesis, 2020.

[45] T. Schaul, I. Antonoglou, and D. Silver. Unit tests for stochastic optimization. *arXiv preprint arXiv:1312.6055*, 2013.

[46] A. J. M. Spencer. *Continuum mechanics.* Courier Corporation, 2004.

[47] S. Wang, H. Wang, and P. Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science advances*, 7(40):eabi8605, 2021.

[48] Wikimedia Commons. File:components stress tensor cartesian.svg — wikimedia commons, the free media repository. `https://commons.wikimedia.org/w/index.php?title=File:Components_stress_tensor_cartesian.svg&oldid=639858293`, 2022. [Online; accessed 19-October-2023].

[49] Wikipedia contributors. Finite strain theory — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Finite_strain_theory&oldid=1169218038`, 2023. [Online; accessed 14-August-2023].

[50] V. Zhou. Neural networks from scratch. `https://victorzhou.com/series/neural-networks-from-scratch/`, 2019.