Multiscale modelling of aortic dissections

Andrew Brown

Submitted in fulfilment of the requirements for the
Degree of Doctor of Philosophy

School of Mathematics and Statistics
College of Science and Engineering
University of Glasgow

October 2024

Abstract

The human aorta is the main and largest artery in the human body, transporting oxygenated blood, water and vital nutrients to all parts of our bodies. An aortic dissection is an acute life-threatening cardiovascular disease. It occurs when a small tear in the artery wall is pressurised by the blood flow, thus causing the tear to propagate. This process can quickly lead to death due to a restricted supply of blood flow, damage to the aortic valve or a full rupture of the artery wall. Understanding the fundamental mechanics of this process is crucial for the prevention and treatment of the disease. The current understanding of this process comes from a number of phenomenological studies which do not fully account for the role of the aortic microstructure. Our arteries are complicated three-layered structures, with each layer of the artery playing a different crucial mechanical role. The mechanical properties of each layer are due to their differing microstructures. These microstructures vary at the cellular level, with some layers being made up on laminated rows of smooth muscle cells and other layers being made up fibrous bundles of tissue. It follows that understanding the tearing process from a microscopic perspective is key to creating a mathematical model of aortic dissections.

The goal of this thesis is to develop a new multiscale model of material failure which fully explains the role of an arbitrary microstructure at a macroscopic level. To do this we first begin by introducing the damage phase field method. Using this method, we create a damage model comprising of two partial differential equations: an equation of motion and a damage evolution equation. Our model also has several physical constraints, including the irreversibility of damage and an energetic criterion for damage evolution, such constraints ground the model in the physical world. By applying this model over an arbitrary microscopic domain we can approximate the aortic microstructure as a composite material made of many different constituents. This provides us with an extremely large set of partial differential equations which one cannot practically work with. However, by upscaling this microscopic model we create a simpler macroscopic model comprising of an equation of motion and a damage evolution equation. This is done by employing asymptotic homogenisation and an assumption of local periodicity at the micro scale. With careful analysis we can fully upscale all the microscopic modelling constraints to the macro scale. Our macroscopic model accounts for the microscopic material properties

and geometry by encoding them in a series of effective macroscopic coefficients. These coefficients themselves are calculated by solving a series of local cell problems relating the microscale to the macroscale.

For completeness, we introduce numerical methods for solving our damage phase model. We apply these numerical methods to both our microscopic and macroscopic models of material failure. By solving our models in one-dimension we perform a parametric analysis of both models. This analysis allows us to demonstrate a good agreement between the microscopic and macroscopic models, as one would expect. We also demonstrate that the microscopic model converges to the macroscopic model as the ratio of the microscopic and macroscopic length scales tends towards zero. Next, we use the finite element method to create a numerical method for solving our damage model in higher dimensions. To illustrate the utility of our model we consider a set of toy simulations. We achieve this by simulating a trouser test, where a rectangular configuration is clamped at one end and at the other end we pull the material apart. This action essentially creates a tear through the material resulting in a final configuration that looks like a pair of trousers. By performing a trousers test we are able to convey how our multiscale model can be used to study the effects of small changes at the micro scale. We find that these small changes can have a large effect on the outcome of the tearing process at the macroscale.

It follows, then that our developed multiscale model of the damage phase field method will be a valuable tool for further study into how changes in a materials microstructure can affect material failure. Throughout the thesis we remark upon how this model can be applied to the phenomena of aortic dissections. However, before this application can be performed it is commented upon how we require more data about the aortic microstructure to be collated. With this information we can calculate the appropriate ranges of effective coefficients describing the aorta. Allowing us to accurately study a wide variety of physical situations leading to aortic dissections.

Acknowledgements

To begin, I would like to begin by thanking my supervisors, Prof. Nicholas Hill, Dr Steven Roper and Dr Raimondo Penta. None of this work would have been possible without all your support throughout the PhD. You have all shown me how extraordinary research can be and it's been a real joy. I would also like to especially thank Raimondo for encouraging me to apply for a scholarship back in 2019.

To the staff and students in the Department of Mathematics and Statistics at the University of Glasgow, thanks for all being amazing and fun at work. I have spent nine years at the University of Glasgow getting my master's and now PhD, I will always remember this time warmly.

To SoftMech, which has been a constant source of awe and motivation throughout this PhD. At every seminar and event, I attended I always learned something truly compelling about the mathematics of biology. Before this PhD I never cared much for biology, or the maths related to it. Now I think it's the most fascinating field in applied maths. I really hope I can spend the rest of my life working in this field.

I would also like to show my appreciation to the EPSRC for awarding me with a research studentship. This financial support has been crucial in supporting me throughout my studies.

I always want to give special appreciation to my four-legged friends that helped me write this thesis by sleeping on my feet. You're a bunch of good boys Bo, Theo and Dempsey.

To my family, who saw this coming? I never thought this is where I'd end up in life. First one to go to university and I have gone the whole way. To my siblings, Liam, Keri and Toni, your kids ever need help with homework just ask. To my nana, Maria, I am so happy you're here to see me make it. To my mum, Helena, I hope your proud, I've thought about all the love and support you gave me growing up every step of the way.

Finaly, I want to thank my partner, Vannessa. You especially have put up with me more than anyone as I have stressed and raged at ever hurdle. After all those rants you're probably also an expert in aortic dissections, multiscale modelling and numerical methods. But you have always loved me, pushed me to keep going and I do love it when you poke fun at my work. I hope you know that I do love you and appreciate you every day.

Declaration

I Declare that all work in this thesis was carried out by myself unless otherwise explicitly stated and has not been submitted for any other degree at the University of Glasgow or any other institution.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

The cardiovascular system, sometimes called the circulatory system, is responsible for delivering nutrient rich, oxygenated blood to every part of the body [60]. Our cardiovascular system consists of a complex network of organs, arteries, and microscopic blood vessels dispersed throughout the whole body [36]. At the centre of this network is the heart, which functions as a muscular pump propelling blood through the cardiovascular system [70]. This process begins with oxygen depleted blood from the body being pumped into the heart's right atrium. Next, blood enters the right ventricle and is sent to the lungs for oxygenation via the pulmonary artery. After which, the now oxygen rich blood travels up the pulmonary vein and enters the heart's left side, first reaching the left atrium. Finally, this blood is pumped into the left ventricle, ready to enter the body's network of arteries.

Blood is pumped out of the heart's left ventricle into a network of arteries. The large blood vessels carry oxygenated blood away from the heart to various tissues and organs. Arteries begin to branch into smaller blood vessels called arterioles, which themselves eventually branch into a system of microscopic capillaries [101]. These thin capillary vessels cause red blood cells to deform and squeeze out all the oxygen and nutrients they are carrying. Gases, water and nutrients in the capillary vessels are then diffused through the capillary walls, feeding the body's various organs and tissues [60]. Capillaries then merge into veins, which carry the oxygen depleted blood back towards the heart. These veins progressively merge into larger vessels, ultimately becoming the superior and inferior venae cavae, which return blood to the right atrium of the heart [95]. This continuous flow of blood ensures the proper functioning of tissues and organs.

The human aorta, a vital component of the cardiovascular system, serves as the body's main artery. Being the largest blood vessel in the body, it stretches the length of the torso, emerging from the heart's left ventricle before arching downwards and travelling towards the abdomen. Its function is to distribute oxygenated blood from the heart to the entire body, ensuring that nutrients and oxygen reach every cell and piece of tissue in our body. To perform this role, the aorta must be able to withstand large pulsating pressure changes.

The large arteries, including the aorta, have elastic properties, allowing them to expand and contract [33]. The elasticity of the aorta is crucial for maintaining optimal blood pressure and ensuring a smooth, continuous flow of blood.

Unfortunately, due to the aorta dealing with such large pressure, it is possible for the aorta to tear. This phenomenon is called an aortic dissection, which is a serious medical emergency that can often lead to death [116]. Aortic dissections occur when tears form on the inner lining of the aorta, allowing blood to flow into the aortic wall and driving further tearing. As blood flows into the tear, it usually propagates longitudinally down the aorta, causing the layers of the aortic wall to delaminate [4]. If the tear fully ruptures the aorta, this will cause an individual to bleed out internally in mere minutes. Usually, an aortic dissection results in the creation of a 'false' lumen, a secondary channel alongside the 'true' lumen for blood to flow into, which may cause the aorta to narrow and even close off entirely. Simultaneously, the dissection may cause the formation of a thrombus from which fragments embolize [96]. Aortic dissections occur either spontaneously due to physiological reasons, which is the interest of this thesis, or they may occur due to physical trauma [2].

One of the earliest records of an aortic dissection was described in an autopsy of George II, King of Great Britain and Ireland, by Frank Nicholls in 1760 [26], who was his personal physician. Our first surgical treatments for aortic dissections were introduced in the 1950s by Michael DeBakey [29]. Later in 2005, he suffered an aortic dissection and received surgical treatment to save his life. However, even with our advances in technology, aortic dissections still carry a mortality rate of around 90% for untreated patients [5] and a mortality rate in the range of 30% for surgically treated patients [87]. With around 3 in 100,000 cases being reported annually throughout western Europe [78, 88, 99], aortic dissections are a rare but still all too deadly form of disease.

For this thesis, we are motivated to create new and novel mathematical models that help us understand the underlying mechanics and causes of aortic dissections. To do this, we introduce a damage phase field model [31] of dissections to approximate the process of aortic dissections. With this approach, we can derive a system of equations describing momentum balance and damage evolution. The former describes the mechanics of the system, namely through its displacement, and the latter describes a damage variable, which is used to measure how close a point is to rupturing. Using this approach to model the aortic microstructure, we can derive a system of partial differential equations that describes each individual constituent part of the aortic wall. To do this, we must approximate the aortic wall as a composite material made up of smooth muscle cells, endothelial cells, collagen, and elastic fibres. From there, we will explain how to upscale such a model using asymptotic homogenisation [103] and exploiting sharp length scale separations between the micro and macro scales. This creates a macroscopic model of

aortic dissections where the role of the aortic microstructure is encoded in a set of effective macroscopic coefficients. By better understanding the effect of the structure of the aortic microstructure on the mechanical properties, this work can lead to a statistical discussion on the variation and ranges, which naturally increase the likelihood of aortic dissections occurring.

In this chapter, we introduce the relevant background knowledge of the aortic anatomy and aortic dissections required. We will also introduce the reader to relevant biological concepts and related literature before outlining possible modelling approaches and previous mathematical modelling work. Finally, we will lay out the overall goals of this thesis and outline the contents of each chapter.

1.1 The aorta

In this section, we provide important background information about the aorta. Namely information about its anatomy and its physiological purpose in §1.1.1. The known mechanical behaviour of the aorta in §1.1.2. We shall also provide a detailed description of an aortic dissection and the disease's aetiology in §1.1.3.

1.1.1 Anatomy of the aorta

To model the aorta, it is important to understand the anatomy of the aorta. Arteries are structured to perform a specific job: withstand blood pressure and transport oxygenated blood through the body. In general, arteries are roughly divided into two types: elastic and muscular. Elastic arteries have relatively large diameters and are located closer to the heart, whereas muscular arteries are narrower and located at the periphery. Be aware that there are exceptions to this rule of thumb, and some arteries exhibit properties of both elastic and muscular arteries. The aorta is classified as the largest elastic artery in the human body. We call it elastic because it has a larger percentage of fibrous tissue, allowing it to stretch more [13, 60]. This allows the aorta to cope with the large variations in blood pressure.

Here we will focus on the microscopic structure of the artery wall, which is composed of three distinct layers: the tunica intima, the tunica media, and the tunica adventitia, as seen in Figure 1.1. We will discuss each of these layers in detail, explaining their overall function, composition, and the role they play in arterial mechanics. For a more complete review of the structure and role of arteries, and specifically the aorta, we recommend the books [60], [36] and [111].

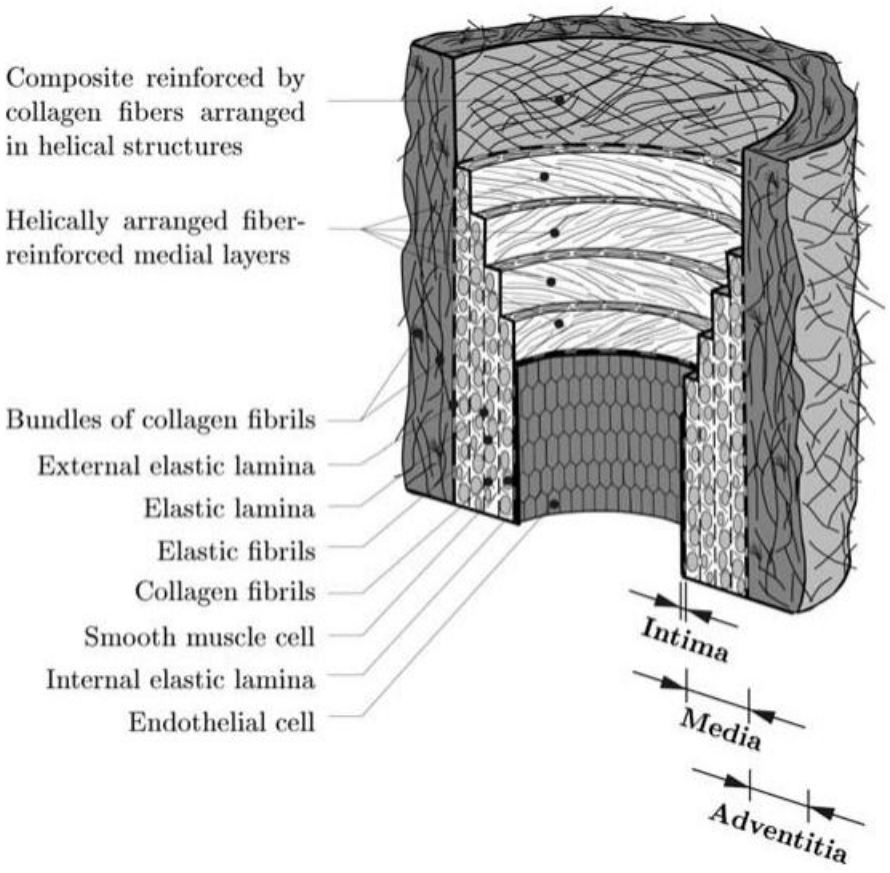


Figure 1.1: Cross section of an elastic artery wall composed of the tunica intima, media and adventitia. Here it can be seen that each layer of the artery wall has a unique microscopic structure built from different constituent parts. Image sourced from [65].

Tunica intima

The innermost layer of the artery, the tunica intima, is a sheet of flattened endothelial cells resting on a thin layer of connective tissues. The endothelial layer acts as a barrier, stopping blood plasma from escaping into the arterial wall. There is also a sub-endothelial layer whose thickness varies with age. For younger people, the intima is very thin and flexible, so it is often considered to be mechanically insignificant. However, in older people, the intima stiffens and thickens with age, making it more mechanically relevant. It also makes it more prone to tearing under higher blood pressure, possibly causing dissections.

Tunica media

The middle layer of the artery, the tunica media, supplies mechanical strength and contractile power to the arterial wall. It consists of spindle-shaped, smooth muscle cells arranged circumferentially. Layers of smooth muscle cells lie between elastic laminae interspliced with elastic collagen fibrils and elastic fibres, providing extra reinforcement. The number of elastic laminae decreases and increases as the size of the artery increases. Our tunica

media is bound by two sheets of elastin known as the internal and external elastic laminae.

The circumferential alignment of the medial layers has a slight helical pitch, making each layer a fibrous helix that provides strength to the structure of the arterial wall. As a result, the media can resist loads in both the circumferential and longitudinal directions. Mechanically, the media is regularly regarded as the most important layer in healthy arteries.

Tunica adventitia

The outermost layer of the artery, the tunica adventitia, is constructed mainly out of thick layers of collagen fibrils and some elastin fibres, as well as sparse populations of fibroblasts and fibrocytes (cells that produce collagen and elastin). These fibrous layers are also arranged helically and serve to reinforce the wall. At low pressures, this layer is far less stiff than the media, since the collagen fibrils are still quite relaxed and bundled. At high pressure, the adventitia becomes stiff as the collagen fibrils reach straightened lengths and reinforce the arterial wall. This helps to prevent the arterial wall from overstretching, thus maintaining appropriate blood pressure and stopping the wall from rupturing. It should also be noted that this layer anchors the artery to the surrounding connective tissue and organs that anchor it in place.

Geometry of the aorta

Anatomically, it is important to understand the macroscopic geometry of the human aorta. To do this, let us consider the depiction of the aorta illustrated in Figure 1.2. The body's aorta starts at the human heart, where blood exits the left ventricle via the aortic valve, which prevents backflow. From here, blood first travels up the ascending aorta before beginning to turn as it reaches the aortic arch. Along the ascending aorta, there are two branches called the coronary arteries, which supply blood to the heart itself. The aortic arch loops over the left pulmonary artery and heads downward, following the spine. Along this arched section, a number of arterial branches appear, including the brachiocephalic trunk, the left common carotid artery, and the left subclavian artery. These branches along the arch feed blood to the body's arms and head. At around the fourth or fifth thoracic vertebrate, we leave the aortic arch and enter the descending aorta. The descending aorta follows the curve of the spine and is anchored at the back by connecting tissue while unattached at the front. This section of the aorta extends down through the thoracic torso into the abdomen. This new part of the aorta is commonly called the abdominal aorta and is responsible for transporting blood to the lower body. Along this section of the aorta, one finds many arterial branches: the celiac trunk branches into all of the gastric arteries; the renal arteries supply the kidneys with blood; the superior and inferior mesenteric arteries feed blood to the body's gastrointestinal tracts; and the body's gonadal arteries feed blood

to important sexual organs. Finally, the aorta ends by bifurcating into the left and right iliac arteries, which supply blood to the lower body. Anatomists subdivide the aorta into two distinct regions: the thoracic aorta and the abdominal aorta. The thoracic aorta is the part of the aorta located in the thorax that contains the ascending, arched, and upper parts of the descending aorta. The body's abdominal aorta is a direct continuation of the descending aorta and is contained in the abdominal cavity of the body.

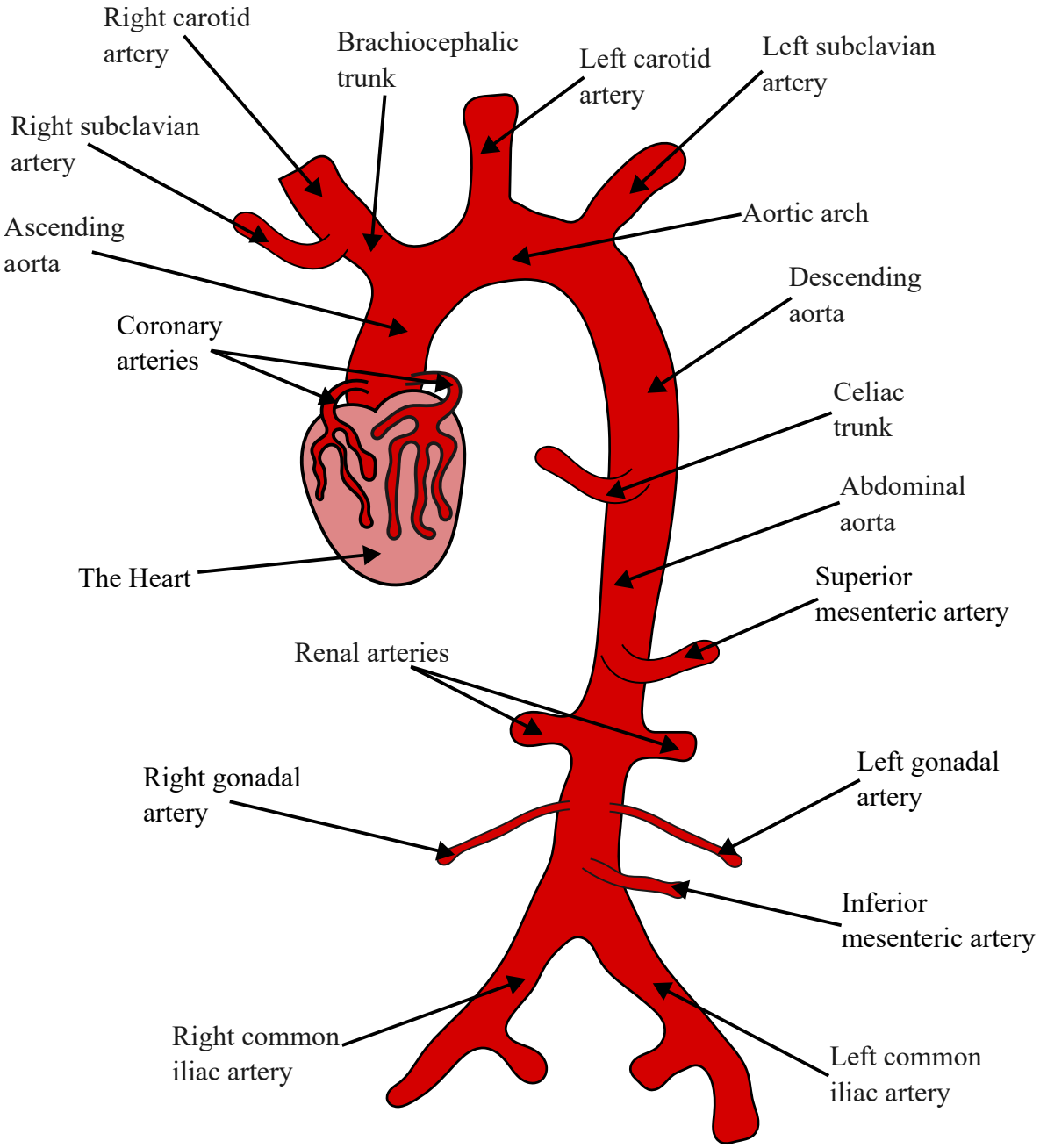


Figure 1.2: A diagram of the human aorta, from the heart extending down to the iliac bifurcation. Each branch from the aorta is labelled along with each section of the aorta. The body's thoracic aorta includes the ascending aorta, the aortic arch and the descending aorta, all contained within the body's thorax. The body's abdominal aorta includes everything below the descending aorta, all contained with the body's abdominal cavity.

1.1.2 Mechanical behaviour of arterial walls

Now that we understand the anatomy and structure of the aorta as well as the role each layer of the aorta plays, we will now discuss the overall mechanical behaviour of the aorta. As a large elastic artery, the aorta must withstand a normal physiological range of blood pressures from 80 to 120 mmHg [123]. Physiologically, the aorta has interesting mechanical properties. These are due to the composite characteristics of arteries; for example, at lower blood pressures around 80 mmHg, the collagen fibres within the wall are not straightened and do not contribute to any stresses; however, the elastin fibres are already completely straightened. As pressure increases, the elastin fibres begin to stretch with little resistance while the collagen fibres unwind and straighten out. At the upper limit of arterial pressure, 120 mmHg, the collagen fibres are completely straightened, adding greater resistance to the arterial wall. The reason for this behaviour is that at lower pressures, the arterial mechanics are dominated by elastin, and therefore the wall can easily stretch. As pressure increases, the stress in the wall transitions from elastin to collagen fibres, and at high pressures, the mechanics of the wall become dominated by the collagen fibres. Therefore, much greater loading forces are now required at higher pressures to continuously deform the wall. The stiffer collagen fibres also prevent the arterial wall from becoming damaged or rupturing [49].

The result of the arterial wall's geometry and overall composition is a non-linear mechanical behaviour, as illustrated in Figure 1.3. Physiologically, the purpose of this mechanical response is to regulate blood flow. At lower stresses the wall's mechanics are dominated by the elastin fibers thus making the wall extremely compliant to deformation. However, at much higher stresses the wall begins to rapidly stiffen. As a result pressure waves are dampened thus regulating the blood flow. We can explain these properties rather simply by considering that every layer of the arterial wall arranges its constituents in a layered helical/circumferential structure [68]. As a result, when blood pressure in the radial direction increases, the wall naturally stretches in the circumferential direction. This causes the fibres to straighten out much faster, and therefore, as blood pressure increases, the stiffness from the wall rapidly increases. An interesting physical phenomena that does occur due to this mechanical setup is known as a pistol shot pulse. Here the rapid stiffening of the aortic wall at extremely high blood pressures causes a shock formation to occur.

Another important physical property of arteries to mention is that they are a pre-stressed material. That is to say in the absence of any loading force, an artery is not stress free. A simple way to illustrate this in a laboratory is to cut out a small segment from an artery. If we measure its initial length after being removed then wait a few hours and measure it again we would observe a contraction in length. If we instead cut another arterial segment along the radius over time we would observe the artery opening. Both

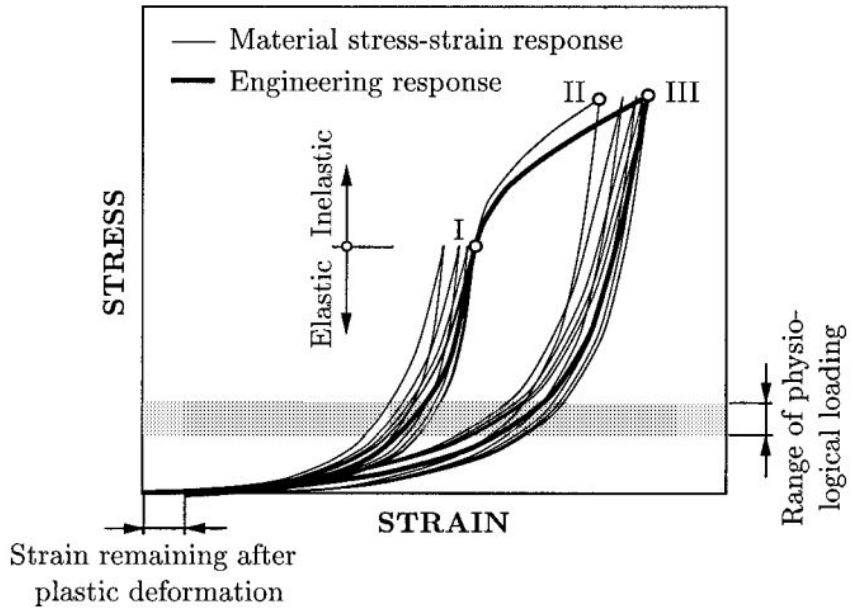


Figure 1.3: Schematic diagram of typical uniaxial stress–strain curves for circumferential arterial strips, from the media, in passive condition (based on tension tests performed in a laboratory see [65]): cyclic loading and unloading, associated with stress softening effects, lead to a pre-conditioned material which behaves perfectly elastically (nearly repeatable cyclic behaviour) – point I. Loading beyond the elastic domain up to point II leads to inelastic deformations. Additional loading and unloading cycles display stress softening again until point III is reached. Then the material exhibits a perfectly elastic response. The thick solid line indicates the (approximate) engineering response of the material. Image sourced from [65].

of these observations are due to pre-stress in the aortic wall [65, 66]. This information is important to consider if one wants to construct an elastic potential density function which accurately describes arterial mechanics. To go from a stress free configuration of an arterial strip to the regular reference frame of an artery one would need to consider bending, extension, torsion and inflation of this strip just to capture all of the pre-stress and stretch in the artery wall. In other words any elastic potential density function must account for the elastic potential density already present in the arterial wall due to the pre-stressed nature of an arterial wall. Only then would we have a realistic model of the arterial wall.

A very important remark we will make, which is illustrated by Figure 1.3, is that we can linearise the mechanics of the aortic wall around some residual stress term. This residual stress term account for all the pre-stress already present in the material as observed in Figure 1.3. The motivation for linearisation is that it makes modelling simpler, finding numerical solutions simpler and upscaling our models simpler. We could linearise in multiple different ways. For example we may introduce a piecewise linearisation of arterial stress under Hooke's law. To do this we would introduce a piecewise form of the stiffness tensor, which would approximate the stress-strain curve visible in the physiological range

of Figure 1.3. Another method for achieving this would be to introduce a strain energy density function which is the sum of Hooke's law potential energy density (explaining the materials elastic response) and the energy density defined by a residual stress term. This residual stress term would account for the residual strain in the material and be a known term. Later we will discuss that our main motivation for linearisation of the mechanics is to ensure that we can achieve an upscaling of our multiscale model which ensures a separation between the microscopic and macroscopic variables. We want this because it will make solving our macroscopic model far more computationally feasible.

1.1.3 Aortic dissections

Aortic dissection refers to a tear in the aortic wall permitting longitudinal propagation of a blood-filled space within the aortic wall. This usually begins because of aortic tissue abnormalities or increased aortic wall stress. As a result, a small tear on the tunica intima occurs, allowing the formation of a small false lumen with minimal effects on blood flow. The false lumen propagates downstream in a longitudinal direction, and as it grows, the blood flow is redistributed. As the false lumen reaches its final shape, it may tear back into the true lumen, allowing blood to flow through either channel. The false lumen may also reach a stable final form in the tunica media, where the blood pressure is very low and non-pulsatile. In both cases, it is possible, but not necessary, that a thrombus may accumulate. However, if a tear ruptures the outer wall of the tunica adventitia, this will result in death. In Figure 1.4 we can see a simplistic diagram of an aortic dissection and the various steps that occur during a dissection.

Luckily, many dissections remain in the tunica media, which can still result in death but are treatable. We believe this is because the most natural way for the aortic wall to tear open is by delamination between the layers of the tunica media [4]. This process can be thought of as the path of least resistance. In Figure 1.5 we can see an example CT scan of an aortic dissection present in the body.

Identifying aortic dissections is itself a challenge. The disease has been called "the great masquerader" [38] because it can produce symptoms related to virtually any organ. Common symptoms can include high blood pressure, vomiting, and dizziness, which could indicate a wide variety of diseases. The most telling symptoms of an aortic dissection are a severe, abrupt onset of chest or back pain, which are widely known as the classic symptoms of an aortic dissection [50, 100]. Clinicians often image patients, using ultrasound, that present unexplainable and severe chest or back pain, looking for a dissection in the thoracic aorta [38].

If an aortic dissection is identified by a clinician, then it is categorised using the Stanford classification system [118]. This is a pragmatic classification system where aortic dissections are classified as either Type A or Type B. The key difference between Types A

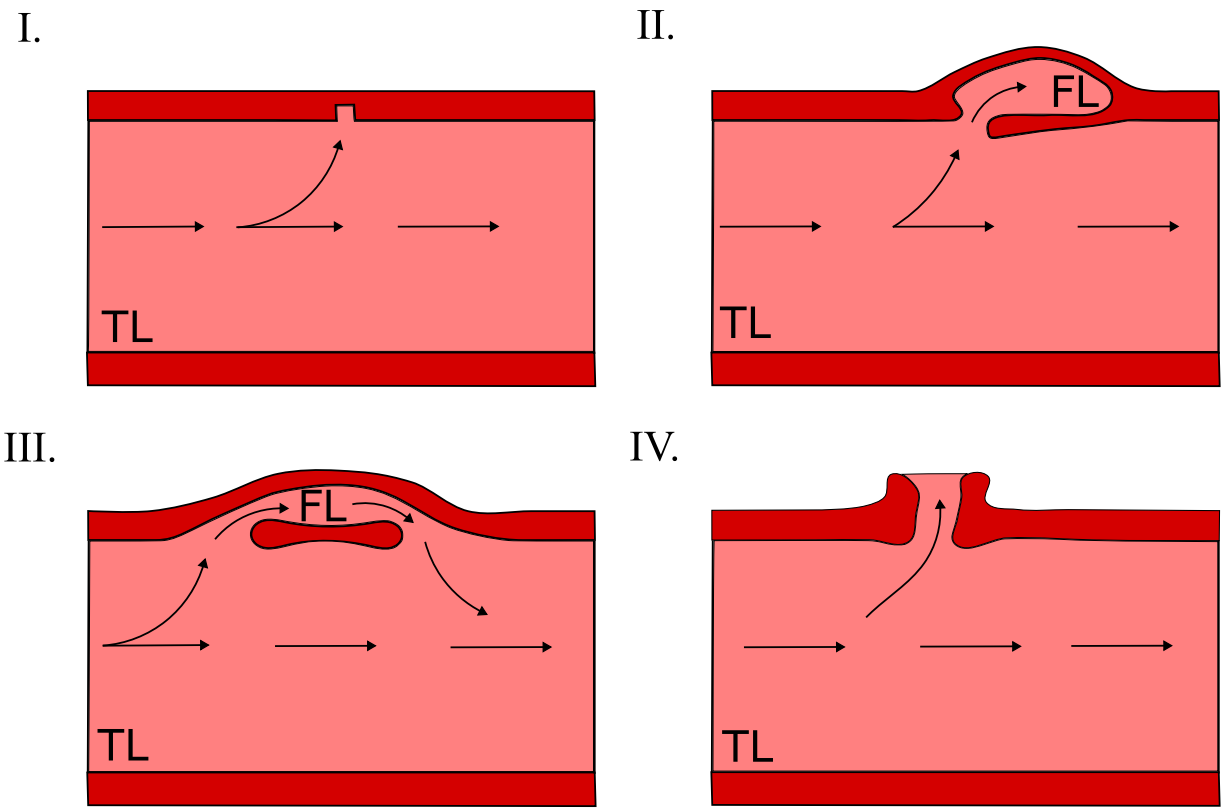


Figure 1.4: This is a simplistic schematic of the process of an aortic dissection. In step I we have the beginnings of an aortic dissection. With the arrows directing blood flow and an indication of the true lumen (TL). Initially, we have a small tear on the inner layer of the aortic wall, the tunica intima. This tear becomes pressurised as blood rushes in, which as a result propagates the tear. In step II the tear propagation has caused the layers of the tunica media to delaminate. As a result we now have a false lumen (FL). This process could now reach a equilibrium causing a large bulge in the wall. The physiological effects would be irregular blood pressure and probably the formation of a thrombosis. If the process does not reach an equilibrium and the wall continues to dissect then we have two potential outcomes. Firstly, in step III we see that the tear re-enters the arteries true lumen. This outcome is usually not deadly but the existence of the false lumen will continue to cause irregular blood pressure. The other possible outcome, step IV, is almost always deadly. In this scenario the dissection ruptures the outer wall, which will result in the patient bleeding out in mere minutes.

and B is the origin of the dissection in the aorta. This is important because the location of the dissection usually dictates the treatment of said dissection. Type A dissections originate in the ascending aorta or the aortic arch and usually require immediate emergency surgical intervention. Type B dissections, however, originate in the descending aorta and are usually treated medically at first, with surgical intervention coming later. Another important surgical classification technique for aortic dissections is the DeBakey classification [17], which divides the dissections into four types according to the location of the entry tear, as well as the extent of the dissection. Here Type I and Type II involve entry tears in the ascending aorta, with Type I propagating past the aortic arch along the descending

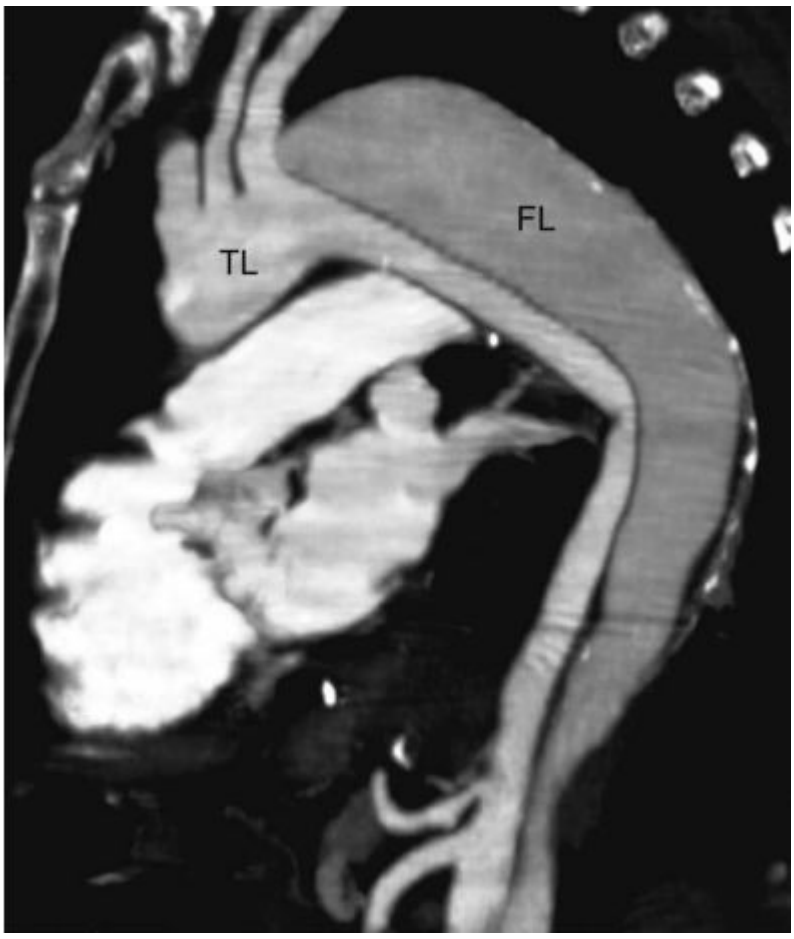


Figure 1.5: CT scan with 3D reconstruction shows a type B aortic dissection of the descending and abdominal aorta. Here we can see the development of a large false lumen, which has propagated the length of the aorta. TL - true lumen; FL - false lumen. This image was originally sourced from [96].

aorta. Type IIIa and Type IIIb involve entry tears originating in the descending aorta, with the distinction being that Type IIIa propagates further along the aortic wall than Type IIIb. An illustration of these classification systems can be seen in Figure 1.6.

The development of aortic dissections requires two pathological conditions: medical degeneration and mechanical wall stress [2]. This can happen at a young age in patients with Marfan Syndrome, Ehlers-Danlos syndrome or other connective tissue disorders [1, 2]. These diseases result in the blood vessel's wall lacking the regular mechanical properties of a regular blood vessel, making it easier for them to tear. In pregnancy, an increased blood mass raises blood pressure causing extra aortic wall stress. However, many pregnant people that develop an aortic dissection often have a connective tissue disorder [143, 71]. Atherosclerosis, which causes a build-up of lesions on the arterial wall, has long been associated with aortic dissections [8, 76].

In other patients aortic dissections can be caused by acquired diseases. A notable example is long term steroid usage disrupting the development of collagen fibres, weakening

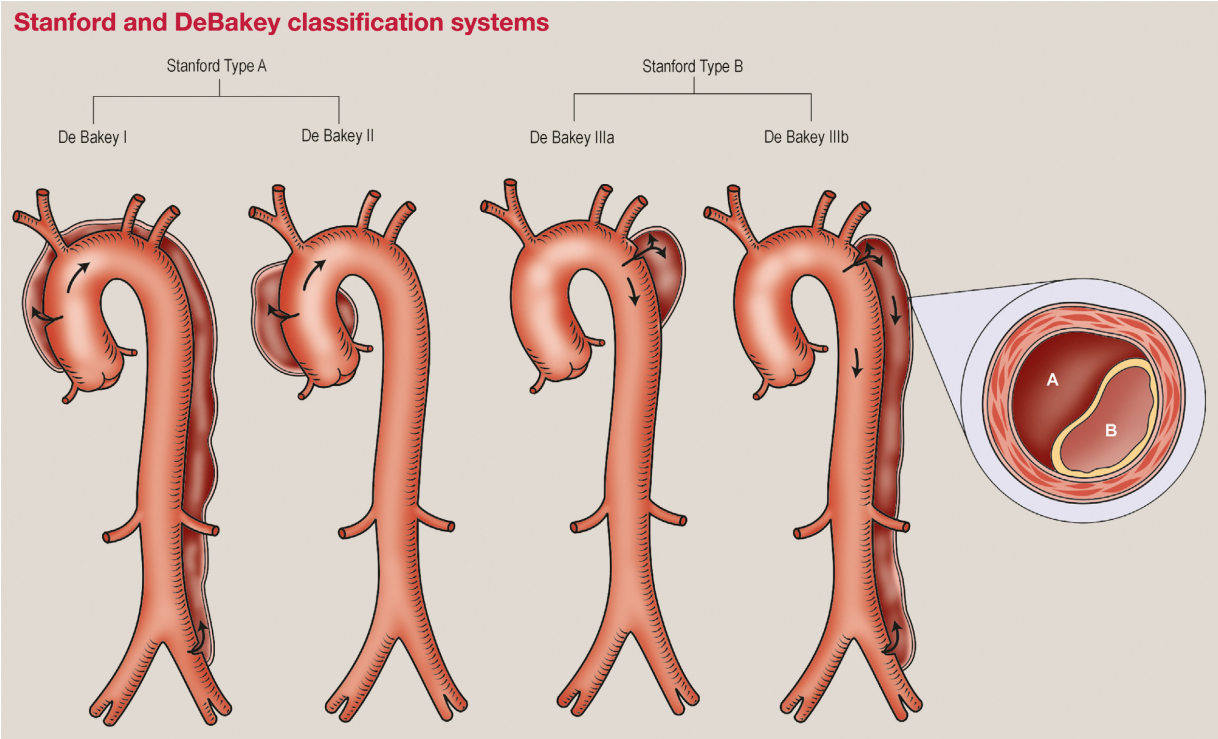


Figure 1.6: This is a diagram of different types of aortic dissections, each being labelled by the appropriate Stanford and DeBakey classifications. On the left we have two dissections originating in the ascending aortic (Stanford Type A). On the right both dissections originate in the descending aorta (Stanford Type B). Both sides illustrate how the DeBakey classification acknowledges the extent of propagation along the artery wall. In the above setup we have a cross section illustrating a circumferential slice of an aortic dissection, where one has that: A - false lumen, B - true lumen. This illustration has been sourced from [17].

the aortic wall and resulting in the development of aortic dissections [61, 120]. Hypertension, also known as high blood pressure, is a very common cause of aortic dissections [96]. The regular range of blood pressure in the aorta is 80 to 120 mmHg but when a patient is suffering from hypertension it can be greater than 180 mmHg [123]. This causes a much greater stress on the aortic wall and can lead to tearing especially in older patients [117]. Other risk factors, contributing to the likelihood of dissections, include smoking, cocaine usage, a bad diet and an unhealthy lifestyle [79, 124]. It seems obvious, but it is important to note unhealthy lifestyle choices can lead to the weakening of the aortic wall contributing to the risk of a dissection and many other cardiovascular diseases.

The development of theoretical models that allow us to accurately analyse and predict the causes of aortic dissections is a necessary step in developing new medical technologies. Aortic dissections are complex events influenced by multiple factors, including arterial wall mechanics, haemodynamics, and patient-specific conditions. Understanding these mechanisms through mathematical modeling can significantly improve clinical decision-making and patient outcomes.

A complete model of aortic dissections would enable the creation of a statistical emulation, using real patient data, to capture the full range of physiological scenarios. Such an emulation would encompass all possible outcomes and be readily accessible, allowing clinicians to quickly analyse patient data and predict likely prognoses. This approach could dramatically enhance the speed and accuracy of medical assessments, ultimately leading to better treatment strategies. This is the main motivation and long term goal of this field of research.

Achieving this goal requires first understanding the experimental findings on arterial mechanics, existing mathematical models of arterial behaviour, and methodologies in modern literature for developing multiscale models. By integrating these insights, we can develop a robust framework that not only explains the underlying causes of aortic dissections but also facilitates predictive analytics for improved patient care. In the next section we will examine the existing literature.

1.2 Literature review

The aorta has been widely researched by biologists, engineers, and mathematicians, creating a wide range of important results and literature. In this section, we name and discuss a series of key papers that cover a variety of theoretical and experimental results. We will cover important experimental results that revealed many of the aorta's mechanical phenomena. Next we will review a number of mathematical models created to explain arterial mechanics and the numerous mechanical properties arteries have, as discovered in experiments. This leads us on to a discussion of how researchers are now exploring modeling questions about aortic dissections, and more generally damage mechanics. Finally, in this section we will discuss asymptotic homogenisation, a multiscale modelling technique.

1.2.1 Experimental results

Early anatomical research into the properties of arteries involved studies relating the mechanical properties and the structure of the aortic wall. For example the work done by Glagov et al. found that the organisation of the media is responsible for much of an arteries mechanical properties [138, 137, 22]. One of the earliest mechanical observations made about the arterial wall is the compressibility of the wall, this was first noted via compressibility experiments performed by Carew [16]. Researchers have also long known that arteries do not obey Hooke's law and behave in a non-linear fashion [110]. An explanation for this non-linear behaviour is the natural anisotropy found in the microstructure of the aortic wall [136]. Anisotropy can be found in the variations of fibre dispersion in the arterial wall. Canham et al. [14] found with polarised light microscopy of arterial tissue

a significant degree of dispersion for collagen fibres in the adventitia and intima layers. Whereas, the media layer has fair less dispersion of collagen fibers in media layer [14].

Stress that exists in unloaded bodies is called residual stress. As previously mentioned residual stress is present in, many blood vessels including, the human aorta. The existence and importance of residual stress was first noted in the experimental research of Choung and Fung [43, 20]. In their experimental research they removed sectors of rat aorta's and cut them once along the radius. This results in the aorta blooming and creating an open sector, which releases most of the residual stress. The observed opening angle of the aorta is commonly used to quantify the residual stress. One can calculate the prestretch from the residual stress with a presumed constitutive law. In addition to radial and circumferential prestretch, it has been observed that arteries also have prestretch in the longitudinal direction [34]. This result was first observed by Dobrin, who measured a longitudinal shrinking in dog arteries once removed from the aorta. Dobrin concluded this elongation of arteries is a necessary effect preventing the tortuosity of arteries. Subsequent studies by Dobrin using elastase and collagenase to selectively remove structural components from the arterial wall, demonstrated that nearly all axial prestretch in healthy arteries is due to the presence of intramural elastin, not collagen [35]. Over time experiments studying residual stress in the human aorta have became more advanced. A systematic study sharing both quantitative and qualitative data by Holzapfel et al. [64] provides us with a detailed experimental methodology for measuring residual stress and strain. The main takeaway from that study is that each layer of the arterial wall, the intima, media and adventitia, all have their own characteristic residual stress and strain. We have also now concluded that the existence of residual stress and strain is due to the different growth rates of different components, in arterial walls [15].

Experimental research into the mechanisms of aortic dissection is extremely important, it provides us with insights about the underlying mechanics of the phenomena. For example, Babu et al. [6] studied samples of aortic tissue removed from patients with type A dissections. Using bi-axial stretch testing, it was observed that the stress strain curves of these tissue samples were much stiffer compared to a control group of healthy aortic tissue. They also found that the stiffness of these tissue samples did not correlate with the age of patients, or the diameter of the artery. Tsai et al. [127] performed an ex vivo experiment to study the impact of dissection size and number of dissections on the false lumen pressure. They found in each of their experimental models that the diastolic false lumen pressure was greater than the diastolic true lumen pressure. This was especially true when the entry tear is proximal and small with the re-entry tear being distal and occluded.

The effects of radial tear depth on the propagation pressure of aortic dissections was studied with an in vitro experiment by Tam et al. [126]. To do this 20 porcine thoracic

aortas were injected with a saline solution into the medial layer of the artery, creating an elliptical bleb. A circumferential slit was made on the intima side of each bleb, connecting the true lumen to the false lumen. From here each aortic true lumen was pressurised with saline solution, which in turn pressurised the false lumens. The pressure was then increased until each dissection propagated, the pressure this occurred at was taken to be the propagation pressure at that particular radial depth. The results of this experiment showed that the critical propagation pressure increased as the number of elastin layers in the wall increased. It was also found that the propagation pressure decreases linearly as the tear depth increases.

To better understand arterial mechanics researchers have studied and catalogued the mechanical properties of each constitutive part of the arterial wall. These measurements have been made with a number of different experimental approaches, by a variety of studies. The three most abundant components of the aortic wall are collagen fibres, elastin and vascular smooth muscle cells, which each play a very important mechanical role in arterial mechanics. A great appraisal of their material properties was collated by Camasão and Mantovanni [13], where they summarised and reviewed the work done by many researchers to understand the mechanical properties of arterial tissue. Researchers have found that components of vascular tissue have very different material properties. For example the Young's modulus of collagen, elastin and smooth muscles cells are $0.1 - 1$ GPa [37, 48], $0.3 - 1$ MPa [55, 80] and $0.1 - 2$ MPa [85, 13], respectively. It should be noted that measuring the correct specimen at the correct scale is of key importance. For example the respective mechanical properties of collagen change massively depending on whether one is discussing a collagen protein, fibril or fiber. The average Young's modulus decreases six fold from a single molecule (≈ 6 GPa) to the fibrillar scale (≈ 0.9 GPa) [47]. To gather this data researchers are performing a number of innovative tensile tests on individual cells and fibers, as outlined by Matsumoto et al. [86]. A single molecule is much stronger than a fibril for a simple reason. When a single molecule is pulled, it is already straightened and derives its strength from atomic forces. In contrast, a fibril consists of many molecules that must first unwind and straighten when stretched, making it more elastic. The same principle applies to fibers, which are even more elastic because the fibrils within them must also unwind and straighten during stretching.

1.2.2 Mathematical modelling of aortic mechanics

Accurately modelling arterial mechanics has been a very active field of research in biomechanics. Many researchers have attempted to create useful models to describe the mechanics of arteries. To discuss these models let us briefly introduce some notation from finite strain theory [56, 97]. Consider a fixed reference configuration Ω which is assumed to be stress free. The deformation gradient, a second order tensor, is represented as $\mathbf{F}$ which

transforms a material point $\mathbf{X}$ to a new point $\mathbf{x}$ during deformation. We define this by saying that $\mathbf{F} = d\mathbf{x}/d\mathbf{X}$, and further that the local volume ratio is defined by $J = \det(\mathbf{F})$, such that $J > 0$. With this notation we can define the Green strain tensor [97]

$$\mathbf{E} = \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbf{I}\right),$$

where $\mathbf{I}$ is the second order identity tensor and $\mathbf{F}^T\mathbf{F}$ is the right Cauchy-Green tensor [97]. From here we can define a strain energy function that represents the elastic potential in a material, with this potential one can derive the material stress and hence calculate material deformations under load. For example consider a linear elastic isotropic material, this can be described by the St. Venant-Kirchhoff [10] strain energy function

$$\Psi = \frac{\lambda}{2}\mathrm{tr}(\mathbf{E})^2 + \mu\mathrm{tr}(\mathbf{E}^2),$$

where λ and μ are the first and second Lame parameters [121], respectively. One can use this strain energy function to derive the associated second Piola-Kirchhoff stress

$$\mathbf{S} = \frac{\partial\Psi}{\partial\mathbf{E}} = \lambda\mathrm{tr}(\mathbf{E})\mathbf{I} + 2\mu\mathbf{E}.$$

The second Piola-Kirchhoff stress [10] as expected clearly describes a linear stress-strain relationship. However, arterial mechanics is dictated by non-linear mechanics and we therefore must consider more complex forms of strain energy functions. Before we do let us introduce some useful notation:

$$\bar{\mathbf{C}} = \left(J^{-\frac{2}{3}}\right)\mathbf{F}^T\mathbf{F} \quad \text{and} \quad \bar{\mathbf{E}} = \frac{1}{2}\left(\bar{\mathbf{C}} - \mathbf{I}\right) \tag{1.1}$$

the modified right Cauchy-Green tensor and modified Green-Lagrange strain tensor, respectively. This is commonly done because it allows for easier analysis of both dilation and distortion parts of material deformation. How this is done is by considering a multiplicative decomposition of the deformation gradient, $\mathbf{F}$. Consider the following multiplicative decomposition

$$\mathbf{F} = \left(J^{1/3}\mathbf{I}\right)\hat{\mathbf{F}}, \tag{1.2}$$

where $\hat{\mathbf{F}}$ represents the distortion of Ω at a constant volume, whereas the term $J^{1/3}$ represents the dilation of Ω [41, 98]. Remember that by definition $J = \det(\mathbf{F})$ and physically we may interpret this as the ratio of change in volume, hence if we imagine an idealised cube, then the cube root of this term gives us the ratio of change in strain. We can see how (1.2) can be simply recovered from the terms in (1.1) when one performs an analysis of the system.

In 2000 Holzapfel et al. [65] wrote a systematic review of the state of the art, analysing

the adequacy of a number of new constitutive laws for the passive response of an arterial wall. To do this a number of Strain energy functions were used to calculate the stress distribution of a thick walled circular cylindrical tube, as an approximate arterial geometry. The tube was subjected to combinations of stretching, inflation and torsion whilst having an already incorporated residual stress. This analysis showed that each of these popular strain energy functions could be improved upon for a variety of reasons, as we will mention. Briefly, we will discuss a couple of these strain energy functions before looking at the popular Holzapfel-Gasser-Ogden model proposed at the end of the analysis presented by Holzapfel et al. [65].

One of the earliest proposed strain energy functions is that by Delfino et al. [30], created to model the mechanics of carotid arteries. The proposed potential was

$$\Psi = \frac{a}{b}\left(\exp\left[\frac{b}{2}\left(\bar{I}_1 - 3\right)\right] - 1\right), \tag{1.3}$$

where $a > 0$ represents a parameter with dimensions of stress and $b > 0$ is a non-dimensional parameter used for strain-stiffening. Here $\bar{I}_1$ is the first invariant of the modified right Cauchy stress tensor $\bar{\mathbf{C}}$ defined as $\bar{I}_1 = \mathrm{tr}(\bar{\mathbf{C}})$. This strain energy function is able to model the typical stiffening effects one would expect in higher pressure domains. However, it is noted that this strain energy functions is isotropic, which does not comply well with the highly anisotropic multilayered human aorta. To address this issue many researchers turned to a strain energy function first introduced by Fung et al. [42]. This was first used by Humphrey et al. [69] and phenomenologically represented the anisotropy of arterial tissues through its material parameters. It is suitable for three dimensional deformation and has the form:

$$\Psi = \frac{1}{2}c\left[\exp\left(Q\right) - 1\right], \tag{1.4}$$

where c is a stress-like parameter and Q is given by

$$\begin{aligned} Q =& b_1\bar{E}_{\Theta\Theta}^2 + b_2\bar{E}_{ZZ}^2 + b_3\bar{E}_{RR}^2 + 2b_4\bar{E}_{\Theta\Theta}\bar{E}_{ZZ} + \\ & 2b_5\bar{E}_{ZZ}\bar{E}_{RR} + 2b_6\bar{E}_{RR}\bar{E}_{\Theta\Theta} + b_7\bar{E}_{\Theta Z}^2 + b_8\bar{E}_{RZ}^2 + b_9\bar{E}_{R\Theta}^2. \end{aligned}$$

Here b_i, $i = 1, ..., 9$, are non-dimensional numbers and $\bar{E}_{ij}$ is the modified Green-Lagrange strain tensor, referred to the cylindrical coordinates (R, Θ, Z). In the work of Fung [42] there are no restrictions on the choice of material parameters b_i, in fact these parameters appear to be without any physical meaning. Their purpose is unclear as discussed by Ogden et al. [65] and Fung et al. [44]. They are generally chosen via non-linear curve fitting, they can be difficult to calculate and non-uniqueness can become a problem as non-convexity is not guaranteed in unconstrained parameter optimisation. Ogden et al. [65]

showed that the convexity, which is necessary to guarantee realistic results and efficiency of numerical computation may not be satisfied for some choices of b_i.

Similarly, Vaishnav et al. [128] introduced a two-dimensional description for the deformation behaviour of the canine thoracic aorta using polynomial expressions. This classic modelling approach presented three polynomial expressions with 3, 7 or 12 material parameters. In [128] it was found that the three parameter model was too inaccurate and the twelve parameter model did not have any significant advantage over the seven parameter model. So for reference we introduce the seven parameter model as

$$\begin{aligned}\Psi =& c_1\bar{E}^2_{\Theta\Theta} + c_2\bar{E}_{\Theta\Theta}\bar{E}_{ZZ} + c_3\bar{E}^2_{ZZ} + c_4\bar{E}^3_{\Theta\Theta} \\ &+ c_5\bar{E}^2_{\Theta\Theta}\bar{E}_{ZZ} + c_6\bar{E}_{\Theta\Theta}\bar{E}^2_{ZZ} + c_7\bar{E}^3_{ZZ},\end{aligned} \tag{1.5}$$

where c_i, $i = 1, ..., 7$, are material parameters with dimensions of stress and $\bar{E}_{\Theta\Theta}$ and $\bar{E}_{ZZ}$ are the components of the modified Green-Lagrange strain tensor in the circumferential and axial directions, respectively. Due to the cubic nature of this strain-energy function, it is not convex for any set of values of the material constants. In fact, Fung [44] showed that two different sets of $c_1, ..., c_7$ produced the same mechanical response of the same artery quite well. This model's lack of uniqueness can be problematic. It should also be noted that this model assumed symmetry in the radial direction, which can not be assumed when considering the complex multilayered aortic wall. Also the aorta is not symmetrical in the radial direction and assuming so would be a gross simplification.

In Holzapfel et al. [65] we are introduced to the Holzapfel-Gasser-Ogden strain energy density function more colloquially known as the HGO model. This is by far the most widely used strain energy function in the field of arterial mechanics, it is also used to model rubber-like polymer materials. This model accounts for residual stress, anisotropy, the differing layers of the arterial wall and is motivated histologically, depending on a set of material parameters. The basic idea of the HGO strain energy density function is to assume each arterial layer has similar mechanics and can hence all be be described by a strain energy function of the same form. Then by considering an additive split of the potential Ψ, we can consider the separate forms of Ψ_{iso} and Ψ_{aniso}. Here Ψ_{iso} is associated with the isotropic part of the deformation and Ψ_{aniso} is associated with anisotropic part of deformation. We associate the mechanical response of the non-collagenous parts of the arterial wall with Ψ_{iso}, which is dominant in a low pressure regime and assumed to be isotropic and neo-Hookean. In higher pressure regime the mechanical response of the collagenous parts of the arterial wall dominates the mechanics and are described by Ψ_{aniso}, an exponential anisotropic strain energy function which only only takes load in tension.

To understand the model consider an incompressible fibre reinforced material with two sets of helically woven fibers, the direction of which are described by the unit vectors $\mathbf{A}_i$ in the fixed reference configuration Ω, for $i = 1, 2$. Then one can write the HGO strain

energy function as

$$\Psi(\bar{I}_1, \bar{I}_4, \bar{I}_6) = \Psi_{\text{iso}}(\bar{I}_1) + \sum_{i=4,6} H(\bar{I}_i)\Psi_{\text{aniso}}(\bar{I}_i), \tag{1.6}$$

where H is the Heaviside function, only allowing fibers to take stretch loads, defined as

$$H(x) = \begin{cases} 0 & x \leq 0 \\ 1 & x > 1. \end{cases}$$

Here we define Ψ_{iso} and Ψ_{aniso} as

$$\Psi_{\text{iso}}(\bar{I}_1) = \frac{c}{2}\left(\bar{I}_1 - 3\right) \quad \text{and} \quad \Psi_{\text{aniso}}(\bar{I}_i) = \frac{k_1}{2k_2}\left(\exp\left[k_2(\hat{I}_i - 1)^2\right] - 1\right), \tag{1.7}$$

where $c > 0$ and $k_1 > 0$ are parameters with dimensions of stress and k_2 is a dimensionless strain-stiffening coefficient. These potentials depend on the invariants $\bar{I}_1$, $\bar{I}_4$ and $\bar{I}_6$ which we define as:

$$\bar{I}_1 = \text{tr}(\bar{\mathbf{C}}), \quad \bar{I}_4 = \bar{\mathbf{C}} : \mathbf{A}_1 \quad \text{and} \quad \bar{I}_6 = \bar{\mathbf{C}} : \mathbf{A}_2. \tag{1.8}$$

Using the HGO strain energy function one can calculate the Cauchy stress of each individual layer of the arterial wall with a dependency on residual stress. An example of this is given in Figure 1.7 illustrating the differences in stress for an arterial wall with and without residual stress.

Gasser et al. [46] provided the first major update to the HGO model by considering the dispersion of fibers in the artery wall. Here dispersion means the dispersion of collagen fibers against local orientations in the artery. The regular HGO model works very well at modelling the media layer where there is little dispersion of tissue fibers, but fails to account for the effects fiber dispersion has on the mechanics of the adventitia and intima [46]. Dispersion of fibers dictates the anisotropy of the arterial wall, to that end Gasser introduced the parameter κ which measures the "degree of anisotropy", or in other words the degree of dispersion. The value of κ can be deduced experimentally or calculated from a density function, $\rho(\mathbf{A})$ which represents the normalised number of fibres with orientations $\mathbf{A}$, in a small interval. Using the density function Gasser explained how one can calculate a second order structural tensor which quantifies the orientation of fibres as

$$\mathbf{M} = \frac{1}{4\pi}\int_{\omega} \rho(\mathbf{A}(\Theta, \Phi))\mathbf{A}(\Theta, \Phi) \otimes \mathbf{A}(\Theta, \Phi)\, d\omega \tag{1.9}$$

where ω is a unit sphere. Note we introduced the two Eulerian angles $\Theta \in [0, \pi]$ and $\Phi \in [0, 2\pi]$, this is done because $\mathbf{A}$ is a unit vector and can be defined simply by the direction it is pointing in (in other words no radial dependence). To simplify the model Gasser defines

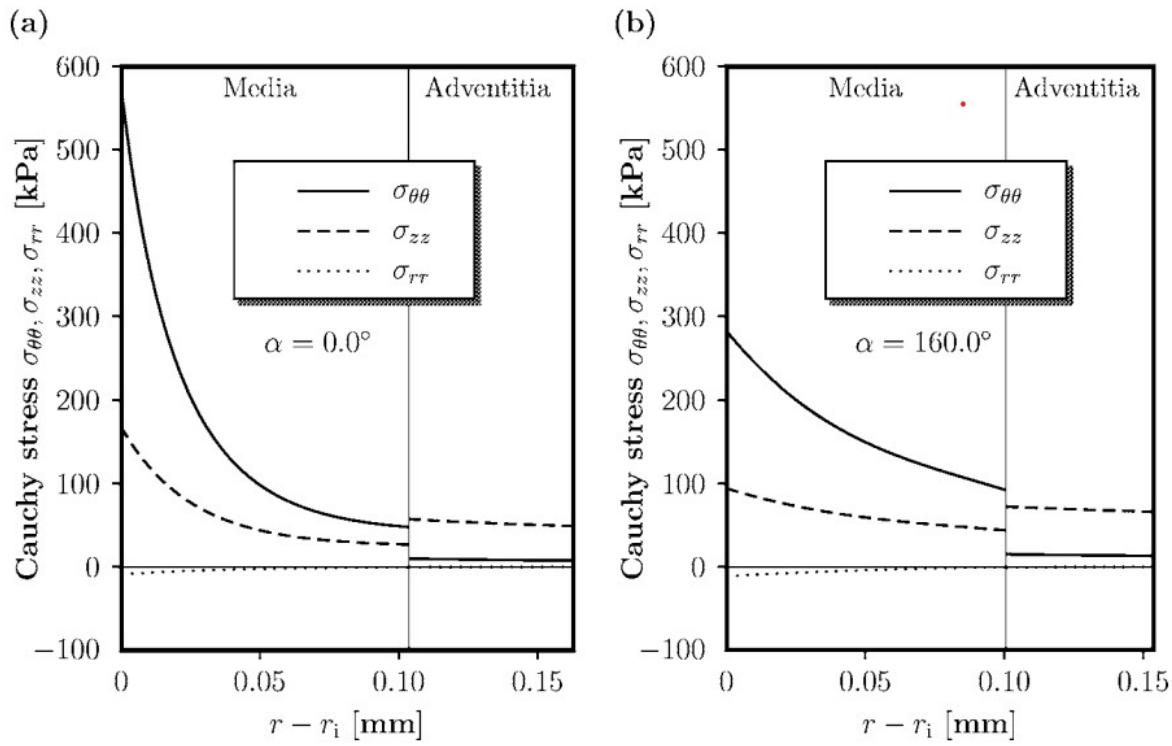


Figure 1.7: Plots of the principal Cauchy stresses $\sigma_{\theta\theta}$, σ_{zz}, σ_{zz} in the circumferential, axial and radial directions of the aortic wall. Stresses where calculated for the deformed media and adventitia layers under a physiological pressure of $p = 13.33\,\text{kPa}$ and an axial stretch of $\lambda_z = 1.7$. Here we can see two states: (a) without and (b) with residual stress. $r - r_i$ is the distance of a point in the arterial wall from the inner radial surface at r_i, and α is the opening angle with respect to the centre of the artery. The stress distribution is significantly lower in (b) where the artery has residual stress compared to (a) which has no residual stress. This figure is Figure 18 from Holzapfel et al. [65].

a pair of preferred referential direction vectors called $\mathbf{a}_i$ for $i = 1, 2$. With these preferred vectors one can represent two orientations of fibers representing two helically woven layers of fibers. Then following Gasser and using (1.9) one can calculate

$$\mathbf{M}_i = \kappa\mathbf{I} + (1 - 3\kappa)\mathbf{a}_i \otimes \mathbf{a}_i \quad \text{for} \quad i = 1, 2. \tag{1.10}$$

Here Gasser introduces the structure parameter κ that describes the dispersion of fibers. With $\mathbf{M}_i$ the stretch of fibres can be quantified by the invariants:

$$\hat{I}_4 = \bar{\mathbf{C}} : \mathbf{M}_1 \quad \text{and} \quad \hat{I}_6 = \bar{\mathbf{C}} : \mathbf{M}_2.$$

Using these invariants we can account for the effects of fibre dispersion in the HGO model, defined by the potential in equation (1.6). It should be noted that this model reduces to the original HGO model for $\kappa = 0$. The model describes an isotropic distribution of fibres for $\kappa = 1/3$. In this model the invariants $\hat{I}_4$ and $\hat{I}_6$ characterises the strain in the direction

of the mean orientation $\mathbf{a}_i$ of the i^{th} family of fibres.

1.2.3 Mathematical modelling of aortic dissections

Mathematical modelling of aortic dissections possess several interesting mathematical challenges and is of great interest to applied mathematicians, engineers and clinicians alike. To model aortic dissections, one must be able to explain the regular mechanics of the aorta, as previously discussed in §1.2.2 this can be done with a variety of continuum models. From there one must establish a modelling technique for the dissection of the aorta wall itself, this is a form of material failure. Researchers have approached this problem with a variety of modelling methods. All these methods are a form of damage mechanics, a field dedicated to modelling tear propagation, material failure and damage accumulation. Generally, to model material failure one models the mechanics as a continuum or discrete system which is then coupled to a damage model. The damage model alters the mechanical model when certain material conditions are met which can results in a change of material properties, a change of material geometry or a combination of both. For a more general review of damage mechanics and many different modelling techniques of material failure we would recommend the textbooks [132]. Here we will discuss a series of interesting approaches, in the literature, taken to model aortic dissections.

The energy release rate (ERR) method was created by Griffith [51] the earliest attempt at modelling material failure, the original theory was then later further developed by Irwin [74]. This material failure criterion is an energy-based method that requires one to calculate the change in potential energy of a creak/tear surface as it is extended by an infinitesimal surface, we call this the energy release rate G. Next one must define the material parameter G_c, the critical energy (or Griffith energy) required to destroy all the bonds at the crack tip. If $G > G_c$ then we have propagation of the tear tip, this is done numerically by modifying the mesh geometry. If $G < G_c$ then the tear is stationary and there is no change to the material geometry. For a guide on how to implement this model we recommend Zehnder et al. [144] which covers many numerical methods for calculating G. Lei Wang et al. [134] used this method to study how arterial dissection propagates along the arterial wall. By coupling this failure criterion with a HGO strain energy density function, they were able to study how inflation of a false lumen propagates a dissection. The study concluded that the existence of fibres gives the wall more strength and slows down the ERR, in fact that fibres aligned parallel will decrease the ERR most. Another interesting conclusion is that the propagation of a tear will only halt when the surrounding connective tissues have a sufficient stiffness. This result was obvious and made clear physiological sense, it mainly serves to support the physical realism an ERR model can have.

One of the simplest approaches is to make use of a cohesive traction separation law as

a criterion for material failure. Cohesive laws are widely used in numerical simulations to approximate tear propagation through a material continuum. The purpose of a cohesive law, as a constitutive law, is to relate the normal and tangential forces on the material surface at the tip of a tear to a failure criterion. When this criterion is satisfied, a material discontinuity is enforced with a jump in displacement, emulating a discontinuity in the material. A cohesive law requires at least two of the following three parameters: the maximum traction before tearing, the maximum displacement jump, and the fracture energy. The fracture energy is the area under the curve of traction against displacement at the tip of the tearing surface. Elices et al. [39] demonstrated that the shape of this curve can have a significant effect on the outcomes of simulations of material failure, expressing that careful experimentation is required to determine the values of parameters related to the cohesive parameters. Originally, this model was developed to study cracks, i.e. tears, in brittle materials like concrete; however, researchers have recently worked to incorporate this type of model into studies of arterial dissections. Gasser et al. [45] used the model to numerically analyse fractures in biological tissue, namely they modelled a peeling test on an arterial strip under an external load. A peeling test is when we measure the force required to separate the adhesive bonds which hold together layers of a laminated material [77]. In Gasser's case they were numerical testing the required force to dissect an arterial strip from the media and comparing the results to other experiments. Volokoh et al. [131] also coupled a cohesive law with the HGO strain energy model and found that the overall arterial strength is dominated by the medial layer and the residual stress in the arterial wall. Volokoh et al. [130] also found that the residual stress dominates the overall arterial strength when modelling with a Fung type description of the arterial mechanics. Another interesting result found using this approach was by Wang et al. [135] who found when describing the arterial wall with the anisotropic hyperelastic HGO model that tearing occurs preferentially along the material axes with the greatest stiffness.

Another modelling technique for material failure is the damage phase field method. This method finds its roots in the works of Miehe [90] and Marigo [83]. This is a relatively new approach that approximates material damage, and ultimately failure, through the introduction of a non-dimensional diffusive variable called the damage phase field, α. It should be noted that the damage phase field method shares a lot of similarities with the gradient damage model, their main difference being one of interpretation [11]. To use this method, one must firstly define the damage phase field α, a non-dimensional variable with a range from 0 to 1. Where α equals 0 means the material at that point in space is completely healthy and undamaged, whereas at damage equals 1 the material has fully ruptured at that point. In this model the damage is constrained such that it is only monotonically increasing from 0 to 1. In other words, we constrain α such that one has $\dot{\alpha} \geq 0$. Here the dot above α indicates a derivative with respect to time. To derive a

model with this method one considers all the material energy due to the material potential energy and the energy driving a tear. By defining a material potential Ψ that depends on both the displacement of the material and the damage in the material we can couple the material deformation to the material damage growth. To ensure these equations describe smooth continuous fields we approximate the shape of the damage phase field as a diffusive variable. In other words, damage is not seen as a sharp crack through the material but instead as a diffusive variable approximated over some local length scale $\ell \ll 1$. To do this we introduce a constraint on the shape of the damage phase field surface in terms of α. This results in α being described by a concentrated diffusive field and allows one to derive a system of partial differential equations. To do this one considers a variational formulation of the material energy. As a result, one can derive an equation of motion and an energy balance equation, the former describes the material deformation, and the latter describes the evolution of α. With this system of partial differential equations one can simulate a wide variety of possible scenarios and predict damage initiation, location, branching, changes in geometry due to change. One can also study tear propagation and arrest in a material due to external loading. With all these useful properties researchers have used the damage phase field method to describe tearing in biological tissue. Raina et al. [108] proposed how one could adapt a damage phase field approach to describe incompressible biological materials. In Chapter 2 we will go into greater detail about the damage phase field method and rigorously explain how to derive a model for a complex arbitrary geometry.

1.2.4 Multiscale modelling with asymptotic homogenisation

Real-world physical systems are typically multiscale in nature. Their microstructures are characterised by heterogeneous constitutive parts and complex geometries, all of which contribute to the system's overall macroscopic behaviour. We can think about this in several ways.

For example, consider water. A mathematician might model it as a liquid—more precisely, as a continuum—completely ignoring its complex microstructure. However, if we examine water through an electron microscope, we see that it consists of countless molecules. Another example is wood. From a human perspective, it appears as a solid block, but under a microscope, we can observe that it is composed of layers of adhesive cellulosic fibres. These examples highlight the importance of carefully choosing the appropriate scale when modelling a material.

In many mechanical models, we adopt a macroscopic perspective because the available data is phenomenological, derived from laboratory experiments that measure the properties of entire structures rather than the individual constituents of a material. This approach can overlook the critical interplay between a material's microstructure and its

macroscopic response to external effects. However, with modern mathematical techniques, we can now develop models that incorporate the role of a system's microstructure. These techniques include representative volume elements [125], asymptotic homogenisation [63, 103], mixture theory [119] and other advanced methods.

In this thesis, we will make extensive use of asymptotic homogenisation. In this subsection, we will explore the motivation behind this technique and review the broader literature on the subject. First, however, we shall examine a mathematical example to motivate the need for a multiscale model.

To illustrate this, we consider the one-dimensional diffusion-type boundary value problem, given by:

$$\frac{d}{dx}\left(D(x)\frac{du}{dx}\right) = f(x); \quad 0 < x < 1, \tag{1.11}$$

$$u(0) = a, \quad u(1) = b \quad \forall a, b \in \mathbb{R}, \tag{1.12}$$

where (1.12) represent non-homogenous Dirichlect boundary conditions. Here $f(x)$ is a known spatially varying volume source, $D(x)$ is the smooth, strictly positive, spatially varying diffusion coefficient, and $u(x)$ an unknown scalar field. We shall assume that $f(x)$ is regular enough such that a unique solution for $u(x)$ exists. Boundary value problems described by (1.11) and (1.12) are quite common in scientific literature and describe a wide panorama of physical phenomena such as: diffusion of pollutants; the temperature distribution for heat conduction; the linear elastic displacement of an elastic string.

We are interested in investigating the solutions of $u(x)$ when the diffusion coefficient $D(x)$ exhibits *multiscale* spatial variations. In other words when the observed behaviour of $D(x)$ changes depending on the resolution we take into account. An example of this is highlighted in Figure 1.8, where a representative solution of (1.11) and (1.12) on the *macroscale* can be seen over the full domain. In a small box we have zoomed in to a smaller *microscale* where one can clearly observe a radically different behaviour compared to the macroscale. In fact, the behaviour clearly has a dependence on local spatial scales when zoomed in.

From Figure 1.8 we can conclude that there exists a clear separation of spatial scales. The problem clearly holds over the full domain and has a solution u dependent on x. However, we are interested in separating the macroscopic and microscopic spatial variations (as is illustrated in Figure 1.8). To do this we can introduce a macroscopic characteristic length scale L (defined as 1 in the above example) and a microscopic length scale δ (defined as 4×10^{-3} in the above example. By introducing these characteristic length scales we can now non-dimensionalise the spatial coordinate x with respect to both the microscale δ and macroscale L:

$$x = Lx_M = \delta x_m. \tag{1.13}$$

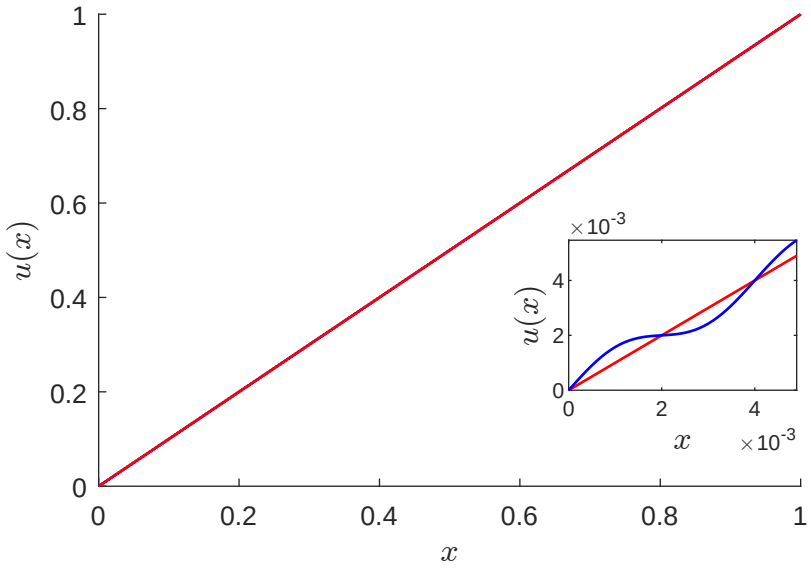


Figure 1.8: The exact solution $u(x) = \left(x + c\epsilon \sin(\frac{x}{\epsilon})\right) / \left(1 + c\epsilon \sin(\frac{1}{\epsilon})\right)$ of the boundary value problem (1.11) with boundary conditions (1.12) defined by $a = 0$, $b = 1$, with problem coefficients $D(x) = (1 + c\cos(x/\epsilon))^{-1}$ and $f(x) = 0$. Here we have set $c = 0.9$ and $\epsilon = (2/\pi) \times 10^{-3}$. The exact solution is defined over the microscale and a macroscopic solution $u(x) = x$ is defined over the macroscale. Notice that over the whole domain $[0, 1]$ that we cannot distinguish between the two scales.

Here x_M represents a non-dimensional coarse scale mapping, as it is non-dimensionalised with respect to L. Whereas, x_m represents a non-dimensional fine scale mapping, as it is non-dimensionalised with respect to δ. These spatial coordinates are related by (1.13) with the relation

$$x_m = x_M/\epsilon, \tag{1.14}$$

where we define

$$\epsilon = \frac{\delta}{L}. \tag{1.15}$$

The small parameter ϵ is a ratio between the micro and macroscales, it is used to measure their separations.

With this notation now introduced we can make an informal argument. The example given in Figure 1.8 has an obvious dependence on ϵ in the complete solution over the microscale. However, the microscale solution does in fact converge to the macroscale solution as ϵ goes to zero. In other words as the scale separation gets increasingly larger the microscopic and macroscopic solutions are indistinguishable. This is the key argument in asymptotic homogenisation. That any physical problem with a necessary large separation of length scales, $\epsilon \ll 1$, has a macroscopic solution separable from any microscopic spatial dependency. However, in using asymptotic homogenisation we are also able to incorporate data about the microscale into a macroscopic solution. This is usually done via a set of effective macroscopic coefficients that depend on solutions to linear local cell problems,

which are used to calculate a series of ansatz which define key relations between the micro and macroscale of the problem.

In order to use asymptotic homogenisation a series of key assumptions must be satisfied. We will discuss these in detail in Chapter 3 when we apply the method to create a multiscale model. For a complete and rigorous introduction to asymptotic homogenisation we recommend the works by Penta et al. [103] and Holmes [63]. Here we will just mention the basic assumptions necessary to use asymptotic homogenisation. The first assumption, as already mentioned, is a large scale separation ($\epsilon \ll 1$) between distinguishable micro and macroscales. With this assumption one assumes that any unknown field can have its spatial dependency decoupled into two independent variables describing the macroscale and microscale, as seen in (1.13). Then one assumes that every unknown field can be described by a power series expansion in factors of ϵ. Of course we also must assume that every unknown field is locally bounded and regular.

With these assumptions in one dimension one can derive a multiscale model of most problems [103]. However, for problems in higher dimensions we require another assumption which is much more important. The previous assumptions can be safely made in most physical systems where a large length scale separation is observed. For higher dimensions we need to be able to satisfy an assumption of *local periodicity*. In other words we need a local microstructure that is periodic. Without it we cannot upscale using asymptotic homogenisation. In the following section we will discuss how we can safely make that assumption for an arterial wall.

Now that we have covered the motivation for upscaling both physically and mathematically, as well as the necessary assumptions for asymptotic homogenisation, we shall cover the literature of the method. Exploring the methods origin, best sources and the most interesting modern research.

The roots of asymptotic homogenisation can be traced back to the works of Sanchez-Palencia [114] and Bensoussan [9], published in the early 1970's. These foundational studies established a method for upscaling systems of partial differential equations that describe periodic media into a simplified set of equations. This approach effectively incorporates information about the medium's local geometry and spatially varying material coefficients into a series of effective macroscopic coefficients. Over the following decades, the applications of this methodology have expanded significantly.

One of the earliest applications of multiscale modelling with asymptotic homogenisation was in the mechanics of composite materials [18, 104, 105, 109]. A composite material consists of multiple constitutive components. In a laboratory setting, one might take a macroscopic perspective, treating the material as a single homogeneous structure, conducting experiments, and developing a purely phenomenological understanding of its mechanics. However, from a multiscale modelling perspective, it is much more effective

to start at the microscale. The challenge, this presents, is that this approach generates an immense number of coupled partial differential equations—on the approximate order of magnitude of one over epsilon, with epsilon being very small. Using asymptotic homogenisation, researchers have successfully upscaled these large systems into a simplified set of equations that describe the mechanics of a single macroscopic material. This was first achieved by Bensoussan et al. [9], who wrote a foundational text which systematically introduces asymptotic homogenisation for partial differential equations, focusing on the applications of homogenisation theory. The authors explored how materials with periodic structures can be analysed using asymptotic homogenisation to derive effective macroscopic properties from microscopic behaviours.

In the broader literature, asymptotic homogenisation has been applied to upscaling solid mechanics across a variety of contexts, including finite strain theories, which typically rely on non-linear mechanical models. Notable examples include the works of Yang et al. [141] and O'Dea et al. [24, 62], who studied thermoelastic composites and tissue growth, respectively. These studies, along with others in the field, highlight an important consideration: a non-linear model does not necessarily allow for a complete separation of micro- and macroscales.

When performing asymptotic homogenisation, power series approximations of each field of interest and the separation of scales typically linearise the problem, allowing for an upscaled macroscopic model that is independent of the microscale. However, this is not always the case. For instance, Miller et al. [93] investigated balance equations for non-linear poroelastic composites and found that their non-linear microscopic model was not adequately linearised by asymptotic homogenisation. As a result, their macroscopic model retained dependencies on the microscale, and vice versa. This underscores a key lesson: to achieve an effective macroscopic model that is independent of microscopic variables, the microscopic model must be one that is properly linearised through the process of asymptotic homogenisation.

Of course, asymptotic homogenisation does not only apply to solid mechanics, but it is also used often to model fluid mechanics [23, 113]. One famous example is a new derivation of Darcy's law [102, 103] which gives the previously empirical law a solid theoretical explanation. This model transformed a microscopic model of Stoke's law into a macroscopic model of Darcy's law. This is an example of how multiscale modelling can explain differing mechanics at different scales.

The upscaling of fluid mechanics has garnered increasing interest in the field of metamaterials — man-made materials engineered for specific properties. In solids, this includes alloys and composites, among others [140]. More intriguingly, meta-fluids [122] have growing theoretical applications, particularly in the study of liquid crystals, which can be controlled using electromagnetic fields or lasers. This means we can apply a virtual energy

source to a liquid crystal, manipulate its motion, or even catalyse a chemical reaction. The primary research application of this technology is in medicine, where liquid crystals can be used to develop drugs and antibiotics [94].

Understanding this phenomenon through asymptotic homogenisation is essential. Fluids in the human body present an extremely complex modelling challenge, involving fluid-structure interactions, intricate microscopic geometries, and multiple interacting fluid components. At the microscale, such a model would be computationally infeasible. However, by upscaling and creating a macroscopic model, we significantly reduce the required resolution for numerical solutions. This is because the microscopic geometry is effectively smoothed out, making calculations more manageable. Additionally, each effective macroscopic coefficient depends only on macroscale variables, further reducing computational complexity while maintaining accuracy.

Macroscopic models also allow for a more comprehensive sensitivity analysis of each parameter, helping researchers understand how variations in microstructure influence macroscopic outcomes. This ability to bridge scales and extract meaningful insights is the primary advantage and motivation behind multiscale modelling.

In Chapter 3 we shall make extensive usage of asymptotic homogenisation and introduce new analytical techniques to upscale a model of material failure. In the following section we will introduce our modelling framework, thesis aims and outline the content of each chapter.

1.3 Thesis framework

Our overarching goal is to develop a comprehensive multiscale model of aortic dissection. However, achieving this in its entirety is beyond the scope of this thesis. Instead, we focus on constructing a multiscale model of material failure, which represents the first rigorous multiscale model of a damage phase field. Later in this thesis, we will demonstrate that this multiscale model converges with its microscopic counterpart. Achieving this will necessitate the development of new numerical methods for solving damage phase field problems.

In this section, we outline the general modeling approach used to create a multiscale model of material failure, with applications to aortic dissections. We also briefly discuss how to establish a microscopic representation of the aortic microstructure that meets the local periodicity requirements of asymptotic homogenisation [103]. Finally, we present the aims of this thesis and summarize the content of each chapter.

1.3.1 General modelling approach

In this thesis we will create a multiscale model of material failure. To do this we need to model from a microscopic perspective and upscale to a macroscopic perspective. Our method is to model material failure of an arbitrary composite microstructure with the damage phase field method and then upscale, using asymptotic homogenisation. This general microstructure can be representative of the aortic microstructure. In recent years, much multiscale modelling has been performed using asymptotic homogenisation to model complex materials in a wide variety of contexts [104, 32]. Asymptotic homogenisation is already used in many biological setups due to the hierarchical structure of biological tissue, and it naturally fulfils the many necessary prerequisites for appropriate usage of asymptotic homogenisation [23, 109]. However, to the best of this author's knowledge, no one has done this while modelling with the damage phase field method. If we describe the composite microstructure as an elastic composite structure then we can model material failure from that microscopic perspective. With this approach we can use the damage phase field method to model the material failure that occurs in the microstructure during a tear. This will require using a variational formulation to derive a set of partial differential equations which describe the linear momentum balance and damage evolution.

With this setup we can assume a vast disparity in length scales between the elastic composites macro and microscales. We may define a representative length scale of the macroscale to be L. We may also define the representative length scale of the composite microstructure, defining the size of a representative microscopic unit called δ. This setup is illustrated with a cylindrical example in Figure 1.9. Assuming L to be much larger that δ allows us to construct the non-dimensional ratio

$$\epsilon = \frac{\delta}{L} \ll 1.$$

As $\epsilon \ll 1$, then we can clearly say that the macro and microscales are well separated. Asymptotic homogenisation would allows us to exploit the smallness of ϵ and derive effective macroscopic equations. Except when upscaling the damage phase field we will see that certain conditions must be met to ensure irreversibility constraints. This motivates us to take a new approach with asymptotic homogenisation in order to satisfy these constraints. We demonstrate how assumptions about local periodicity in the microstructure allow us to upscale with a number of inequality constraints in the microscopic model. To the best of our knowledge this modelling approach is novel. As a result we produce a macroscopic model which embeds plenty of information from the microstructure and still satisfies the irreversibility constraints of the damage phase field method. This model allows us to study the effects of the microstructure on the macroscopic results. This is done via a set of effective macroscopic coefficients. We calculate the coefficients by solving a set of linear

periodic cell problems for each position in our macroscopic system. These cell problems account for the geometry and parameter values of the aortic microstructure, embedding them into the macroscopic system.

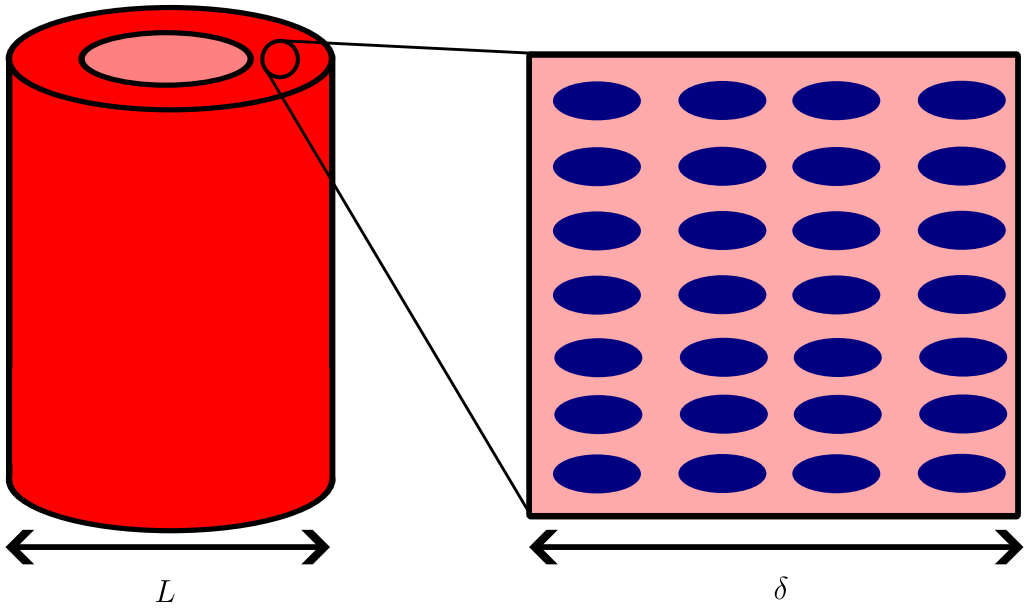


Figure 1.9: Diagram of the two different length scales of a macroscopic cylinder with a periodic microscale. On the left we can see a large macroscopic cylinder with a length scale of L, from a macroscopic perspective this cylinder looks homogeneous. On the right we have a zoom in of the cylinder surface, defined by the length scale δ. Here the cylinder is clearly not homogeneous and in fact a composite structure. This phenomena occurs often in nature especially with soft tissue, with a length disparity such that $L \gg \delta$.

By solving our macroscopic model numerically we will demonstrate that the microscopic and macroscopic models converge as $\epsilon \to 0$. This can be seen with a simple one-dimensional numerical solution to our multiscale model. Using the finite element method to solve our numerical model in higher dimensions will allow us to simulate material failure in higher dimensions. In each case we will need to give special consideration to the constraints in our multiscale model for our numerical solution to be accurate.

Using our macroscopic system, we can study the effects that various changes of the microstructure have on the outcome of macroscopic material failure. We demonstrate this through an example of a two-dimensional trouser test at the end of Chapter 5, studying the effects of subtle changes in the material's microscopic properties. The trouser test works by holding one end of a rectangle-shaped material and pulling the other end apart. This creates a tear down the middle, making the material look like a pair of trousers. We find that small changes in the microscale lead to large notable changes in the resulting macroscopic tearing process. Hence, exemplifying the true utility of our model. With the correct data to construct accurate periodic cells and calculate effective macroscopic coefficients. We, as well as other researchers, will be able to analyse aortic dissections

in-depth. Potentially creating endless clinical applications.

1.3.2 Upscaling of the aortic microstructure

Our motivation is a multiscale model of aortic dissections. In this thesis we create a multiscale model of material failure, based on the assumptions of asymptotic homogenisation. Here we will briefly discuss how the biophysics and architecture of the aorta are suitable for such assumptions.

Firstly and most importantly, asymptotic homogenisation requires a large-scale disparity between a macroscale and a microscale. With this approach we can model each of the three main layers of the aortic wall independently and keep a level of arbitrariness in such a model. Consider Figure 1.9 as an illustration of a cylindrical structure with a periodic microstructure. This is an excellent and simple approximation of the aorta from a human scale. Doing so, we can define a representative length scale of the macroscale to be L, which could be the wall thickness, circumference or the longitudinal length of the aorta itself. We may also define the representative length scale of the aortic microstructure, defining the size of a representative microscopic unit called δ, this could be the size of a smooth muscle cell, as illustrated in Figure 1.9. Estimating L to be measured in millimetres [60] and δ to be measured in microns [13, 22] allows us to construct the non-dimensional ratio

$$\epsilon = \frac{\delta}{L} \ll 1.$$

Hence satisfying the requirement for a disparity between the microscale and macroscale.

The next necessary requirement for using asymptotic homogenisation, in greater than one dimension, is a locally periodic microstructure. As discussed in §1.1.1 the aorta is a multilayered structure, with three layers. Each of these layers has a unique local periodic structure. This local periodic structure was first qualified by Glagov et al. [22, 137, 138] and has been further reported in numerous medical textbooks [60, 111]. These structures can be approximated as local elastic composite domains, as illustrated in Figure 1.10.

The first, and innermost, layer of an artery is the tunica intima. This layer is a sheet of flattened endothelial cells resting on a thin layer of connective tissue [13]. Mathematically we can easily approximate this locally as periodic adjacent elastic constituents, as seen in Figure 1.10. Thus satisfying the requirements of asymptotic homogenisation for a multiscale model. However, mechanically the the tunica intima plays a very small role in the aorta [65]. It could therefore be simply ignored. The most important role this layer plays in our multiscale model is that of interacting with the fluid (blood) in the true lumen, providing a fluid structure interaction as a boundary condition. Another mechanical role the endothelial cells play is signalling the muscle cells in the arterial wall to contract and regulate blood pressure [60], when blood pressure is irregularly high of course. As we will

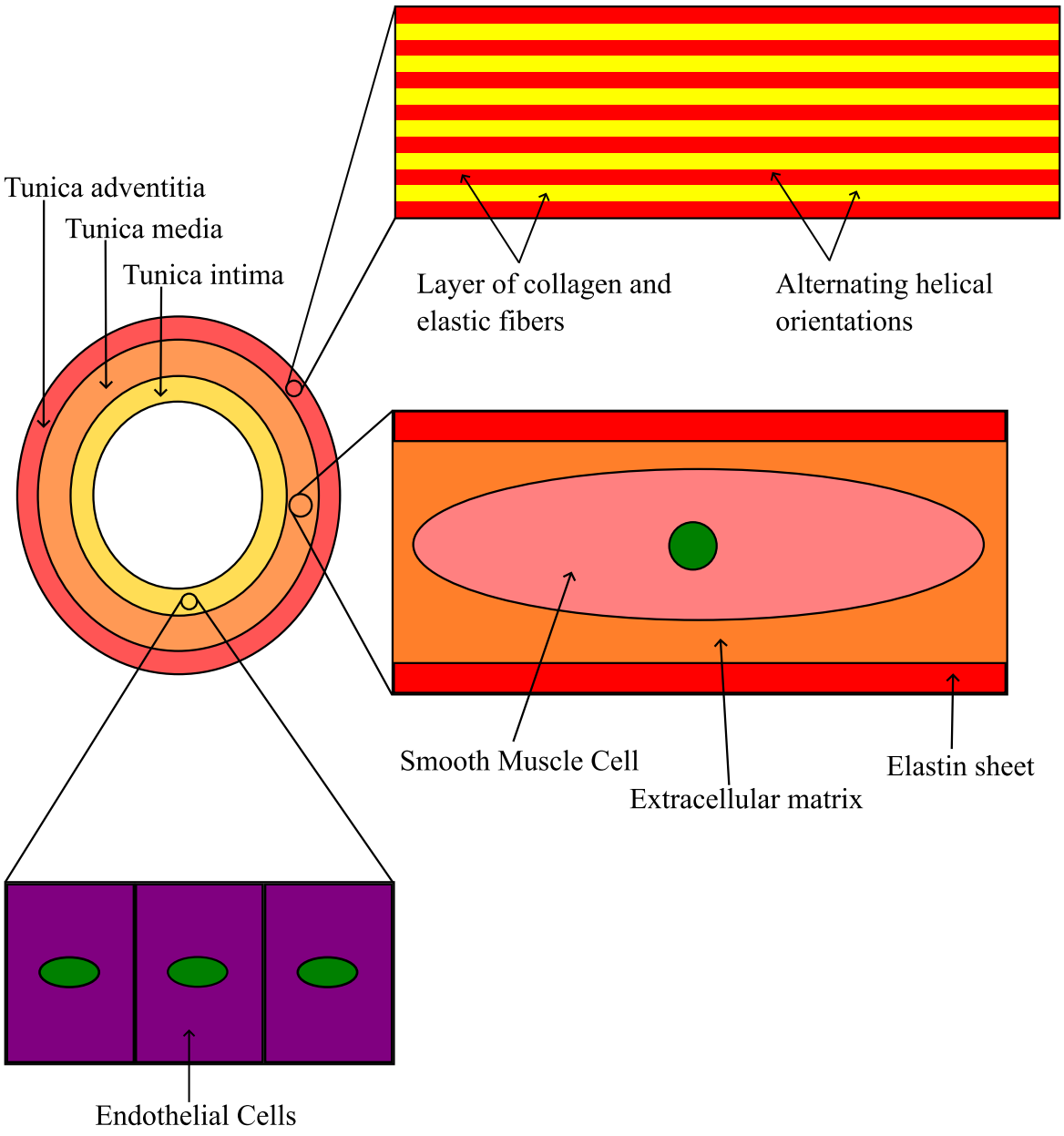


Figure 1.10: This schematic represents the aortic microstructure. Here we can see how each of the arteries three layers can be approximated by a simple periodic elastic composite structure. Firstly, the tunica intima is approximated as a local composite constructed of adjacent endothelial cells. Secondly, the tunica media is made up of circumferential layers of spindle shaped smooth muscle cells. Each smooth muscle cell is surrounded by an extracellular matrix made up of collagen fibers and other fibrous tissue. Each circumferential layer is then bounded by sheets of elastin. Thirdly, the tunica adventitia is built up by layers of collagen and elastic fibers. That alternate in helical woven orientations with each layer.

not account for this in our model then we can further argue to ignore this layer in any potential multiscale model of aortic dissections.

The middle layer of an artery is called the tunica media. It has a periodic structure in both the circumferential and longitudinal direction, made up of numerous laminated

layers [22, 137, 138]. This periodic structure is encapsulated by layers of smooth muscle cells that lie between elastic laminae interspliced with elastic collagen fibrils and elastic fibers. This structure has a slight helical pitch [13], making each layer a fibrous helix that provides extra strength to the structure of the arterial wall. The biomechanics of this layer are quite simple: the layers of fibrous tissue provide extra strength and resistance against inflation due to pressure waves of blood. When signalled by the endothelial cells, the layers of smooth muscle cells in the tunica media can contract. This provides extra resistance against inflation, thus regulating blood flow [60]. Mathematically we can approximate this as a simple periodic elastic composite, as seen in Figure 1.10. Here we treat the extracellular matrix as a host matrix, with elastin sheets and smooth muscle cells as inclusions within this composite structure. This is a simple mathematical structure and a composite structure of a host matrix and inclusions is easily upscaled [103, 105].

The final, and outermost, layer of an artery is the tunica adventitia. It is made up of thick layers of collagen fibrils and elastin fibers, as well as sparse populations of fibroblasts and fibrocytes (cells that produce collagen and elastin) [13]. These fibrous layers are arranged in alternating helical pitches and serve to reinforce the wall. As the artery becomes more inflated the layers of fibers in this section unfurl straighten out and stiffen, providing the wall with more strength. This helps to prevent the wall from overstretching. It should also be noted that this layer acts as a connective sheath anchoring the artery to surrounding connective tissue and organs that anchor it in place [60]. Mathematically in some places this could be treated as a Dirichlect boundary condition, where anchored in place, or as a stress free Neumann boundary condition, where unattached. To appropriately upscale this layer we require that it can be modelled as a locally periodic cell. We illustrate how this may be done in Figure 1.10. The adventitia can be modelled mathematically as an elastic composite structure, made up of layers of elastic constituents. Each layer represents an alternating helically pitched layers of fibrous tissue. This mathematical structure of a layered composite is easily upscaled [109].

As now discussed we can see that the aorta has a large scale disparity providing a very small ϵ, as required. We can also approximate each layer of the aortic wall as locally periodic cells, as required. The final assumptions of asymptotic homogenisation are that every field can be approximated as a power series in factors of ϵ; that every field is locally bounded and regular. These are obvious assumptions that can be made since we would be dealing with a biomechanical problem which is surely locally bounded and regular. Hence we would argue that the multiscale model that we will derive in this thesis can be applied to a future multiscale model of aortic dissections.

With a complete multiscale model of aortic dissections, we could study how various changes in the aortic microstructure affect macroscopic material failure. This would allow us to investigate the role of microstructural properties in the progression of aortic

dissections. Specifically, we could explore how altering the orientations of fibrous tissue influences dissection behaviour. Additionally, we could examine the effects of stiffened or relaxed elastin and collagen on initiating aortic dissections.

A numerical solution to this model would enable the simulation of various scenarios across a broad range of parameters. Using these simulated results, we could develop a statistical emulator to generate accurate approximations of the model. This approach would facilitate rapid parametric analysis, allowing for quick exploration of different conditions. By emulating numerical solutions, we could obtain approximate results in moments, making our multiscale model practical for clinical applications.

In the following chapters, we will provide a detailed explanation of how to construct a multiscale model of material failure capable of addressing these aspects. However, the aims of this thesis are more general.

1.3.3 Thesis aims

The aim of this thesis is to explain the mechanical factors that drive aortic dissection by creating accurate multiscale models that do not take a macroscopic phenomenological approach to modelling. Such models will allow us to understand how microscopic components of the aortic wall contribute to dissections. More generally, we want a multiscale model of material failure that can be used for a wide variety of scenarios. This includes other soft tissue tearing phenomena and failure of composite materials.

To do this, we will create a model describing the tearing process on a continuum. This model will then be used to model the aortic wall at the microscale. Using appropriate upscaling techniques, we will derive a macroscopic system of partial differential equations from our microscopic system. Our macroscopic model will effectively account for the role of the aortic microstructure, allowing one to study how the microstructure affects dissections at the macroscale. Throughout this thesis, we will attempt to answer the following questions:

1. How can we model the tearing of a continuum using the damage phase field method?
2. How can we upscale a microscopic damage model to a macroscopic modelling framework?
3. How do we account for the effects of the microstructure on the macroscopic tearing process?
4. Will our model account for the architecture or geometry of the aortic wall in dissections?
5. Can we analyse with our model which physical parameters play the biggest roles in dissections?

6. How do we account for residual stress in our model?

7. Is this work applicable to more soft tissue tearing problems in the human body?

8. Is this work applicable in general for modelling multiscale material failure?

1.3.4 Thesis outline

The rest of the thesis is organised into chapters as follows:

In Chapter 2, we introduce the damage phase field method for modelling sharp tears through a material. The purpose of this chapter is to provide the reader with all the necessary background knowledge required to understand this method of material failure modelling. We begin by introducing the non-dimensional damage phase field that measures the damage at every point in a material, ranging from completely healthy to totally ruptured. Next, we explain a series of constitutive choices made to ensure several important properties of the damage phase field. These properties include the shape of the damage phase field (an approximation of a Dirac delta function) and that damage is an irreversible process. For us to perform a variational formulation on the material energy, we need to introduce the Karush-Khun-Tucker conditions. These conditions will ensure any variational formulation we perform produces the necessary conditions that ensure the irreversibility of damage evolution. With this information we derive a set of partial differential equations that describe both the mechanics of material displacement and the evolution of the damage phase field variable in an arbitrary continuum. Using this methodology, we derive a set of partial differential equations governing the underlying mechanics and damage evolution for an arbitrary linear elastic composite material. This allows one to describe the material failure experienced in human tissue, such as the aorta, at a microscopic level by approximating the tissue as a composite material.

In Chapter 3, a new mathematical model is developed for the macroscopic behaviour of a damage phase field on an arbitrary composite material. This derivation uses a formal two scale asymptotic expansion to exploit two well separated length scales: the microscale of material inclusions in a composite and a large macroscale where the material appears homogeneous to an observer. As a result, one may use asymptotic homogenisation to develop a multiscale model described by a system of macroscopic partial differential equations supplemented by a series of effective macroscopic coefficients. Information on the effects of the materials microscopic properties and geometry are encoded into the effective macroscopic coefficients. In deriving this model special considerations must be given to a system of constraints inherent in the damage phase field method, namely irreversibility conditions. We show that these constraints are not an obstacle to homogenisation in

the case of local periodicity. This new model is applicable to the modelling of an aortic dissection, as well as many other forms of soft tissue tearing.

In Chapter 4, we prove the validity of our multiscale model. This is done by showing that our macroscopic and microscopic models converge given a large enough disparity in both the microscopic and macroscopic length scales. To do this we consider a simple one-dimensional setup that allows us to intuitively create a numerical algorithm for solving our multiscale model. From there we create the appropriate finite difference schemes required for solving both the equation of motion and the damage evolution equation. With this new numerical scheme, we can simulate our one-dimensional model of both our macroscopic and microscopic models. To finally demonstrate the validity of our model we perform a parametric analysis of all the model's parameters against a variety of microscopic and macroscopic scale disparities. This in turn allows us to gather useful conclusions on the appropriate scaling of parameters for the multiscale model. The most important conclusion of this chapter is, however, the convergence of our multiscale model.

In Chapter 5, we create new numerical solutions for solving our model in higher dimensions. This is done with the finite element method, which allows us to create numerical schemes to solve both the equations of motion and damage evolution for our multiscale model. Working in higher dimensions also allows us to introduce a mesh that can dynamically delete nodes as damage approaches the maximum value of one. As a result, we can simulate the tearing of a real material. This requires a new numerical algorithm implementing a dynamic update of the mesh geometry. With this algorithm and the appropriate numerical schemes we can therefore simulate a variety of elastic models that develop tears due to strong external loads. These numerical schemes allow us to study how microscopic changes in a material's microscopic geometry and physical properties affect the material from a macroscopic perspective. We illustrate this through an example of a two-dimensional trouser test at the end of the chapter, studying the effects of subtle changes in the material's microscopic properties. In the trouser test simulation, a rectangular sample is clamped at one end and pulled apart at the other, resulting in a tear that forms a final shape similar to that of a pair of trousers. We find that small changes in the materials microscale properties lead to large notable changes in the resulting macroscopic tearing process. Thereby, illustrating the many uses of our multiscale model for the study of aortic dissections. This exemplifies the full applicability of our new multiscale model to study material failure in complex multiscale materials.

In Chapter 6, we discuss the final conclusions of this thesis. This will begin with a discussion of the most important and novel results of each chapter. We then put all of our results into a completer picture within the context of aortic dissections and general soft tissue tearing. Of course, we shall discuss the limitations of this work and how they

may be over come with more theoretical and/or numerical development. This chapter will be completed with suggestions for possible future works in the mathematical research of aortic dissections, damage models and multiscale modelling.

Chapter 2

The damage phase field method

This chapter presents the essential background knowledge required to understand the damage phase field method. We begin by introducing the damage phase field variable α and explaining the constitutive choices made to approximate a sharp tear in a continuum. Ideally, in a reference configuration, the damage phase field exists on a co-dimensional 1 set, making it discontinuous and therefore not differentiable. To address this, we approximate the damage phase field as a sharply localized diffusive variable. To achieve this, we construct a functional that describes the energy required to tear a material, ensuring that the damage phase field remains sharply diffusive. Additionally, we introduce irreversibility constraints that prevent damage from decreasing, ensuring that it can only grow over time. These choices are made to ensure that the model remains physically meaningful and accurately approximates sharp tears in a continuum. A key advantage of this modelling technique is that it remains within the framework of continuum mechanics. By employing a variational formulation, we derive a coupled system of partial differential equations governing momentum balance and damage evolution. These equations describe both the displacement of the continuum and the evolution of the damage phase field. Since partial differential equations offer significant advantages—particularly for upscaling—this approach ensures broad applicability. By generating the most general set of partial differential equations, the model can be applied to a wide range of scenarios, including crack formation in concrete, paper tearing or an aortic dissection. Mathematically, this framework also provides a foundation for modelling plasticity and materials that exhibit strain softening. At the end of this chapter, we derive a damage phase field model for a linear elastic composite. This extension further generalizes the approach, enabling the modelling of tearing in hierarchical structures with complex microstructures.

2.1 Damage phase field method

Fractures, also known as tears, are one of the main failure mechanisms of materials under stress. Modelling of this phenomena has been of keen interest to many engineers and mathematicians, with the theoretical foundations of this field being created by Griffith [51] and Irwin [74]. In both these works a crack may propagate through a material if an energy release rate surpasses a threshold. However, these models cannot predict the initiation, location, branching or geometry of such tears through a material. Since we are modelling aortic dissections we want a model that can predict all these properties.

In this section we introduce the damage phase field method which finds its roots in the work of Marigo [83] and Miehe [90]. One should note that the damage phase field is sometimes also called the gradient damage model, as they essentially refer to the same modelling method [11]. The damage phase field method allows us to take a variational approach to modelling damage in a continuum. The idea behind this method is to define a non-dimensional damage variable, which describes the degree of damage at every point $\mathbf{X}$ in the reference configuration $\Omega \subset \mathbb{R}^n$ for $n \in \mathbb{N}$. With this variable we can later derive a system of partial differential equations that describe the evolution of damage in our continuum. This is done with an equation of momentum balance and equation of damage evolution.

Before we can begin to construct a system of partial differential equations however we must introduce two essential fields. Both fields are defined in an arbitrary domain Ω, the reference configuration. Firstly we define α as the the non-dimensional damage variable:

$$\alpha(\mathbf{X}, t) := \begin{cases} \Omega \times \mathcal{T} \to [0, 1], \\ (\mathbf{X}, t) \mapsto \alpha(\mathbf{X}, t). \end{cases}$$

Here $\mathcal{T} \subset \mathbb{R}$ represent an interval of time that we are interested in. We enforce the simple definitions that $\alpha = 0$ for completely undamaged material and $\alpha = 1$ for completely damaged material. In the damage phase field model, a tear in the material is approximated by a region where the damage is concentrated and near total. For our model we will require that α grows monotonically from 0 to 1, making damage an irreversible process. This can be a gradual process slowly resulting in a degradation of material properties or it can be a swift process leading to rapid failure in a material. The second field we need to introduce is the displacement field

$$\mathbf{u}(\mathbf{X}, t) := \begin{cases} \Omega \times \mathcal{T} \to \mathbb{R}^n \\ (\mathbf{X}, t) \mapsto \mathbf{u}(\mathbf{X}, t). \end{cases}$$

The displacement field will inform us on the mechanics of our system. Note that here $\mathbf{X}$

is the coordinate vector for a reference configuration and that t is time. Here $\mathbf{X}$ has no time dependency and neither does Ω, as they refer to a reference configuration.

Next, to understand how the damage variable α approximates damage in a material we will explain the constitutive choices we must make. We have two important constitutive choices to explain: the localisation of the damage phase field and irreversibility constraints. We will begin by explaining in one dimension how we ensure that the damage phase field approximates a Dirac delta function and why we do this. After that we will extend this approximation to higher dimensions, making the model applicable to the real world. The last constitutive choice we will explain is irreversibility: why we have it and what constraints enforce it.

2.1.1 Approximation of sharp tears in one dimension

Consider a one dimensional continuum $\Omega = (-\infty, +\infty)$ with a tear defined by the sub-domain $\mathcal{B} = \{0\}$. The idealised way to model this with the damage phase field method would be to create a damage field in the form of a Dirac delta style function, as illustrated in figure 2.1a. We could define this idealised version of α as the piecewise function

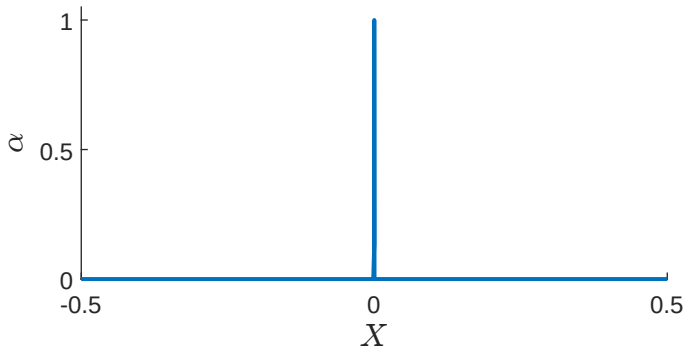
$$\alpha(X,t) := \begin{cases} 1 & X \in \mathcal{B} \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

This function gives the profile of α the desired sharpness one would expect of a tear or fracture in any material. However, this idealised damage phase field defined by (2.1) is clearly not a continuous or differentiable field. Which is not useful for our purposes of trying to create a damage model defined by a set of partial differential equations. Therefore, to ensure our damage phase field is continuous, differentiable and has a sharp profile throughout the continuum, we approximate the Dirac delta function. This is done by putting α in the form of a sharp diffusive variable, as illustrated in figure 2.1b. In one dimension we could approximate this as the solution

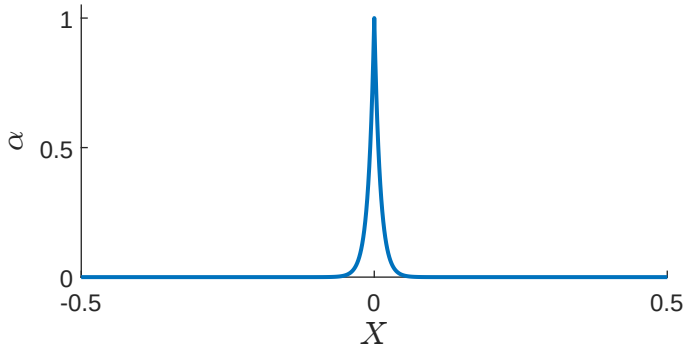
$$\alpha = e^{-|X|/\ell}, \tag{2.2}$$

with ℓ acting as a diffusive length scale which localises the width of the damage profile. Taking the limit $\ell \to 0$ we can approximate (2.1) with (2.2) as seen in figure 2.1. The accuracy of this approximation improves as ℓ gets smaller, so we would recommend the smallest possible choice of ℓ such that $0 < \ell \ll 1$.

Our goal is to develop a variational formulation that enables us to apply calculus of variations to derive the strong form equations governing the mechanics and damage evolution of the body Ω. We can use (2.2) to write down a Galerkin-type weak formulation,



(a) The ideal damage tear profile of the damage variable as defined in (2.1).



(b) The damage profile of our approximation (2.2) with $\ell = 0.01$.

Figure 2.1: For the one dimensional continuum $\Omega = [-\frac{1}{2}, \frac{1}{2}]$ with a tear present on the subdomain $\mathcal{B} = \{0\}$ we have two versions of the damage phase field approximating such a tear. In (a) we illustrate how we can model the tear with the damage phase field as a Dirac delta style function where tears are present. In (b) we illustrate an approximation of (a) by setting up the damage phase field as a sharp diffusive variable.

a functional of α, that describes a sharp diffusive form of α. With some dimensional analysis we can modify this functional to create a new energy functional. This functional describes how energy is spent to evolve α, in other words how a tear spreads in Ω. The advantage of this method is that the functional we write down based on (2.2) will ensure that α is a sharp diffusive variable that will be continuous and differentiable, unlike (2.1).

To construct such a functional we pair (2.2) with Dirichlet boundary conditions. However, (2.2) is not differentiable at the point $X = 0$, as can be seen in Figure 2.1b. So we must consider how to construct a functional for both cusps in Figure 2.1b from either side of the non-differentiable point, $X = 0$. In other words we have two domains to consider $(-\infty, 0)$ and $(0, \infty)$. For clarity let us redefine (2.2) as the piecewise function

$$\alpha = \begin{cases} e^{X/\ell} \text{ when } X \in (-\infty, 0), \\ e^{-X/\ell} \text{ when } X \in (0, \infty), \\ 1 \text{ when X} = 0. \end{cases} \tag{2.3}$$

In the first domain we will pair (2.3) with the boundary conditions $\alpha(-\infty) = 0$ and

$\alpha(0) = 1$. Then we can say that α is the solution to the differential equation

$$\alpha - \ell^2 \alpha'' = 0 \quad \text{in } (-\infty, 0). \tag{2.4}$$

In the second domain we will pair (2.3) with the boundary conditions $\alpha(0) = 1$ and $\alpha(\infty) = 0$. Then we can say that α is the solution to the differential equation

$$\alpha - \ell^2 \alpha'' = 0 \quad \text{in } (0, \infty). \tag{2.5}$$

Notice that (2.4) and (2.5) are the same differential equation, over different domains with different boundary conditions. This differential equation describes the shape of a sharp function of α at both sides of the tear at $X = 0$. Using this differential equation we can construct a functional describing the energy spent in damage evolution. Hence, we may say that in one dimension the shape of the damage phase field is governed by an equation of the form

$$\alpha - \ell^2 \alpha'' = 0 \quad \Omega. \tag{2.6}$$

We have not provided any boundary conditions for (2.6), we will use this equation to construct a Galerkin-type weak formulation. That functional will be used to create a functional describing the energy spent evolving α during a tearing process. As a result this functional will be responsible for constraining the geometry of the phase field α. As this functional will be one of several terms in a variational formulation it will be paired with other energy terms that drive the evolution of α. With this in mind let us begin to construct such a functional.

The differential equation (2.6) represents an approximation of a sharp tear in Ω, described by the diffusive variable α and regularised by a local length scale ℓ. Using equation (2.6) we can construct a Galerkin-type weak formulation of this approximation such that

$$I(\alpha, \alpha') = \frac{1}{2} \int_\Omega \alpha^2 + \ell^2 \alpha'^2 \, dX.$$

The functional I allows us to describe the geometry of the damage phase field α without explicitly writing the solution of α. In fact all one needs is the functional I paired with the appropriate boundary conditions for α and we can describe a tear at any point in Ω. We want to be even more general than this so consider that the units of I are units of length. Using this functional we can define $I = \ell \Gamma_\ell$ such that

$$\Gamma_\ell = \frac{1}{2\ell} \int_\Omega \alpha^2 + \ell^2 \alpha'^2 \, dX. \tag{2.7}$$

Here Γ_ℓ is a non-dimensional functional which also represents a Galerkin-type weak formulation describing the diffusive variable α. The reason we have used ℓ to non-dimensionalise

I is because most of damage profile is defined a distance of ℓ from the point of tearing, as observed in Figure 2.1b.

The reason we have created this functional Γ_ℓ is to constrain the shape of the damage profile in Ω. We now want to understand how to use Γ_ℓ to create an energy functional D. This is done by taking the non-dimensional functional Γ_ℓ and giving it the dimensions of energy. This can be achieved by multiplying the integrand of Γ_ℓ by some energy function. Therefore, let us define $G(X)$ a Griffith type fracture energy. Here $G(X)$ is the total energy required to lift α from 0 up to 1 at each point $X \in \Omega$. For any material stretched to the point of failure, this parameter in an ideal sense would be the area under the curve of the corresponding stress-strain curve in the zone of plasticity up to the point of material failure. We can therefore say the energy required or spent damaging our continuum, in one dimension, is

$$D[\alpha, \alpha'] = \frac{1}{2\ell} \int_\Omega G(X) \left(\alpha^2 + \ell^2 \alpha'^2\right) \, dX. \tag{2.8}$$

This term can be implemented into a variational formulation of an energy balance in Ω. Therefore allowing one to derive a set of partial differential equations describing the mechanics and damage evolution whilst also constraining α to be a sharp diffusive variable.

We have explained this mathematical formulation in one dimension to make the next steps simpler to follow. Next we will consider the same formulation in higher dimensions, as the material failure we are interested happens in our three dimensional world.

2.1.2 Approximation of sharp tears in higher dimensions

To generalise the damage phase field method we need to consider how we would approximate a sharp damage profile in higher dimensions. This is done in a near identical fashion to that of our previous explanation, in one dimension. Let us consider the continuum $\Omega \subset \mathbb{R}^n$ as our reference configuration, for $n \in \mathbb{N}$. Then a tear in this continuum would be defined by some co-dimension 1 set $\mathcal{B} \subset \mathbb{R}^{n-1}$ such that $\mathcal{B} \subset \Omega$. To understand why the tear domain $\mathcal{B}$ is a dimension lower than the domain Ω, consider tearing apart a block. In our world the block is a three dimensional object with a number of surfaces, at least one. When we tear it into two separate blocks the tear is defined along a newly created surface. A surface is a dimension lower than a three dimensional block. The same logic can be applied to the one dimensional case considered in the previous subsection. Where a tear is defined at a singular dimensionless point. A dimension lower than one dimension. Hence, ideally, the damage phase field is defined on a lower dimension set, $\mathcal{B}$.

Generally then, a tear in Ω approximated by the damage phase field would ideally be defined by

$$\alpha(\mathbf{X}, t) := \begin{cases} 1 & \mathbf{X} \in \mathcal{B} \\ 0 & \text{otherwise.} \end{cases} \tag{2.9}$$

Here α is again idealised as a Dirac delta style function as illustrated in 2D in figure 2.2. This version of α has the quality of perfectly approximating a sharp tear in Ω but is clearly not continuous or differentiable.

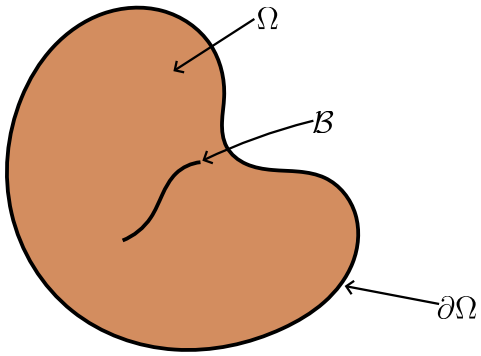


Figure 2.2: We have illustrated Ω as a two dimensional continuum with a tear present. This tear is the path $\mathcal{B}$ found in the centre of Ω. One should note that generally the tear surface are in a material is defined at a dimension lower, in other words $\text{Dim}(\Omega) = \text{Dim}(\mathcal{B}) + 1$.

Therefore, just like in the one dimension formulation, we seek to constrain the geometry of the phase field α to an approximation of a Dirac delta function. We desire that α is again a diffusive variable regularised by some local length scale ℓ. To do this we define the differential equation

$$\alpha - \ell^2 \nabla^2 \alpha = 0 \quad \text{in } \Omega, \tag{2.10}$$

with the Dirichlect boundary conditions $\alpha(\mathbf{X}) = 1$ for all $\mathbf{X} \in \mathcal{B}$ and $\alpha(\mathbf{X}) = 0$ for all $\mathbf{X} \in \partial\Omega$. The solution to such a differential equation is a sharp diffusive variable which rapidly evolves over the local length scale ℓ, as desired. We recast (2.10) as a Galerkin-weak type formulation such that

$$I(\alpha, \nabla\alpha) = \frac{1}{2} \int_\Omega \alpha^2 + \ell^2 \nabla\alpha \cdot \nabla\alpha \, dV.$$

The idea here being to incorporate the localisation of α, due to ℓ, into a variational formulation. This functional has the dimensions of volume for the continuum Ω. We want a functional that describes the localisation of damage and to do this we define $\Gamma_\ell = \ell I$, such that

$$\Gamma_\ell = \frac{1}{2\ell} \int_\Omega \alpha^2 + \ell^2 \nabla\alpha \cdot \nabla\alpha \, dV. \tag{2.11}$$

The idea here being that this approximation of the damage localisation improves as $\ell \to 0$. Again, we divided by ℓ so that our functional (2.11) describes the magnitude of the damage profile around $\alpha = 1$. In other words we setup Γ_ℓ such that

$$\Gamma_\ell = |\mathcal{B}| \quad \text{as } \ell \to 0.$$

Therefore our damage profile approximated by α has the same dimensions as $\mathcal{B}$ which we generally describe as co-dimensional 1 to Ω. By defining the magnitude of the damage profile Γ_ℓ we can also redefine G to be the tearing energy per unit of $\mathrm{Dim}(\mathcal{B})$. Here $G(\mathbf{X})$ is the amount of energy required to create a new tear in Ω. In the context of the damage phase field method this energy is the amount of energy required to transform α at each point $\mathbf{X} \in \Omega$ from 0 to 1. We can multiply the integrand of Γ_ℓ by $G(\mathbf{X})$ to define an energy functional

$$D[\alpha, \nabla\alpha] = \frac{1}{2\ell} \int_\Omega G(\mathbf{X}) \left(\alpha^2 + \ell^2 \nabla\alpha \cdot \nabla\alpha\right) \, dV. \tag{2.12}$$

First note that (2.12) is simply a generalisation of (2.8) in higher dimensions. Using this energy functional in a variational formulation we can derive a set of partial differential equations governing both the mechanics and damage evolution of Ω.

Our wish is to be as general as possible and it is possible to define a more general form of Γ_ℓ. To do this let us define

$$\Gamma_\ell = \int_\Omega \gamma(\alpha, \nabla\alpha) \, dV, \tag{2.13}$$

where we define the integrand $\gamma(\alpha, \nabla\alpha)$ as the damage surface area per unit volume. It is possible that one could define multiple different forms of γ to constrain many different forms of the damage profile in Ω. For simple sharp tears in Ω we would use

$$\gamma(\alpha, \nabla\alpha) = \frac{1}{2\ell} \left(\alpha^2 + \ell^2 \nabla\alpha \cdot \nabla\alpha\right), \tag{2.14}$$

mimicking the integrand of (2.11).

We can actually imbue more physical properties to the damage phase field α by introducing a structure tensor. The reason we do this is because some materials are naturally anisotropic and have natural paths a tear would follow. For example consider grain patterns in wood or other laminated composite materials. It is much easier to tear these composites along the laminated layers than it is tear against those layers. So to incorporate these physical properties in our model let us rewrite (2.14) as

$$\gamma(\alpha, \nabla\alpha; \mathbb{L}) = \frac{1}{2\ell} \left(\alpha^2 + \nabla\alpha \cdot \mathbb{L}\nabla\alpha\right). \tag{2.15}$$

This is the form of γ that will be used throughout the rest of this thesis. Here we have introduced the second order tensor anisotropic structure tensor $\mathbb{L}$. This tensor aligns the damage profile with local fibre orientations that may exist in our continuum Ω. We can write this as

$$\mathbb{L} = \ell^2 \left(\mathbb{I} + \omega \mathbf{f} \otimes \mathbf{f}\right),$$

where $\mathbb{I}$ is the second order identity tensor. The anisotropy parameter ω regulates a transition from an isotropic material, at $\omega = 0$, to an anisotropic material, for $\omega > 0$. Anisotropy in our continuum Ω is described by the unit vector $\mathbf{f}$. For example in a material with fibers this tensor $\mathbb{L}$ aligns the damage profile and also controls the width of the profile we will later refer to it as a diffusivity tensor when discussing the damage evolution equation.

Remark. *Some interesting conclusion about the function γ were reported in [53]. The function γ as defined in* (2.14)*, is an isotropic function satisfying the condition $\gamma(\alpha, \nabla\alpha) = \gamma(\alpha, \mathbb{Q}\nabla\alpha)$, $\forall \mathbb{Q} \in \mathcal{O}(3)$. The tensor $\mathbb{Q}$ denotes a rotation and/or reflection in the orthogonal group $\mathcal{O}(3)$. Similarly the function γ as defined in* (2.15)*, satisfies $\gamma(\alpha, \nabla\alpha; \mathbb{L}) = \gamma(\alpha, \mathbb{Q}\nabla\alpha; \mathbb{L})$, $\forall \mathbb{Q} \in \mathcal{G} \subset \mathcal{O}(3)$, where $\mathcal{G}$ designates a symmetry group as a subset of $\mathcal{O}(3)$.*

2.1.3 Irreversibility conditions

When we tear an object we naturally assume that to be an irreversible process and that the damage is permanent. Of course some materials can heal themselves, biological tissues for example. However, in this thesis we consider materials that cannot heal. That is to say damage is irreversible. In other words we are assuming damage evolution to be a completely dissipative process.

As a consequence the natural postulation of the damage phase field method is the irreversibility of the damage growth. Another way of expressing this is to say that the growth of the damage surface area is an irreversible process and we can express this mathematically by saying that

$$\frac{d}{dt}\Gamma_\ell \geq 0 \quad \text{and} \quad \dot{\alpha} \geq 0, \quad \forall \mathbf{X} \in \Omega \quad \text{and} \quad \forall t \in [0, \infty).$$

Here the dot above α indicates a Lagrangian time derivative in the reference domain Ω, this notation will be used throughout the thesis. We set these inequalities as a constitutive choice and from them we can find more general constraints. By recalling (2.11) we can derive that, generally,

$$\begin{aligned}
\frac{d}{dt}\Gamma_\ell &= \int_\Omega \frac{d\gamma}{dt}\, dV \\
&= \int_\Omega \frac{\partial\gamma}{\partial\alpha}\dot{\alpha} + \frac{\partial\gamma}{\partial(\nabla\alpha)} \cdot \nabla\dot{\alpha}\, dV \\
&= \int_\Omega \left(\frac{\partial\gamma}{\partial\alpha} - \nabla \cdot \left(\frac{\partial\gamma}{\partial(\nabla\alpha)} \right) \right) \dot{\alpha}\, dV + \int_{\partial\Omega} \left(\frac{\partial\gamma}{\partial(\nabla\alpha)}\dot{\alpha} \right) \cdot \mathbf{n}\, dA.
\end{aligned}$$

Here we have defined $\mathbf{n}$ as an outward unit normal vector on $\partial\Omega$. Therefore as we are assuming that damage evolution is irreversible we must have that, for all possible evolu-

tions,

$$\int_\Omega \left(\frac{\partial \gamma}{\partial \alpha} - \nabla \cdot \left(\frac{\partial \gamma}{\partial (\nabla \alpha)} \right) \right) \dot{\alpha} \, dV + \int_{\partial \Omega} \left(\frac{\partial \gamma}{\partial (\nabla \alpha)} \dot{\alpha} \right) \cdot \mathbf{n} \, dA \geq 0.$$

Given that Ω is an arbitrary domain we can conclude with the localisation theorem that in order for damage evolution to be an irreversible process we require that

$$\int_\Omega \left(\frac{\partial \gamma}{\partial \alpha} - \nabla \cdot \left(\frac{\partial \gamma}{\partial (\nabla \alpha)} \right) \right) \dot{\alpha} \, dV \geq 0 \quad \text{and} \quad \int_{\partial \Omega} \left(\frac{\partial \gamma}{\partial (\nabla \alpha)} \dot{\alpha} \right) \cdot \mathbf{n} \, dA \geq 0.$$

As we are assuming that the constraint $\dot{\alpha} \geq 0$ is true, then for each integral to always be positive for every domain Ω, the localisation theorem tells us that, we require that

$$\left(\frac{\partial \gamma}{\partial \alpha} - \nabla \cdot \left(\frac{\partial \gamma}{\partial (\nabla \alpha)} \right) \right) \geq 0 \quad \text{in } \Omega, \tag{2.16a}$$

$$\frac{\partial \gamma}{\partial (\nabla \alpha)} \cdot \mathbf{n} \geq 0 \quad \text{on } \partial \Omega. \tag{2.16b}$$

These inequalities act as constraints that must be satisfied in order for α to grow monotonically in the positive direction. The former constraint (2.16a) acts as a functional derivative implying that the damage surface area must grow in the positive direction and the volume under such a field is positive, therefore enforcing a sharp tear geometry. The latter constraint (2.16b) acts on the boundary of Ω ensuring that any Neumann type boundary conditions are positive definite on $\partial \Omega$. In other words any flux must be positive definite to ensure irreversibility. We can see this by recalling the definition of (2.15), such that

$$\mathbb{L} \nabla \alpha \cdot \mathbf{n} \geq 0 \quad \text{on } \partial \Omega. \tag{2.17}$$

As flux is always positive or zero then the damage phase field cannot leave the material Ω, in other words a tear can not travel from one object to another.

Remark. *In this thesis we are not discussing materials that can heal. The reason we do this is because the governing set of partial differential equations we derive couples this displacement* $\mathbf{u}$ *and the damage* α*. Damage growth is therefore directly linked to the material deformations. If we allow the system to be reversible then we would allow material healing to be directly linked to deformation. For example we would be saying that when a material is stretched it begins to break and tear but as it is compressed it then suddenly begins to heal, which would not make any physical sense. If one wanted to factor healing into the system then it would have to be included into the system separate from the deformation and it would also have to stop healing the material once* α *has reached zero again. This is an interesting research question we are not tackling in this thesis. In our opinion most material failure happens on a time scale much smaller than healing, anyway. So it would not make to much sense to model them together in a general sense.*

2.2 The damage mechanics

In the previous section we discussed the motivation behind the damage phase field method and how it can be used to model damage throughout a continuum. Here we are concerned with deriving a set of governing equations and specifying constitutive choices. In any damage model we must take care to respect the irreversibility of damage growth and the degradation of material properties caused by damage. To derive a set of governing equations we consider the action Π of a balance of energy present in any continuum. Considering a variational formulation of Π will allow us to derive a constrained system of partial differential equations which ensure that the irreversibility constraints are met.

As a point of departure let us begin by considering a continuum body at time $t_0 \in \mathcal{T}$, which we refer to as the reference configuration, designated as $\Omega \subset \mathbb{R}^n$, with the spatial point $\mathbf{X} \in \Omega$. Similarly, the deformed body at current time $t \in \mathcal{T}$, which we refer to as the current configuration, is denoted by $\Omega_t \subset \mathbb{R}^n$ with the spatial point $\mathbf{x} \in \Omega_t$. The deformation from the reference configuration to the current configuration is mapped by the displacement $\mathbf{u}(\mathbf{X}, t)$, as seen in figure 2.3. For clarity please note that the overdot notation on the term $\dot{\mathbf{u}}(\mathbf{X}, t)$ is a Lagrangian time derivative in the reference domain Ω, as $\mathbf{X}$ has no time dependency.

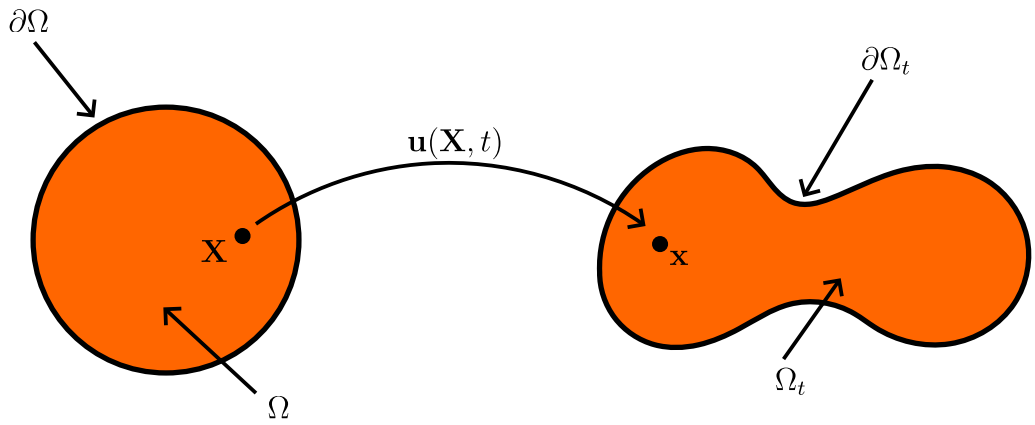


Figure 2.3: The deformation of a solid material. Here we can see the displacement $\mathbf{u}$ from the reference configuration $\Omega \subset \mathbb{R}^n$ at time t_0 to the current configuration $\Omega_t \subset \mathbb{R}^n$ at time t. Spatial points are denoted by the vector $\mathbf{X}$ in the reference configuration and the vector $\mathbf{x}$ in the current configuration. The boundaries are denoted by $\partial\Omega$ in the reference configuration and by $\partial\Omega_t$ in the current configuration.

In our modelling we will work in the reference configuration and use a Lagrangian description of the mechanics to understand material deformation and damage growth. We will therefore need to create a description of energy balance in the reference configuration. To do this we will introduce the action Π which has the dimensions of Joule seconds. If we were dealing with a regular continuum with no constraints on the damage then we could

say that

$$\Pi[\mathbf{u}, \alpha] = \int_{\mathcal{T}} K[\mathbf{u}, \dot{\mathbf{u}}, \alpha] + B[\mathbf{u}] - P[\mathbf{u}, \alpha]\, dt,$$

where K is the kinetic energy, B is the external energy from body and surface forces and P is the elastic potential energy from deformation. The external energy due to external loading, B, is positive and contributes to the kinetic energy in this Lagrangian. Here K, B and P would all be functionals calculating the total energy over the reference configuration Ω. To include the damage mechanics into the formulation we would need to consider firstly the energy spent creating a tear. We discussed this in detail in the last section and to include this into our energy balance we would write that

$$\Pi[\mathbf{u}, \alpha] = \int_{\mathcal{T}} K[\mathbf{u}, \dot{\mathbf{u}}, \alpha] + B[\mathbf{u}] - P[\mathbf{u}, \alpha] - D[\gamma, \nabla\gamma; \mathbb{L}]\, dt.$$

Where D is the energy spent creating new damage surface area or the tearing energy functional. This functional will be informed by the previous discussion, in §2.1, concerning the damage profile. We could use this form of Π to derive a strong set of governing equations for $\mathbf{u}$ and α but we first must introduce an energetic damage threshold.

Without a threshold for damage growth any slight external loading on Ω would cause damage growth. That would be senseless, all materials can withstand some level of deformation before either plasticising or failing. So we introduce the energy functional $T(\alpha, \mathbf{X})$. Here T will act as the damage threshold such that we can only initiate damage growth at a spatial point $\mathbf{X}$ if the elastic potential at that point is greater than the damage threshold. If this is not satisfied then there will no damage growth at the point $\mathbf{X}$. We will see that the reason both P and T depend on α is because as damage grows the material properties that they define degrade. We need to think carefully about the sign of T when we introduced it to the variational formulation Π.

As mentioned the elastic potential energy P is what will drive the tearing in Ω. However, we must introduce a threshold value to the required energy necessary to begin tearing Ω. Otherwise we will derive a model where any elastic potential energy will begin to drive damage evolution. This would not make any physical sense, we know that most materials have a range of strains they can withstand before the onset of plasticity. Hence the new energy driving the tearing process is $P - T$. Here T is a functional representing the energy density threshold for the onset of material degradation. Therefore in the above variational formulation, Π, we may simply replace the term potential term P with the term $P - T$.

By introducing T as the damage threshold of Ω, we can calculate the balance of energy. This balance of energy describes both the damage evolution and deformation at every point in Ω. This allows us to model the relationship between the mechanics of a continuum Ω

and the growth of damage. We can therefore write the action as

$$\Pi[\mathbf{u}, \alpha] = \int_{\mathcal{T}} K[\mathbf{u}, \dot{\mathbf{u}}] + B[\mathbf{u}] + T[\alpha, \mathbf{X}] - P[\mathbf{u}, \alpha] - D[\gamma, \nabla\gamma; \mathbb{L}]\, dt. \tag{2.18}$$

Later we will consider a variation of the action, $\delta\Pi$, this will allow us to derive a system of partial differential equations describing $\mathbf{u}$ and α. Special consideration will be given to the irreversibility constraints and result in us introducing the idea of Karush-Khun-Tucker conditions. To do this we must first begin by defining specific forms of all the energy functionals. Thereby, making a set of constitutive choices on the mechanics and damage evolution of the system Ω. We will also need to consider how to introduce material degradation into the model before deriving a system of governing equations.

2.2.1 The energy functionals

Here we will specify the specific forms of each energy functional present in the action (2.18). Let us first begin by discussing the kinetic energy of Ω, before we proceed it is important that we comment that damage does not effect the density at any point $\mathbf{X}$ in Ω. Defining $\rho(\mathbf{X})$ as the material density of Ω then the kinetic energy of the body is

$$K[\dot{\mathbf{u}}] = \int_{\Omega} \frac{1}{2} \rho \dot{\mathbf{u}} \cdot \dot{\mathbf{u}}\, dV. \tag{2.19}$$

Next we account for external loading on our body by considering the energy contributions from external body and surfaces forces on our system. Consider the external loading as defined by the external energy functional

$$B[\mathbf{u}] = \int_{\Omega} \rho \mathbf{b} \cdot \mathbf{u}\, dV + \int_{\partial\Omega} \mathbf{t} \cdot \mathbf{u}\, dA, \tag{2.20}$$

where $\mathbf{b}(\mathbf{X}, t)$ is defined as the body force per unit mass and $\mathbf{t}(\mathbf{X}, t)$ is defined as the traction force per unit area. Body forces and traction would drive the displacement of material points in Ω. As a result this drives the growth of elastic potential and thus damage in Ω which inevitably leads to tears. The traction will prescribe a Neumann type boundary condition by balancing itself against the surface stress of Ω. By keeping the body force and traction arbitrary we allow our self the opportunity to enforce a wide variety of external loading.

Throughout this thesis we will be focusing on a linear theory of elasticity for solids. Here though we can preserve some generality by defining a strain energy density function in terms of $\mathbf{F}$. Where $\mathbf{F}$ is the deformation gradient, defined by

$$\mathbf{F} = \nabla \mathbf{x}, \tag{2.21}$$

with $\mathbf{x}$ being a spatial point in the current configuration. The deformation gradient is a second order tensor that represent the gradient of a mapping function, from the reference to the current configuration. Note that ∇ is the gradient with respect to the reference configuration, in terms of $\mathbf{X}$. We may redefine as $\mathbf{x} = \mathbf{X} + \mathbf{u}$. Therefore, we can write the deformation gradient as

$$\mathbf{F} = \mathbf{I} + \nabla \mathbf{u}. \tag{2.22}$$

This will be useful later when performing calculating our variational formulation.

The potential energy is probably the most important energy functional we need to consider. To make constitutive choices about the mechanics of a solid we choose a strain energy density function $\Phi(\mathbf{F}, \alpha)$. It governs the stored elastic energy of Ω which in turns drives the growth of damage in Ω. As a start we may write the potential functional as

$$P[\mathbf{u}, \alpha] = \int_{\Omega} \Phi(\mathbf{F}, \alpha)\, dV. \tag{2.23}$$

Here the potential functional depends on both the displacement and the damage. This is because as the damage grows at every spatial point $\mathbf{X}$ the elastic properties of Ω at those points will degrade. We therefore factor this degradation into the chosen form of Φ. Doing so will couple the damage evolution and mechanics of Ω.

As mentioned earlier we need to introduce a damage threshold functional T to our energy balance. The threshold functional T is defined as

$$T[\alpha, \mathbf{X}] = \int_{\Omega} \Psi(\mathbf{X}, \alpha)\, dV, \tag{2.24}$$

where $\Psi(\mathbf{X}, \alpha)$ is a volumetric energy density that will ensure damage may only occur whenever the strain energy density is large enough. In essence for damage evolution we are adding the constraint that for initial damage at a point we require that

$$\Phi - \Psi \geq 0.$$

Another consideration to be made about the damage threshold is its dependence on the local damage. Generally if a material becomes damaged then it becomes easier to create further damage. Therefore as damage increases the threshold should decrease. We will discuss this further in §2.2.2.

The energy functional, D, is the tearing energy functional which is a dissipative process in Ω and describes how the internal energy of Ω contributes towards the creation of damage. To calculate this tearing energy (that leads to new tear surface area) we have to consider what contributes to the dissipation of energy via damage growth. Firstly lets think about the energy actively used to create new tears. Recall that an approximation to the damage

profile can be written as described in (2.11). By defining $G(\mathbf{X})$, the energy per units of the co-dimension 1 of Ω to create a tear at $\mathbf{X}$, we can directly write this functional as

$$D[\alpha, \nabla\alpha] = \int_\Omega G(\mathbf{X})\gamma(\alpha, \nabla\alpha; \mathbb{L})\, dV.$$

This is the energy used to grow the damage profile and create a new tear surface area. We may write a more specific form of this functional by choosing a specific form of γ. So to be as general as possible we use the anisotropic definition (2.15). This allows us to write our form of the dissipation functional as

$$D[\alpha, \nabla\alpha] = \frac{1}{2\ell}\int_\Omega G(\mathbf{X})\left(\alpha^2 + \nabla\alpha \cdot \mathbb{L}\nabla\alpha\right)\, dV \tag{2.25}$$

Note that the choice of γ is a particular constitutive choice, we are sticking with our definition because we are interested in sharp tears, as discussed in §2.1. There are of course many possibilities for γ which may be constructed to ensure that the damage phase field α has other geometric properties.

2.2.2 The degradation function

Our damage phase field obviously interacts with the continuum, as a material becomes more damaged we would expect the material properties to change. We can account for this by introducing the multiplicative degradation function $g(\alpha)$. This function will allow us to take account for the effects damage is playing on the material properties. In the case of the elastic potential $\Phi(\nabla\mathbf{u}, \alpha)$ we can assume a multiplicative decomposition of the potential such that

$$\Phi(\nabla\mathbf{u}, \alpha) = g(\alpha)\phi(\mathbf{F}).$$

Here ϕ is the elastic potential function of the material if we assume it can never become damaged. The degradation function however has to have important constitutive properties in order for this model to make sense. We require that when there is no damage the material properties are unaffected and when the damage is full ($\alpha = 1$) the material properties totally fail. In practicality the material point will lose all elastic resistance when $\alpha = 1$.

We also require that degradation is monotonically decreasing, therefore we write the constitutive properties as

$$g(0) = 1, \quad g(1) = 0, \quad g'(\alpha) \leq 0 \quad \text{and} \quad g'(1) = 0. \tag{2.26}$$

The final condition we defined is a mathematical condition, that forces the point $\alpha = 1$ to behave as a stable point and stop α diverging in our numerical solutions. These properties

ensure that $g(\alpha)$ captures the change in material properties as the material degrades with increasing α. An example of a set of functions that satisfy the conditions (2.26) would be the family of functions

$$g(\alpha) = (1-\alpha)^p \quad \text{for } p \geq 2.$$

The simplest choice of these functions would be for the exponent p to be set equal to two. Though it is important to note that as the exponent p increases this heavily effects the stress softening of the material [139]. With these ideas in mind we can now write our potential energy functional which captures the role of damage as

$$P[\mathbf{u}, \alpha] = \int_\Omega g(\alpha)\phi(\mathbf{F})\, dV. \tag{2.27}$$

In an analogous fashion we introduce the damage degradation function $w(\alpha)$ to model the degradation of the damage threshold $\Psi(\mathbf{X}, \alpha)$. With this in mind we introduce a multiplicative decomposition of Ψ, such that

$$\Psi(\mathbf{X}, \alpha) = w(\alpha)\psi(\mathbf{X}).$$

Here $w(\alpha)$ is the degradation function for the damage threshold. It shares all the constitutive properties of $g(\alpha)$ as described in (2.26). Analogously then we can write that

$$w(0) = 1, \quad w(1) = 0, \quad w'(\alpha) \leq 0 \quad \text{and} \quad w'(1) = 0.$$

Therefore we can write a final form of the threshold energy functional as

$$T[\alpha, \mathbf{X}] = \int_\Omega w(\alpha)\psi(\mathbf{X})\, dV. \tag{2.28}$$

Here ψ is the threshold energy function of the material if we assume it can never be damaged. As α grows $w(\alpha)$ will decrease towards zero and thus the damage threshold increases as damage decreases.

2.2.3 Karush-Khun-Tucker conditions

Before we calculate our variational formulation it is important we consider how the constraints on the damage phase field will effect that formulation. Let us then consider the action $\Pi(\mathbf{u}, \alpha)$, a functional that describes how the balance of kinetic and potential energy of a physical system changes with the trajectories of displacement and damage. The variation of this action, $\delta\Pi$, is not only a functional of $\mathbf{u}$ and α but also of their admissible variations $\mathbf{v}$ and μ. Usually these admissible variations have no constraints on them, in our case special consideration must be given to what is an acceptable variation. As we are

assuming that damage is irreversible then the only admissible variations of α are $\mu \geq 0$.

The standard approach in a variational formulation is usually to find a minimal stationary point of the paths represented by Π. This is done by considering

$$\delta\Pi = 0.$$

It should be noted that the variation $\delta\Pi$ is linear in terms of $\mathbf{v}$ and μ such that

$$\delta\Pi = \int_{\mathcal{T}} \int_{\Omega} \mathbf{A}(\mathbf{u}, \alpha) \cdot \mathbf{v} + B(\mathbf{u}, \alpha)\mu \, dV dt, \tag{2.29}$$

for some arbitrary vector and scalar functions $\mathbf{A}$ and B, respectively. In the standard approach we use the fundamental lemma of the calculus of variations, and that $\mathbf{v}$ and μ are entirely arbitrary and admissible, to deduce a system of equations. One would conclude with a set of governing equations defined by $\mathbf{A} = \mathbf{0}$ and $B = 0$. In our case $\mathbf{v} \in \mathbb{R}^n$ and $\mu \in \mathbb{R}_{\geq 0}$. Hence the variations of damage, μ, are not totally unrestricted and we can therefore not apply the fundamental lemma of calculus of variations to derive a model using the standard Euler-Lagrange approach.

Let us thus consider that the variation $\delta\Pi(\mathbf{u}, \alpha, \dot{\mathbf{u}}, \dot{\alpha})$ has the property of being a minimum amongst all possible variations. Then any variation $\delta\Pi(\mathbf{u}, \alpha, \mathbf{v}, \mu)$ where $\mathbf{v} \in \mathbb{R}^n$ and $\mu > 0$ are virtual velocities can be assumed to satisfy

$$\begin{aligned} &\delta\Pi(\mathbf{u}, \alpha, \dot{\mathbf{u}}, \dot{\alpha}) \leq \delta\Pi(\mathbf{u}, \alpha, \mathbf{v}, \mu) \\ &\forall \mathbf{v}, \forall \mu \geq 0. \end{aligned} \tag{2.30}$$

This is a mathematical expression of the principle of least action.

First we will show that in the case where one considers no variations in the damage phase field, $\mu = 0$, the variational problem reduces to $\delta\Pi(\mathbf{u}, \alpha, \mathbf{v}, 0) = 0$. Which is just the regular mechanical variational formulation which should produce an equation of linear momentum balance. Let the virtual velocity field be $\mathbf{v} = \dot{\mathbf{u}} + \bar{\mathbf{v}}$ and $\mu = \dot{\alpha}$. Since μ is an arbitrary positive field, the choice $\mu = \dot{\alpha}$ is admissible as we have constrained $\dot{\alpha} \geq 0$. So we have,

$$\delta\Pi(\mathbf{u}, \alpha, \dot{\mathbf{u}}, \dot{\alpha}) \leq \delta\Pi(\mathbf{u}, \alpha, \dot{\mathbf{u}} + \bar{\mathbf{v}}, \dot{\alpha}). \tag{2.31}$$

Now also consider the case $\mathbf{v} = \dot{\mathbf{u}} - \bar{\mathbf{v}}$ and $\mu = \dot{\alpha}$, such that

$$\delta\Pi(\mathbf{u}, \alpha, \dot{\mathbf{u}}, \dot{\alpha}) \leq \delta\Pi(\mathbf{u}, \alpha, \dot{\mathbf{u}} - \bar{\mathbf{v}}, \dot{\alpha}). \tag{2.32}$$

Thus by the linearity condition (2.29) we must have that

$$\delta\Pi(\mathbf{u}, \alpha, \bar{\mathbf{v}}, 0) \geq 0 \quad \text{and} \quad \delta\Pi(\mathbf{u}, \alpha, \bar{\mathbf{v}}, 0) \leq 0.$$

The only solution that satisfies both of these inequalities is therefore that

$$\delta\Pi(\mathbf{u}, \alpha, \bar{\mathbf{v}}, 0) = 0.$$

Thus we have shown that in the case of assuming no damage variation the variational formulation of Π reduces to the regular mechanical problem.

Next we want to derive an expression of $\delta\Pi$ that we can use for a localisation process for the damage problem. Recall that $\mathbf{v}$ is not constrained and can take any from such that $\mathbf{v} \in \mathbb{R}^n$, whereas μ is a scalar function constrained such that $\mu \geq 0$. We begin by setting $\mathbf{v} = \dot{\mathbf{u}}$ and $\mu = 0$, substitution in (2.30) yields

$$\delta\Pi(\mathbf{u}, \alpha, \dot{\mathbf{u}}, \dot{\alpha}) \leq \delta\Pi(\mathbf{u}, \alpha, \dot{\mathbf{u}}, 0). \tag{2.33}$$

We make the second choice $\mathbf{v} = \dot{\mathbf{u}}$ and $\mu = 2\dot{\alpha}$ substituting into (2.30) then yields

$$\delta\Pi(\mathbf{u}, \alpha, \dot{\mathbf{u}}, \dot{\alpha}) \leq \delta\Pi(\mathbf{u}, \alpha, \dot{\mathbf{u}}, 2\dot{\alpha}). \tag{2.34}$$

Then, again, by the linearity condition (2.29) we have that

$$\delta\Pi(\mathbf{u}, \alpha, \mathbf{0}, \dot{\alpha}) \leq 0 \quad \text{and} \quad \delta\Pi(\mathbf{u}, \alpha, \mathbf{0}, \dot{\alpha}) \geq 0.$$

Here the only solution that satisfies both solutions is that

$$\delta\Pi(\mathbf{u}, \alpha, \mathbf{0}, \dot{\alpha}) = 0. \tag{2.35}$$

This alone is not suitable for a localisation process as here $\dot{\alpha} \geq 0$ everywhere in Ω. So we consider (2.30) for the case $\mathbf{v} = \dot{\mathbf{u}}$, if one accounts for (2.29) then its clear that

$$\delta\Pi(\mathbf{u}, \alpha, \mathbf{0}, \mu) \geq \delta\Pi(\mathbf{u}, \alpha, \mathbf{0}, \dot{\alpha}) \quad \forall \mu \geq 0.$$

Combining this with the result (2.35) we have derived the following Karush-Khun-Tucker conditions

$$\begin{aligned} &\delta\Pi(\mathbf{u}, \alpha, \mathbf{0}, \dot{\alpha}) = 0 \\ &\delta\Pi(\mathbf{u}, \alpha, \mathbf{0}, \mu) \geq 0 \\ &\forall \mu \geq 0. \end{aligned} \tag{2.36}$$

With the integral form in (2.36) due to the arbitrariness of the virtual velocity μ this variational formulation is now suitable for localisation. The derivation of these conditions are not an original result and they have been discussed before in context with the damage phase field [84, 106].

2.2.4 Variational formulation

To derive a governing system of equations that dictate the behaviour of $\mathbf{u}$ and α we now consider the action as originally mentioned in (2.18). Our action describes the balance of energy as

$$\Pi = \int_{\mathcal{T}} K[\dot{\mathbf{u}}, \alpha] + B[\mathbf{u}] + T[\alpha, \mathbf{X}] - P[\mathbf{u}, \alpha] - D[\gamma, \nabla\gamma; \mathbb{L}]\, dt.$$

To define Π we substitute in (2.19), (2.20), (2.28), (2.27) and (2.25) to the respective functionals K, B, T, P and D. Therefore we can write that

$$\begin{aligned}\Pi = &\int_{\mathcal{T}}\int_{\Omega} \frac{1}{2}\rho\dot{\mathbf{u}}\cdot\dot{\mathbf{u}} + \rho\mathbf{b}\cdot\mathbf{u} + w(\alpha)\psi \\ &- g(\alpha)\phi(\nabla\mathbf{u}) - \frac{1}{2\ell}G\left(\alpha^2 + \nabla\alpha\cdot\mathbb{L}\nabla\alpha\right)\, dV\, dt \\ &+ \int_{\mathcal{T}}\int_{\partial\Omega} \mathbf{t}\cdot\mathbf{u}\, dA\, dt.\end{aligned}$$

This is a weak form statement of our system for all admissible forms of $\mathbf{u}$ and α. To create a strong form set of equations we need to first find a general form of the variation $\delta\Pi$. To do these we will use the directional derivative to define the variation. This is done by calculating

$$\delta\Pi = \left[\frac{d}{dh}\Pi(\mathbf{u} + h\mathbf{v}, \alpha + h\mu)\right]_{h=0}, \tag{2.37}$$

where $\mathbf{v} \in \mathbb{R}^n$ and $\mu \geq 0$ are admissible variations of the displacement and damage, respectively. Here h is a small positive number such that $h \ll 1$, this ensures that the variations are small. It should also be noted that both $\mathbf{v}$ and μ have the standard boundary conditions

$$\mathbf{v}(t_0) = \mathbf{v}(t_1) = \mathbf{0} \quad \text{and} \quad \mu(t_0) = \mu(t_1) = 0,$$

such that $\mathcal{T} := [t_0, t_1]$. Before we take the directional derivative it is useful for us to first write down that

$$\begin{aligned}\Pi[\mathbf{u} + h\mathbf{v}, \alpha + h\mu] = &\int_{\mathcal{T}}\int_{\Omega} \frac{1}{2}\rho\dot{\mathbf{u}}\cdot\dot{\mathbf{u}} + \rho h\dot{\mathbf{u}}\cdot\dot{\mathbf{v}} \\ &+ \rho\mathbf{b}\cdot\mathbf{u} + \rho\mathbf{b}\cdot h\mathbf{v} + w(\alpha + h\mu)\psi \\ &- g(\alpha + h\mu)\phi(\mathbf{I} + \nabla\mathbf{u} + h\nabla\mathbf{v}) - \frac{G}{2\ell}\left(\alpha + h\mu\right)^2 \\ &- \frac{G}{2\ell}\left(\nabla\alpha\cdot\mathbb{L}\nabla\alpha + 2h\nabla\mu\cdot\mathbb{L}\nabla\alpha\right) + \mathcal{O}(h^2)\, dV\, dt \\ &+ \int_{\mathcal{T}}\int_{\partial\Omega} \mathbf{t}\cdot\mathbf{u} + \mathbf{t}\cdot h\mathbf{v}\, dA\, dt.\end{aligned}$$

Given that the above holds for arbitrary domains Ω, it is therefore possible to derive a governing set of partial differential equations for any arbitrary domain. We can now

calculate the variation by taking a derivative with respect to h then setting $h = 0$. Doing so leaves us with the variation

$$\begin{aligned}\delta\Pi(\mathbf{u},\alpha,\mathbf{v},\mu) = &\int_{\mathcal{T}}\int_{\Omega}\left[-\rho\ddot{\mathbf{u}} + \rho\mathbf{b} + \nabla\cdot\left(g(\alpha)\frac{\partial\phi}{\partial\mathbf{F}}\right)\right]\cdot\mathbf{v} \\ &+\left[-g'(\alpha)\phi(\mathbf{F}) + w'(\alpha)\psi - \frac{G}{\ell}\alpha + \nabla\cdot\left(\frac{G}{\ell}\mathbb{L}\nabla\alpha\right)\right]\mu\,dV\,dt \\ &+\int_{\mathcal{T}}\int_{\partial\Omega}\left[\mathbf{t} - \frac{\partial\phi}{\partial\mathbf{F}}\mathbf{n}\right]\cdot\mathbf{v} - \left(\frac{G}{\ell}\mathbb{L}\nabla\alpha\cdot\mathbf{n}\right)\mu\,dA\,dt.\end{aligned} \tag{2.38}$$

Here we have made use of the product rule, the definition of the deformation gradient (2.22) and the divergence theorem twice to put the variation into its most useful form for our purposes. We also used integration by parts and the boundary conditions for $\mathbf{v}$ to eliminate some other terms.

Now that we have the variation of the action we want to apply the results discussed in §2.2.3 to derive a set of governing equations. To begin we consider a case in Ω where we have no damage growth. So let us imagine that the variation $\mu = 0$ everywhere in Ω. Obviously then we can write that

$$\delta\Pi(\mathbf{u},\alpha,\mathbf{v},0) = \int_{\Omega}\left[\rho\mathbf{b} - \rho\ddot{\mathbf{u}} + \nabla\cdot\left(g(\alpha)\frac{\partial\phi}{\partial\mathbf{F}}\right)\right]\cdot\mathbf{v}\,dV + \int_{\partial\Omega}\left[\mathbf{t} - g(\alpha)\frac{\partial\phi}{\partial\mathbf{F}}\mathbf{n}\right]\cdot\mathbf{v}\,dA.$$

This variation holds for every $\mathbf{v}\in\mathbb{R}^n$, so let us consider a minimum. To do this we just set $\delta\Pi = 0$ and therefore

$$0 = \int_{\Omega}\left[\rho\mathbf{b} - \rho\ddot{\mathbf{u}} + \nabla\cdot\left(g(\alpha)\frac{\partial\phi}{\partial\mathbf{F}}\right)\right]\cdot\mathbf{v}\,dV + \int_{\partial\Omega}\left[\mathbf{t} - g(\alpha)\frac{\partial\phi}{\partial\mathbf{F}]}\mathbf{n}\right]\cdot\mathbf{v}\,dA.$$

From here we can argue if this is true for all admissible choices of $\mathbf{v}$ in $\mathbb{R}^n$ then we must have, by the fundamental lemma of calculus of variations, that

$$\rho\mathbf{b} + \nabla\cdot\left(g(\alpha)\frac{\partial\phi}{\partial\mathbf{F}}\right) = \rho\ddot{\mathbf{u}} \quad \text{in } \Omega \tag{2.39}$$

$$\mathbf{t} = g(\alpha)\frac{\partial\phi}{\partial\mathbf{F}}\mathbf{n} \quad \text{on } \partial\Omega. \tag{2.40}$$

The former (2.39) is a momentum balance equation and the latter (2.40) is a traction boundary condition. This result should be expected because if we ignored any damage in our model then the action defined by (2.18) would reduce to a classic action. Our momentum balance equation couples the effects that damage degradation has on the elastic properties of Ω and therefore the displacement $\mathbf{u}$. This property is also present in our traction boundary condition (2.40). As a result increased damage causes the degradation function $g(\alpha)$ to approach zero and therefore softens the material stress in (2.39). This

means that at every point $\mathbf{X}$ as α increases the elastic resistance decreases, as intended.

With the results (2.39) and (2.40) we can re-evaluate our variation of the action. By doing so we can eliminate all the terms relating to the momentum balance leaving us with terms relating to the damage evolution. Therefore we rewrite the variation (2.38) as

$$\begin{aligned}\delta\Pi(\mathbf{u},\alpha,\mathbf{v},\mu) = &\int_{\mathcal{T}}\int_{\Omega}\left[-g'(\alpha)\phi(\mathbf{F}) + w'(\alpha)\psi - \frac{G}{\ell}\alpha + \nabla\cdot\left(\frac{G}{\ell}\mathbb{L}\nabla\alpha\right)\right]\mu\, dV\, dt \\ &- \int_{\mathcal{T}}\int_{\partial\Omega}\left(\frac{G}{\ell}\mathbb{L}\nabla\alpha\cdot\mathbf{n}\right)\mu\, dA\, dt,\end{aligned}$$

where $\mathbf{u}$ and α are the solutions of (2.39) and (2.40). To derive a damage evolution equation by making an argument analogous to the above, will not work here because μ is a constrained function. Therefore we need to form an argument similar to that made at the end of §2.2.3. To do this let us begin by setting $\mathbf{v} = \mathbf{0}$ and $\mu \geq 0$ everywhere in Ω. This now describes a system in which damage evolution is irreversible and $\mathbf{u}$ satisfies $\delta\Pi = 0$. According to the Karush-Khun-Tucker conditions (2.36) we can derive an integral constraint $\forall\mu(\mathbf{X}) \geq 0$ such that $\delta\Pi(\mathbf{u},\alpha,\mathbf{0},\mu) \geq 0$. Therefore, we must have that

$$\begin{aligned}&\int_{\mathcal{T}}\int_{\Omega}\left[-g'(\alpha)\phi(\mathbf{F}) + w'(\alpha)\psi - \frac{G}{\ell}\alpha + \nabla\cdot\left(\frac{G}{\ell}\mathbb{L}\nabla\alpha\right)\right]\mu\, dV\, dt \\ &- \int_{\mathcal{T}}\int_{\partial\Omega}\left(\frac{G}{\ell}\mathbb{L}\nabla\alpha\cdot\mathbf{n}\right)\mu\, dA\, dt \geq 0.\end{aligned} \tag{2.41}$$

We can not use the fundamental lemma of calculus of variations or the localisation theorem here to extract any information from the integral constraint (2.41). This is because we cannot permit every admissible scalar function μ in $\mathbb{R}$. Here the variation μ is constrained such that $\mu \geq 0$, this constraint comes from the fact that $\dot{\alpha} \geq 0$. As a result we have the constrain $\delta\Pi \geq 0$ which is an inequality, we require an equality for the fundamental lemma of calculus of variations and the localisation theorem. Instead we will introduce a new Lemma, which will allow us to extract a strong form constraint for when $\mu(\mathbf{X}) \geq 0$ is true.

Lemma 2.2.1. *If $f : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}$ is continuous $\forall n \in \mathbb{N}$ and if $\int_\Omega f(\mathbf{X})\mu(\mathbf{X})\, dV \geq 0$ $\forall\mu$ satisfying $\mu(\mathbf{X}) \geq 0$ $\forall\mathbf{X} \in \Omega$ then $f(\mathbf{X}) \geq 0$ $\forall\mathbf{X} \in \Omega$.*

Proof. Consider a function $f(\mathbf{X}) : \Omega \subseteq \mathbb{R}^n \to \mathbb{R}$ that is continuos $\forall n \in \mathbb{N}$. This function satisfies the integral constraint $\int_\Omega f(\mathbf{X})\mu(\mathbf{X})\, dV \geq 0$ $\forall\mu$ satisfying $\mu(\mathbf{X}) \geq 0$ $\forall\mathbf{X} \in \Omega$. Suppose $\exists$ a ball $\mathcal{B}$ centred at $\mathbf{X}_0$ with radius δ that is completely contained in Ω and an $\epsilon > 0$ with $f(\mathbf{X}) < -\epsilon$, $\forall\mathbf{X} \in \mathcal{B}(\mathbf{X}_0,\delta)$. If we than take μ such that

$$\mu(\mathbf{X}) = \begin{cases} 1 - \frac{|\mathbf{X}-\mathbf{X}_0|}{\delta} & \forall\mathbf{X} \in \mathcal{B}(\mathbf{X}_0,\delta), \\ 0 & \forall\mathbf{X} \in \Omega \setminus \mathcal{B}(\mathbf{X}_0,\delta), \end{cases}$$

then we can say that

$$\begin{aligned}\int_{\Omega} f(\mathbf{X})\mu(\mathbf{X})\,dV &= \int_{\mathcal{B}(\mathbf{X}_0,\delta)} f(\mathbf{X})\mu(\mathbf{X})\,dV \\ &< -\epsilon \int_{\mathcal{B}(\mathbf{X}_0,\delta)} \mu(\mathbf{X})\,dV \\ &< 0.\end{aligned}$$

This is a contradiction as $\int_{\Omega} f(\mathbf{X})\mu(\mathbf{X})\,dV \geq 0\ \forall \mu(\mathbf{X}) \geq 0$, hence we must have that $f(\mathbf{X}) \geq 0\ \forall \mathbf{X} \in \Omega$. Therefore completing the proof. □

If we know take the integral constraint (2.41) and apply the Lemma 2.2.1 it is possible to extract strong form inequality constraints. It is important to remember that when we do this we can exploit the freedom of choice for $\mu(\mathbf{X})$. We can thus write down the inequalities

$$-g'(\alpha)\phi(\mathbf{F}) + w'(\alpha)\psi - \frac{G}{\ell}\alpha + \nabla \cdot \left(\frac{G}{\ell}\mathbb{L}\nabla\alpha\right) \geq 0 \quad \text{in } \Omega \tag{2.42a}$$

$$\text{and} \quad \frac{G}{\ell}\mathbb{L}\nabla\alpha \cdot \mathbf{n} \leq 0 \quad \text{on } \partial\Omega \tag{2.42b}$$

$$\forall \mu \geq 0 \quad \text{in } \Omega. \tag{2.42c}$$

These three conditions are what ensure damage irreversibility. The first condition (2.42a) is the damage criterion, if this is not satisfied then we do not have damage growth and if it is satisfied at any point $\mathbf{X} \in \Omega$ we can have damage evolution at that point $\mathbf{X} \in \Omega$. The second condition (2.42b) is a criterion for damage evolution on the boundary $\partial\Omega$. The last inequality (2.42c) is an expression of damage irreversibility. Coupling (2.42b) with the irreversibility constraint (2.17) gives us a Neumann Boundary condition on $\partial\Omega$ for α such that

$$\mathbb{L}\nabla\alpha \cdot \mathbf{n} = 0, \tag{2.43}$$

which satisfies both inequalities.

At this point it is important that we clarify what exactly our derivation via the Karush-Khun-Tucker conditions is saying. In other words physically what do the inequalities (2.42) actually mean. Well this variational formulation is being calculated in the setting that the damage threshold is being satisfied. We are assuming that there is an energy driving the growth of α at some point $\mathbf{X} \in \Omega$. However, this is not always the case in fact we have two distinct physical regions undergoing different mechanics. The first region is everywhere that satisfies $\dot{\alpha} = 0$, at these points α just remains constant and the mechanics of our system is dictated by the governing equations (2.39) and (2.40). In these regions we would only be interested in calculating the displacement $\mathbf{u}$. The damage criterion

(2.42a) is not satisfied in these regions, you can use this inequality to check which points will be undergoing damage evolution. Physically this is the correct way to understand the model, (2.42a) is a criterion not an absolute constraint. At every region in Ω where (2.42a) is positive definite we will have that $\dot{\alpha} > 0$. In those regions the mechanics of Ω will be dictated by our momentum balance (2.39), a traction boundary condition (2.40), a damage evolution equation and a boundary condition for the damage phase field.

From here we can now make an argument to derive a strong set of governing equations for the damage evolution equation. To do this we recall that the last Karush-Khun-Tucker condition in (2.36) we need to consider, is $\delta\Pi(\mathbf{u}, \alpha, \mathbf{0}, \dot{\alpha}) = 0$. We can write this down as

$$\int_{\mathcal{T}}\int_{\Omega}\left[-g'(\alpha)\phi(\mathbf{F}) + w'(\alpha)\psi - \frac{G}{\ell}\alpha + \nabla\cdot\left(\frac{G}{\ell}\mathbb{L}\nabla\alpha\right)\right]\dot{\alpha}\,dV\,dt$$
$$-\int_{\mathcal{T}}\int_{\partial\Omega}\left(\frac{G}{\ell}\mathbb{L}\nabla\alpha\cdot\mathbf{n}\right)\dot{\alpha}\,dA\,dt = 0.$$

As $\dot{\alpha} \geq 0$ everywhere in Ω then we cannot use the fundamental lemma of calculus to derive a set of governing equations for damage evolution. However, we may apply the localisation theorem here and extract the integrands of the above integrals to form a set of governing equations for the damage phase field. From the localisation theorem we therefore have the following damage evolution equations

$$\left[-g'(\alpha)\phi(\mathbf{F}) + w'(\alpha)\psi - \frac{G}{\ell}\alpha + \nabla\cdot\left(\frac{G}{\ell}\mathbb{L}\nabla\alpha\right)\right]\dot{\alpha} = 0 \quad \text{in } \Omega, \tag{2.44}$$

$$\left[\frac{G}{\ell}\mathbb{L}\nabla\alpha\cdot\mathbf{n}\right]\dot{\alpha} = 0 \quad \text{on } \partial\Omega. \tag{2.45}$$

Since these expressions are only true for the irreversibility constraint $\dot{\alpha} \geq 0$ we cannot eliminate the $\dot{\alpha}$ terms from the left hand side of (2.44) and (2.45). Here (2.44) is a damage evolution equation which dictates how α changes with respect to deformations defined by $\mathbf{u}$. We also have another boundary condition defined by (2.45), similar to the previous boundary condition (2.43) we have already derived.

Hence we have the following system of equations for damage evolution

$$\left[-g'(\alpha)\phi(\mathbf{F}) + w'(\alpha)\psi - \frac{G}{\ell}\alpha + \nabla\cdot\left(\frac{G}{\ell}\mathbb{L}\nabla\alpha\right)\right]\dot{\alpha} = 0 \quad \text{in } \Omega, \tag{2.46a}$$

$$\mathbb{L}\nabla\alpha\cdot\mathbf{n} = 0 \quad \text{on } \partial\Omega, \tag{2.46b}$$

$$\dot{\alpha} \geq 0 \quad \text{in } \Omega. \tag{2.46c}$$

These equations and inequality act as a complete system describing the damage evolution through Ω. The first equation is an energy balance dictating damage evolution. The second is a no-flux boundary condition, that ensure damage in a material cannot leave one material

and enter another. Our damage evolution equation clearly treats the elastic potential ϕ of Ω as a driving energy behind damage evolution and uses the damage threshold ψ as an energetic threshold for initial damage evolution. This system is better understood when coupled with a damage criterion that controls where damage takes place. This criterion for damage evolution at every point $\mathbf{X} \in \Omega$ is

$$-g'(\alpha)\phi(\mathbf{F}) + w'(\alpha)\psi - \frac{G}{\ell}\alpha + \nabla \cdot \left(\frac{G}{\ell}\mathbb{L}\nabla\alpha\right) \geq 0. \tag{2.47}$$

Note that the equations (2.46a) and (2.47) are independent of each other. We also need to be careful not to confuse the fact that (2.46a) is a strict equality and (2.47) is a criterion that prescribes when and where damage evolution takes place in Ω.

We can simplify the system of equations (2.46) by introducing new notation. This notation must ensure that damage evolution only occurs when the damage criterion (2.47) is satisfied and is driven by a positive contribution of energy. Accordingly we introduce the following notation for any positive scalar function f that ensures only positive contributions of f are accounted for

$$\langle f(\mathbf{X})\rangle_+ = \begin{cases} f(\mathbf{X}) & \text{if } f(\mathbf{X}) \geq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2.48}$$

We can think of this as a ramp function, known as the Macaulay brackets [89, 31]. They are commonly used in many engineering problems where irreversibility is an important requirement. To understand how this notation can simplify the system of equations (2.46) let us define

$$F(\mathbf{X}) = -g'(\alpha)\phi(\mathbf{F}) + w'(\alpha)\psi - \frac{G}{\ell}\alpha + \nabla \cdot \left(\frac{G}{\ell}\mathbb{L}\nabla\alpha\right) \quad \text{in } \Omega.$$

Hence we may write a simplified system of equations

$$\begin{aligned} \dot{\alpha} &\geq 0 \quad \text{in } \Omega, \\ F(\mathbf{X})\dot{\alpha} &= 0 \quad \text{in } \Omega. \end{aligned}$$

The first equation being the damage irreversibility constraint and the second equation being the damage evolution equation. For this argument we may ignore the boundary condition (2.46b). The above two equations can be combined into one expression using the Macaulay brackets. We can do this by writing

$$\langle F(\mathbf{X})\rangle_+ = 0,$$

thus combining both equations. The reason this works is rather simple. We have two distinct regions for damage evolution. The first being the region $\Omega_\alpha \subseteq \Omega$ that contains all $\mathbf{X} \in \Omega$ such that $F(\mathbf{X}) \geq 0$; and the second region being $\Omega_E \subseteq \Omega$ that contains all $\mathbf{X} \in \Omega$ such that $F(\mathbf{X}) < 0$. Regions of damage evolution and no damage evolution, Ω_α and Ω_E, respectively. We have defined Ω_α such that we have $\dot{\alpha} > 0 \quad \forall \mathbf{X} \in \Omega_\alpha$ therefore

$$F(\mathbf{X})\dot{\alpha} = 0 \iff F(\mathbf{X}) = 0 \quad \forall \mathbf{X} \in \Omega_\alpha.$$

Therefore when we have damage growth in Ω_α it is defined by the damage evolution equation $F(\mathbf{X}) = 0$. This may seems as an ill defined damage evolution equation without boundary conditions but it does implicitly have boundary conditions. If $\partial\Omega_\alpha \subseteq \partial\Omega$ then we apply the no flux boundary condition (2.46b) but if $\partial\Omega_\alpha \subseteq \overset{\circ}{\Omega}$ then we simply apply a Dirichlet boundary conditions defined by the surrounding fixed values of α. Now consider the region Ω_E where we have no damage evolution such that $\dot{\alpha} = 0 \quad \forall \mathbf{X} \in \Omega_E$ and therefore

$$\dot{\alpha} = 0 \iff F(\mathbf{X}) < 0.$$

These two cases can be constructed into a single expression by using the Macaulay brackets. Physically we are saying that the damage evolution is governed only by the positive definite energy contributions of the damage criterion (2.47). As a result we may write the simplified system of equations

$$\left\langle -g'(\alpha)\phi(\mathbf{F}) + w'(\alpha)\psi - \frac{G}{\ell}\alpha + \nabla \cdot \left(\frac{G}{\ell}\mathbb{L}\nabla\alpha\right)\right\rangle_+ = 0 \quad \text{in } \Omega, \tag{2.49}$$

$$\mathbb{L}\nabla\alpha \cdot \mathbf{n} = 0 \quad \text{on } \partial\Omega. \tag{2.50}$$

Here we are left with a balance of energy (2.49) which acts as our damage evolution equation. In essence this is a piece-wise equation where the left hand side is either positive at some point $\mathbf{X}$ hence implying damage evolution at that point to balance the equation, otherwise it is zero and at that point we have no damage evolution. This therefore ensures that damage irreversibility conditions are in fact satisfied. Pairing the damage evolution equation with the boundary condition described in (2.50) we have a complete system describing damage. This paired with the system describing the displacement gives us a complete coupled set of partial differential equations with boundary conditions describing the mechanics and damage mechanics in Ω.

Consider equation (2.49), this is a rate independent equation of damage evolution. Which means that damage evolution will happen instantaneously. This leads to a lot of problems when trying to solve the equation numerically. To that end we introduce a viscous term to our equation of damage evolution which as a result makes the evolution of damage smoother. We can also therefore use numerical schemes such as the Crank-

Nicholson method, which generally converge, to make solving our system computationally feasible. The reason this works is because it effectively changes the definition of our damage evolution equation from an elliptic type to a parabolic type and we can therefore solve the model with numerical schemes that are far more likely to converge. Hence we rewrite equation (2.49) as

$$\eta\dot{\alpha} = \left\langle -g'(\alpha)\phi(\mathbf{F}) + w'(\alpha)\psi - \frac{G}{\ell}\alpha + \nabla\cdot\left(\frac{G}{\ell}\mathbb{L}\nabla\alpha\right)\right\rangle_{+} \qquad \text{in } \Omega. \tag{2.51}$$

Here we have introduced the parameter $\eta(\mathbf{X})$ as the damage viscosity. We therefore now have a rate dependent model which can recover the rate independent model by taking $\eta \to 0$. This idea can be seen in the literature with other models introducing a viscous term for numerical reasons [28]. For clarity not that the term $\dot{\alpha}$ on the left hand side of (2.51) is a Lagrangian time derivative, as this equation is defined in the reference configuration Ω.

This model is not unique to the literature, its derivation and form share many similarities to other works. One feature however that we have introduced here is the energetic threshold function ψ, as defined by (2.28) in §2.2.1. As a result we can imbue any material we are modelling with some natural resistance to damage. It also means that we have a model that can be used to simulate complex mechanical problems where damage growth may never take place. In the literature of the damage phase field model a threshold function is never considered [84, 90, 31, 54], which may mean other models cannot accurately assess the onset or termination of material failure. If we consider a material in a state of no damage such that $\alpha = 0$ and $\nabla\alpha = 0$ everywhere in Ω. Then the damage criterion (2.47) that would first need to be met such that damage evolution occurs would be

$$-g'(\alpha)\phi \geq -w'(\alpha)\psi.$$

Here we are simply saying that for the initiation of damage in Ω the elastic potential must be greater than the required threshold energy.

2.3 Governing equations

For convenience we lay out the governing equations in Ω as a single system of partial differential equations. For a single continuum Ω we have a coupled pair of partial differential equations, (2.39) and (2.51), describing the displacement $\mathbf{u}$ and damage α. This is paired with two Neumann style boundary conditions completing the system, (2.40) and (2.50). Our system is written as

$$\begin{aligned}
&\rho\mathbf{b} + \nabla\cdot\left(g(\alpha)\frac{\partial\phi}{\partial\mathbf{F}}\right) = \rho\ddot{\mathbf{u}} \quad \text{in } \Omega,\\
&\eta\dot{\alpha} = \left\langle -g'(\alpha)\phi(\mathbf{F}) + w'(\alpha)\psi - \frac{G}{\ell}\alpha + \nabla\cdot\left(\frac{G}{\ell}\mathbb{L}\nabla\alpha\right)\right\rangle_+ \quad \text{in } \Omega,\\
&\mathbf{t} = g(\alpha)\frac{\partial\phi}{\partial\mathbf{F}}\mathbf{n} \quad \text{and} \quad \frac{G}{\ell}\mathbb{L}\nabla\alpha\cdot\mathbf{n} = 0 \quad \text{on } \partial\Omega.
\end{aligned} \tag{2.52}$$

The first equation in our system is a linear momentum balance equation describing the displacement $\mathbf{u}$ of Ω. This is coupled with the damage evolution via the degradation function $g(\alpha)$. Damage growth leads to a degradation of elastic properties, resulting in material softening. This is done via softening of the stress term in our momentum balance equation. The second equation in our system is the damage evolution equation governing the damage phase field α. On the right hand side of this equation we can see a pair of Macaulay brackets that filter out any negative energy contributions to damage evolution. Therefore driving an irreversible damage growth in the positive direction. The damage evolution equation is coupled to the linear momentum equation via the elastic strain energy function ϕ. To close this differential problem we have two Neumann style boundary conditions for the displacement and damage on $\partial\Omega$, respectively. The first boundary condition is a traction boundary condition for the linear momentum balance equation. The second boundary condition is a flux condition for the damage evolution equation. It is a no-flux boundary condition, which means that α cannot leave the domain Ω. Physically you can interpret that as a tear being confined to the body that is being torn. For example, if you start snapping a piece of wood, a crack will run down its body. When that crack reaches the boundary of the wood it does not keep travelling and then suddenly tear apart the bodies surrounding it, like the air. No, it is confined to the wood.

2.3.1 Rate independent model

A convenient feature of our model is the rate dependent property dictated by the viscosity term η. Our problem will need to be solved numerically and a viscosity term makes solving our system more numerically convenient, as it ensures our model is a parabolic type partial differential equation which can be solved with reliable numerical schemes such as the Crank-Nicholson scheme. Also rate dependency is a common material property of many materials. Except many materials are modelled as rate independent. To model such materials we may simply set $\eta = 0$ in (2.52). As a result our rate dependent model reduces to the model defined by (2.39), (2.40), (2.46) and the damage criterion (2.47). This model is almost identical in every way to the rate independent version. Obviously this version is also irreversible and satisfies every other constitutive property previously mentioned. Though it should be noted this model is much harder to work with numerically. Damage growth happens instantaneously in this model meaning that the elastic properties degrade

near instantly making it easier to deform a material. As a result its far more difficult to create a numerical scheme that converges well for all possible cases.

2.4 Adaption for composite structures

Most materials are not homogeneous materials but are in fact extremely heterogeneous. This is especially true in biological tissues and materials with hierarchical structures. To avoid a phenomenological approach of modelling material failure we propose modelling the entire microstructure of a complex material. Doing this would allow one to attempt multiscale modelling from this large microscopic model. To do this we the materials as a continuum made up of many smaller continua, forming a composite structure. In figure 2.4 we propose an approximation of a linear elastic composite. Our idea is to simplify complex materials by treating them as a composite of elastic continua. We are motivated to do this because there is a lot of mathematical precedent for multiscale modelling of composite materials. Since the structure of this setup would be similar to the problem proposed in §2.2 we would only need to consider an adaption of the variational formulation to derive a governing system of equations for a linear elastic composite.

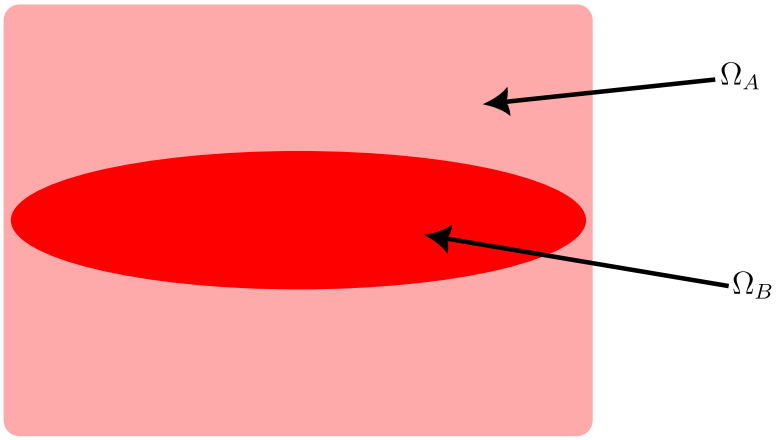


Figure 2.4: Here we have a host matrix Ω_A which acts as the glue holding a composite material together. The host matrix is full of inclusions that define the subdomain Ω_B.

Formally we can setup a composite materials as a host matrix with a number of disjoint inclusions, as illustrated in Figure 2.4. We represent the composite as a bounded continuum $\Omega \subset \mathbb{R}^3$, such that $\bar{\Omega} = \bar{\Omega}_A \cup \bar{\Omega}_B$ and $\overset{\circ}{\Omega}_A \cap \overset{\circ}{\Omega}_B = \emptyset$. Here, Ω_A represents a host matrix and

$$\Omega_B = \bigcup_{\beta=1}^{N} \Omega_\beta$$

represents a number N of disjoint embedded inclusions Ω_β. Our damage model was setup over a single continuum but now we must adapt it to this composite setup. This will result in a momentum balance equation and damage evolution equation for every constituent

part of the composite. For completion of our differential equations we require a set of boundary conditions and interface conditions. Therefore we must define our interfaces and boundaries of the composite as

$$\Gamma_\beta = \partial\Omega_A \cap \partial\Omega_\beta \quad \text{and} \quad \partial\Omega = (\partial\Omega_A \cup \partial\Omega_B) \setminus \left(\bigcup_{\beta=1}^{N} \Gamma_\beta \right).$$

$\partial\Omega$ is the boundary surface of our composite made up of a union of the boundary surfaces of the host matrix and the inclusions. The subscript A throughout will indicate the host matrix and the subscript B will indicate the domain of disjoint inclusions. Whereas the subscript β indicates an individual inclusion in Ω_B for $\beta = 1, ..., N$. To avoid any confusion, let us also state that all the inclusions included in Ω_B never intersect or even touch. We write this as

$$\Omega_i \bigcap_{i \neq j} \Omega_j = \emptyset \quad \forall \Omega_i, \Omega_j \subseteq \Omega_B.$$

2.4.1 Adaption of variational formulation

In the previous sections we discussed a derivation for a damage model over a single continuum Ω. Now we want to model damage growth over $N+1$ continua represented by Ω_A and Ω_B. A straightforward approach would be to intuitively adapt (2.52) but we would then have to effectively guess the interface conditions between subdomains. Therefore, for completeness, we will replicate the derivation of §2.2 for the composite setup. This will allow us to derive the necessary interface conditions.

To begin we must introduce new functionals of the kinetic, body force, threshold, potential and dissipation energies. This can be done by adapting the functionals (2.19), (2.20), (2.28), (2.27) and (2.25), respectively. As an example we will begin by writing down a new functional for the kinetic energy. By summing up the contributions of kinetic energy over our composite setup we can calculate the total kinetic energy. Hence, we write

$$K[\dot{\mathbf{u}}_A, \dot{\mathbf{u}}_\beta] = \int_{\Omega_A} \frac{1}{2} \rho_A \dot{\mathbf{u}}_A \cdot \dot{\mathbf{u}}_A \, dV + \sum_{\beta=1}^{N} \int_{\Omega_\beta} \frac{1}{2} \rho_\beta \dot{\mathbf{u}}_\beta \cdot \dot{\mathbf{u}}_\beta \, dV \tag{2.53}$$

where, for every $\mathbf{X} \in \Omega$ our displacement $\mathbf{u}_A$ and $\mathbf{u}_\beta$ denotes the restricted displacement of $\mathbf{u}$ in Ω_A and Ω_β, respectively. Notice that we have defined two functions of density $\rho_A(\mathbf{X})$ and $\rho_\beta(\mathbf{X})$ in each domain Ω_A and Ω_β, respectively. Next we calculate the functional for

body and surface forces as

$$\begin{aligned} B[\mathbf{u}_A, \mathbf{u}_\beta] = &\int_{\Omega_A} \rho_A \mathbf{b} \cdot \mathbf{u}_A \, dV + \sum_{\beta=1}^{N} \int_{\Omega_\beta} \rho_\beta \mathbf{b} \cdot \mathbf{u}_\beta \, dV \\ &+ \int_{\partial\Omega \cap \partial\Omega_A} \mathbf{t} \cdot \mathbf{u}_A \, dA + \sum_{\beta=1}^{N} \int_{\partial\Omega \cap \partial\Omega_\beta} \mathbf{t} \cdot \mathbf{u}_\beta \, dA. \end{aligned} \tag{2.54}$$

Here for the body force $\mathbf{b}(\mathbf{X})$ and $\mathbf{t}(\mathbf{X})$ we do not partition these contributions because they are external to Ω.

In a similar fashion we want to reformulate the potential energy functional (2.27). We can write this down as the new potential energy functional

$$P[\mathbf{u}_A, \mathbf{u}_\beta, \alpha_A, \alpha_\beta] = \int_{\Omega_A} g_A(\alpha_A)\phi_A(\mathbf{F}_A) \, dV + \sum_{\beta=1}^{N} \int_{\Omega_\beta} g_\beta(\alpha_\beta)\phi_\beta(\mathbf{F}_\beta) \, dV. \tag{2.55}$$

Here we define the elastic potentials for the domains Ω_A and Ω_β as ϕ_A and ϕ_β, respectively. The deformation gradients $\mathbf{F}_A$ and $\mathbf{F}_\beta$ have been partitioned to each of their domains Ω_A and Ω_β, respectively. The degradation functions g_A and g_β for the elastic potentials in the domains Ω_A and Ω_β, respectively. The threshold functional now becomes for the composite domain

$$T[\alpha_A, \alpha_\beta, \mathbf{X}] = \int_{\Omega_A} w_A(\alpha_A)\psi_A(\mathbf{X}) \, dV + \sum_{\beta=1}^{N} \int_{\Omega_\beta} w_{B_\beta}(\alpha_\beta)\psi_\beta(\mathbf{X}) \, dV. \tag{2.56}$$

The thresholds and degradation functions are defined as w_A and ψ_A for Ω_A, analogously the threshold and degradation is w_β and ψ_β for Ω_β. We also introduce for every $\mathbf{X} \in \Omega$ our damage phase fields α_A and α_β denoting the restricted damage in Ω_A and Ω_β, respectively.

Lastly we need to introduce a new dissipation functional for our composite domain. Considering the dissipation functional (2.25) then its clear that we can easily reformulate a new functional for the composite. This is done by summing up the dissipation for every subdomain in Ω such that

$$D[\gamma_A, \gamma_\beta] = \int_{\Omega_A} G_A \gamma_A \, dV + \sum_{\beta=1}^{N} \int_{\Omega_\beta} G_\beta \gamma_\beta \, dV$$

Here we are accounting for the energy dissipated to create new tear surface in both Ω_A and Ω_B. The energy per unit surface area parameters $G_A(\mathbf{X})$ and $G_\beta(\mathbf{X})$ are defined for their respective domains Ω_A and Ω_β. Now we have also defined the damage surface area densities γ_A in γ_β which control the shape of the damage tear field through their respective domains Ω_A and Ω_β. In our case we want a nice sharp tear that accounts for

local orientations of fibers. Therefore we recall the definition (2.15) and by substituting for γ_A and γ_β into the above functional we find that

$$\begin{aligned} D[\alpha_A, \alpha_\beta, \nabla\alpha_A, \nabla\alpha_\beta] = &\int_{\Omega_A} \frac{G_A}{2\ell} \left(\alpha_A^2 + \nabla\alpha_A \cdot \mathbb{L}_A \nabla\alpha_A\right) \, dV \\ &+ \sum_{\beta=1}^{N} \int_{\Omega_\beta} \frac{G_\beta}{2\ell} \left(\alpha_\beta^2 + \nabla\alpha_\beta \cdot \mathbb{L}_\beta \nabla\alpha_\beta\right) \, dV \end{aligned} \tag{2.57}$$

The last thing we need to introduce are assumptions about continuity at the interfaces Γ_β for all β. First we will assume that in Ω we have continuity of the damage phase field on the interfaces. We can formulate this as

$$\alpha_A = \alpha_\beta \quad \text{on } \Gamma_\beta \quad \forall\beta,$$

thereby ensuring continuity over the interfaces for the damage phase field. We make this assumption for damage so that we can firstly formulate a simple interface condition and secondly perform a straightforward multiscale formulation in the following chapter. We will also assume a continuity of displacement on the interface Γ_β for all β. In other words we have assumed that on the interface one can say

$$\mathbf{u}_A = \mathbf{u}_\beta \quad \text{and} \quad \mathbf{n}_A = -\mathbf{n}_\beta \quad \text{on } \Gamma_\beta \quad \forall\beta.$$

Here $\mathbf{n}_A$ and $\mathbf{n}_\beta$ are unit normal vectors pointing outwards from the surfaces of there respective domains Ω_A and Ω_β.

Now that we have defined a functional of kinetic, body, threshold, potential and dissipative energies for the composite domain Ω, we set about defining a balance of energy analogous to that in §2.2. So by balancing these contributions of energy we can reformulate the an action as defined in (2.18). Using the new functionals (2.53), (2.54), (2.55), (2.56) and (2.57) we have the new action

$$\begin{aligned} \Pi = &\int_{\mathcal{T}} \int_{\Omega_A} \frac{1}{2}\rho_A \dot{\mathbf{u}}_A \cdot \dot{\mathbf{u}}_A + \rho_A \mathbf{b} \cdot \mathbf{u}_A + w_A(\alpha_A)\psi_A \\ &- g_A(\alpha_A)\phi_A(\mathbf{F}_A) - \frac{G_A}{2\ell} \left(\alpha_A^2 + \nabla\alpha_A \cdot \mathbb{L}_A \nabla\alpha_A\right) \, dV \, dt \\ &+ \sum_{\beta=1}^{N} \int_{\mathcal{T}} \int_{\Omega_\beta} \frac{1}{2}\rho_\beta \dot{\mathbf{u}}_\beta \cdot \dot{\mathbf{u}}_\beta + \rho \mathbf{b} \cdot \mathbf{u}_\beta + w_\beta(\alpha_\beta)\psi_\beta \\ &- g_\beta(\alpha_\beta)\phi_\beta(\mathbf{F}_\beta) - \frac{G_\beta}{2\ell} \left(\alpha_\beta^2 + \nabla\alpha_\beta \cdot \mathbb{L}_\beta \nabla\alpha_\beta\right) \, dV \, dt \\ &+ \int_{\mathcal{T}} \int_{\partial\Omega \cap \partial\Omega_A} \mathbf{t}_A \cdot \mathbf{u}_A \, dA \, dt + \sum_{\beta=1}^{N} \int_{\mathcal{T}} \int_{\partial\Omega \cap \partial\Omega_\beta} \mathbf{t}_\beta \cdot \mathbf{u}_\beta \, dA \, dt. \end{aligned}$$

The above action is considerably larger than in the previous derivation however it is notably similar. The next step in this derivation is to calculate the variation of the action, $\delta\Pi$. This is done by taking a directional derivative of the above action in the direction $(\mathbf{u} + h\mathbf{v}, \alpha + h\mu)$, as defined in (2.37). Here $\mathbf{v} \in \mathbb{R}^n$ are admissible variations of the displacement and $\mu \geq 0$ is admissible variations of the damage phase field. These variations are again equipped with the boundary conditions

$$\mathbf{v}(t_0) = \mathbf{v}(t_1) = \mathbf{0} \quad \text{and} \quad \mu(t_0) = \mu(t_1) = 0,$$

such that $\mathcal{T} := [t_0, t_1]$. We of course appropriately partition these variations over the composite domain defined by Ω. Calculating the variation of action gives us

$$\begin{aligned}
\delta\Pi = &\int_{\mathcal{T}} \int_{\Omega_A} \left[-\rho_A \ddot{\mathbf{u}}_A + \rho_A \mathbf{b} + \nabla \cdot \left(g_A \frac{\partial \phi_A}{\partial \mathbf{F}_A} \right) \right] \cdot \mathbf{v}_A \\
&+ \left[-g_A'(\alpha_A)\phi_A(\mathbf{F}_A) + w_A'(\alpha_A)\psi_A - \frac{G_A}{\ell}\alpha_A + \nabla \cdot \left(\frac{G_A}{\ell} \mathbb{L}_A \nabla \alpha_A \right) \right] \mu_A \, dV \, dt \\
&+ \sum_{\beta=1}^{N} \int_{\mathcal{T}} \int_{\Omega_\beta} \left[-\rho_\beta \ddot{\mathbf{u}}_\beta + \rho_\beta \mathbf{b} + \nabla \cdot \left(g_\beta \frac{\partial \phi_\beta}{\partial \mathbf{F}_\beta} \right) \right] \cdot \mathbf{v}_\beta \\
&+ \left[-g_\beta'(\alpha_\beta)\phi_\beta(\mathbf{F}_\beta) + w_\beta'(\alpha_\beta)\psi_\beta - \frac{G_\beta}{\ell}\alpha_\beta + \nabla \cdot \left(\frac{G_\beta}{\ell} \mathbb{L}_\beta \nabla \alpha_\beta \right) \right] \mu_\beta \, dV \, dt \\
&+ \sum_{\beta=1}^{N} \int_{\mathcal{T}} \int_{\Gamma_\beta} \left[g_A(\alpha_A) \frac{\partial \phi_A}{\partial \mathbf{F}_A} - g_\beta(\alpha_\beta) \frac{\partial \phi_\beta}{\partial \mathbf{F}_\beta} \right] \mathbf{n}_A \cdot \mathbf{v}_A \\
&+ \left[\frac{G_A}{\ell} \mathbb{L}_A \nabla \alpha_A - \frac{G_\beta}{\ell} \mathbb{L}_\beta \nabla \alpha_\beta \right] \cdot \mathbf{n}_A \mu_A \, dV \, dt \\
&+ \int_{\mathcal{T}} \int_{\partial\Omega \cap \partial\Omega_A} \left[\mathbf{t}_A - \frac{\partial \phi_A}{\partial \mathbf{F}_A} \mathbf{n}_A \right] \cdot \mathbf{v}_A - \left(\frac{G_A}{\ell} \mathbb{L}_A \nabla \alpha_A \cdot \mathbf{n} \right) \mu_A \, dA \, dt \\
&+ \sum_{\beta=1}^{N} \int_{\mathcal{T}} \int_{\partial\Omega \cap \partial\Omega_\beta} \left[\mathbf{t}_\beta - \frac{\partial \phi_\beta}{\partial \mathbf{F}_\beta} \mathbf{n}_\beta \right] \cdot \mathbf{v}_\beta - \left(\frac{G_\beta}{\ell} \mathbb{L}_\beta \nabla \alpha_\beta \cdot \mathbf{n}_\beta \right) \mu_\beta \, dA \, dt.
\end{aligned} \tag{2.58}$$

In this variation we can see clear strong form equations that will make up our governing equations for the composite and their complementary boundary conditions. We have already derived these strong form equations and boundary conditions for a single arbitrary continuum. They are the equation of momentum balance (2.39), damage evolution (2.46a), the traction boundary condition (2.40) and the flux boundary condition (2.46b), found in §2.2. Here we can apply them directly in the above variational formulation and eliminate a number of terms. We can therefore say that the first four lines and the last two lines of

the variation (2.58) go to zero. As a result the above variation reduces to

$$\begin{aligned}\delta\Pi = \sum_{\beta=1}^{N} \int_{\mathcal{T}} \int_{\Gamma_\beta} \left[g_A(\alpha_A) \frac{\partial \phi_A}{\partial \mathbf{F}_A} - g_\beta(\alpha_\beta) \frac{\partial \phi_\beta}{\partial \mathbf{F}_\beta} \right] \mathbf{n}_A \cdot \mathbf{v}_A \\ + \left[\frac{G_A}{\ell} \mathbb{L}_A \nabla \alpha_A - \frac{G_\beta}{\ell} \mathbb{L}_\beta \nabla \alpha_\beta \right] \cdot \mathbf{n}_A \mu_A \, dV \, dt.\end{aligned} \tag{2.59}$$

This variation of the action is now formed of what are clearly interface conditions. To extract them we will follow the same process laid out in §2.2. So to begin we acknowledge that the above action is true for all admissible forms of $\mathbf{v}$ and μ. Our variation of action (2.59) applies for our arbitrary domains defining the host matrix Ω_A and the inclusions Ω_B.

To begin we will consider no variation of the damage phase field by setting $\mu_A = 0$ and $\mu_\beta = 0$ for all β. Doing so we are only left with a variation of the action due to variations in the displacement. As the admissible variations of displacement $\mathbf{v}_A$ and $\mathbf{v}_\beta$ are unrestricted then to find a minimum of $\delta\Pi$ we just set $\delta\Pi = 0$. This leaves us with the expression

$$0 = \sum_{\beta=1}^{N} \int_{\Gamma_\beta} \left[g_A(\alpha_A) \frac{\partial \phi_A}{\partial \mathbf{F}_A} - g_\beta(\alpha_\beta) \frac{\partial \phi_\beta}{\partial \mathbf{F}_\beta} \right] \mathbf{n}_A \cdot \mathbf{v}_A \, dA.$$

If we then exploit the fundamental Lemma of calculus of variations we can derive a set of interface conditions. Therefore, we find that

$$g_A(\alpha_A) \frac{\partial \phi_A}{\partial \mathbf{F}_A} \cdot \mathbf{n}_A = g_\beta(\alpha_\beta) \frac{\partial \phi_\beta}{\partial \mathbf{F}_\beta} \cdot \mathbf{n}_A \text{ on } \Gamma_\beta \, \forall \beta. \tag{2.60}$$

This interface condition is a stress continuity on the interfaces defined by $\Gamma_\beta \, \forall \beta$. Plugging this result back into the variation of action (2.59) leaves us with

$$\delta\Pi = \sum_{\beta=1}^{N} \int_{\Gamma_\beta} \left[\frac{G_A}{\ell} \mathbb{L}_A \nabla \alpha_A - \frac{G_\beta}{\ell} \mathbb{L}_\beta \nabla \alpha_\beta \right] \cdot \mathbf{n}_A \, \mu_A \, dA$$

which is true for all admissible damage variations μ_A. Here we do not need to give consideration to the Karush-Khun-Tucker conditions because the remaining variation of action does not drive damage evolution. In fact, what remains is simply an interface condition. Therefore given that each Γ_β is an arbitrary surface then its clear that, by the fundamental Lemma of calculus of variations, we must have

$$\frac{G_A}{\ell} \mathbb{L}_A \nabla \alpha_A \cdot \mathbf{n}_A = \frac{G_\beta}{\ell} \mathbb{L}_\beta \nabla \alpha_\beta \cdot \mathbf{n}_A \text{ on } \Gamma_\beta \, \forall \beta. \tag{2.61}$$

Which is an interface condition for the damage phase fields throughout our composite.

With these interface conditions, the governing equations and boundary equations defined in (2.52) and the continuities of displacement and damage we can describe the coupled damage evolution and linear momentum balance for a linear elastic composite material, as defined by the continuum Ω.

2.5 Summary of results

In this chapter we have introduced the damage phase field model and explained how one can use it to model sharp tears in a continuum. From there we have used a variational formulation to derive a governing set of equations. This was done by introducing the Karush-Khun-Tucker conditions coupled with Lemma 2.2.1 to complete a variational formulation of our governing equations. We have adapted our governing materials to a linear elastic composite domain. To the best of this authors knowledge this is the first time this has been done. With this model we will show it is possible to derive a novel multiscale model of the damage phase field model in the next chapter. This model will have many applications for modelling tears and fractures in complex hierarchical materials that can not be described as simply homogeneous. It will allow researchers to model the failure of biological tissues from a microscopic perspective.

We understand this model is not simple, so we will write it out clearly here for convenience. Remember we have defined a system of partial differential equations that explain the mechanics and damage evolution for a composite domain Ω. This composite domain is formed by a host matrix Ω_A and the inclusions Ω_β for $\beta = 1, ..., N$. The interface between the inclusion and host matrix is denoted as Γ_β for all β. Let us first write the governing equations

$$\begin{aligned}
&\rho_A \mathbf{b} + \nabla \cdot \left(g_A(\alpha_A) \frac{\partial \phi_A}{\partial \mathbf{F}_A} \right) = \rho_A \ddot{\mathbf{u}}_A \quad \text{in } \Omega_A, \\
&\rho_\beta \mathbf{b} + \nabla \cdot \left(g_\beta(\alpha_\beta) \frac{\partial \phi_\beta}{\partial \mathbf{F}_\beta} \right) = \rho_\beta \ddot{\mathbf{u}}_\beta \quad \text{in } \Omega_\beta \, \forall \beta, \\
&\eta_A \dot{\alpha}_A = \left\langle -g_A'(\alpha)\phi_A(\mathbf{F}_A) + w_A'(\alpha)\psi_A - \frac{G_A}{\ell}\alpha + \nabla \cdot \left(\frac{G_A}{\ell} \mathbb{L}_A \nabla \alpha_A \right) \right\rangle_+ \quad \text{in } \Omega_A, \\
&\eta_\beta \dot{\alpha}_\beta = \left\langle -g_\beta'(\alpha)\phi_\beta(\mathbf{F}_\beta) + w_\beta'(\alpha)\psi_\beta - \frac{G_\beta}{\ell}\alpha_\beta + \nabla \cdot \left(\frac{G_\beta}{\ell} \mathbb{L}_\beta \nabla \alpha_\beta \right) \right\rangle_+ \quad \text{in } \Omega_\beta \, \forall \beta.
\end{aligned} \tag{2.62}$$

The first two equations are linear momentum balance equations. The next two are damage evolution equations. These equations show a direct relationship between the displacement and damage in the system. Here the terms η_A and η_β are viscosity terms, introduced to ensure we can solve our model numerically with computationally robust methods like the Crank-Nicholson scheme. If we want to recover the rate independent version of the model then we simply take $\eta_A = \eta_\beta = 0 \ \forall \beta$. This system is of course completed with a set of

boundary equations and interface conditions

$$
\begin{aligned}
&\mathbf{t} = g_A(\alpha_A)\frac{\partial \phi_A}{\partial \mathbf{F}_A}\mathbf{n}_A \quad \text{and} \quad \mathbf{t} = g_\beta(\alpha_\beta)\frac{\partial \phi_\beta}{\partial \mathbf{F}_\beta}\mathbf{n}_A \quad \text{on } \partial\Omega \quad \forall\beta,\\
&\frac{G_A}{\ell}\mathbb{L}_A\nabla\alpha_A\cdot\mathbf{n}_A = 0 \quad \text{and} \quad \frac{G_\beta}{\ell}\mathbb{L}_\beta\nabla\alpha_\beta\cdot\mathbf{n}_\beta = 0 \quad \text{on } \partial\Omega \quad \forall\beta,\\
&g_A(\alpha_A)\frac{\partial \phi_A}{\partial \mathbf{F}_A}\cdot\mathbf{n}_A = g_\beta(\alpha_\beta)\frac{\partial \phi_\beta}{\partial \mathbf{F}_\beta}\cdot\mathbf{n}_A \quad \text{on } \Gamma_\beta \quad \forall\beta,\\
&\frac{G_A}{\ell}\mathbb{L}_A\nabla\alpha_A\cdot\mathbf{n}_A = \frac{G_\beta}{\ell}\mathbb{L}_\beta\nabla\alpha_\beta\cdot\mathbf{n}_A \quad \text{on } \Gamma_\beta \quad \forall\beta,\\
&\mathbf{u}_A = \mathbf{u}_\beta \quad \text{and} \quad \alpha_A = \alpha_\beta \quad \text{on } \Gamma_\beta \quad \forall\beta.
\end{aligned}
\tag{2.63}
$$

The first two boundary conditions are traction boundary conditions for the linear momentum balance equations. The next two are flux boundary conditions for the damage evolution equations. Next we have a continuity of stress and flux condition on the interfaces Γ_β. This system is complete with continuity of the damage phase field and displacement on the interfaces.

2.5.1 Concluding remarks

The overall purpose of introducing this composite damage phase field model (2.62) is to allow us to model aortic dissections from a microscopic perspective. Now that we have a general model for elastic composites, we need to consider how we would approximate the aortic microstructure. As discussed in Chapter 1, the aortic wall is a multilayered structure made up of many constitutive parts. It is important to note that the majority of dissections are contained within the tunica media. If we then revisit the media's anatomy we can begin to create an approximate domain geometry for the aortic microstructure as an elastic composite. As a result, we can adapt our damage phase field model for dissections in the aortic microstructure. Therefore, creating a large set of partial differential equations, boundary conditions and interface conditions, which approximate the displacement and damage of the aortic wall at the microscale.

Recalling §1.1.1 we can draw a simple schematic of the tunica media in Figure 2.5. Here we are describing a small section of the media as circumferential layers of smooth muscle cells which are surrounded by an extracellular matrix. Approximately the tunica media is: 33.5% smooth muscle cells, 37% collagen and 24.5% elastin [13]. The complimentary 5% would be made up of ground substances such as glycosaminoglycan and proteoglycans. Importantly, this is only the dry mass of an artery which makes up around 30% of the arteries mass, because in reality 70% of an artery is water. From this microscopic perspective we cannot just treat the aorta as a simple single elastic continuum approximation. In fact, we need to model this as a composite of many continua representing many different elastic constitutive parts of the tunica media. To do this we just consider a simple host

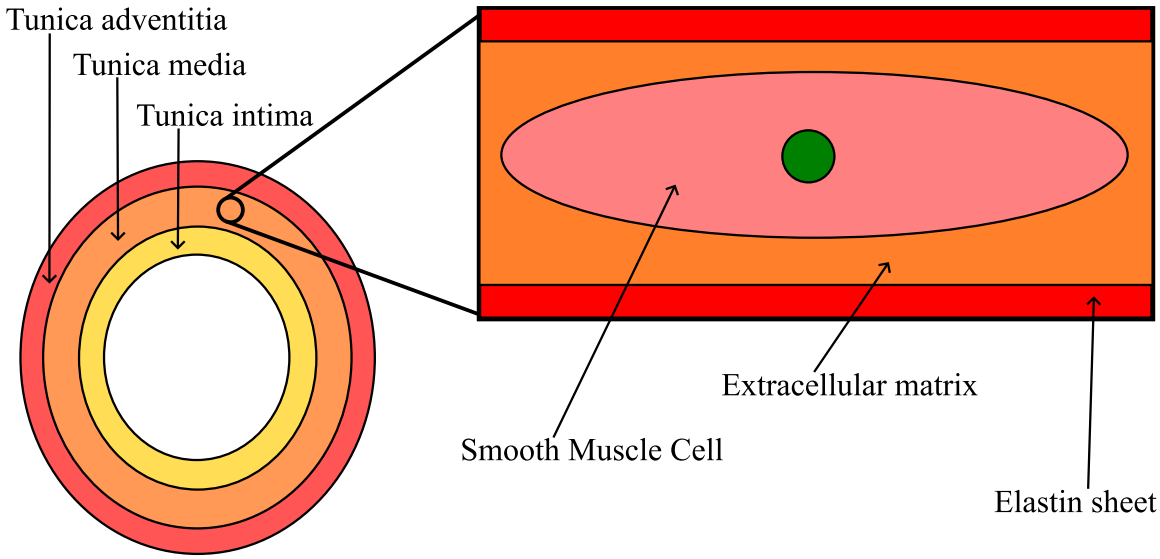


Figure 2.5: This schematic represents the tunica media's microstructure. Here the media is made up of circumferential layers of spindle shaped smooth muscle cells. Each smooth muscle cell is surrounded by an extracellular matrix made up of collagen fibers and other fibrous tissue. Each circumferential layer is then bounded by sheets of elastin. All these constitutive components provide important physical properties to the aorta at the macroscale.

matrix setup with a number of constitutive phases, like that in Figure 2.4.

By approximating the extracellular matrix of the tunica media as a host matrix Ω_A and approximating the smooth muscle cells and sheets of elastin as inclusions Ω_B. Here Ω_B is the union of the subdomains Ω_β where for each $\beta = 1, ..., N$ represents a different inclusion. With this setup we can apply our composite damage phase field model (2.62) as an approximation of aortic dissections at a microscale. This microscale is defined by the length scale of a smooth muscle cell. With this setup we can begin to create a multiscale model.

It should be noted that, as mentioned, around 70% of the artery is water. This obviously motivates a model that includes viscoelastic effects. Upscaling viscoelastic models is also possible and has been explored in the literature many times [129, 142]. We have not come across a damage phase field model that includes viscoelastic effects so such a derivation would possibly be a new addition to the literature. In this thesis we will not explore this approach as we are more interested in just finding a general linear multiscale model of material failure. Though we do want to explicitly note that viscoelastic effects would be an interesting research idea. It is also a point of order that must be addressed when creating a more specialised model of aortic dissections in future applications.

To create a multiscale model we will need to assume some form of local periodicity in the media. However, the media does have a repetitive structure of circumferentially layered smooth muscle cells, surrounded by layers of collagen and elastin. We discuss

this assumption in much more detail in §1.3.2. This locally periodic structure was first observed by Glagov in mammalian elastic arteries [22, 137]. In this work a lamellar unit was described as a smooth muscle cell surrounded by an extracellular matrix. Using this idea of periodic lamellar units as a basis for modelling the aortic microstructure we can formulate an approximation of the microstructure as locally periodic. We are motivated to do this because there is a lot of mathematical precedent for multiscale modelling of composite materials, as a locally periodic structure, using asymptotic homogenisation.

With this approximation in mind, in the next chapter we will explore how to upscale our composite model into a simpler macroscopic system which encodes information about the microscale. This will allow us to study how variations in the aortic microstructure effect the outcomes and onset of aortic dissections. It should be noted that many researchers are finding that degenerative changes in the structure of vascular walls is responsible for aortic dissections [12], which our multiscale model will allow us to study swiftly.

Chapter 3

Creating a multiscale model

In this chapter, we aim to create an effective macroscopic model of aortic dissections, accounting for variations in the aortic microstructure. We introduce the theory of asymptotic homogenisation and its key assumptions. Next we set up a constitutive framework for the elastic potential and damage degradation. Then, non-dimensionalisation of our two-phase damage model and an appropriate approximation of the aortic microstructure, are carried out to derive a macroscopic model in a prototypical fashion. During our multiscale analysis, we take special considerations to ensure the irreversibility constraints of our model apply on the macroscale. Our work will result in a macroscopic model that accounts for microscopic variations in material properties for an arbitrary body. This model will be formed by a macroscopic damage evolution equation and a momentum balance equation, completed by appropriate boundary conditions. Information on the effective contributions of the microscopic system will be stored in a set of effective coefficients, informing the macroscopic model. By creating such a model, we can study how variations in the aortic microstructure affect the mechanics and damage evolution at the macroscale.

3.1 Asymptotic homogenisation

Biological tissue is a multiscale system characterised by geometrical complexity, a varying number of constituents and strong interplay between the hierarchical levels. In fact, most real-world physical system are usually multiscale in nature. From a modelling perspective this introduces a very important question: at which scale should one create an effective model? Modelling the aorta solely from a macroscopic perspective and treating it as a single continuum would only be appropriate when embracing a phenomenological approach. Modelling up from a microscopic perspective, however, we can derive a simplified macroscopic system. This allows us to model each constitutive part of the aortic wall and the interplay among them. Unfortunately, this approach, as seen in the previous chapter, results in an extremely large set of coupled partial differential equations which cannot be

solved easily. Numerically it would be computationally expensive and inefficient. We are therefore motivated to create a multiscale model by starting from a microscopic perspective and then upscaling to a macroscopic model. Since we already have an appropriate microscopic model, we just need an appropriate upscaling technique. A technique which provides a computationally feasible macroscopic mathematical model; however, it must crucially encode the role of the aortic microstructure. Most upscaling techniques lead to a macroscale description where information on the role of the microstructure is partially or entirely lost [67].

Asymptotic homogenisation is an analytical technique which is used to study rapidly oscillating fields. It allows us to exploit systems with sharp length scale disparities by considering a power series representation of any relevant fields and as a result we can provide a macroscopic system of partial differential equations. The derived macroscopic models from this technique encode information from the microstructure into a set of effective coefficients [103]. Computation of these coefficients require solving a set of differential problems known as *local cell problems* which relate the micro and macroscale. A downside to this approach is the necessary assumptions one must satisfy to use asymptotic homogenisation and to decouple the microscale and macroscale spatial variations of the fields. To perform the calculations, we require a very large-scale disparity, an assumption of local periodicity and that all fields are sufficiently regular.

As discussed in §1.2.4, we can find the roots of this theory in the work of Sanchez-Palencia [114] and Bensoussan [9]. Since then, this method has become a crucial tool in multiscale modelling and has been formalised in several modern textbooks dealing with mathematical modelling techniques [63, 103]. Popularisation of asymptotic homogenisation has led to it becoming a crucial tool in many areas of applied mathematics. Despite asymptotic homogenisation's requirement for spatial periodicity, it has found many uses in modelling fluid and solid mechanics. The technique has been extremely relevant in a bio-mechanical context where work has included modelling of drug advection through malignant tumours [102] and a new derivation of Darcy's Law [115]. Modelling of tissue as a hierarchical structure has also found plenty of uses for multiscale modelling techniques [27]. Here, the motivation is simplifying complex elastic structures into a single continuum model informed by some complex microstructure. In terms of biomechanics, researchers have used this approach to study such phenomena as layered hard tissues [109] and myocardium infractions [92].

In our case we will be dealing with a two-phase composite material and upscaling this into a single macroscopic continuum model. Composite materials have been studied a lot for various practical engineering purposes [75, 19]. The use of asymptotic homogenisation to approach modelling composites is not a new idea and has been championed in recent years for a variety of problems. This gives us a strong basis to approach our problem

from. However, none of these works are related to damage mechanics by considering the coupling between elastic deformations and damage evolution. None of them also deal with considerations of irreversibility constraints to a physical process. So, the following multiscale formulation is novel in the multiscale modelling panorama and constitutes the main theoretical result of this thesis.

3.1.1 Basic assumptions

In §1.2.4 we briefly mentioned the basic assumptions necessary to perform upscaling with asymptotic homogenisation. Here we will introduce them more formally. In order to perform upscaling using asymptotic homogenisation there is a set of necessary assumptions we need to satisfy. Let us first consider the two fields we are interested in: the damage phase field $\alpha(\tilde{\mathbf{X}}, t)$ and the displacement field $\mathbf{u}(\tilde{\mathbf{X}}, t)$. Here we are saying that both fields have a dependency on time t and a single spatial vector $\tilde{\mathbf{X}} \subset \mathbb{R}^n$, for $n = 1, 2, 3$ in general. From a macroscopic perspective this would make sense that we have a single smooth continuum, that can easily be described from a single spatial variable. Except that we are modelling a composite material which has many microscopic inclusions and upon zooming in we can see on a microscopic scale that the material is actually much more complex, this is illustrated in figure 3.1. This motivates the idea to decouple the microscopic and macroscopic spatial variables.

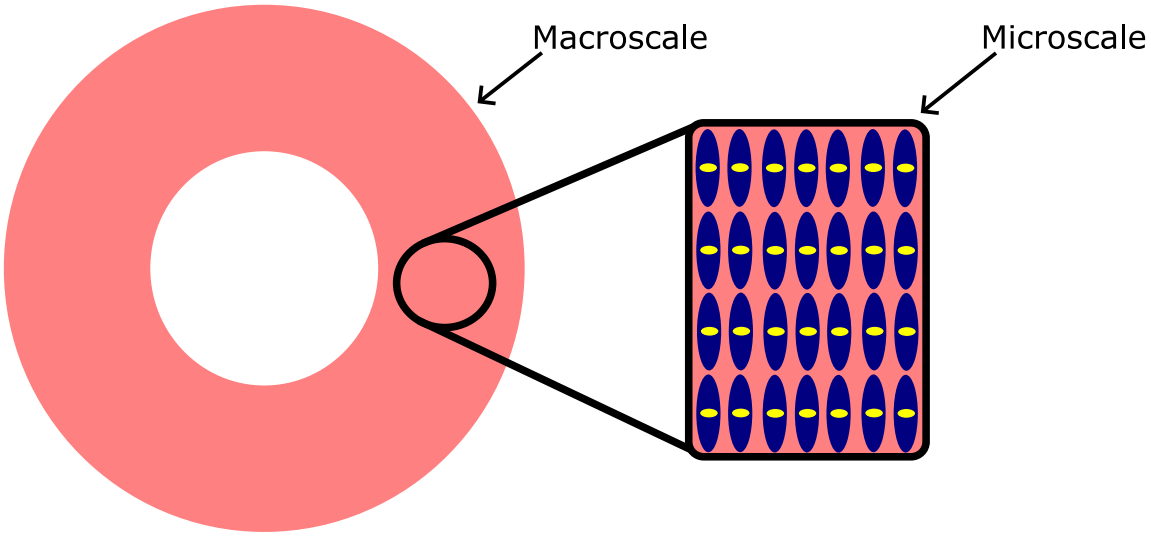


Figure 3.1: A 2D schematic representing both the macro and micro scales of a continuum. On the left-hand side, we see the macroscale domain, where from such a large length scale perspective, the microscopic structure is smoothed out. On the right-hand side, a zoom in of, a periodic composite structure representing the microscale is shown, and the difference between a host matrix and the inclusions is clear. The difference from each perspective, of the left and right hand image, indicates how large scale perspectives can simplify a clearly multiscale system.

One can identify two distinct length scales here the microscale δ and the macroscale L. The former can describe the length scale of small individual inclusions within a host matrix, or some other related distinctive feature at the microscale. The latter is related to the length scale of some large macroscopic dimension. With these length scales we can non-dimensionalise the spatial coordinate $\tilde{\mathbf{X}}$ in two ways:

$$\tilde{\mathbf{X}} = L\mathbf{X} = \delta\mathbf{Y}.$$

Here $\mathbf{X}$ represents a non-dimensional macroscale spatial variable, whereas $\mathbf{Y}$ represents a non-dimensional microscale spatial variable. This in turn motivates the scale ratio

$$\epsilon = \frac{\delta}{L} \tag{3.1}$$

which relates the two spatial coordinates $\mathbf{X}$ and $\mathbf{Y}$. In order to satisfy the theoretical requirements of asymptotic homogenisation we assume ϵ is suitably small, such that $\epsilon \ll 1$.

With this assumption about ϵ we can decouple the spatial variable $\tilde{\mathbf{X}}$. By assuming that for some unknown field $f(\tilde{\mathbf{X}})$ dependent on some variable $\tilde{\mathbf{X}}$ may be decoupled by the variables

$$\tilde{\mathbf{X}} = \mathbf{X} \quad \text{and} \quad \mathbf{Y} = \frac{\mathbf{X}}{\epsilon}. \tag{3.2}$$

It must be clear that what we are assuming is that $\mathbf{X}$ and $\mathbf{Y}$ are completely independent of one another. With this new definition of our spatial variables we can now write that for any unknown field

$$f(\tilde{\mathbf{X}}) = \hat{f}(\mathbf{X}, \mathbf{Y}).$$

Here $\hat{f}$ is a function of both $\mathbf{X}$ and $\mathbf{Y}$. It is easier to think of $\mathbf{X}$ to be the global spatial variable and $\mathbf{Y}$ to be the local spatial variable. With this change of variable the differential operators transform accordingly by the chain rule:

$$\nabla \to \nabla_{\mathbf{X}} + \frac{1}{\epsilon}\nabla_{\mathbf{Y}} \quad \text{and} \quad \Delta \to \nabla_{\mathbf{X}} \cdot \nabla_{\mathbf{X}} + \frac{2}{\epsilon}\nabla_{\mathbf{X}} \cdot \nabla_{\mathbf{Y}} + \frac{1}{\epsilon^2}\nabla_{\mathbf{Y}} \cdot \nabla_{\mathbf{Y}}. \tag{3.3}$$

As discussed previously we intend to make use of a power series expansion of our damage and displacement variables. So, we also make the assumption that all unknown fields can be expanded on multiple spatial scales as a power series of ϵ. In other words for any unknown field f we can define the power series

$$f(\mathbf{X}, \mathbf{Y}, t) = \sum_{j=0}^{\infty} f^{(j)}(\mathbf{X}, \mathbf{Y}, t)\epsilon^j. \tag{3.4}$$

Here we have defined each component, $f^{(j)}$, of the power series as a function of $\mathbf{X}$, $\mathbf{Y}$ and

t. We also assume local boundedness of any function $f^{(j)}$ such that

$$\left| f^{(j)}(\mathbf{X}, \mathbf{Y}, t) \right| \not\to \infty$$

for all $\mathbf{Y}$ and $\mathbf{X} \in \mathbb{R}^n$ for $n = 1, 2, 3$ in general. To justify this assumption one should also note that any real world system is not well behaved if it diverges to infinity. Since we are modelling displacement and damage this seems to be a very reasonable assumption. We could also say that there exists local upper and lower bounds for each field such that

$$f_m(\mathbf{X}) \leq f(\mathbf{X}, \mathbf{Y}) \leq f_M(\mathbf{X}).$$

Here f_m and f_M are the respective lower and upper bounds for the arbitrary function f.

The last important assumption that we must make is local periodicity of the unknown fields. We can justify this assumption by accounting for the fact that the aortic microstructure is composed of locally repeating structures. Mathematically we need this assumption otherwise we would not be able to use asymptotic homogenisation to upscale models in $\mathbb{R}^n$. We can still have fields with globally varying properties but generally we are assuming a periodicity in the spatial variable $\mathbf{Y}$. More rigorously one would say that there exists a family of vectors

$$\mathbf{V}(a_1, ..., a_n) = \sum_{i=1}^{n} a_i \, \mathbf{E}_i, \qquad \text{with} \qquad a_i \in \mathbb{Z}$$

and fixed vectors $\mathbf{E}_i \in \mathbb{R}^n$ constituting a basis of $\mathbb{R}^n$. Thereby every unknown field $f(\mathbf{X}, \mathbf{Y})$ is locally periodic such that

$$f(\mathbf{X}, \mathbf{Y}) = f(\mathbf{X}, \mathbf{Y} + \mathbf{V}(a_1, ..., a_n)) \quad \forall \, a_i \in \mathbb{Z}. \tag{3.5}$$

Note that this assumption of local periodicity is stated for arbitrarily shaped periodic cells. Thus, microscopic variations of each field can now be studied on a single local periodic cell. For a more complete overview of these assumptions and there necessity in asymptotic homogenisation we refer you to the textbooks by Penta et al. [103] and by Holmes [63].

3.1.2 Homogenisation process

With the above assumptions one can follow a simple process to upscale a microscopic model to a macroscopic model. Here we will layout the simple steps that are usually taken when applying asymptotic homogenisation. This will familiarise the reader with this technique and also make it clear where our derivation of a multiscale model is clearly deviating from the norm. Here we are assuming that usually a series of constitutive choices are made with the design of reducing the model to a linear model when scale separations are assumed. This is a necessary requirement to find a macroscopic model independent of

the microscopic variables. This is further discussed in §1.2.4.

The goal of asymptotic homogenisation is to produce a multiscale model over two scales: the microscale and macroscale. This is done by first identifying the scales across which the variation occurs and defining two (or more) coordinate systems to describe different geometries. If the non-dimensional ratio between the scales, ϵ, is sufficiently small then the two scales are well-separated and the two coordinate systems can be assumed as independent of one another. Next one can make a series of assumptions about the boundedness and periodicity of the variables of interest as explained above. All unknown variables are then expressed as power series expansions in ϵ. Therefore, systems of equations may then be separated by equating the coefficients of powers of ϵ. Asymptotic homogenisation captures the behaviour observed over multiple scales by then separating these coefficients of ϵ and broadly following these next steps:

1. Equate the coefficients of ϵ^0 to construct simplified differential equations. Solving these equations one can establish which macroscopic variables are independent of the microscale.

2. Next by equating coefficients of ϵ^1 and applying the previous results one can construct a system of differential equations that relate the micro and macroscales. These equations cannot be directly solved. Therefore one usually exploits the linearity of these differential equations. As a result one can construct a series of ansatz solutions to this system. These ansätze solutions are constructed by introducing a number of auxiliary variables. These auxiliary variables are the solutions to a series of linear differential equations, which we call local cell problems, derived from our ϵ^1 system of equations. These cell problems inform the macroscopic variables to the effects of the underlying microstructure.

3. The last step is for one to construct a system of differential equations by equating coefficients of ϵ^2 and taking a local integral average over a local periodic cell Ω. Then by making use of all the previous results and local periodicity one can simplify the system into a series of governing macroscopic equations. These macroscopic equations are defined by a series of effective macroscopic coefficients. Each coefficient is a local integral average that incorporates the solutions to our local cell problems. Thereby, encoding the properties of microstructure at the macroscale.

This method results in a homogenised problem which describes the leading order terms of each field. Our macroscopic system is separate from our cell problems and both problems are solved separately. As a result by solving different variations of the cell problems we can study different variations of the underlying microstructure. This could allow us to study a great variety of physical phenomena relevant to aortic dissections, as discussed in §1.3.2. Broadly speaking this protocol is standard for most problems involving asymptotic

homogenisation. In our case we will see further analysis performed in order to ensure that damage growth is irreversible on every scale of our homogenised problem. For a more complete study of the procedures of asymptotic homogenisation I would recommend a number of textbooks [7, 21, 63, 103].

3.2 Multiscale formulation

Let us consider the case of an arbitrary composite material with a host matrix and set of disjoint inclusions. This geometry is chosen for simplicity but is also applicable for a wide variety of arbitrary sub-phases. We setup a linear elastic damage problem as defined from the system of equation (2.62) with the interface and boundary conditions (2.63). This system describes a balance of momentum and damage evolution equation across a locally periodic bounded continuum Ω. Here $\Omega \in \mathbb{R}^n$ for $n \geq 2$, such that $\bar{\Omega} = \bar{\Omega}_A \cup \bar{\Omega}_B$ and $\overset{\circ}{\Omega}_A \cap \overset{\circ}{\Omega}_B = \emptyset$. Here, Ω_A represents a host matrix and

$$\Omega_B = \bigcup_{\beta=1}^{N} \Omega_\beta$$

represents a number N of disjoint inclusions Ω_β. For completion we define our boundaries and interfaces of Ω as follows:

$$\Gamma_\beta = \partial\Omega_A \cap \partial\Omega_\beta \quad \text{and} \quad \partial\Omega = (\partial\Omega_A \cup \partial\Omega_B) \setminus \left(\bigcup_{\beta=1}^{N} \Gamma_\beta \right).$$

To avoid any confusion, let us also state that all the inclusions included in Ω_B never intersect or even touch. We write this as

$$\Omega_i \bigcap_{i \neq j} \Omega_j = \emptyset \quad \forall \Omega_i, \Omega_j \subseteq \Omega_B.$$

An illustration of this setup is given in Figure 3.2 for an arbitrary microstructure defined over a microscale defined by some length scale δ. In this section we will be assuming that all the basic and necessary assumptions required are met to perform asymptotic homogenisation.

In the following formulation, we will equate coefficients of ϵ^k for $k = 0, 1, 2, ...$ for our linear elastic and damage problem. Doing so allows us to create a homogenised macroscopic model of our system Ω at the zeroth order. To smooth out this problem and remove any dependencies on a local microscopic spatial variable $\mathbf{Y}$ we perform local integral averages. We integrate with respect to $\mathbf{Y}$ over a local cell Ω. To this end it is important to define the following operators for local integral averages of any field $f(\mathbf{X}, \mathbf{Y})$ over a local cell Ω,

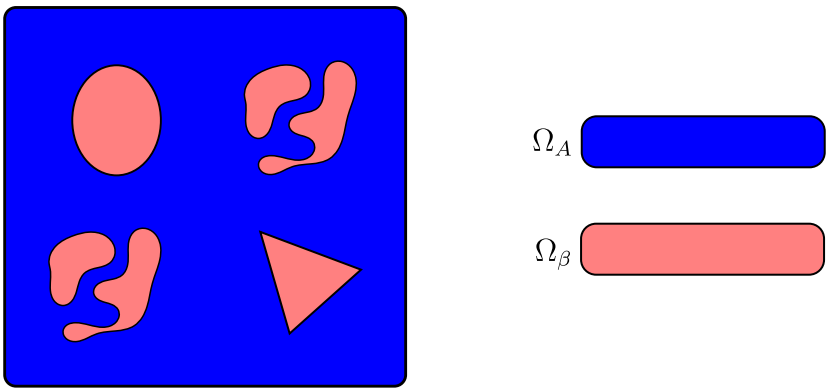


Figure 3.2: A simple 2D schematic depicting our composite microscale. This is a simple periodic unit Ω defined across the microscale, δ, with a number of arbitrary local inclusions. It should be noted that the host matrix Ω_A and the inclusions Ω_β are only restricted in our formulation by local periodicity. In other words one is free to choose a wide variety of arbitrary setups for a local microstructure.

we define the operators:

$$\begin{aligned} \{f\}_\Omega &= \frac{1}{|\Omega|}\int_\Omega f(\mathbf{X},\mathbf{Y})\,dV_\mathbf{Y}; \\ \{f\}_{\Omega_A} &= \frac{1}{|\Omega|}\int_{\Omega_A} f(\mathbf{X},\mathbf{Y})\,dV_\mathbf{Y}; \\ \{f\}_{\Omega_\beta} &= \frac{1}{|\Omega|}\int_{\Omega_\beta} f(\mathbf{X},\mathbf{Y})\,dV_\mathbf{Y}, \end{aligned} \tag{3.6}$$

where $|\Omega|$ represents the volume of a periodic cell. Throughout the homogenisation process we will operate from the perspective of a single periodic cell. Geometric variations will be smoothed out by the process of a local integral averaging.

3.2.1 Constitutive framework

Before we write down a system of microscopic equations that we would like to upscale it is important that we make a set of constitutive choices. We are working in the frame of elasticity but to ensure that we can upscale our microscopic model we will assume a linearisation of our model. To be as applicable as possible, however, we will setup a system with some residual stress and strain. This could represent a linearisation around some pre-stretched material. In our damage phase field model stress and elastic potential are related so both will be affected by whatever linear elastic energy ϕ we introduce. Taking all of this into account we introduce ϕ as combination of Hooke's law for elastic potential and the potential from some residual stress as

$$\phi = \frac{1}{2}\xi(\mathbf{u}) : \mathbb{C} : \xi(\mathbf{u}) + \sigma(\mathbf{X}) : \xi(\mathbf{u}). \tag{3.7}$$

Here, we have introduced the second rank tensor σ which is a constant tensor field accounting for any residual stress in our system. Here we introduce $\mathbb{C}$ as the elastic modulus, a fourth rank tensor that contains both minor and major symmetries [103, 104]. We can express major symmetries of a fourth rank tensor $\mathbb{C}$ as

$$\mathbb{C}_{ijkl} = \mathbb{C}_{ijkl}$$

and similarly we have the left and right minor symmetries:

$$\mathbb{C}_{ijkl} = \mathbb{C}_{jikl} \quad \text{and} \quad \mathbb{C}_{ijkl} = \mathbb{C}_{ijlk}.$$

We have also introduced the strain tensor, a second rank tensor, defined as

$$\xi(\mathbf{u}) = \frac{\nabla\mathbf{u} + (\nabla\mathbf{u})^\intercal}{2}.$$

It is important to note that this is a symmetric tensor such that, $\xi = \xi^\intercal$.

The other important constitutive choice we must make is the form of the damage degradation function. We require a form of the degradation function that satisfies all of the constraints described by (2.26), laid out in §2.2.2. Here we will introduce a polynomial form of the degradation function. Our intention is to choose a form of the degradation function that satisfies the necessary constraints of the degradation function and that will also upscale with ease, whilst covering a very wide range of possible degradation functions. We can write our degradation function as

$$g(\alpha) = (1-\alpha)^p \quad \text{for } p \geq 2 \text{ with } p \in \mathbb{N}. \tag{3.8}$$

Note we can do this for a general exponent p that will regulate the material softening. The stiffness of $\mathbb{C}$ will degrade faster for larger values of p and slower for smaller values of p. One could introduce a more exact form of the degradation function but here in this derivation we are trying to be as arbitrary as possible to maximise the applicability of our multiscale model.

Taking the constitutive choices for elastic potential (3.7) and the degradation function (3.8), then substituting them into our damage model (2.62) and the interface condition (2.63) giving us a new damage phase field model. Here we label every field and parameter with the appropriate subscript: f_A for the host matrix and f_β for each inclusion. We write

our microscopic system as

$$
\begin{aligned}
&\rho_A \mathbf{b} + \nabla \cdot \left[(1-\alpha_A)^{p_A}\left(\mathbb{C}_A : \xi(\mathbf{u}_A) + \sigma_A\right)\right] = \rho_A \ddot{\mathbf{u}}_A \quad \text{in } \Omega_A, \\
&\rho_\beta \mathbf{b} + \nabla \cdot \left[(1-\alpha_\beta)^{p_\beta}\left(\mathbb{C}_\beta : \xi(\mathbf{u}_\beta) + \sigma_\beta\right)\right] = \rho_\beta \ddot{\mathbf{u}}_\beta \quad \text{in } \Omega_\beta \, \forall \beta, \\
&\eta_A \dot{\alpha}_A = \left\langle p_A (1-\alpha_A)^{p_A - 1}\left(\frac{1}{2}\xi(\mathbf{u}_A) : \mathbb{C}_A : \xi(\mathbf{u}_A) + \sigma_A : \xi(\mathbf{u}_A) - \psi_A\right) + \right. \\
&\qquad \left. - \frac{G_A}{\ell}\alpha_A + \nabla \cdot \left(\frac{G_A}{\ell}\mathbb{L}_A \nabla \alpha_A\right)\right\rangle_+ \text{in } \Omega_A, \\
&\eta_\beta \dot{\alpha}_\beta = \left\langle p_\beta (1-\alpha_\beta)^{p_\beta - 1}\left(\frac{1}{2}\xi(\mathbf{u}_\beta) : \mathbb{C}_\beta : \xi_A(\mathbf{u}_\beta) + \sigma_\beta : \xi(\mathbf{u}_\beta) - \psi_\beta\right) + \right. \\
&\qquad \left. - \frac{G_\beta}{\ell}\alpha_\beta + \nabla \cdot \left(\frac{G_\beta}{\ell}\mathbb{L}_\beta \nabla \alpha_\beta\right)\right\rangle_+ \text{in } \Omega_\beta \, \forall \beta.
\end{aligned}
\tag{3.9}
$$

Here the first two equations describe the balance of linear momentum in our system for linear elastic materials with residual stress. Similarly the other two equations are the damage evolution equations for each subdomain. Here the damage evolution is described by as the elastic potential from Hooke's law with contributions from the residual stress. Now as for our interface conditions they read as

$$
\begin{aligned}
&(1-\alpha_A)^{p_A}\left(\mathbb{C}_A : \xi(\mathbf{u}_A) + \sigma_A\right) \cdot \mathbf{n}_A = (1-\alpha_\beta)^{p_\beta}\left(\mathbb{C}_\beta : \xi(\mathbf{u}_\beta) + \sigma_\beta\right) \cdot \mathbf{n}_A \ \text{ on } \Gamma_\beta \, \forall \beta, \\
&\frac{G_A}{\ell}\mathbb{L}_A \nabla \alpha_A \cdot \mathbf{n}_A = \frac{G_\beta}{\ell}\mathbb{L}_\beta \nabla \alpha_\beta \cdot \mathbf{n}_A \ \text{ on } \Gamma_\beta \, \forall \beta.
\end{aligned}
\tag{3.10}
$$

Our interface conditions describe the continuity of stress and damage flux, respectively. For our multiscale formulation we do not need to consider boundary conditions. The reason for this is because when we upscale our model we are constructing a set of governing equations by taking a local integral average over some representative periodic cell Ω. This local periodic cell exists in the domains interior and does not require microscopic boundary conditions for this upscaling. Instead when we have a macroscopic model we can simply infer some macroscopic boundary conditions.

For every parameter in the system we are assuming that they are continuous and smooth in $\mathbf{X}$ within the domains Ω_A and Ω_B. However, over the interfaces Γ_β we must note that material properties are not necessarily continuous and in, general, considered to be discontinuous. Mathematically this can be described by the statement that for every field $f_A(\mathbf{X})$ and $f_\beta(\mathbf{X})$ one has that

$$
f_A(\mathbf{X}) - f_\beta(\mathbf{X}) \neq 0 \text{ on } \Gamma_\beta \, \forall \, \beta.
$$

This principle is true for all the above parametric fields described in the system of equations (3.9).

3.2.2 Non-dimensionalisation

Equations (3.9) and the interface conditions (3.10) equipped with appropriate boundary conditions, represent a complete set of coupled partial differential equations. The next step in our multiscale formulation is to non-dimensionalise our system, thereby allowing us to analyse inherent length or time scales in our system. We now rewrite our system in terms of the non-dimensional variables:

$$\mathbf{X} = L\mathbf{X}', \quad \mathbf{u} = L\mathbf{u}', \quad \mathbb{C} = CL\mathbb{C}', \quad \mathbb{L} = \ell^2\mathbb{L}',$$
$$\mathbf{b} = \frac{C}{\rho_r}\mathbf{b}' \quad \text{and} \quad t = \sqrt{\frac{\rho_r L}{C}}t'.$$

The length scales L and ℓ, represent the macroscopic length scale and the diffusive length scale, respectively. Here we introduce C and ρ_r as a pressure gradient and reference density, respectively. The corresponding governing equations after this non-dimensionalisation (dropping the primes for simplicity) is

$$\begin{aligned}
&\hat{\rho}_A\mathbf{b} + \nabla\cdot\left[(1-\alpha_A)^{p_A}\left(\mathbb{C}_A : \xi(\mathbf{u}_A) + \hat{\sigma}_A\right)\right] = \hat{\rho}_A\ddot{\mathbf{u}}_A \quad \text{in } \Omega_A,\\
&\hat{\rho}_\beta\mathbf{b} + \nabla\cdot\left[(1-\alpha_\beta)^{p_\beta}\left(\mathbb{C}_\beta : \xi(\mathbf{u}_\beta) + \hat{\sigma}_\beta\right)\right] = \hat{\rho}_\beta\ddot{\mathbf{u}}_\beta \quad \text{in } \Omega_\beta\,\forall\beta,\\
&\hat{\eta}_A\dot{\alpha}_A = \Bigg\langle p_A(1-\alpha_A)^{p_A-1}\left(\frac{1}{2}\xi(\mathbf{u}_A) : \mathbb{C}_A : \xi(\mathbf{u}_A) + \hat{\sigma}_A : \xi(\mathbf{u}_A) - \hat{\psi}_A\right) + ...\\
&\qquad - \hat{G}_A\alpha_A + \nabla\cdot\left(\mathbb{D}_A\nabla\alpha_A\right)\Bigg\rangle_+ \quad \text{in } \Omega_A,\\
&\hat{\eta}_\beta\dot{\alpha}_\beta = \Bigg\langle p_\beta(1-\alpha_\beta)^{p_\beta-1}\left(\frac{1}{2}\xi(\mathbf{u}_\beta) : \mathbb{C}_\beta : \xi_A(\mathbf{u}_\beta) + \hat{\sigma}_\beta : \xi(\mathbf{u}_\beta) - \hat{\psi}_\beta\right) + ...\\
&\qquad - \hat{G}_\beta\alpha_\beta + \nabla\cdot\left(\mathbb{D}_\beta\nabla\alpha_\beta\right)\Bigg\rangle_+ \quad \text{in } \Omega_\beta\,\forall\beta.
\end{aligned} \tag{3.11}$$

Our non-dimensional system is supplemented by the interface conditions

$$\begin{gathered}
(1-\alpha_A)^{p_A}\left(\mathbb{C}_A : \xi(\mathbf{u}_A) + \hat{\sigma}_A\right)\cdot\mathbf{n}_A = (1-\alpha_\beta)^{p_\beta}\left(\mathbb{C}_\beta : \xi(\mathbf{u}_\beta) + \hat{\sigma}_\beta\right)\cdot\mathbf{n}_A \text{ on } \Gamma_\beta\,\forall\beta,\\
\mathbb{D}_A\nabla\alpha_A\cdot\mathbf{n}_A = \mathbb{D}_\beta\nabla\alpha_\beta\cdot\mathbf{n}_A \text{ on } \Gamma_\beta\,\forall\beta.
\end{gathered} \tag{3.12}$$

We have defined a number of new non-dimensional parameters for this non-dimensionalised system, each indicated by a hat. Of course every function and field in the above system has the appropriate subscript indicating which subdomain it exists in. The non-dimensional

functions introduced above are defined as:

$$\hat{\rho}(\mathbf{X}) = \frac{\rho(\mathbf{X})}{\rho_r}, \quad \hat{\sigma}(\mathbf{X}) = \frac{\sigma(\mathbf{X})}{CL}, \quad \hat{\eta}(\mathbf{X}) = \sqrt{\frac{\eta^2(\mathbf{X})}{CL^3\rho_r}} \tag{3.13}$$
$$\hat{\psi}(\mathbf{X}) = \frac{\psi(\mathbf{X})}{CL}, \quad \hat{G}(\mathbf{X}) = \frac{G(\mathbf{X})}{CL\ell} \text{ and } \mathbb{D}(\mathbf{X}) = \frac{G(\mathbf{X})\ell}{CL^3}\mathbb{L}(\mathbf{X}).$$

Here $\hat{\rho}_r$ represents the relative material density, whereas $\hat{\sigma}$ and $\hat{\eta}$ are the non-dimensional residual stress and viscosity, respectively. The scalar functions $\hat{\psi}$ and $\hat{G}$ represent the non-dimensional damage threshold and tearing energy, whereas the second order tensor $\mathbb{D}$ is defined as the non-dimensional damage diffusion.

Remark. *When performing asymptotic homogenisation paying special attention to the scaling of all parameters is extremely important. The key assumption of asymptotic homogenisation is that every parameter under consideration is asymptotically larger than* $\mathcal{O}(\epsilon)$*. If this assumption is not satisfied then we cannot perform a power series expansion on the fields of interest about* ϵ*. However, this is not clear for all the parameters defined by* (3.13)*. As* ℓ *is required to be extremely small then it is not clear whether* $\hat{G}$ *and* $\mathbb{D}$ *are reasonably scaled for asymptotic homogenisation. First consider* $\hat{G}$*, which could be much larger than any other terms unless the denominator in this expression is balanced by the numerator. Since* G *is the energy per unit of damage surface area then we can make several obvious assumptions about its scale. First the scale of energy should be similar to the scale of the elastic potential energy density stretching the material, which as we know is* CL*. As this is an energy per unit volume we must multiply it be a characteristic length scale of a tear surface area which is defined by* ℓ*. Therefore, it is clear that* $G \sim CL\ell$ *and thus from* (3.13) *we can safely say that* $\hat{G}$ *is of a regular scale and suitable for upscaling.*

Next we are concerned with the scale of $\mathbb{D}$ *which as defined by* (3.13) *could be very small. Again this is because we require that* $\ell \ll 1$ *ensuring that damage diffusion is narrow, characterising a sharp tear in a material. However, for asymptotic homogenisation to work we require that* $\epsilon \ll 1$ *is the smallest parameter in our system. Using the scaling that* $G \sim CL\ell$ *and substituting into* (3.13) *would give us the non-dimensional second-order tensor*

$$\mathbb{D}(\mathbf{X}) = \frac{\ell^2}{L^2}\mathbb{L}(\mathbf{X}).$$

If we define $\tau = \ell/L$ *then we can say that* $\mathbb{D} \sim \tau^2$*. As* $\ell \ll L$ *then it is obvious that* $\tau \ll 1$*. This could create a major issue when upscaling with asymptotic homogenisation if* $\tau^2 < \epsilon$*. In this section we are assuming for the sake of upscaling that* ϵ *is by far the smallest parameter and hence* $\epsilon \ll \tau^2$*. We could write this in terms of our length scales as* $\delta \ll \ell^2$*. In Chapter 4 we will explore this relationship and show that in order for our macroscopic model to indeed converge with our macroscopic model that we do in fact require* $\epsilon \ll \tau^2$*.*

It would be an interesting further research question to explore if $\tau \leq \epsilon$*. Doing so would*

require one to explore a three scaled homogenisation [109]. In such a case one may choose to introduce a macroscale L, a mesoscale δ and a microscale ℓ. For the purpose of this thesis we will take all the coefficients to be asymptotically lager than $\mathcal{O}(\epsilon)$. It is important though to note that by exploring the extremes of scaling one could study a vastly different model.

3.2.3 The homogenised problem

In this section we will apply asymptotic homogenisation to the non-dimensional model described by (3.11) and (3.12). Since $\epsilon \ll 1$, we enforce the large length scale difference between the macroscale of Ω and the microscale of the disjoint inclusions Ω_B. From now on we are assuming that all fields and material properties depend on the independent spatial variables $\mathbf{X}$ and $\mathbf{Y}$. Therefore we now have the functions:

$$\begin{aligned}
&\mathbf{u}_A = \mathbf{u}_A(\mathbf{X},\mathbf{Y}), \quad \mathbf{u}_\beta = \mathbf{u}_\beta(\mathbf{X},\mathbf{Y}), \quad \alpha_A = \alpha_A(\mathbf{X},\mathbf{Y}), \quad \alpha_\beta = \alpha_\beta(\mathbf{X},\mathbf{Y}),\\
&\mathbb{C}_A = \hat{\mathbb{C}}_A(\mathbf{X},\mathbf{Y}), \quad \mathbb{C}_\beta = \mathbb{C}_\beta(\mathbf{X},\mathbf{Y}), \quad \mathbb{D}_A = \mathbb{D}_A(\mathbf{X},\mathbf{Y}), \quad \mathbb{D}_\beta = \mathbb{D}_\beta(\mathbf{X},\mathbf{Y}),\\
&\hat{\rho}_A = \hat{\rho}_A(\mathbf{X},\mathbf{Y}), \quad \hat{\rho}_\beta = \hat{\rho}_\beta(\mathbf{X},\mathbf{Y}), \quad \hat{\sigma}_A = \hat{\sigma}_A(\mathbf{X},\mathbf{Y}), \quad \hat{\sigma}_\beta = \hat{\sigma}_\beta(\mathbf{X},\mathbf{Y}),\\
&\hat{\eta}_A = \hat{\eta}_A(\mathbf{X},\mathbf{Y}), \quad \hat{\eta}_\beta = \hat{\eta}_\beta(\mathbf{X},\mathbf{Y}), \quad \hat{G}_A = \hat{G}_A(\mathbf{X},\mathbf{Y}), \quad \hat{G}_\beta = \hat{G}_\beta(\mathbf{X},\mathbf{Y}),\\
&\hat{\psi}_A = \hat{\psi}_A(\mathbf{X},\mathbf{Y}), \quad \hat{\psi}_\beta = \hat{\psi}_\beta(\mathbf{X},\mathbf{Y}) \quad \mathbf{b}_A = \mathbf{b}_A(\mathbf{X},\mathbf{Y}) \text{ and } \mathbf{b}_\beta = \mathbf{b}_\beta(\mathbf{X},\mathbf{Y}),
\end{aligned}$$

which is true for all β. Due to this change of variable the derivatives are transformed according to (3.3). Therefore the balance of momentum equations in (3.11) become

$$\begin{aligned}
&\epsilon^2\nabla_{\mathbf{X}}\cdot[(1-\alpha_A)^{p_A}\mathbb{C}_A:\xi_{\mathbf{X}}(\mathbf{u}_A)] + \epsilon\nabla_{\mathbf{X}}\cdot[(1-\alpha_A)^{p_A}\mathbb{C}_A:\xi_{\mathbf{Y}}(\mathbf{u}_A)]\\
&+\epsilon\nabla_{\mathbf{Y}}\cdot[(1-\alpha_A)^{p_A}\mathbb{C}_A:\xi_{\mathbf{X}}(\mathbf{u}_A)] + \nabla_{\mathbf{Y}}\cdot[(1-\alpha_A)^{p_A}\mathbb{C}_A:\xi_{\mathbf{Y}}(\mathbf{u}_A)]\\
&+\epsilon^2\nabla_{\mathbf{X}}\cdot[(1-\alpha_A)^{p_A}\hat{\sigma}_A] + \epsilon\nabla_{\mathbf{Y}}\cdot[(1-\alpha_A)^{p_A}\hat{\sigma}_A] + \epsilon^2\hat{\rho}_A\mathbf{b}_A = \epsilon^2\hat{\rho}_A\ddot{\mathbf{u}}_A \quad \text{in } \Omega_A,
\end{aligned} \tag{3.14}$$

$$\begin{aligned}
&\epsilon^2\nabla_{\mathbf{X}}\cdot[(1-\alpha_\beta)^{p_\beta}\mathbb{C}_\beta:\xi_{\mathbf{X}}(\mathbf{u}_\beta)] + \epsilon\nabla_{\mathbf{X}}\cdot[(1-\alpha_\beta)^{p_\beta}\mathbb{C}_A:\xi_{\mathbf{Y}}(\mathbf{u}_\beta)]\\
&+\epsilon\nabla_{\mathbf{Y}}\cdot[(1-\alpha_\beta)^{p_\beta}\mathbb{C}_\beta:\xi_{\mathbf{X}}(\mathbf{u}_\beta)] + \nabla_{\mathbf{Y}}\cdot[(1-\alpha_\beta)^{p_\beta}\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\mathbf{u}_\beta)]\\
&+\epsilon^2\nabla_{\mathbf{X}}\cdot[(1-\alpha_\beta)^{p_\beta}\hat{\sigma}_\beta] + \epsilon\nabla_{\mathbf{Y}}\cdot[(1-\alpha_\beta)^{p_\beta}\hat{\sigma}_\beta] + \epsilon^2\hat{\rho}_\beta\mathbf{b}_\beta = \epsilon^2\hat{\rho}_\beta\ddot{\mathbf{u}}_\beta \quad \text{in } \Omega_\beta\,\forall\beta,
\end{aligned} \tag{3.15}$$

and are equipped with the interface conditions

$$\begin{aligned}
&(1-\alpha_A)^{p_A}\left(\epsilon\mathbb{C}_A:\xi_{\mathbf{X}}(\mathbf{u}_A) + \mathbb{C}_A:\xi_{\mathbf{Y}}(\mathbf{u}_A) + \epsilon\hat{\sigma}_A\right)\cdot\mathbf{n}_A\\
&= (1-\alpha_\beta)^{p_\beta}\left(\epsilon\mathbb{C}_\beta:\xi_{\mathbf{X}}(\mathbf{u}_\beta) + \mathbb{C}_\beta:\xi_{\mathbf{Y}}(\mathbf{u}_\beta) + \epsilon\hat{\sigma}_\beta\right)\cdot\mathbf{n}_A \text{ on } \Gamma_\beta\,\forall\beta.
\end{aligned} \tag{3.16}$$

With this change of variable we have the corresponding damage evolution equations

$$\begin{aligned}\epsilon^2\hat{\eta}_A\dot{\alpha}_A = \Big\langle \frac{p_A}{2}(1-\alpha_A)^{p_A-1}\Big(&\epsilon^2\xi_{\mathbf{X}}(\mathbf{u}_A):\mathbb{C}_A:\xi_{\mathbf{X}}(\mathbf{u}_A)+\epsilon\xi_{\mathbf{X}}(\mathbf{u}_A):\mathbb{C}_A:\xi_{\mathbf{Y}}(\mathbf{u}_A)\\&+\epsilon\xi_{\mathbf{Y}}(\mathbf{u}_A):\mathbb{C}_A:\xi_{\mathbf{X}}(\mathbf{u}_A)+\xi_{\mathbf{Y}}(\mathbf{u}_A):\mathbb{C}_A:\xi_{\mathbf{Y}}(\mathbf{u}_A)+\epsilon^2\hat{\sigma}_A:\xi_{\mathbf{X}}(\mathbf{u}_A)\\&+\epsilon\hat{\sigma}_A:\xi_{\mathbf{Y}}(\mathbf{u}_A)-\epsilon^2\hat{\psi}_A\Big)-\epsilon^2\hat{G}_A\alpha_A+\epsilon^2\nabla_{\mathbf{X}}\cdot\Big(\mathbb{D}_A\nabla_{\mathbf{X}}\alpha_A\Big)\\&+\epsilon\nabla_{\mathbf{X}}\cdot\Big(\mathbb{D}_A\nabla_{\mathbf{Y}}\alpha_A\Big)+\epsilon\nabla_{\mathbf{Y}}\cdot\Big(\mathbb{D}_A\nabla_{\mathbf{X}}\alpha_A\Big)+\nabla_{\mathbf{Y}}\cdot\Big(\mathbb{D}_A\nabla_{\mathbf{Y}}\alpha_A\Big)\Big\rangle_+ \text{ in } \Omega_A,\end{aligned} \tag{3.17}$$

$$\begin{aligned}\epsilon^2\hat{\eta}_\beta\dot{\alpha}_\beta = \Big\langle \frac{p_\beta}{2}(1-\alpha_\beta)^{p_\beta-1}\Big(&\epsilon^2\xi_{\mathbf{X}}(\mathbf{u}_\beta):\mathbb{C}_\beta:\xi_{\mathbf{X}}(\mathbf{u}_\beta)+\epsilon\xi_{\mathbf{X}}(\mathbf{u}_\beta):\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\mathbf{u}_\beta)\\&\epsilon\xi_{\mathbf{Y}}(\mathbf{u}_\beta):\mathbb{C}_\beta:\xi_{\mathbf{X}}(\mathbf{u}_\beta)+\xi_{\mathbf{Y}}(\mathbf{u}_\beta):\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\mathbf{u}_\beta)+\epsilon^2\hat{\sigma}_\beta:\xi_{\mathbf{X}}(\mathbf{u}_\beta)\\&+\epsilon\hat{\sigma}_\beta:\xi_{\mathbf{Y}}(\mathbf{u}_\beta)-\epsilon^2\hat{\psi}_\beta\Big)-\epsilon^2\hat{G}_\beta\alpha_\beta+\epsilon^2\nabla_{\mathbf{X}}\cdot\Big(\mathbb{D}_\beta\nabla_{\mathbf{X}}\alpha_\beta\Big)\\&+\epsilon\nabla_{\mathbf{X}}\cdot\Big(\mathbb{D}_\beta\nabla_{\mathbf{Y}}\alpha_\beta\Big)+\epsilon\nabla_{\mathbf{Y}}\cdot\Big(\mathbb{D}_\beta\nabla_{\mathbf{X}}\alpha_\beta\Big)+\nabla_{\mathbf{Y}}\cdot\Big(\mathbb{D}_\beta\nabla_{\mathbf{Y}}\alpha_\beta\Big)\Big\rangle_+ \text{ in } \Omega_\beta\,\forall\beta,\end{aligned} \tag{3.18}$$

which are also equipped with the interface conditions:

$$\epsilon\mathbb{D}_A\nabla_{\mathbf{X}}\alpha_A\cdot\mathbf{n}_A+\mathbb{D}_A\nabla_{\mathbf{Y}}\alpha_A\cdot\mathbf{n}_A=\epsilon\mathbb{D}_\beta\nabla_{\mathbf{X}}\alpha_\beta\cdot\mathbf{n}_A+\mathbb{D}_\beta\nabla_{\mathbf{Y}}\alpha_\beta\cdot\mathbf{n}_A \text{ on } \Gamma_\beta\,\forall\beta. \tag{3.19}$$

The above system is also supplemented by the assumption of local periodicity as expressed in (3.5) and by continuity over the interfaces:

$$\mathbf{u}_A=\mathbf{u}_\beta \quad \text{and} \quad \alpha_A=\alpha_\beta \text{ on } \Gamma_\beta\,\forall\beta. \tag{3.20}$$

To create a macroscopic system of equations we next introduce a pair of power series expansions about ϵ for the damage and displacement. So by recalling (3.4) we write that

$$\alpha=\sum_{j=0}^{\infty}\alpha^{(j)}(\mathbf{X},\mathbf{Y},t)\epsilon^j \quad \text{and} \quad \mathbf{u}=\sum_{j=0}^{\infty}\mathbf{u}^{(j)}(\mathbf{X},\mathbf{Y},t)\epsilon^j, \tag{3.21}$$

which will allows us to perform a multiscale expansion of the relevant fields and to collect coefficients for particular powers of ϵ. A common point of confusion here is considering how α can ever approach the upper limit of 1 if it is a power series expansion about ϵ. By observing (3.21) then it is clear that we must have $\alpha=\alpha^{(0)}$ as $\epsilon\to 0$. Therefore the leading order term, $\alpha^{(0)}$ must be of order $\mathcal{O}(1)$ and as $\alpha\to 1$ then clearly $\alpha^{(0)}\to 1$.

By using our power series (3.21) we can collect coefficients of ϵ^0 for the equations (3.14),

(3.15) and (3.16) by considering the limit $\epsilon \to 0$. This yields

$$\begin{aligned}
&\nabla_{\mathbf{Y}} \cdot \left[(1-\alpha_A^{(0)})^{p_A}\mathbb{C}_A : \xi_{\mathbf{Y}}(\mathbf{u}_A^{(0)})\right] = \mathbf{0} \text{ in } \Omega_A,\\
&\nabla_{\mathbf{Y}} \cdot \left[(1-\alpha_A^{(0)})^{p_\beta}\mathbb{C}_\beta : \xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(0)})\right] = \mathbf{0} \text{ in } \Omega_\beta \,\forall\beta,\\
&(1-\alpha_A^{0})^{p_A}\mathbb{C}_A : \xi_{\mathbf{Y}}(\mathbf{u}_A^{(0)})\mathbf{n}_A = (1-\alpha_\beta^{(0)})^{p_\beta}\mathbb{C}_\beta : \xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(0)})\mathbf{n}_A \text{ on } \Gamma_\beta \,\forall\beta,\\
&\mathbf{u}_A^{(0)} = \mathbf{u}_\beta^{(0)} \text{ on } \Gamma_\beta \,\forall\beta,
\end{aligned}$$

which is a linear, elastic-type periodic boundary value problem equipped with stress continuity conditions on the interface Γ_β. As this problem is $\mathbf{Y}$ periodic and has continuity over the interface Γ_β then its a classic problem in asymptotic homogenisation. For simplicity we just treat all the terms of the form $(1-\alpha)\mathbb{C}$ as simple fourth order elasticity tensors. Thereby, allowing us to employ a well-known result in asymptotic homogenisation [7, 21], giving us our solution. By solving for the leading order displacements, $\mathbf{u}_A^{(0)}$ and $\mathbf{u}_\beta^{(0)} \,\forall\beta$, we see the solutions are $\mathbf{Y}$ constant functions over each interface Γ_β such that

$$\bar{\mathbf{u}}(\mathbf{X},t) = \mathbf{u}_A^{(0)}(\mathbf{X},t) = \mathbf{u}_\beta^{(0)}(\mathbf{X},t).$$

Here $\bar{\mathbf{u}}$ is a simplification of the notation for the leading order displacement. Employing this result in (3.17), (3.18) and (3.19) allows us write a simplified set of coefficients for $\epsilon^{(0)}$. For the multiscale damage evolution problem by taking $\epsilon \to 0$ we can create a leading order differential problem for $\alpha^{(0)}$. Collecting all the remaining terms yields

$$\begin{aligned}
&\left\langle \nabla_{\mathbf{Y}} \cdot \left(\mathbb{D}_A \nabla_{\mathbf{Y}} \alpha_A^{(0)}\right)\right\rangle_+ = 0 \text{ in } \Omega_A,\\
&\left\langle \nabla_{\mathbf{Y}} \cdot \left(\mathbb{D}_\beta \nabla_{\mathbf{Y}} \alpha_\beta^{(0)}\right)\right\rangle_+ = 0 \text{ in } \Omega_\beta \,\forall\beta,\\
&\mathbb{D}_A \nabla_{\mathbf{Y}} \alpha_A^{(0)} \cdot \mathbf{n}_A = \mathbb{D}_\beta \nabla_{\mathbf{Y}} \alpha_\beta^{(0)} \cdot \mathbf{n}_A \text{ on } \Gamma_\beta \,\forall\beta,\\
&\alpha_A^{(0)} = \alpha_\beta^{(0)} \text{ on } \Gamma_\beta \,\forall\beta,
\end{aligned}$$

which is a diffusion-type cell problem. This problem is equipped with continuity over the interface and local periodicity. Our Macaulay brackets are enforcing irreversibility of damage growth but are also making this a non-standard diffusion-type problem. To solve this problem consider an analogous one dimensional example:

$$\langle f'(X)\rangle_+ = 0 \text{ in } \Omega \text{ with periodic boundary conditions on } \partial\Omega. \tag{3.22}$$

The Macaulay brackets are a piecewise operator enforcing an irreversibility constraint for damage growth everywhere in Ω. Our one dimensional system is constrained by the Macaulay brackets such that $f'(X) \leq 0$ everywhere in Ω. For clarity this comes from the

definition of the Macaulay brackets (2.48). Applying the definition to (3.22) allows us to write

$$\langle f'(X)\rangle_+ = \begin{cases} f'(X) \text{ when } f'(X) \geq 0, \\ 0 \text{ when } f'(X) < 0, \end{cases} \tag{3.23}$$

$$\iff 0 = \begin{cases} f'(X) \text{ when } f'(X) \geq 0, \\ 0 \text{ when } f'(X) < 0. \end{cases} \tag{3.24}$$

From (3.24) it is clear that the positive contributions of $f'(X)$ are constrained to zero. Therefore, we can write down the constraint that $f'(X) \leq 0$. As $f(X)$ is periodic on Ω and assumed to be continuously differentiable then we can introduce the following lemma.

Lemma 3.2.1. *Let $f : \mathbb{R} \to \mathbb{R}$ be T-periodic and continuously differentiable. Then if $f'(x) \leq 0 \, \forall x \in \mathbb{R}$ then f is constant.*

Proof. Let $f : \mathbb{R} \to \mathbb{R}$ be T-periodic, continuously differentiable and $f'(x) \leq 0 \, \forall x \in \mathbb{R}$. Assume f is not constant then $\exists \, a, \, b \in \Omega$ with $a \neq b$ and $f(a) \neq f(b)$. We can assume that $a < b$ and that $a, \, b \in [0, T]$. If $f(a) < f(b)$ then by the Mean Value Theorem $\exists \, y \in (a, b)$ such that

$$f'(y) = \frac{f(b) - f(a)}{b - a} > 0.$$

If $f(a) > f(b)$ then by periodicity $a + T > b$ and $f(a + T) > f(b)$. Therefore according to the Mean Value Theorem $\exists \, y \in (b, a + T)$ such that

$$f'(y) = \frac{f(a + T) - f(b)}{a + T - b} > 0.$$

In both cases we have a contradiction as we require that $f'(x) \leq 0 \, \forall x \in \mathbb{R}$. Hence f must be constant. □

Therefore by Lemma 3.2.1 we can say in our one dimensional example that the solution is a constant function. Similarly in $\mathbb{R}^n$ for our diffusion type problem we can say that we have

$$\langle \nabla \cdot \mathbf{F} \rangle_+ = 0,$$

where $\mathbf{F}(\mathbf{X})$ is a continuously differentiable vector valued function. As previously observed the Macaulay brackets are constraining the function $\mathbf{F}$ such that $\nabla \cdot \mathbf{F} \leq 0$ everywhere in Ω. In our case we know can that $\mathbf{F}$ is a continuously differentiable function and is periodic on $\partial\Omega$. We can therefore infer zero flux boundary conditions, which allows us to introduce the following Lemma.

Lemma 3.2.2. *A continuously differentiable function* $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$*, for* $n \in \mathbb{N}$*, which exists in the domain* $\Omega \in \mathbb{R}^n$ *such that* $\mathbf{F}$ *is periodic on* $\partial\Omega$*. If* $\nabla \cdot \mathbf{F} \leq 0$ *everywhere in* Ω *then* $\mathbf{F}$ *is solenoidal i.e.* $\nabla \cdot \mathbf{F} = 0$*.*

Proof. If $\mathbf{F}$ is periodic on the boundary $\partial\Omega$ then $\mathbf{F}$ has zero flux. Therefore by the divergence theorem we may write that

$$\int_{\partial\Omega} \mathbf{F} \cdot \mathbf{n}\, dA = \int_{\Omega} \nabla \cdot \mathbf{F}\, dV = 0.$$

As $\nabla \cdot \mathbf{F} \leq 0$ everywhere in Ω then by the localisation theorem we must have that $\nabla \cdot \mathbf{F} = 0$ in Ω. Thus, $\mathbf{F}$ is solenoidal. □

Now that we are equipped with Lemma 3.2.2 we can apply it to our diffusion type cell problem. Doing so yields

$$\begin{aligned}
&\nabla_{\mathbf{Y}} \cdot \left(\mathbb{D}_A \nabla_{\mathbf{Y}} \alpha_A^{(0)} \right) = 0 \ \text{ in } \Omega_A, \\
&\nabla_{\mathbf{Y}} \cdot \left(\mathbb{D}_\beta \nabla_{\mathbf{Y}} \alpha_\beta^{(0)} \right) = 0 \ \text{ in } \Omega_\beta \, \forall \beta, \\
&\mathbb{D}_A \nabla_{\mathbf{Y}} \alpha_A^{(0)} \cdot \mathbf{n}_A = \mathbb{D}_\beta \nabla_{\mathbf{Y}} \alpha_\beta^{(0)} \cdot \mathbf{n}_A \ \text{ on } \Gamma_\beta \, \forall \beta, \\
&\alpha_A^{(0)} = \alpha_\beta^{(0)} \ \text{ on } \Gamma_\beta \, \forall \beta,
\end{aligned}$$

which is a standard diffusion type problem with periodic boundary conditions. Being another classic problem in asymptotic homogenisation [103] we can easily solve this problem for the leading order damage. The solution is a $\mathbf{Y}$ constant function over the interfaces Γ_β such that

$$\bar{\alpha}(\mathbf{X}, t) = \alpha_A^{(0)}(\mathbf{X}, t) = \alpha_\beta^{(0)}(\mathbf{X}, t).$$

Leading order displacement and damage both only depend on time and the macroscopic spatial variable with this result we can simplify the equations (3.14), (3.15), (3.17) and (3.18). Equating coefficients of order ϵ^1 for our multiscale momentum problem yields:

$$\begin{aligned}
&\nabla_{\mathbf{Y}} \cdot [(1-\bar{\alpha})^{p_A} \mathbb{C}_A : \xi_{\mathbf{X}}(\bar{\mathbf{u}})] + \nabla_{\mathbf{Y}} \cdot \left[(1-\bar{\alpha})^{p_A} \mathbb{C}_A : \xi_{\mathbf{Y}}(\mathbf{u}_A^{(1)}) \right] \\
&+ \nabla_{\mathbf{Y}} \cdot [(1-\bar{\alpha}_A)^{p_A} \hat{\sigma}_A] = \mathbf{0} \ \text{ in } \Omega_A, \\
&\nabla_{\mathbf{Y}} \cdot [(1-\bar{\alpha})^{p_\beta} \mathbb{C}_\beta : \xi_{\mathbf{X}}(\bar{\mathbf{u}})] + \nabla_{\mathbf{Y}} \cdot \left[(1-\bar{\alpha})^{p_\beta} \mathbb{C}_\beta : \xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(1)}) \right] \\
&+ \nabla_{\mathbf{Y}} \cdot [(1-\alpha_\beta)^{p_\beta} \hat{\sigma}_\beta] = \mathbf{0} \ \text{ in } \Omega_\beta \ \text{ in } \Omega_\beta \, \forall \beta, \\
&(1-\bar{\alpha})^{p_A} \left(\mathbb{C}_A : \xi_{\mathbf{X}}(\bar{\mathbf{u}}) + \mathbb{C}_A : \xi_{\mathbf{Y}}(\mathbf{u}_A^{(1)}) + \hat{\sigma}_A \right) \cdot \mathbf{n}_A \\
&= (1-\bar{\alpha})^{p_\beta} \left(\mathbb{C}_\beta : \xi_{\mathbf{X}}(\bar{\mathbf{u}}) + \mathbb{C}_\beta : \xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(1)}) + \hat{\sigma}_\beta \right) \cdot \mathbf{n}_A \ \text{ on } \Gamma_\beta \, \forall \beta, \\
&\mathbf{u}_A^{(1)} = \mathbf{u}_\beta^{(1)} \ \text{ on } \Gamma_\beta \, \forall \beta.
\end{aligned}$$

This a linear elastic-type periodic boundary value problem equipped with continuity and

stress continuity interface conditions. The above problem reads as a classic auxiliary problem which arises from the asymptotic homogenisation of a linear elastic composite [104, 7, 21]. To solve this problem we exploit linearity and introduce the following ansätze:

$$\mathbf{u}_A^{(1)} = \chi_A : \xi_X(\bar{\mathbf{u}}) + \tilde{\mathbf{u}}_A \quad \text{and} \quad \mathbf{u}_\beta^{(1)} = \chi_\beta : \xi_X(\bar{\mathbf{u}}) + \tilde{\mathbf{u}}_\beta \ \forall \beta. \tag{3.25}$$

These solutions are linear functions in terms of the leading order displacement gradient. The coefficients of these linear functions are the third rank auxiliary tensor χ and the auxiliary vector $\tilde{\mathbf{u}}$. We took special care to ensure that the ansätze we introduce are all vectors, just like the displacements. This is why χ is a third rank tensor and $\tilde{\mathbf{u}}$ is a vector. By substituting these ansätze back into our ϵ^1 problem we can generate a number of local cell problems which define our auxiliary variables. The third rank auxiliary tensor χ is the solution to the following local cell problem:

$$\nabla_{\mathbf{Y}} \cdot \mathbb{C}_A = -\nabla_{\mathbf{Y}} \cdot [\mathbb{C}_A : \xi_{\mathbf{Y}}(\chi_A)] \ \text{ in } \Omega_A, \tag{3.26a}$$

$$\nabla_{\mathbf{Y}} \cdot \mathbb{C}_\beta = -\nabla_{\mathbf{Y}} \cdot [\mathbb{C}_\beta : \xi_{\mathbf{Y}}(\chi_\beta)] \ \text{ in } \Omega_\beta \, \forall \beta, \tag{3.26b}$$

$$\begin{aligned} &(1-\bar{\alpha})^{p_A} \left(\mathbb{C}_A + \mathbb{C}_A : \xi_{\mathbf{Y}}(\chi_A)\right) \cdot \mathbf{n}_A \\ &= (1-\bar{\alpha})^{p_\beta} \left(\mathbb{C}_\beta + \mathbb{C}_\beta : \xi_{\mathbf{Y}}(\chi_\beta)\right) \cdot \mathbf{n}_A \ \text{ on } \Gamma_\beta \, \forall \beta, \end{aligned} \tag{3.26c}$$

$$\chi_A = \chi_\beta \ \text{ on } \Gamma_\beta \, \forall \beta. \tag{3.26d}$$

The auxiliary vector $\tilde{\mathbf{u}}$ is the microscopic residual strain such that it solves the local cell problem:

$$\nabla_{\mathbf{Y}} \cdot [\mathbb{C}_A : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A)] = -\nabla_{\mathbf{Y}} \cdot \sigma_A \ \text{ in } \Omega_A, \tag{3.27a}$$

$$\nabla_{\mathbf{Y}} \cdot [\mathbb{C}_\beta : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta)] = -\nabla_{\mathbf{Y}} \cdot \sigma_\beta \ \text{ in } \Omega_\beta \, \forall \beta, \tag{3.27b}$$

$$\begin{aligned} &(1-\bar{\alpha})^{p_A} \left(\mathbb{C}_A : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A) + \sigma_A\right) \cdot \mathbf{n}_A = \\ &(1-\bar{\alpha})^{p_\beta} \left(\mathbb{C}_\beta : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta) + \sigma_\beta\right) \cdot \mathbf{n}_A \ \text{ on } \Gamma_\beta \, \forall \beta, \end{aligned} \tag{3.27c}$$

$$\tilde{\mathbf{u}}_A = \tilde{\mathbf{u}}_\beta \ \text{ on } \Gamma_\beta \, \forall \beta. \tag{3.27d}$$

In essence χ and $\tilde{\mathbf{u}}$ must be solved for this ansatz across each periodic cell in our system. Therefore one can construct the true solution of $\mathbf{u}^{(1)}$ by solving the systems (3.26) and (3.27), which are linear differential equations. Numerically these are easy to solve and in fact many solutions already exist for these types of cell problems [3, 52, 82]. These cell problems are considered classical to the linear elastic homogenisation process and have been studied extensively, consider the work of Hassani et al. [58, 57, 59]. Our cell problems (3.26) and (3.27) are closed by their periodic boundary conditions but for

uniqueness we also impose the requirement that

$$\{\chi_A\}_{\Omega_A} + \sum_{\beta=1}^{N} \{\chi_\beta\}_{\Omega_\beta} = 0 \quad \text{and} \quad \{\tilde{\mathbf{u}}_A\}_{\Omega_A} + \sum_{\beta=1}^{N} \{\tilde{\mathbf{u}}_\beta\}_{\Omega_\beta} = 0 \quad \forall\, \beta.$$

Remark. *Special consideration should be granted to the physical role of the auxiliary vectors $\tilde{\mathbf{u}}_A$ and $\tilde{\mathbf{u}}_\beta$. Other auxiliary variables only role in our multiscale model is to relay microscopic information about a single associated parameter up to a macroscopic scale. This is done via effective macroscopic coefficients. However, $\tilde{\mathbf{u}}_A$ and $\tilde{\mathbf{u}}_\beta$ both encode information about two associated parameters respectively; $\mathbb{C}_A$ and σ_A, $\mathbb{C}_\beta$ and σ_β. This can be seen in the differential equations* (3.27)*. As a result of this phenomena both the microscopic elasticity and residual stress terms will play a role in defining a macroscopic residual stress. A result which is quite different compared to most other effective macroscopic coefficients seen in the literature [105, 32, 92]. In Chapter 4, we will see that in one dimension the effective residual stress is defined by a macroscopic weighted mean of the local microscopic residual stress, weighted by the residuals of the microscopic elasticities. This implies that the macroscopic residual stress is dominated by the microscopic residual stress contributions of regions that are more elastic. Of course, as this is a weighted mean then the macroscopic residual stress may by dominated by the microscopic residual stress contributions of the largest regions in our local periodic cell. We find this to be an interesting result because it differs from classical asymptotic homogenisation, where we usually find that effective coefficients in one dimension are defined by harmonic means of their respective microscopic contributions.*

Now we approach the damage problem for coefficients of $\epsilon^{(1)}$, one should note that at this order the damage and displacement problems are actually independent of each other. Solving them therefore does not need to be performed in a specific order and each can be approached individually. If we equate all the $\epsilon^{(1)}$ coefficients of (3.17), (3.18) and (3.19) we have:

$$\begin{aligned}
&\left\langle \nabla_{\mathbf{Y}} \cdot \left(\mathbb{D}_A \nabla_{\mathbf{X}} \bar{\alpha}_A + \mathbb{D}_A \nabla_{\mathbf{Y}} \alpha_A^{(1)} \right) \right\rangle_+ = 0 \text{ in } \Omega_A, \\
&\left\langle \nabla_{\mathbf{Y}} \cdot \left(\mathbb{D}_\beta \nabla_{\mathbf{X}} \bar{\alpha}_\beta + \mathbb{D}_\beta \nabla_{\mathbf{Y}} \alpha_\beta^{(1)} \right) \right\rangle_+ = 0 \text{ in } \Omega_\beta \, \forall \beta, \\
&\mathbb{D}_A \nabla_{\mathbf{X}} \bar{\alpha} \cdot \mathbf{n}_A + \mathbb{D}_A \nabla_{\mathbf{Y}} \alpha_A^{(1)} \cdot \mathbf{n}_A = \mathbb{D}_\beta \nabla_{\mathbf{X}} \bar{\alpha} \cdot \mathbf{n}_A + \mathbb{D}_\beta \nabla_{\mathbf{Y}} \alpha_\beta^{(1)} \cdot \mathbf{n}_A \text{ on } \Gamma_\beta \, \forall \beta, \\
&\alpha_A^{(1)} = \alpha_\beta^{(1)}.
\end{aligned}$$

Making use of Lemma 3.2.2 we can drop the Macaulay brackets and simplify this problem.

Hence we are left with a more classical homogenisation problem

$$\begin{aligned}
&\nabla_{\mathbf{Y}} \cdot \left(\mathbb{D}_A \nabla_{\mathbf{X}} \bar{\alpha}_A + \mathbb{D}_A \nabla_{\mathbf{Y}} \alpha_A^{(1)} \right) = 0 \text{ in } \Omega_A, \\
&\nabla_{\mathbf{Y}} \cdot \left(\mathbb{D}_\beta \nabla_{\mathbf{X}} \bar{\alpha}_\beta + \mathbb{D}_\beta \nabla_{\mathbf{Y}} \alpha_\beta^{(1)} \right) = 0 \text{ in } \Omega_\beta \, \forall \beta, \\
&\mathbb{D}_A \nabla_{\mathbf{X}} \bar{\alpha} \cdot \mathbf{n}_A + \mathbb{D}_A \nabla_{\mathbf{Y}} \alpha_A^{(1)} \cdot \mathbf{n}_A = \mathbb{D}_\beta \nabla_{\mathbf{X}} \bar{\alpha} \cdot \mathbf{n}_A + \mathbb{D}_\beta \nabla_{\mathbf{Y}} \alpha_\beta^{(1)} \cdot \mathbf{n}_A \text{ on } \Gamma_\beta \, \forall \beta, \\
&\alpha_A^{(1)} = \alpha_\beta^{(1)} \text{ on } \Gamma_\beta \, \forall \beta,
\end{aligned}$$

which is a linear differential problem for $\alpha_A^{(1)}$ and $\alpha_\beta^{(1)}$. Exploiting linearity we can infer the simple ansatz

$$\alpha_A^{(1)} = \zeta_A \cdot \nabla \bar{\alpha}_A \quad \text{and} \quad \alpha_\beta^{(1)} = \zeta_\beta \cdot \nabla \bar{\alpha}_\beta \quad \forall \beta, \tag{3.28}$$

which solves the ϵ^1 damage problem. The auxiliary variable ζ is a scalar variable connecting the microscopic and macroscopic fields of damage. To calculate the vector ζ we need to solve the local cell problem:

$$\nabla_{\mathbf{Y}} \cdot (\mathbb{D}_A + \mathbb{D}_A \nabla_{\mathbf{Y}} \zeta_A) = \mathbf{0} \text{ in } \Omega_A, \tag{3.29a}$$

$$\nabla_{\mathbf{Y}} \cdot (\mathbb{D}_\beta + \mathbb{D}_\beta \nabla_{\mathbf{Y}} \zeta_\beta) = \mathbf{0} \text{ in } \Omega_\beta \, \forall \beta, \tag{3.29b}$$

$$(\mathbb{D}_A + \mathbb{D}_A \nabla_{\mathbf{Y}} \zeta_A) \, \mathbf{n}_A = (\mathbb{D}_\beta + \mathbb{D}_\beta \nabla_{\mathbf{Y}} \zeta_\beta) \, \mathbf{n}_A \text{ on } \Gamma_\beta \, \forall \beta, \tag{3.29c}$$

$$\zeta_A = \zeta_\beta \text{ on } \Gamma_\beta \, \forall \beta. \tag{3.29d}$$

This problem is closed by its periodic boundary conditions but for uniqueness we require that

$$\{\zeta_A\}_{\Omega_A} + \sum_{\beta=1}^{N} \{\zeta_\beta\}_{\Omega_\beta} = 0 \quad \forall \, \beta.$$

In total then we have three auxiliary variables which need to be calculated but they are all solved linearly. Therefore one can calculate them using simple numerical methods and as previously mentioned they are classical cell problems. By this we mean there is a wealth of literature and even existing software codes that exist which one can use as the foundation of any analytical or numerical methods.

So far we have determined that the leading order fields $\bar{\mathbf{u}}$ and $\bar{\alpha}$ fields depend solely on the macroscopic spatial variable $\mathbf{X}$ and time t. Our cell problems relay the effects the microscopic parameters have on macroscopic coefficients and create a direct linear relation. With these results we can now seek to construct a set of leading order governing equations which can be done by considering the coefficients of ϵ^2. To do this we take coefficients of ϵ^2 from the momentum balance equations or damage evolution equations sum them together and take local integral averages. For the ϵ^2 coefficients we need to consider the power series expansions of the degradation function. More specifically by using the binomial theorem

we can write that

$$\begin{aligned}(1-\alpha)^p &= \sum_{k=0}^{p}\binom{p}{k}(-\alpha)^k \\ &= \sum_{k=0}^{p}\binom{p}{k}\left(\sum_{j=0}^{2}-\epsilon^j\alpha^{(j)}\right)^k + \mathcal{O}(\epsilon^3) \\ &= \sum_{k=0}^{p}\binom{p}{k}(-\bar{\alpha})^k + \sum_{k=0}^{p}\binom{p}{k}\left(-\epsilon\alpha^{(1)}-\epsilon^2\alpha^{(2)}\right)^k + \mathcal{O}(\epsilon^3) \\ &= (1-\bar{\alpha})^p - \epsilon p\alpha^{(1)} + \epsilon^2\left(p\alpha^{(2)} + \binom{p}{2}(\alpha^{(1)})^2\right) + \mathcal{O}(\epsilon^3)\end{aligned}$$

with this we can equate the ϵ^2 coefficients of the momentum balance equation. Taking (3.14), (3.15) and (3.16) we can write down the expansions

$$\begin{aligned}&\nabla_{\mathbf{X}}\cdot\left[(1-\bar{\alpha})^{p_A}\left(\mathbb{C}_A:\xi_{\mathbf{X}}(\bar{\mathbf{u}})+\mathbb{C}_A:\xi_{\mathbf{Y}}(\mathbf{u}^{(1)})+\hat{\sigma}_A\right)\right]+\\ &\nabla_{\mathbf{Y}}\cdot\left[(1-\bar{\alpha})^{p_A}\left(\mathbb{C}_A:\xi_{\mathbf{X}}(\mathbf{u}^{(1)})+\mathbb{C}_A:\xi_{\mathbf{Y}}(\mathbf{u}^{(2)})\right)\right]-\\ &\nabla_{\mathbf{Y}}\cdot\left[p_A\alpha_A^{(1)}\left(\mathbb{C}_A:\xi_{\mathbf{X}}(\bar{\mathbf{u}})+\mathbb{C}_A:\xi_{\mathbf{Y}}(\mathbf{u}^{(1)})+\hat{\sigma}_A\right)\right]+\hat{\rho}_A\mathbf{b}_A=\hat{\rho}_A\ddot{\bar{\mathbf{u}}}\ \text{ in }\Omega_A,\end{aligned}\tag{3.30}$$

$$\begin{aligned}&\nabla_{\mathbf{X}}\cdot\left[(1-\bar{\alpha})^{p_\beta}\left(\mathbb{C}_\beta:\xi_{\mathbf{X}}(\bar{\mathbf{u}})+\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\mathbf{u}^{(1)})+\hat{\sigma}_\beta\right)\right]+\\ &\nabla_{\mathbf{Y}}\cdot\left[(1-\bar{\alpha})^{p_\beta}\left(\mathbb{C}_\beta:\xi_{\mathbf{X}}(\mathbf{u}^{(1)})+\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\mathbf{u}^{(2)})\right)\right]-\\ &\nabla_{\mathbf{Y}}\cdot\left[p_\beta\alpha_\beta^{(1)}\left(\mathbb{C}_\beta:\xi_{\mathbf{X}}(\bar{\mathbf{u}})+\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\mathbf{u}^{(1)})+\hat{\sigma}_\beta\right)\right]+\hat{\rho}_\beta\mathbf{b}_\beta=\hat{\rho}_\beta\ddot{\bar{\mathbf{u}}}\ \text{ in }\Omega_\beta\,\forall\beta,\end{aligned}\tag{3.31}$$

$$\begin{aligned}&(1-\bar{\alpha}_A)^{p_A}\left(\mathbb{C}_A:\xi_{\mathbf{X}}(\mathbf{u}_A^{(1)})+\mathbb{C}_A:\xi_{\mathbf{Y}}(\mathbf{u}_A^{(2)})\right)\cdot\mathbf{n}_A\\ &-p_A\alpha_A^{(1)}\left(\mathbb{C}_A:\xi_{\mathbf{X}}(\bar{\mathbf{u}})+\mathbb{C}_A:\xi_{\mathbf{Y}}(\mathbf{u}_A^{(1)})+\hat{\sigma}_A\right)\cdot\mathbf{n}_A\\ &=(1-\bar{\alpha}_\beta)^{p_\beta}\left(\mathbb{C}_\beta:\xi_{\mathbf{X}}(\mathbf{u}_\beta^{(1)})+\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(2)})\right)\cdot\mathbf{n}_A\\ &-p_\beta\alpha_\beta^{(1)}\left(\mathbb{C}_\beta:\xi_{\mathbf{X}}(\bar{\mathbf{u}})+\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(1)})+\hat{\sigma}_\beta\right)\cdot\mathbf{n}_A\ \text{ on }\Gamma_\beta\,\forall\beta.\end{aligned}\tag{3.32}$$

We now aim to construct a macroscopic momentum balance equation for our composite material. To do this we apply the local integral operator (3.6) over Ω_A and Ω_β, on (3.30) and (3.31), for $\beta = 1, ..., N$, respectively. We then sum every resulting contribution and

apply the divergence theorem in $\mathbf{Y}$, accounting for periodicity on $\partial\Omega$, allowing us to obtain:

$$\begin{aligned}
&\sum_{\beta=1}^{N}\frac{1}{|\Omega|}\Bigg[\int_{\Gamma_\beta}(1-\bar{\alpha}_A)^{p_A}\left(\mathbb{C}_A:\xi_{\mathbf{X}}(\mathbf{u}_A^{(1)})+\mathbb{C}_A:\xi_{\mathbf{Y}}(\mathbf{u}_A^{(2)})\right)\cdot\mathbf{n}_A\,dA_{\mathbf{Y}}\\
&-\int_{\Gamma_\beta}p_A\alpha_A^{(1)}\left(\mathbb{C}_A:\xi_{\mathbf{X}}(\bar{\mathbf{u}})+\mathbb{C}_A:\xi_{\mathbf{Y}}(\mathbf{u}_A^{(1)})+\hat{\sigma}_A\right)\cdot\mathbf{n}_A\,dA_{\mathbf{Y}}\\
&-\int_{\Gamma_\beta}(1-\bar{\alpha}_\beta)^{p_\beta}\left(\mathbb{C}_\beta:\xi_{\mathbf{X}}(\mathbf{u}_\beta^{(1)})+\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(2)})\right)\cdot\mathbf{n}_A\,dA_{\mathbf{Y}}\\
&+\int_{\Gamma_\beta}p_\beta\alpha_\beta^{(1)}\left(\mathbb{C}_\beta:\xi_{\mathbf{X}}(\bar{\mathbf{u}})+\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(1)})+\hat{\sigma}_\beta\right)\cdot\mathbf{n}_A\,dA_{\mathbf{Y}}\Bigg]\\
&+\frac{1}{|\Omega|}\int_{\Omega_A}\nabla_{\mathbf{X}}\cdot\left[(1-\bar{\alpha})^{p_A}\left(\mathbb{C}_A:\xi_{\mathbf{X}}(\bar{\mathbf{u}})+\mathbb{C}_A:\xi_{\mathbf{Y}}(\mathbf{u}^{(1)})+\hat{\sigma}_A\right)\right]\,dV_{\mathbf{Y}}\\
&+\sum_{\beta=1}^{N}\frac{1}{|\Omega|}\int_{\Omega_\beta}\nabla_{\mathbf{X}}\cdot\left[(1-\bar{\alpha})^{p_\beta}\left(\mathbb{C}_\beta:\xi_{\mathbf{X}}(\bar{\mathbf{u}})+\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\mathbf{u}^{(1)})+\hat{\sigma}_\beta\right)\right]\,dV_{\mathbf{Y}}\\
&+\frac{1}{|\Omega|}\left[\int_{\Omega_A}\rho_A\mathbf{b}_A\,dV_{\mathbf{Y}}+\sum_{\beta=1}^{N}\int_{\Omega_\beta}\rho_\beta\mathbf{b}_\beta\,dV_{\mathbf{Y}}\right]=\frac{1}{|\Omega|}\left[\int_{\Omega_A}\rho_A\,dV_{\mathbf{Y}}+\sum_{\beta=1}^{N}\int_{\Omega_\beta}\rho_\beta\,dV_{\mathbf{Y}}\right]\ddot{\bar{\mathbf{u}}}.
\end{aligned}$$

Exploiting the interface relation (3.32) we can see that all the integrals over the interfaces Γ_β reduce to zero. Finally by accounting for macroscopic uniformity of leading order terms and our ansatz (3.25) we can simplify our system. We are left with

$$\begin{aligned}
&\nabla_{\mathbf{X}}\cdot\left[(1-\bar{\alpha})^{p_A}\left\{\mathbb{C}_A+\mathbb{C}_A:\xi_{\mathbf{Y}}(\chi_A)\right\}_{\Omega_A}:\xi_{\mathbf{X}}(\bar{\mathbf{u}})+(1-\bar{\alpha})^{p_A}\left\{\hat{\sigma}_A+\mathbb{C}_A:\xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A)\right\}_{\Omega_A}\right]\\
&+\sum_{\beta=1}^{N}\nabla_{\mathbf{X}}\cdot\left[(1-\bar{\alpha})^{p_\beta}\left\{\mathbb{C}_\beta+\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\chi_\beta)\right\}_{\Omega_\beta}:\xi_{\mathbf{X}}(\bar{\mathbf{u}})+(1-\bar{\alpha})^{p_\beta}\left\{\hat{\sigma}_\beta+\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta)\right\}_{\Omega_\beta}\right]\\
&+\bar{\mathbf{B}}=\bar{\rho}\ddot{\bar{\mathbf{u}}},
\end{aligned}\tag{3.33}$$

which is a macroscopic governing momentum balance equation. Each constitutive part of the composite Ω has its own unique degradation function governed by an exponent that controls material softening. In order to simplify this into a macroscopic governing equation we limit ourselves to the case $p_A=p_\beta=p\quad\forall\beta$. Therefore our above formulation yields

$$\nabla_{\mathbf{X}}\cdot\left[g(\bar{\alpha})\left(\bar{\mathbb{C}}:\xi_{\mathbf{X}}(\bar{\mathbf{u}})+\bar{\sigma}\right)\right]+\bar{\mathbf{B}}=\bar{\rho}\ddot{\bar{\mathbf{u}}},\tag{3.34}$$

where every variable with a bar is a leading order term and every coefficient marked with a bar is an effective macroscopic coefficient. We have reintroduced the notation $g(\bar{\alpha})$ as the damage degradation function (3.8) for the leading order damage. Note that the macroscopic momentum balance equation is quite similar to the microscopic formulations of momentum balance.

Our effective macroscopic coefficients for the momentum balance are defined by:

$$\begin{aligned}
\bar{\mathbb{C}} &= \{\mathbb{C}_A + \mathbb{C}_A : \xi_{\mathbf{Y}}(\chi_A)\}_{\Omega_A} + \sum_{\beta=1}^{N} \{\mathbb{C}_\beta + \mathbb{C}_\beta : \xi_{\mathbf{Y}}(\chi_\beta)\}_{\Omega_\beta}\,, \\
\bar{\sigma} &= \{\hat{\sigma}_A + \mathbb{C}_A : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A)\}_{\Omega_A} + \sum_{\beta=1}^{N} \{\hat{\sigma}_\beta + \mathbb{C}_\beta : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta)\}_{\Omega_\beta}\,, \\
\bar{\rho} &= \{\rho_A\}_{\Omega_A} + \sum_{\beta=1}^{N} \{\rho_\beta\}_{\Omega_\beta}\,, \quad \bar{\mathbf{B}} = \{\rho_A \mathbf{b}_A\}_{\Omega_A} + \sum_{\beta=1}^{N} \{\rho_\beta \mathbf{b}_\beta\}_{\Omega_\beta}\,.
\end{aligned} \tag{3.35}$$

The effective coefficients $\bar{\mathbb{C}}$ and $\bar{\sigma}$ depend on the ansatz solutions to the local cell problems (3.26) and (3.27). This entangles information about the microscopic parameters and geometry into the macroscopic variables. However, it is important to note that the coefficient $\bar{\rho}$ and $\bar{\mathbf{B}}$, are just a local averaging of the material density and body forces, respectively.

An important thing to note is the large number of parameters in our multiscale model. Let us first consider the mechanical effective coefficients defined by (3.35) if our model was in $\mathbb{R}^2$, two dimensions: $\bar{\rho}$ is a scalar accounting for one parameter and $\bar{\mathbf{B}}$ is a vector accounting for two parameters; the effective residual stress $\bar{\sigma}$ is a second rank tensor and therefore in two dimensions accounts for four parameters; $\bar{\mathbb{C}}$ is a fourth rank tensor and is defined by sixteen parameters. In total then a two dimensional version of (3.34) would account for two equations dependent on three unknowns with twenty three parameters. However, it grows rapidly in higher dimensions. Let us consider $\mathbb{R}^3$, the parameters defined by (3.35) would account for many more parameters: $\bar{\rho}$ still accounts for one; $\bar{\mathbf{B}}$ accounts for three; $\bar{\sigma}$ account for nine and $\bar{\mathbb{C}}$ accounts for eighty one. That is a total of ninety four parameters across three equations, defined by four unknowns. Now consider if we tried to computationally model a damage problem with a full microscopic problem. If in that problem we had $k \in \mathbb{N}$ local cells where each of which was made up of $N+1$ elastic bodies then in that system, for three dimensions, we would have $94k(N+1)$ parameters. Defined across $4k(N+1)$ equations with $4k(N+1)$ unknowns. This is just not computationally feasible or efficient in the slightest. We consider this the best argument for why effective multiscale models are necessary to perform an efficient analysis. It should also be noted that later in this chapter we shall conclude that $\bar{\mathbb{C}}$ also has minor and major symmetries, thus lowering the number of parameters it depends on, simplifying our system.

To formulate a macroscopic damage evolution equation we follow a similar process as the one which derived equation (3.34). So we equate the coefficients of ϵ^2 for (3.17), (3.18)

and (3.19) which yields the system:

$$\begin{aligned}\eta_A\dot{\bar{\alpha}} = \Bigg\langle \frac{p_A}{2}(1-\bar{\alpha})^{p_A-1}\Bigg(&\xi_{\mathbf{X}}(\bar{\mathbf{u}}):\mathbb{C}_A:\xi_{\mathbf{X}}(\bar{\mathbf{u}}) + \xi_{\mathbf{X}}(\bar{\mathbf{u}}):\mathbb{C}_A:\xi_{\mathbf{Y}}(\mathbf{u}_A^{(1)}) \\ &+ \xi_{\mathbf{Y}}(\mathbf{u}_A^{(1)}):\mathbb{C}_A:\xi_{\mathbf{X}}(\bar{\mathbf{u}}) + \xi_{\mathbf{Y}}(\mathbf{u}_A^{(1)}):\mathbb{C}_A:\xi_{\mathbf{Y}}(\mathbf{u}_A^{(1)}) + \hat{\sigma}_A:\xi_{\mathbf{X}}(\bar{\mathbf{u}}) \\ &+ \hat{\sigma}_A:\xi_{\mathbf{Y}}(\mathbf{u}_A^{(1)}) - \hat{\psi}_A\Bigg) - \hat{G}_A\bar{\alpha} + \nabla_{\mathbf{X}}\cdot\left(\mathbb{D}_A\nabla_{\mathbf{X}}\bar{\alpha}\right) + \nabla_{\mathbf{X}}\cdot\left(\mathbb{D}_A\nabla_{\mathbf{Y}}\alpha_A^{(1)}\right) \\ &+ \nabla_{\mathbf{Y}}\cdot\left(\mathbb{D}_A\nabla_{\mathbf{X}}\alpha_A^{(1)}\right) + \nabla_{\mathbf{Y}}\cdot\left(\mathbb{D}_A\nabla_{\mathbf{Y}}\alpha_A^{(2)}\right)\Bigg\rangle_+ \text{ in } \Omega_A,\end{aligned}$$

$$\begin{aligned}\eta_\beta\dot{\bar{\alpha}} = \Bigg\langle \frac{p_\beta}{2}(1-\bar{\alpha})^{p_\beta-1}\Bigg(&\xi_{\mathbf{X}}(\bar{\mathbf{u}}):\mathbb{C}_\beta:\xi_{\mathbf{X}}(\bar{\mathbf{u}}) + \xi_{\mathbf{X}}(\bar{\mathbf{u}}):\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(1)}) \\ &+ \xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(1)}):\mathbb{C}_\beta:\xi_{\mathbf{X}}(\bar{\mathbf{u}}) + \xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(1)}):\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(1)}) + \hat{\sigma}_\beta:\xi_{\mathbf{X}}(\bar{\mathbf{u}}) \\ &+ \hat{\sigma}_\beta:\xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(1)}) - \hat{\psi}_\beta\Bigg) - \hat{G}_\beta\bar{\alpha} + \nabla_{\mathbf{X}}\cdot\left(\mathbb{D}_\beta\nabla_{\mathbf{X}}\bar{\alpha}\right) + \nabla_{\mathbf{X}}\cdot\left(\mathbb{D}_\beta\nabla_{\mathbf{Y}}\alpha_\beta^{(1)}\right) \\ &+ \nabla_{\mathbf{Y}}\cdot\left(\mathbb{D}_\beta\nabla_{\mathbf{X}}\alpha_\beta^{(1)}\right) + \nabla_{\mathbf{Y}}\cdot\left(\mathbb{D}_\beta\nabla_{\mathbf{Y}}\alpha_\beta^{(2)}\right)\Bigg\rangle_+ \text{ in } \Omega_\beta\,\forall\beta,\end{aligned}$$

$$\left(\mathbb{D}_A\nabla_{\mathbf{X}}\alpha_A^{(1)} + \mathbb{D}_A\nabla_{\mathbf{Y}}\alpha_A^{(2)}\right)\mathbf{n}_A = \left(\mathbb{D}_\beta\nabla_{\mathbf{X}}\alpha_\beta^{(1)} + \mathbb{D}_\beta\nabla_{\mathbf{Y}}\alpha_\beta^{(2)}\right)\mathbf{n}_A \text{ on } \Gamma_\beta\,\forall\beta. \tag{3.36}$$

Previously we have made use of Lemma 3.2.2 to account for the irreversibility constraints enforced by the Macaulay brackets, unfortunately the above problems no longer satisfies the Lemma 3.2.2. To tackle the irreversibility constraints we need to reformulate the above problem into an equivalent system of equations that satisfy the irreversibility constraints. By doing so we can then drop the Macaulay brackets, putting the system into a much more malleable form to work with. Our irreversibility constraint is that $\dot{\alpha} \geq 0$ and by considering the power series expansion

$$\dot{\alpha} = \dot{\bar{\alpha}} + \sum_{i=1}^{\infty}\epsilon^j\dot{\alpha}^{(j)} \to \dot{\bar{\alpha}} \text{ as } \epsilon \to 0,$$

it is clear that leading order damage irreversibility must satisfy $\dot{\bar{\alpha}} \geq 0$ as $\epsilon \to 0$. We can bypass the Macaulay brackets by considering a different formulation of our damage evolution equation. Consider $\dot{\alpha} = \langle F\rangle_+$ an equivalent formulation is the constrained system

$$\dot{\alpha}\left[\dot{\alpha} - F(\alpha, \mathbf{u})\right] = 0 \text{ with the constraint } \dot{\alpha} \geq 0.$$

Note this formulation is equivalent to Karush-Khun-Tucker conditions for a constrained variational problem. An interesting paper that remarks on this is Pham et al. [84] which explores a rate independent damage phase field problem. If we then multiply our ϵ^2 damage

evolution equations through by the leading order damage growth rate, $\dot{\bar{\alpha}}$, this yields

$$\begin{aligned}
&\dot{\bar{\alpha}}\Bigg[\eta_A\dot{\bar{\alpha}} - p_A(1-\bar{\alpha})^{p_A-1}\Bigg(\frac{1}{2}\xi_{\mathbf{X}}(\bar{\mathbf{u}}):\mathbb{C}_A:\xi_{\mathbf{X}}(\bar{\mathbf{u}}) + 2\xi_{\mathbf{X}}(\bar{\mathbf{u}}):\mathbb{C}_A:\xi_{\mathbf{Y}}(\mathbf{u}_A^{(1)})\\
&+\frac{1}{2}\xi_{\mathbf{Y}}(\mathbf{u}_A^{(1)}):\mathbb{C}_A:\xi_{\mathbf{X}}(\bar{\mathbf{u}}) + \xi_{\mathbf{Y}}(\mathbf{u}_A^{(1)}):\mathbb{C}_A:\xi_{\mathbf{Y}}(\mathbf{u}_A^{(1)}) + \hat{\sigma}_A:\xi_{\mathbf{X}}(\bar{\mathbf{u}})\\
&+\hat{\sigma}_A:\xi_{\mathbf{Y}}(\mathbf{u}_A^{(1)}) - \hat{\psi}_A\Bigg) + \hat{G}_A\bar{\alpha} - \nabla_{\mathbf{X}}\cdot\Big(\mathbb{D}_A\nabla_{\mathbf{X}}\bar{\alpha}\Big) - \nabla_{\mathbf{X}}\cdot\Big(\mathbb{D}_A\nabla_{\mathbf{Y}}\alpha_A^{(1)}\Big)\\
&-\nabla_{\mathbf{Y}}\cdot\Big(\mathbb{D}_A\nabla_{\mathbf{X}}\alpha_A^{(1)}\Big) - \nabla_{\mathbf{Y}}\cdot\Big(\mathbb{D}_A\nabla_{\mathbf{Y}}\alpha_A^{(2)}\Big)\Bigg] = 0 \text{ in } \Omega_A,
\end{aligned} \tag{3.37}$$

$$\begin{aligned}
&\dot{\bar{\alpha}}\Bigg[\eta_\beta\dot{\bar{\alpha}} - p_\beta(1-\bar{\alpha})^{p_\beta-1}\Bigg(\frac{1}{2}\xi_{\mathbf{X}}(\bar{\mathbf{u}}):\mathbb{C}_\beta:\xi_{\mathbf{X}}(\bar{\mathbf{u}}) + 2\xi_{\mathbf{X}}(\bar{\mathbf{u}}):\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(1)})\\
&+\frac{1}{2}\xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(1)}):\mathbb{C}_\beta:\xi_{\mathbf{X}}(\bar{\mathbf{u}}) + \xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(1)}):\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(1)}) + \hat{\sigma}_\beta:\xi_{\mathbf{X}}(\bar{\mathbf{u}})\\
&+\hat{\sigma}_\beta:\xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(1)}) - \hat{\psi}_\beta\Bigg) + \hat{G}_\beta\bar{\alpha} - \nabla_{\mathbf{X}}\cdot\Big(\mathbb{D}_\beta\nabla_{\mathbf{X}}\bar{\alpha}\Big) - \nabla_{\mathbf{X}}\cdot\Big(\mathbb{D}_\beta\nabla_{\mathbf{Y}}\alpha_\beta^{(1)}\Big)\\
&-\nabla_{\mathbf{Y}}\cdot\Big(\mathbb{D}_\beta\nabla_{\mathbf{X}}\alpha_\beta^{(1)}\Big) - \nabla_{\mathbf{Y}}\cdot\Big(\mathbb{D}_\beta\nabla_{\mathbf{Y}}\alpha_\beta^{(2)}\Big)\Bigg] = 0 \text{ in } \Omega_\beta\ \forall\beta,
\end{aligned} \tag{3.38}$$

To construct a macroscopic damage evolution equation for our composite material we apply the local integral operator (3.6) over Ω_A and Ω_β, in (3.37) and (3.38), for $\beta = 1, ..., N$, respectively. We then sum every resulting contribution and apply the divergence theorem in $\mathbf{Y}$, accounting for periodicity on $\partial\Omega$, allows us to obtain:

$$\begin{aligned}
&\sum_{\beta=1}^{N}\frac{1}{|\Omega|}\int_{\Gamma_\beta}\dot{\bar{\alpha}}\Bigg[\Big(\mathbb{D}_\beta\nabla_{\mathbf{X}}\alpha_\beta^{(1)} + \mathbb{D}_\beta\nabla_{\mathbf{Y}}\alpha_\beta^{(2)}\Big)\mathbf{n}_A - \Big(\mathbb{D}_A\nabla_{\mathbf{X}}\alpha_A^{(1)} + \mathbb{D}_A\nabla_{\mathbf{Y}}\alpha_A^{(2)}\Big)\mathbf{n}_A\Bigg]dA_{\mathbf{Y}}\\
&+\frac{1}{|\Omega|}\int_{\Omega_A}\dot{\bar{\alpha}}\Bigg[\eta_A\dot{\bar{\alpha}} - \frac{p_A}{2}(1-\bar{\alpha})^{p_A-1}\Bigg(\xi_{\mathbf{X}}(\bar{\mathbf{u}}):\mathbb{C}_A:\xi_{\mathbf{X}}(\bar{\mathbf{u}}) + \xi_{\mathbf{X}}(\bar{\mathbf{u}}):\mathbb{C}_A:\xi_{\mathbf{Y}}(\mathbf{u}_A^{(1)})\\
&+\xi_{\mathbf{Y}}(\mathbf{u}_A^{(1)}):\mathbb{C}_A:\xi_{\mathbf{X}}(\bar{\mathbf{u}}) + \xi_{\mathbf{Y}}(\mathbf{u}_A^{(1)}):\mathbb{C}_A:\xi_{\mathbf{Y}}(\mathbf{u}_A^{(1)}) + \hat{\sigma}_A:\xi_{\mathbf{X}}(\bar{\mathbf{u}})\\
&+\hat{\sigma}_A:\xi_{\mathbf{Y}}(\mathbf{u}_A^{(1)}) - \hat{\psi}_A\Bigg) + \hat{G}_A\bar{\alpha} - \nabla_{\mathbf{X}}\cdot\Big(\mathbb{D}_A\nabla_{\mathbf{X}}\bar{\alpha}\Big) - \nabla_{\mathbf{X}}\cdot\Big(\mathbb{D}_A\nabla_{\mathbf{Y}}\alpha_A^{(1)}\Big)\Bigg]dV_{\mathbf{Y}}\\
&+\sum_{\beta=1}^{N}\frac{1}{|\Omega|}\int_{\Omega_\beta}\dot{\bar{\alpha}}\Bigg[\eta_\beta\dot{\bar{\alpha}} - \frac{p_\beta}{2}(1-\bar{\alpha})^{p_\beta-1}\Bigg(\xi_{\mathbf{X}}(\bar{\mathbf{u}}):\mathbb{C}_\beta:\xi_{\mathbf{X}}(\bar{\mathbf{u}}) + \xi_{\mathbf{X}}(\bar{\mathbf{u}}):\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(1)})\\
&+\xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(1)}):\mathbb{C}_\beta:\xi_{\mathbf{X}}(\bar{\mathbf{u}}) + \xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(1)}):\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(1)}) + \hat{\sigma}_\beta:\xi_{\mathbf{X}}(\bar{\mathbf{u}})\\
&+\hat{\sigma}_\beta:\xi_{\mathbf{Y}}(\mathbf{u}_\beta^{(1)}) - \hat{\psi}_\beta\Bigg) + \hat{G}_\beta\bar{\alpha} - \nabla_{\mathbf{X}}\cdot\Big(\mathbb{D}_\beta\nabla_{\mathbf{X}}\bar{\alpha}\Big) - \nabla_{\mathbf{X}}\cdot\Big(\mathbb{D}_\beta\nabla_{\mathbf{Y}}\alpha_\beta^{(1)}\Big)\Bigg]dV_{\mathbf{Y}} = 0.
\end{aligned}$$

By exploiting our interface condition (3.36) we can reduce the top line of the above equation

to zero. Now if we introduce the ansatz solutions (3.28) and (3.25) we can construct an equation of leading order terms $\bar{\mathbf{u}}$ and $\bar{\alpha}$. To create a macroscopic system of equations its important that we remind ourselves of the transpose definition for a fourth order tensor [97, 104].

Definition 3.2.1. Let $\mathbb{T}$ be a fourth rank tensor. The transpose of $\mathbb{T}$, denoted by $\mathbb{T}^T$, is the unique fourth rank tensor which satisfies

$$A : \mathbb{T} : B = B : \mathbb{T}^T : A \tag{3.39}$$

for all second rank tensors A and B. For any $\mathbb{T}$ with components T_{ijkl} with $i, j, k, l = 1, 2, 3$ then the component-wise representation of the transpose $\mathbb{T}^T$ reads as T_{klij}, therefore $\mathbb{T} = \mathbb{T}^T$ if and only if $\mathbb{T}$ contains major symmetries. Applying the component-wise representation it is clear that for two four rank tensors $\mathbb{S}$ and $\mathbb{T}$, we can find that

$$(\mathbb{S}\mathbb{T})^T = \mathbb{T}^T\mathbb{S}^T.$$

We further notice that, whenever $\mathbb{T}$ is major symmetric, then for every fourth rank tensor $\mathbb{T}$ we have

$$\left(\mathbb{A}^T\mathbb{T}\mathbb{A}\right)^T = \left(\mathbb{A}^T\left(\mathbb{A}^T\mathbb{T}\right)^T\right) = \mathbb{A}^T\mathbb{T}\mathbb{A}, \tag{3.40}$$

meaning that $\mathbb{A}^T\mathbb{T}\mathbb{A}$ is major symmetric.

Now we may exploit the macroscopic uniformity of the leading order variables and reintroducing the notation for local integral averages (3.6) we can create a number of

simplifications. So that we may write

$$\begin{aligned}
&\dot{\bar{\alpha}}\frac{1}{|\Omega|}\Bigg[\int_{\Omega_A}\eta_A\dot{\bar{\alpha}}-p_A(1-\bar{\alpha})^{p_A-1}\Bigg[\frac{1}{2}\xi_{\mathbf{X}}(\bar{\mathbf{u}}):\Big(\mathbb{C}_A+2\mathbb{C}_A:\xi_{\mathbf{Y}}(\chi_A)+\\
&+\xi_{\mathbf{Y}}(\chi_A)^T:\mathbb{C}_A:\xi_{\mathbf{Y}}(\chi_A)\Big):\xi_{\mathbf{X}}(\bar{\mathbf{u}})-\hat{\psi}_A+\left(\hat{\sigma}_A+\frac{1}{2}\xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A):\mathbb{C}_A\right):\xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A)\\
&+(\hat{\sigma}_A+\mathbb{C}_A:\xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A)+\hat{\sigma}_A:\xi_{\mathbf{Y}}(\chi_A)+\xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A):\mathbb{C}_A:\xi_{\mathbf{Y}}(\chi_A)):\xi_{\mathbf{X}}(\bar{\mathbf{u}})\Bigg]+\hat{G}_A\bar{\alpha}+\\
&-\nabla_{\mathbf{X}}\cdot\Big((\mathbb{D}_A+\mathbb{D}_A\nabla_{\mathbf{Y}}\zeta_A)\nabla_{\mathbf{X}}\bar{\alpha}\Big)dV_{\mathbf{Y}}+\\
&+\sum_{\beta=1}^{N}\int_{\Omega_\beta}\eta_\beta\dot{\bar{\alpha}}-p_\beta(1-\bar{\alpha})^{p_\beta-1}\Bigg[\frac{1}{2}\xi_{\mathbf{X}}(\bar{\mathbf{u}}):\Big(\mathbb{C}_\beta+2\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\chi_\beta)+\\
&+\xi_{\mathbf{Y}}(\chi_\beta)^T:\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\chi_\beta)\Big):\xi_{\mathbf{X}}(\bar{\mathbf{u}})-\hat{\psi}_\beta+\left(\hat{\sigma}_\beta+\frac{1}{2}\xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta):\mathbb{C}_\beta\right):\xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta)\\
&+(\hat{\sigma}_\beta+\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta)+\hat{\sigma}_\beta:\xi_{\mathbf{Y}}(\chi_\beta)+\xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta):\mathbb{C}_\beta:\xi_{\mathbf{Y}}(\chi_\beta)):\xi_{\mathbf{X}}(\bar{\mathbf{u}})\Bigg]+\hat{G}_\beta\bar{\alpha}+\\
&-\nabla_{\mathbf{X}}\cdot\Big((\mathbb{D}_\beta+\mathbb{D}_\beta\nabla_{\mathbf{Y}}\zeta_\beta)\nabla_{\mathbf{X}}\bar{\alpha}\Big)dV_{\mathbf{Y}}\Bigg]=0.
\end{aligned}\tag{3.41}$$

The above formulation is a macroscopic model with no dependencies on the microscale defined by $\mathbf{Y}$ but it can be massively simplified. To do so we assume as previously that $p_A=p_\beta=p$ for all $\beta=1,...,N$. As a result we can now write that

$$\dot{\bar{\alpha}}\left[\bar{\eta}\dot{\bar{\alpha}}+g'(\bar{\alpha})\left(\frac{1}{2}\xi_{\mathbf{X}}(\bar{\mathbf{u}}):\mathbb{P}:\xi_{\mathbf{X}}(\bar{\mathbf{u}})+\bar{\kappa}:\xi_{\mathbf{X}}(\bar{\mathbf{u}})-\bar{\psi}\right)+\bar{G}\bar{\alpha}-\nabla_{\mathbf{X}}\cdot\left(\bar{\mathbb{D}}\nabla_{\mathbf{X}}\bar{\alpha}\right)\right]=0,\tag{3.42}$$

where $\bar{\eta}$, $\bar{\mathbb{P}}$, $\bar{\kappa}$, $\bar{\psi}$, $\bar{G}$ and $\bar{\mathbb{D}}$ form a set of effective macroscopic coefficients. Note that the macroscopic damage evolution equation is similar to the microscopic system of equations for the damage evolution equations. This damage evolution equation is paired with the macroscopic irreversibility constraint:

$$\dot{\bar{\alpha}}\geq 0.\tag{3.43}$$

We can therefore make use of Macaulay brackets to reformulate the macroscopic damage evolution equation into a simpler form

$$\dot{\bar{\eta}}\bar{\alpha}=\left\langle -g'(\bar{\alpha})\left(\frac{1}{2}\xi_{\mathbf{X}}(\bar{\mathbf{u}}):\mathbb{P}:\xi_{\mathbf{X}}(\bar{\mathbf{u}})+\bar{\kappa}:\xi_{\mathbf{X}}(\bar{\mathbf{u}})-\bar{\psi}\right)-\bar{G}\bar{\alpha}+\nabla_{\mathbf{X}}\cdot\left(\bar{\mathbb{D}}\nabla_{\mathbf{X}}\bar{\alpha}\right)\right\rangle_{+}.\tag{3.44}$$

As seen with the macroscopic momentum balance equation we have a set of effective macroscopic coefficients that encode information from the microscale to the macroscale. Here the effective macroscopic coefficients for the damage evolution equation are defined

as:

$$
\begin{aligned}
\bar{\mathbb{P}} &= \left\{\mathbb{C}_A + 2\mathbb{C}_A : \xi_{\mathbf{Y}}(\chi_A) + \xi_{\mathbf{Y}}(\chi_A)^T : \mathbb{C}_A : \xi_{\mathbf{Y}}(\chi_A)\right\}_{\Omega_A} + \\
&\quad + \sum_{\beta=1}^{N} \left\{\mathbb{C}_\beta + 2\mathbb{C}_\beta : \xi_{\mathbf{Y}}(\chi_\beta) + \xi_{\mathbf{Y}}(\chi_\beta)^T : \mathbb{C}_\beta : \xi_{\mathbf{Y}}(\chi_\beta)\right\}_{\Omega_\beta}, \\
\bar{\kappa} &= \left\{\hat{\sigma}_A + \mathbb{C}_A : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A) + \hat{\sigma}_A : \xi_{\mathbf{Y}}(\chi_A) + \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A) : \mathbb{C}_A : \xi_{\mathbf{Y}}(\chi_A)\right\}_{\Omega_A} + \\
&\quad + \sum_{\beta=1}^{N} \left\{\hat{\sigma}_\beta + \mathbb{C}_\beta : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta) + \hat{\sigma}_\beta : \xi_{\mathbf{Y}}(\chi_\beta) + \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta) : \mathbb{C}_\beta : \xi_{\mathbf{Y}}(\chi_\beta)\right\}_{\Omega_\beta}, \\
\bar{\psi} &= \left\{\hat{\psi}_A + \left(\hat{\sigma}_A + \frac{1}{2}\xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A) : \mathbb{C}_A\right) : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A)\right\}_{\Omega_A} + \\
&\quad + \sum_{\beta=1}^{N} \left\{\hat{\psi}_\beta + \left(\hat{\sigma}_\beta + \frac{1}{2}\xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta) : \mathbb{C}_\beta\right) : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta)\right\}_{\Omega_\beta}, \\
\bar{\mathbb{D}} &= \left\{\mathbb{D}_A + \mathbb{D}_A \nabla_{\mathbf{Y}} \zeta_A\right\}_{\Omega_A} + \sum_{\beta=1}^{N} \left\{\mathbb{D}_\beta + \mathbb{D}_\beta \nabla_{\mathbf{Y}} \zeta_\beta\right\}_{\Omega_\beta}, \\
\bar{G} &= \left\{G_A\right\}_{\Omega_A} + \sum_{\beta=1}^{N} \left\{G_\beta\right\}_{\Omega_\beta}, \quad \bar{\eta} = \left\{\eta_A\right\}_{\Omega_A} + \sum_{\beta=1}^{N} \left\{\eta_\beta\right\}_{\Omega_\beta}.
\end{aligned}
\tag{3.45}
$$

Recall that the curly brackets are the local integral averages defined by (3.6). The coefficients $\bar{\mathbb{P}}$, $\bar{\kappa}$, $\bar{\psi}$ and $\bar{\mathbb{D}}$ depend on the ansatz defined by (3.25) and (3.28), which are calculated from the local cell problems (3.26), (3.27) and (3.29). $\bar{\mathbb{P}}$ is an effective fourth order elasticity tensor, $\bar{\kappa}$ is an effective second order residual stress tensor, $\bar{\psi}$ is an effective damage threshold and $\bar{\mathbb{D}}$ is an effective second order diffusion tensor. The coefficients $\bar{G}$ and $\bar{\eta}$ are simply local integral averages of the tearing energy and viscosity, respectively. These effective coefficients for the damage evolution equation are quite complex and to the best of our knowledge have not been documented in the literature.

The above system (3.45) gives us a new set of effective coefficients defining our macroscopic damage evolution equation (3.44). Let us note how many independent parameters that (3.45) actually represents as several of our coefficients are tensors of differing orders. First let us consider a two dimensional case, in $\mathbb{R}^2$. Then the scalar functions $\bar{\psi}$, $\bar{G}$ and $\bar{\eta}$ represent three independent parameters. The fourth order elasticity tensor $\bar{\mathbb{P}}$ in two dimensions represents sixteen independent parameters. The second order tensors $\bar{\kappa}$ and $\bar{\mathbb{D}}$ each represent four independent parameters in two dimensions. Therefore, in two dimensions our damage evolution equation depends on twenty seven independent parameters. This of course dramatically increases when we go to three dimensions, in $\mathbb{R}^3$. In three dimensions we can say that: $\bar{\mathbb{P}}$ depends on eighty one parameters; $\bar{\sigma}$ and $\bar{\kappa}$ each depend on nine parameters; $\bar{\psi}$, $\bar{G}$ and $\bar{\eta}$ each depend on one parameter. Therefore, in three dimensions the effective coefficients defined by (3.45) depend on a total of one hundred and two parameters. However, as previously mentioned, later in this chapter we will show that

$\bar{\mathbb{P}}$ has both minor and major symmetries. This will lower the number of parameters our system depends on, simplifying our model.

A keen observer will note that our macroscopic system at this point has a subtle inconsistency. Our two sets of effective coefficients, for the momentum balance (3.35) and damage evolution (3.45), have two pairs of coefficients doing the same job. Here $\bar{\mathbb{P}}$ and $\bar{\mathbb{C}}$ are both the effective macroscopic elasticity tensors while $\bar{\sigma}$ and $\bar{\kappa}$ are both defined as the effective macroscopic residual stress tensor. This is a major inconsistency in our system. How can two coupled equations describing the physical process of the same object have different material parameters? The fact is they cannot. In the next section we will see that by exploring the properties of these effective coefficients and their respective cell problems allows us to make a number of simplifications. In turn we will see that $\bar{\mathbb{C}} = \bar{\mathbb{P}}$ and $\bar{\sigma} = \bar{\kappa}$, showing that our macroscopic system of equations is in fact consistent.

Remark. *It should be noted that the equations* (3.33) *and* (3.41) *along with a set of boundary conditions for* $\bar{\mathbf{u}}$ *and* $\bar{\alpha}$ *form a macroscopic system of equations. Unfortunately, every linear elastic subdomain in* Ω *has its own unique form of the degradation function in this system of equations. Making this an extremely difficult system of equations to work with. This is because one would have to work with a set of* $N + 1$ *degradation functions defined for each linear elastic subdomain in* Ω*. It would be very possible to implement a numerical model via software that allows one to have a widely varying degradation function across the macroscale. For this to work one would have to recalculate the effective coefficients for every time step, for a number of representative local periodic cells across the macroscale, possibly more than once if an iterative solver is implemented. Therefore, one could numerically solve* (3.33) *and* (3.41) *for the macroscopic variables* $\bar{\mathbf{u}}$ *and* $\bar{\alpha}$*. This system would be far more computationally feasible than solving the full microscopic system, because it would require a lower resolution, but it is still less efficient than the system defined by* (3.34) *and* (3.44)*.*

3.3 Properties of the effective coefficients

At this point of our derivation the partial differential equations (3.34) and (3.44) represent the macroscopic governing equations of our system. They are parameterised by a set of complex effective coefficients, (3.35) and (3.45), giving the system most of its properties. A better understanding of these effective coefficients will help us to simplify the system. In fact in this section we will see that some simple manipulation of the local cell problems, defining the ansatze, produce a number of relations that vastly simplify our effective coefficients. In this section we will prove that $\bar{\mathbb{C}} = \bar{\mathbb{P}}$ and that $\bar{\sigma} = \bar{\kappa}$, thus simplifying the macroscopic governing equations. This will also make our system physically consistent by making sure all the effective coefficients are consistent across our momentum balance and

damage evolution equations.

In this section we will discuss five effective coefficients: the effective elasticity tensors $\bar{\mathbb{C}}$ and $\bar{\mathbb{P}}$, the effective residual stress tensors $\bar{\sigma}$ and $\bar{\kappa}$; and the effective threshold function $\bar{\psi}$. It should be noted similar analysis of the properties of effective macroscopic coefficients has been performed [57, 104]. The novelty here is the simplifications these results will cause, alongside ensuring our model makes complete physical sense. It would be completely non-nonsensical if $\bar{\mathbb{P}} \neq \bar{\mathbb{C}}$ and the elasticity of Ω at the macroscale was defined by two different coefficients. Let us consider that we did not construct a multiscale model and just modelled from a macroscale treating Ω as a single locally homogenous elastic body. Then following the derivation of the governing equations in §2.2 we would of course expect to have a single elasticity tensor at the macroscale. Currently, we have two elasticity tensors and residual stress tensors in the macroscopic governing equations defined by (3.34) and (3.44). That is totally inconsistent and hence for our multiscale model and derivation to be physically consistent we must have $\bar{\mathbb{C}} = \bar{\mathbb{P}}$. Similarly, we can make the same physical argument that for mathematical consistency we must have $\bar{\sigma} = \bar{\kappa}$. This motivates our investigation of the properties of the effective coefficients.

Before we proceed, it is useful to introduce some new notation. This section relies heavily on summation notation, where summing up repeated indices is understood to be the norm. The strain tensor ξ appears in all of the local cell problems. So its important for us to understand how to use this function in summation notation. For a third rank tensor χ we would write,

$$\xi_{ij}^{kl}(\chi) = \frac{1}{2}\left(\frac{\partial \chi_{jkl}}{\partial y_i} + \frac{\partial \chi_{ikl}}{\partial y_j}\right). \tag{3.46}$$

Similarly for a vector $\mathbf{v}$ we would write

$$\xi_{ij}(\chi) = \frac{1}{2}\left(\frac{\partial v_j}{\partial y_i} + \frac{\partial v_i}{\partial y_j}\right).$$

We also need to introduce some background theory about the properties of $\xi_{\mathbf{Y}}(\chi_A)$ and $\xi_{\mathbf{Y}}(\chi_\beta)$ before we can proceed. The following results where first reported by Penta et al. [104]. This theory will play a role in proving the positive definiteness of $\bar{\mathbb{C}}$.

Theorem 3.3.1. *Let $\mathbb{M}$ be the fourth rank tensor defined by*

$$\mathbb{M} = \begin{cases} \xi_{\mathbf{Y}}(\chi_A), & \mathbf{Y} \in \Omega_A \\ \xi_{\mathbf{Y}}(\chi_\beta), & \mathbf{Y} \in \Omega_\beta. \end{cases} \tag{3.47}$$

Then the following facts hold:

1. *$\mathbb{M}$ possesses both left and right minor symmetries.*

2. $\{\mathbb{M}\} = 0$

Proof. 1. The tensor $\mathbb{M}$ is left minor symmetric as $\xi_{\mathbf{Y}}(\chi_A)$ and $\xi_{\mathbf{Y}}(\chi_\beta)$ are both left minor symmetric, this can be seen in (3.46). Right minor symmetry of $\mathbb{M}$ is deduced from the cell problems in (3.26) employing the right minor symmetry of $\mathbb{C}_A$ and $\mathbb{C}_\beta$.

2. As $\mathbb{M}$ is minor symmetric, then $\{\mathbb{M}\}$ is also minor symmetric, accordingly we only need to prove that for every $\mathbf{Y}$-constant second rank symmetric tensor A that

$$\{\mathbb{M}\}\, A = 0.$$

Let us define

$$\mathbf{v} = \begin{cases} \mathbf{v}_A, & \mathbf{Y} \in \Omega_A \\ \mathbf{v}_\beta, & \mathbf{Y} \in \Omega_\beta, \end{cases} \tag{3.48}$$

$$\mathbf{v}_A = \chi_A A \quad \text{and} \quad \mathbf{v}_\beta = \chi_\beta A. \tag{3.49}$$

We can therefore rewrite $\{\mathbb{M}\}\, A$ with (3.48) and (3.49) to calculate

$$\begin{aligned} \{\mathbb{M}\}\, A &= \{\mathbb{M}A\} \\ &= \{\xi_{\mathbf{Y}}(\chi_A)A\}_A + \sum_{\beta=1}^{N} \{\xi_{\mathbf{Y}}(\chi_\beta)A\}_\beta \\ &= \{\xi_{\mathbf{Y}}(\mathbf{v}_A)\}_A + \sum_{\beta=1}^{N} \{\xi_{\mathbf{Y}}(\mathbf{v}_\beta)\}_\beta \\ &= \frac{1}{2|\Omega|} \sum_{\beta=1}^{N} \int_{\Gamma_\beta} (\mathbf{v}_A - \mathbf{v}_\beta) \otimes \mathbf{n}_A + \mathbf{n}_A \otimes (\mathbf{v}_A - \mathbf{v}_\beta)\; dA_{\mathbf{Y}} \\ &= 0. \end{aligned}$$

Here the contributions over the boundary $\partial\Omega$ cancel due to $\mathbf{Y}$-periodicity and we applied the continuity of the third rank tensors χ_A and χ_β as defined in (3.26), which in turn implies the continuity of $\mathbf{v}$ across the interfaces Γ_β, for all β. We have therefore completed the proof and shown

$$\{\mathbb{M}\} = 0.$$

□

3.3.1 The effective elasticity tensor

Consider a leading order macroscopic potential energy term $\psi^{(0)}$ such that

$$\psi^{(0)} = \frac{1}{2}\xi_{\mathbf{X}}(\bar{\mathbf{u}}) : \mathbb{S} : \xi_{\mathbf{X}}(\bar{\mathbf{u}}) + \bar{\kappa} : \xi_{\mathbf{X}}(\bar{\mathbf{u}}),$$

this implies the existence of a leading order linear stress term. One can therefore directly calculate a leading order stress term by taking a partial derivative with respect to the infinitesimal strain tensor. Doing so produces the leading order stress

$$\frac{\partial \psi^{(0)}}{\partial \left(\xi_{\mathbf{X}}(\bar{\mathbf{u}})\right)} = \mathbb{S} : \xi_{\mathbf{X}}(\bar{\mathbf{u}}) + \bar{\kappa},$$

which has an elasticity term $\mathbb{S}$ consistent with the leading order elastic potential energy. This intuitive example is to illustrate how strange it is that the governing macroscopic equations (3.34) and (3.44) have two different terms representing the effective macroscopic elasticity. Initially it was thought that this may be due to how different parts of the model interact with the microscale. However, after careful analysis we can conclude that actually $\bar{\mathbb{C}} = \bar{\mathbb{P}}$.

We also managed to derive a number of other properties that would be expected of a leading order elasticity tensor. We will show that $\bar{\mathbb{C}}$ posses both minor and major symmetries. Then finally we will conclude that $\bar{\mathbb{C}}$ is a positive definite fourth order tensor as would be expected of an elasticity term. This is all done in the following Theorem 3.3.2.

As $\bar{\mathbb{C}}$ has both minor and major symmetries then the number of parameters it depends on now decreases. In $\mathbb{R}^2$ we can say that $\bar{\mathbb{C}}$ now only depends on six parameters, and in $\mathbb{R}^3$ we can say that $\bar{\mathbb{C}}$ now only depends on twenty one parameters. This result makes our model simpler and easier to implement numerically later in Chapter 5.

Theorem 3.3.2. *Let us consider the fourth rank tensors $\bar{\mathbb{C}}$ and $\bar{\mathbb{P}}$ defined by* (3.35) *and* (3.45)*, respectively. These tensors have the following properties:*

1. *The tensors are in fact equivalent, $\bar{\mathbb{C}} = \bar{\mathbb{P}}$.*
2. *$\bar{\mathbb{C}}$ has minor and major symmetries.*
3. *$\bar{\mathbb{C}}$ is positive definite.*

Proof. 1. We begin by recalling the local cell problems (3.26) that define χ_A and χ_β for all β. In summation notation we can write down equations (3.26a) and (3.26b) as

$$-\frac{\partial}{\partial y_j}\mathbb{C}^A_{ijkl} = \frac{\partial}{\partial y_j}\left[\mathbb{C}^A_{ijpq}\xi^{kl}_{pq}(\chi^A)\right] \quad \text{and} \quad -\frac{\partial}{\partial y_j}\mathbb{C}^\beta_{ijkl} = \frac{\partial}{\partial y_j}\left[\mathbb{C}^\beta_{ijpq}\xi^{kl}_{pq}(\chi^\beta)\right]. \tag{3.50}$$

If we multiply each of these expressions by χ^A_{irs} and χ^β_{irs}, respectively, then using the product rule and the symmetries of the elasticity tensors we can sum these expressions up and integrate over the local cell Ω. As a result we are left with the

expression

$$\begin{aligned}
&\int_{\Omega_A} -\frac{\partial}{\partial y_j}\left[\mathbb{C}^A_{ijkl}\chi^A_{irs}\right] + \mathbb{C}^A_{ijkl}\xi^{ij}_{rs}(\chi^A)\,dV_{\mathbf{Y}} \\
&+\sum_{\beta=1}^{N}\int_{\Omega_\beta} -\frac{\partial}{\partial y_j}\left[\mathbb{C}^\beta_{ijkl}\chi^\beta_{irs}\right] + \mathbb{C}^\beta_{ijkl}\xi^{rs}_{ij}(\chi^\beta)\,dV_{\mathbf{Y}} \\
&= \int_{\Omega_A}\frac{\partial}{\partial y_j}\left[\mathbb{C}^A_{ijpq}\xi^{kl}_{pq}(\chi^A)\chi^A_{irs}\right] - \mathbb{C}^A_{ijpq}\xi^{kl}_{pq}(\chi^A)\xi^{rs}_{ij}(\chi^A)\,dV_{\mathbf{Y}} \\
&+\sum_{\beta=1}^{N}\int_{\Omega_\beta}\frac{\partial}{\partial y_j}\left[\mathbb{C}^\beta_{ijpq}\xi^{kl}_{pq}(\chi^\beta)\chi^\beta_{irs}\right] - \mathbb{C}^\beta_{ijpq}\xi^{kl}_{pq}(\chi^\beta)\xi^{rs}_{ij}(\chi^\beta)\,dV_{\mathbf{Y}}.
\end{aligned}$$

If we now exploit the divergence theorem and local periodicity we can simplify the above expression. We are now left with

$$\begin{aligned}
&\int_{\Omega_A}\mathbb{C}^A_{ijkl}\xi^{rs}_{ij}(\chi^A)\,dV_{\mathbf{Y}} + \sum_{\beta=1}^{N}\int_{\Omega_\beta}\mathbb{C}^\beta_{ijkl}\xi^{rs}_{ij}(\chi^\beta)\,dV_{\mathbf{Y}} \\
&-\sum_{\beta=1}^{N}\int_{\Gamma_\beta}\mathbb{C}^A_{ijkl}\chi^A_{irs}n^A_i + \mathbb{C}^\beta_{ijkl}\chi^\beta_{irs}n^A_i\,dA_{\mathbf{Y}} \\
&= -\int_{\Omega_A}\mathbb{C}^A_{ijpq}\xi^{kl}_{pq}(\chi^A)\xi^{rs}_{ij}(\chi^A)dV_{\mathbf{Y}} - \sum_{\beta=1}^{N}\int_{\Omega_\beta}\mathbb{C}^\beta_{ijpq}\xi^{kl}_{pq}(\chi^\beta)\xi^{rs}_{ij}(\chi^\beta)\,dV_{\mathbf{Y}} \\
&+\sum_{\beta=1}^{N}\int_{\Gamma_\beta}\mathbb{C}^A_{ijpq}\xi^{kl}_{pq}(\chi^A)\chi^A_{irs}n^A_i + \mathbb{C}^\beta_{ijpq}\xi^{kl}_{pq}(\chi^\beta)\chi^\beta_{irs}n^A_i\,dA_{\mathbf{Y}}.
\end{aligned}$$

We can eliminate the interface terms on Γ_β by making use of the interface condition for elasticity (3.26c) and continuity of χ at the interface (3.26d). Now using notation for local integral averages (3.6), the definition of the tensor transpose and dropping the summation notation allows us to define the relation

$$\begin{aligned}
&\left\{\mathbb{C}_A : \xi_{\mathbf{Y}}(\chi_A)\right\}_{\Omega_A} + \sum_{\beta=1}^{N}\left\{\mathbb{C}_\beta : \xi_{\mathbf{Y}}(\chi_\beta)\right\}_{\Omega_\beta} \\
&= -\left\{\xi^T_{\mathbf{Y}}(\chi_A) : \mathbb{C}_A : \xi_{\mathbf{Y}}(\chi_A)\right\}_{\Omega_A} - \sum_{\beta=1}^{N}\left\{\xi^T_{\mathbf{Y}}(\chi_\beta) : \mathbb{C}_\beta : \xi_{\mathbf{Y}}(\chi_\beta)\right\}_{\Omega_\beta}.
\end{aligned} \tag{3.51}$$

If we now plug in this result to the expression for $\bar{\mathbb{P}}$ we will recover $\bar{\mathbb{C}}$:

$$\begin{aligned}\bar{\mathbb{P}} &= \left\{\mathbb{C}_A + 2\mathbb{C}_A : \xi_{\mathbf{Y}}(\chi_A) + \xi_{\mathbf{Y}}(\chi_A)^T : \mathbb{C}_A : \xi_{\mathbf{Y}}(\chi_A)\right\}_{\Omega_A} + \\ &+ \sum_{\beta=1}^{N} \left\{\mathbb{C}_\beta + 2\mathbb{C}_\beta : \xi_{\mathbf{Y}}(\chi_\beta) + \xi_{\mathbf{Y}}(\chi_\beta)^T : \mathbb{C}_\beta : \xi_{\mathbf{Y}}(\chi_\beta)\right\}_{\Omega_\beta} \\ &= \left\{\mathbb{C}_A + \mathbb{C}_A : \xi_{\mathbf{Y}}(\chi_A)\right\}_{\Omega_A} + \sum_{\beta=1}^{N} \left\{\mathbb{C}_\beta + \mathbb{C}_\beta : \xi_{\mathbf{Y}}(\chi_\beta)\right\}_{\Omega_\beta} \\ &= \bar{\mathbb{C}}\end{aligned}$$

2. To prove major symmetries we use the relationship (3.51) to reformulate $\bar{\mathbb{C}}$ as

$$\begin{aligned}\bar{\mathbb{C}} = &\left\{\left(\mathbb{I} + \xi_{\mathbf{Y}}(\chi_A)\right)^T : \mathbb{C}_A : \left(\mathbb{I} + \xi_{\mathbf{Y}}(\chi_A)\right)\right\}_{\Omega_A} \\ &+ \sum_{\beta=1}^{N} \left\{\left(\mathbb{I} + \xi_{\mathbf{Y}}(\chi_\beta)\right)^T : \mathbb{C}_\beta : \left(\mathbb{I} + \xi_{\mathbf{Y}}(\chi_\beta)\right)\right\}_{\Omega_\beta}.\end{aligned} \tag{3.52}$$

This is clearly major symmetric according to linearity and the transpose property (3.40). To prove minor symmetries of $\bar{\mathbb{C}}$ it is clear that the only terms we need to show that have left and right minor symmetries are

$$\left\{\mathbb{C}_A : \xi_{\mathbf{Y}}(\chi_A)\right\}_{\Omega_A} \quad \text{and} \quad \left\{\mathbb{C}_\beta : \xi_{\mathbf{Y}}(\chi_\beta)\right\}_{\Omega_\beta} \quad \forall \beta.$$

Left symmetries for these terms are employed from the left symmetries of the elasticity tensors. The right symmetries of these terms are deduced from the cell problems (3.50) employing the right minor symmetries of the elasticity tensors. Thus, $\bar{\mathbb{C}}$ is both minor and major symmetric.

3. To prove positive definiteness of $\bar{\mathbb{C}}$ it is useful to use the formulation (3.52) of $\bar{\mathbb{C}}$. We need to show that for every $\mathbf{Y}$-constant, second rank symmetric tensor A that

$$A\bar{\mathbb{C}}A = \left\{B_A : \mathbb{C}_A : B_A\right\}_{\Omega_A} + \sum_{\beta=1}^{N} \left\{B_\beta : \mathbb{C}_\beta : B_\beta\right\}_{\Omega_\beta} \geq 0,$$

where we have set:

$$\begin{aligned}B_A &= \left(\mathbb{I} + \xi_{\mathbf{Y}}(\chi_A)\right) A \\ B_\beta &= \left(\mathbb{I} + \xi_{\mathbf{Y}}(\chi_\beta)\right) A \quad \forall\, \beta.\end{aligned}$$

It follows that $A\bar{\mathbb{C}}A = 0$ if and only if $B_A = 0$ and $B_\beta = 0$ for all β. Therefore, to prove positive definiteness then we only need to show that

$$A = 0 \iff B_A = 0 \quad \text{and} \quad B_\beta = 0 \quad \forall\, \beta.$$

Firstly, its obvious that if $A = 0$ then we must have $B_A = 0$ and $B_\beta = 0$ for all β. On the other hand if we have $B_A = 0$ and $B_\beta = 0$ for all β then we must consider

$$\{\mathbb{I} + \xi_{\mathbf{Y}}(\chi_A)\}_A A + \sum_{\beta=1}^{N} \{\mathbb{I} + \xi_{\mathbf{Y}}(\chi_\beta)\}_\beta A = 0 \quad \forall\, \beta.$$

We can simplify this expression by making use of the definition for $\mathbb{M}$ in (3.47). This allows us to write

$$\{\mathbb{I} + \mathbb{M}\} A = 0.$$

Theorem 3.3.1 tells us that $\{\mathbb{M}\} = 0$ and we know that $\{\mathbb{I}\} = 1$. Thus we must have that $A = 0$, completing the proof of positive definiteness for $\bar{\mathbb{C}}$.

□

3.3.2 The effective residual stress tensor

The effective coefficient $\bar{\sigma}$ defines the residual stress term in equation (3.34), similarly $\bar{\kappa}$ defines the residual stress term in equation (3.44). It does not seem intuitive that a multiscale model of one system would have two effective coefficients fulfilling the same role in different equations. To overcome this we can derive a relation from the local cell problem (3.27) which defines the auxiliary vectors $\tilde{\mathbf{u}}_A$ and $\tilde{\mathbf{u}}_\beta$ for all β. Using this relation we will show that $\bar{\kappa} = \bar{\sigma}$, this is done in the following Theorem 3.3.3. The benefit of this work is a simplification in the notation and providing physical consistency to the multiscale model.

Theorem 3.3.3. *Consider the second rank residual stress tensors $\bar{\sigma}$ and $\bar{\kappa}$ defined by* (3.35) *and* (3.45)*, respectively. It is true that $\bar{\kappa} = \bar{\sigma}$.*

Proof. We begin by recalling the local cell problem that defines $\tilde{\mathbf{u}}_A$ and $\tilde{\mathbf{u}}_\beta$ for all β. In summation notation we can write down equations (3.27a) and (3.27b) as

$$-\frac{\partial \hat{\sigma}^A_{ijkl}}{\partial y_j} = \frac{\partial}{\partial y_j}\left[\mathbb{C}^A_{ijpq}\xi^{kl}_{pq}(\tilde{\mathbf{u}}^A_{pq})\right] \quad \text{and} \quad -\frac{\partial \hat{\sigma}^\beta_{ijkl}}{\partial y_j} = \frac{\partial}{\partial y_j}\left[\mathbb{C}^\beta_{ijpq}\xi^{kl}_{pq}(\tilde{\mathbf{u}}^\beta_{pq})\right] \quad \forall\, \beta. \tag{3.53}$$

We multiply each of these expressions by χ^A_{irs} and χ^β_{irs}, respectively, then using the product rule we can sum up these expressions and integrate over a local cell Ω. As a result we are

left with the expression

$$\begin{aligned}
&\int_{\Omega_A} -\frac{\partial}{\partial y_j}\left(\hat{\sigma}^A_{ij}\chi^A_{irs}\right) + \hat{\sigma}^A_{ij}\xi^{rs}_{ij}(\chi^A)\,dV_{\mathbf{Y}} \\
&+\sum_{\beta=1}^{N}\int_{\Omega_\beta} -\frac{\partial}{\partial y_j}\left(\hat{\sigma}^\beta_{ij}\chi^\beta_{irs}\right) + \hat{\sigma}^\beta_{ij}\xi^{rs}_{ij}(\chi^\beta)\,dV_{\mathbf{Y}} \\
&=\int_{\Omega_A} -\frac{\partial}{\partial y_j}\left(\mathbb{C}^A_{ijpq}\xi_{pq}(\tilde{\mathbf{u}}^A)\chi^A_{irs}\right) - \mathbb{C}^A_{ijpq}\xi_{pq}(\tilde{\mathbf{u}}^A)\xi^{rs}_{ij}(\chi^A)\,dV_{\mathbf{Y}} \\
&+\sum_{\beta=1}^{N}\int_{\Omega_\beta} -\frac{\partial}{\partial y_j}\left(\mathbb{C}^\beta_{ijpq}\xi_{pq}(\tilde{\mathbf{u}}^\beta)\chi^A_{irs}\right) - \mathbb{C}^\beta_{ijpq}\xi_{pq}(\tilde{\mathbf{u}}^\beta)\xi^{rs}_{ij}(\chi^\beta)\,dV_{\mathbf{Y}}.
\end{aligned}$$

Making use of divergence theorem, local periodicity, the interface condition (3.26c) and continuity conditions on the interface (3.26d) we can simplify the above expression. We can now write down

$$\begin{aligned}
&\int_{\Omega_A} \hat{\sigma}^A_{ij}\xi^{rs}_{ij}(\chi^A)\,dV_{\mathbf{Y}} + \sum_{\beta=1}^{N}\int_{\Omega_\beta} \hat{\sigma}^\beta_{ij}\xi^{rs}_{ij}(\chi^\beta)\,dV_{\mathbf{Y}} \\
&= -\int_{\Omega_A} \mathbb{C}^A_{ijpq}\xi_{pq}(\tilde{\mathbf{u}}^A)\xi^{rs}_{ij}(\chi^A)\,dV_{\mathbf{Y}} - \sum_{\beta=1}^{N}\int_{\Omega_\beta} \mathbb{C}^\beta_{ijpq}\xi_{pq}(\tilde{\mathbf{u}}^\beta)\xi^{rs}_{ij}(\chi^\beta)\,dV_{\mathbf{Y}}.
\end{aligned}$$

Dropping summation notation and making use of the local integral average notation (3.6) we have the expression

$$\begin{aligned}
&\left\{\hat{\sigma}_A : \xi(\chi_A)\right\}_A + \sum_{\beta=1}^{N}\left\{\hat{\sigma}_\beta\xi(\chi_\beta)\right\}_\beta \\
&= -\left\{\xi(\tilde{\mathbf{u}}_A) : \mathbb{C}_A : \xi(\chi_A)\right\}_A - \sum_{\beta=1}^{N}\left\{\xi(\tilde{\mathbf{u}}_\beta) : \mathbb{C}_\beta : \xi(\chi_\beta)\right\}_\beta.
\end{aligned}$$

If we now plug this expression into κ we find that

$$\begin{aligned}
\bar{\kappa} &= \left\{\hat{\sigma}_A + \mathbb{C}_A : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A) + \hat{\sigma}_A : \xi_{\mathbf{Y}}(\chi_A) + \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A) : \mathbb{C}_A : \xi_{\mathbf{Y}}(\chi_A)\right\}_{\Omega_A} + \\
&+\sum_{\beta=1}^{N}\left\{\hat{\sigma}_\beta + \mathbb{C}_\beta : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta) + \hat{\sigma}_\beta : \xi_{\mathbf{Y}}(\chi_\beta) + \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta) : \mathbb{C}_\beta : \xi_{\mathbf{Y}}(\chi_\beta)\right\}_{\Omega_\beta}, \\
&= \left\{\hat{\sigma}_A + \mathbb{C}_A : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A)\right\}_{\Omega_A} + \sum_{\beta=1}^{N}\left\{\hat{\sigma}_\beta + \mathbb{C}_\beta : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta)\right\}_{\Omega_\beta}, \\
&= \bar{\sigma}
\end{aligned}$$

□

3.3.3 The effective damage threshold

The last effective coefficient we are interested in is, $\bar{\phi}$, the effective damage threshold. Currently it has a lot of contributing microscopic terms, such as microscopic residual stress, elasticity and of course the damage threshold. By manipulating the local cell problems (3.27), which defines the auxiliary vectors $\tilde{\mathbf{u}}_A$ and $\tilde{\mathbf{u}}_\beta$ for all β, we can find a relation that will simplify $\bar{\phi}$. This is done in Lemma 3.3.4 and follows a similar procedure to Theorem 3.3.2 and 3.3.3.

Lemma 3.3.4. *It can be shown from the local cell problem* (3.27) *that the following equality holds:*

$$-\left\{\xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A) : \mathbb{C}_A : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A)\right\}_{\Omega_A} - \sum_{\beta=1}^{N}\left\{\xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta) : \mathbb{C}_\beta : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta)\right\}_{\Omega_\beta} = \left\{\sigma_A : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A)\right\}_{\Omega_A} + \sum_{\beta=1}^{N}\left\{\sigma_\beta : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta)\right\}_{\Omega_\beta}.$$

Proof. We begin by recalling the local cell problem that defines $\tilde{\mathbf{u}}_A$ and $\tilde{\mathbf{u}}_\beta$ for all β. In summation notation we can write down equations (3.27a) and (3.27b) as

$$-\frac{\partial \hat{\sigma}^A_{ijkl}}{\partial y_j} = \frac{\partial}{\partial y_j}\left[\mathbb{C}^A_{ijpq}\xi^{kl}_{pq}(\tilde{\mathbf{u}}^A_{pq})\right] \quad \text{and} \quad -\frac{\partial \hat{\sigma}^\beta_{ijkl}}{\partial y_j} = \frac{\partial}{\partial y_j}\left[\mathbb{C}^\beta_{ijpq}\xi^{kl}_{pq}(\tilde{\mathbf{u}}^\beta_{pq})\right] \quad \forall\, \beta. \tag{3.54}$$

We multiply each of these expressions by $\tilde{\mathbf{u}}^A_i$ and $\tilde{\mathbf{u}}^\beta_i$, respectively, then using the product rule we can sum up these expressions and integrate over a local cell Ω. As a result we are left with the expression

$$\begin{aligned}
&\int_{\Omega_A} -\frac{\partial}{\partial y_j}\left(\hat{\sigma}^A_{ij}\tilde{\mathbf{u}}^A_i\right) + \hat{\sigma}^A_{ij}\xi_{ij}(\tilde{\mathbf{u}}^A)\, dV_{\mathbf{Y}} \\
&+ \sum_{\beta=1}^{N}\int_{\Omega_\beta} -\frac{\partial}{\partial y_j}\left(\hat{\sigma}^\beta_{ij}\tilde{\mathbf{u}}^\beta_i\right) + \hat{\sigma}^\beta_{ij}\xi_{ij}(\tilde{\mathbf{u}}^\beta)\, dV_{\mathbf{Y}} \\
&= -\int_{\Omega_A}\frac{\partial}{\partial y_j}\left(\mathbb{C}^A_{ijpq}\xi_{pq}(\tilde{\mathbf{u}}^A)\tilde{\mathbf{u}}^A_i\right) + \mathbb{C}^A_{ijpq}\xi_{pq}(\tilde{\mathbf{u}}^A)\xi_{ij}(\tilde{\mathbf{u}}^A)\, dV_{\mathbf{Y}} \\
&- \sum_{\beta=1}^{N}\int_{\Omega_\beta}\frac{\partial}{\partial y_j}\left(\mathbb{C}^\beta_{ijpq}\xi_{pq}(\tilde{\mathbf{u}}^\beta)\tilde{\mathbf{u}}^\beta_i\right) + \mathbb{C}^\beta_{ijpq}\xi_{pq}(\tilde{\mathbf{u}}^\beta)\xi_{ij}(\tilde{\mathbf{u}}^\beta)\, dV_{\mathbf{Y}}.
\end{aligned}$$

Making use of divergence theorem, local periodicity, the interface condition (3.27c) and continuity conditions on the interface (3.27d) we can simplify the above expression. We

can now write down

$$\int_{\Omega_A} \hat{\sigma}^A_{ij}\xi_{ij}(\tilde{\mathbf{u}}^A)\,dV_{\mathbf{Y}} + \sum_{\beta=1}^{N}\int_{\Omega_\beta} \hat{\sigma}^\beta_{ij}\xi_{ij}(\tilde{\mathbf{u}}^\beta)\,dV_{\mathbf{Y}}$$
$$= -\int_{\Omega_A} \mathbb{C}^A_{ijpq}\xi_{pq}(\tilde{\mathbf{u}}^A)\xi_{ij}(\tilde{\mathbf{u}}^A)\,dV_{\mathbf{Y}} - \sum_{\beta=1}^{N}\int_{\Omega_\beta} \mathbb{C}^\beta_{ijpq}\xi_{pq}(\tilde{\mathbf{u}}^\beta)\xi_{ij}(\tilde{\mathbf{u}}^\beta)\,dV_{\mathbf{Y}}.$$

Dropping summation notation and making use of the local integral average notation (3.6) we have the expression

$$-\left\{\xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A) : \mathbb{C}_A : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A)\right\}_{\Omega_A} - \sum_{\beta=1}^{N}\left\{\xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta) : \mathbb{C}_\beta : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta)\right\}_{\Omega_\beta} =$$
$$\left\{\sigma_A : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A)\right\}_{\Omega_A} + \sum_{\beta=1}^{N}\left\{\sigma_\beta : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta)\right\}_{\Omega_\beta}.$$

□

3.4 Summary of results

In this Chapter we have derived a novel multiscale model of the damage phase field method. We have homogenised both the mechanical behaviour and damage evolution of an arbitrary linear elastic composite in $\mathbb{R}^n$. This is a novel mathematical model which will allow us to study a large variety of microscopic variations from a macroscopic perspective. We understand that the model is large and complicated so in this section we will write the model out clearly and compactly, using all the simplifications derived in §3.3.

We have derived two governing equations in the form of partial differential equations, which must be solved at every macroscopic point in space. Asymptotic homogenisation has reduced the number of necessary governing equations from $2N+1$ equations for every local cell to just two equations at the macroscale. Inferring the results of Theorem 3.3.2, 3.3.3 and Lemma 3.3.4 we can simplify the model even further. To do this we recall the macroscopic equations (3.34) and (3.44) for the composite domain Ω with local periodicity and large scale disparity such that ϵ is necessarily small. Writing it down we have the macroscopic equation of motion

$$\nabla_{\mathbf{X}} \cdot \left[g(\bar{\alpha})\left(\bar{\mathbb{C}} : \xi_{\mathbf{X}}(\bar{\mathbf{u}}) + \bar{\sigma}\right)\right] + \bar{\mathbf{B}} = \bar{\rho}\ddot{\bar{\mathbf{u}}}, \tag{3.55}$$

which governs the macroscopic displacement $\bar{\mathbf{u}}$. Similarly we can write down a simplified

version of the damage evolution equation

$$\bar{\eta}\dot{\bar{\alpha}} = \left\langle -g'(\bar{\alpha})\left(\frac{1}{2}\xi_{\mathbf{X}}(\bar{\mathbf{u}}) : \bar{\mathbb{C}} : \xi_{\mathbf{X}}(\bar{\mathbf{u}}) + \bar{\sigma} : \xi_{\mathbf{X}}(\bar{\mathbf{u}}) - \bar{\psi}\right) - \bar{G}\bar{\alpha} + \nabla_{\mathbf{X}} \cdot \left(\bar{\mathbb{D}}\nabla_{\mathbf{X}}\bar{\alpha}\right)\right\rangle_{+}, \quad (3.56)$$

which governs the macroscopic damage $\bar{\alpha}$. Applying the results of Theorem 3.3.2, 3.3.3 and Lemma 3.3.4 we can reduce the list of effective coefficients from ten to eight and further simplify several terms. Doing so allows us to write the list of effective coefficients as:

$$\begin{aligned}
\bar{\mathbb{C}} &= \{\mathbb{C}_A + \mathbb{C}_A : \xi_{\mathbf{Y}}(\chi_A)\}_{\Omega_A} + \sum_{\beta=1}^{N} \{\mathbb{C}_\beta + \mathbb{C}_\beta : \xi_{\mathbf{Y}}(\chi_\beta)\}_{\Omega_\beta}, \\
\bar{\sigma} &= \{\hat{\sigma}_A + \mathbb{C}_A : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A)\}_{\Omega_A} + \sum_{\beta=1}^{N} \{\hat{\sigma}_\beta + \mathbb{C}_\beta : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta)\}_{\Omega_\beta}, \\
\bar{\rho} &= \{\rho_A\}_{\Omega_A} + \sum_{\beta=1}^{N} \{\rho_\beta\}_{\Omega_\beta}, \quad \bar{\mathbf{B}} = \{\rho_A \mathbf{b}_A\}_{\Omega_A} + \sum_{\beta=1}^{N} \{\rho_\beta \mathbf{b}_\beta\}_{\Omega_\beta}, \\
\bar{\psi} &= \left\{\hat{\psi}_A + \frac{1}{2}\hat{\sigma}_A : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A)\right\}_{\Omega_A} + \sum_{\beta=1}^{N} \left\{\hat{\psi}_\beta + \frac{1}{2}\hat{\sigma}_\beta : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta)\right\}_{\Omega_\beta}, \\
\bar{\mathbb{D}} &= \{\mathbb{D}_A + \mathbb{D}_A \nabla_{\mathbf{Y}} \zeta_A\}_{\Omega_A} + \sum_{\beta=1}^{N} \{\mathbb{D}_\beta + \mathbb{D}_\beta \nabla_{\mathbf{Y}} \zeta_\beta\}_{\Omega_\beta}, \\
\bar{G} &= \{G_A\}_{\Omega_A} + \sum_{\beta=1}^{N} \{G_\beta\}_{\Omega_\beta}, \quad \bar{\eta} = \{\eta_A\}_{\Omega_A} + \sum_{\beta=1}^{N} \{\eta_\beta\}_{\Omega_\beta}.
\end{aligned} \quad (3.57)$$

Here, $\bar{\mathbb{C}}$ is the fourth order elasticity tensor, $\bar{\sigma}$ is the second order residual stress tensor, $\bar{\psi}$ is the effective damage threshold and $\bar{\mathbb{D}}$ is the second order effective diffusion tensor. The rest of the macroscopic coefficients $\bar{\rho}$, $\bar{\mathbf{B}}$, $\bar{G}$ and $\bar{\eta}$ are local integral averages. $\bar{\rho}$ is the macroscopic density, $\bar{\mathbf{B}}$ is the macroscopic body force, $\bar{G}$ is the macroscopic tearing energy and $\bar{\eta}$ is the macroscopic viscosity. As discussed earlier, since many of these effective coefficients are tensors, then in reality our system depends on many independent spatial parameters. We can simplify this by applying the conclusions of minor and major symmetries of $\bar{\mathbb{C}}$ as seen in Theorem 3.3.2. In two dimensions then our system would depend on twenty independent spatial parameters and in three dimensions this would be forty six spatial parameters.

However, the number of independent spatial parameters in the full three dimensional microscopic problem would depend on is $46k(N+1)$, where $k \in \mathbb{N}$ is the number of local cells and $N \in \mathbb{N}$ is the number of subdomains defining each local cell. As k and N get large this quickly becomes unfeasible to compute. Especially as the full microscopic problem would depend on $4k(N+1)$ coupled partial differential equations.

The effective coefficients in our multiscale model are dependent on a set of auxiliary variables. To calculate these auxiliary variables we must solve a series of local cell problems informed by the microscopic geometry and the associated physical parameters. These local cell problems have now been simplified in our model since we have simplified the degradation functions such that $p_A = p_\beta = p$ for all $\beta = 1, ..., N$. Therefore we have the first local cell problem defining the third order auxiliary tensors χ_A and χ_β,

$$\begin{aligned}
&\nabla_{\mathbf{Y}} \cdot \mathbb{C}_A = -\nabla_{\mathbf{Y}} \cdot [\mathbb{C}_A : \xi_{\mathbf{Y}}(\chi_A)] \text{ in } \Omega_A, \\
&\nabla_{\mathbf{Y}} \cdot \mathbb{C}_\beta = -\nabla_{\mathbf{Y}} \cdot [\mathbb{C}_\beta : \xi_{\mathbf{Y}}(\chi_\beta)] \text{ in } \Omega_\beta \, \forall \beta, \\
&(\mathbb{C}_A + \mathbb{C}_A : \xi_{\mathbf{Y}}(\chi_A)) \cdot \mathbf{n}_A = (\mathbb{C}_\beta + \mathbb{C}_\beta : \xi_{\mathbf{Y}}(\chi_\beta)) \cdot \mathbf{n}_A \text{ on } \Gamma_\beta \, \forall \beta, \\
&\chi_A = \chi_\beta \text{ on } \Gamma_\beta \, \forall \beta.
\end{aligned} \tag{3.58}$$

The next local cell problem defines the auxiliary vectors $\tilde{\mathbf{u}}_A$ and $\tilde{\mathbf{u}}_\beta$,

$$\begin{aligned}
&\nabla_{\mathbf{Y}} \cdot [\mathbb{C}_A : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A)] = -\nabla_{\mathbf{Y}} \cdot \sigma_A \text{ in } \Omega_A, \\
&\nabla_{\mathbf{Y}} \cdot [\mathbb{C}_\beta : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta)] = -\nabla_{\mathbf{Y}} \cdot \sigma_\beta \text{ in } \Omega_\beta \, \forall \beta, \\
&(\mathbb{C}_A : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_A) + \sigma_A) \cdot \mathbf{n}_A = (\mathbb{C}_\beta : \xi_{\mathbf{Y}}(\tilde{\mathbf{u}}_\beta) + \sigma_\beta) \cdot \mathbf{n}_A \text{ on } \Gamma_\beta \, \forall \beta, \\
&\tilde{\mathbf{u}}_A = \tilde{\mathbf{u}}_\beta \text{ on } \Gamma_\beta \, \forall \beta.
\end{aligned} \tag{3.59}$$

Our final local cell problem defines the auxiliary vectors ζ_A and ζ_β,

$$\begin{aligned}
&\nabla_{\mathbf{Y}} \cdot (\mathbb{D}_A + \mathbb{D}_A \nabla_{\mathbf{Y}} \zeta_A) = \mathbf{0} \text{ in } \Omega_A, \\
&\nabla_{\mathbf{Y}} \cdot (\mathbb{D}_\beta + \mathbb{D}_\beta \nabla_{\mathbf{Y}} \zeta_\beta) = \mathbf{0} \text{ in } \Omega_\beta \, \forall \beta, \\
&(\mathbb{D}_A + \mathbb{D}_A \nabla_{\mathbf{Y}} \zeta_A) \, \mathbf{n}_A = (\mathbb{D}_\beta + \mathbb{D}_\beta \nabla_{\mathbf{Y}} \zeta_\beta) \, \mathbf{n}_A \text{ on } \Gamma_\beta \, \forall \beta, \\
&\zeta_A = \zeta_\beta \text{ on } \Gamma_\beta \, \forall \beta.
\end{aligned} \tag{3.60}$$

All of the above defines our complete multiscale model of a damage phase field model on a linear elastic composite material with residual stress.

3.4.1 Concluding remarks

This chapter contains the main theoretical result of this thesis. We have derived the homogenised macroscopic model for the coupled mechanical behaviour of a composite material, incorporating the damage evolution into the model. To the best of the authors knowledge, this is the first time such a model has been derived, making it a novel result. It is also believed to be the first upscaling of a damage phase field model. Other authors have attempted to derive such a multiscale model of the damage phase field method but have not properly incorporated the microscopic contributions of damage evolution or the constraints of irreversibility and the damage criterion into their models [40, 81]. Instead,

they either ignored the constraints of irreversibility and the damage criterion, or they did not include the damage phase field in the upscaling whatsoever.

In the work by Fantoni et al. [40], they simply derived a multiscale model of the momentum balance with no coupled damage degradation or damage evolution, in fact there was no damage phase field incorporated in their multiscale derivation at all. Instead once a multiscale model of the momentum balance was obtained they then post hoc coupled a damage evolution equation to the momentum balance by multiplicatively decomposing the effective macroscopic elasticity. This totally ignores how we ensure that the microscopic irreversibility constraints and damage criterion's of the damage phase field are effectively upscaled. It also means that the macroscopic coefficients in the damage evolution equation will not be effective equations that encode any representative information about the microscale. Effectively they ignored any microscopic variations in the damage phase field.

In the work by Ma et al. [81], they performed an asymptotic homogenisation upscaling of a coupled momentum balance and damage evolution equation. This model is similar to our model in structure at the microscale. The issue with their model is that they simply ignored any requirement of damage irreversibility and a damage criterion at the microscale. These were again added post hoc after upscaling at the macroscale. This approach, unlike our model, lacks mathematical rigour as it neither accounts for the upscaling of the irreversibility constraint or the damage criterion.

Either of the discussed above modelling simplifications will not produce a model which truly encapsulates the full contribution of the damage phase field on the microscale. As a truly multiscale model, we have created an approach to modeling complex heterogeneous structures from small microscopic samples that accounts for all the necessary constraints required with the damage phase field method. This allows one to then study material failure on a macroscopic level. Using these new theoretical results researchers can take a non-phenomenological macroscopic approach to evaluating material failure due to microscopic variations in a material.

An interesting feature of our model are the non-linear terms present in the macroscopic model. In the macroscopic equation of motion, the degradation function is a non-linear polynomial function in terms of $\bar{\alpha}$. Similarly, in the macroscopic damage evolution equation, the elastic potential term and gradient of the degradation function are non-linear terms of the macroscopic displacement $\bar{\mathbf{u}}$ and damage $\bar{\alpha}$. This is interesting because the local cell problems are all linear. Meaning that calculation of the auxiliary variables and effective coefficients is fast and easy. Whereas, the more interesting non-linear features of the model behave on the macroscale.

One of the main advantages of a multiscale model is found in its computational efficiency. The calculation of the macroscopic coefficients and auxiliary variables need only be done once, as they have no time dependencies. As the microscale has been effectively

homogenised into the macroscopic model, instead of solving a computational model with many microscopic variations, we need only solve for a singular smooth homogenised continuum model. This means solving a numerical model with far fewer spatial nodes. Overall, this means saving lots of computational time and thus improving efficiency.

It should be noted the model is not perfect. Asymptotic homogenization relies on several assumptions. It would be a vast improvement if, in the future, we could derive a multiscale model that does not require an assumption of local periodicity. Further, it should be noted that we upscaled a linear elastic composite model. To improve upon this, a non-linear elastic composite model would allow one to model much more generally. Our current approximation uses an elastic potential that includes contribution from an inherent residual stress.

The overall objective of this thesis is to create a multiscale model of aortic dissections. In this chapter we have created a multiscale model of the damage phase field method for an arbitrary linear elastic composite material with residual stress. It is possible to model all three layers of the aorta as a linear elastic composite material with residual stress, as discussed in §1.3.2. One could therefore use our multiscale model as a good approximation for studying aortic dissections. In our case it would allow one to study how microscopic changes at each macroscopic point effects the outcome of a dissection. This can be done by studying changes in a local microscopic geometry or local physical properties. We would be able to therefore study a wide variety of biological conditions that effect the aortic wall and see in theory whether they contribute to a dissection or play in a role in determining the outcome of a dissection. First, though we need to consider how to solve this model numerically.

Chapter 4

Convergence of the multiscale model

In Chapter 3 we derived a novel multiscale model for material failure. The goal of this chapter is to test the validity of the model and explain how we can solve it numerically. By focusing on the one-dimensional rate independent case, we can test the case of a bar being stretched to the point of tearing. A one-dimensional reduction will also allow us to design a numerical method for approximating the solutions, satisfying the irreversibility constraints and damage criterion. Once we have an algorithm, we can explain how in one dimension we can solve each part of our model numerically. This will require creating a scheme for solving the linear momentum balance and returning a solution for the displacement, and a scheme for solving the damage energy equation and returning a solution for the damage. Each of these numerical schemes are implemented using finite difference approximations. Putting all of this together we will have a numerical method that we have implemented in MATLAB [72]. All code for the simulations in this chapter can be found in Appendix A. The final step in this chapter will be to perform a parametric analysis of our one-dimensional example, studying how varying each variable with respect to the scale separation, ϵ, effects the convergence of the microscopic and macroscopic models. As a result we will present numerical evidence of convergence of the microscopic model to the macroscopic model.

4.1 One dimensional test case

To create a numerical scheme for studying the convergence of our multiscale model we can reduce our problem to one dimension. It is convenient, easier and requires much less computational power than simulating a two or three dimensional test case. In fact, if our macroscopic model required 10^3 nodes and we consider a scale separation of $\epsilon = 0.01$ then we would require at least 10^5 nodes for the microscopic model. Evidently, this shows that one of the main advantages of our macroscopic model is saving significant computation time, because computing the macroscopic solution require significantly less nodes by at least two orders of magnitude compared to the microscopic solution. These computational

savings will compound as dimensions increase.

In this Chapter we will be interested in one simple but mechanically interesting example of material failure. Consider the case illustrated in Figure 4.1, where we have a one-dimensional bar clamped on the left boundary whilst being displaced on the right boundary. The displacement at the right boundary is $U = kt$ where k is a speed and t is time. This choice of boundary condition ensures, in one dimension, that at every time step we can apply a Dirichlet boundary condition which gives us increased control over our simulations. The reference state of the bar is the domain $\Omega := [0, 1]$. The bar could be a composite or homogeneous material. In our case we will consider a homogeneous domain and material properties with rapidly oscillating coefficients. This example will provide evidence that our macroscopic model converges with the respective full microscopic problem as $\epsilon \to 0$. A similar one dimensional case was studied in Pham et al. [83].

Figure 4.1: This is a schematic of a one dimensional bar. The bar is clamped at the left hand side, $X = 0$. On the other end at $X = 1$, the bar is being displaced. The displacement on the right hand side is defined as $U(1, t) = kt$, which is a linear displacement to the right with respect to time. Here k acts as a non-dimensional speed.

This mechanical problem will allow us to test the convergence between our microscopic and macroscopic models. We will test that the microscopic and macroscopic models do in fact converge as $\epsilon \to 0$. By varying the scaling of each microscopic coefficients we can test how this affects the error between between our microscopic and macroscopic models. This test will be performed using the rate independent version of the microscopic and macroscopic model. We do this because the rate dependency was only introduced in §2.2.4 as a feature to make numerical solutions more efficient. Recall that the rate independent model is the original form of the damage model derived in Chapter 2. Without any rate dependency in the damage evolution equations then damage will grow instantaneously to satisfy all the energy requirements. Therefore, showing the microscopic and macroscopic models converge in a rate independent environment is a strong demonstration of our multiscale convergence. In Chapter 5 we will consider the rate dependent damage model, because this helps guarantee numerical convergence when, in higher dimensions, we use the finite element method.

An advantageous detail about this test is that the the structure of partial differential equations defining microscopic model (2.46), for a homogeneous domain, is the same as

the macroscopic model defined by (3.55) and (3.56). Therefore, we can solve both models with the same numerical scheme. The only difference will being the coefficients used and the necessary resolution for spatial and temporal nodes to solve each model.

We are considering a single domain Ω to show the convergence of our multiscale model, where our microscopic coefficients can be spatially piecewise discontinuous or rapidly oscillating in Ω. This therefore mimics a composite material whilst simplifying the complexity of our numerical model, because the microscopic model will not require the interface conditions defined by (3.10). In the same domain we will be solving the same set of partial differential equations, for the microscopic and macroscopic model. The main difference between the models will be the coefficients: microscopic coefficients will be rapidly oscillating or piecewise discontinuous; the macroscopic coefficients will be calculated from the microscopic conditions. The other difference is that when we solve the microscopic model numerically this will require a significantly larger resolution than the macroscopic model, depending on the scale ratio ϵ.

In the rest of this section we will mathematically define the exact problems we are solving. Then we will explain in one dimension how to calculate the effective coefficients for the macroscopic problem.

4.1.1 Problem setup

We are not the first to attempt to solve this form of damage model numerically. Pham et al. [84] and Miehe et al. [90] used a strategy of numerical minimisation to solve the weak form of (4.1). We found this to be a simple methodology to implement with many numerical software packages already having inbuilt tools for numerical minimisation but it was quite inefficient, with one dimensional simulations of one hundred spatial and time nodes taking around four hours to complete. In this thesis we solve all of our numerical problems using the strong forms of the governing equations.

Now that we have a particular problem we directly formulate our one dimensional system of partial differential equations. As mentioned the macroscopic model defined by (3.55) and (3.56) are effectively the same as the microscopic model defined by (2.46). This is especially true as we are only considering a single domain material with rapidly oscillating material properties. Such a material with sharp variations in its coefficients at such a small scale is a great approximation of a composite material but far easier to solve numerically. As we are interested in the physical problem illustrated in Figure 4.1 then we can ignore all body forces in our problem. Since this problem is also meant to be the simplest case possible we assume that the one dimensional bar is isotropic and linear elastic without any residual stress. Reducing our problem to one dimension allows us to write our problem in terms of the one dimensional displacement in the horizontal direction $U(X,t)$ and the damage variable $\alpha(X,t)$. Here, $X \subset \Omega$ is our one dimensional spatial variable

and $t \subset \mathcal{T}$ is time. There is a final simplification that can be made here to make this the simplest test possible. That would be to drop the inertial terms present in the equation of linear momentum balance, effectively reducing the equation to an equilibrium equation. Since we have control of k it can always be taken small enough such that the inertial terms can be ignored, this would be the case if we were to pull the one dimensional bar very slowly. In Chapter 3, asymptotic homogenisation focused on upscaling and encoding the effects of spatial, and not temporal, variations in the material. Hence, we limit ourselves to the equilibrium equation, which is an elliptic partial differential equation and can be solved numerically using a variety of techniques.

Let us write down the microscopic and macroscopic models we want to solve numerically. For the microscopic model, we are interested in solving

$$\begin{aligned} &\frac{\partial}{\partial X}\left[g(\alpha)\left(\mathbb{C}\frac{\partial U}{\partial X}\right)\right] = 0, \\ &\left\langle -g'(\alpha)\left(\frac{1}{2}\mathbb{C}\left(\frac{\partial U}{\partial X}\right)^2 - \psi\right) - G\alpha + \frac{\partial}{\partial X}\left(\mathbb{D}\frac{\partial \alpha}{\partial X}\right)\right\rangle_+ = 0. \end{aligned} \tag{4.1}$$

Let us also reintroduce the coefficients and variables of the equation for convenience of the reader. The microscopic variables of interest are the one dimensional horizontal displacement $U(X,t)$ and the damage phase field $\alpha(X,t)$. Our microscopic equilibrium equation is dependent on the scalar elasticity function $\mathbb{C}(X)$. In the rate independent case we may call the damage evolution equation the damage energy equation. It depends on the unique microscopic scalar coefficients: $\psi(X)$ the positive semi-definite energy threshold for damage; $G(X)$ the tearing energy and $\mathbb{D}(X)$ the damage diffusion coefficient. In this one dimensional problem setup we have two equations, two unknowns and four parameters of interest. The equilibrium equation and damage energy equation are coupled by the damage degradation function. The damage degradation function was introduced in §2.2.2, it has the necessary constraints

$$g(0) = 1, \quad g(1) = 0, \quad g'(1) = 0 \quad \text{and} \quad g'(\alpha) \leq 0.$$

In Chapter 3, we restricted the specific form of our damage degradation function to a polynomial function such that

$$g(\alpha) = (1-\alpha)^p \quad \text{such that} \quad p \geq 2 \in \mathbb{N}. \tag{4.2}$$

To complete the microscopic model we require a set of boundary conditions. For the displacement field we have the Dirichlet boundary conditions, as discussed in Figure 4.1,

$$U(0,t) = 0 \quad \text{and} \quad U(1,t) = kt. \tag{4.3}$$

The damage phase field has a set of no flux boundary conditions which in one dimension can be written as

$$\mathbb{D}\frac{\partial \alpha}{\partial X}(0) = \mathbb{D}\frac{\partial \alpha}{\partial X}(1) = 0. \tag{4.4}$$

In a similar fashion let us write down the corresponding macroscopic model. For the macroscopic model, we are interested in solving

$$\begin{aligned} &\frac{\partial}{\partial X}\left[g(\bar{\alpha})\left(\bar{\mathbb{C}}\frac{\partial \bar{U}}{\partial X}\right)\right] = 0, \\ &\left\langle -g'(\bar{\alpha})\left(\frac{1}{2}\bar{\mathbb{C}}\left(\frac{\partial \bar{U}}{\partial X}\right)^2 - \bar{\psi}\right) - \bar{G}\bar{\alpha} + \frac{\partial}{\partial X}\left(\bar{\mathbb{D}}\frac{\partial \bar{\alpha}}{\partial X}\right)\right\rangle_+ = 0. \end{aligned} \tag{4.5}$$

The macroscopic unknowns of interest are $\bar{U}(X,t)$ and $\bar{\alpha}(X,t)$ the macroscopic one dimensional horizontal displacement and damage, respectively. Note that every parameter and variable with a bar is the macroscopic counterpart to the microscopic model. Also note here we are performing a clear abuse of notation, by setting X as both the spatial variable over the full microscopic domain and macroscopic domain. This does not cause any issues with our analysis in this chapter. The macroscopic coefficients in this model are as follows: $\bar{\mathbb{C}}(X)$ the scalar macroscopic elasticity; the scalar positive semidefinite macroscopic energy threshold $\bar{\psi}(X)$; the scalar macroscopic tearing energy $\bar{G}(X)$ and the scalar macroscopic damage diffusion $\bar{\mathbb{D}}(X)$. Again, in this one dimensional setup we have two unknowns, two equations and four scalar parameters. The damage degradation function couples the equilibrium equation and damage energy equation. To complete the macroscopic problem we need a set of boundary conditions. For the displacement field we have the Dirichlet boundary conditions, as discussed in Figure 4.1,

$$\bar{U}(0,t) = 0 \quad \text{and} \quad \bar{U}(1,t) = kt. \tag{4.6}$$

The macroscopic damage phase field has a set of no flux boundary conditions which in one dimension can be written as

$$\bar{\mathbb{D}}\frac{\partial \bar{\alpha}}{\partial X}(0) = \bar{\mathbb{D}}\frac{\partial \bar{\alpha}}{\partial X}(1) = 0. \tag{4.7}$$

As we have mentioned (4.1) and (4.5) are the same set of partial differential equations for two differing sets of of coefficients, and hence unknowns. They have the exact same set of boundary conditions for the displacements, (4.3) and (4.6). The damage no flux conditions are the same, (4.4) and (4.7), for both microscopic and macroscopic models only differing in terms of diffusion coefficients. Hence, we only need to develop one numerical scheme to solve both of these models, as we will do in this chapter.

The above system of equations describe both the macroscopic and microscopic models.

In a sense by solving both the macroscopic and microscopic models will allow us to show multiscale convergence such that

$$\lim_{\epsilon\to 0} U = \bar{U} + \mathcal{O}(\epsilon) \quad \text{and} \quad \lim_{\epsilon\to 0} \alpha = \bar{\alpha} + \mathcal{O}(\epsilon).$$

It should be noted that all the coefficients in this chapter have been reduced to one dimension. When performing our simulations the microscopic models coefficients will be provided and known. We can then calculate the macroscopic coefficients, using these microscopic coefficients, as defined in (3.57). This will require solving a set of one dimensional cell problem to then calculate the macroscopic coefficients. We can do this analytically in one dimension and we shall explain how to do this in the following subsection.

4.1.2 One dimensional effective coefficients

Our macroscopic model depends on a series of effective coefficients related to a series of microscopic coefficients. In one dimension we can actually calculate the effective coefficients analytically in terms of the microscopic coefficients. We are going to calculate the one dimensional effective coefficients in this section. It should be noted these are not novel results and in the field of asymptotic homogenisation they are considered classic results [103].

The macroscopic coefficients $\bar{\psi}$ and $\bar{G}$ are local integral averages, they can therefore already be directly calculated. We can write this in one dimension as

$$\bar{\psi}(X) = \frac{1}{P}\int_{\omega} \psi(X,Y)\,dY \quad \text{and} \quad \bar{G}(X) = \frac{1}{P}\int_{\omega} G(X,Y)\,dY, \tag{4.8}$$

where ψ and G are the microscopic coefficients, Y is the microscopic spatial variable and P is the length of the periodic cell ω. These averages can be calculated very easily with Matlab on the one dimensional local cell ω, defined to be $[0, P]$. The relation between the macroscopic and microscopic coefficients here are very clear and the integral averages can be seen to in a sense smooth out the microscopic variations.

The effective macroscopic coefficients $\bar{\mathbb{C}}$ and $\bar{\mathbb{D}}$ (3.57) depend on the solutions to the ansätze (3.58) and (3.60) defining χ and ζ, respectively, as seen in Chapter 3. In one dimension we can say that $\bar{\mathbb{C}}$ and $\bar{\mathbb{D}}$ are calculated as

$$\bar{\mathbb{C}} = \frac{1}{P}\int_{\omega} \left(\mathbb{C} + \mathbb{C}\frac{\partial \chi}{\partial Y}\right) dY \quad \text{and} \quad \bar{\mathbb{D}} = \frac{1}{P}\int_{\omega} \left(\mathbb{D} + \mathbb{D}\frac{\partial \zeta}{\partial Y}\right) dY. \tag{4.9}$$

Here $\mathbb{C}$ and $\mathbb{D}$ are the microscopic coefficients of elasticity and damage diffusivity. Clearly to calculate these effective coefficients we need to find the derivatives of χ and ζ. To find these derivatives we need to solve the local cell problems defined for χ and ζ defined

in (3.59) and (3.60), respectively. In one dimension for a homogenous domain Ω we can reduce these local cell problems into the following:

$$\frac{\partial}{\partial Y}\Big[\mathbb{C}\frac{\partial \chi}{\partial Y}\Big] = -\frac{\partial \mathbb{C}}{\partial Y} \quad \text{and} \quad \frac{\partial}{\partial Y}\Big[\mathbb{D}\frac{\partial \zeta}{\partial Y}\Big] = -\frac{\partial \mathbb{D}}{\partial Y}. \tag{4.10}$$

These differential problems can be solved by taking local periodicity on the cell boundary $\partial\omega$ into account. The periodicity conditions can be used by defining the boundary conditions

$$\chi(0) = \chi(P) \quad \text{and} \quad \zeta(0) = \zeta(P). \tag{4.11}$$

Using the periodicity conditions (4.11) we can calculate the general solutions of (4.10) for χ and ζ in one dimension as

$$\begin{aligned} \chi(Y) &= -Y + \frac{P}{\int_0^P \mathbb{C}^{-1}(Y)\,dY}\int_0^Y \frac{1}{\mathbb{C}(s)}ds + C_1, \\ \zeta(Y) &= -Y + \frac{P}{\int_0^P \mathbb{D}^{-1}(Y)\,dY}\int_0^Y \frac{1}{\mathbb{D}(s)}ds + C_2. \end{aligned} \tag{4.12}$$

Here C_1 and C_2 are constants of integration and s is a dummy variable for integration. We do not need these general solution but rather their derivatives. We can write these down, using (4.12), as

$$\frac{\partial \chi}{\partial Y} = -1 + \frac{P\,\mathbb{C}^{-1}(Y)}{\int_0^P \mathbb{C}^{-1}(Y)dY} \quad \text{and} \quad \frac{\partial \zeta}{\partial Y} = -1 + \frac{P\,\mathbb{D}^{-1}(Y)}{\int_0^P \mathbb{D}^{-1}(Y)dY}. \tag{4.13}$$

Substituting the derivatives (4.13) into our cell problems (4.9) allows us to calculate our effective coefficients of $\bar{\mathbb{C}}$ and $\bar{\mathbb{D}}$ in terms of the microscopic coefficients $\mathbb{C}$ and $\mathbb{D}$. Doing so leaves us with

$$\bar{\mathbb{C}}(X) = \frac{P}{\int_0^P \mathbb{C}^{-1}(X,Y)dY} \quad \text{and} \quad \bar{\mathbb{D}}(X) = \frac{P}{\int_0^P \mathbb{D}^{-1}(X,Y)dY}. \tag{4.14}$$

We can see this is clearly not just an integral average and the effective coefficients $\bar{\mathbb{C}}$ and $\bar{\mathbb{D}}$ are encoding microscopic information. In fact the one dimensional effective coefficients (4.14) are a pair of harmonic means for the microscopic coefficients, $\mathbb{C}$ and $\mathbb{D}$, over the local cell ω.

Now we have defined all of our one dimensional macroscopic coefficients in terms of the microscopic coefficients as seen in (4.8) and (4.14). Therefore if we find a numerical solution for the one dimensional partial differential equations (4.1) we can use that solution to solve a microscopic problem and a macroscopic problem. We can therefore compare the solutions of these problems and show that the microscopic and macroscopic solutions converge as $\epsilon \to 0$.

Remark. *In this chapter we are omitting the macroscopic residual stress $\bar{\sigma}$ but we should note that it can also be calculated in terms of the microscopic coefficients $\mathbb{C}$ and σ. Here we will briefly comment on this solution because it is somewhat interesting compared to the macroscopic coefficients $\bar{\mathbb{C}}$ and $\bar{\mathbb{D}}$ (4.14). In one dimension we can define the macroscopic residual stress, defined in (3.57), as*

$$\bar{\sigma} = \frac{1}{P}\int_{\omega}\left(\sigma + \mathbb{C}\frac{\partial \tilde{U}}{\partial Y}\right)\, dY. \tag{4.15}$$

To calculate $\bar{\sigma}$ we must first calculate the one dimensional ansatz $\tilde{U}$. This can be done by solving the following one dimensional cell problem, reduced from (3.59),

$$\frac{\partial}{\partial Y}\left(\mathbb{C}\frac{\partial \tilde{U}}{\partial Y}\right) = -\frac{\partial \sigma}{\partial Y} \;\; in\ \omega.$$

Here our differential problem is defined on ω, our one dimensional local periodic cell $[0, P]$. The periodicity condition is defined as

$$\tilde{U}(0) = \tilde{U}(P).$$

Using this periodicity condition we can write down the general solution of the local cell problem as

$$\tilde{U} = -\int_0^Y \frac{\sigma(s)}{\mathbb{C}(s)}\, ds + \frac{\int_0^P \sigma(Y)\,\mathbb{C}^{-1}(Y)\, dY}{\int_0^P \mathbb{C}^{-1}(Y)\, dY}\int_0^Y \mathbb{C}^{-1}(s)\, ds + C_3,$$

where C_3 is a constant of integration. To calculate (4.15) we do not need the general solution of $\tilde{U}$ but rather its derivative with respect to the microscopic spatial variable Y. This derivative can be written down as

$$\frac{\partial \tilde{U}}{\partial Y} = -\frac{\sigma}{\mathbb{C}} + \frac{\int_0^P \sigma(X,Y)\mathbb{C}^{-1}(X,Y)\, dY}{\int_0^{\omega} \mathbb{C}^{-1}(X,Y)\, dY}\,\mathbb{C}^{-1}(Y),$$

substituting this result into (4.15) gives us the desired solution

$$\bar{\sigma}(X) = \frac{\int_0^P \sigma(s)\,\mathbb{C}^{-1}(X,Y)\, dY}{\int_0^P \mathbb{C}^{-1}(X,Y)\, dY}. \tag{4.16}$$

What makes this so interesting in comparison to $\bar{\mathbb{C}}$ and $\bar{\mathbb{D}}$ is that they are clearly harmonic means as seen in (4.14). Whereas, $\bar{\sigma}$ is a weighted mean as we can see in (4.16), in fact it is weighted by the residual of the local elasticity. It is very interesting to question why the residual stress has its own unique solution in one dimension in comparison to the other

effective coefficients.

4.2 One dimensional numerical solution

In this section we introduce a numerical solution for our damage model. This numerical solution guarantees that all of our modelling constraints are met. We will need to satisfy damage irreversibility and ensure that damage growth only occurs when the damage criterion is met. The damage criterion is defined by the terms inside the Macaulay brackets of (4.1) such that we have damage growth at any point that satisfies

$$-g'(\alpha)\left(\frac{1}{2}\bar{\mathbb{C}}\left(\frac{\partial U}{\partial X}\right)^2-\bar{\psi}\right)-\bar{G}\alpha+\frac{\partial}{\partial X}\left(\bar{\mathbb{D}}\frac{\partial \alpha}{\partial X}\right)>0. \tag{4.17}$$

To the best of this author's knowledge all numerical methods used in the literature are iterative in the following sense: we would iterate between solving for the displacement and solving for the damage field. This allows us to correctly calculate the damage field, the displacement field, satisfy the damage criterion and irreversibility constraint. To do this we assume that $\alpha(X,t)$ is fixed at time t then we can use it to solve the displacement immediately at time t. We can then take that solution for the displacement substitute it into the damage criterion and find which points have damage growth. With that information we can solve the damage energy equation (4.1) at every point where damage growth is occurring and every point can remain constant. Then we repeat the process over using this new updated solution for the damage phase field α at time t.

This is an iterative scheme where we will solve the displacement problem, then use the damage criterion to find out where we can apply the damage energy equation. By repeating this process until α converges to an appropriate tolerance that we control. We choose to use finite differences to approach this problem. A great resource for finite differences is the textbook Numerical recipes [107]. Our numerical method was fully implemented in Matlab and can be found in Appendix A.

The reason we are using separate numerical schemes to solve for the displacement and damage phase field at each time step instead of solving both fields at each time step (with a monolithic numerical scheme) is because our methodology saves us a lot of time. We found that solving every field monolithically at every time step with an iterative Newton-Raphson scheme was incredibly slow. For comparison solving a simple damage phase field model, as defined by (2.6), for 1000 evenly spaced nodes over 1000 time steps with the monolith method took around four hours and with our separation method we found the same problem took around 7 minutes.

4.2.1 Numerical algorithm

To introduce a numerical algorithm we need to break our problem down into three distinct parts for every time step. Then we need to structure how each part interacts with one another and how we flow through the algorithm. Let us begin by breaking our problem into smaller simpler parts. In Figure 4.1 we begin with a bar in a reference configuration where all initial information is known. When we stretch the bar we displace the right hand side by $U(1) = k\delta t$, where δt is a small positive forward time step. If we assume that there is no damage growth then we can easily calculate the solution to the equilibrium equation. Let us call this part of the problem the elastic phase where we are assuming we are in some configuration with no damage growth, just a simple linear elastic problem. To check whether this is true or not we substitute our solution directly into the damage criterion (4.17). At this point there are two distinct possibilities: the damage criterion is satisfied everywhere and we have no damage growth; the damage criterion is not satisfied everywhere and we must have damage growth.

If in the first case we have satisfied the damage criterion everywhere then we can move forward onto the next time step and repeat the whole process. However, if we have not satisfied the damage criterion then we need to solve the damage energy equation. To do this we can define a field called the criterion field, we establish the field such that at every point the damage criterion is not satisfied the criterion field is 1 and at every other point it is 0. Using this field we can solve the damage energy equation assuming that the damage equation criterion only applies at every point where the criterion field is 1 and constant at every other point where the criterion field is 0. To solve the damage energy equation we will have to use a iterative scheme because the problem is non-linear. Once we calculate the damage phase field we can recalculate the displacement starting the process again. Once the damage phase field converges using this method we can say we have solved this time step and move onto the next one.

This logic is the basis of our algorithm and now we need to write it down more formally with pseudocode. Before we do this let us introduce some useful notation and language so that the algorithm is clear. Firstly we split the algorithm into calculating the solution to three distinct partial differential equations. The first of which we call the elastic phase, which is our equilibrium problem. In the elastic phase we solve for the displacement U using the assumed damage phase field α and the current time t which informs the boundary condition. Next we calculate the criterion, the damage criterion, we do this by using our solution for U and α, the solution to this criterion is a field which tells us where the damage criterion is and is not satisfied. Finally we solve the damage phase, which is our damage energy equation. We solve the damage energy equation using our current solution for the displacement U, the time step δt and the criterion which informs where the damage energy equation is assumed to be valid and where damage stays constant. The structure

of this algorithm can be seen in Algorithm 1. It should be noted that similar numerical algorithms where proposed in the works of Pham et al. [84] and Miehe et al. [90].

Algorithm 1 Algorithm for calculating the damage and displacement at each time step.

$t = t_0 + \delta t$
while $t \leq T$ **do**
  $U^{(k)} = \text{ElasticPhase}(\alpha^{(k-1)}, t)$
  $\text{Crit} = \text{Criterion}(U^{(k)}, \alpha^{(k-1)})$
  if any Crit ≥ 0 **then**
    $\alpha^{(p)} =$ "guess"
    $\alpha^{(p-1)} = \alpha^{(k-1)}$
    while $\text{Norm}\big(\alpha^{(p)} - \alpha^{(p-1)}\big) \geq$ tolerance **do**
      $\alpha^{(p-1)} \leftarrow \alpha^{(p)}$
      $\alpha^{(p)} = \text{DamagePhase}(U^{(k)}, \text{Crit}, \delta t)$
      $U^{(k)} = \text{ElasticPhase}(\alpha^{(p)}, t)$
      $\text{Crit} = \text{Criterion}(U^{(k)}, \alpha^{(p)})$
    end while
    $\alpha^{(k)} \leftarrow \alpha^{(p)}$
  end if
  $t = t + \delta t$
end while

In this algorithm our iterative scheme for finding the correct damage phase field runs till a desired tolerance is met. The tolerance should be appropriately set such that its smaller than the order of the implemented finite difference schemes. In our implementation we usually set a base tolerance of 10^{-8}. The code we created that makes use of this algorithm can be found in §A.2.

4.2.2 The elastic problem

The elastic problem we solve in Algorithm 1 is our linear momentum balance equation (4.1). Since we assume that the damage phase field is a known constant field $\alpha(X, t)$ then we can write down a solution of the equilibrium equation. Doing so we can say in general that the solution for the displacement is

$$U(X, t; \alpha) = U_L + (U_R - U_L)\, \frac{\int_0^X \left(g(\alpha)\mathbb{C}(s)\right)^{-1} ds}{\int_0^L \left(g(\alpha)\mathbb{C}(X)\right)^{-1} dX}. \tag{4.18}$$

Here U_L and U_R are the boundary conditions in one dimension at the left and right most bounds, respectively. In one dimension we can see that $\mathbb{C}$ is a stiffness defined by Hooke's law, with units of force per unit length. As we solved this generally we are assuming that the domain we worked in is $[0, L]$, where L is just some length of a one dimensional domain. To calculate the displacement here we have two options: if $\mathbb{C}$ is appropriate then we directly calculate the integrals in (4.18); if we cannot calculate the

integrals analytically then we can solve these them in one dimension with simple numerical integration. Importantly when we solve our damage energy equation (4.1) we only need the gradient of displacement. We can save ourselves some work then by directly calculating the displacement gradient. Therefore,

$$\frac{\partial U}{\partial X} = (U_R - U_L) \frac{(g(\alpha)\mathbb{C}(X))^{-1}}{\int_0^L (g(\alpha)\mathbb{C}(X))^{-1} \, dX}. \tag{4.19}$$

In our particular case, depicted in Figure 4.1, we are interested in a domain defined across $[0, 1]$. We have a set of boundary conditions defined by (4.3), so we can plug those in directly. Doing this we have the particular solution for the displacement field

$$U(X, t; \alpha) = kt \frac{\int_0^X (g(\alpha)\mathbb{C}(X))^{-1} \, ds}{\int_0^1 (g(\alpha)\mathbb{C}(X))^{-1} \, dX}. \tag{4.20}$$

Of course we can extend this solution directly onto the displacement gradient. Which we write as

$$\frac{\partial U}{\partial X} = kt \frac{(g(\alpha)\mathbb{C}(s))^{-1}}{\int_0^1 (g(\alpha)\mathbb{C}(X))^{-1} \, dX}. \tag{4.21}$$

These solutions can be directly implemented in software for some discretized domain. We reiterate that here we are assuming that α is a known constant field because this solution is part of an iterative process, as illustrated in Algorithm 1. These solutions were implemented in code and can be found in §A.3.

4.2.3 The damage criterion

The one dimensional damage criterion (4.17) tells us in our iterative process where damage growth is taking place. So after calculating the displacement field, we must check whether our current displacement field and damage field satisfy the damage criterion. Therefore what we are calculating is the criterion field we call "Crit". This is done such that such that Crit $= 1$ anywhere the damage field is not satisfied and Crit $= 0$ anywhere the damage field is satisfied. In other words we are defining a piecewise function such that

$$\text{Crit}(X) = \begin{cases} 1, & F > 0 \\ 0, & F \leq 0 \end{cases} \tag{4.22}$$

where F is shorthand notation for the damage criterion (4.17). We can write this as

$$F(U, \alpha) = -g'(\alpha) \left(\frac{1}{2} \bar{\mathbb{C}} \left(\frac{\partial U}{\partial X} \right)^2 - \bar{\psi} \right) - \bar{G}\alpha + \frac{\partial}{\partial X} \left(\bar{\mathbb{D}} \frac{\partial \alpha}{\partial X} \right).$$

To implement this in software one can easily discretize $\mathrm{Crit}(X)$ and F then substituting in our assumed solutions of U and α will return a solution. In a discretized solution using finite differences, Crit will simply be a long vector of 1's and 0's, essentially a binary string that will inform our iterative scheme, in Algorithm 1, where we have to calculate a new solution for α. This method can also easily be extended to higher dimensions for some discretized domain. For one dimension we implemented this method in code, which can be found in §A.4.

4.2.4 The damage problem

The damage energy equation (4.1) is the most complicated part of our numerical algorithm. To solve it, special consideration must be given to the non-linear terms and the Macaulay brackets which induce a criterion for damage growth. Using the function $\mathrm{Crit}(X)$ (4.22), we know where in our iterative scheme to implement the damage energy equation. In other words we have to iteratively solve two problems depending of the value of $\mathrm{Crit}(X)$ on that region, whilst also satisfying the boundary conditions.

Since $\mathrm{Crit}(X)$ and $U(X)$ are fixed when we solve the damage energy equation in this iterative scheme it is a fairly straight forward process. To simplify the notation we define Q as the driving energy behind a tear. We define this term as

$$Q(U;\mathbb{C},\psi) = \frac{1}{2}\mathbb{C}\left(\frac{\partial U}{\partial X}\right)^2 - \psi(X). \tag{4.23}$$

Recall that ψ is a positive semidefinite scalar function and its physical role is to act as the energy threshold for when a materially can initially be damaged. Therefore, we can generally say that for a growth of damage we require $Q > 0$, but we will never have any growth of damage if $Q < 0$. The role of the driving energy will become clear by writing the problem down. The problem we are solving can be expressed as

$$\begin{aligned}
&\dot{\alpha} = 0 \quad \text{where } \mathrm{Crit}(X) = 0,\\
&-g'(\alpha)Q(U;\mathbb{C},\psi) - G\alpha + \frac{\partial}{\partial X}\left(\bar{\mathbb{D}}\frac{\partial \alpha}{\partial X}\right) = 0 \quad \text{where } \mathrm{Crit}(X) = 1,\\
&\mathbb{D}\frac{\partial \alpha}{\partial X}(0) = \mathbb{D}\frac{\partial \alpha}{\partial X}(1) = 0 \quad \text{on the boundaries.}
\end{aligned} \tag{4.24}$$

Numerically this is a very straight forward problem, we can think of it as a classic non-linear problem that can be solved with Newton's method in one dimension [107]. It is generally non-linear because of the degradation function (which is a polynomial function of lowest order two) and hence $g'(\alpha)$ is a non-linear polynomial for every case except the lowest order case of the degradation function. For our numerical scheme we want to implement Newtons method to solve (4.24) at every discrete point where $\mathrm{Crit}(X) = 1$ and

simply implement that α remains constant at every other point. The boundary conditions in (4.24) can be accounted for by making use of fictitious nodes at the boundaries. We implemented our solution to (4.24) with the code that can be found in §A.5.

An important point of consideration for the numerical scheme described by (4.24) is what boundary conditions do we apply for every damage growth region contained entirely within the interior of Ω. How we deal with this, as it is implicitly described in (4.24), is to apply the fixed values of α on the boundaries of these regions as a Dirichlet boundary condition to the damage energy equation. In one dimension this methodology works really well in our numerical scheme. However, in Chapter 5 we will point out that we must allow for numerical diffusion throughout Ω to facilitate numerical convergence.

Remark. *The rate independent problem defined in* (4.24) *is solved with Newton's method. If this were the rate dependent case of the model as defined in* (3.56)*, then we would need to take time into account. To do this we found using the Crank-Nicholson method [107] to be extremely effective. In fact the Crank-Nicholson scheme for the rate dependent case is much more computationally robust and runs a lot faster in comparison to the rate independent case. Therefore, for practical purposes we would recommend using the rate dependent model with a Crank-Nicholson scheme, as will be seen in Chapter 5. However, we should be aware that when the viscosity* $\eta \to 0$ *then the Crank-Nicholson scheme requires a much larger number of time steps to ensure convergence of the numerical scheme.*

4.2.5 Convergence of the numerical scheme

Now that we have a numerical scheme as described above, one of the first things we should do is reassure ourselves that the numerical scheme does converge to a consistent solution. To do this we are going to consider a test case with a set of coefficients, as defined by the damage phase field problem (4.1) with the boundary conditions (4.3) and (4.4). The test case we will consider is illustrated in Figure 4.1, the stretching of a one dimensional bar on its right boundary whilst it is clamped, stationary, on its left boundary. To check that our numerical scheme does converge we will solve the same problem multiple times with an increasing resolution each time. If our numerical scheme is consistent then as our resolution increases our solutions will converge.

As we are only testing the convergence of our numerical scheme then we can define simple constant coefficients. They are defined as

$$\mathbb{C}(X) = 1, \quad \psi(X) = 1 + 0.9\cos(2\pi X), \qquad G(X) = 1, \quad \text{and} \quad \mathbb{D}(X) = 10^{-4}. \tag{4.25}$$

Our choice of ψ was made to ensure that our one dimensional bar in Figure 4.1 will have a weak point at the centre of the bar where ψ is at its minimum. The final constitutive choice we must make before we can employ our numerical scheme is the choice of degradation

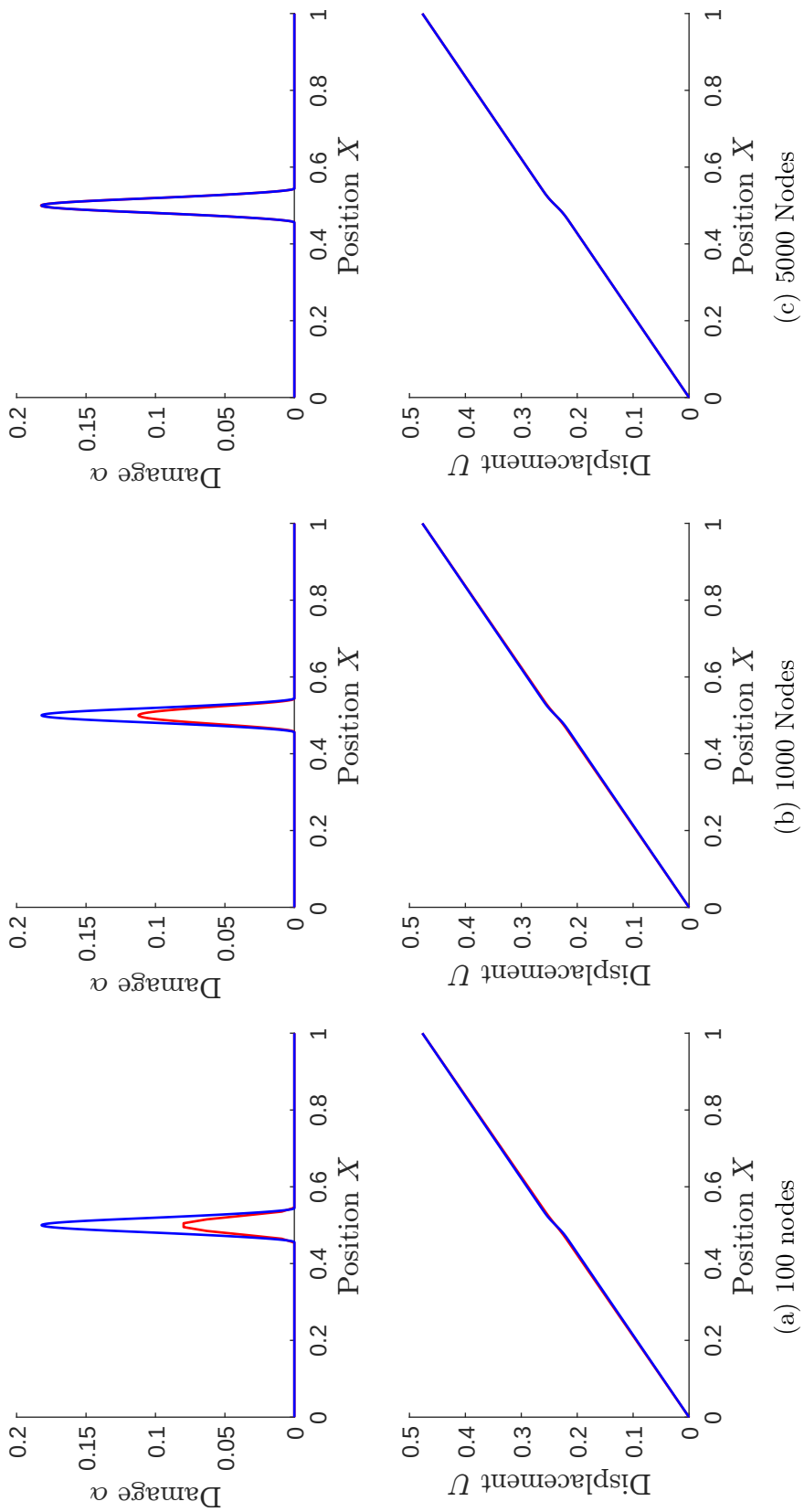
function. As discussed in Chapter 2 the damage degradation function is constrained to ensure physical realism, see §2.2.2. In Chapter 3 we limited our choice of degradation function to a polynomial form (4.2). Here we will use the simplest option, which is

$$g(\alpha) = (1-\alpha)^2. \tag{4.26}$$

An important thing to note is when our numerical scheme terminates. It will terminate if either of the following two events occur. The first event being our numerical scheme reaches the final time step. For example, if we have that initial time is t_0 and the final time t_f then we terminate when we calculate the time step where $t = t_f$. The other situation that causes our numerical scheme to terminate, in one dimension, is if we have a tear in our domain Ω. In other words if we have a point $X_{\mathcal{T}} \in \Omega$ such that $\alpha(X_{\mathcal{T}}) = 1$. If this happens then the domain Ω is essentially split in two pieces (or more points if multiple tears occur simultaneously). Hence, we terminate the numerical scheme when a tear occurs.

Now the first simple check we may do to see if our numerical scheme is converging is to solve the above described damage phase field problem for increasing mesh resolution. In each case our mesh will be defined by evenly spaced nodes on the domain $\Omega := [0, 1]$ starting from a low resolution of 100 nodes up to the high resolution of $10,000$ nodes. If these numerical solutions are converging then our numerical scheme is in fact consistent. As can be seen in Figure 4.2 our simulations are in fact converging towards the same solution as resolution increases. Another check we can do to ensure that our solutions are consistent as our mesh resolution increases is to see at what time a tear occurs. A tear here being of course when at some point $X \in \Omega$ we have that $\alpha(X) = 1$. These times are displayed in Figure 4.3 and it is clear that as the mesh resolution increases the final tearing times do converge. Another indication that our numerical scheme is consistent.

In conclusion we can say it is clear that our numerical scheme is converging as the mesh resolution increases, and is therefore a consistent numerical scheme. With this reassurance we can now consider checking the convergence of our multiscale model (4.5) with the full microscopic problem (4.1). This analysis is performed in the next section.



(a) 100 nodes (b) 1000 Nodes (c) 5000 Nodes

Figure 4.2: Plots comparing the displacement and damage phase field solutions of low and high resolution meshes, in red and blue respectively. Here we are simulating a one dimensional bar as described by the governing equations (4.1) with the Dirichlet boundary condition defined by $k = 1$, $t = 0.4768$ on the right boundary. The problem coefficients are as described in (4.25). In these simulations the blue curve is the same in all three cases representing the simulated solution found when using a mesh of 10,000 nodes, the red curves are of lower resolution: (a) 100 nodes, (b) 1000 nodes and (c) 5000 nodes. We can see that as the resolution of the red curves increase they converge with the blue curve. Note that the displacement fields illustrate much better agreement for lower resolutions than the damage phase field.

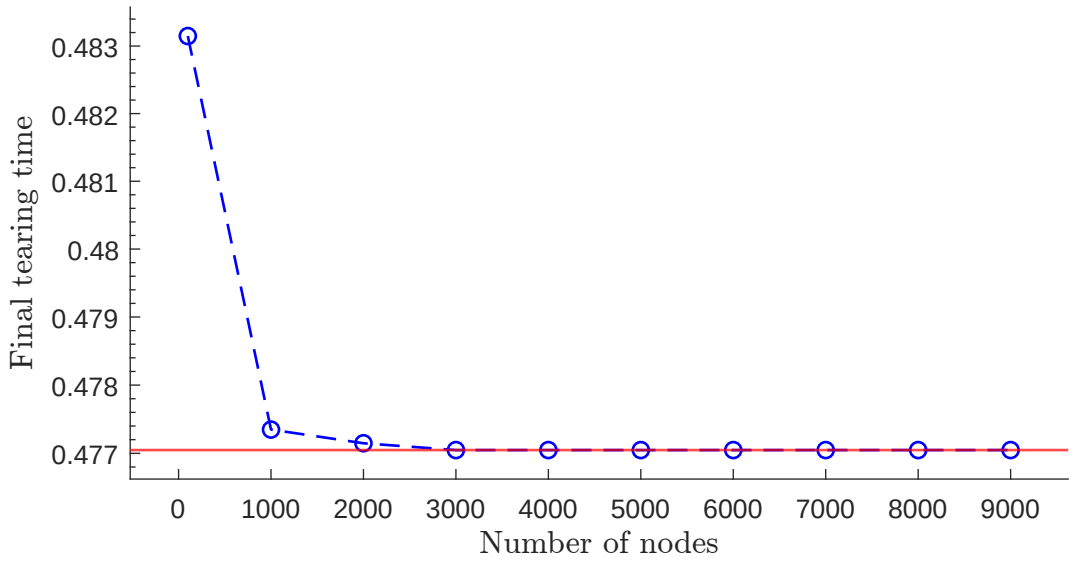


Figure 4.3: A plot of the final tearing times, when $\alpha = 1$, for a simulation of a one dimensional bar being stretched till it breaks, as described by the governing equations (4.1), with the boundary condition $k = 1$ on the right hand side and the microscopic coefficients are as described in (4.25). The red line in the above graph demonstrates the final tearing time for the high resolution simulation of a 10,000 nodes. The blue circles represent our low resolution simulations. In these simulations we can see that the final tearing times of the simulations converge as mesh resolution increases.

4.3 Parametric analysis

As we have explained in §4.2 how one can solve our microscopic and macroscopic damage models, then we can now illustrate that the models do converge. To do this we are going to consider a test case with a set of microscopic coefficients. Then by varying each microscopic coefficient one by one we can analyse how well the microscopic and macroscopic solutions converge as ϵ decreases. This is done by implementing the above numerical methods in Matlab using finite difference methods as explained.

The test case we will consider is illustrated in Figure 4.1 governed by the system of microscopic equations (4.1), with the boundary conditions (4.3) and (4.4). The microscopic coefficients we will vary around will be defined by

$$\begin{aligned} \mathbb{C}(X) = \frac{1}{1 + A\cos\left(2\pi\frac{X}{\epsilon}\right)}, \quad \psi(X) = 1 + B\cos(2\pi X), \\ G(X) = C, \quad \text{and} \quad \mathbb{D}(X) = D. \end{aligned} \tag{4.27}$$

Here A and B are constants that let us vary the microscopic coefficients $\mathbb{C}$ and ψ. These microscopic coefficients are locally periodic on a one dimensional cell with the general domain $[0, \epsilon]$, or in microscopic coordinates Y a domain defined by $[0, 1]$.

Our corresponding macroscopic system of equations is defined by (4.5), with macroscopic boundary conditions (4.6) and (4.7). We can directly calculate the macroscopic coefficients in terms of the microscopic coefficients (4.27) as discussed in §4.1.2. By remembering that $Y = X/\epsilon$, then we can write down the macroscopic coefficients

$$\begin{aligned} \bar{\mathbb{C}}(X) = 1, \quad \bar{\psi}(X) = 1 + B\cos(2\pi X), \\ \bar{G}(X) = C, \quad \text{and} \quad \bar{\mathbb{D}}(X) = D. \end{aligned} \tag{4.28}$$

Note that G and $\mathbb{D}$ in this example have no microscopic dependency and are thus equivalent to their corresponding macroscopic coefficients. This setup will perfectly illustrate both the pros and cons of our multiscale model. The function ψ was chosen so that our one dimensional bar in Figure 4.1 will have a weak point at the centre of the bar where ψ is at its minimum. It should be noted that if we performed a simulation where all the physical properties are uniform then we would have a uniform displacement and damage field. Such a scenario would be highly unrealistic, since it would be like assuming some material in nature is truly perfect and uniform in every way and furthermore when the damage phase field in that scenario grows to the point where $\alpha = 1$, this would happen simultaneously at every point in our system.

The final constitutive choice we must make for our parametric analysis is the form of the damage degradation function. As discussed in Chapter 2 the damage degradation function must satisfy a number of constraints, see §2.2.2. Furthermore our multiscale model as defined in Chapter 3 is limited to a polynomial form (4.2) for the degradation function. We therefore select the simplest option, which is

$$g(\alpha) = (1 - \alpha)^2 . \tag{4.29}$$

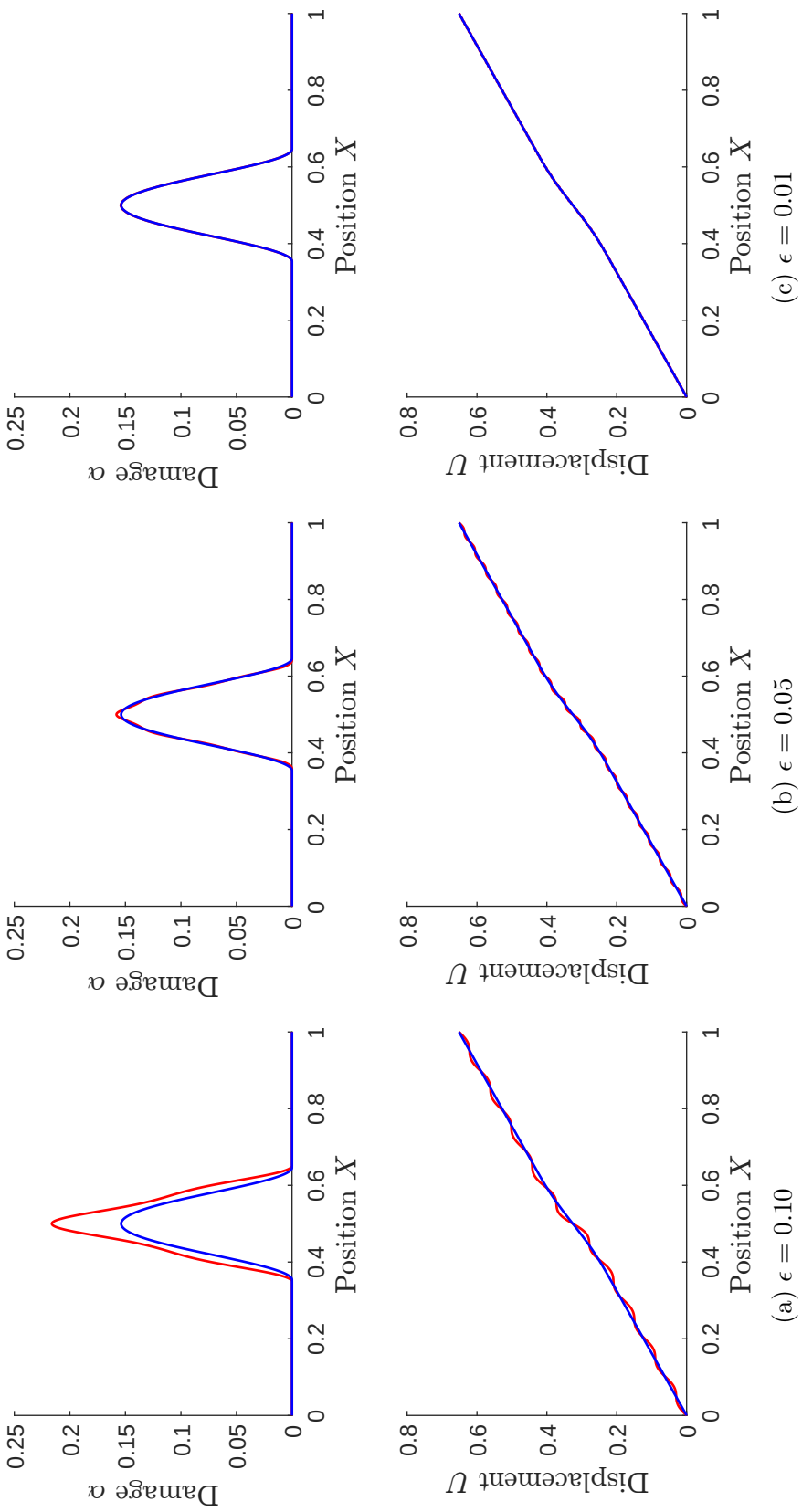
With this defined we can now perform all of our simulations.

An important thing to note is when our numerical scheme terminates. It will terminate if either of the following two events occur. The first event being our numerical scheme reaches the final time step. For example, if we have that initial time is t_0 and the final time t_f then we terminate when we calculate the time step where $t = t_f$. The other situation that causes our numerical scheme to terminate, in one dimension, is if we have a tear in our domain Ω. In other words if we have a point $X_{\mathcal{T}} \in \Omega$ such that $\alpha(X_{\mathcal{T}}) = 1$. If this happens then the domain Ω is essentially split in two pieces (or more points if multiple tears occur simultaneously). Hence, we terminate the numerical scheme when a tear occurs.

4.3.1 Varying the elasticity

To begin we consider varying the microscopic elasticity $\mathbb{C}(X)$. As defined in (4.27) we can see that the constant A must lie in the domain $[0, 1)$. At $A = 0$ we clearly have $\mathbb{C} = \bar{\mathbb{C}}$ but as A increases $\mathbb{C}$ becomes a rapidly oscillating function with larger oscillations. In fact at $A = 0.99$ it can be shown that $\mathbb{C}$ would oscillate roughly between the lower limit 0.5 and the upper limit 100, with a period of ϵ. The macroscopic counter-part, $\bar{\mathbb{C}}$, in this case is a constant equal to 1. According to our multiscale model this macroscopic effective coefficient is adequate for approximating the rapidly oscillating microscopic coefficient $\mathbb{C}$, as long as we have a large enough scale separation such that $\epsilon \ll 1$. To illustrate then how well our macroscopic and microscopic models converge let us consider the examples illustrated in Figure 4.4.

Here we are considering the case where $A = 0.9$ and decreasing ϵ for each case. We can clearly see that for $\epsilon = 0.1$ we have good agreement for the displacement. However, this is not the case for the damage, they disagree, especially at the maximum centred around the point $X = 0.5$. The convergence between the microscopic and macroscopic damage does improve in the case $\epsilon = 0.05$ and even more so for $\epsilon = 0.01$, as seen in Figure 4.4. In each case as ϵ decreases we can see that the agreement, for both fields of the microscopic and macroscopic models, improves drastically. In other words we can see a clear improvement in the convergence as $\epsilon \to 0$. In our research we have of course tested our one dimensional simulation for a wide variety of scenarios and every time we have found that the microscopic and macroscopic models converge as $\epsilon \to 0$. Even more interestingly, we found that the macroscopic effective coefficient $\bar{\mathbb{C}} = 1$ accurately approximates all the variations of A in the microscopic coefficient $\mathbb{C}$ as described in (4.27). The convergence of our multiscale modelling works as expected no matter how large the oscillations of $\mathbb{C}$ are. As long as the oscillations are occurring rapidly then we find in general great convergence, because rapid oscillations describe $\epsilon \to 0$. If we did these simulations for a composite material, instead of considering rapid oscillations we would consider very small local cells, equivalent to an extremely small scale ratio ϵ.



(a) $\epsilon = 0.10$ (b) $\epsilon = 0.05$ (c) $\epsilon = 0.01$

Figure 4.4: Plots of the macroscopic and microscopic displacement and damage phase fields, in blue and red respectively. Here we are simulating a one dimensional bar as described by the governing equations (4.1) with the Dirichlet boundary condition defined by $k = 1$, $t = 0.6514$ on the right hand side. The microscopic coefficients are set-up with $A = 0.9$, $B = 0.9$, $C = 1$ and $D = 0.01$ as described in (4.27) and the macroscopic coefficients are as described in (4.28). In these simulations the blue macroscopic curve is the same in all three cases and we can see that the models converge as $\epsilon \to 0$.

Another, more direct way to test our convergence is to consider the mean absolute error at every single time step. At every time step we can simply calculate the mean absolute error between the macroscopic and microscopic models. We define the mean absolute error (MAE) between two fields $f(X,t)$ and $g(X,t)$ on a discretised domain as

$$\mathrm{MAE}(t) = \frac{1}{N}\sum_{i=1}^{N} \left| f(X_i,t) - g(X_i,t) \right|, \tag{4.30}$$

where N is the number of nodes. It should be noted that the mean absolute error defined by (4.30) is analogous to the L^1 norm. Using (4.30) we produce the plots shown in Figure 4.5. Here it can be seen that in each case our models do in fact agree really well across all time steps. As damage grows so does this error, which happens for a mixture of reasons. Firstly, the error will compound with every time step, which can be dealt with by a mixture of increasing spatial nodes and time steps. Secondly, in the model itself damage growth can compound exponentially. The reason for this can be seen in the damage criterion (4.17). As α increases at each point $X \in \Omega$, at that same point the damage criterion itself degrades. In other words it takes less elastic potential energy to cross the energetic threshold for damage and to cause the growth of α. Therefore, if damage grows a little faster or a little slower in the microscopic models this effect can compound leading to a divergence as we move through each time step. However, we can see that the mean absolute error between microscopic and macroscopic models is decreasing as ϵ decreases. At the final time steps, however, we do see a rapid growth in the mean absolute error. This is because damage grows most rapidly when α is closest to 1. Therefore the error compounds exponentially around $\alpha \approx 1$. This problem can again be resolved by increasing spatial nodes and/or the number of time steps to ensure convergence of the numerical scheme. Overall though we can see that the models do converge as $\epsilon \to 0$, which is a very strong demonstration of the validity of our multiscale model. It should be noted that since the macroscopic model requires far less nodes to solve, in comparison to the microscopic model, and it can be solved in far less time.

Another interesting measure of if the microscopic and macroscopic models do converge would be to consider at what time do the models reach a complete tear. In other words when does each simulation reach the maximum of $\alpha = 1$. Well in Figure 4.6 we can see that the final tearing times for the microscopic and macroscopic models demonstrate good agreement. Most importantly they appear to have an error within the order of $\mathcal{O}(\epsilon)$. This is another indication that the theory presented in Chapter 3 is correct. That our multiscale model of the damage phase field method correctly homogenises a microscopic model into a macroscopic model. Overall, we can conclude that varying the material elasticity does not negatively effect the convergence of our multiscale model. We have found that the multiscale model does in fact converge as $\epsilon \to 0$, for a large variety of scenarios.

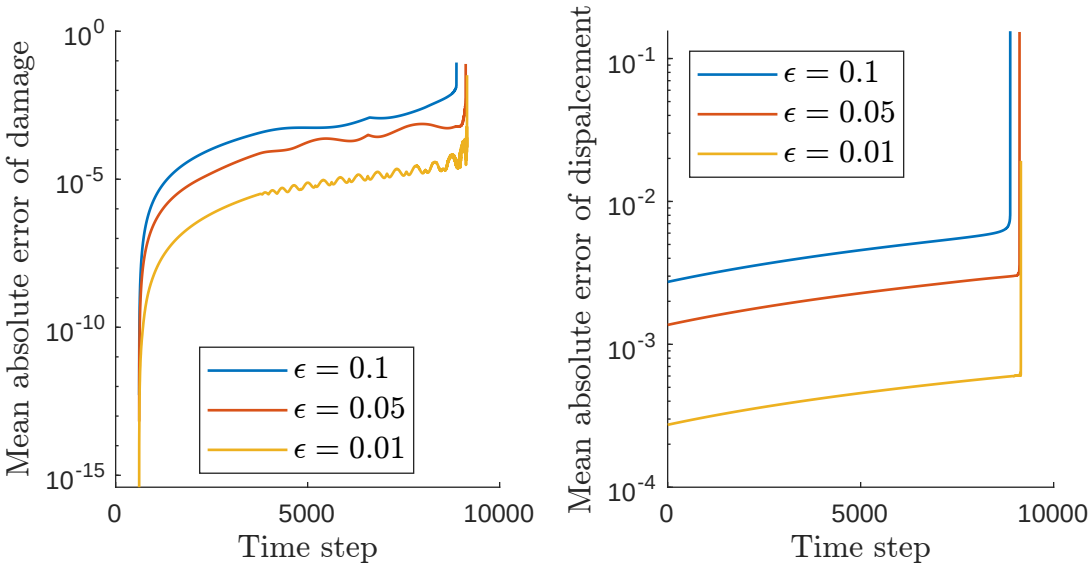


Figure 4.5: Plots of the mean absolute error between the macroscopic and microscopic models. Here we are simulating a one dimensional bar being stretched on one side till breaking as described by the governing equations (4.1). The microscopic coefficients are setup with $A = 0.9$, $B = 0.9$, $C = 1$ and $D = 0.01$ as described in (4.27) and the macroscopic coefficients are as described in (4.28). In these simulations we can see that the mean absolute error decreases for each time step as $\epsilon \to 0$.

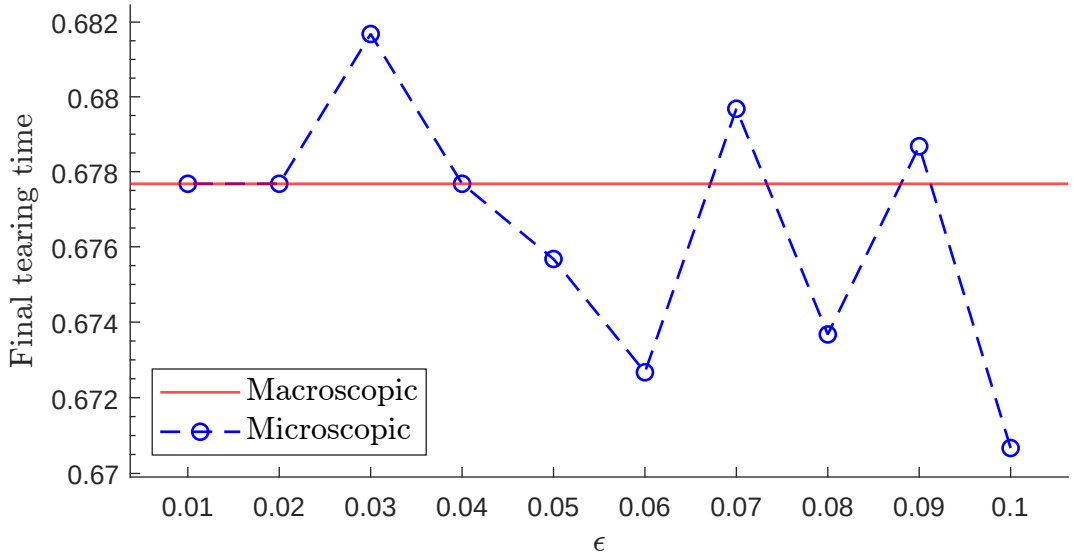


Figure 4.6: A plot of the final tearing times, when $\alpha = 1$, for a macroscopic model and the associated microscopic models for $\epsilon = 0.1, 0.05$ and 0.01. This is for a simulation of a one dimensional bar being stretched till it breaks, as described by the governing equations (4.1), with the boundary condition $k = 1$ on the right hand side. The microscopic coefficients are setup with $A = 0.9$, $B = 0.9$, $C = 1$ and $D = 0.01$ as described in (4.27) and the macroscopic coefficients are as described in (4.28). In these simulations we can see that the final tearing times of the models demonstrate good agreement.

4.3.2 Varying the diffusivity

The diffusivity tensor is an important parameter in the damage phase field model. This parameter plays a significant role in controlling the geometry of the damage phase field

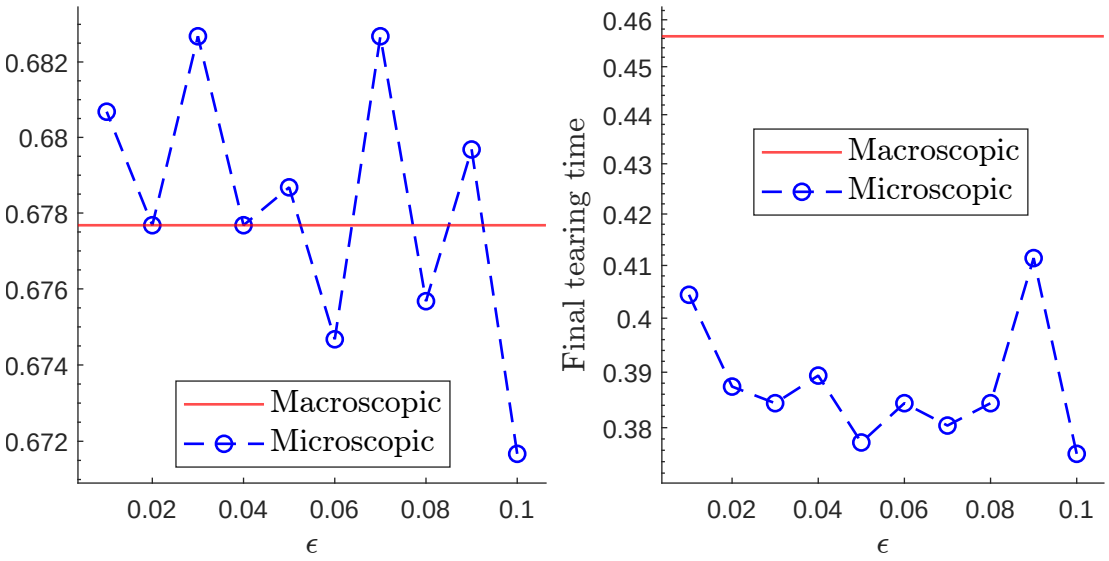
and influencing the damage energy equation. In one dimension of course this tensor has been reduced to a scalar parameter. As noted in Chapter 2 the damage phase field depends on an artificial length scale $\ell \ll 1$, which localises damage. After non-dimensionalisation and other transformations, as introduced in §3.2.2, it can still be seen that the diffusivity should be scaled such that $\mathbb{D} \ll 1$. This presents an interesting problem in terms of asymptotic homogenisation. For asymptotic homogenisation to work we require that ϵ is asymptotically smaller than any other parameter, otherwise the macroscopic and microscopic models will not converge.

To show this we begin by considering the final tearing times for a variety of diffusivities, $\mathbb{D}$. If we therefore consider the scenarios presented in Figure 4.7, it can be seen that the diffusivity being smaller than ϵ is leading to a divergence. However, we can still see that ϵ being of a smaller or similar scale to $\mathbb{D}$, produces microscopic and macroscopic results that are converging. Therefore when we use our multiscale model we must be careful to check that parameters are scaled such that $\epsilon < \mathbb{D} \ll 1$ (this is generally true for all parameters). This feature will of course be present in higher dimensions and one should be careful to choose a diffusivity tensor which has a norm larger than ϵ.

Another illustration of the importance of picking an appropriate scale for $\mathbb{D}$ can be seen in Figure 4.8. Here we can see for a set of simulations with $\mathbb{D} = 10^{-4}$ that the microscopic and macroscopic models still converge as ϵ decreases. We can also see that the simulations last for a larger number of time steps as ϵ decreases. This is because as the material properties oscillate more rapidly, since ϵ is smaller, then according to the theory of asymptotic homogenisation the overall material properties homogenise from a macroscopic perspective. Therefore the sharp diffusivity of damage does not force the damage growth to completely localise, meaning it can diffuse more smoothly and thus will grow slower since all the energy spent creating a new tear is not focused entirely at one point.

4.3.3 Varying the tearing energy

The microscopic tearing energy, defined by G, is homogenised in our macroscopic model by a local integral average (4.8), defined by $\bar{G}$. Any microscopic dependencies in this parameters are essentially just smoothed out at the macroscale. As a physical parameter though it does have an effect on the time it takes for material failure to occur. The time it takes for the evolution from $\alpha = 0$ to $\alpha = 1$ does play an interesting role in the convergence of the multiscale model. If G is small then it does not require a lot of elastic potential energy to create damage in our material, therefore damage can increase rapidly for less energy. Conversely, if G is large then it does require a lot of elastic potential energy to create damage in our material, meaning damage increases more slowly. We can explain this behaviour by studying the damage energy equation (4.24). It is very clear in this



(a) $\mathbb{D} = 10^{-2}$ (b) $\mathbb{D} = 10^{-6}$

Figure 4.7: Plots of the final tearing time for two set of simulations with different values for diffusivity. The simulations are of a one dimensional bar clamped on one side and stretched on the other until breaking, this is described by the governing equations (4.1). Here on the right hand side the boundary condition is set as $k = 1$, the microscopic coefficients are setup with $A = 0.5$, $B = 0.9$ and $C = 1$ as described in (4.27), except the diffusivity which is specified above. The macroscopic coefficients are described by (4.28). In (a) we can see that as ϵ and $\mathbb{D}$ are of similar magnitudes we see a strong agreement between the microscopic models and corresponding macroscopic model. In (b) we can see that $\mathbb{D}$ is of a much lower order of magnitude than ϵ, as a result the final tearing times of the microscopic models and macroscopic models do not agree well. These plots show that convergence requires ϵ to be the smallest parameter, as we would expect.

damage energy equation that the term with the coefficient G balances the driving energy term Q. Hence, large G implies more driving energy is necessary to create a tear and small G implies less driving energy is necessary to create a tear. It should be noted that the driving energy is proportional to the elastic potential energy which is proportional to the strain.

In theory we obviously anticipate this behaviour from the tearing energy G. In Figure 4.9 we demonstrate that a larger choice of G leads to a slower damage growth and larger tearing time; whereas a smaller choice of G leads to a faster damage growth and smaller tearing time. Note that the complete damage growth in Figure 4.9a is completed at time $t \approx 0.60$. Whereas the damage growth in Figure 4.9b took longer, due to the larger value of G as expected, completing at time $t \approx 0.93$.

Here G can have an effect on the convergence of the microscopic and macroscopic damage phase fields. In theory this is because if G is quite large then the growth of the damage phase field will be small in comparison to the relative strain, making it easier to resolve our numerical scheme, as with each time step we do not expect rapid changes in the

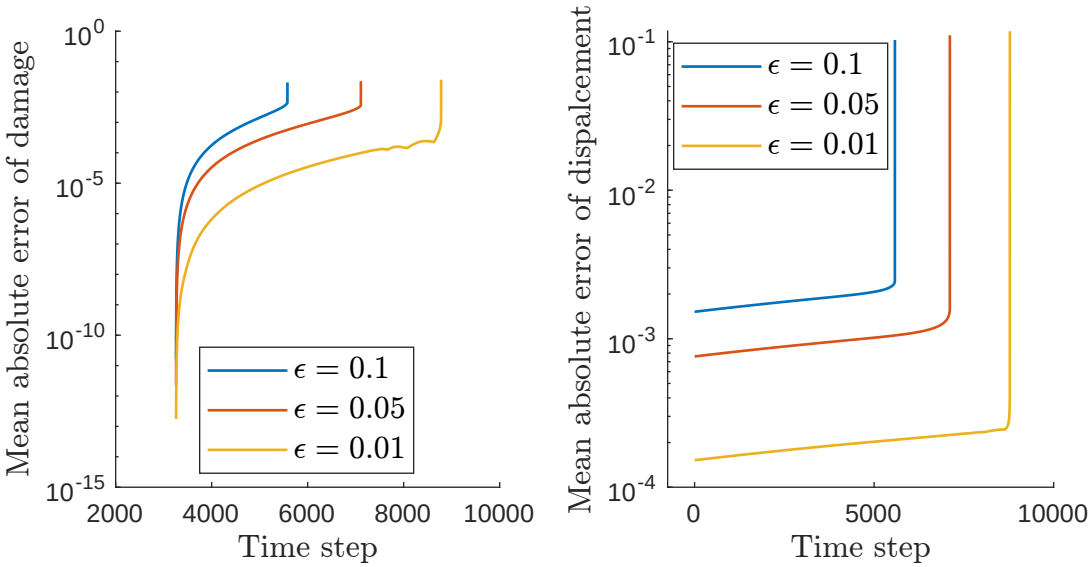


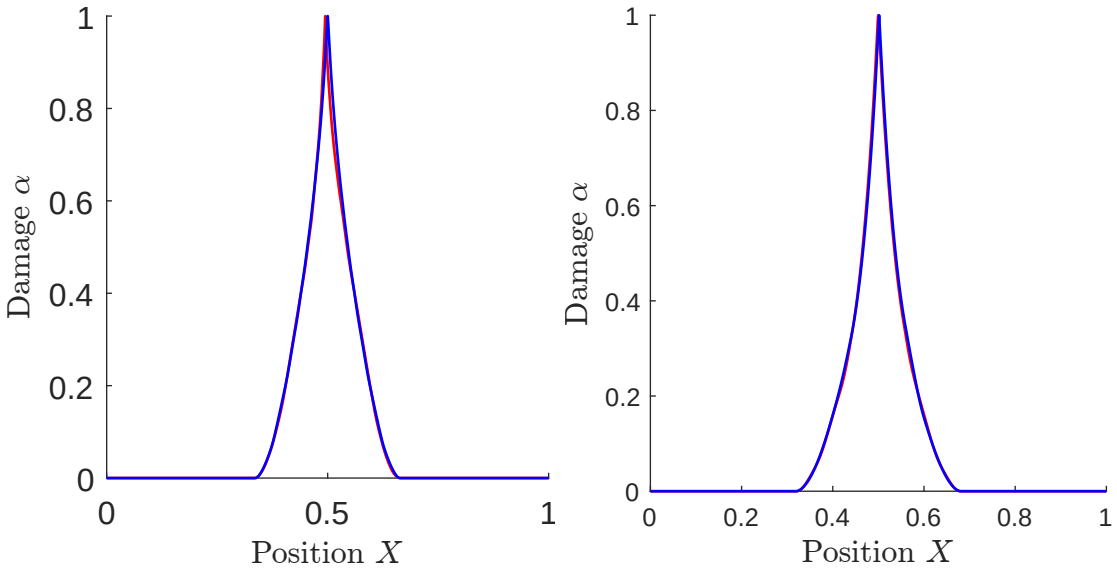
Figure 4.8: Plots of the mean absolute error between the macroscopic and microscopic models. Here we are simulating a one dimensional bar being stretched on one side till breaking as described by the governing equations (4.1). The microscopic coefficients are setup with $A = 0.5$, $B = 0.9$, $C = 1$ and $D = 10^{-4}$ as described in (4.27) and the macroscopic coefficients are as described in (4.28). In these simulations we can see that the mean absolute error decreases for each time step as $\epsilon \to 0$.

damage phase field. The converse is also true, if G is small then the growth of the damage phase field will be large to balance the elastic energy from related strains. Therefore a convergence issue can occur due to a lack of spatial nodes. To fix this issue we can just generally add more spatial nodes, this method should resolve this issue. Again though we can see that in Figure 4.9b we have generally great convergence if ϵ is the smallest order parameter in the model. We know that convergence issues between the microscopic and macroscopic models only occur at the final few time steps when an increases in damage happen most rapidly. This can be seen by considering the mean absolute error for each time step of the damage phase field described in 4.9a.

We illustrate this in Figure 4.10. Here we can see that across most time steps the error is between the microscopic and macroscopic cases is extremely negligible. However, at the final few time steps there is a big jump in the mean absolute error. It happens because of how rapid the damage growth is for a small value of G. To counteract this issue we find that to resolve α well we simply require more spatial nodes. However, if we are modelling using our multiscale model we will be saving ourselves many spatial nodes by at least an order of magnitude, saving back plenty of computing time.

4.3.4 Varying the damage threshold

The last parameter we have to comment on is the damage threshold $\psi(X)$. It does not really have much of an interesting physical role to play in our model, in terms of the



(a) $G = 0.1$ (b) $G = 4$

Figure 4.9: Plots of the final damage phase field for two set of simulations with different values for G. The simulations are of a one dimensional bar clamped on one side and stretched on the other until breaking, this is described by the governing equations (4.1) with the red line representing the microscopic case and the blue representing the macroscopic case. Here on the right hand side the boundary condition is set as $k = 1$, the microscopic coefficients are setup with $A = 0.5$, $B = 0.9$ and $D = 0.01$ as described in (4.27), except for the tearing energy which is prescribed above. The macroscopic coefficients are described by (4.28).

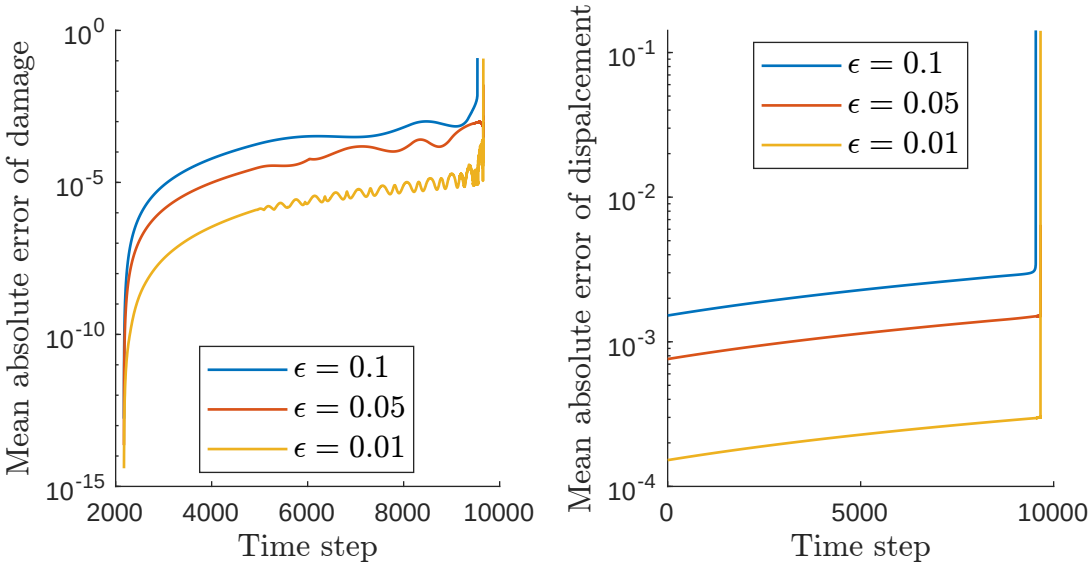
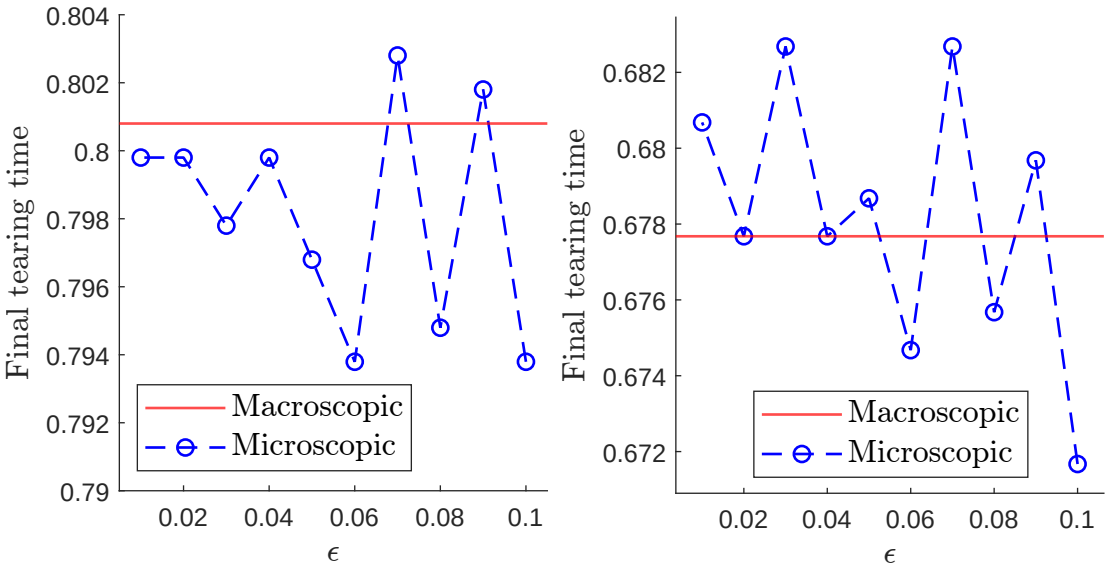


Figure 4.10: Plots of the mean absolute error between the macroscopic and microscopic models. Here we are simulating a one dimensional bar being stretched on one side till breaking as described by the governing equations (4.1). The microscopic coefficients are setup with $A = 0.5$, $B = 0.9$, $C = 0.1$ and $D = 0.01$ as described in (4.27) and the macroscopic coefficients are as described in (4.28).

convergence. All it really does is control how early and where we can expect the initiation of damage growth to take place. Throughout these simulations we found that ψ affected the outcomes of the model as expected. A simple example of this is the final tearing times of the microscopic and macroscopic models for two different cases: $B = 0.6$ and $B = 0.99$, as defined in (4.27). These two sets of final tearing times are illustrated in Figure 4.11. As should be expected the case with the smaller value of $\psi(X)$ reaches $\alpha = 1$ first. We can also see that yet again the final tearing times of the microscopic and macroscopic models all demonstrate good agreement.



(a) $B = 0.75$ (b) $B = 0.9$

Figure 4.11: Plots of the final tearing time for two set of simulations with different values for diffusivity. The simulations are of a one dimensional bar clamped on one side and stretched on the other until breaking, this is described by the governing equations (4.1). Here on the right hand side the boundary condition is set as $k = 1$, the microscopic coefficients are setup with $A = 0.5$, $C = 1$ and $D = 0.01$ as described in (4.27), except for B. The macroscopic coefficients are described by (4.28).

4.4 Summary of results

This chapter focused on two main goals: firstly to explain how we can solve our damage model numerically and secondly use these numerically methods to illustrate the convergence of the microscopic and macroscopic models. Both of these goals where fulfilled, we now have a simple methodology for solving our damage model numerically as explained in §4.2. Using our new numerical schemes we were able to perform many simulations on a model problem of a stretched one dimensional bar, illustrated in Figure 4.1. The results of these simulations are outlined in §4.3, where we found for every case we tested that the microscopic and macroscopic models converge as ϵ decreases.

In doing many simulations we also found that the model does behave as expected. The threshold function does control damage initiation, damage growth only occurs when the criterion is satisfied, damage is irreversible and diffusion controls the localisation of the damage phase field. All of these properties where originally outlined in Chapter 2 and we now know that after the upscaling performed in Chapter 3 that all these properties are still satisfied. Not only that but we have strong evidence that the theory presented in Chapter 3 is in fact correct and that our novel multiscale model correctly homogenises a microscopic damage phase field model.

The macroscopic model also proved itself to be very computationally efficient. As it uses a significantly smaller mesh size than the microscopic model we found that the macroscopic model can be solved exceptionally quicker than the microscopic model. The difference between solving a model described by 10^3 rather than 10^5 nodes is the difference between minutes and hours. In higher dimensions when the multiscale model is used to model something much more practical, like an aortic dissection, this time saving will be extremely important.

4.4.1 Concluding remarks

To create a multiscale model of aortic dissections we cannot treat an aorta as a one dimensional object. It is a complicated multilayered three dimensional structure. Therefore, we need to expand the numerical work of this chapter to higher dimensions. We can do this by using the finite element method and by introducing a new numerical algorithm, which we will do in the following chapter. However, this chapter has fulfilled its goal, we have provided evidence that the theoretical work of Chapter 3 appears to be correct. So we should trust that we have a good multiscale model of damage evolution in an arbitrary material. Applying that model to the human aorta will allow us to study in great depth how many possible microscopic factors contribute to aortic dissections.

Chapter 5

Applications in higher dimensions

In this chapter, we will introduce a set of numerical methods for solving our multiscale models in higher dimensions. To do this, we use the finite element method, which allows us to solve our multiscale model in any dimension. We can implement this solution in any domain $\Omega \subseteq \mathbb{R}^n$ with $n \in \mathbb{N}$, for a large variety of external loading conditions. An added advantage of working in higher dimensions (dimensions greater than one) is that we can implement a tear in Ω whenever a localised region of Ω becomes completely damaged. This will require us to construct a new numerical algorithm that accounts for a changing domain Ω. To implement these tears numerically we alter the mesh that approximates Ω as a localised region of this mesh becomes completely damaged. This will allow us to visualise how tears spread through Ω.

In this chapter we will introduce the finite element method, a new numerical algorithm which accounts for a tearing mesh and the required numerical methods to solve both the equation of motion and the damage evolution equation. This method was directly implemented into Matlab [72] and used to perform all the simulations in this chapter. All the code used in this chapter can be found in Appendix B. Using our code we can explore our model through an example of a trouser test. During the trouser test simulation, a rectangular piece of material is held steady at one end while force is applied to the other, splitting it down the middle. This tearing process produces a configuration that visually resembles trouser legs. By considering two simulations that are entirely similar except for their microscopic geometries, we will be able to convey how small microscopic changes affect outcomes on the macroscale.

5.1 Finite element method

In modern applied mathematics, science and engineering many multidimensional models cannot be solved analytically. The approach most commonly adopted to solve these models numerically is the finite element method, which finds its roots in the works of Ritz

and Galerkin [133]. With modern computing we can use their work to approximate the solutions for extremely complicated setups of partial differential equations. We do this by approximating the domain $\Omega \in \mathbb{R}^n$ we are solving over into a mesh, where $n \in \mathbb{N}$. The essence of the finite element method is to partition the domain of the problem into non-overlapping elements and to provide an approximate solution that has a simple form within each element. When using the finite element method we locally discretise our variables by describing them as approximate sums of N polynomial functions, where $N \in \mathbb{N}$. As these local forms are simple, accuracy is achieved by making the elements as small as possible. When using this method we discretise the damage and displacement variables as follows:

$$\begin{aligned} \mathbf{u}(\mathbf{X},t) &\approx \mathbf{u}^h(\mathbf{X},t) = \sum_{i=1}^{N} p^{(i)}(\mathbf{X})\mathbf{u}^{(i)}(t), \\ \alpha(\mathbf{X},t) &\approx \alpha^h(\mathbf{X},t) = \sum_{i=1}^{N} p^{(i)}(\mathbf{X})\alpha^{(i)}(t) \\ &\text{such that} \quad p^{(i)}(\mathbf{X}) : \Omega \to \mathbb{R}. \end{aligned} \tag{5.1}$$

Here, $\mathbf{u}^h$ and α^h are the approximate solutions of the displacement and damage, respectively. With h representing the spacing of nodes, a proxy for the size of elements, as $h \to 0$ the approximation of course improves. The subscript i indicates the indices of all the discretised nodes, across a meshed domain. We have also introduced the scalar shape functions $p^{(i)}$ which interpolate the solutions of displacement and damage inside each element. This is done with a linear interpolation between nodes that construct an element. Here our shape functions satisfy the general definition

$$p^{(i)}(\mathbf{X}) = \begin{cases} 1, & \mathbf{X} = \mathbf{X}_i \\ 0, & \mathbf{X} = \mathbf{X}_j, \quad j \neq i, \end{cases} \tag{5.2}$$

where j are all the nodes neighbouring i. Here we define neighbouring nodes as nodes that belong to the same elements. It should be noted that every shape functions satisfies the property

$$\sum_{j=1}^{N} p^{(j)}(\mathbf{X}) = 1,$$

such that $\mathbf{u}^h(\mathbf{X}_i,t) = \mathbf{u}^{(i)}(t)$ and $\alpha^h(\mathbf{X}_i,t) = \alpha^{(i)}(t)$. Local elements, described by the shape functions, can generally be any shape. In our coding the shape functions will be defined by triangles constructed out of three nodes, one for each vertex, because it is simple and easy to implement. With the finite element method one can efficiently approximate any geometry given that $h \ll 1$, for whatever shape function is chosen.

By considering a weak form of our governing macroscopic equations, (3.55) and (3.56), we can use the above method of discretisation to construct systems of equations in terms

of $\mathbf{u}_i$ and α_i. We can solve these systems of equations for every $\mathbf{u}_i$ and α_i using general linear methods and iterative methods, respectively. For a better introduction to the finite element method we would recommend the textbook by Wait et al. [133] and of course Numerical Recipes [107].

5.2 Problem Setup

In this chapter we will not solve the most general version of our macroscopic model as defined in (3.55) and (3.56). We will instead solve a simplified version of the model, to make things easier to solve numerically. Here we shall assume that the equation of motion is steady state and that our problem has no residual stress. The assumption of a steady state is made to ensure our equation is motion is not a hyperbolic partial differential equation. We are ignoring residual stress in our model because we are not currently modelling any physical phenomena that experiences residual stress, so we ignore them to make calculating our numerical solution simpler. As a result we can write down the simplified version of the model as

$$\nabla_{\mathbf{X}} \cdot \left[g(\bar{\alpha}) \bar{\mathbb{C}} : \xi_{\mathbf{X}}(\bar{\mathbf{u}}) \right] + \bar{\mathbf{B}} = \mathbf{0} \quad \text{in } \Omega, \tag{5.3a}$$

$$\bar{\eta} \dot{\bar{\alpha}} = \left\langle -g'(\bar{\alpha}) \left(\frac{1}{2} \xi_{\mathbf{X}}(\bar{\mathbf{u}}) : \bar{\mathbb{C}} : \xi_{\mathbf{X}}(\bar{\mathbf{u}}) - \bar{\psi} \right) - \bar{G} \bar{\alpha} + \nabla_{\mathbf{X}} \cdot \left(\bar{\mathbb{D}} \nabla_{\mathbf{X}} \bar{\alpha} \right) \right\rangle_{+} \quad \text{in } \Omega. \tag{5.3b}$$

In this setup $\Omega \subseteq \mathbb{R}^n$ is our reference configuration, with $n \in \mathbb{N}$. Note here we employ the rate-dependent version of the model, as it transforms the damage evolution equation from an elliptic to a parabolic partial differential equation. This will make our model much more computationally robust because we will be able to implement a Crank-Nicholson style numerical scheme which will be more efficient for simulating a larger number of nodes (as we would expect in higher dimensions). Using the contents of the Macaulay brackets in (5.3b) we can define our multidimensional damage criterion. We write this as

$$-g'(\bar{\alpha}) \left(\frac{1}{2} \xi_{\mathbf{X}}(\bar{\mathbf{u}}) : \bar{\mathbb{C}} : \xi_{\mathbf{X}}(\bar{\mathbf{u}}) - \bar{\psi} \right) - \bar{G} \bar{\alpha} + \nabla_{\mathbf{X}} \cdot \left(\bar{\mathbb{D}} \nabla_{\mathbf{X}} \bar{\alpha} \right) > 0, \tag{5.4}$$

which is comparable to the one-dimensional damage criterion (4.17) defined in Chapter 4.

The governing equations are of course completed with a set of appropriate boundary conditions. For the macroscopic equation of motion (5.3a) we apply traction boundary conditions to complete the model. This boundary condition is defined on $\partial\Omega$ as

$$\mathbf{t} = \left(\bar{\mathbb{C}} : \xi_{\mathbf{X}}(\bar{\mathbf{u}}) \right) \cdot \mathbf{n} \quad \text{on } \partial\Omega. \tag{5.5}$$

For the macroscopic damage evolution equation we of course apply a no flux boundary

conditions on $\partial\Omega$ such that

$$\left(\bar{\mathbb{D}}\nabla\bar{\alpha}\right)\cdot\mathbf{n}=0 \quad \text{ on } \partial\Omega. \tag{5.6}$$

Note that one can also apply Dirichlet boundary conditions for the macroscopic displacement $\bar{\mathbf{u}}$. External loading is applied to our system through two mechanisms: the body force in (5.3a) and the surface traction defined by (5.5).

For clarity let us be explicit about the number of equations, the number of unknowns and the number of parameters in our system (5.3) for a given n-dimension. The number of equation of motions described by (5.3a) is equal to n and the damage evolution equation (5.3b) is a scalar equation, hence we have a system of $n+1$ equations. As the displacement and damage are defined such that $\mathbf{u}\in\mathbb{R}^n$ and $\alpha\in\mathbb{R}$, respectively, then our system depends on $n+1$ unknowns. Here our effective coefficients define a number of parameters: $\bar{\mathbb{C}}$ is a fourth order rank tensor with minor and major symmetries that defines 6 parameters in two dimensions and 21 parameters in three dimensions; $\bar{\mathbf{B}}$ is a vector with n parameters; $\bar{\eta}$ is a scalar parameter; $\bar{\psi}$ is a scalar parameter; $\bar{G}$ is a scalar parameter and $\bar{\mathbb{D}}$ is a second order tensor with n^2 parameters. In total our system (5.3) depends on: 6 parameters in one dimension; 20 parameters in two dimensions and 46 parameters in three dimensions.

5.2.1 Effective coefficients

Our macroscopic model (5.3) depends on a series of effective coefficients to encode and upscale information about the microscopic materials properties and geometry. This is done by calculating the effective coefficients using local integral averages of the microscopic coefficients. As the effective coefficients do not have any time dependencies, then we only need to do this once in software which is conveniently efficient. For a clearer understanding recall the macroscopic coefficients (3.57) and the associated local cell problems (3.58), (3.59) and (3.60), all of which can be found in §3.4. Consider Figure 5.1 of a two dimensional local microscopic geometry defined by the domain $\omega\subseteq\mathbb{R}^n$, where $n\in\mathbb{N}$ is the dimension of ω. This geometry is of a host matrix ω_A with a number of inclusions ω_β for $\beta=1,\ldots,B$ with $B\in\mathbb{N}$. Hence we can say our microscopic domain ω is defined such that

$$\omega=\omega_A\cup\left(\bigcup_{\beta=1}^{B}\omega_\beta\right). \tag{5.7}$$

For completion we define the boundaries and interfaces of ω as follows:

$$\partial\omega=(\partial\omega_A\cup\partial\omega_\beta)\setminus\Gamma_\beta \quad \text{ where } \quad \Gamma_\beta=\omega_A\cap\omega_\beta. \tag{5.8}$$

Here $\partial\omega$ is the boundary of our local cell and Γ_β is the interface between the host matrix and each respective inclusion. For our purposes in each subdomain we will say the material properties are uniform which still gives us the freedom to choose a complex microscopic

composite geometry with large variations in material properties per subdomain. We set the material properties in each subdomain as uniform, for convenience, when calculating the macroscopic effective coefficients. To avoid any confusion, let us also state that all the inclusions $\omega_\beta \ \forall \beta = 1, ..., B$ never intersect or even touch. We write this as

$$\omega_i \bigcap_{i \neq j} \omega_j = \emptyset \quad \forall i, j = 1, .., B.$$

It should be noted that we when calculating effective coefficients using our numerical scheme, as found in Appendix B, we never experienced any issues when using domains with sharp corners. We are mentioning this so that the reader does not feel limited by the visualisation in Figure 5.1, however please note that in soft tissue problems (which we are motivated by) sharp corners are extraordinarily rare.

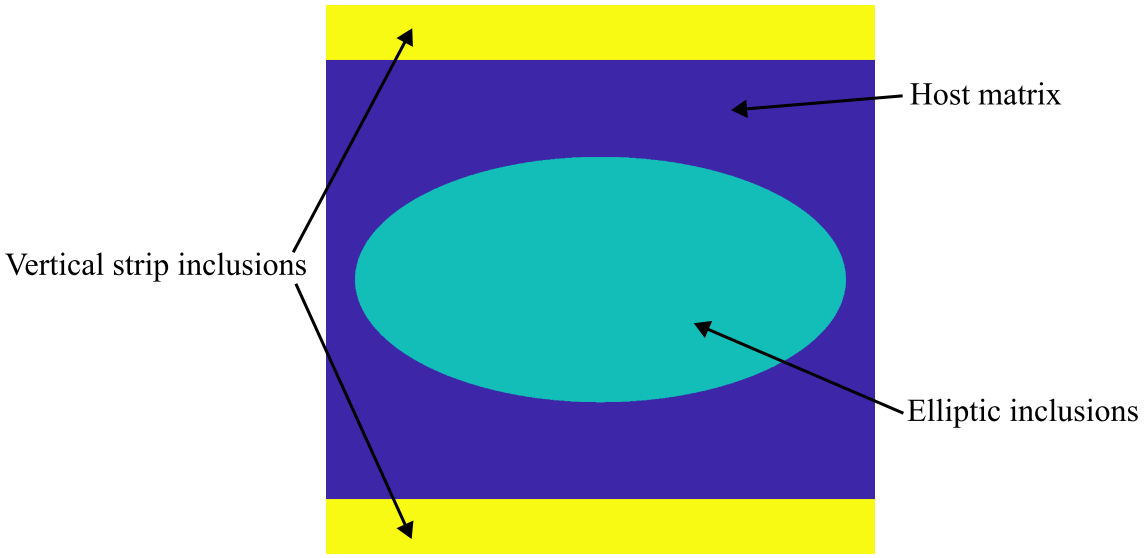


Figure 5.1: An example of the two dimensional microscopic geometries we can generate in Matlab. This microscopic geometry encapsulates a host matrix with three inclusions of various shapes and sizes. Each subdomain has its own uniform material properties.

Let us begin by explaining how to calculate the simplest form of effective coefficients: local integral averages. The local integral averages, defined by notation from Chapter 3 (3.6), as presented in (5.3) can be written as

$$\begin{aligned} &\bar{\rho} = \{\rho_A\}_{\omega_A} + \sum_{\beta=1}^{B} \{\rho_\beta\}_{\omega_\beta}, \quad \bar{\mathbf{B}} = \{\rho_A \mathbf{b}_A\}_{\omega_A} + \sum_{\beta=1}^{B} \{\rho_\beta \mathbf{b}_\beta\}_{\omega_\beta}, \\ &\bar{\psi} = \left\{\hat{\psi}_A\right\}_{\omega_A} + \sum_{\beta=1}^{B} \left\{\hat{\psi}_\beta\right\}_{\omega_\beta}, \quad \bar{G} = \{G_A\}_{\omega_A} + \sum_{\beta=1}^{B} \{G_\beta\}_{\omega_\beta} \\ &\text{and} \quad \bar{\eta} = \{\eta_A\}_{\omega_A} + \sum_{\beta=1}^{B} \{\eta_\beta\}_{\omega_\beta}. \end{aligned} \tag{5.9}$$

All of these local integral averages can be calculated quickly and easily in software. This can be seen by considering Figure 5.1, we can use this approximate geometry in Matlab to calculate the local integral averages (5.9). To do this at every single macroscopic point $\mathbf{X} \in \Omega$ we approximate the local integral averages as Riemann sums. This is done by discretising our microscopic coefficients over the microscopic domain. The geometry presented in Figure 5.1 is discretised by a number of pixels, where at each pixel we define the local microscopic properties. For example consider the macroscopic coefficient $\bar{\rho}$, we can discretise ρ_A and ρ_β for all β such that

$$\rho_A = \sum_{k=1}^{M} \rho_A^{(k)} \quad \text{and} \quad \rho_\beta = \sum_{k=1}^{M} \rho_\beta^{(k)} \quad \forall \beta = 1, \ldots, B, \tag{5.10}$$

where $M \in \mathbb{N}$ is the number of pixels we are discretising the microscopic geometry by. Each pixel is being indexed by the subscript k. The functions $\rho_A^{(k)}$ and $\rho_\beta^{(k)}$ are the local values of ρ_A and ρ_B in each pixel k, respectively. Here we have

$$\rho_A : \omega_A \to \mathbb{R} \quad \text{and} \quad \rho_\beta : \omega_\beta \to \mathbb{R} \quad \forall \beta = 1, \ldots, B, \tag{5.11}$$

such that we can discretise these functions as seen in (5.10), where

$$\rho_A^{(k)} : \omega_A^{(k)} \to \mathbb{R} \quad \text{and} \quad \rho_\beta^{(k)} : \omega_\beta^{(k)} \to \mathbb{R} \quad \forall \beta = 1, \ldots, B. \tag{5.12}$$

For clarity we define $\rho_A^{(k)}$ and $\rho_\beta^{(k)}$ $\forall \beta$ to be

$$\begin{aligned} \rho_A^{(k)}(\mathbf{X}, \mathbf{Y}) &= \begin{cases} \rho_A(\mathbf{X}, \mathbf{Y}^{(k)}) & \text{if} \quad \mathbf{Y} \in \omega_A^{(k)}, \\ 0 \quad \text{otherwise} , \end{cases} \\ \rho_\beta^{(k)}(\mathbf{X}, \mathbf{Y}) &= \begin{cases} \rho_\beta(\mathbf{X}, \mathbf{Y}^{(k)}) & \text{if} \quad \mathbf{Y} \in \omega_\beta^{(k)}, \\ 0 \quad \text{otherwise} \end{cases} \quad \forall \beta = 1, \ldots, B. \end{aligned} \tag{5.13}$$

Here $\omega_A^{(k)}$ and $\omega_\beta^{(k)}$ are the subdomains defined by each pixel in Figure 5.1 and $\mathbf{Y}^{(k)}$ is the central point in each of these subdomains. Using this discretisation method we can approximate our local integral average by calculating

$$\bar{\rho}(\mathbf{X}) \approx \frac{1}{|\omega|} \left(\sum_{k=1}^{M} \rho_A^{(k)}(\mathbf{X}, \mathbf{Y}) \left| \omega_A^{(k)} \right| + \sum_{\beta=1}^{B} \sum_{k=1}^{M} \rho_\beta^{(k)}(\mathbf{X}, \mathbf{Y}) \left| \omega_\beta^{(k)} \right| \right). \tag{5.14}$$

Accuracy in (5.14) increases as M gets large, analogous to how the accuracy of a Riemann sum approximation of an integral increases. We apply the above approximation of the local integral average (5.14) in software for all the macroscopic coefficients which are

simply local integral averages. As the material properties are assigned at every pixel in Figure 5.1 then we can very simply sum up the volume fractions of the subdomains in Matlab to calculate the integrals. The implementation of this methodology for calculating local integral averages can be found in Appendix B.3.

Our macroscopic problem (5.3) has two more effective coefficients that are more difficult to calculate. This is because the macroscopic coefficients $\bar{\mathbb{C}}$ and $\bar{\mathbb{D}}$ depend on a series of ansätze, which we need to calculate from a series of linear cell problems (3.58) and (3.60). We can write these effective coefficients as

$$\begin{aligned}\bar{\mathbb{C}} &= \left\{\mathbb{C}_A + \mathbb{C}_A : \xi_{\mathbf{Y}}(\chi_A)\right\}_{\omega_A} + \sum_{\beta=1}^{B} \left\{\mathbb{C}_\beta + \mathbb{C}_\beta : \xi_{\mathbf{Y}}(\chi_\beta)\right\}_{\omega_\beta}, \\ \bar{\mathbb{D}} &= \left\{\mathbb{D}_A + \mathbb{D}_A \nabla_{\mathbf{Y}} \zeta_A\right\}_{\omega_A} + \sum_{\beta=1}^{B} \left\{\mathbb{D}_\beta + \mathbb{D}_\beta \nabla_{\mathbf{Y}} \zeta_\beta\right\}_{\omega_\beta},\end{aligned} \tag{5.15}$$

with the ansatz χ_A, χ_β, ζ_A and ζ_β calculated from the local cell problems:

$$\begin{aligned}&\nabla_{\mathbf{Y}} \cdot \mathbb{C}_A = -\nabla_{\mathbf{Y}} \cdot \left[\mathbb{C}_A : \xi_{\mathbf{Y}}(\chi_A)\right] \text{ in } \omega_A, \\ &\nabla_{\mathbf{Y}} \cdot \mathbb{C}_\beta = -\nabla_{\mathbf{Y}} \cdot \left[\mathbb{C}_\beta : \xi_{\mathbf{Y}}(\chi_\beta)\right] \text{ in } \omega_\beta \, \forall \beta, \\ &\left(\mathbb{C}_A + \mathbb{C}_A : \xi_{\mathbf{Y}}(\chi_A)\right) \cdot \mathbf{n}_A = \left(\mathbb{C}_\beta + \mathbb{C}_\beta : \xi_{\mathbf{Y}}(\chi_\beta)\right) \cdot \mathbf{n}_A \text{ on } \Gamma_\beta \, \forall \beta, \\ &\chi_A = \chi_\beta \text{ on } \Gamma_\beta \, \forall \beta\end{aligned} \tag{5.16}$$

and

$$\begin{aligned}&\nabla_{\mathbf{Y}} \cdot \left(\mathbb{D}_A + \mathbb{D}_A \nabla_{\mathbf{Y}} \zeta_A\right) = \mathbf{0} \text{ in } \omega_A, \\ &\nabla_{\mathbf{Y}} \cdot \left(\mathbb{D}_\beta + \mathbb{D}_\beta \nabla_{\mathbf{Y}} \zeta_\beta\right) = \mathbf{0} \text{ in } \omega_\beta \, \forall \beta, \\ &\left(\mathbb{D}_A + \mathbb{D}_A \nabla_{\mathbf{Y}} \zeta_A\right) \mathbf{n}_A = \left(\mathbb{D}_\beta + \mathbb{D}_\beta \nabla_{\mathbf{Y}} \zeta_\beta\right) \mathbf{n}_A \text{ on } \Gamma_\beta \, \forall \beta, \\ &\zeta_A = \zeta_\beta \text{ on } \Gamma_\beta \, \forall \beta.\end{aligned} \tag{5.17}$$

The key to calculating the effective coefficients (5.15) is exploiting local periodicity. Fortunately, these local cell problems are what we would consider as classic problems in the field of asymptotic homogenisation, which for us means that the numerical solutions for (5.15) exist in the literature [3, 52, 82]. It should be noted that a great resource for studying the effective coefficients, their properties and roles can be found in the work of Hassani et al. [58, 57, 59].

In the literature we found an excellent paper by Andreasen et al. [3] which provides code for calculating the effective coefficients in (5.15). We have since adopted it and it can be found in the Appendix B.3, it can not be understated how useful a resource this is. To use the code we provide discretised domains of ω as can be seen in Figure 5.1. This code does have limitations for what form the microscopic elasticity and diffusion can take.

First, let us consider the microscopic elasticity $\mathbb{C}$, which is a fourth order tensor. In this

chapter we will assume that the elasticity is isotropic for all microscopic constituents which allows us to use the code provided by Andreasen et al. [3]. We can define an isotropic elasticity in terms of the first and second Lamé parameters: λ and μ, respectively. The Lamé parameters will be constant within the local cell ω. Thus we define the microscopic elasticity as

$$\mathbb{C}_{ijkl}(\mathbf{X}, \mathbf{Y}) = \lambda(\mathbf{X})\delta_{ij}\delta_{kl} + \mu(\mathbf{X})\left(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}\right), \tag{5.18}$$

where δ_{ij} is the Kronecker delta, defined such that

$$\delta_{ij} = \begin{cases} 0 \text{ if } i \neq j \\ 1 \text{ if } i = j. \end{cases}$$

Similarly, for the effective microscopic diffusion we are limited to an uniform case. Therefore we can define the microscopic diffusion as a diagonal matrix such that

$$\mathbb{D}_{ij}(\mathbf{X}, \mathbf{Y}) = d(\mathbf{X})\delta_{ij}, \tag{5.19}$$

where d is some non-dimensional scale for diffusion. As expected we need to set $\epsilon < d \ll 1$ for proper asymptotic homogenisation to work, this result was discussed in detail in §4.3.

Remark. *If we take d as a length scale for the diffusion then we need to be very careful to choose an appropriate size for our finite elements. Setting h as the length scale for the finite elements then a general rule of thumb that should be considered is*

$$2h < d.$$

This will provide the numerical scheme with the appropriate amount of resolution to capture the behaviour of the problem. If we do not satisfy this need in general for diffusive problems when using the finite element method then we will either have code that fails to converge or unrealistic non-smooth results.

5.3 n-dimensional numerical solution

In this section we introduce a numerical solution for the damage problem introduced in §5.2 and defined by the equations (5.3a) and (5.3b), for the macroscopic displacement and damage, respectively. We will of course need to correctly satisfy the irreversibility of damage growth and the damage criterion. Fortunately, we already solved this problem in §4.2 and introduced Algorithm 1 for solving the one dimensional damage problem. This of course also applies at higher dimensions but we need to introduce a finite element solution for both the elastic and damage problem, which solve the displacement and damage,

respectively. Other works that mention finite element solutions for the damage phase field can be found in the works of Miehe et al. [90] and Wu et al. [139].

In higher dimensions we can actively change the topology of our problem as our mesh is torn. The mesh becomes torn whenever we have a localised concentration of nodes such that $\bar{\alpha} = 1$. To implement this feature to our numerical solution, we will need to firstly consider how we will actively change the mesh at each time step. Secondly, we will need to consider how this will effect how we solve our problem. Of course if we change the mesh this could change the result calculated at each time step, we therefore must reformulate our numerical algorithm to account for this.

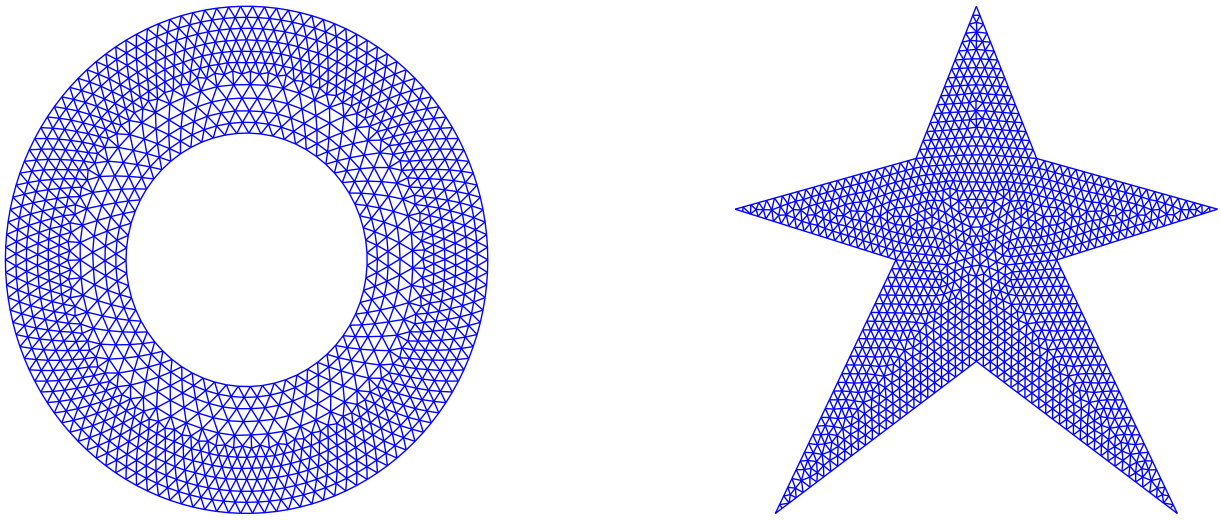
Throughout the rest of this section we will explain what geometries we can consider for our damage problem. Introduce a number of methods for deleting nodes actively in our code, which is written again in Matlab [72]. Next we will introduce a new algorithm for solving our damage problem in dimensions greater than or equal to two. After this we will introduce the specific methods to solve the elastic problem, calculate our damage criterion and solve the damage problem. The full implementation of our numerical techniques can be found in the code listings throughout Appendix B.

5.3.1 Mesh generation

The greatest utility of the finite element method is our ability to approximate any geometry using a mesh defined by a simple local element. In our case we construct our finite elements using triangles constructed by three nodes defining the vertices of the triangles. Our elements are of course then these triangles, and this approach of creating a mesh with triangles can be easily done for any dimension greater than one. It should be noted that the accuracy of this method could be greatly improved by increasing the numbers of nodes used to define the triangles. In our case we use three nodes, one for each vertex, which is called a linear triangle in finite element analysis [133]. The accuracy of our method can be increased by using a shape function defined by more nodes. For example we could define a shape function as a triangle described by six nodes (P2 elements): three for the vertices and three for the midpoints of each edge. Here we will use three nodes for each of our triangles to keep everything simple. Conveniently most software packages will convert most geometries to an approximate mesh constructed by these triangular elements. In Matlab this can be done using a number of the functions available in the partial differential equations toolbox [73]. When we generate a mesh we create vectors for each spatial dimension in $\mathbf{X}$ that record the positions of every node in each spatial dimension. The triangles used to construct the mesh are described by a connectivity matrix which has all the information about which nodes form which triangles. This is done by defining the connectivity matrix to be three columns for the nodes by the number of triangles for the rows. In each row of this matrix you will find three whole numbers

referring to the indices for three nodes in our mesh which together form a single triangular element. This is a very elegant methodology to work with and is the universal approach to finite element analysis in software.

Using this approach we can construct a variety of interesting geometries. For example in Figure 5.2 we have created an example of two complicated macroscopic geometries approximated by nothing more than simple triangular elements. We have an annulus which of course is made up of curved lines but we can see is well approximated by triangular elements. Our other geometry is a star which is also very well approximated by triangular elements. These are of course two dimensional approximations but we can easily extend this methodology to higher dimensions using Matlab's inbuilt partial differential equation toolbox. Our implementation of creating a mesh can be found in Appendix B.1.



(a) An annulus geometry

(b) A star geometry

Figure 5.2: A simple example of two different possible geometries one can generate using triangular elements. These where created in Matlab using the partial differential toolbox, in particular the "polyshape" and "generate mesh" Matlab functions.

5.3.2 Deletion of nodes

As mentioned one of the great advantages of working in higher dimensions is the ability to actually simulate tears through our material. In software it is possible to implement a system where at any point $\mathbf{X}$ that $\bar{\alpha} = 1$ we can simulate a tear, just like a material being torn. This can be a rather difficult process to implement and requires careful consideration. To implement this idea in software we need to consider two things: how close does the damage α have to be to one for us to consider it fully torn and which nodes should we delete from our mesh?

The best way to evaluate a tear in software is to consider a small interval $[1 - \delta, 1]$ with $\delta \ll 1$. Whenever $\bar{\alpha}$ falls into this interval it is best to consider that point effectively

torn and set $\bar{\alpha} = 1$ at the relevant node. We do this because if in our iterative method the solution for $\bar{\alpha}$ skips above 1 it is possible that $\bar{\alpha}$ may diverge upwards. It is also extremely unlikely that an iterative scheme will ever fully reach a stable node like $\bar{\alpha} = 1$ because numerical methods at best produce approximations and not exact solutions. Since we know the desired upper limit of $\bar{\alpha}$ is 1 then its easy for us in software to set $\bar{\alpha} = 1$ whenever $\bar{\alpha}$ is really close to one. Now just because a node has $\bar{\alpha}_i = 1$ does not mean we should remove that node from the mesh.

We have in fact found three methodologies for when we should delete nodes based on the local values of $\bar{\alpha}_i$. Our first approach is the simplest where we of course just delete any nodes from the mesh when that node satisfies $\bar{\alpha}_i = 1$, followed by an update to the connectivity matrix and mesh as appropriate. The second approach is to delete any triangles (rows of the connectivity matrix) when all of its vertices have reached $\bar{\alpha}_i = 1$. Then one can delete nodes when they are no longer referenced by any rows in the connectivity matrix. The third, and final, approach is to delete any node at when in all the elements it is a part of we have that all of the nodes satisfy $\bar{\alpha}_i = 1$. When we use this approach we would need to remove all the relevant rows from the connectivity matrix and delete the node common to all of these elements from the mesh. In the many simulations we have done we found all of these methods work well. In our simulations we will use the second method. We make this choice because we have found it to provide the best computational feasibility and usually allows our simulations to converge for more time steps and more efficiently.

To track the changes in the mesh we can simply define the indicator function $\mathrm{Torn}(\mathbf{X})$, which relays information in our numerical solution about which points $\mathbf{X} \in \Omega$ have reached $\alpha(\mathbf{X}) = 1$. We define this in a very similar way to the field $\mathrm{Crit}(\mathbf{X})$, defined by (4.22) in §4.2. So for numerical convenience let us define

$$\mathrm{Torn}(\mathbf{X}) = \begin{cases} 1, & \bar{\alpha}(\mathbf{X}) = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{5.20}$$

This can easily be discretised across our mesh and defined at every node. Our implementation of the techniques discussed in this subsection can be found in Appendix B.8.

Remark. *As the mesh changes due to the deletion of nodes then so does the boundary $\partial\Omega$ of our domain Ω. This of course raises a very important question: what happens to our boundary conditions? Well as we delete nodes in Matlab and alter the connectivity matrix we can use the inbuilt Matlab function "freeBoundary" [72] to find which nodes now define our boundary $\partial\Omega$ from our altered connectivity matrix. Our code then simply applies our boundary conditions on these new boundary nodes.*

5.3.3 Numerical algorithm

As we are creating a numerical solution for a multidimensional problem, introducing finite element solutions and allowing for the mesh to change as damage evolves, we must introduce a new numerical algorithm to compensate. In §4.2 we introduced Algorithm 1 which does solve the damage problem over a series of time steps but does not account for any change in the mesh. Our new algorithm will facilitate for a possibly changing mesh by adding a new nested iterations that account for the effects of changing the mesh at each time step and how that effects the solution to our damage problem. This is achieved by using the methods discussed in §5.3.2 and by defining a new step to our algorithm called "Update Mesh". The purpose of this step is to change the mesh given the current solution for the damage phase field.

Let us begin by considering the process that we would have to go through at each time step. Firstly, our time would be updated by a small positive step δt, this information will be used to update any time dependent boundary conditions. Our first step is to calculate the displacement, in the elastic phase, assuming that the damage is fixed from the previous time step. Next we would calculate the criterion for our new displacement which returns $\text{Crit}(\mathbf{X})$ a field providing information on which points are not satisfying the damage criterion. If all points satisfy the criterion we have no need for damage growth and simply move onto the next time step. However, if we do have a positive Crit at some points $\mathbf{X}$ then we must have damage growth. In that case we enter an iterative scheme to solve this time step. The first thing we must do is setup a new field called $\text{Torn}(\mathbf{X})$ as defined by (5.20). Once this is setup we enter a nested iterative step for calculating the solution for the damage, in the exact same fashion as explored in §4.2. Once this iterative method is satisfied we then use the solution for damage in the next step called "UpdateMesh". Here we calculate how to change the mesh depending on the solution of damage at every point $\mathbf{X} \in \Omega$. This provides a new solution for $\text{Torn}(\mathbf{X})$. We iteratively repeat this process until the mesh topology settles and the form of $\text{Torn}(\mathbf{X})$ converges. A pseudocode outline of this numerical algorithm can be found below in Algorithm 2.

In this algorithm our iterative scheme for finding the correct damage phase field runs till a desired tolerance is met. The tolerance should be appropriately set such that it is smaller than the order of the implemented finite elements. Our implementation of Algorithm 2 can be found in Appendix B.4.

5.3.4 The elastic phase

To solve the elastic phase, and calculate the displacement, in Algorithm 2 we consider the steady state problem (5.3a). To simplify the problem in our numerical scheme we are assuming that $\bar{\alpha}(\mathbf{X}, t)$ is fixed, and we use whatever solution we currently have for α in

Algorithm 2 Algorithm to calculate damage, displacement and torn points at each time step.

$t = t_0 + \delta t$
while $t \leq T$ **do**
  $\bar{\mathbf{u}}^{(k)} = \texttt{ElasticPhase}(\bar{\alpha}^{(k-1)}, t)$
  $\text{Crit} = \texttt{Criterion}(\bar{\mathbf{u}}^{(k)}, \bar{\alpha}^{(k-1)})$
  if any Crit ≥ 0 **then**
    $\text{Torn}^{(r)} =$ "guess"
    $\text{Torn}^{(r-1)} = \text{Torn}^{(k-1)}$
    while Norm $\left(\text{Torn}^{(r)} - \text{Torn}^{(r-1)}\right) \neq \mathbf{0}$ **do**
      $\text{Torn}^{(r-1)} \leftarrow \text{Torn}^{(r)}$
      $\bar{\alpha}^{(p)} =$ "guess"
      $\bar{\alpha}^{(p-1)} = \bar{\alpha}^{(k-1)}$
      while Norm $\left(\bar{\alpha}^{(p)} - \bar{\alpha}^{(p-1)}\right) \geq$ tolerance **do**
        $\bar{\alpha}^{(p)} \leftarrow \bar{\alpha}^{(p-1)}$
        $\bar{\alpha}^{(p)} = \texttt{DamagePhase}(\bar{\mathbf{u}}^{(k)}, \mathbf{Crit})$
        $\bar{\mathbf{u}}^{(k)} = \texttt{ElasticPhase}(\bar{\alpha}^{(p)}, t)$
        $\text{Crit} = \texttt{Criterion}(\bar{\mathbf{u}}^{(k)}, \bar{\alpha}^{(p)})$
      end while
      if $\text{Max}(\bar{\alpha}^{(p)}) = 1$ **then**
        $\text{Torn}^{(r)} = \text{UpdateMesh}(\bar{\alpha}^{(p)})$
      end if
    end while
    $\bar{\alpha}^{(k)} \leftarrow \bar{\alpha}^{(p)}$
    $\text{Torn}^{(k)} \leftarrow \text{Torn}^{(r)}$
  end if
  $t = t + \delta t$
end while

our iterations. In order to construct a finite element solution to this problem we need to rewrite the problem in the weak form. In Chapter 2, we can see lots of weak form problems that described both the displacement and damage. Here we will multiply equation (5.3a) by any admissible test function $\mathbf{P} \in \mathbb{R}^n$ and by integrating over the whole domain Ω we may write the weak form equation

$$\int_\Omega \left[\nabla_{\mathbf{X}} \cdot (g(\alpha)\mathbb{C} : \xi(\bar{\mathbf{u}}))\right] \cdot \mathbf{P} + \bar{\mathbf{B}} \cdot \mathbf{P}\, dV = 0. \tag{5.21}$$

By applying the product rule, the divergence theorem and our traction boundary conditions (5.5) we can transform (5.21) such that

$$\int_\Omega \nabla \mathbf{P} : \left(g(\bar{\alpha})\bar{\mathbb{C}} : \xi(\bar{\mathbf{u}})\right)\, dV = \int_\Omega \bar{\mathbf{B}} \cdot \mathbf{P}\, dV + \int_{\partial\Omega} \mathbf{t} \cdot \mathbf{P}\, dA. \tag{5.22}$$

All the terms on the right hand side of (5.22) are external loading terms, which are balanced by the stress term on the left hand side. To create a solution for the displacement here we are going to say that α is a fixed term. Next we want to write equation (5.22) in summation notation, where summation over double indices is implied, such that

$$\int_\Omega \frac{\partial \mathbf{P}_i}{\partial \mathbf{X}_j} g(\bar{\alpha}) \bar{\mathbb{C}}_{ijkl} \xi_{kl}(\bar{\mathbf{u}})\, dV = \int_\Omega \bar{\mathbf{B}}_i \cdot \mathbf{P}_i\, dV + \int_{\partial\Omega} \mathbf{t}_i \cdot \mathbf{P}_i\, dA. \tag{5.23}$$

Recalling the definition of the strain tensor ξ allows us to write

$$\begin{aligned} &\frac{1}{2}\int_\Omega \frac{\partial \mathbf{P}_i}{\partial \mathbf{X}_j} g(\bar{\alpha}) \bar{\mathbb{C}}_{ijkl} \frac{\partial \bar{\mathbf{u}}_k}{\partial \mathbf{X}_l}\, dV + \frac{1}{2}\int_\Omega \frac{\partial \mathbf{P}_i}{\partial \mathbf{X}_j} g(\bar{\alpha}) \bar{\mathbb{C}}_{ijkl} \frac{\partial \bar{\mathbf{u}}_l}{\partial \mathbf{X}_k}\, dV \\ &= \int_\Omega \bar{\mathbf{B}}_i \cdot \mathbf{P}_i\, dV + \int_{\partial\Omega} \mathbf{t}_i \cdot \mathbf{P}_i\, dA. \end{aligned} \tag{5.24}$$

This expression can be deeply simplified by exploiting the minor symmetries present in the macroscopic elasticity $\bar{\mathbb{C}}$, as shown in Theorem 3.3.2. Then by remembering that the indices i, j, k, l are all dummy variables we can write that

$$\int_\Omega \frac{\partial \mathbf{P}_i}{\partial \mathbf{X}_j} g(\bar{\alpha}) \bar{\mathbb{C}}_{ijkl} \frac{\partial \bar{\mathbf{u}}_k}{\partial \mathbf{X}_l}\, dV = \int_\Omega \bar{\mathbf{B}}_i \cdot \mathbf{P}_i\, dV + \int_{\partial\Omega} \mathbf{t}_i \cdot \mathbf{P}_i\, dA. \tag{5.25}$$

With expression (5.25) we can begin to create a linear system of equations that approximate a solution for $\bar{\mathbf{u}}$. To do this we need to first substitute in a specific form for $\mathbf{P}$. We choose $\mathbf{P} = P_I \mathbf{E}_I$, where $\mathbf{E}_I$ is an orthogonal basis vector of $\mathbb{R}^n$ and P_I is some arbitrary function as defined by (5.2). Using this form of $\mathbf{P}$ allows us to write down a linear solution for one dimension of the steady state problem defined by (5.25). Thus we have,

$$\int_\Omega \frac{\partial P_I}{\partial \mathbf{X}_j} g(\bar{\alpha}) \bar{\mathbb{C}}_{Ijkl} \frac{\partial \bar{\mathbf{u}}_k}{\partial \mathbf{X}_l}\, dV = \int_\Omega \bar{\mathbf{B}}_I P_I\, dV + \int_{\partial\Omega} \mathbf{t}_I P_I\, dA, \quad \forall I = 1, ..., n. \tag{5.26}$$

This can be thought of as the weak form of each equation of motion described by (5.3a). Now we can create an approximate solution for $\bar{\mathbf{u}}$ by substituting in the approximate interpolation defined by (5.1). Here we write that

$$P_I = p_I^{(a)} \quad \text{and} \quad \bar{\mathbf{u}}_k(\mathbf{X}, t) = \sum_{b=1}^{N} \bar{\mathbf{u}}_k^{(b)}(t) p_k^{(b)}(\mathbf{X}), \tag{5.27}$$

approximating our system in terms of the shape functions. By substitution we find that $\forall I = 1, \ldots, n$ and $\forall a = 1, \ldots, N$

$$\sum_{b=1}^{N} \int_\Omega \frac{\partial p_I^{(a)}}{\partial \mathbf{X}_j} g(\bar{\alpha}) \bar{\mathbb{C}}_{Ijkl} \frac{\partial p_k^{(b)}}{\partial \mathbf{X}_l}\, dV\, \bar{\mathbf{u}}_k^{(b)} = \sum_{b=1}^{N} \int_\Omega \bar{\mathbf{B}}_I p_I^{(b)}\, dV + \sum_{b=1}^{N} \int_{\partial\Omega} \mathbf{t}_I p_I^{(b)}\, dA. \tag{5.28}$$

The above expression appears daunting but we can easily write it as a simple linear system which solves for each term of $\bar{\mathbf{u}}_k^{(b)}$. It should be noted that (5.28) is a linear system in terms of $\bar{\mathbf{u}}_k^{(b)}$. We can write this down as the following linear system

$$\begin{bmatrix} \mathbb{K}_{11} & \dots & \mathbb{K}_{1R} & \dots & \mathbb{K}_{1n} \\ \vdots & \ddots & \vdots & & \vdots \\ \mathbb{K}_{I1} & \dots & \mathbb{K}_{II} & \dots & \mathbb{K}_{In} \\ \vdots & & \vdots & \ddots & \vdots \\ \mathbb{K}_{n1} & \dots & \mathbb{K}_{nR} & \dots & \mathbb{K}_{nn} \end{bmatrix} \begin{bmatrix} \mathbf{U}_1 \\ \vdots \\ \mathbf{U}_R \\ \vdots \\ \mathbf{U}_n \end{bmatrix} = \begin{bmatrix} \mathbf{T}_1 \\ \vdots \\ \mathbf{T}_I \\ \vdots \\ \mathbf{T}_n \end{bmatrix}. \tag{5.29}$$

Here the above expression has introduced new notation: each $\mathbb{K}_{IR}$ is an $N \times N$ matrix of integral coefficients on the left hand side of (5.28), each $\mathbf{U}_R$ is a vector of N elements each representing the R^{th} displacement elements on the left hand side of (5.28) and each $\mathbf{T}_I$ is a vector of N elements representing the terms on the right hand side of (5.28). Here R is defined for $R = 1, \dots, n$ and informs us what dimension of the displacement our integral coefficient matrix is interacting with. Our matrices $\mathbb{K}_{IR}$ have two possible values, the diagonal terms and the off diagonal terms in (5.29), defined as

$$\begin{aligned} \left(\mathbb{K}_{II}\right)_{ab} &= \int_\Omega \frac{\partial p_I^{(a)}}{\partial \mathbf{X}_j} g(\bar{\alpha}) \bar{\mathbb{C}}_{IjIl} \frac{\partial p_I^{(b)}}{\partial \mathbf{X}_l} \, dV \quad \forall I = 1, \dots, n, \\ \left(\mathbb{K}_{IR}\right)_{ab} &= \int_\Omega \frac{\partial p_I^{(a)}}{\partial \mathbf{X}_j} g(\bar{\alpha}) \bar{\mathbb{C}}_{IjRl} \frac{\partial p_R^{(b)}}{\partial \mathbf{X}_l} \, dV \quad \forall I, R = 1, \dots, n. \end{aligned} \tag{5.30}$$

Keep in mind that n is the dimension of Ω. It should also be noted that each

$$\mathbb{K}_{IR} = \mathbb{K}_{RI}^T, \tag{5.31}$$

this symmetry cuts the computing time of the coefficients in our linear matrix in half. This symmetry comes from the formulation of (5.28). We also define the notation

$$\left(\mathbf{T}_I\right)_b = \int_\Omega \bar{\mathbf{B}}_I p_I^{(b)} \, dV + \int_{\partial\Omega} \mathbf{t}_I p_I^{(b)} \, dA, \quad \forall I = 1, \dots, n. \tag{5.32}$$

These vectors account for the external loading at every node in our discretised domain. Finally, we define for each dimension

$$\mathbf{U}_R = \left[\bar{\mathbf{u}}_1^{(1)}, \dots, \bar{\mathbf{u}}_R^{(b)}, \dots, \bar{\mathbf{u}}_n^{(N)}\right]^T, \quad \forall R = 1, \dots, n. \tag{5.33}$$

With this linear system of equations we can calculate an approximate solution for the displacement. It is very easy to code this linear system and in general many finite element type software packages already exist which solve this type of problem, where we have no

damage evolution. Our implementation of this numerical scheme can be found in Appendix B.5.

Remark. *When implementing the integral coefficients* (5.30) *and the external loading vector* (5.32) *in code there is an important fact we must remember to save ourselves a lot of computational time. All of the piecewise shape functions labelled as* $p_I^{(b)}$ *do not need to be integrated over the full domain* Ω *when calculating our linear system of equations* (5.28)*. This is because the piecewise shape functions for each element are defined by* (5.2) *such that they are* 1 *at the central node of the element and 0 at every other node. We also do not need to calculate the integrals of the derivatives of shape functions over the full domain. This is because outside the finite element the shape function is a constant zero and thus the derivative with respect to* $\mathbf{X}_j$ *for* $j = 1, \ldots, n$ *is zero outside the finite element.*

To illustrate this we can define the subdomain for each finite element $\Omega_E^{(b)} \subset \Omega$ *for all* $b = 1, \ldots, N$*. If we integrate any shape function* $p_I^{(b)}$ *over the domain* Ω *then we find that*

$$\int_\Omega p_I^{(b)}\, dV = \int_{\Omega_E^{(b)}} p_I^{(b)}\, dV \quad \text{and} \quad \int_\Omega \frac{\partial p_I^{(b)}}{\partial \mathbf{X}_j}\, dV = \int_{\Omega_E^{(b)}} \frac{\partial p_I^{(b)}}{\partial \mathbf{X}_j}\, dV. \tag{5.34}$$

Using the fact illustrated by (5.34) *will save us a lot of computational time because we do not need to calculate the integrals in* (5.30) *and* (5.32) *over the full domain but only subsets of the full domain. This efficiency trick is very present throughout our code as can be seen in Appendix B.*

5.3.5 The damage criterion

The n-dimensional damage criterion (5.4) tells us in our iterative process where damage growth is taking place. So after calculating the displacement field, we must check whether our current displacement field and damage field satisfy the damage criterion. We have already done this for a one dimensional damage criterion in §4.2. Similarly, what we are calculating is the criterion field we call "Crit" for our n-dimensional system. This is done such that such that $\text{Crit}(\mathbf{X}) = 1$ anywhere the damage field is not satisfied and $\text{Crit}(\mathbf{X}) = 0$ anywhere the damage field is satisfied. It should be noted that $\text{Crit}(\mathbf{X})$ can also be thought of as an indicator function just like the torn function (5.20). In other words we are defining a piecewise function such that

$$\text{Crit}(\mathbf{X}) = \begin{cases} 1, & F > 0 \\ 0, & F \leq 0 \end{cases} \tag{5.35}$$

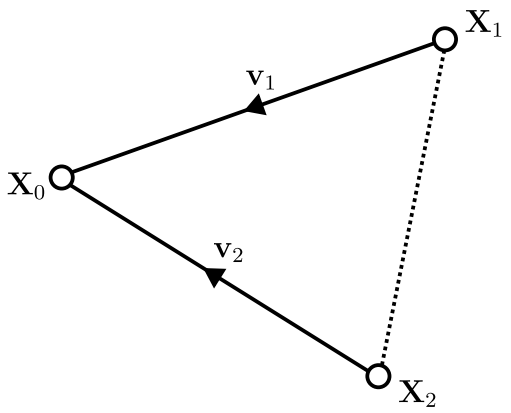


Figure 5.3: A schematic of a triangle with three vertices defined by position vectors. The position vectors are $\mathbf{X}_j$ for $j = 0, 1, 2$. On this triangle we have illustrated two directional vectors $\mathbf{v}_1$ and $\mathbf{v}_2$, that transport you from $\mathbf{X}_1$ or $\mathbf{X}_2$ towards $\mathbf{X}_0$, respectively. Using these position vectors and directional vectors we can exploit the definition of a directional derivative to calculate the regular gradient of a function defined at the positional vectors.

where F is shorthand notation for the damage criterion (5.4). We can write this as

$$F(\bar{\mathbf{u}}, \alpha) = -g'(\bar{\alpha}) \left(\frac{1}{2} \xi_{\mathbf{X}}(\bar{\mathbf{u}}) : \bar{\mathbb{C}} : \xi_{\mathbf{X}}(\bar{\mathbf{u}}) - \bar{\psi} \right) - \bar{G}\bar{\alpha} + \nabla_{\mathbf{X}} \cdot \left(\bar{\mathbb{D}} \nabla_{\mathbf{X}} \bar{\alpha} \right). \tag{5.36}$$

To implement this in software one can easily discretize $\mathrm{Crit}(\mathbf{X})$ and F, then substituting in our assumed solutions of $\bar{\mathbf{u}}$ and $\bar{\alpha}$ will return a solution. In a discretized solution using finite differences, Crit will simply be a long vector of 1's and 0's. Essentially a binary string that will inform our iterative scheme, in Algorithm 2, where we have to calculate a new solution for α. This method can easily be extended to higher dimensions where we would use indices values to indicate locations in some discretized domain. Similar to Chapter 4 we implemented this technique in code, which can be found in Appendix B.6.

In one-dimension calculating derivatives is quite a straight forward process, we would use methods of forward, centre or backward differences. This is done on a mesh, that is usually uniform, but not necessarily. Importantly, in one dimension we can only travel along the same line. In n-dimensions this is not the case and when using the finite element methods there is absolutely no guarantee of mesh uniformity. In the code we implemented we used triangles to generate a mesh of Ω. Consider a triangle with three vertices defined by the points $\mathbf{X}_i$ for $i = 0, 1, 2$. This simple set-up is illustrated in Figure 5.3. We can generally calculate the derivatives of the displacement or damage at $\mathbf{X}_0$ using the directional derivative. This is done by exploiting the definition of the directional derivative

$$\partial_{\mathbf{v}} f = \nabla f \cdot \frac{\mathbf{v}}{|\mathbf{v}|}, \tag{5.37}$$

for some scalar function $f = f(\mathbf{X})$ in the direction of the vector $\mathbf{v}$ which is a vector connecting two vertices of our standard triangle. If we want to calculate derivatives at the

vertex $\mathbf{X}_0$ then we need to define two vectors. Let us define

$$\mathbf{v}_1 = \mathbf{X}_0 - \mathbf{X}_1 \quad \text{and} \quad \mathbf{v}_2 = \mathbf{X}_0 - \mathbf{X}_2, \tag{5.38}$$

which are the direction vectors taking us from each of the other vertices to $\mathbf{X}_0$, as seen on Figure 5.3. Next we can define two directional derivatives

$$\partial_{\mathbf{v}_1} f = \nabla f \cdot \frac{\mathbf{v}_1}{|\mathbf{v}_1|} \quad \text{and} \quad \partial_{\mathbf{v}_2} f = \nabla f \cdot \frac{\mathbf{v}_2}{|\mathbf{v}_2|}, \tag{5.39}$$

which can be used to form a linear system thus allowing us to calculate ∇f. Doing so we can write that

$$\begin{bmatrix} \frac{1}{|\mathbf{v}_1|}\mathbf{v}_1^T \\ \frac{1}{|\mathbf{v}_2|}\mathbf{v}_2^T \end{bmatrix} \nabla f = \begin{bmatrix} \partial_{\mathbf{v}_1} f \\ \partial_{\mathbf{v}_2} f \end{bmatrix}. \tag{5.40}$$

Calculating the directional derivatives in the linear system (5.40) can be implemented in code very easily. Assuming the triangle is adequately small then the lengths of the triangle edges are small enough to simply perform a forward difference. If we then define the value of f at each node as f_0, f_1 and f_2 then we can approximate the directional derivatives as

$$\partial_{\mathbf{v}_1} f \approx \frac{f_0 - f_1}{|\mathbf{v}_1|} + \mathcal{O}(|\mathbf{v}_1|) \quad \text{and} \quad \partial_{\mathbf{v}_2} f \approx \frac{f_0 - f_2}{|\mathbf{v}_2|} + \mathcal{O}(|\mathbf{v}_2|). \tag{5.41}$$

Substituting the approximation (5.41) into (5.40) gives us a simplified linear system

$$\begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \end{bmatrix} \nabla f = \begin{bmatrix} f_0 - f_1 \\ f_0 - f_2 \end{bmatrix}, \tag{5.42}$$

using this linear system we can quickly calculate ∇f. To improve the result of this calculation in code we can calculate the derivative of a node using every triangle that contains it and then take a weighted average based on the area of these triangles. Another way to improve the accuracy of this result is to use shape functions of triangles defined by more than three nodes. If each triangle was defined by six nodes, three nodes for the vertices and three for the midpoints of each edge, then we could do a second order accurate approximation of the directional derivatives.

This methodology is used consistently in our numerical methods, for calculating derivatives of known values. For example when calculating the damage criterion with known values and for calculating the driving force in the damage phase. The implementation of this technique can be seen in Appendix B.6. The accuracy of this method improves as the size of the elements decreases, similar to how finite differences improves as the space between nodes decreases.

It should be noted the above example calculation (5.42) was done with a triangle

on a plane in mind. Which is only a two dimensional setup but this method can be easily extended to higher dimensions where finite elements are constructed out of more triangles accounting for the extra dimensions to be calculated in ∇f. For example in three dimensions the finite elements are constructed out of tetrahedrons.

5.3.6 The damage phase

In our numerical solution to calculate $\bar{\alpha}$ in higher dimensions we again need to use an iterative solution just as we did in §4.2. Here though we are dealing with the rate dependent version of the model and will not be able to implement Newtons method while using the finite element method. Instead we will use a Crank-Nicholson scheme [107] to calculate steps forward in time. Though as we are fixing $\bar{\mathbf{u}}$ as a constant that we know, that vastly simplifies the problem. To begin consider the damage evolution equation (5.3b) such that

$$\eta\dot{\bar{\alpha}} = \langle F(\bar{\alpha};\bar{\mathbf{u}})\rangle_+ , \tag{5.43}$$

here F is as defined in (5.36) and this problem is just the rate dependent damage problem, with a fixed displacement $\bar{\mathbf{u}}$. Recall that the angular brackets in our damage evolution equation are Macaulay brackets (2.48) as defined in Chapter 2 §2.2.4 on page 61. Using the Crank-Nicholson method we can discretise the damage evolution equation in time such that

$$r\left(\bar{\alpha}^{(k)} - \bar{\alpha}^{(k-1)}\right) = \langle F(\bar{\alpha};\bar{\mathbf{u}})\rangle_+^{(k)} + \langle F(\bar{\alpha};\bar{\mathbf{u}})\rangle_+^{(k-1)} . \tag{5.44}$$

In this setup we have defined $r(\mathbf{X}) = 2\eta(\mathbf{X})/\Delta t$ and the superscript indicates the discretisation in time. Here Δt is of course just the difference in time such that $\Delta t = t^{(k)} - t^{(k-1)}$. The time step k is the current time step which we want to calculate and $k-1$ is the previous time step which we have a solution for.

The next step in solving the damage evolution equation is to create an iterative scheme which solves (5.44). To do this we multiply (5.44) by any admissible test function $P(\mathbf{X})$ and integrate over Ω. Doing so we have the weak form

$$\int_\Omega P\left[r\left(\bar{\alpha}^{(k)} - \bar{\alpha}^{(k-1)}\right) - \langle F(\bar{\alpha};\bar{\mathbf{u}})\rangle_+^{(k)} - \langle F(\bar{\alpha};\bar{\mathbf{u}})\rangle_+^{(k-1)}\right] dV = 0. \tag{5.45}$$

The next step is to introduce a discretisation of $P(\mathbf{X})$ and $\bar{\alpha}(\mathbf{X},t)$, we do this using the formulations defined by (5.1). One can therefore say that

$$P(\mathbf{X}) = p_i(\mathbf{X}) \quad \text{and} \quad \bar{\alpha}^{(k)}(\mathbf{X}) = \sum_j^N \bar{\alpha}_j^{(k)} p_j(\mathbf{X}). \tag{5.46}$$

Substitution into (5.45) then gives us the weak form

$$\sum_{j=1}^{N}\left(\bar{\alpha}_j^{(k)}-\bar{\alpha}_j^{(k-1)}\right)\int_\Omega rp_ip_j\,dV-\int_\Omega p_i\left(\langle F(\bar{\alpha};\bar{\mathbf{u}})\rangle_+^{(k)}+\langle F(\bar{\alpha};\bar{\mathbf{u}})\rangle_+^{(k-1)}\right)dV=0. \quad (5.47)$$

This is close to being a numerical system we can solve, we need to consider how to handle the Macaulay brackets in this expression. Since we know the solutions for the previous time step we can define

$$\langle F(\bar{\alpha};\bar{\mathbf{u}})\rangle_+^{(k-1)}=\tilde{f}.$$

If we next interpolate the function $\tilde{f}$ using our shape functions we can write

$$\tilde{f}=\sum_{j=1}^{N}\tilde{f}_jp_j(\mathbf{X}), \quad (5.48)$$

similar to how we interpolate the functions $\bar{\alpha}$ and $\bar{\mathbf{u}}$, see (5.1). If we substitute this into (5.47) we are now left with

$$\sum_{j=1}^{N}\left(\bar{\alpha}^{(k)}-\bar{\alpha}^{(k-1)}\right)\int_\Omega rp_ip_j\,dV-\tilde{f}_j\int_\Omega p_ip_j\,dV-\int_\Omega p_i\left\langle F(\bar{\alpha};\bar{\mathbf{u}})\right\rangle_+^{(k)}dV=0. \quad (5.49)$$

So now, as we can see, the last step is to deal with the final Macaulay bracket term in (5.49). To do this we implement the Crit($\mathbf{X}$) function, it will inform us where we should or should not solve the damage problem. Here we are going to assume we have to solve for a positive damage criterion and hence can drop the Macaulay brackets. We will explain later how to handle cases where we do not have damage evolution. So for now we may simplify (5.49) and say that

$$\sum_{j=1}^{N}\left(\bar{\alpha}^{(k)}-\bar{\alpha}^{(k-1)}\right)\int_\Omega rp_ip_j\,dV-\tilde{f}_j\int_\Omega p_ip_j\,dV-\int_\Omega p_iF^{(k)}(\bar{\alpha};\bar{\mathbf{u}})\,dV=0. \quad (5.50)$$

The last part of this problem now is to simply discretise $F^{(k)}$ which we can do by considering (5.36). From (5.36) we can immediately say that

$$\begin{aligned}\int_\Omega p_iF^{(k)}(\bar{\alpha};\bar{\mathbf{u}})\,dV&=\int_\Omega p_i\left[-g'(\bar{\alpha}^{(k)})Q(\bar{\mathbf{u}}^{(k)})-\bar{G}\bar{\alpha}^{(k)}+\nabla\cdot\left(\bar{\mathbb{D}}\nabla\bar{\alpha}^{(k)}\right)\right]dV\\&=\int_\Omega p_i\left[-g'(\bar{\alpha}^{(k)})Q(\bar{\mathbf{u}}^{(k)})-\bar{G}\bar{\alpha}^{(k)}\right]-\left(\bar{\mathbb{D}}\nabla\bar{\alpha}^{(k)}\right)\cdot\nabla p_i\,dV\end{aligned}$$

Here we have applied the product rule, the divergence theorem and applied our no flux boundary conditions (5.6) for the damage variable. We have also introduced the driving

force $Q(\bar{\mathbf{u}})$, defined by the elastic potential and threshold function,

$$Q(\bar{\mathbf{u}}) = \frac{1}{2}\xi_{\mathbf{X}}(\bar{\mathbf{u}}) : \bar{\mathbb{C}} : \xi_{\mathbf{X}}(\bar{\mathbf{u}}) - \bar{\psi}. \tag{5.51}$$

If we next substitute in our discretisation of $\bar{\alpha}$ (5.46) then its clear that

$$\begin{aligned}\int_\Omega p_i F^{(k)}(\bar{\alpha}; \bar{\mathbf{u}})\, dV = &- \sum_{j=1}^{N} \bar{\alpha}_j^{(k)} \int_\Omega \bar{G} p_i p_j \, dV - \sum_{j=1}^{N} \bar{\alpha}_j^{(k)} \int_\Omega \left(\bar{\mathbb{D}}\nabla p_j\right) \cdot \nabla p_i \, dV \\ &- \sum_{j=1}^{N} \int_\Omega p_i g'(\bar{\alpha}^{(k)}) Q(\bar{\mathbf{u}}^{(k)})\, dV.\end{aligned}$$

Notice we have not substituted an approximation for the derivative of the degradation function. That is because the function is not necessarily linear. Therefore we need to introduce an approximation of this function. What we do here is setup our iterative scheme by creating an approximation of $g'(\bar{\alpha}^{(k)})$ based on a previous iteration. Therefore we define the following iterative approximation

$$g'(\bar{\alpha}^{(k)}) = \sum_{j=1}^{N} \tilde{g}_j^{(k,m-1)} p_j \quad \text{where} \quad \tilde{g}_j^{(k,m-1)} = g'(\bar{\alpha}_j^{(k,m-1)}), \tag{5.52}$$

where k indicates the time step and m indicates the iterative step in our solution. If we put everything together we can write down the following iterative scheme

$$\begin{aligned}&\sum_{j=1}^{N} \left[\bar{\alpha}_j^{(k,m)} \int_\Omega \left(r + \bar{G}\right) p_i p_j \, dV + \bar{\alpha}_j^{(k,m)} \int_\Omega \left(\bar{\mathbb{D}}\nabla p_j\right) \cdot \nabla p_i \, dV \right] \\ &= \sum_{j=1}^{N} \left[\tilde{f}_j \int_\Omega p_i p_j \, dV + \bar{\alpha}_j^{(k-1)} \int_\Omega r p_i p_j \, dV - \tilde{g}_j^{(k,m-1)} \int_\Omega Q(\bar{\mathbf{u}}^{(k)}) p_i p_j \, dV \right].\end{aligned} \tag{5.53}$$

One can use this iterative system to calculate each term of $\bar{\alpha}_j^{(k)}$ at the time step $k \; \forall \, j$. For convenience we can convert this into a simple linear system, which we do by writing

$$\mathbb{B}\mathbf{A}^{(m)} = -\mathbb{H}_1 \mathbf{g}^{(m-1)} + \mathbb{H}_2 \mathbf{f} + \mathbb{H}_3 \mathbf{a}. \tag{5.54}$$

This linear system defined by matrices and vectors is convenient for coding in Matlab. Here though we have defined the following vectors at step m of our iteration as

$$\mathbf{A}^{(m)} = \begin{bmatrix} \vdots \\ \bar{\alpha}_j^{(k,m)} \\ \vdots \end{bmatrix} \quad \mathbf{g}^{(m-1)} = \begin{bmatrix} \vdots \\ \tilde{g}_j^{(k,m-1)} \\ \vdots \end{bmatrix} \quad \mathbf{f} = \begin{bmatrix} \vdots \\ \tilde{f}_j \\ \vdots \end{bmatrix} \quad \mathbf{a} = \begin{bmatrix} \vdots \\ \bar{\alpha}_j^{(k-1)} \\ \vdots \end{bmatrix}. \tag{5.55}$$

Next we have four matrices making up all the coefficients of this linear system. These can be written as

$$(\mathbb{B})_{ij} = \int_\Omega \left(r + \bar{G}\right) p_i p_j + \left(\bar{\mathbb{D}}\nabla p_j\right) \cdot \nabla p_i \, dV \tag{5.56a}$$

$$(\mathbb{H}_1)_{ij} = \int_\Omega Q(\bar{\mathbf{u}}^{(k)}) p_i p_j \, dV \tag{5.56b}$$

$$(\mathbb{H}_2)_{ij} = \int_\Omega p_i p_j \, dV \tag{5.56c}$$

$$(\mathbb{H}_3)_{ij} = \int_\Omega r p_i p_j \, dV \tag{5.56d}$$

We would run this iterative scheme for $M \in \mathbb{N}$ iterations until the values of $\bar{\alpha}^{(k,m)}$ and $\bar{\alpha}^{(k,m-1)}$ converge to an acceptable tolerance. The appropriate way to implement this iterative scheme in the context of simulating a real scenario with material failure is outlined in Algorithm 2.

To correctly implement this numerical scheme for the damage phase field we need to use the field $\text{Crit}(\mathbf{X})$. With the criterion field we know where this numerical scheme should and should not be implemented in our domain Ω. However, a problem that will occur if we implement the above numerical scheme for the damage phase field in its present form as defined by (5.54), with the damage evolution informed by the criterion field, is that our above numerical scheme will actually fail to converge. This happens because the criterion field controls which region in Ω where damage evolution can occur, which restricts the diffusion of damage to that strict region. If that region is too small then the diffusive profile of the damage phase field cannot converge using the finite element method. In Chapter 4 we do not have this same issue, this is because in one dimension we use a finite difference method which can correctly converge even if the damage evolution is only allowed to occur at a single node in our system.

We therefore cannot follow the same methodology used in Chapter 4, laid out in §4.2.4, to implement our finite element method. So in dimensions greater than one in order for the damage phase field to correctly converge using the above numerical scheme we need to allow the damage phase field to diffuse throughout our domain. To do this we need to consider what terms in the damage evolution controls the diffusion and shape of the damage phase field. We can isolate this from the tearing energy and the diffusion term in (5.56a). Therefore at every point $\text{Crit}(\mathbf{X}) = 0$ we will solve the linear problem

$$\mathbb{B}\mathbf{A}^{(m)} = \mathbf{0}. \tag{5.57}$$

In the rows where we have $\text{Crit}(\mathbf{X}) = 0$ we will have that

$$(\mathbb{B})_{ij} = \int_\Omega \bar{G} p_i p_j + \left(\bar{\mathbb{D}} \nabla p_j\right) \cdot \nabla p_i \, dV. \tag{5.58}$$

In summary the iterative scheme we will implement must make use of the criterion field and also the torn field. We write the iterative scheme (5.54) as

$$\mathbb{B}\mathbf{A}^{(m)} = -\mathbb{H}_1 \mathbf{g}^{(m-1)} + \mathbb{H}_2 \mathbf{f} + \mathbb{H}_3 \mathbf{a}, \tag{5.59}$$

where the vectors $\mathbf{A}^{(m)}$, $\mathbf{g}^{(m-1)}$, $\mathbf{f}$ and $\mathbf{a}$ are defined by (5.55). We run this iterative scheme until $\mathbf{A}^{(m)}$ and $\mathbf{A}^{(m-1)}$ converge to an acceptable tolerance. To implement the criterion and torn fields into this iterative system we define the elements of the matrices in (5.59) to be piecewise functions. Here we define them to be

$$(\mathbb{B})_{ij} = \begin{cases} 1 \quad \text{if} \quad \text{Torn}(\mathbf{X}) = 1, \\ \int_\Omega \left(r + \bar{G}\right) p_i p_j + \left(\bar{\mathbb{D}} \nabla p_j\right) \cdot \nabla p_i \, dV \quad \text{if} \quad \text{Crit}(\mathbf{X}) = 1, \\ \int_\Omega \bar{G} p_i p_j + \left(\bar{\mathbb{D}} \nabla p_j\right) \cdot \nabla p_i \, dV \quad \text{if} \quad \text{Crit}(\mathbf{X}) = 0, \end{cases} \tag{5.60a}$$

$$(\mathbb{H}_1)_{ij} = \begin{cases} 0 \quad \text{if} \quad \text{Crit}(\mathbf{X}) = 0 \quad \text{or} \quad \text{Torn}(\mathbf{X}) = 1, \\ \int_\Omega Q(\bar{\mathbf{u}}^{(k)}) p_i p_j \, dV \quad \text{if} \quad \text{Crit}(\mathbf{X}) = 1, \end{cases} \tag{5.60b}$$

$$(\mathbb{H}_2)_{ij} = \begin{cases} 0 \quad \text{if} \quad \text{Crit}(\mathbf{X}) = 0 \quad \text{or} \quad \text{Torn}(\mathbf{X}) = 1, \\ \int_\Omega p_i p_j \, dV \quad \text{if} \quad \text{Crit}(\mathbf{X}) = 1, \end{cases} \tag{5.60c}$$

$$(\mathbb{H}_3)_{ij} = \begin{cases} 1 \quad \text{if} \quad \text{Torn}(\mathbf{X}) = 1, \\ \int_\Omega r p_i p_j \, dV \quad \text{if} \quad \text{Crit}(\mathbf{X}) = 1, \\ 0 \quad \text{if} \quad \text{Crit}(\mathbf{X}) = 0. \end{cases} \tag{5.60d}$$

Defining these matrix elements in terms of piecewise functions makes it far simpler to create a code. Notice that for nodes that are completely damaged we have setup our iterative scheme (5.59) so that the node stays constant as $\bar{\alpha} = 1$. This was implemented in the above form of the matrices (5.60) for the cases where $\text{Torn}(\mathbf{X}) = 1$. We implemented the above iterative scheme in Matlab, our code can be found in Appendix B.7.

Remark. *When we implement the scheme discussed in this subsection we need it in code to be as efficient as possible. To do that we made use of two main techniques: sparse matrices and adaptive time steps. Sparse matrices allow us to store large matrices in memory by only storing data about non-zero elements in the matrix. This is done by storing the rows, columns and elements as three column vectors. Sparse matrices are far more efficient than regular matrices when the matrices are not very dense. In our case each row represents the interactions a single node in our mesh has and the columns represent all the other*

nodes it is interacting with. If we have a hundred nodes in our system and each node is only connected in our Delaunay triangulation to around 12 other nodes then the matrix will be at least 80% empty. Generally then, when using finite element methods it is best to always set up our mass matrices in our numerical schemes as sparse matrices. Another common issue with iterative schemes is the quality of the initial guess that we make. In our iterative scheme (5.59) *our initial guess will be the current or previous solution for the damage phase field. If the damage phase field needs to make a large sudden growth then it may not converge. To deal with this if the solution does not converge we perform an adaptive time step. This is done by taking a smaller time step forward and we can keep making the time steps smaller until our numerical scheme converges. Of course if the time step becomes to small we simply terminate the iterative scheme.*

5.4 The trouser test

Now that we have explained our numerical methods, and have a code as can be seen in Appendix B, the first thing we should do is a test case. In damage mechanics a classic toy simulation is the trouser test. A trouser test consists of a simple block square or rectangle that is clamped at one end and at the other end has the top half pulled upwards and the bottom half pulled downwards. As a result the block should get torn along the middle tearing the block into the approximate shape of trousers, hence a trouser test. Figure 5.4 outlines this toy simulation. To be clear we are gripping two regions of Ω and pulling one upwards and one downwards.

Since we are doing a toy simulation here and want to illustrate the full utility of our macroscopic model then we will consider two simulations with comparatively different microscopic geometries. This will be a useful example of how small changes in the microscale can effect the macroscale. To do this we will consider two cases with the exact same material properties but differing microscopic geometries. The first microscopic geometry we will consider is a case of laminated vertical strips and the second will be of a host matrix with a singular circular inclusion. In each case we have two subdomains forming the microscopic cell and both subdomains will encompass exactly half of the microscopic domain.

In the rest of this section we will introduce the microscopic domains, their physical properties and calculate the related effective macroscopic coefficients. After which we will compare the outcome of each simulation. For a simple comparison we will define a two dimensional local cell $\omega := [0,1] \times [0,1]$ with two subdomains ω_A and ω_B. Here we have that $|\omega| = 1$ such that $|\omega_A| = 0.5$ and $|\omega_B| = 0.5$. All the physical properties of ω_A and ω_B will be exactly the same. Except, for the elasticity instead ω_A will be a stiff material and ω_B will be a comparatively elastic material. So for the rest of this section let us define all

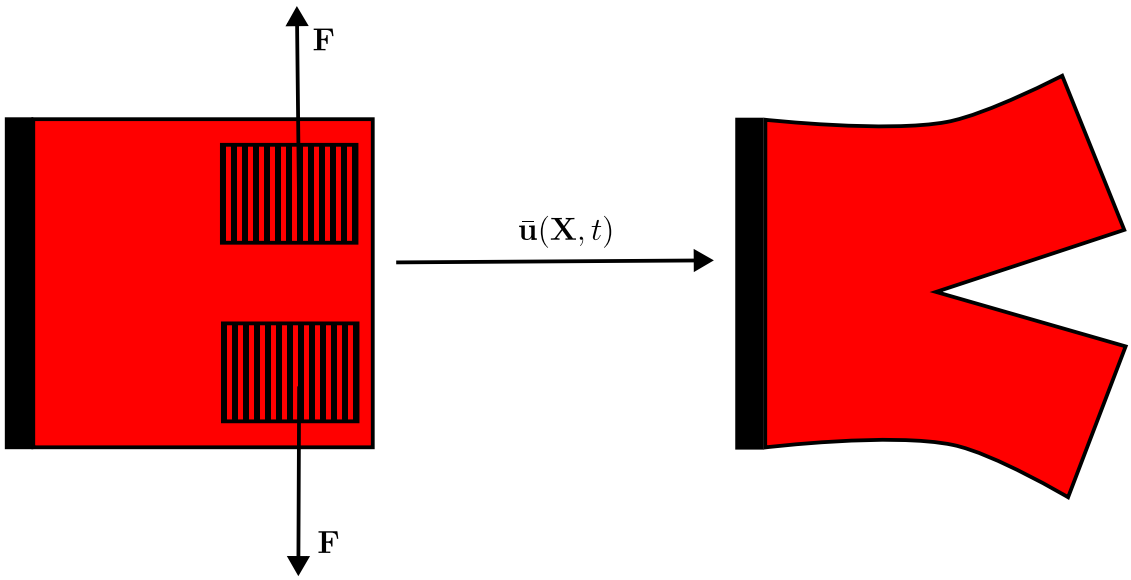


Figure 5.4: In the above figure we have a diagram of the trouser test. On the left we have a referential domain of some block, which is clamped on its left boundary and on its right we can striped regions where forces are being applied to pull the block up an down. On the right we have the current domain where the block has been displaced and torn down the middle by the forces pulling it apart.

of microscopic material properties as:

$$\begin{aligned} &\rho_A = \rho_B = 10, \quad d_A = d_B = 0.05, \quad \psi_A = \psi_B = 0.01, \\ &\mu_A = 15 \times 10^4, \quad \lambda_A = 15 \times 10^4, \quad \mu_B = 10^4, \quad \lambda_B = 10^4, \\ &G_A = G_B = 1 \quad \text{and} \quad \eta_A = \eta_B = 1. \end{aligned} \tag{5.61}$$

We can instantly calculate all the effective coefficients for all the material coefficients except for the elasticity. Let us write them down as

$$\begin{aligned} &\bar{\rho} = 10, \quad \bar{\psi} = 0.01, \quad \bar{G} = 1, \\ &\bar{\eta} = 1 \quad \text{and} \quad \bar{\mathbb{D}} = 0.05\mathbb{I}_{2\times 2}, \end{aligned} \tag{5.62}$$

corresponding to the coefficients in the macroscopic model (5.3). Here $\mathbb{I}_{2\times 2}$ is the identity matrix of order 2. Finally for both of our simulations we will need to apply some external loading. As mentioned in the trouser test we will apply external loading on the right, where the top half of our block is pulled upwards and the bottom half is pulled downwards. We won't apply any surface stress and instead apply the following uniform microscopic body

force as loading

$$\mathbf{b}_A = \mathbf{b}_B = \begin{cases} 500t\mathbf{E}_2, & \forall \mathbf{X} \in [0.8, 1] \times (0.5, 1] \\ -500t\mathbf{E}_2, & \forall \mathbf{X} \in [0.8, 1] \times [0, 0.5) \\ \mathbf{0}, & \text{otherwise,} \end{cases} \tag{5.63}$$

where $\mathbf{E}_2$ is a vertical unit vector in the reference domain. Since the density and body forces are uniform on the microscale then we can define the effective body force to be

$$\bar{\mathbf{B}} = \bar{\rho}\mathbf{b}_A = \bar{\rho}\mathbf{b}_B. \tag{5.64}$$

Here our external loading (5.63) is not smooth across the domain Ω, which makes the governing equations across the regions where forces are applied and not applied discontinuous. This certainly leads to a number of problems when trying to solve any diffusive model analytically, it can also create issues when solving models numerically. In our case we have not had any issues performing simulations with such sharp external loading terms but when implementing our model generally we advice making considerate choices of smooth external loading functions. The other form of external loading we can apply is surface traction. For this test case we will simply set the traction to zero everywhere on $\partial\Omega$ such that

$$\mathbf{t} = \mathbf{0} \quad \text{on } \partial\Omega. \tag{5.65}$$

Lastly, we need to define a damage degradation function for our damage model (5.3). For ease we will make the simplest choice

$$g(\alpha) = (1 - \alpha)^2, \tag{5.66}$$

which has the derivative

$$g'(\alpha) = -2(1 - \alpha). \tag{5.67}$$

We now have everything we need to setup our two local cell problems calculate their effective elasticity tensors and perform the simulations. After which we can compare the results. We must also clarify what the threshold is for considering a node to be torn. In other words how close does $\alpha(\mathbf{X})$ at that point $\mathbf{X} \in \Omega$ need to be to one. In the following simulations we set a point $\mathbf{X} \in \Omega$ as being torn if $\alpha \geq 0.97$. We can therefore alter the definition of $\text{Torn}(\mathbf{X})$, (5.20), to be

$$\text{Torn}(\mathbf{X}) = \begin{cases} 1, & \bar{\alpha}(\mathbf{X}) \geq 0.97, \\ 0, & \text{otherwise.} \end{cases}$$

5.4.1 Vertical strip setup

The first microscopic geometry we are interested in is laminated vertical strips. Consider a square cell ω composed of two laminated vertical strips ω_A and ω_B, respectively. For simplicity we will give each subdomain all the same physical properties except for the elasticity. Instead ω_A will be a stiff material and ω_B will be a much more compliant material. An illustration of this setup can be see in Figure 5.5.

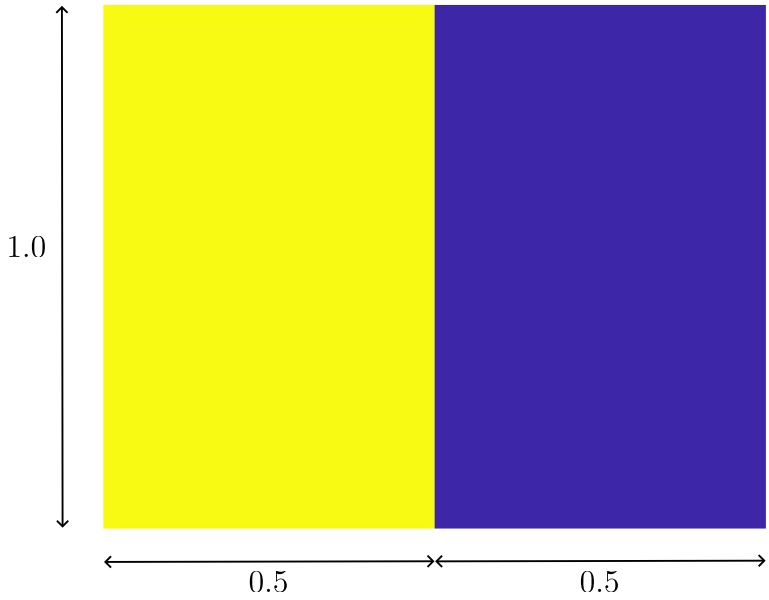


Figure 5.5: Two dimensional representation of the microscopic geometry. In this case we have two laminated vertical strips. Here the blue strip is much stiffer than the yellow strip.

Using our code and the methods discussed in §5.2.1 alongside the geometry provided in Figure 5.5 we can calculate the effective elasticity tensor. Using the Lamé parameters provided in (5.61) allows us to quickly find the effective elasticity tensor. However, before we calculate this effective elasticity tensor let us discuss some notation for clarity. As we are working in two dimensions with a fourth order elasticity tensor that has minor and major symmetries then we can write this down in the following shorthand notation

$$\bar{\mathbb{C}} = \begin{bmatrix} \bar{\mathbb{C}}_{1111} & \bar{\mathbb{C}}_{1122} & \bar{\mathbb{C}}_{1112} \\ \bar{\mathbb{C}}_{2211} & \bar{\mathbb{C}}_{2222} & \bar{\mathbb{C}}_{2212} \\ \bar{\mathbb{C}}_{1211} & \bar{\mathbb{C}}_{1222} & \bar{\mathbb{C}}_{1212} \end{bmatrix}, \tag{5.68}$$

which is a convenient way to write the tensor down as a representative 3×3 symmetric matrix which encapsulates the minor and major symmetries in our fourth order tensor.

Using this shorthand notation we can calculate our effective elasticity tensor to be

$$\bar{\mathbb{C}} \approx \begin{bmatrix} 56250 & 18750 & 0 \\ 18750 & 219583 & 0 \\ 0 & 0 & 18750 \end{bmatrix}, \tag{5.69}$$

Here we rounded to whole numbers. It illustrates that effective elasticity coefficients calculated from isotropic elasticity coefficients are not also necessarily isotropic. You can also intuitively explain the material properties of (5.69) by imagining how external loading to the setup in Figure 5.5 would play out. Notice that $\bar{\mathbb{C}}_{1111}$ is significantly smaller than $\bar{\mathbb{C}}_{2222}$. That is because if we apply loading purely in the horizontal direction the stiff component ω_A will not be as compliant as ω_B, therefore ω_A will resist stretching in the horizontal direction whilst ω_B begins to stretch just like an accordion. If we only applied external loading in the vertical direction then the deformation would predominantly be in the domain of, the soft material, ω_B along the direction parallel to the pulling. Whereas the stiff material ω_A will not be as compliant and thus not allow the material to stretch vertically. In terms of material shearing the geometry present in Figure 5.5 does not have any preferential bias hence why $\bar{\mathbb{C}}_{1122} = \bar{\mathbb{C}}_{1212}$.

5.4.2 Inclusion setup

The second microscopic geometry we are interested in is a square cell ω consisting of a host matrix ω_A with a circular inclusion ω_B. We can see an illustration of this setup in Figure 5.6. Again we will assume all material properties are uniform, except the elasticity, as described in (5.61).

Using our code and the methods described in §5.2.1 alongside the geometry provided in Figure 5.6 will allow us to calculate the effective elasticity tensor. We find that

$$\bar{\mathbb{C}} \approx \begin{bmatrix} 159484 & 35557 & 0 \\ 35557 & 159484 & 0 \\ 0 & 0 & 30840 \end{bmatrix}, \tag{5.70}$$

which is another interesting solution, here we rounded to whole numbers and made use of the shorthand notation defined by (5.68). Here in (5.70) we can see that in this case that the effective elasticity is also no longer isotropic. This can be explained by noting that the microscopic geometry in Figure 5.6 is obviously not isotropic. Again we can intuitively explain the material properties of (5.70) by imagining how external loading to the setup in Figure 5.6 would play out. Notice that $\bar{\mathbb{C}}_{1111} = \bar{\mathbb{C}}_{2222}$ which is because our microscopic geometry in Figure 5.6 has a rotational symmetry of 90° degrees. The most interesting observation we can make is how different (5.70) is from (5.69). Both microscopic domains

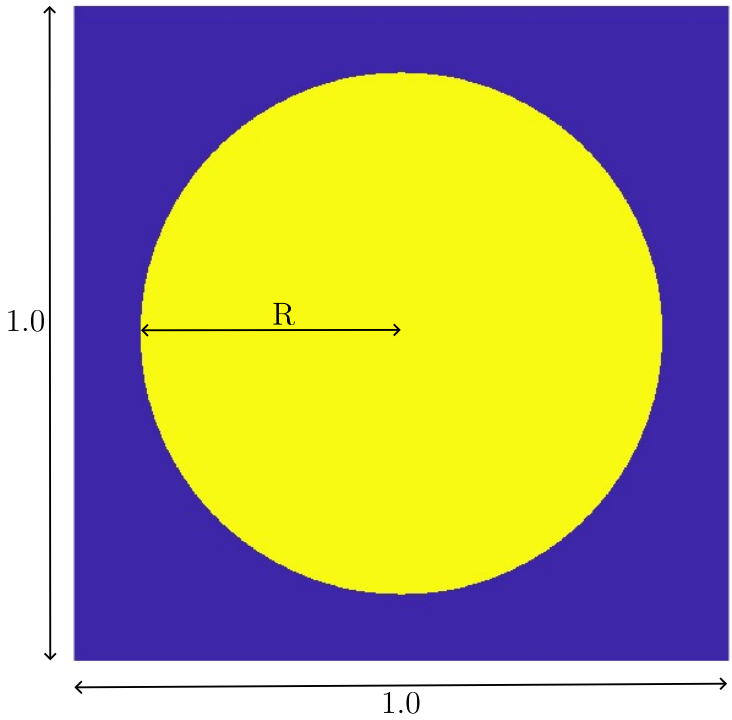


Figure 5.6: Two dimensional representation of the microscopic geometry. In this case we have a host matrix in blue with a circular inclusion in yellow. The radius of this circular inclusion is $R = \sqrt{\frac{05}{\pi}}$, such that $\omega_B = 0.5$. Here the blue host matrix is much stiffer than the yellow inclusion.

have the exact same material properties and the same proportion of each material in their composite domain, however the microscopic geometry can massively sway the effective macroscopic elasticity. Now lets analyse how this small difference in the microscale will effect the damage evolution and tearing process.

5.4.3 Comparison of simulation results

In Matlab, using our code in Appendix B, we run both our simulations with the microscopic setups described in §5.4.1 and §5.4.2. The simulation ran from $t = 0$ to $t = 1$, unless the code failed to keep converging due to us altering the mesh by deleting nodes that are completely damaged, this loss of nodes in the mesh lowers the resolution and can lead to the numerical scheme not being able to converge. If this occurs we simply terminate our simulation early, this is an interesting numerical problem that can be revisited in the future: How to have a changing mesh that does not affect the numerical schemes ability to converge? At each time step the external loading applied is described by the effective body forces (5.64). In this subsection we will discuss a few comparative plots to illustrate how small changes in the microscale can cause large differences at the macroscale. First

though let us define our reference domain as a unit square such that $\Omega = [0,1] \times [0,1]$. Using our code we will set the maximum element size to be $h = 0.01$ as a result we will produce a mesh with $11,717$ nodes and a connectivity matrix with $23,032$ elements. For reference we plotted the mesh in Figure 5.7.

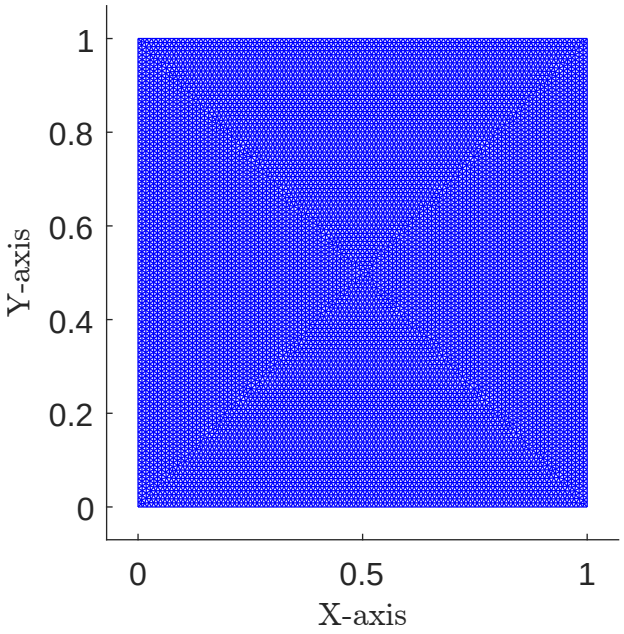
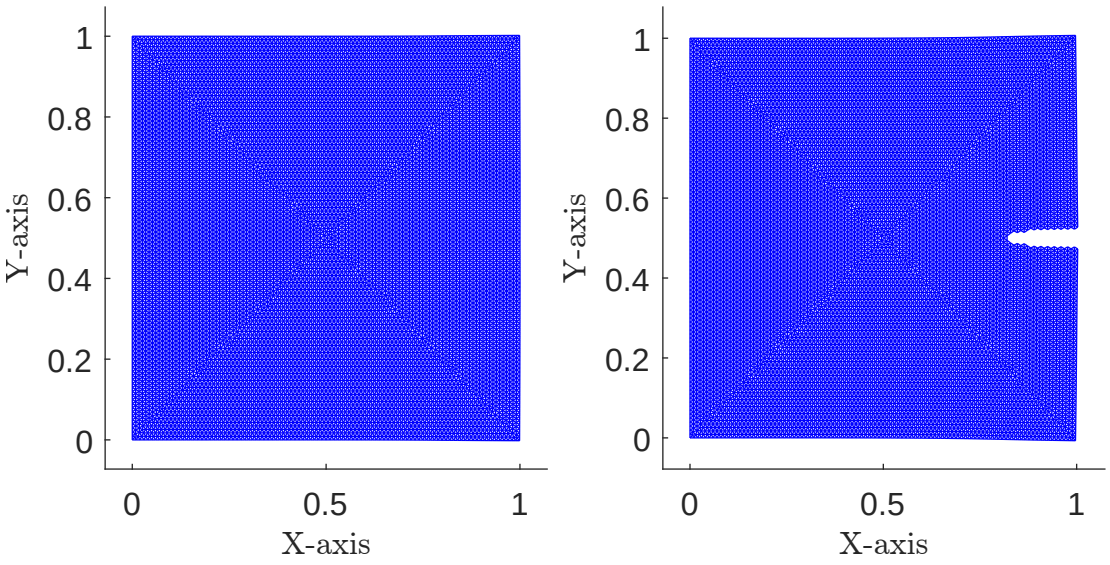


Figure 5.7: The mesh approximation of our reference domain $\Omega = [0,1] \times [0,1]$. Using our code in Appendix 5.3.1 we produced a mesh with $11,717$ nodes and a connectivity matrix with $23,032$ elements.

We simulated both the scenarios described by §5.4.1 and §5.4.2, until completion. In both cases the simulation ended abruptly when each numerical scheme was no longer capable of converging. This was because as mentioned altering the mesh and deleting nodes does affect the numerical schemes ability to converge, and when it fails to converge we simply terminate our simulation. However, we still have plenty of comparable data from these simulations. Let us begin by comparing how frankly different the outcome of both simulations are at the same time step. Consider Figure 5.8 both simulations are at the same time $t \approx 0.3653$. We can see a huge difference in both simulations the vertical strip scenario is only slightly deformed whereas the inclusion setup has a large tear through it. Because the inclusion setup has suffered a tear through its domain we can see that this has allowed for a greater deformation. In fact we can see that the inclusion setup is beginning to form the desired shape of trousers. Figure 5.8 is a great indicator of how different the macroscopic displacement $\bar{\mathbf{u}}$ is due to the differing microscopic geometries.

To compare the damage phase fields of our two simulations we use contour plots, as seen in Figure 5.9. Again these contour plots are at times $t \approx 0.3653$ and we can see that both our microscopic geometries have very different damage phase fields. The vertical strip setup is less concentrated and not that steep but we can see that a peak is beginning



(a) Vertical strip setup. (b) Inclusion setup.

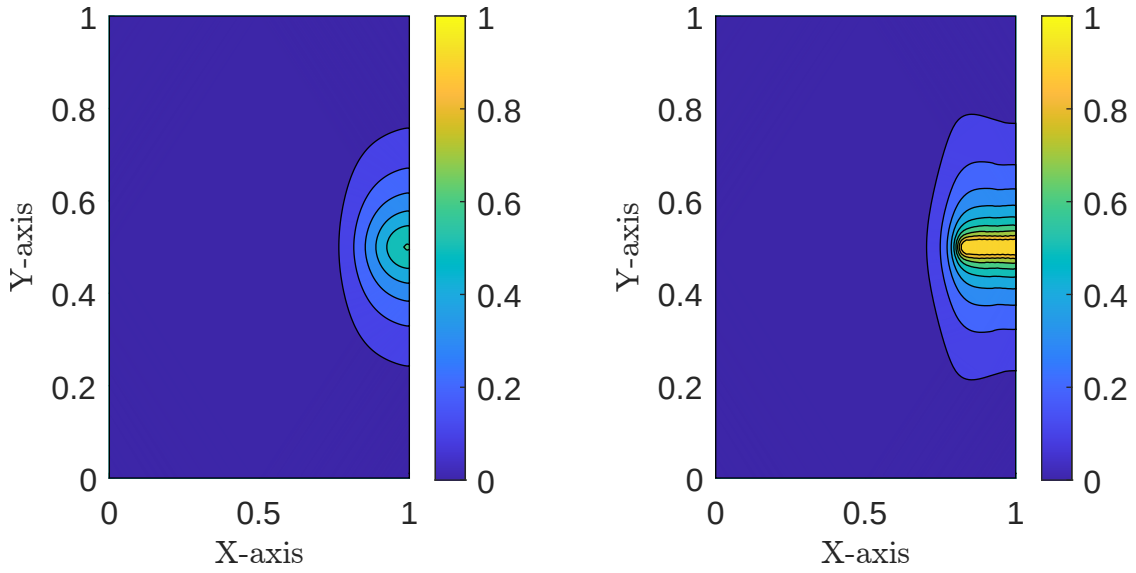
Figure 5.8: Two plots of the deformed meshes with different microscopic geometries at time $t \approx 0.3653$. In (a) we have our vertical strip setup as described in §5.4.1 and in (b) we have our inclusion setup as described in §5.4.2. The simulation we performed was done with the physical parameters and external loading described in §5.4.

to form on the right edge of Ω. Whereas, in the inclusion setup we can see that the damage phase field is further evolved. We can see a sharp concentration protruding from the right hand side that is comparable to the tear visible in Figure 5.8b. The main conclusion though is that the damage phase fields in Figure 5.9a and 5.9b are clearly not the same. Again this is due to differing microscopic geometries.

In Figure 5.9 we can see a clear discrepancy in the maximum values of the damage phase field. Therefore, to get a clear comparison of how different each damage evolution process was due to the microscopic geometries, let us consider a plot of the maximum damage against time, as can be seen in Figure 5.10. In that figure it is abundantly clear that the inclusion setup of the microstructure fails far more rapidly in this physical scenario. The inclusion setup reaches the maximum damage of $\bar{\alpha} = 1$ at time $t \approx 0.35635$ compared to the vertical strip setup which reaches the maximum damage later at time $t \approx 0.37147$.

A final qualitative result we consider is a stress-strain curve comparison of both microscopic geometries. To do this let us consider the material point $\tilde{\mathbf{X}} = [1, 0.5] \in \Omega$, in particular let us look at the vertical components of stress and strain in the vertical direction. Firstly, let us define the components of our displacement as $\bar{\mathbf{u}} = (\bar{u}, \bar{v})$, where u is the horizontal component of displacement and v is the vertical component of displacement. Then in two dimensions we may define the vertical components of stress and strain in the vertical direction as

$$\sigma_{YY} = \left[g(\bar{\alpha})\bar{\mathbb{C}} : \xi(\bar{\mathbf{u}})\right]_{22} \quad \text{and} \quad \xi_{YY}(\bar{\mathbf{u}}) = \frac{\partial \bar{v}}{\partial Y}. \tag{5.71}$$



(a) Vertical strip setup.

(b) Inclusion setup.

Figure 5.9: Two contour plots of the damage phase fields with different microscopic geometries at time $t \approx 0.3653$. In (a) we have our vertical strip setup as described in §5.4.1 and in (b) we have our inclusion setup as described in §5.4.2. The simulation we performed was done with the physical parameters and external loading described in §5.4.

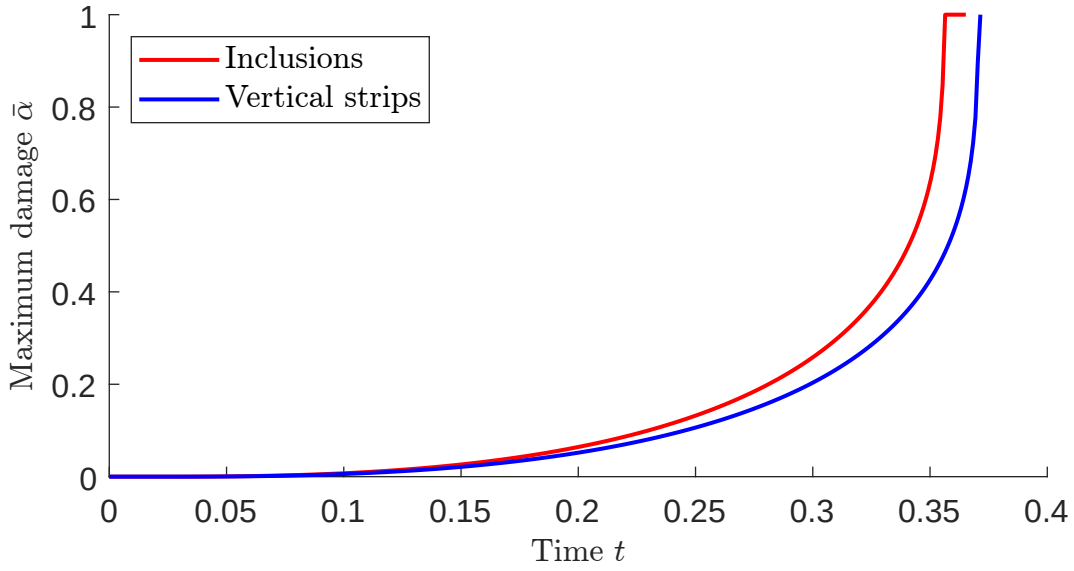


Figure 5.10: This is a plot comparing the maximum damage of both simulations at each time step against time. The two simulations we are comparing are the inclusion setup described in §5.4.2 and the vertical setup described in §5.4.1. Our trouser test and external loading process is described thoroughly in §5.4. Here we can see that the inclusion setup is far less resistant to damage evolution compared to the vertical strip setup.

We may now plot these stress and strain components as defined by (5.71) against each other at the point $\tilde{\mathbf{X}}$ as seen in Figure 5.11. Here we can see an interesting example of plasticity. In Figure 5.11 we have marked the point where damage evolution begins and ends. The point were damage begins marks when in the deformation of the trouser test

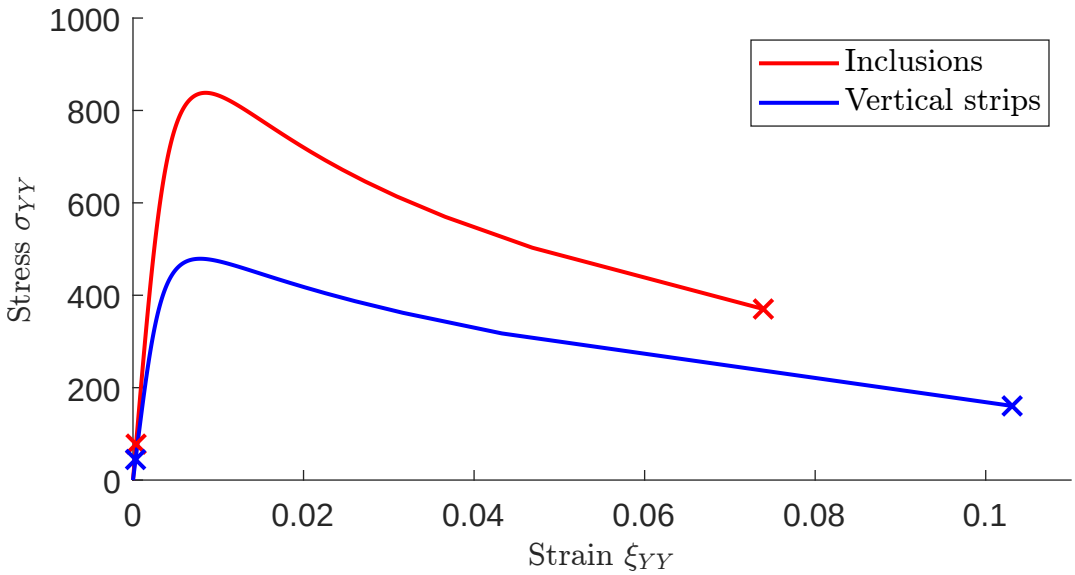


Figure 5.11: Here we are plotting the stress σ_{YY} against the strain ξ_{YY} as defined by (5.71) for our trouser test simulations. In particular this is the stress-strain curve of the vertical components in the vertical direction at the point $\tilde{\mathbf{X}} = [1, 0.5] \in \Omega$. The red line is the microscopic inclusion geometry and the blue line is the microscopic vertical strips geometry. The crosses on the left indicate the strains when damage evolution begins to occur and the crosses on the right indicate the strains when a tear occurs, in other words when $\alpha(\tilde{X}) = 1$.

we begin to soften the material stiffness. Whereas, the point the where damage ends indicates when a tear occurs in the material. Comparatively we can see that the microscopic inclusions geometry experiences far more stress, as defined by (5.71), than the microscopic vertical strips geometry. This larger stress in turns means a proportionally larger linear elastic potential energy, which of course drives the damage evolution. As a result we can understand why the damage evolution occurs more rapidly for the inclusions geometry, as seen in Figure 5.10. This result was of course expected because we are in the realm of linear elasticity to understand which material would have the larger vertical component of stress in the vertical direction one would only need to inspect the stiffness tensors. In particular we would need to inspect the component $\bar{\mathbb{C}}_{2222}$ in §5.4.1 and §5.4.2, and we of course would see, as expected, that this component is larger for the microscopic vertical strip geometry compared to the microscopic inclusion geometry. Another interesting feature we can remark upon in Figure 5.11 is that we can clearly see that a tear in Ω occurs for a smaller strain in our trouser test simulation for the microscopic inclusion geometry compared to the strain where a tear occurs for the microscopic vertical strips geometry. However, we can also note that the microscopic vertical strip geometry begins damage evolution for a smaller strain in our trouser test simulation.

From the simulations in this section it is clear how we can use our multiscale model to study the effects the microscale geometry and material properties can have on the

damage evolution process at the macroscale. This comparison was done with elements of maximum size $h = 0.01$, by performing a more detailed simulation we would obviously get a more accurate result and could make more definitive conclusions about the macroscopic material properties. However, even with such low resolution at the macroscale we where able to interpret the effects the microscale geometry can have on the tearing process. By being able to simulate these processes we can save researchers a lot of time and money with the investigations of material properties.

5.5 Summary of results

The focus of this chapter was to convey how we can use our multiscale model to study tearing processes in higher dimensions. To do this we began by explaining how we can use the finite element method to solve our macroscopic model. This included introducing methods from the literature to calculate the effective macroscopic coefficients [3] and a new numerical algorithm to handle altering our mesh so that we may simulate tears in Ω. The main body of this numerical work was introduced in §5.3 with the corresponding code available in Appendix B. With our code we preformed a set of simulations to highlight what is possible with our multiscale model. Through a trouser test example we illustrated how we can study the effects of slight changes in the microscale. What we found is that a small change at the mircoscale can have a large effect at the macroscale. Hence, our model makes it possible to study the full role that the microscale plays in macroscopic tearing processes.

5.5.1 Concluding remarks

This chapter has introduced a large avenue of future directions for research. With our multiscale model it is now possible to perform a large set of in silico experiments, studying the effects that microscopic changes will have on aortic dissections. However, to perform these experiments we need more data to model the aortic microstructure for each of the three layers making up the aortic wall. This would include calculating the residual stress, tearing energy and threshold energy for each constitutive part of the aortic microstructure. Getting this data will require a mixture of scouring the literature and collaborating with experimentalists.

To judge how well the model can approximate a real aortic dissection would require a statistical comparison to the medical data. Though we do believe this is an exciting research opportunity. The first step of which would be to extend our code to three dimensions and improve the efficient of said code, this could be done in collaboration with software engineers. One we have a fully implemented code that can perform three dimensional simulations then we can perform as many in silico experiments as we want.

It should be noted that this multiscale model of the damage phase field method is not solely limited to modelling aortic dissections. We could model many different scenarios of soft tissue tearing. Consider the tearing of tendons and ligaments, modelling them effectively could provide new insights for prevention and recovery from injury. We are also not solely limited to the human body; the field of material science could gain many insights from having a more effective method of studying microscopic changes in materials.

Engineers could benefit from the ability to study the effects microscopic materials have on overall material failure. It could be possible to study composite, alloy or fibrous materials with our model. Through many simulations we have learned our model can be used to study more than sharp tears. We can use our multiscale model to investigate cyclic damaging, say from general wear tear like walking on a pavement. Plasticity in many materials could also be investigated using our model via the degradation function. All these future research possibilities simply require some upgrades to our current code, in Appendix B.

Chapter 6

Discussion

This thesis has provided several interesting advances in the modelling of aortic dissections, the damage phase field method and the usage of asymptotic homogenisation. In this chapter the main findings of the thesis will be reviewed, the key assumptions of the thesis discussed and possible future work based on these conclusions will be suggested.

6.1 Thesis summary

In Chapter 1, we discussed our interest in aortic dissections and the development of a multiscale model to account for how changes in the aortic microstructure affect dissections. With this objective in mind, this thesis focuses on deriving a novel multiscale model of the damage phase field. We aimed to maintain the model's general applicability while ensuring its relevance to aortic dissections.

Starting in Chapter 2, we present a re-derivation of the damage phase-field method to provide a complete introduction to the method. We began with the fundamental assumptions of the damage phase-field method which we used to construct an energy balance described by the displacement and damage. From there, we employed the Karush-Khun-Tucker conditions which allowed us to utilise calculus of variations to derive a model incorporating damage. As a result our damage model has a criterion for damage evolution and is constrained by the irreversibility of the damage phase field. Our model was not completely original and is heavily based on the works of Miehe et al. [90] and Marigo et al. [83]. However, we did introduce some new ideas to the theory: a threshold function to set an energetic bound to material damage evolution and we expanded the model to a composite domain. This was done with foresight, we created a damage model on as general as possible of an arbitrary composite domain. Hence allowing our approach to maximise the scope of what we can describe. Using our composite model we approximated tearing in the human aorta at the microscale with a system of partial differential equations.

In Chapter 3, we derived an original *multiscale* damage phase-field model, using asymp-

totic homogenisation to upscale our microscopic damage phase-field model. This began with us making the appropriate constitutive choices for the form of the damage degradation function as a polynomial function and that the strain energy density would be a linearised choice, in our case we used a combination of Hooke's law and a residual stress term. Next we introduced the appropriate scaling of the microscale across the constitutive parts of composite domain and the macroscale across the full domain. Assuming that the scale ratio ϵ of the micro to macroscale is reasonably small then we can decouple the spatial scales. By introducing some novel analysis we could appropriately upscale the full model and all its constraints to the macroscale. In order to do this we had to use the local periodicity at every scale of our homogenisation process, we found that the assumption of local periodicity guarantees that the irreversibility and damage criterion can be upscaled to the macroscale. The resulting system of partial differential equations consists of equation of motion and a damage evolution equation, that encodes information about the microstructure into the coefficients of the model. These coefficients are computed by solving appropriate cell problems and calculating a series of local integral averages, which reflect the complexity of the given microstructure. On top of this we prove rigorous properties of the coefficients, simplifying the model. We prove that the macroscopic elasticity tensor has major and minor symmetries; and is positive definite. Similar results about macroscopic coefficients have been reported in the literature [104]. However, this is the first multiscale model of the damage phase-field which satisfies all the modelling constraints and formally upscaled all the models' properties.

As our multiscale model was based on assumptions of local periodicity and required new analysis, we compared the numerical simulations of the full microscopic model and the corresponding macroscopic model to show that both models converge as the scale ratio ϵ decreases towards zero. This numerical analysis was performed in Chapter 4, where we set about explaining the required numerical methods to solve our damage model. As our model was constrained by a damage criterion, and irreversibility constraints we knew our numerical methods would be unorthodox. Therefore, we reduced the problem to one dimension and assumed a simplistic setup of a one-dimensional bar being stretched at one end. In one dimension we were able to analytically calculate the macroscopic effective coefficients in terms of the microscopic coefficients. One interesting observation we did make is that the effective coefficients of the elasticity and diffusion where both harmonic means in terms of their microscopic components. Interestingly, we also remarked upon how the effective residual stress term took the form of a weighted mean in terms of the microscopic residual stress. From here we produced a numerical algorithm for solving the problem numerically. Our numerical method was then implemented in Matlab and can be seen in Appendix A. Using this code we performed a parametric analysis and consistently found that the full microscopic problem and its macroscopic approximation do converge as

ϵ goes to zero. This gives evident backing to all the theoretical work found in this thesis. Another important observation made in Chapter 4 was the importance of correctly scaling all the modelling parameters. In one dimension we found that if the diffusion was set to be of an order of magnitude lower than ϵ then the microscopic and macroscopic models would fail to correctly converge. This makes sense as in asymptotic homogenisation we require that ϵ goes to zero and is essentially the smallest parameter value. However, the diffusion is also set artificially small to localise the damage phase field so one must be careful to ensure that diffusion is small but still larger than ϵ. It should also be noted in Chapter 4 we found that regularly that the macroscopic and microscopic solutions of our simulations where converging within two to three significant figures of one another at a value of $\epsilon = 0.1$, which is not very small. This is a significant observation because it means our damage model could be used to model even more structures in nature with ϵ much larger than one would think plausible.

Lastly, in Chapter 5 we wanted to provide a numerical framework for our multiscale model in higher dimensions. Aortic dissections, and other forms of soft tissue tearing, are not one-dimensional processes but rather they take place in complex three-dimensional geometries. To solve our multiscale model in higher dimensions we used the finite element method. First, we introduced the finite element method to the reader. Next, we adapted our numerical algorithm from Chapter 4 to higher dimensions, accounting for a mesh that evolves to approximate tears in our domain. From here we used the finite element method to create numerical schemes for both our equation of motion and damage evolution equation. This involved taking careful account of irreversibility constraints and the damage criterion. Our numerical framework was then implemented in Matlab and this code can be found in Appendix B. Using a trouser test, we illustrated how we can use our multiscale model to study how slight differences in the microscopic geometry can majorly affect the outcome of the macroscopic tearing processes.

We believe that this work has expanded the current literature. There are of course many extensions to the possible future research developed by this thesis and have specifically commented on this throughout the thesis. Overall, there is a variety of novel work throughout this thesis that could still be developed further in order to increase our current understanding of multiscale modelling, damage modelling and aortic dissections.

6.1.1 Key assumptions of the thesis

The conclusions of this thesis are based on a number of assumptions. These assumptions are necessary for the proper use of asymptotic homogenisation and we made them safely understanding that most structures in nature do somewhat obey them. One of the key assumptions made is the local periodicity of the microstructure, without which you cannot perform asymptotic homogenisation in dimensions greater than one. You also cannot

upscale the damage evolution equation and all its constraints without the assumption of local periodicity as widely observed in Chapter 3. Performing asymptotic homogenisation without assuming local periodicity is still not possible. Another assumption we made for the benefit of asymptotic homogenisation was the linearisation of the strain energy density. We linearised around some residual stress term σ which meant when we analysed our full damage problem at the scales of ϵ^0 and ϵ^1 we only found linear relationships. Obviously, in nature many materials do not obey linear elasticity. That is why we linearised around a residual stress term, to account for any pre-existing stress in the material. However, it should be noted that progress is now being made in the literature with performing asymptotic homogenisation on non-linear systems [24, 62, 92, 112, 141]. It would be interesting at a later date to revisit the work in this thesis and model the aortic microstructure with non-linear properties. For the damage phase-field method to localise we assumed that the length scale of diffusion is extremely small, similar to the scale ratio ϵ. As already mentioned, in Chapter 4, we know that for the model to correctly homogenise we must be sure to set diffusion on a scale equable or larger than ϵ. Otherwise, the full microscopic and macroscopic models will not converge.

6.1.2 Directions for future research

The work in this thesis has opened up lots of exciting potential future research opportunities. We can divide these into two different areas: theoretical and experimental work. Lots of new theoretical questions can now be explored using the multiscale modelling techniques illustrated throughout this thesis. This of course will require a collection of microscopic data to test any new potential models, such as our multiscale model of aortic dissections.

The first theoretical question that will need to be explored is the design of accurate microstructures to represent many different parts of the aortic wall for each of the wall's three layers. This will of course require exploring much of the existing literature and consulting with anatomical researchers. With this information we can begin to calculate a physiologically accurate range for the effective coefficients. This will require cataloguing all the correct material properties for the microstructure. Material properties such as the Young's Modulus or the range of Poisson's ratio for collagen, smooth muscle cells and other elastic fibres are well documented in the literature [13, 47, 145]. However, our multiscale damage phase field model requires information about the energy threshold for damage initiation and the tearing energy. These are not well reported in the literature, mainly because the damage phase field method is a rather new technique. Therefore, experimental work will have to be conducted at the cellular level to obtain accurate physiological ranges for these parameter values. Diffusion on the other hand needs to be set to some small value to force the localisation of damage evolution. It should be noted however that using existing data we could at least constrain the model parameters. Then using a number of

statistical methods it would be possible to estimate the realistic range of these parameters by testing our model against existing medical data. This approach includes statistical methods such as maximum likelihood estimations [91] and Bayesian estimators [25].

An interesting question that will need to be asked when performing experimentation will be choosing which mechanical properties to use. Consider collagen, if you look at collagen molecules, fibrils and fibres at each scale they have very different mechanical properties [13]. The collagen molecules of course are the stiffest and require the most energy to break, then the same properties for collagen fibrils are of a lower order of magnitude and the fibres are again of an even lower order of magnitude. Therefore, when all the data is collected careful consideration must be made when using it to calculate the effective coefficients.

From here it will be possible to perform both interesting and informative simulations. To do this well collaboration with software engineers would allow us to construct a very efficient piece of software. Importantly we would need to extend the software in Appendix B to do three dimensional simulations and to include contributions from the residual stress. Using this new software, we would be able to perform a wide range of simulated aortic dissections. Comparing the results to real medical data we could make a series of conclusions about the utility of our model. From there we could make any necessary adjustments to our model. After which in collaboration with statisticians we could perform a large statistical emulation. This would require performing many simulations for a huge variety of aortic microstructures, macroscopic geometries and microscopic material properties. In effect this would be a large sensitivity analysis of all possible parameters within relation to an aortic dissection. With the help of statisticians, we could take the results of these many simulations and perform a large emulation. In a sense interpolating across the range of all possible solutions for all physiologically possible scenarios. This would create a powerful tool in software to quickly simulate and determine the outcome of many potential dissection scenarios.

With this tool we could work with clinicians and test how accurately our model predicts and simulates an aortic dissection. We could gain many new insights with such a tool by being able to predict the potential causes of aortic dissections and the outcome of different treatments for the disease. The practical applications alone would be tremendous if clinicians could accurately prevent and treat the disease whilst being better informed by our model. However, any inaccuracies in our models' predictions would of course be used to better inform our theoretical setup. This is of course how we do research, with theory being informed by experimental results and experiments being devised by theoretical ideas.

Moreover, other than potential future research related to our model for aortic dissections, there are potentially many other clinical applications. Soft tissue tearing in the body could be modelled in general with our multiscale damage phase-field model. This

could include physical conditions such as the tearing of tendons and ligaments. The applications could include better prevention of the tears and a more informed recover process for clinicians and patients alike. Our model is also not restricted to biological problems either, we could model many other structures in nature. Material science is of course very important for engineering purposes. A more complete understanding of how alloys, composites or laminated materials can break or degrade could be achieved with our multiscale model. The scope of engineering applications would be quite large. Our model has also shown the ability in simulations to model less sudden forms of damage like dissections. We can model fatigue in structures and cyclic damaging. Practically this could be used to model how structures slowly fail over time, like roads or bridges. We can also use our model to study plasticity in more elastic materials, to gain a better understanding of how composite structures fail.

Overall, this thesis has been a necessary first step in introducing an approach to multiscale modelling of soft tissue tearing. More generally we would say that we have introduced a multiscale technique for studying material failure in many different scenarios. We do hope that eventually the work in this thesis does lead to a practical tool for clinicians which helps to prevent and treat aortic dissections.

Appendix A

Finite difference method code

This appendix contains the code used to perform all the simulations discussed in Chapter 4. For a breakdown of how the code is structured and the key ideas behind our numerical algorithm and schemes please see §4.2. Each of the following sections will give a brief description of what each script or function of code does. All of this code was designed for and ran in Matlab [72]. It should be noted however, many software languages are extremely similar and converting this code would not be an ordeal.

A.1 Input code

The first piece of code is the input script as seen below in the Listing A.1, and the purpose of this script is for us to setup an appropriate simulation. We have freedom to choice the size of our one dimensional reference domain and the time domain we want to simulate over. The next information we need to provide is the coefficients describing the material properties we would wish to simulate. Of course we can set an error tolerance for our iterative scheme and setup our displacement boundary conditions. The damage degradation function can also be setup here but one should be careful to choose an appropriate function for damage degradation as thoroughly discussed in Chapters 2 and 3.

When the Listing A.1 is ran it will call the function "SolveProblem" which of course performs the simulations. After the results of the simulation are calculated the last job this script will perform is to create a set of plots. First of which are two plots of the final displacement and damage fields at the final time step. The other two plots are surfaces which illustrate the damage and displacement evolution throughout the one dimensional simulation.

Listing A.1: A Matlab script which allows us to input all the necessary data for our one dimensional simulation. The script then calls our main function to perform the simulation and then produces some plots of the simulation results.

```
% In this code we define a spatial domain, time domain and boundary
% condition.
%
% We also define our elasticity, threshold, diffusion and energy at every
% node.

clc
close all
clear Usol Asol T
beep off
format long

% We begin by defining our domain of interest and the number of nodes
% we wish to solve for. Note that larger n means a better resolution
% but more computational time.

% Our epsilon.
epsilon = 1;

% Define our bounds.
xlim = [0 1];
tlim = [0 1];

% Number of time steps we wish to save.
Nt = 1000;

% Define domains and nodes for damage and displacement.
n_x = 1000;
X = linspace(xlim(1),xlim(2),n_x);

% Error tolerance.
tol = 1e-6;

% Our Displacement Boundary Condition.
```

```
k = [0 1]; % Some stretch factor

% Our initial Damage and displacement.
a0 = zeros(1,n_x);
u0 = zeros(1,n_x);

% Define our degredation function.
g = @(y) (1 - y).^2;

% Our Problem Coefficients
E = ones(1,n_x); % Elasticity
P = 1 + 0.9*cos(2*pi*X); % Threshold
G = ones(1,n_x); % Energy
D = 1e-2*ones(1,n_x); % Diffusion

% Code we run.
tic
[Usol,Asol,T] = SolveProblem(X,tlim,Nt,k,g,u0,a0,E,P,G,D,tol);
toc

% Reinterperate data.
x = linspace(X(1),X(end),100);
t = linspace(T(1),T(end),100)';

Asurf = interp2(X,T,Asol,x,t);
Usurf = interp2(X,T,Usol,x,t);

% Plots of the final solution.
figure(1)
hold on
plot(X,Asol(end,:))
title('Final Damage')
xlabel('Position X')
ylabel('Damage')
ylim([0 1])
hold off

figure(2)
```

```
hold on
plot(X,Usol(end,:))
title('Final Displacement')
xlabel('Position X')
ylabel('Displacement')
hold off

% Surfaces of the solutions U(X,t) and A(X,t).
figure(3)
hold on
surf(x,t,Usurf)
title('Displacement Field')
xlabel('X Position'), ylabel('t Time'), zlabel('Displacement')
hold off

figure(4)
hold on
surf(x,t,Asurf)
title('Damage Field')
xlabel('X Position'), ylabel('t Time'), zlabel('Damage')
zlim([0 1])
hold off
```

A.2 Numerical algorithm

In §4.2.1 we introduced Algorithm 1 which was designed to iterate between numerical schemes that independently calculate solutions for the displacement field, damage criterion and damage field. In Listing A.2 we present our Matlab function which performs the role of Algorithm 1. For efficiency in Listing A.2 instead of saving the solutions of damage and displacement at every time step in memory we instead export the solutions to appropriate text files. At the completion of all time steps or when the damage reaches the maximum of 1 we reload all the data from our text files into memory as solution matrices. In these solutions every row is a time step and the columns represent nodes throughout the one dimensional domain.

Listing A.3 below is a short Matlab function which calculates the first and second partial derivative of the degradation function with respect to the damage phase field. We require the first partial derivative because the damage criterion (4.17) of course depends on the first partial derivative of the degradation function. The second partial derivative

is needed for our Newton method style solution of the damage evolution equation (4.24). Therefore, calculating both of these derivatives early in Listing A.2 allows us to save them in memory and call on them whenever we need them.

Listing A.2: This listing is a Matlab function that performs the role of Algorithm 1. The role of this Matlab function is to go through each time step of our simulation correctly iterating between solving the displacement and damag field. Solutions of each time step are saved in the appropriate text files which is efficient when memory is sparse because we are dealing with many time steps, nodes or both.

```
% This code takes all our input data and proceeds to solve the
% damage phase field problem.

function [Usol,Asol,T] = SolveProblem(X,tlim,Nt,k,g,u0,a0,E,P,G
   ,D,tol)

% All the solved time steps are to be solved and saved as rows
   in seperate
% text files as follows. Here we
% are saving the first time steps as the initial values.
damage_file = fopen('Damage.txt','w');
displacement_file = fopen('Displacement.txt','w');
fprintf(damage_file,'%d,',a0);
fprintf(damage_file,'\n');
fprintf(displacement_file,'%d,',u0);
fprintf(displacement_file,'\n');

% Firstly we define the matrices dU, U and A, strain,
   displacement and
% damage respectively. The first row represents the previous
   time step, the
% second row reprsent the current time step.
A = [a0 ; a0];
U = [u0 ; u0];
dU = zeros(size(U));

% Next we calculate the first and second partial derivative of
   the
% degradation function for convience.
[g1,g2] = DegradationPartials(g);
```

```
26
27 % Now we move onto the first time step to be solved, so
      importantly we
28 % calculate the step size for each time steps as follows.
29
30 step = (tlim(2) - tlim(1))/(Nt - 1);
31
32 % Firstly we caculate the final time step and the first time
      step t we are
33 % solving.
34 tfinal = tlim(2);
35 t = tlim(1) + step;
36 % Track previous time step.
37 t0 = tlim(1);
38
39 % Now we can initalise the problem solving process as a while
      function
40 % where we condition the loop to run till we pass the final
      time step
41 % tfinal or if a tear occurs in our model.
42
43 % Loop we are solving.
44 while t <= tfinal
45     % Display the step to track the progress of the code.
46     disp(['At time ' num2str(t,12)])
47
48     % Update our BCs at each time step as the dirchlect
          conditions.
49     BC = [k(1)*t k(2)*t];
50
51     % The first thing we do is calculate the current
          displacement i.e.
52     % U(:,2) to do this we use the elastic phase function.
53     [U(2,:), dU(2,:)] = ElasticPhase(BC,g,A(1,:),E,X);
54
55     % Next we calculate if the damage criterion is satified
          anywhere and
56     % hence we have damage growth. To do this we calculate the
```

```
      criterion as
% follows.
Crit = CriterionCheck(g1,A(2,:),dU(2,:),E,P,G,D,X);

% Next we check to see if the criterion is or is not satisfied at every
% point. If it is satisfied everywhere we move on to the next time
% step. However if it is not then we work to solve all the unsatisfied
% points coupled with the satisfied points.
if sum(Crit) > 0
    % Now we set up an iterative process to solve for A(2,:).
    Y_curr = A(2,:);
    Y_old = -ones(size(A));
    while norm(Y_curr - Y_old, Inf) > tol
        % disp(['The Norm is ' num2str(norm(Y_curr - Y_old, Inf))])
        Y_old = Y_curr;

        % Solve the damage problem.
        A = DamagePhase(A,g1,g2,dU,E,P,G,D,Crit,X);

        % Firstly we try to satisfy the time step with an elastic sol.
        [U(2,:), dU(2,:)] = ElasticPhase(BC,g,A(2,:),E,X);

        % Then we check if we have satisfied the criterion.
        Crit = CriterionCheck(g1,A(2,:),dU(2,:),E,P,G,D,X);

        % Update iteration
        Y_curr = A(2,:);

        % Break condition if criterion is satisfied
        if sum(Crit) == 0
            break
        end
```

```
88          end
89      end
90
91      if max(A(2,:)) <= 0.99
92      % Save data for this time step.
93      fprintf(damage_file,'%d,',A(2,:));
94      fprintf(damage_file,'\n');
95      fprintf(displacement_file,'%d,',U(2,:));
96      fprintf(displacement_file,'\n');
97      % Update our current damage and displacement fields.
98      A = [A(2,:) ; A(2,:)];
99      U = [U(2,:) ; U(2,:)];
100     dU = [dU(2,:) ; dU(2,:)];
101     % Update time.
102     t0 = t;
103     t = t + step;
104
105     % Conditions for time step.
106     elseif max(A(2,:)) > 0.99
107         if max(A(2,:)) <= 1
108             % Save data for this time step.
109             fprintf(damage_file,'%d,',A(2,:));
110             fprintf(damage_file,'\n');
111             fprintf(displacement_file,'%d,',U(2,:));
112             fprintf(displacement_file,'\n');
113             break
114         elseif max(A(2,:)) > 1
115             A = [A(1,:) ; A(1,:)];
116             U = [U(1,:) ; U(1,:)];
117             dU = [dU(1,:) ; dU(1,:)];
118             t = t - (t-t0)/2;
119         end
120     end
121 end
122
123 % Note that the column vectors U and D are saved here as rows,
      transpose
124 % the data.
```

```
fclose(damage_file);
fclose(displacement_file);

% Convert files to solutions U(x,t) and D(x,t).
Usol = readmatrix('Displacement.txt');
Asol = readmatrix('Damage.txt');
T = Usol(:,end)/k(2);

end
```

Listing A.3: This Matlab function calculates the first and second partial derivative of the degradation function with respect to the damage phase field.

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Calculate Partial Derivative of Degradation Function.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [g1,g2] = DegradationPartials(g)
syms y

g1 = eval(['@(y)' char(diff(g(y)))]);

g2 = eval(['@(y)' char(diff(g1(y)))]);

clear y
end
```

A.3 Elastic phase

In Listing A.4 we have the code that calculates the solution to the displacement field and the gradient of the displacement field. This is based of the analytical solutions calculated in §4.2.2. Of course we are working with a discretised domain so the solutions of the displacement field (4.18) and its gradient (4.19) are calculated on the same discretised domain. To do this we have employed the Matlab function "cumtrapz" which calculates integrals over a discretised domain using the cumulative trapezoid method. The integrals we are calculating are the ones present in the analytical solution of the displacement field (4.18) and the gradient of the displacement field (4.19).

Listing A.4: This Matlab function calculates the analytical solutions of the displacement field (4.18) and the gradient of the displacement field (4.19). Here we make use of the Matlab function "cumtrapz" which calculates integrals using the cumulative trapezoid method.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This function solves the Elastic phase problems implicitly.
%
% d/dx( (1-D)^2 E dU/dX  ) = 0 ~ basic elastic problem.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [U , dU] = ElasticPhase(BC,g,A,E,X)

% g is our degredation function, BC our dichlect BCs, A is our
   damage phase
% field, E our elasticity and X our domain.

% Define the left and right boundary conditions.
UL = BC(1);
UR = BC(2);

% Define the Integral term.
Int = cumtrapz(X,1./(E.*g(A))) ;

% Solution to U.
U = UL + (UR - UL) * (Int./Int(end)) ;

% Derivative of U;
dU = (UR - UL) * (1./(E.*g(A)))/Int(end) ;

end
```

A.4 Damage criterion

In §4.2.3 we introduced the notion of the criterion field. The criterion field tracks all the points in the reference domain for the current iterative step which do not satisfy the damage criterion. As defined by (4.22) we set $\mathrm{Crit}(X) = 1$ at every point that does not satisfy the damage criterion and $\mathrm{Crit}(X) = 0$ at every point that does satisfy the damage

criterion. We coded this step up for our numerical method as seen in Listing A.5. Here we use our current solutions for the displacement and damage fields to calculate the damage criterion as defined by (4.17). This is done in our Matlab function "DamageEquation" which is described by Listing A.6.

To calculate the damage criterion (4.17) we use the code in Listing A.6. This function is using the solutions of the displacement field and damage field to calculate the damage criterion. We calculate the derivatives present in the damage criterion with second order centre difference methods across the discretised reference domain. For the boundary nodes we make use of first order forward and backward difference methods to employ fictitious nodes.

Listing A.5: This Matlab function uses the current solutions of the displacement and damage field to calculate the damage criterion defined by (4.17). With these solutions it assigns the correct value for Crit(X) at every point in our discretised domain, as defined by (4.22).

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Here we simply check at which points we are passing the
   damage criterion.
% We also check that the time resolution is good, with a
   tolerance check.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Crit = CriterionCheck(g1,A,dU,E,P,G,D,X)

F = DamageEquation(g1,A,dU,E,P,G,D,X);

Crit =  0 * (F<=0) + 1 * (F>0);

end
```

Listing A.6: A Matlab function for calculating the damage criterion (4.17). This function makes use of finite difference approximations to calculate the damage criterion at every node in the mesh.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This function calculates the damage criterion/equation and we
    store it
% here instead of retyping it over and over.
```

```
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function F = DamageEquation(g1,A,dU,E,P,G,D,X)

% Our spatial discretisation step size.
h = (X(end) - X(1))/(length(X) - 1);

% We start by solving all the internal values of the damage criterion from
% i = 2 to i = n-1.

% We define the Second Order Centre second Derivative for this step.
Cntr = @(y,h) [0 y(3:end) - 2*y(2:end-1) + y(1:end-2) 0]/h^2;

% Calculate the damage criterion.
Q = - g1(A).*(0.5*E.*dU.^2 - P) - G.*A + 0.5*(Cntr(D.*A,h)...
    + D.*Cntr(A,h) - A.*Cntr(D,h));

% Next we deal with the boundary conditions which have nuemann BCs for
% damage.

% Boundary Conditions.
LeftBC = - g1(A(1))*(0.5*E(1).*dU(1).^2 - P(1)) - G(1).*A(1)...
        + (1/(h^2))*(D(2) + D(1))*(A(2) - A(1)) ;

RightBC = - g1(A(end))*(0.5*E(end).*dU(end).^2 - P(end)) - G(end).*A(end)...
        + (1/(h^2))*(D(end-1) + D(end)) * (A(end-1) - A(end)) ;

% Check Criterion at every step.
F = [LeftBC Q(2:end-1) RightBC];
end
```

A.5 Damage phase

In §4.2.4 we discussed how we calculate the damage phase field whilst assuming our current solution for the displacement is a fixed constant. This is done while using the criterion field to explain where damage evolution is taking place in the reference domain. With that information we can proceed to solve the damage evolution equation described by the system of equations (4.24). We use Newtons method to solve this system of equations as can be seen in Listing A.7. Since we are working in a one-dimensional discretised domain with n nodes then the number of discretised equations we will be solving is also n. This system will of course be non-linear because of the degradation function present in (4.24). Thus Newtons method is an appropriate iterative scheme for solving our discretised damage evolution equations. In Listing A.7 we have defined two other Matlab functions which calculate the solutions of the damage evolution equation and of course the required Jacobian matrix for Newtons Method.

The Listing A.8 calculates the required solution of the damage evolution equation (4.24) using the code in Listing A.6 which is discussed in §A.4. Here we also use $\mathrm{Crit}(X)$ to correctly inform our numerical scheme on where damage evolution is actually occurring. Multiplying element by element with $\mathrm{Crit}(X)$ will keep all the information required for any node where damage evolution is taking place, whilst setting all the other nodes to zero and thus ensuring those nodes do not experience damage evolution.

To calculate the Jacobian matrix we use the code in Listing A.9. This code makes use of sparse matrices to be as efficient as possible and allow us to work with really high resolution meshes. We also make use of $\mathrm{Crit}(X)$ to eliminate rows from the Jacobian that correspond to nodes not experiencing damage evolution. As a result any node that does not experience damage evolution will remain constant.

Listing A.7: This Matlab function uses Newtons method to calculate solutions for the damage phase field and solve the damage evolution equation defined by (4.24).

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %
3  % This function solves the damage phase of our problem using an
       iterative
4  % scheme.
5  %
6  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7  function A = DamagePhase(A,g1,g2,dU,E,P,G,D,Crit,X)
8
9  % Tolerance for newtons method and constant.
10 tol = 1e-12;
```

```
11
12 % Here we are iteratively going to calculate A(2,:) to do this
       we represent
13 % Y_curr and Y_old as the iterations of A(2,:).
14 Y_curr = A(2,:);
15 Y_old = -ones(size(X));
16
17 while norm(Y_curr - Y_old, "inf") > tol
18     Y_old = Y_curr;
19
20     % Create the RHS of our newtons equation
21     RHS = RightHandSide(A(2,:),g1,dU(2,:),E,P,G,D,Crit,X);
22
23     % Create the Jacobian
24     J = Jacobian(A(2,:),g2,dU(2,:),E,P,G,D,Crit,X);
25
26     % Update our solution.
27     B = J\RHS;
28     Y_curr = B' + Y_old;
29
30     % Update Damage Phase field and eliminate any rounding
           errors.
31     A(2,:) = max(Y_curr,A(1,:));
32 end
33
34 end
```

Listing A.8: This Matlab function calculates the solutions of the damage evolution equation for previous iterative solution of the damage phase field. In this function we make use of the function shown in Listing A.6.

```
1 %&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
2 %
3 % Creates the Right Hand side of our newtons method for non-
      linear
4 % damage phase field problem.
5 %
6 %&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
7 function RHS = RightHandSide(A,g1,dU,E,P,G,D,Crit,X)
```

```
8
9  % Calculate the damage equations for the known and unknowns.
10 F = DamageEquation(g1,A,dU,E,P,G,D,X);
11
12 % Use the criterion to setup the RHS properly.
13 RHS = -(Crit.*F)';
14 end
```

Listing A.9: This Matlab function calculates the required Jacobian matrix to solve our Newton method solution in Listing A.7. In this function we make use of sparse matrices to make this code more efficient allowing us to use a much higher resolution.

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %
3  % This calculates our Jacobian for our problem. This will be a
4  % very large sparse matrix.
5  %
6  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7  function J = Jacobian(A,g2,dU,E,P,G,D,Crit,X)
8
9  % Discretised step size for X.
10 h = (X(end) - X(1))/(length(X) - 1);
11 n = length(X);
12
13 % Saves us notation
14 Q = 0.5*(dU.^2).*E - P;
15
16 % Main body of derivatives
17 F = - g2(A(2:end-1)).*Q(2:end-1) - G(2:end-1)...
18             - (D(3:end) + 2*D(2:end-1) + D(1:end-2))/(2*h^2);
19 % Main upper diag.
20 F_U = (1/(2*h^2))*(D(2:end-1) + D(3:end));
21 %Main Lower diag
22 F_L = (1/(2*h^2))*(D(1:end-2) + D(2:end-1)) ;
23
24 % Create internal Jacobian
25 K = sparse(1:n-2,1:n-2,Crit(2:n-1).*F_L,n-2,n) +...
26     sparse(1:n-2,2:n-1,1 + Crit(2:n-1).*(F - 1),n-2,n) +...
27     sparse(1:n-2,3:n,Crit(2:n-1).*F_U,n-2,n) ;
```

```
28
29 clear F_L F_U F
30
31 % Next we build the left boundary condition.
32 L1 = - g2(A(1)).*Q(1) - G(1) - D(1)/(h^2);
33 L2 = D(1)/(h^2);
34
35 L = [1 0] + Crit(1).*([L1-1 L2]);
36
37 LeftBC = [L zeros(1,n-2)];
38
39 % Next we build the right boundary condition.
40 R1 = - g2(A(end)).*Q(end) - G(end) - D(end)/(h^2);
41 R2 = D(end)/(h^2);
42
43 R = [0 1] + Crit(n).*([R2 R1-1]);
44
45 RightBC = [zeros(1,n-2) R];
46
47 % Finally we create the jacobian matrix.
48 J = [LeftBC; K; RightBC];
49
50 end
```

Appendix B

Two-dimensional finite element code

This appendix contains the code used to perform all the simulations discussed in Chapter 5. For a breakdown of how the code is structured and the key ideas behind our numerical algorithm and schemes please see §5.3. Each of the following sections will give a brief description of what each script or function of code does. All of this code was designed for and ran in Matlab [72]. It should be noted however, many software languages are extremely similar and converting this code would not be an ordeal.

B.1 Mesh generation

The first step in our numerical method is to create a mesh approximating the macroscopic geometry, as discussed in §5.3.1. This mesh will encapsulate the referential domain. To generate the mesh we make use of Matlabs' partial differential equations toolbox. As this code is designed for a two dimensional geometry, we can define a number of vertices approximating the convex hull of our desired geometry. Next we make use of Matlabs' inbuilt tools to generate a triangulated mesh. Matlab does this by using a Delaunay triangulation which subdivides the convex hull of our macroscopic geometry into triangles whose circumcircles do not contain any of the nodes which are not the triangles vertices. We can set the maximum and minimum radii of these circumcircles. Thus controlling the size of our finite elements and the accuracy of our numerical solution. The output of this process is saved as two text files, the first is all the coordinates and the second is the corresponding connectivity matrix. All of this is done in code Listing B.1. It should be noted we run this Matlab script separate from the others as a piece of preprocessing work.

Listing B.1: This matlab code allows us to generate a macroscopic two-dimensional geoemtry. Then discretise it into a triangular mesh using Delaunay triangulation methods. All the functions in this code are inbuilt Matlab functions.

```
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %
```

```
% Here we create a delauney triangulation of our desired 2D
   geometry.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear DTR X Y TR q Nodes gm triangles
clc

% Coordinates for Polyshape
X = [1 0 0 1];
Y = [0 0 1 1];

% Firstly create a standard square mesh.
polyout = polyshape(X,Y);

% Create Mesh
TR = triangulation(polyout);
gm = fegeometry(TR);
gm = addVertex(gm, 'Coordinates', [1 0.5]);
gm = generateMesh(gm,"GeometricOrder","linear", Hmax = 0.01);

% Create Delauney Triangulation.
q = gm.Mesh;
triangles = q.Elements';
Nodes = q.Nodes';

Triangles_file = fopen('Triangles.txt','w');
    for i = 1:size(triangles,1)
        fprintf(Triangles_file,'%d,',triangles(i,:));
    end
fclose(Triangles_file);

Nodes_file = fopen('Nodes.txt','w');
    for j = 1:size(Nodes,1)
        fprintf(Nodes_file,'%d,',Nodes(j,:));
    end
fclose(Nodes_file);

disp(gm.Mesh)
```

```
figure
pdemesh(gm.Mesh)
axis padded
```

B.2 Input code

Now that we have setup our macroscopic geometry and its discretised mesh we must define all the other parameters our model relies on. This is done in Listing B.2 where we begin running our code by defining the microscopic material properties and geometry that are required for calculating the effective coefficients. We also must choose our time domain and the number of time steps we want to calculate a solution for. It is also important that we define the degradation function which plays a crucial role in damage evolution. When we run Listing B.2 our code will take all the information mentioned and calculate solutions at every time step saving all the data in accessible text files. For convenience at the end of Listing B.2 we produce some useful figures to help the user visualise the outcome of this process.

In Listing B.2 we call a number of custom functions for our convenience. The first of these is called "Elements" which loads into memory our Delaunay triangulation of the macroscopic geometry. This code can be found in Listing B.5. We also use custom Matlab functions to define the external loading being applied throughout the simulation. This is done in the functions "BoundaryCondition" and "AppliedForces", which define the traction and body force contributions of external loading at each time step, respectively. These codes can be found in Listings B.4 and B.3, respectively. Most importantly we run our function "FEMSolver" which is our implementation of Algorithm 2 and actually performs the jobs of solving our damage model.

Listing B.2: This script is how we input all the appropriate microscopic properties and geometry we want our microscale to have. We also setup the time steps we want to solve over and load into memory our discretised mesh. Next we calculate the effective macroscopic coefficients and use them to calulate our numerical solution at every time step. These solutions are loaded into memory for the macroscopic displacement and damage, then used to produce some useful figures.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This is our initial attempt at creating a FEM method to solve the
% equilibrium equation and damage equations in two dimensions.
%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% First we clear the cache and command window.
clear all; close all; clc;

% Just stop the silly beep.
beep off
warning('off','all')
format long

% Time Domain and steps we want to save.
tlim = [0 1 1e-8];
Nt = 1000;

% Read in our Delauney triangulation from saved files for linux.
DTR = Elements;

% Small Reminder of the grid size.
disp(['Its Alive!! The grid size is ' num2str(numel(DTR.Points(:,1))) ])

% Initial values of the system.
a0 = zeros(size(DTR.Points(:,1)));
u0 = zeros(size(DTR.Points(:,1)));
v0 = zeros(size(DTR.Points(:,1)));

% Traction BCs for trouser test.
Traction = @(x,y,t) BoundaryCondition(x,y,t);
BodyForce = @(x,y,t) AppliedForces(x,y,t);

% Define Our Material Properties.
% Note Poison Ratio = Lambda/2*(Lambda + Mu).
% Must be between -1 and 0.5
% The first entry in each vector are the host matrix and the second is the
% inclusion.
Lambda = 1000*[150 10]; % First Lame Parameter.
```

```
Mu = 1000*[150 10]; % Second Lame Parameter.

% Microscopic density, diffusion, threshold and viscosity.
rho = 10*[1 1];
diff = 0.05*[1 1];
energy = 0.1*[1 1];
psi = 0.01*[1 1];
eta = [1 1];

% Microscopic Geometry.
Lx = 1;
Ly = 1;
phi = 90;
R = sqrt(0.5/pi);
N = 300;

% Calculate our Macroscopic Parameters.
[C,Rho,D,G,Psi,Eta] = MacroscopicParameters(Lambda,Mu,rho,diff,
   energy,psi,eta,phi,Lx,Ly,R,N);

% tol sets the maxiumum and minimum  acceptable tolerances.
tol = 1e-6;

% Delta sets the acceptable max and min interval to delete a
   node.
delta = 3e-2;

% Degradation function, keep this piecewise scalar.
g = @(y) (1 - y).^2 + delta^2;
g1 = @(y) 2*y - 2;

tic
[Usol,Vsol,Asol,T,Torn,Tol] =  FEMsolver(g,g1,Psi,C,Rho,G,D,Eta
   ,Traction,BodyForce,u0,v0,a0,tlim,Nt,DTR,tol,delta);
toc

% Create points and triangles.
X = DTR.Points(:,1);
```

```
76 Y = DTR.Points(:,2);
77 triangles = DTR.ConnectivityList;
78
79 % Triangle of final Mesh.
80 [FinalTriangles,~] = UpdatedMesh(Asol(:,end),triangles,delta);
81
82 % Figure of our final mesh.
83 figure('visible','off')
84 hold on
85 trisurf(FinalTriangles,X,Y,Usol(:,end))
86 title('Horizontal Displacement')
87 xlabel('X-axis')
88 ylabel('Y-axis')
89 zlabel('U displacement')
90 xlim([min(X) max(X)])
91 ylim([min(Y) max(Y)])
92 view(3)
93 hold off
94 saveas(gcf,'figure1.jpg')
95
96 % Figure of our final mesh.
97 figure('visible','off')
98 hold on
99 trisurf(FinalTriangles,X,Y,Vsol(:,end))
100 title('Vertical Displacement')
101 xlabel('X-axis')
102 ylabel('Y-axis')
103 zlabel('V displacement')
104 xlim([min(X) max(X)])
105 ylim([min(Y) max(Y)])
106 view(3)
107 hold off
108 saveas(gcf,'figure2.jpg')
109
110 % Figure of our final mesh.
111 figure('visible','off')
112 hold on
113 trisurf(triangles,X,Y,Asol(:,end))
```

```
title('Damage Field')
xlabel('X-axis')
ylabel('Y-axis')
zlabel('Alpha')
xlim([min(X) max(X)])
ylim([min(Y) max(Y)])
zlim([0 1])
view(3)
hold off
saveas(gcf,'figure3.jpg')

% Figure of our final mesh.
figure('visible','off')
hold on
triplot(FinalTriangles,X+Usol(:,end),Y+Vsol(:,end))
title('Our Final Mesh')
xlabel('X-axis')
ylabel('Y-axis')
xlim([min(X+Usol(:,end))-0.5 max(X+Usol(:,end))+0.5])
ylim([min(Y+Vsol(:,end))-0.5 max(Y+Vsol(:,end))+0.5])
hold off
saveas(gcf,'figure4.jpg')

figure('visible','off')
hold on
semilogy(T,Tol);
title('Convergence at each time step')
xlabel('Time')
ylabel('Tolerance')
axis padded
yscale log
hold off
saveas(gcf,'figure6.jpg')
```

Listing B.3: This function allows us to define the body forces in our mesh for each time step. This is a very important part of our external loading as it allow us to pull apart our macroscopic mesh however we want.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
2  %
3  % Simple piecewise funcion for calculating the applied body
       forces.
4  %
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6  function Z = AppliedForces(x,y,t)
7
8  if y > 0.5 && x >= 0.8
9      % Upwards vertical force.
10     Z = 500*t*[0 1];
11 elseif y < 0.5 && x >= 0.8
12     % Downwards vertical force.
13     Z = -500*t*[0 1];
14 else
15     % No Force.
16     Z = [0 0];
17 end
```

Listing B.4: This function allows us to define the traction stress tensor that is applied on the boundary at each time step. Effectively this allows us to define a boundary condition for the displacement.

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%#
2  %
3  % Simple piecewise funcion for calculating the traction stress
       tensor
4  % applied at the boundary.
5  %
6  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7  function Z = BoundaryCondition(x,y,t)
8
9  if y > 0.5 && x == 1
10     % Upwards vertical force.
11     Z = 0*t*[0 1; 1 0];
12 elseif y < 0.5 && x == 1
13     % Downwards vertical force.
14     Z = 0*t*[0 1; 1 0];
15 else
16     % No Force.
```

```
    Z = [0 0; 0 0];
end
```

Listing B.5: In this matlab function we load into memory from two text files the coordinates and conectivity matrix that define our discretised mesh. We then use this data to form a Delaunay triangulation in Matlab and produce a plot of our referential mesh domain.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Elements function creates the triangles and mesh etc.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function DTR = Elements

% Read in presaved Mesh information.
T = readmatrix('Triangles.txt');
N =  readmatrix('Nodes.txt');

% Setup triangles and nodes vectors.
triangles = zeros(size(T,2)/3,3);
nodes = zeros(size(N,2)/2,2);

% Reshape.
for i = 1:size(T,2)/3
    triangles(i,:) = T(3*(i-1) + 1:3*i);
end
for i = 1:size(N,2)/2
    nodes(i,:) = N(2*(i-1) + 1:2*i);
end

% Build the delaunay triangulation of the grid points
DTR = triangulation(triangles,nodes);

% % % Figure of our reference mesh.
figure('visible','off')
hold on
triplot(DTR)
```

```
32 title('Our Reference Mesh')
33 xlabel('X-axis')
34 ylabel('Y-axis')
35 axis padded
36 hold off
37 saveas(gcf,'figure0.jpg')
38
39 end
```

B.3 Effective coefficients

Calculating the effective macroscopic coefficients allows us to encode microscopic information about a materials local properties and composite structure into our multiscale model. We define all of our microscopic properties and geometry first in the input script, Listing B.2. Then we use the function "MacroscopicParameters" to calculate our effective coefficients, as discussed in detail in §5.2.1. The code for this can be seen in Listing B.6. The first job this code performs is to define an array outlining the composite structure of our microscopic domain, this role is performed in the function "Inclusion". The code can be found in Listing B.7, where we define an array structure for every discrete point in our microscopic domain. In Listing B.7 we set each point to either be evaluated as a 1 or a 2. Referring back to this array will allow us to know what microscopic properties apply at which point. In Listing B.7 we can essentially setup the microscopic structure of our multiscale model. Here we can see that we have setup the scenario for a single circular inclusion in a host matrix, as outlined in §5.4.2.

Next we use this array and the microscopic properties of the elasticity and the diffusion to calculate our effective macroscopic elasticity and diffusion. To calculate these effective properties we use the Matlab codes provided by Andreasen et al. [3]. We will not report on this code here as it is not our own and recommend that the reader should look at cited paper. However, the codes are here for completeness and can be seen in Listing B.9 and Listing B.10, which respectively calculate the effective elasticity and diffusion. The remaining effective coefficients are simply local integral averages and we calculate them using our custom function "LocalIntegralAverage", the code for which can be found in Listing B.8. The purpose of this Matlab function is to calculate local integral averages as defined by (3.6), which we approximate using (5.14). With these Matlab codes we can calculate a great number of effective coefficients for a wide variety of microscopic scenarios.

Listing B.6: The role of this Matlab function is to outline the calulcation of our effective macroscopic coefficients. Firstly, the code generates an appropriate microscopic geometry as an array of points labelled with a number. Using this information we calculate the effective macroscopic coefficients of the diffusion and elasticity with a Matlab function provided by Andreasen et al. [3]. The remaining effective coefficients are calculated as local integral averages. We then define all of these coefficients as functions in Matlab using function handles so they can be quickly recalled at any point in our following calculations. We also have the oppourtunity to interpolate these effective coefficients if the microscale varies across the macroscale or apply some form of rotation if we are in a polar domain.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This function calculates the macroscopic parameters by solving the local
% cell problems then taking integral averages.
%
% Here the small letters are the microscopic parameters and capital letters
% are the macroscopic parametrs we are calculating.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [C,Rho,D,G,Psi,Eta] = MacroscopicParameters(Lambda,Mu,rho,diff,g,psi,eta,phi,Lx,Ly,R,N)

% Create the microscopic geometry.
% Inclusions creates an array of 1s and 2s for a circle at the centre, radius R,
% of the rectangle with dimensions Lx by Ly.
X = Inclusion(Lx,Ly,R,N);
figure('visible','off')
hold on
imagesc(X);
xlabel('X-axis')
ylabel('Y-axis')
axis tight
axis image
axis off
hold off
saveas(gcf,'figureMicro.jpg')
```

```
26
27 % Effective Macroscopic Parameters.
28 C = homogenizeElasticity(Lx,Ly,Lambda,Mu,phi,X);
29 D = homogenizeDiffusion(Lx,Ly,[0 0],diff,phi,X);
30
31 % Local Inegral Average Macroscopic Parameters.
32 Rho = LocalIntegralAverage(Lx,Ly,rho,X);
33 G = LocalIntegralAverage(Lx,Ly,g,X);
34 Psi = LocalIntegralAverage(Lx,Ly,psi,X);
35 Eta = LocalIntegralAverage(Lx,Ly,eta,X);
36
37
38 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
39 %
40 %   Second part of this code is for rotations and interpolation
    of results.
41 %
42 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
43 % Make spatial functions.
44 C = @(x,y) C;
45 Rho = @(x,y) Rho;
46 D = @(x,y) D;
47 G = @(x,y) G;
48 Psi = @(x,y) Psi;
49 Eta = @(x,y) Eta;
50 end
```

Listing B.7: This Matlab function creates our local periodic cell that defines the microscale. To do this we create an array where each subdomain of our microscale is defined by a whole number as a label. In the below example we create a circular inclusion at the centre of the host matrix.

```
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %
3 % This function creates a periodic cell with a circular
     inclusion at its
4 % centre. 1's for host and 2's for inclusion.
5 %
6 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function X = Inclusion(Lx, Ly, R, Nodes)

% Host matrix
X = ones(Nodes,Nodes);

% local dimensions.
x = linspace(0,Lx,Nodes);
y = linspace(0,Ly,Nodes);

% Circular inclusions.
for i = 1:Nodes
    for j = 1:Nodes
        if (x(i)-Lx/2)^2  + (y(j) - Ly/2)^2 - R^2 < 0
            X(j,i) = 2;
        else
            X(j,i) = 1;
        end
    end
end

end
```

Listing B.8: This Matlab function calculates the local integral averages as defined by (3.6). Here we make use of the approximation (5.14) as outlined in §5.2.1 to calculate the approximate local integral averages.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This function calculates the local integral averages of the
   microscopic
% parameters. Here Lx is the width of the cell and Ly is the
   height.
% Read Andreasen et al. for more detail. X is a map of the
   inclusions.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Y = LocalIntegralAverage(Lx,Ly,y,X)

% Volume of the cell.
```

```
V = Lx*Ly;
dV = V/numel(X);

% Calculate the integral average.
Y = (sum(sum(X == 1))*y(1)*dV + sum(sum(X == 2))*y(2)*dV)/V;

end
```

Listing B.9: This Matlab functions purpose is to calculate the effective coefficient of elasticity as explained in §5.2.1. To do this we have used this code provided by Andreassen et al. [3], for a better explanation of how the code works we recommend reading the publicaiton.

```
function CH = homogenizeElasticity(lx, ly, lambda, mu, phi, x)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% lx        = Unit cell length in x-direction.
% ly        = Unit cell length in y-direction.
% lambda    = Lame's first parameter for both materials. Two
   entries.
% mu        = Lame's second parameter for both materials. Two
   entries.
% phi       = Angle between horizontal and vertical cell wall.
   Degrees
% x         = Material indicator matrix. Size used to determine
    nelx/nely
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% INITIALIZE
% Deduce discretization
[nely, nelx] = size(x);
% Stiffness matrix consists of two parts, one belonging to
   lambda and
% one belonging to mu. Same goes for load vector
dx = lx/nelx; dy = ly/nely;
nel = nelx*nely;
[keLambda, keMu, feLambda, feMu] = elementMatVec(dx/2, dy/2,
   phi);
% Node numbers and element degrees of freedom for full (not
   periodic) mesh
nodenrs = reshape(1:(1+nelx)*(1+nely),1+nely,1+nelx);
```

```
edofVec = reshape(2*nodenrs(1:end-1,1:end-1)+1,nel,1);
edofMat = repmat(edofVec,1,8)+repmat([0 1 2*nely+[2 3 0 1] -2 -1],nel,1);
%% IMPOSE PERIODIC BOUNDARY CONDITIONS
% Use original edofMat to index into list with the periodic dofs
nn = (nelx+1)*(nely+1); % Total number of nodes
nnP = (nelx)*(nely);    % Total number of unique nodes
nnPArray = reshape(1:nnP, nely, nelx);
% Extend with a mirror of the top border
nnPArray(end+1,:) = nnPArray(1,:);
% Extend with a mirror of the left border
nnPArray(:,end+1) = nnPArray(:,1);
% Make a vector into which we can index using edofMat:
dofVector = zeros(2*nn, 1);
dofVector(1:2:end) = 2*nnPArray(:)-1;
dofVector(2:2:end) = 2*nnPArray(:);
edofMat = dofVector(edofMat);
ndof = 2*nnP; % Number of dofs
%% ASSEMBLE STIFFNESS MATRIX
% Indexing vectors
iK = kron(edofMat,ones(8,1))';
jK = kron(edofMat,ones(1,8))';
% Material properties in the different elements
lambda = lambda(1)*(x==1) + lambda(2)*(x==2);
mu     = mu(1)*(x==1) + mu(2)*(x==2);
% The corresponding stiffness matrix entries
sK = keLambda(:)*lambda(:).' + keMu(:)*mu(:).';
K  = sparse(iK(:), jK(:), sK(:), ndof, ndof);
% Ensure Matlab recognises K as a symmetric matrix
K = 0.5*(K + K');
%% LOAD VECTORS AND SOLUTION
% Assembly three load cases corresponding to the three strain cases
sF = feLambda(:)*lambda(:).'+feMu(:)*mu(:).';
iF = repmat(edofMat',3,1);
jF = [ones(8,nel); 2*ones(8,nel); 3*ones(8,nel)];
F  = sparse(iF(:), jF(:), sF(:), ndof, 3);
```

```
% Solve (remember to constrain one node)
chi(3:ndof,:) = K(3:ndof,3:ndof)\F(3:ndof,:);
%% HOMOGENIZATION
% The displacement vectors corresponding to the unit strain cases
chi0 = zeros(nel, 8, 3);
% The element displacements for the three unit strains
chi0_e = zeros(8, 3);
ke = keMu + keLambda; % Here the exact ratio does not matter, because
fe = feMu + feLambda; % it is reflected in the load vector
chi0_e([3 5:end],:) = ke([3 5:end],[3 5:end])\fe([3 5:end],:);
% epsilon0_11 = (1, 0, 0)
chi0(:,:,1) = kron(chi0_e(:,1)', ones(nel,1));
% epsilon0_22 = (0, 1, 0)
chi0(:,:,2) = kron(chi0_e(:,2)', ones(nel,1));
% epsilon0_12 = (0, 0, 1)
chi0(:,:,3) = kron(chi0_e(:,3)', ones(nel,1));
CH = zeros(3);
cellVolume = lx*ly;
for i = 1:3
  for j = 1:3
    sumLambda = ((chi0(:,:,i) - chi(edofMat+(i-1)*ndof))*keLambda).*...
      (chi0(:,:,j) - chi(edofMat+(j-1)*ndof));
    sumMu = ((chi0(:,:,i) - chi(edofMat+(i-1)*ndof))*keMu).*...
      (chi0(:,:,j) - chi(edofMat+(j-1)*ndof));
    sumLambda = reshape(sum(sumLambda,2), nely, nelx);
    sumMu = reshape(sum(sumMu,2), nely, nelx);
    % Homogenized elasticity tensor
    CH(i,j) = 1/cellVolume*sum(sum(lambda.*sumLambda + mu.*sumMu));
  end
end
CH = 0.5*(CH + CH');
disp('--- Homogenized elasticity tensor ---'); disp(CH)

%% COMPUTE ELEMENT STIFFNESS MATRIX AND FORCE VECTOR (
```

```
   NUMERICALLY)
function [keLambda, keMu, feLambda, feMu] = elementMatVec(a, b,
    phi)

% Constitutive matrix contributions
CMu = diag([2 2 1]); CLambda = zeros(3); CLambda(1:2,1:2) = 1;

% Define the quadrature for integration
% Two Gauss points in both directions
xx=[-1/sqrt(3), 1/sqrt(3)]; yy = xx;
ww=[1,1];

% Initialize
keLambda = zeros(8,8); keMu = zeros(8,8);
feLambda = zeros(8,3); feMu = zeros(8,3);

%
L = zeros(3,4); L(1,1) = 1; L(2,4) = 1; L(3,2:3) = 1;

for ii=1:length(xx)
  for jj=1:length(yy)
    % Integration point
    x = xx(ii); y = yy(jj);
    % Differentiated shape functions
    dNx = 1/4*[-(1-y)  (1-y) (1+y) -(1+y)];
    dNy = 1/4*[-(1-x) -(1+x) (1+x)  (1-x)];
    % Jacobian
    J = [dNx; dNy]*[-a a a+2*b/tan(phi*pi/180) 2*b/tan(phi*pi
       /180)-a; ...
         -b -b b b]';
    detJ = J(1,1)*J(2,2) - J(1,2)*J(2,1);
    invJ = 1/detJ*[J(2,2) -J(1,2); -J(2,1) J(1,1)];
    % Weight factor at this point
    weight = ww(ii)*ww(jj)*detJ;
    % Strain-displacement matrix
    G = [invJ zeros(2); zeros(2) invJ];
    dN = zeros(4,8);
    dN(1,1:2:8) = dNx;
```

```
124     dN(2,1:2:8) = dNy;
125     dN(3,2:2:8) = dNx;
126     dN(4,2:2:8) = dNy;
127     B = L*G*dN;
128     % Element matrices
129     keLambda = keLambda + weight*(B' * CLambda * B);
130     keMu = keMu + weight*(B' * CMu * B);
131     % Element loads
132     feLambda = feLambda + weight*(B' * CLambda * diag([1 1 1]))
            ;
133     feMu = feMu + weight*(B' * CMu * diag([1 1 1]));
134   end
135 end
```

Listing B.10: This Matlab functions purpose is to calculate the effective coefficient of Diffusion as explained in §5.2.1. To do this we have used this code provided by Andreassen et al. [3], for a better explanation of how the code works we recommend reading the publicaiton.

```
1  function D = homogenizeDiffusion(lx, ly, lambda, mu, phi, x)
2  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3  % lx        = Unit cell length in x-direction.
4  % ly        = Unit cell length in y-direction.
5  % lambda    = Lame's first parameter for both materials. Two
       entries.
6  % mu        = Lame's second parameter for both materials. Two
       entries.
7  % phi       = Angle between horizontal and vertical cell wall.
       Degrees
8  % x         = Material indicator matrix. Size used to determine
        nelx/nely
9  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
10 %% INITIALIZE
11 % Deduce discretization
12 [nely, nelx] = size(x);
13 % Stiffness matrix consists of two parts, one belonging to
       lambda and
14 % one belonging to mu. Same goes for load vector
15 dx = lx/nelx; dy = ly/nely;
```

```
nel = nelx*nely;
[keLambda, keMu, feLambda, feMu] = elementMatVec(dx/2, dy/2, phi);
keMu(1:2:end,1:2:end) = keMu(1:2:end,1:2:end) + keMu(2:2:end, 2:2:end);
% Node numbers and element degrees of freedom for full (not periodic) mesh
nodenrs = reshape(1:(1+nelx)*(1+nely),1+nely,1+nelx);
edofVec = reshape(2*nodenrs(1:end-1,1:end-1)+1,nel,1);
edofMat = repmat(edofVec,1,8)+repmat([0 1 2*nely+[2 3 0 1] -2 -1],nel,1);
%% IMPOSE PERIODIC BOUNDARY CONDITIONS
% Use original edofMat to index into list with the periodic dofs
nn = (nelx+1)*(nely+1); % Total number of nodes
nnP = (nelx)*(nely);    % Total number of unique nodes
nnPArray = reshape(1:nnP, nely, nelx);
% Extend with a mirror of the top border
nnPArray(end+1,:) = nnPArray(1,:);
% Extend with a mirror of the left border
nnPArray(:,end+1) = nnPArray(:,1);
% Make a vector into which we can index using edofMat:
dofVector = zeros(2*nn, 1);
dofVector(1:2:end) = 2*nnPArray(:)-1;
dofVector(2:2:end) = 2*nnPArray(:);
edofMat = dofVector(edofMat);
ndof = 2*nnP; % Number of dofs
%% ASSEMBLE STIFFNESS MATRIX
% Indexing vectors
iK = kron(edofMat,ones(8,1))';
jK = kron(edofMat,ones(1,8))';
% Material properties in the different elements
lambda = lambda(1)*(x==1) + lambda(2)*(x==2);
mu     = mu(1)*(x==1) + mu(2)*(x==2);
% The corresponding stiffness matrix entries
sK = keLambda(:)*lambda(:).' + keMu(:)*mu(:).';
K  = sparse(iK(:), jK(:), sK(:), ndof, ndof);
%% LOAD VECTORS AND SOLUTION
```

```
% Assembly three load cases corresponding to the three strain cases
sF = feLambda(:)*lambda(:).'+feMu(:)*mu(:).';
iF = repmat(edofMat',3,1);
jF = [ones(8,nel); 2*ones(8,nel); 3*ones(8,nel)];
F  = sparse(iF(:), jF(:), sF(:), ndof, 3);
% Solve (remember to constrain one node)
chi = zeros(ndof,2); chi(3:2:ndof,:) = K(3:2:end,3:2:end)\[F(3:2:end,1) F(4:2:end,2)];
%% HOMOGENIZATION
% The displacement vectors corresponding to the unit strain cases
chi0 = zeros(nel, 8, 3);
% The element displacements for the three unit strains
chi0_e = zeros(8, 3);
% ke = keMu + keLambda; % Here the exact ratio does not matter, because
% fe = feMu + feLambda; % it is reflected in the load vector
chi0_e(3:2:end,1:2) = keMu(3:2:end,3:2:end)\[feMu(3:2:end,1) feMu(4:2:end,2)];
% epsilon0_11 = (1, 0, 0)
chi0(:,:,1) = kron(chi0_e(:,1)', ones(nel,1));
% epsilon0_22 = (0, 1, 0)
chi0(:,:,2) = kron(chi0_e(:,2)', ones(nel,1));
% epsilon0_12 = (0, 0, 1)
chi0(:,:,3) = kron(chi0_e(:,3)', ones(nel,1));
CH = zeros(3);
cellVolume = lx*ly;
for i = 1:2
  for j = 1:2
    sumLambda = ((chi0(:,:,i) - chi(edofMat+(i-1)*ndof))*keLambda).*...
      (chi0(:,:,j) - chi(edofMat+(j-1)*ndof));
    sumMu = ((chi0(:,:,i) - chi(edofMat+(i-1)*ndof))*keMu).*...
      (chi0(:,:,j) - chi(edofMat+(j-1)*ndof));
    sumLambda = reshape(sum(sumLambda,2), nely, nelx);
    sumMu = reshape(sum(sumMu,2), nely, nelx);
    % Homogenized elasticity tensor
```

```
81      CH(i,j) = 1/cellVolume*sum(sum(lambda.*sumLambda + mu.*
           sumMu));
82    end
83  end
84  % Grab diffusion tensor from results.
85  D = CH(1:2,1:2);
86  disp('--- Homogenized diffusion tensor ---'); disp(D)
87
88  %% COMPUTE ELEMENT STIFFNESS MATRIX AND FORCE VECTOR (
       NUMERICALLY)
89  function [keLambda, keMu, feLambda, feMu] = elementMatVec(a, b,
        phi)
90  % Constitutive matrix contributions
91  CMu = diag([1 1 0]); CLambda = zeros(3);
92  % Two Gauss points in both directions
93  xx=[-1/sqrt(3), 1/sqrt(3)]; yy = xx;
94  ww=[1,1];
95  % Initialize
96  keLambda = zeros(8,8); keMu = zeros(8,8);
97  feLambda = zeros(8,3); feMu = zeros(8,3);
98  L = zeros(3,4); L(1,1) = 1; L(2,4) = 1; L(3,2:3) = 1;
99  for ii=1:length(xx)
100   for jj=1:length(yy)
101     % Integration point
102     x = xx(ii); y = yy(jj);
103     % Differentiated shape functions
104     dNx = 1/4*[-(1-y)  (1-y) (1+y) -(1+y)];
105     dNy = 1/4*[-(1-x) -(1+x) (1+x)  (1-x)];
106     % Jacobian
107     J = [dNx; dNy]*[-a a a+2*b/tan(phi*pi/180) 2*b/tan(phi*pi
           /180)-a; ...
108          -b -b b b]';
109     detJ = J(1,1)*J(2,2) - J(1,2)*J(2,1);
110     invJ = 1/detJ*[J(2,2) -J(1,2); -J(2,1) J(1,1)];
111     % Weight factor at this point
112     weight = ww(ii)*ww(jj)*detJ;
113     % Strain-displacement matrix
114     G = [invJ zeros(2); zeros(2) invJ];
```

```
115     dN = zeros(4,8);
116     dN(1,1:2:8) = dNx;
117     dN(2,1:2:8) = dNy;
118     dN(3,2:2:8) = dNx;
119     dN(4,2:2:8) = dNy;
120     B = L*G*dN;
121     % Element matrices
122     keLambda = keLambda + weight*(B' * CLambda * B);
123     keMu = keMu + weight*(B' * CMu * B);
124     % Element loads
125     feLambda = feLambda + weight*(B' * CLambda * diag([1 1 1]))
            ;
126     feMu = feMu + weight*(B' * CMu * diag([1 1 1]));
127   end
128 end
```

B.4 Numerical algorithm

In Chapter 5 we outlined Algorithm 2 in §5.3.3 for solving our multiscale model step by step. We have implemented that numerical algorithm here in Matlab and this can be found in Listing B.11. This code is extremely similar to Listing A.2 in Appendix A which implements numerical algorithm 1. The main difference between these Listings is that the one in this section must account for a changing mesh. We have also implemented an adaptive time step method in this code to ensure that at each time step the solution converges to a desired tolerance. Importantly, we setup a serious of text files in our implementation to save all the important data we want to calculate, this data is only saved at the desired time steps. We create text files for the horizontal and vertical displacements, the damage at every point, which nodes are torn, the time at each step and the tolerance each step converged to.

When we use this implementation the first thing we do is calculate the first derivative of the degradation function using Listing B.12. From there we perform an iterative loop to solve every time step. At each time step we call the function "ElasticPhase" to calculate the displacement of our system. Next we use the function "CriterionCheck" to calculate the criterion at every node at our system. If at our time step the damage criterion is not satisfied at every point we then enter a set of iterated loops to calculate the damage evolution and potential changes in the mesh due to tearing. To do this we iteratively loop over the "ElasticPhase" and the "DamagePhase" functions. When the solutions from "DamagePhase" for the damage phase field have converged we cease the iterative solution.

Next we check if our mesh needs to be updated. If it does then we iteratively loop again and check that the sudden change in the mesh still allows our solution to converge for this time step. If it does the time step is complete and we save the data and move onto the next time step until all the steps are complete.

If we fail to converge at any point of the time step then we use our inbuilt adaptive time stepping to perform a smaller step forward in time. However, if the time steps becomes to small due to failure to converge then we halt the code and return the solutions as they are. This may happen because the parameters are poorly defined. It can also happen when the mesh becomes totally degraded due to many nodes being deleted and the damage phase field becoming prominent everywhere, leading to all the material properties degrading everywhere. This causes the numerical method to become unstable and then the model fails. To overcome this we recommend using an extremely fine mesh with a model that also has a very small diffusion.

Listing B.11: This Matlab function is the implementation of Algorithm 2 as outlined in §5.3.3. Following the algorithm step by step allows us to solve for our displacement and damage at every time step. The solutions at each time step are saved in text files so that the memory in used efficiently. Our code also makes use of an adaptive time stepping method to ensure that the code always converges to a desired tolerance. Failure to converge for an adequately small time step will cause the iterative scheme to halt and report back the solutions up to the most recently solved time step.

```
1  function [Usol,Vsol,Asol,T,Torn,Tol] = FEMsolver(g,g1,Psi,C,Rho
       ,G,D,eta,Traction,BodyForce,u0,v0,a0,tlim,Nt,DTR,tol,delta)
2
3  % All the solved time steps would be saved to respective text
       files for the
4  % horizontal displacement, vertical displacement and damage.
       Then at the
5  % end of this scheme we will reload those time steps as pages
       in Usol, Vsol
6  % and Asol respectively.
7  Horizontal_file = fopen('Horizontal.txt','w');
8  Vertical_file = fopen('Vertical.txt','w');
9  Damage_file = fopen('Damage.txt','w');
10 Torn_file = fopen('Torn.txt','w');
11 Time_file = fopen('Time.txt','w');
12 Tol_file = fopen('Tolerance.txt','w');
13
14 % We define our TornNodes which records where a tear has
```

```
   occured.
TornNodes = zeros(numel(DTR.Points(:,1)),2);

% Setup the current and previous time steps for damage and
   displacement.
U = [u0 u0];
V = [v0 v0];
A = [a0 a0];

% % Next we calculate the first and second partial derivative
   of the
% % degradation function for convience.
% [g1,~] = DegradationPartials(g);

% Now we move onto the first time step to be solved, so
   importantly we
% calculate the step size for each time steps as follows.
step = (tlim(2) - tlim(1))/(Nt-1);

% Firstly we caculate the first time step we will be solving
   for.
t = tlim(1) + step;
t_frame = t;
t0 = tlim(1);

% Secondly, set up triangles and coordinates.
X = DTR.Points(:,1);
Y = DTR.Points(:,2);
OriginalTriangles = DTR.ConnectivityList;
triangles = OriginalTriangles;

% Setup data printing.
fprintf(Horizontal_file,'%d,',u0);
fprintf(Vertical_file,'%d,',v0);
fprintf(Damage_file,'%d,',a0);
fprintf(Torn_file,'%d,',TornNodes(:,1));
fprintf(Time_file,'%d,',t0);
fprintf(Tol_file,'%d,',tol);
```

```
48
49 % Setup Not Converging condition and StepCondition.
50 NotConverging = false;
51 StepCondition = true;
52
53 % Loop we are solving.
54 while t <= tlim(2)
55
56     % Define Boundary Conditions for this time step.
57     BC = @(x,y) Traction(x,y,t);
58     BF = @(x,y) BodyForce(x,y,t);
59
60     % Solve the equilibrium equation at this time step.
61     [U(:,2),V(:,2)] = ElasticPhase(g,A(:,1),C,Rho,triangles,X,Y
          ,BC,BF);
62
63 %   Setup tolerance,
64     K = 0;
65
66 %   Check the criterion at every time step.
67     Crit = CriterionCheck(g1,A(:,2),U(:,2),V(:,2),Psi,C,G,D,
          triangles,X,Y,TornNodes(:,2));
68
69 %     If We have Critical points then we must solve a damage
       problem.
70     if sum(Crit == 1) > 0
71 %         Now we set up an iterative process to solve for A
       (:,2).
72         [~,T_curr] = UpdatedMesh(A(:,2),OriginalTriangles,delta
              );
73         T_old = -ones(size(T_curr));
74
75         while norm(T_curr - T_old,"inf") ~= 0 && ~NotConverging
76     %           Update T_old.
77             T_old = T_curr;
78
79     %         Now we set up an iterative process to solve for A
          (:,2).
```

```
80              Y_curr = [A(:,1) A(:,2)];
81              Y_old = -ones(size(Y_curr));
82      %         Setup adaptive time step condition.
83              Condition_Old = -1;
84              Condition_New = norm(Y_curr(:,2) - Y_old(:,2), Inf)
                    ;
85
86              % Setup Counter
87              stuck = false;
88              SkipStep = true;
89              N = 0;
90              while abs(Condition_New) > tol || N < 5
91                  % Update Y_old.
92                  Y_old = Y_curr;
93
94  %                 Adaptive time step Method.
95                  if stuck || (Condition_New/Condition_Old >= 1
                        && N  > 10)
96
97                      % Adaptive Time Steps.
98                      step = step/10;
99
100                     % First case the time step is smaller than
                            acceptable.
101                     if step < tlim(3)
102                         % Break and end the program.
103                         NotConverging = true;
104                         break
105
106                     % Second case the time step is in
                            acceptable range.
107                     elseif step >= tlim(3)
108                         t = t0 + step;
109
110                         % Reset all fields.
111                         U(:,2) = U(:,1);
112                         V(:,2) = V(:,1);
113                         A(:,2) = A(:,1);
```

```
114
115                         % Reset Torn Nodes.
116                         TornNodes(:,2) = TornNodes(:,1);
117
118                         % Reset BCs.
119                         BC = @(x,y) Traction(x,y,t);
120                         BF = @(x,y) BodyForce(x,y,t);
121
122                         % Reset our Mesh appropriatley.
123                         [triangles, ~] = UpdatedMesh(A(:,2),
                               OriginalTriangles,delta);
124                         T_curr = TornNodes(:,2);
125
126                         % Reset iterative process.
127                         Y_curr = A;
128                         Y_old = -ones(size(Y_curr));
129
130                         % Keep Count.
131                         Condition_New = -1;
132                         T_old = T_curr;
133                         StepCondition = false;
134                         SkipStep = true;
135                         N = 0;
136                         K = 0;
137                     end
138                 end
139
140                 % If all triangles are destroyed stop and send
                       a warning.
141                 if isempty(triangles)
142                     NotConverging = true;
143                     disp(['Connectivity Matrix is now empty, at
                           time '...
144                         num2str(t,10)])
145                     break
146                 end
147
148     %             Now we resolve the elastic Phase;
```

```
149                 [U(:,2),V(:,2)] = ElasticPhase(g,Y_curr(:,2),C,
                        Rho,triangles,X,Y,BC,BF);
150
151     %             Then we check if we have satisfied the
            criterion.
152                 Crit = CriterionCheck(g1,Y_curr(:,2),U(:,2),V
                        (:,2),Psi,C,G,D,triangles,X,Y,TornNodes(:,2)
                        );
153
154     %             Solve the damage problem.
155                 [Y_curr(:,2), stuck] = DamagePhase(g1,Y_curr,U,
                        V,Psi,C,G,D,eta,Crit,triangles,X,Y,step,tol,
                        TornNodes(:,2));
156
157                 if ~(stuck) && ~(SkipStep)
158                     % Adaptive time step Condition.
159                     Condition_Old = Condition_New;
160                     Condition_New = norm(Y_curr(:,2) - Y_old
                            (:,2), Inf);
161
162                     % Update Alpha.
163                     A(:,2) = Y_curr(:,2);
164                 elseif SkipStep == true
165                     SkipStep = false;
166                 end
167
168                 % Keep count of iterations.
169                 N = N + 1;
170             end
171
172             if any(A(:,2).*(1 - TornNodes(:,2)) >= 1 - delta)
173                 % Update our Mesh appropriatley.
174                 [triangles, TornNodes(:,2)] = UpdatedMesh(A
                        (:,2),OriginalTriangles,delta);
175                 T_curr = TornNodes(:,2);
176             end
177             K = max(K,Condition_New);
178         end
```

```
179
180         % Adaptive time stepping
181         if StepCondition && N <= 10
182             step = 10*step;
183                 if step > (tlim(2) - tlim(1))/(Nt-1)
184                     % Reset the step size if convergence is
                           fast.
185                     step = (tlim(2) - tlim(1))/(Nt-1);
186                 end
187         else
188             StepCondition = true;
189         end
190     end
191
192     % Save Data
193     if (~NotConverging)
194         if t >= t_frame
195             % Save data for this time step.
196             fprintf(Horizontal_file,'%d,',U(:,2));
197             fprintf(Vertical_file,'%d,',V(:,2));
198             fprintf(Damage_file,'%d,',A(:,2));
199             fprintf(Torn_file,'%d,',TornNodes(:,2));
200             fprintf(Time_file,'%d,',t);
201             fprintf(Tol_file,'%d,',K);
202
203             % New t_frame.
204             t_frame = t_frame + (tlim(2) - tlim(1))/(Nt-1);
205         end
206
207     % Update our current damage and displacement fields.
208     A = [A(:,2) A(:,2)];
209     U = [U(:,2) U(:,2)];
210     V = [V(:,2) V(:,2)];
211     TornNodes = [TornNodes(:,2) TornNodes(:,2)];
212
213     % Stop Criterion is code is not converging.
214     else
215        break
```

```
216     end
217
218     % Message display.
219     disp(['At time ' num2str(t,10) ' number of torn points is '
           ...
220         num2str(sum(TornNodes(:,1)),10)  ', the max damage is '
              ...
221         num2str(max(A(:,2).*(1-TornNodes(:,1))),10)...
222         ' and tolerance is ' num2str(K,4)'.'])
223
224     % Update time.
225     t0 = t;
226     if sum(Crit == 1) == 0
227         t = t0 + (tlim(2) - tlim(1))/(Nt-1);
228     else
229         t = t0 + step;
230     end
231 end
232
233 if NotConverging == false
234     % Display that we are finished.
235     disp('Completed all time steps.')
236 end
237
238 % Here we close the open text files which are saved local in
      this folder.
239 fclose(Horizontal_file);
240 fclose(Vertical_file);
241 fclose(Damage_file);
242 fclose(Torn_file);
243 fclose(Time_file);
244 fclose(Tol_file);
245
246 % Read in all the data as a single vector.
247 U = readmatrix('Horizontal.txt');
248 V = readmatrix('Vertical.txt');
249 A = readmatrix('Damage.txt');
250 Torn = readmatrix('Torn.txt');
```

```
T = readmatrix('Time.txt');
Tol = readmatrix('Tolerance.txt');

% This fixes weird bug where readmatrix adds an extra entry of
   NaN at the
% end of U and V. Very strange indeed.
RemainderU = rem(length(U),numel(X));
if RemainderU > 0
    U(end) = [];
end
RemainderV = rem(length(V),numel(X));
if RemainderV > 0
    V(end) = [];
end
RemainderA = rem(length(A),numel(X));
if RemainderA > 0
    A(end) = [];
end
RemainderTorn = rem(length(Torn),numel(X));
if RemainderTorn > 0
    Torn(end) = [];
end

% Reshape these nodes into spatial meshes, each page is a time
   step.
Usol = reshape(U,numel(X),[]);
Vsol = reshape(V,numel(X),[]);
Asol = reshape(A,numel(X),[]);
Torn = reshape(Torn,numel(X),[]);

% Finally Reshape Time and Tol.
RemainderT = rem(length(T),size(Usol,2));
if RemainderT > 0
    T(end) = [];
end
RemainderTol = rem(length(Tol),size(Usol,2));
if RemainderTol > 0
    Tol(end) = [];
```

```
287 end
288
289 % Reshape these nodes into vectors, each row is a time step.
290 T = reshape(T,[],1);
291 Tol = reshape(Tol,[],1);
292
293 end
```

Listing B.12: This Matlab function calculates the first and second derivatives of the degradation function. It then saves the solution as a function handle that we can call upon at any point in the code.

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %
3  % Calculate Partial Derivative of Degradation Function.
4  %
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6  function [g1,g2] = DegradationPartials(g)
7  syms y
8
9  g1 = str2func(['@(y)' char(diff(g(y)))]);
10
11 g2 = str2func(['@(y)' char(diff(g(y),2))]);
12
13 clear y
14 end
```

B.5 Elastic phase

In §5.3.4 we outline with the finite element method how to calculate the displacement from our equation of motion (5.3a), whilst assuming that the damage phase field remains constant. Implementing the finite element method can be a very delicate process and difficult to debug. We implemented numerical scheme in Listing B.13 for calculating the displacement. Doing so required a number of custom Matlab functions each for doing very particular jobs. Firstly though, we should note that to make our code as efficient as possible we perform all our calculations with sparse matrices. For convenience, we also create an array at the start of this calculation of all the surface normal unit vectors at each boundary node, by using Listing B.14. We also use inbuilt Matlab tools to create a list of all the nodes in our system which are boundary nodes so we can correctly apply

boundary conditions. With this information our code can begin to create a sparse matrix by iterating through every node in the system, where each row of our sparse matrix represents a particular node. For each row of the sparse matrix we calculate the correct corresponding columns which represent the interaction between the current node we are interested in and one of its neighbouring nodes in our Delaunay triangulation. We use Listing B.15 to find each of these neighbouring nodes commonly shared triangles, thus defining a domain for us to integrate over. To do these calculations we use our numerical scheme in §5.3a, which tells us every column we are calculating corresponds to an integral coefficient. Using our custom function "DisplacementMatrix", Listing B.16, we are able to calculate the correct integral coefficient for each matrix element.

Listing B.16 for each node and a neighbouring node takes into account all the commonly shared triangles and then performs the necessary integral calculation over each triangle, iteratively summing the solution. These integrals for each triangle are calculated by the function "ElementIntegralDisplacement" which is Listing B.17. In this Listing we use a Newtons-Cotes method for approximating all of the integrals. This method is very accurate even with only three nodes for each integral. As the triangles are going to be very small then the three nodes provide a great approximation. Of course if we defined our triangles quadratically with six nodes the accuracy would increase to a much higher order. These integrals require us to calculate the gradient of the piecewise shape functions, which we can quickly calculate using a Matlab function, Listing B.18.

So far we have only explained how to produce the mass matrix for the numerical scheme implemented in Listing B.13. This leaves the external loading contributions from traction and surface forces. It should be noted though that one could easily implement Dirichlect boundary conditions at each node instead of external loading. The external loading contributions are calculated by the functions "SurfaceForces" and "BodyForces". Surface forces are calculated in Listing B.19 and the Body forces B.20 are calculated in Listing. With all parts of the numerical scheme calculated we can finally solve a simple linear system to gain our solution for the horizontal and vertical displacement contributions.

Listing B.13: This Matlab function implements the numerical schemes outlined in §5.3.4, for calculating the solutions of the displacement whilst assuming that α remains constant everywhere. The numerical scheme implemented in this listing is a finite element scheme. With this code we can apply a number of different external loading conditions or Dirichlect boundary conditions.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Elastic Phase to solve the displacement problem.
%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [U,V] = ElasticPhase(g,A,C,R,triangles,X,Y,BC,BF)

% Stiffness matrix is initially an empty (N+1)x(N+1) matrix
Nodes = numel(X);

% Setup Trimesh system.
TR = triangulation(triangles,X,Y);

% Find Boundary Points.
Facets = freeBoundary(TR);
Boundary = unique(Facets);

% Calculate the NormalVector.
NormalVector = BoundaryNorms(Boundary,Facets,X,Y);

% % Stop NaN.
% [tt,~] = find(A == 1);
% A(tt) = 1 - tol;

% Setup a counter for I,J,V which are the rows, cols and values for their
% respective sparse matrices. Here N just progresses the rows they are on.
N = 0;

I11 = zeros(10*Nodes,1);
J11 = zeros(10*Nodes,1);
V11 = zeros(10*Nodes,1);

I12 = zeros(10*Nodes,1);
J12 = zeros(10*Nodes,1);
V21 = zeros(10*Nodes,1);

I21 = zeros(10*Nodes,1);
J21 = zeros(10*Nodes,1);
V12 = zeros(10*Nodes,1);

```

```
41 I22 = zeros(10*Nodes,1);
42 J22 = zeros(10*Nodes,1);
43 V22 = zeros(10*Nodes,1);
44
45 % RHS of the system representing body and surface forces, this
      term incorporates
46 % the boundary conditions.
47 RHS1 = zeros(Nodes,1);
48 RHS2 = zeros(Nodes,1);
49
50 % For every grid point.
51 for j = 1:Nodes
52     % Obtain the index of the node.
53     idx_i = j;
54     % Filter out torn points from our linear FEM system.
55     if sum(sum(triangles == idx_i)) == 0
56         RHS1(idx_i) = 0;
57         RHS2(idx_i) = 0;
58
59         I11(N+1) = idx_i;
60         J11(N+1) = idx_i;
61         V11(N+1) = 1;
62
63         I22(N+1) = idx_i;
64         J22(N+1) = idx_i;
65         V22(N+1) = 1;
66
67         N = N+1;
68
69     % Now to solve the actual linear problem.
70     else
71     % Find all the triangles containing this node.
72     [tt,~] = find(triangles == idx_i);
73
74     % Now we find all the adjacent nodes connected by the
          triangles.
75     idx_k = unique(triangles(tt,:));
76
```

```
77      % Establish if this is a boundary node,
78      if any(j == Boundary)
79                  % Left edge.
80              if X(idx_i) == min(X)
81                  RHS1(idx_i) = 0;
82                  RHS2(idx_i) = 0;
83
84                  I11(N+1) = idx_i;
85                  J11(N+1) = idx_i;
86                  V11(N+1) = 1;
87
88                  I22(N+1) = idx_i;
89                  J22(N+1) = idx_i;
90                  V22(N+1) = 1;
91
92                  N = N+1;
93                  % Right edge.
94              elseif X(idx_i) == max(X)
95                  % % Dirchlect BCs
96                  % RHS1(idx_i) = 0;
97                  % RHS2(idx_i) = BC(X(idx_i),Y(idx_i));
98                  %
99                  % I11(N+1) = idx_i;
100                 % J11(N+1) = idx_i;
101                 % V11(N+1) = 1;
102                 %
103                 % I22(N+1) = idx_i;
104                 % J22(N+1) = idx_i;
105                 % V22(N+1) = 1;
106                 %
107                 % N = N+1;
108
109                 S = SurfaceForces(NormalVector,Boundary,BC,X,Y,
                        idx_i,idx_k);
110                 % B = BodyForces(R,BF,X,Y,triangles,idx_i);
111
112                 RHS1(idx_i) = S(1);
113                 RHS2(idx_i) = S(2);
```

```
114                 % Internal Stress.
115                 for i = 1:length(idx_k)
116                     idx = idx_k(i);
117 
118                     % Obtain the triangles that contain both
                           idx_i and idx
119                     Domain = LocalElements(triangles,idx_i,idx)
                           ;
120 
121                     % Here We generate the stiffness Matrix
122                     [Q11, Q12, Q21, Q22] = DisplacementMatrix(g
                           ,Domain,idx_i,idx,A,C,X,Y);
123 
124                     % Log the entries for each matrix.
125                     I11(N+1) = idx_i;
126                     J11(N+1) = idx;
127                     V11(N+1) = Q11;
128 
129                     I12(N+1) = idx_i;
130                     J12(N+1) = idx;
131                     V12(N+1) = Q12;
132 
133                     I21(N+1) = idx_i;
134                     J21(N+1) = idx;
135                     V21(N+1) = Q21;
136 
137                     I22(N+1) = idx_i;
138                     J22(N+1) = idx;
139                     V22(N+1) = Q22;
140 
141                     N = N+1;
142                 end
143             else
144                 % Traction and Surface forces affects RHS of
                       the sytem.
145                 S = SurfaceForces(NormalVector,Boundary,BC,X,Y,
                       idx_i,idx_k);
146                 % B = BodyForces(R,BF,X,Y,triangles,idx_i);
```

```
147
148                 RHS1(idx_i) = S(1);
149                 RHS2(idx_i) = S(2);
150                 for i = 1:length(idx_k)
151                     idx = idx_k(i);
152
153                     % Obtain the triangles that contain both
                           idx_i and idx
154                     Domain = LocalElements(triangles,idx_i,idx)
                           ;
155
156                     % Here We generate the stiffness Matrix
157                     [Q11, Q12, Q21, Q22] = DisplacementMatrix(g
                           ,Domain,idx_i,idx,A,C,X,Y);
158
159                     % Log the entries for each matrix.
160                     I11(N+1) = idx_i;
161                     J11(N+1) = idx;
162                     V11(N+1) = Q11;
163
164                     I12(N+1) = idx_i;
165                     J12(N+1) = idx;
166                     V12(N+1) = Q12;
167
168                     I21(N+1) = idx_i;
169                     J21(N+1) = idx;
170                     V21(N+1) = Q21;
171
172                     I22(N+1) = idx_i;
173                     J22(N+1) = idx;
174                     V22(N+1) = Q22;
175
176                     N = N+1;
177                 end
178             end
179
180     % If it is not a boundary node then it must clearly be an
          internal
```

```
181     % node.
182     else
183         % Here we apply the body forces applied to the RHS of
              the system.
184         B = BodyForces(R,BF,X,Y,triangles,idx_i);
185
186         RHS1(idx_i) = B(1);
187         RHS2(idx_i) = B(2);
188         for i = 1:length(idx_k)
189             idx = idx_k(i);
190
191             % Obtain the triangles that contain both idx_i and
                  idx
192             Domain = LocalElements(triangles,idx_i,idx);
193
194             % Here We generate the stiffness Matrix
195             [Q11, Q12, Q21, Q22] = DisplacementMatrix(g,Domain,
                  idx_i,idx,A,C,X,Y);
196
197             % Log the correct indice values for each matrix.
198             I11(N+1) = idx_i;
199             J11(N+1) = idx;
200             V11(N+1) = Q11;
201
202             I12(N+1) = idx_i;
203             J12(N+1) = idx;
204             V12(N+1) = Q12;
205
206             I21(N+1) = idx_i;
207             J21(N+1) = idx;
208             V21(N+1) = Q21;
209
210             I22(N+1) = idx_i;
211             J22(N+1) = idx;
212             V22(N+1) = Q22;
213
214             N = N+1;
215         end
```

```
216     end
217     end
218 end
219
220 % Delete all empty entries;
221 [tt,~] = find(V11 == 0);
222 I11(tt) = [];
223 J11(tt) = [];
224 V11(tt) = [];
225
226 [tt,~] = find(V12 == 0);
227 I12(tt) = [];
228 J12(tt) = [];
229 V12(tt) = [];
230
231 [tt,~] = find(V21 == 0);
232 I21(tt) = [];
233 J21(tt) = [];
234 V21(tt) = [];
235
236 [tt,~] = find(V22 == 0);
237 I22(tt) = [];
238 J22(tt) = [];
239 V22(tt) = [];
240
241 % Create the correct matrices.
242 K11 = sparse(I11,J11,V11,Nodes,Nodes);
243 K12 = sparse(I12,J12,V12,Nodes,Nodes);
244 K21 = sparse(I21,J21,V21,Nodes,Nodes);
245 K22 = sparse(I22,J22,V22,Nodes,Nodes);
246
247 % Here we create the stiffness Matrix, exploit symmetry to
      lessen calculation.
248 K = [K11 K12; K21 K22];
249
250 % Create our RHS of the linear system.
251 RHS = [RHS1; RHS2];
252
```

```
% Solve the linear system.
Solution = K\RHS;

% Solution Matrix Space.
U = Solution(1:Nodes);
V = Solution(Nodes+1:end);

end
```

Listing B.14: This Matlab functions calculates all the boundary nomal unit vectors. To do this we input all the boundary nodes, then loop through every single boundary node. We can find both the neighbouring nodes, using this geometric information we can take the average of the normalised path vectors through this neighbourhood of nodes. Performing an anticlockwise rotation of this normal vector allows us to find, a perpendicular, unit normal vector at each boundary node.

```
function NormalVector = BoundaryNorms(Boundary,Facets,X,Y)

% Number of nodes.
N = length(Boundary);

% Our Norm Matrix, column 1 is the x component and column 2 is
% the y component.
Norms = zeros(N,2);

% Generate ID Vector.
ID = Boundary;

for i = 1:N
    [rows,~] = find(Facets == Boundary(i));
    A = Facets(rows(1),:);
    B = Facets(rows(2),:);

    % Calculate each FacetNorm.
    FacetNorm1 = ([X(A(2)) Y(A(2))] - [X(A(1)) Y(A(1))])'/norm
       ([X(A(2)) Y(A(2))] - [X(A(1)) Y(A(1))]);
    FacetNorm2 = ([X(B(2)) Y(B(2))] - [X(B(1)) Y(B(1))])'/norm
       ([X(B(2)) Y(B(2))] - [X(B(1)) Y(B(1))]);

```

```
22      % Take the average and Rotate.
23      Norms(i,:) = 0.5*[0 1;-1 0]*(FacetNorm1 + FacetNorm2);
24  end
25
26  % Now we combine the norms with the IDs.
27  NormalVector = [Norms ID];
28
29  end
```

Listing B.15: This Matlab function calculates all the commonly shared triangles two neighbouring nodes share. To do this all we require is the connectivity matrix and the node index for each node. This function is used many times throughout the code.

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %
3  % This function finds the triangles that contain both nodes,
       essentially
4  % finding the domain we will define our integrals and basis
       functions over.
5  %
6  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7
8  function [Domain] = LocalElements(triangles,idx_i,idx)
9
10 [rows1, ~] = find(triangles == idx);
11 Domain = triangles(sort(rows1),:);
12 [rows2, ~] = find(Domain == idx_i);
13 Domain = Domain(sort(rows2),:);
14
15 end
```

Listing B.16: This Matlab function calculates the integral coefficients for each of the elements in the mass matrix. We calculate each integral over a single triangle and then sum the solution up over the fintie element.

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %
3  % Here we calculate the value at K(i,j) for the displacement
       stiffnes
4  % matrix. To do this we require the local elements, index i,j
```

```
   and the
% geometry X,Y,
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [K11,K12,K21,K22] = DisplacementMatrix(g,Domain,idx_i,
   idx,A,C,X,Y)

% Number of elements we have to calculate over.
loop = size(Domain,1);

% Let K = 0, to reset the memory.
K11 = 0;
K12 = 0;
K21 = 0;
K22 = 0;

% Loop to calculate
for i = 1:loop

    Triangle = Domain(i,:);

    % Calculate each integral over
    [I11,I12,I21,I22] = ElementIntegralDisplacement(g,Triangle,
       idx_i,idx,A,C,X,Y);

    % Calculate the integral over every triangle.
    K11 = I11 + K11;
    K12 = I12 + K12;
    K21 = I21 + K21;
    K22 = I22 + K22;
end

end
```

Listing B.17: This matlab function calculates integrals over each triangle in our system using the Newtons-Cotes method. All this requires is that over our triangle we sum up the product of the elasticity elements and the gradient of the piecewise shape functions. Then take the average by dividing by three the number of vertices on the triangles and multiplying by the area of said triangle.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Calculate the integrals over each individual element for the
% displacement.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [I11,I12,I21,I22] = ElementIntegralDisplacement(g,
   Triangle,idx_i,idx,A,C,X,Y)

% Define the vertices of the triangle.
x1 = [X(Triangle(1)) Y(Triangle(1))];
x2 = [X(Triangle(2)) Y(Triangle(2))];
x3 = [X(Triangle(3)) Y(Triangle(3))];

% Calculate the area over the element.
TriangleArea = 0.5*det([x1 1; x2 1; x3 1]);

% Calculate the Basis functions for idx and idx_i.
Grad_i = GradPhi(Triangle,idx_i,X,Y);
Grad_j = GradPhi(Triangle,idx,X,Y);

% Setup sums.
C11 = 0;
C12 = 0;
C13 = 0;
C22 = 0;
C23 = 0;
C33 = 0;

% Sum of Coefficients at the vertices.
for i = 1:length(Triangle)
    % Calculate the elastic tensor.
    E = C(X(Triangle(i)),Y(Triangle(i)));
```

```
33
34      % Calculate the sum of each component.
35      C11 = C11 + E(1,1)*g(A(Triangle(i)));
36      C12 = C12 + E(1,2)*g(A(Triangle(i)));
37      C13 = C13 + E(1,3)*g(A(Triangle(i)));
38      C22 = C22 + E(2,2)*g(A(Triangle(i)));
39      C23 = C23 + E(2,3)*g(A(Triangle(i)));
40      C33 = C33 + E(3,3)*g(A(Triangle(i)));
41  end
42
43  % Calculate the integral volume, newton-cotes method.
44  I11 = (1/3)*(C11*Grad_i(1)*Grad_j(1) + C13*Grad_i(1)*Grad_j(2)
        ...
45      + C13*Grad_i(2)*Grad_j(1) + C33*Grad_i(2)*Grad_j(2))*
            TriangleArea;
46
47  I12 = (1/3)*(C13*Grad_i(1)*Grad_j(1) + C12*Grad_i(1)*Grad_j(2)
        ...
48      + C33*Grad_i(2)*Grad_j(1) + C23*Grad_i(2)*Grad_j(2))*
            TriangleArea;
49
50  I21 = (1/3)*(C13*Grad_i(1)*Grad_j(1) + C33*Grad_i(1)*Grad_j(2)
        ...
51      + C12*Grad_i(2)*Grad_j(1) + C23*Grad_i(2)*Grad_j(2))*
            TriangleArea;
52
53  I22 = (1/3)*(C33*Grad_i(1)*Grad_j(1) + C23*Grad_i(1)*Grad_j(2)
        ...
54      + C23*Grad_i(2)*Grad_j(1) + C22*Grad_i(2)*Grad_j(2))*
            TriangleArea;
55
56  end
```

Listing B.18: This Matlab function calculates the gradient of each piecewise shape function. To do this we take the triangle we are interested in with the vertex we want to calculate the derivative at. Then by using the positions of nodes we can directly calculate the derivate of the basis function.

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%
% Calculates the Gradient of our basis function at each point. x1 is where
% the basis equals 1 and 0 at x2 and x3. This function is linear. We solve
% this by using a transformation from a reference triangle.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Grad_i = GradPhi(Triangle,idx_i,X,Y)

% This is a really annoying algorithim we set up to designate x1, x2 and
% x3.
loop = length(Triangle);

x2NotExists = true;
x3NotExists = true;

for i = 1:loop
    if Triangle(i) == idx_i
        x1 = [X(Triangle(i)) Y(Triangle(i))];
    elseif (x2NotExists)
        x2 = [X(Triangle(i)) Y(Triangle(i))];
        x2NotExists = false;
    elseif (x3NotExists)
        x3 = [X(Triangle(i)) Y(Triangle(i))];
        x3NotExists = false;
    end
end

% Now we do some very simple linear algebra to find the gradient of phi.
% At is a transposed matrix.
At = [x2-x1; x3-x1];

Grad_i = -At\[1;1];

end
```

Listing B.19: This Matlab function calculates the contribution of surface forces for external loading as defined in (5.32) at each boundary node given the surface stress tensor, node position and shape of the local boundary. To do this we use our boundary condition, the normal vector and the length of the path integral to perform a Newton-Cotes approximation of this integral.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This function calculates the contribution of surface forces.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function S = SurfaceForces(NormalVector,Boundary,BC,X,Y,idx_i,
   idx_k)

% Calculate the unit Norm and traction vector.
Norm = NormalVector(NormalVector(:,3) == idx_i,1:2);
P = (BC(X(idx_i),Y(idx_i))*Norm')';

% Neighbouring Nodes
Neighbours = intersect(idx_k,Boundary);
idx_remove = Neighbours == idx_i;
Neighbours(idx_remove) = [];

% Find position of nodes making up line element.
x1 = [X(idx_i) Y(idx_i)];
x2 = [X(Neighbours(1)) Y(Neighbours(1))];
x3 = [X(Neighbours(2)) Y(Neighbours(2))];

% Calculate domain length.
ds = norm(x1-x2) + norm(x3-x1);

% Calculate surface force.
S = P*ds/3;

end
```

Listing B.20: This Matlab function calculates the contribution of body forces to external loading as defined in (5.32). We can calculate this contribution with a very simple Newton-Cotes approximation. By multiplying the area of the finite element by the force and density of the node at the centre then taking an average we find our approximation.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This function calculates the contribution of the body forces.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function B = BodyForces(R,BF,X,Y,triangles,idx_i)

% First we calculate the total area of the element, by summing up the area
% of all the triangles which form it.
[tt,~] = find(triangles == idx_i);
TriangleArea = 0;

for i = 1:length(tt)
    % Define a triangle.

    Triangle = triangles(tt(i),:);
    % Define the vertices of the triangle.
    x1 = [X(Triangle(1)) Y(Triangle(1))];
    x2 = [X(Triangle(2)) Y(Triangle(2))];
    x3 = [X(Triangle(3)) Y(Triangle(3))];

    % Calculate the area over the element.
    TriangleArea = TriangleArea + 0.5*det([x1 1; x2 1; x3 1]);
end

% Finally we define the Body force.
B = (1/3)*TriangleArea*R(X(idx_i),Y(idx_i))*BF(X(idx_i),Y(idx_i));

end
```

B.6 Damage criterion

In Chapter 4 and 5 we introduced the idea of creating a "Crit" function which is our criterion field. In §5.3.5 we explained that the criterion field tracks all the points in the reference domain for the current iterative step which do not satisfy the damage criterion. As defined by (5.35) at every point where $\text{Crit}(\mathbf{X}) = 1$ we have not satisfied the damage criterion and expect damage evolution to take place. Whereas, at every point $\text{Crit}(\mathbf{X}) = 0$ we have that no damage evolution is taking place. However, to ensure that the damage phase field can diffuse properly throughout the reference domain we also define the points where $\text{Crit}(\mathbf{X}) = 2$ which are all the nodes neighbouring nodes that are experiencing damage evolution. To do this we use our Matlab function "CriterionCheck" which can be seen in Listing B.21. The overall role of the criterion field is to inform our numerical scheme for calculating the damage phase field.

To calculate the criterion field we must begin by evaluating the damage criterion (5.4) at every node in our mesh. This is done with our Matlab function "DamageEquation" which directly calculates the damage criterion (5.4), this can be seen in Listing B.22. This calculation is performed very directly, the first thing we do is calculate the driving force term which accounts for the energy per unit volume driving damage growth. We calculate this using the Matlab function "Driving force" which can be seen in Listing B.23. This requires calculating the gradients of strain and then performing the double dot product against the elasticity tensor twice. We can use the gradients of displacement to calculate the Jacobian at every point in our mesh.

As mentioned we calculate gradients whilst calculating the Driving force in our damage criterion, we also need to calculate the flux of the damage phase field throughout our reference domain. Calculating gradients of any variable at any point in a triangular mesh is not as simple as finite differences on a uniform mesh. We discussed in §5.3.5 how we can use a two dimensional triangular mesh to calculate the triangular derivative and then perform a transformation to find the regular derivatives. To do these calculations we make use of the Matlab function "TriGradient" as can be seen in Listing B.24. When calculating the directional derivatives we use first order finite difference methods.

Listing B.21: This Matlab function calculates the Criterion field (5.35) at every point in our mesh. This is done by calculating the damage criterion (5.4) at each point using the code in Listing B.22. Then we assign Crit = 1 at every point where the damage criterion is not satisfied and the Jacobian is equal to or greater than one. We automatically assign every other point to be Crit = 0. Then we assign every node that is Crit = 0 but close to any point Crit = 1 to be Crit = 2 so that we know this point must account for diffusion of the damage phase field.

```
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%
% Check which points are in the damage phase and need
   recalculated.
% Criterion == 0 means no damage growth.
% Criterion == 1 means damage growth.
% Criterion == 2 means damage diffusion.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Crit = CriterionCheck(g1,A,U,V,Psi,C,G,D,triangles,X,Y
   ,TornNodes)

% Calculate the damage equation to test the criterion.
[F,J] = DamageEquation(g1,A,U,V,Psi,C,G,D,triangles,X,Y);

% Finds the exact points where the equation is not satisfied.
Crit =  (0*(F<0) + 1*(F>=0)).*(1*(TornNodes ~= 1) + 0*(
   TornNodes == 1)).*(0*(J<1) + 1*(J>=1));

% Find all the 1st and 2nd neighbours of each Crit point.
if sum(Crit) > 0

    % Find all the critical points.
    Q = find(Crit == 1);

    % Determine all neighbouring points in the diffusion scale.
    % That are not a critical point.
    for i = 1:length(Q)
            k = Q(i);
        for j = 1:length(X)
            if Crit(j) == 0
                if norm([X(k) Y(k)] - [X(j) Y(j)]) <= 100*max(
                   max(D(X(k),Y(k))))
                    Crit(j) = 2;
                end
            elseif Crit(j) == 1
                Crit(j) = 1;
            end
        end
```

```
36      end
37  end
38
39  % Remove all compressed points.
40  Crit = Crit.*(1*(TornNodes ~= 1) + 0*(TornNodes == 1)).*(0*(J
        <1) + 1*(J>=1));
41
42  end
```

Listing B.22: This Matlab function calculates the damage criterion (5.4) at every point in the mesh. It should be noted this Listing is used multiple times in the code as it represents the right hand side of the damage evolution equation (5.3b).

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %
3  % Calculate the RHS of the damage equation.
4  %
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6  function [F,J] = DamageEquation(g1,A,U,V,Psi,C,G,D,triangles,X,
       Y)
7
8  % Total Number of Nodes.
9  Nodes = numel(X);
10
11 % Here we calculate the driving force of the damage.
12 [Q,J] = DrivingForce(U,V,Psi,C,triangles,X,Y);
13
14 % Empty Matrices that will hold the X and Y components of D*
       Grad(a) on each
15 % page respectively.
16 P1 = zeros(Nodes,1);
17 P2 = zeros(Nodes,1);
18 H = zeros(Nodes,1);
19 K = zeros(Nodes,1);
20
21 % First Derivative of Alpha.
22 [ax, ay] = TriGradient(A,triangles,X,Y);
23
24 for i = 1:Nodes
```

```
    Diffusion = D(X(i),Y(i));
    P1(i) = Diffusion(1,:)*[ax(i) ; ay(i)];
    P2(i) = Diffusion(2,:)*[ax(i) ; ay(i)];
    H(i) = g1(A(i));
    K(i) = G(X(i),Y(i));
end

% These are the X and Y components of the divergence term.
[DivergenceX,~] = TriGradient(P1,triangles,X,Y);
[~,DivergenceY] = TriGradient(P2,triangles,X,Y);

% Damage Equation.
F = -H.*Q - K.*A + DivergenceX + DivergenceY;

end
```

Listing B.23: This Matlab function calculates the driving force behind the damage evolution in our material. To do this we need to calculate the linear elastic energy potentil minus the threshold at every point. This requires calculating the displacement gradients at every point which we can use to caulcate the Jacobian at every point.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Calculate the linear elastic energy minus the threshold i.e.
   the driving
% force.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Q,J] = DrivingForce(U,V,Psi,C,triangles,X,Y)

% Reset Q the driving force
Q = zeros(numel(X),1);

% Reset J the determinant of the Gradient function.
J = zeros(numel(X),1);

% Symetric Gradient Matrix.
[ux, uy] = TriGradient(U,triangles,X,Y);
[vx, vy] = TriGradient(V,triangles,X,Y);
```

```
18
19 for i = 1:numel(X)
20     % Infintisemal Strain
21     Chi = [ux(i) 0.5*(uy(i)  + vx(i)); 0.5*(uy(i) + vx(i)) vy(i
           )];
22
23     % Elastic Tensor.
24     E = C(X(i),Y(i));
25
26     Q(i) = 0.5*(E(1,1)*Chi(1,1)^2 + 4*E(3,3)*Chi(1,2)^2 + E
           (2,2)*Chi(2,2)^2)...
27          + Chi(1,1)*E(1,2)*Chi(2,2) + 2*Chi(1,2)*E(2,3)*Chi(2,2)
                + 2*Chi(1,1)*E(1,3)*Chi(1,2)...
28          - Psi(X(i),Y(i));
29
30     % Displacement matrix.
31     A = [1+ux(i), uy(i); vx(i), 1+vy(i)];
32
33     % Jacobian.
34     J(i) = det(A);
35 end
36
37 end
```

Listing B.24: This Matlab function is used to calculate the horizontal and vertical gradients at each node of our triangular mesh. To do this at each node we use all the triangles that node is contained within. We can calculate for each triangle a pair of directional derivatives connecting the other vertices to our node. Then we use a simple transformation (5.37) to generate the horizontal and vertical derivatives at the node. We take the average of all of the results from each triangle.

```
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %
3 % Calculate the gradient at the vertices of triangular elements
      in our mesh
4 % system.
5 %
6 % To do this we must calculate the triangular derivative from
     each triangle
```

```
% in the mesh and then assign the average to that point.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Fx, Fy] = TriGradient(F,triangles,X,Y)

% Number of nodes that exist is the size of Fx and Fy
   respectively.
Nodes = numel(F);

Fx = zeros(Nodes,1);
Fy = zeros(Nodes,1);

    % We need to go Node by node so we do this with a loop.
    for i = 1:Nodes

        % Firstly we identify a triangular element associated
           with the node.
        [tt,~] = find(triangles == i);

        % Filter between points that exist and are destroyed.
        if isempty(tt)
            continue
        else
                % Reset derivative sums.
                Dx = 0;
                Dy = 0;

            for k = 1:length(tt)
                % Current triangle.
                Triangle = triangles(tt(k),:);

                % Short algorithim to label the vertices of the
                    triangle.
                x1NotExists = true;
                x2NotExists = true;

                for j = 1:length(Triangle)
                    if Triangle(j) == i
```

```
42                          x0 = [X(Triangle(j)), Y(Triangle(j))];
43                          f0 = F(Triangle(j));
44                      elseif (x1NotExists)
45                          x1 = [X(Triangle(j)), Y(Triangle(j))];
46                          f1 = F(Triangle(j));
47                          x1NotExists = false;
48                      elseif (x2NotExists)
49                          x2 = [X(Triangle(j)), Y(Triangle(j))];
50                          f2 = F(Triangle(j));
51                          x2NotExists = false;
52                      end
53                  end
54
55                  % Now we create a linear system relating the
                       directional derivative to the euclidean
                       derivative.
56                  B = [(x1 - x0); (x2 - x0)];
57
58                  % The right hand side represents the
                       directional derivative.
59                  C = [(f1 - f0); (f2 - f0)];
60
61                  % Now we can calculate the euclidean
                       derivatives.
62                  Derivatives = B\C;
63
64                  % Then we seperate the x and y derivatives into
                        our solutions.
65                  Dx = Dx + Derivatives(1);
66                  Dy = Dy + Derivatives(2);
67              end
68              % Take average of each triangle to calculate the
                   derivative.
69              Fx(i) = Dx/length(tt);
70              Fy(i) = Dy/length(tt);
71          end
72      end
73  end
```

B.7 Damage phase

In §5.3.6 we explained how to calculate the damage phase field using the finite element method. We implemented this method of calculating the damage phase field in code, which can be seen in Listing B.25. The code is essentially implementing an iterative method for solving a Crank-Nicholson Galerkin method. To make the iterative method simpler we are assuming here that the displacement is fixed in place and we are only solving for the damage phase field. In §5.3.6 we outline the iterative scheme that we must calculate. To do this we need to construct a number of matrices (5.60) and vectors (5.55) before we can calculate a new solution for α. Using this solution to update our iterative method we can calculate another solution, and repeat the process until our solution for α converges to a desired tolerance. Importantly, this is done whilst using the criterion field and the torn nodes field to inform us on how each node is calculated.

Constructing all the necessary matrices (5.60) can be difficult. So we devised three Matlab functions to do the job for us. The first of these is the function "DamageMatrix" which can be found in Listing B.29. We use this function to calculate the matrix (5.60a) in our iterative scheme, which essentially forms the mass matrix of (5.59). The next function is "RightHandSide2" which can be found in Listing B.26. This function is used to calculate the matrices (5.60b) and (5.60c) on the right hand side of our linear system (5.59). We also needed to create the function "RightHandSide" to calculate (5.60d), the final matrix, on the right hand side of (5.59). The code for this can be found in Listing B.28.

All of the matrices in the linear system (5.59) have integral coefficients for each of their elements. So we created the Matlab function "InnerProduct" in Listing B.27. This function calculates an integral using the newtons cotes method the accuracy of which increases as the triangular elements get smaller. Notice that we call on this function in Listings (5.60b), (5.60c) and (5.60d).

For the left hand side of the linear system (5.59) we use two custom functions. The first is "LeftHandSide" which can be found in Listing B.30. The second is "LeftHandSide2" which can be found in Listing B.31. Both of which use Newtons-Cotes method to calculate the integral coefficients for each element in the matrix (5.60a). We use the function "LeftHandSide" for all the points where $\mathrm{Crit}(\mathbf{X}) = 1$ and the function "LeftHandSide2" for all the points $\mathrm{Crit}(\mathbf{X}) = 2$. Both of these functions also use the Newtons-Cotes method to calculate the necessary integrals. At all the points where the we have $\mathrm{Torn}(\mathbf{X}) = 1$ or $\mathrm{Crit}(\mathbf{X}) = 0$ we are sure to set up our linear system such that no damage evolution takes place.

Listing B.25: This Matlab function takes the current solution for the displacement and all the other relevant model parameters to calculate the solution for the damage phase field at each time step. To do this we implement the iterative shceme laid out in §5.3.6. The iterative scheme solves a non-linear Crank-Nicholson style problem, using the finite element method.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Here we use an iterartive method to solve a non-linear crank nicholson
% Galerkin Method. Refer to Chapter 5 of my thesis.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [Alpha, stuck] = DamagePhase(g1,A,U,V,Psi,C,G,D,eta,Crit,triangles,X,Y,step,tol,TornNodes)

% Tolerance for newtons method and constant.
tol = min(max(tol*1e-2,1e-12),1e-6);
r = zeros([size(X)]);
N = 0;
stuck = false;

% Generate r = 2*eta/timestep.
for j = 1:numel(X)
    r(j) = (2*eta(X(j),Y(j)))/step;
end

% Here we are iteratively going to calculate A(:,2) to do this we represent
% Y_curr and Y_old as the iterations of A(:,2).
Y_curr = A(:,2);
Y_old = -ones([size(A(:,2))]);

% Calculate the damage equation approximation for the previous time step.
[F,~] = DamageEquation(g1,A(:,1),U(:,1),V(:,1),Psi,C,G,D,triangles,X,Y);
F1 = RightHandSide2(ones(length(F),1),ones(length(F),1),triangles,X,Y);
```

```
28
29 % Next term on the right hand side.
30 F2 = RightHandSide(r,Crit,triangles,X,Y);
31
32 % Last right hand side term dependent on the most recent
      displacement.
33 Q = DrivingForce(U(:,2),V(:,2),Psi,C,triangles,X,Y);
34 F3 = RightHandSide2(Q.*(Crit == 1),Crit,triangles,X,Y);
35
36 % Create the Damage Matrix this never changes.
37 J = DamageMatrix(r,G,D,Crit,triangles,X,Y);
38
39 % Limit for ensuring that max alpha is equal to or below one.
40 Aold = A(:,1);
41 f1 = F1;
42 f2 = F2;
43 f3 = F3;
44 j = J;
45
46 % Restate the problem for points where alpha = 1.
47 if any(TornNodes)
48     % First delete all the relevant rows.
49     [M,~] = find(TornNodes);
50     f1(M,:) = 0;
51     f2(M,:) = 0;
52     f3(M,:) = 0;
53     j(M,:) = 0;
54     Aold(M) = 1;
55
56     % Next change the indicies.
57     K = sub2ind(size(j),M,M);
58     j(K) = 1;
59     f2(K) = 1;
60 end
61
62 while norm(Y_curr - Y_old,Inf) > tol
63     % Convergence problem.
64     if N > 100
```

```
        stuck = true;
        Alpha = A(:,1);
        return
    end

    % Update iteration.
    Y_old = Y_curr;

    % Create the terms dependent on the iteration.
    g = DegradationApproximation(g1,Y_old);

    % Update our solution.
    Y_curr = j\(f1*max(F,0) + f2*Aold - f3*g);

    % Irreversibility.
    Y_curr = min(max(Y_curr,Aold),1);

    N = N + 1;
end

% Final solution.
Alpha = Y_curr;

end
```

Listing B.26: This Matlab function is used to calculate the elements of matrices (5.60b) and (5.60c). We defined these matrices in §5.3.6 they are used to formulate our iterative scheme (5.59). All the elements in the matrices are integrals and we calculate them using the Newtons-Cotes method. However, we only do this for nodes that have $\text{Crit}(\mathbf{X}) = 1$ otherwsie we set up our matrices such that no damage evolution takes place.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Here we create the matrix on the right hand side of the
   damage phase for
% iteration method RHS = integral of ( Function X Phi_i X Phi_j
   ).
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function RHS = RightHandSide2(Z,Crit,triangles,X,Y)

% Number of Nodes in the system.
Nodes = numel(X);

% Setup a counter for I,J,V which are the rows, cols and values for their
% respective sparse matrices.
I = zeros(10*Nodes,1);
J = zeros(10*Nodes,1);
V = zeros(10*Nodes,1);

% Here N just progresses the rows they are on.
N = 0;

% Loop to calculate
for i = 1:Nodes
    idx_i = i;
    if Crit(idx_i) == 1
        % Find all the triangles containing this node.
        [tt,~] = find(triangles == idx_i);

        % Now we find all the adjacent nodes connected by the triangles.
        idx_k = unique(triangles(tt,:));
        for j = 1:length(idx_k)
            % j node.
            idx_j = idx_k(j);

            % Obtain the triangles that contain both idx_i and idx
            Domain = LocalElements(triangles,idx_i,idx_j);

            % Calculate the corresponding point in the damage matrix at
            % i,j.
            Q = InnerProduct(Domain,idx_i,idx_j,Z,X,Y);

```

```
            % Setup Sparse Matrix.
            I(N+1) = idx_i;
            J(N+1) = idx_j;
            V(N+1) = Q;

            % Update the counter.
            N = N + 1;
        end
    end
end

% Delete all empty entries.
[tt,~] = find(V == 0);
I(tt) = [];
J(tt) = [];
V(tt) = [];

% Create our sparse system.
RHS= sparse(I,J,V,Nodes,Nodes);

end
```

Listing B.27: This Matlab function is used to calculate the integrals that formulate the matrices being calculated in Listing B.26. We do this using a Newtons-Cotes method which requires us to calculate the area of each triangular element and take the average of each node contribution. Summing up the contribution from each triangle in the finite element gives us our full approximate solution.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Here we calculate the value at Q(i,j) for the damage matrix
   on the Left
% hand side for our iterative damage method.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Q = InnerProduct(Domain,idx_i,idx_j,Z,X,Y)

% Number of elements we have to calculate over.
loop = size(Domain,1);
```

```
11
12 % Let Q = 0, to reset the memory.
13 Q = 0;
14
15 % Loop to calculate
16 for i = 1:loop
17
18     Triangle = Domain(i,:);
19
20     % Define the vertices of the triangle.
21     x1 = [X(Triangle(1)) Y(Triangle(1))];
22     x2 = [X(Triangle(2)) Y(Triangle(2))];
23     x3 = [X(Triangle(3)) Y(Triangle(3))];
24
25     % Calculate the area over the element.
26     TriangleArea = 0.5*det([ x1 1; x2 1; x3 1]);
27
28     % Centroid of the element.
29     x = (x1 + x2 + x3)/3;
30
31     % Calculate the Basis functions for idx and idx_i.
32     Phi_i = Phi(Triangle,idx_i,X,Y,x);
33     Phi_j = Phi(Triangle,idx_j,X,Y,x);
34
35     % Averages.
36     F = sum(Z(Triangle))/3;
37
38     % Calculate the integral volume, newton-cotes method.
39     I = F*Phi_i*Phi_j*TriangleArea;
40
41     % Calculate the integral over every triangle.
42     Q = I + Q;
43 end
44
45 end
```

Listing B.28: This Matlab function is used to calculate the elements of matrix (B.29). We defined this matrix in §5.3.6 and it is used to formulate our iterative scheme (5.59). All the elements in the matrix are integrals and we calculate them using the Newtons-Cotes method. However, we only do this for nodes that have Crit($\mathbf{X}$) = 1 otherwsie we set up our matrix such that no damage evolution takes place.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Here we create the matrix on the right hand side of the
   damage phase for
% iteration method RHS = integral of ( Function X Phi_i X Phi_j
   ).
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function RHS = RightHandSide(Z,Crit,triangles,X,Y)

% Number of Nodes in the system.
Nodes = numel(X);

% Setup a counter for I,J,V which are the rows, cols and values
    for their
% respective sparse matrices.
I = zeros(10*Nodes,1);
J = zeros(10*Nodes,1);
V = zeros(10*Nodes,1);

% Here N just progresses the rows they are on.
N = 0;

% Loop to calculate
for i = 1:Nodes
    idx_i = i;
    if Crit(idx_i) == 1
        % Find all the triangles containing this node.
        [tt,~] = find(triangles == idx_i);

        % Now we find all the adjacent nodes connected by the
           triangles.
        idx_k = unique(triangles(tt,:));
```

```
30          for j = 1:length(idx_k)
31              % j node.
32              idx_j = idx_k(j);
33
34              % Obtain the triangles that contain both idx_i and
                    idx
35              Domain = LocalElements(triangles,idx_i,idx_j);
36
37              % Calculate the corresponding point in the damage
                    matrix at
38              % i,j.
39              Q = InnerProduct(Domain,idx_i,idx_j,Z,X,Y);
40
41              % Setup Sparse Matrix.
42              I(N+1) = idx_i;
43              J(N+1) = idx_j;
44              V(N+1) = Q;
45
46              % Update the counter.
47              N = N + 1;
48          end
49      elseif Crit(idx_i) == 0
50          % Setup Sparse Matrix.
51          I(N+1) = idx_i;
52          J(N+1) = idx_i;
53          V(N+1) = 1;
54
55          % Update the counter.
56          N = N + 1;
57      end
58  end
59
60  % Delete all empty entries.
61  [tt,~] = find(V == 0);
62  I(tt) = [];
63  J(tt) = [];
64  V(tt) = [];
65
```

```
% Create our sparse system.
RHS= sparse(I,J,V,Nodes,Nodes);

end
```

Listing B.29: This Matlab function is used to calculate the elements of matrix (5.60a). We defined this matrix in §5.3.6 and it is used to formulate our iterative scheme (5.59). All the elements in the matrix are integrals and we calculate them using the Newtons-Cotes method. However, we only do this for nodes that have $\text{Crit}(\mathbf{X}) = 1$ otherwsie we set up our matrix such that no damage evolution takes place. Unless we have $\text{Crit}(\mathbf{X}) = 2$ then we set up the matrix so it accounts for damage diffusion.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Here we calculate the value of J(i,j) the damage matrix for
   the right
% hand side at the nodes (idx_i,idx_j).
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function DamageMatrix = DamageMatrix(r,G,D,Crit,triangles,X,Y)

% Number of Nodes in the system.
Nodes = size(X,1)*size(X,2);

% Setup a counter for I,J,V which are the rows, cols and values
    for their
% respective sparse matrices.
I = zeros(10*Nodes,1);
J = zeros(10*Nodes,1);
V = zeros(10*Nodes,1);

% Here N just progresses the rows they are on.
N = 0;

% Loop to calculate
for i = 1:Nodes
    idx_i = i;
    if Crit(idx_i) == 1
        % Find all the triangles containing this node.
```

```
26          [tt,~] = find(triangles == idx_i);
27
28          % Now we find all the adjacent nodes connected by the
               triangles.
29          idx_k = unique(triangles(tt,:));
30          for j = 1:length(idx_k)
31              % j node.
32              idx_j = idx_k(j);
33
34              % Obtain the triangles that contain both idx_i and
                   idx
35              Domain = LocalElements(triangles,idx_i,idx_j);
36
37              % Calculate the corresponding point in the damage
                   matrix at
38              % i,j.
39              Q = LeftHandSide(Domain,idx_i,idx_j,r,G,D,X,Y);
40
41              % Setup Sparse Matrix.
42              I(N+1) = idx_i;
43              J(N+1) = idx_j;
44              V(N+1) = Q;
45
46              % Update the counter.
47              N = N + 1;
48          end
49      elseif Crit(idx_i) == 2
50          % Find all the triangles containing this node.
51          [tt,~] = find(triangles == idx_i);
52
53          % Now we find all the adjacent nodes connected by the
               triangles.
54          idx_k = unique(triangles(tt,:));
55          for j = 1:length(idx_k)
56              % j node.
57              idx_j = idx_k(j);
58
59              % Obtain the triangles that contain both idx_i and
```

```
            idx
            Domain = LocalElements(triangles,idx_i,idx_j);

            % Calculate the corresponding point in the damage
              matrix at
            % i,j.
            Q = LeftHandSide2(Domain,idx_i,idx_j,G,D,X,Y);

            % Setup Sparse Matrix.
            I(N+1) = idx_i;
            J(N+1) = idx_j;
            V(N+1) = Q;

            % Update the counter.
            N = N + 1;
        end
    elseif Crit(idx_i) == 0
        % Setup Sparse Matrix.
        I(N+1) = idx_i;
        J(N+1) = idx_i;
        V(N+1) = 1;

        % Update the counter.
        N = N + 1;
    end
end

% Delete all empty entries.
[tt,~] = find(V == 0);
I(tt) = [];
J(tt) = [];
V(tt) = [];

% Create our sparse system.
DamageMatrix = sparse(I,J,V,Nodes,Nodes);

end
```

Listing B.30: This Matlab function is used to calculate the integrals that formulate the matrices being calculated in Listing B.29. Here we account for the contributions of damage diffusion, the tearing energy and therate of change of damage. We do this using a Newtons-Cotes method which requires us to calculate the area of each triangular element and take the average of each node contribution. Summing up the contribution from each triangle in the finite element gives us our full approximate solution.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Here we calculate the value at Q(i,j) for the damage matrix
   on the Left
% hand side for our iterative damage method.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Q = LeftHandSide(Domain,idx_i,idx_j,r,G,D,X,Y)

% Number of elements we have to calculate over.
loop = size(Domain,1);

% Let Q = 0, to reset the memory.
Q = 0;

% Loop to calculate
for i = 1:loop

    Triangle = Domain(i,:);

    % Define the vertices of the triangle.
    x1 = [X(Triangle(1)) Y(Triangle(1))];
    x2 = [X(Triangle(2)) Y(Triangle(2))];
    x3 = [X(Triangle(3)) Y(Triangle(3))];

    % Calculate the area over the element.
    TriangleArea = 0.5*det([ x1 1; x2 1; x3 1]);

    % Centroid of the element.
    x = (x1 + x2 + x3)/3;

    % Calculate the Basis functions for idx and idx_i.
```

```
32      Phi_i = Phi(Triangle,idx_i,X,Y,x);
33      Phi_j = Phi(Triangle,idx_j,X,Y,x);
34
35      % Calculate the Basis functions for idx and idx_i.
36      Grad_i = GradPhi(Triangle,idx_i,X,Y);
37      Grad_j = GradPhi(Triangle,idx_j,X,Y);
38
39      % Averages.
40      R = sum(r(Triangle));
41      g = 0;
42      DiffusionTerm = 0;
43      for j = 1:length(Triangle)
44          g = g + G(X(Triangle(j)),Y(Triangle(j)));
45          DiffusionTerm = DiffusionTerm + dot(D(X(Triangle(j)),Y(
                Triangle(j)))*Grad_j,Grad_i);
46      end
47
48      % Calculate the integral volume, newton-cotes method.
49      I1 = (1/3)*(R + g)*Phi_i*Phi_j*TriangleArea;
50
51      I2 = (1/3)*DiffusionTerm*TriangleArea;
52
53      I = I1 + I2;
54
55      % Calculate the integral over every triangle.
56      Q = I + Q;
57  end
58
59  end
```

Listing B.31: This Matlab function is used to calculate the integrals that formulate the matrices being calculated in Listing B.29. Here we account for the contributions of damage diffusion and the tearing energy. We do this using a Newtons-Cotes method which requires us to calculate the area of each triangular element and take the average of each node contribution. Summing up the contribution from each triangle in the finite element gives us our full approximate solution.

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %
```

```
% Here we calculate the value at Q(i,j) for the damage matrix
   on the Left
% hand side for our iterative damage method.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Q = LeftHandSide2(Domain,idx_i,idx_j,G,D,X,Y)

% Number of elements we have to calculate over.
loop = size(Domain,1);

% Let Q = 0, to reset the memory.
Q = 0;

% Loop to calculate
for i = 1:loop

    Triangle = Domain(i,:);

    % Define the vertices of the triangle.
    x1 = [X(Triangle(1)) Y(Triangle(1))];
    x2 = [X(Triangle(2)) Y(Triangle(2))];
    x3 = [X(Triangle(3)) Y(Triangle(3))];

    % Calculate the area over the element.
    TriangleArea = 0.5*det([ x1 1; x2 1; x3 1]);

    % Centroid of the element.
    x = (x1 + x2 + x3)/3;

    % Calculate the Basis functions for idx and idx_i.
    Phi_i = Phi(Triangle,idx_i,X,Y,x);
    Phi_j = Phi(Triangle,idx_j,X,Y,x);

    % Calculate the Basis functions for idx and idx_i.
    Grad_i = GradPhi(Triangle,idx_i,X,Y);
    Grad_j = GradPhi(Triangle,idx_j,X,Y);

    % Averages.
```

```
40      g = 0;
41      DiffusionTerm = 0;
42      for j = 1:length(Triangle)
43          g = g + G(X(Triangle(j)),Y(Triangle(j)));
44          DiffusionTerm = DiffusionTerm + dot(D(X(Triangle(j)),Y(
                Triangle(j)))*Grad_j,Grad_i);
45      end
46
47      % Calculate the integral volume, newton-cotes method.
48      I1 = (1/3)*g*Phi_i*Phi_j*TriangleArea;
49
50      I2 = (1/3)*DiffusionTerm*TriangleArea;
51
52      I = I1 + I2;
53
54      % Calculate the integral over every triangle.
55      Q = I + Q;
56  end
57
58  end
```

B.8 Updating the mesh

The last part of our code that we need to explain and introduce is the function "UpdateMesh", which can be found in the below Listing B.32. This function uses the methods discussed in §5.3.2 to decide when to change the connectivity matrix. By deleting rows from the connectivity matrix we basically remove the interactions between nodes that are all sufficiently damaged. A node is sufficiently damaged when we have that $\alpha \geq 1 - \delta$, where $\delta \ll 1$. We assign $\text{Torn}(\mathbf{X}) = 1$ for each of these nodes in this function, which the rest of our code will use to inform us if a node has reached the max damage. If all the triangles related to a node are removed from from the connectivity matrix then it has effectively been deleted. Visible tears in the macroscopic mesh form as sufficiently many triangles are deleted from the connectivity matrix.

Listing B.32: This Matlab function is used to update the connectivity matrix and record which nodes in our system are sufficiently damaged such that we can list them as torn. To do this we update our torn field such that $\text{Torn}(\mathbf{X}) = 1$ for any node where $\alpha \geq 1 - \delta$, where $\delta \ll 1$. Using this list of torn nodes in our system we can remove any triangle from the connectivity matrix where all the vertices are torn.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% So this function creates the updated connectivity list based
   on the
% TornNodes in our system.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [triangles, TornNodes] = UpdatedMesh(A,
   OriginalTriangles,delta)

% Foward Nudge to get Alpha == 1.
TornNodes = 0.*(A < 1 - delta) + 1.*(A >= 1 - delta);

% Setup broken triangle array.
Broke = zeros(size(OriginalTriangles,1),1);

% Systematically delete all triangles surronded by torn points.
for i = 1:size(Broke,1)
    K = OriginalTriangles(i,:);
    if sum(TornNodes(K)) >= length(K)
        Broke(i) = 1;
    end
end

% Locate all broken triangles
[B,~] = find(Broke == 1);
OriginalTriangles(B,:) = [];

% Hence our mesh is.
triangles = OriginalTriangles;

% Lastly update TornNodes to include all points that are no
   longer
```

```
% connected to the mesh.
for i = 1:length(TornNodes)
    if sum(sum(triangles == i)) == 0
        TornNodes(i) = 1;
    end
end

end
```

Bibliography

[1] Salma Adham et al. "Spontaneous cervical artery dissection in vascular Ehlers-Danlos syndrome: a cohort study". In: *Stroke* 52.5 (2021), pp. 1628–1635.

[2] Koichi Akutsu. "Etiology of aortic dissection". In: *General thoracic and cardiovascular surgery* 67 (2019), pp. 271–276.

[3] Erik Andreassen and Casper Schousboe Andreasen. "How to determine composite material properties using numerical homogenization". In: *Computational Materials Science* 83 (2014), pp. 488–495.

[4] Dimitrios C. Angouras, Eleftherios P. Kritharis, and Dimitrios P. Sokolis. "Regional distribution of delamination strength in ascending thoracic aortic aneurysms." In: *Journal of the mechanical behavior of biomedical materials* 98 (2019), pp. 58–70.

[5] J Auer, R Berent, and B Eber. "Aortic dissection: incidence, natural history and impact of surgery". In: *Journal of Clinical and Basic Cardiology* 3.3 (2000), pp. 151–154.

[6] Anju R Babu, Achu G Byju, and Namrata Gundiah. "Biomechanical properties of human ascending thoracic aortic dissections". In: *Journal of biomechanical engineering* 137.8 (2015), p. 081013.

[7] Nikolai Sergeevich Bakhvalov and Grigory Panasenko. *Homogenisation: averaging processes in periodic media: mathematical problems in the mechanics of composite materials*. Vol. 36. Springer Science & Business Media, 2012.

[8] John Barbetseas et al. "Atherosclerosis of the aorta in patients with acute thoracic aortic dissection". In: *Circulation Journal* 72.11 (2008), pp. 1773–1776.

[9] Alain Bensoussan, Jacques-Louis Lions, and George Papanicolaou. *Asymptotic analysis for periodic structures*. Vol. 374. American Mathematical Soc., 2011.

[10] Javier Bonet and Richard D Wood. *Nonlinear continuum mechanics for finite element analysis*. Cambridge university press, 1997.

[11] René de Borst and Clemens V Verhoosel. "Gradient damage vs phase-field approaches for fracture: Similarities and differences". In: *Computer Methods in Applied Mechanics and Engineering* 312 (2016), pp. 78–94.

[12] Benjamin S Brooke, Satyajit K Karnik, and Dean Y Li. "Extracellular matrix in vascular morphogenesis and disease: structure versus signal". In: *Trends in cell biology* 13.1 (2003), pp. 51–56.

[13] DB Camasão and DJMTB Mantovani. "The mechanical characterization of blood vessels and their substitutes in the continuous quest for physiological-relevant performances. A critical review". In: *Materials Today Bio* 10 (2021), p. 100106.

[14] Peter B Canham et al. "Measurements from light and polarised light microscopy of human coronary arteries fixed at distending pressure". In: *Cardiovascular research* 23.11 (1989), pp. 973–982.

[15] L Cardamone et al. "Origin of axial prestretch and residual stress in arteries". In: *Biomechanics and modeling in mechanobiology* 8 (2009), pp. 431–446.

[16] Thomas E Carew, Ramesh N Vaishnav, and Dali J Patel. "Compressibility of the arterial wall". In: *Circulation research* 23.1 (1968), pp. 61–68.

[17] Raminta Cerneviciute and Colin D Bicknell. "Acute type B aortic dissection". In: *Surgery (Oxford)* (2024).

[18] Krishna S Challagulla, AV Georgiades, and AL Kalamkarov. "Asymptotic homogenization modeling of thin composite network structures". In: *Composite structures* 79.3 (2007), pp. 432–444.

[19] A Cherkaev and R Kohn. *Topics in the Mathematical Modelling of Composite Materials*. Birkhauser Cham, 2018.

[20] Cheng-Jen Chuong and Yuan-Cheng Fung. "Residual stress in arteries". In: *Frontiers in biomechanics*. Springer, 1986, pp. 117–129.

[21] Doina Cioranescu and Patrizia Donato. *An introduction to homogenization*. Oxford university press, 1999.

[22] JM Clark and S Glagov. "Transmural organization of the arterial media. The lamellar unit revisited." In: *Arteriosclerosis* (1985), pp. 19–34.

[23] Joe Collis, Matthew E Hubbard, and Reuben D O'DEA. "A multi-scale analysis of drug transport and response for a multi-phase tumour model". In: *European Journal of Applied Mathematics* 28.3 (2017), pp. 499–534.

[24] Joe Collis et al. "Effective equations governing an active poroelastic medium". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 473.2198 (2017), p. 20160755.

[25] Peter Congdon. *Bayesian statistical modelling*. John Wiley & Sons, 2007.

[26] Frank J Criado. "Aortic dissection: a 250-year perspective". In: *Texas Heart Institute Journal* 38.6 (2011), p. 694.

[27] Oscar Luis Cruz-González et al. "A hierarchical asymptotic homogenization approach for viscoelastic composites". In: *Mechanics of Advanced Materials and Structures* 28.21 (2021), pp. 2190–2201.

[28] Franz Dammaß et al. "Phase-field modelling and analysis of rate-dependent fracture phenomena at finite deformation". In: *Computational Mechanics* 72.5 (2023), pp. 859–883.

[29] Michael E DeBakey, Antonio M Gotto Jr, and J Willis Hurst. "Profiles in cardiology: Michael E. DeBakey". In: *Clinical Cardiology* 14.12 (1991), pp. 1007–1011.

[30] A Delfino et al. "Residual strain effects on the stress field in a thick wall finite element model of the human carotid bifurcation". In: *Journal of biomechanics* 30.8 (1997), pp. 777–786.

[31] Funda Aksu Denli et al. "A phase-field model for fracture of unidirectional fiber-reinforced polymer matrix composites". In: *Computational Mechanics* 65 (2020), pp. 1149–1166.

[32] Salvatore Di Stefano et al. "Effective balance equations for electrostrictive composites". In: *Zeitschrift für angewandte Mathematik und Physik* 71 (2020), pp. 1–36.

[33] Robert S Dieter, Raymond A Dieter Jr, and Raymond A Dieter III. *Diseases of the Aorta*. Springer Nature, 2019.

[34] P Dobrin, T Canfield, and S Sinha. "Development of longitudina retraction of carotid arteries in neonatal dogs". In: *Experientia* 31.11 (1975), pp. 1295–1296.

[35] Philip B Dobrin, Thomas H Schwarcz, and Robert Mrkvicka. "Longitudinal retractive force in pressurized dog and human arteries". In: *Journal of Surgical Research* 48.2 (1990), pp. 116–120.

[36] Richard L. Drake, A. Wayne Vogl, and Adam W.M. Mitchell. *Gray's Anatomy for Students (Fifth Edition)*. ClinicalKey, 2024. Chap. 1 - Cardiovascular System, e1–e62.

[37] Pavel Dutov et al. "Measurement of elastic modulus of collagen type I single fiber". In: *PloS one* 11.1 (2016), e0145711.

[38] John A Elefteriades. "Thoracic aortic aneurysm: reading the enemy's playbook". In: *Current problems in cardiology* 33.5 (2008), pp. 203–277.

[39] MGGV Elices et al. "The cohesive zone model: advantages, limitations and challenges". In: *Engineering fracture mechanics* 69.2 (2002), pp. 137–163.

[40] Francesca Fantoni et al. "A phase field approach for damage propagation in periodic microstructured materials". In: *International Journal of Fracture* 223 (2020), pp. 53–76.

[41] PJ128117 Flory. "Thermodynamic relations for high elastic materials". In: *Transactions of the Faraday Society* 57 (1961), pp. 829–838.

[42] YC Fung, K Fronek, and P Patitucci. "Pseudoelasticity of arteries and the choice of its mathematical expression". In: *American Journal of Physiology-Heart and Circulatory Physiology* 237.5 (1979), H620–H631.

[43] Yuan Cheng Fung. "What are the residual stresses doing in our blood vessels?" In: *Annals of biomedical engineering* 19 (1991), pp. 237–249.

[44] Yuan-cheng Fung. *Biomechanics: mechanical properties of living tissues*. Springer Science & Business Media, 2013.

[45] T Christian Gasser and Gerhard A Holzapfel. "Modeling the propagation of arterial dissection". In: *European Journal of Mechanics-A/Solids* 25.4 (2006), pp. 617–633.

[46] T Christian Gasser, Ray W Ogden, and Gerhard A Holzapfel. "Hyperelastic modelling of arterial layers with distributed collagen fibre orientations". In: *Journal of the royal society interface* 3.6 (2006), pp. 15–35.

[47] Alfonso Gautieri et al. "Modeling and measuring visco-elastic properties: From collagen molecules to collagen fibrils". In: *International Journal of Non-Linear Mechanics* 56 (2013), pp. 25–33.

[48] Eileen Gentleman et al. "Mechanical characterization of collagen fibers and scaffolds for tissue engineering". In: *Biomaterials* 24.21 (2003), pp. 3805–3813.

[49] Alessandro Giudici, Ian B Wilkinson, and Ashraf W Khir. "Review of the techniques used for investigating the role elastin and collagen play in arterial wall mechanics". In: *IEEE reviews in biomedical engineering* 14 (2020), pp. 256–269.

[50] Jonathan Golledge and Kim A Eagle. "Acute aortic dissection". In: *The Lancet* 372 (2008), pp. 55–66.

[51] Alan Arnold Griffith. "VI. The phenomena of rupture and flow in solids". In: *Philosophical transactions of the royal society of london. Series A, containing papers of a mathematical or physical character* 221.582-593 (1921), pp. 163–198.

[52] JoséMiranda Guedes and Noboru Kikuchi. "Preprocessing and postprocessing for materials based on the homogenization method with adaptive finite element methods". In: *Computer methods in applied mechanics and engineering* 83.2 (1990), pp. 143–198.

[53] Osman Gültekin, Hüsnü Dal, and Gerhard A Holzapfel. "Numerical aspects of anisotropic failure in soft biological tissues favor energy-based criteria: A rate-dependent anisotropic crack phase-field model". In: *Computer methods in applied mechanics and engineering* 331 (2018), pp. 23–52.

[54] Osman Gültekin et al. "Computational modeling of progressive damage and rupture in fibrous biological tissues: application to aortic dissection". In: *Biomechanics and modeling in mechanobiology* 18 (2019), pp. 1607–1628.

[55] Namrata Gundiah, Mark B Ratcliffe, and Lisa A Pruitt. "Determination of strain energy function for arterial elastin: experiments using histology and mechanical tests". In: *Journal of biomechanics* 40.3 (2007), pp. 586–594.

[56] Koichi Hashiguchi and Yuki Yamakawa. *Introduction to finite strain theory for continuum elasto-plasticity*. English. Chichester, West Sussek, U.K: Wiley, 2012.

[57] Behrooz Hassani and E Hinton. "A review of homogenization and topology opimization II—analytical and numerical solution of homogenization equations". In: *Computers & structures* 69.6 (1998), pp. 719–738.

[58] Behrooz Hassani and Ernest Hinton. "A review of homogenization and topology optimization I—homogenization theory for media with periodic structure". In: *Computers & Structures* 69.6 (1998), pp. 707–717.

[59] Behrooz Hassani and Ernest Hinton. "A review of homogenization and topology optimization III—topology optimization using optimality criteria". In: *Computers & structures* 69.6 (1998), pp. 739–756.

[60] Neil Herring and David J. Paterson. *Levick's introduction to cardiovascular physiology*. CRC Press, 2018. Chap. 1 - Overview of the cardiovascular system, pp. 1–13.

[61] Aghigh Heydari, Atefeh Asadmobini, and Feridoun Sabzi. "Anabolic steroid use and aortic dissection in athletes: a case series". In: *Anabolic steroid use and aortic dissection in athletes: a case series* 35 (2020).

[62] Elizabeth C Holden et al. "A multiphase multiscale model for nutrient limited tissue growth". In: *The ANZIAM Journal* 59.4 (2018), pp. 499–532.

[63] Mark H Holmes. *Introduction to perturbation methods*. Vol. 20. Springer Science & Business Media, 2012.

[64] G.A. Holzapfel, G. Sommer, and M. Auer et al. "Layer-Specific 3D Residual Deformations of Human Aortas with Non-Atherosclerotic Intimal Thickening". In: *Annals of Biomedical Engineering* 35 (2007), pp. 530–545.

[65] Gerhard A Holzapfel, Thomas C Gasser, and Ray W Ogden. "A new constitutive framework for arterial wall mechanics and a comparative study of material models". In: *Journal of elasticity and the physical science of solids* 61 (2000), pp. 1–48.

[66] Gerhard A Holzapfel and Ray W Ogden. "Modelling the layer-specific three-dimensional residual stresses in arteries, with an application to the human aorta". In: *Journal of the Royal Society Interface* 7.46 (2010), pp. 787–799.

[67] Muneo Hori and Sia Nemat-Nasser. "On two micromechanics theories for determining micro–macro relations in heterogeneous solids". In: *Mechanics of materials* 31.10 (1999), pp. 667–682.

[68] Lukas Horny et al. "Orientations of collagen fibers in aortic histological section". In: *Bulletin of applied mechanics* 6.22 (2010), pp. 25–29.

[69] Jay D Humphrey. "Mechanics of the arterial wall: review and directions". In: *Critical Reviews™ in Biomedical Engineering* 23.1-2 (1995).

[70] Paul A. Iaizzo. *Handbook of Cardiac Anatomy, Physiology, and Devices.* Humana Press, 2005. Chap. 1.

[71] Franz F Immer et al. "Aortic dissection in pregnancy: analysis of risk factors and outcome". In: *The Annals of thoracic surgery* 76.1 (2003), pp. 309–314.

[72] The MathWorks Inc. *MATLAB version: 9.13.0 (R2022b)*. Natick, Massachusetts, United States, 2022. URL: `https://www.mathworks.com`.

[73] The MathWorks Inc. *Partial Differential Equations Toolbox (R2022b)*. Natick, Massachusetts, United States, 2022. URL: `https://www.mathworks.com`.

[74] George R Irwin. "Analysis of stresses and strains near the end of a crack traversing a plate". In: *Journal of Applied Mechanics* (1957).

[75] R.M Jones. *Mechanics Of Composite Materials (2nd ed.).* CRC Press., 1999.

[76] Ijaz A. Khan and Chandra K. Nair. "Clinical, diagnostic, and management perspectives of aortic dissection". In: *Chest* 122 (2002), pp. 311–328.

[77] Kyung-Suk Kim and N Aravas. "Elastoplastic analysis of the peel test". In: *International Journal of Solids and Structures* 24.4 (1988), pp. 417–435.

[78] Nicholas T. Kouchoukos and Dimitrios Dougenis. "Surgery of the Thoracic Aorta". In: *New England Journal of Medicine* 336.26 (1997), pp. 1876–1889.

[79] Maya Landenhed et al. "Risk profiles for aortic dissection and ruptured or surgically treated aneurysms: a prospective cohort study". In: *Journal of the American Heart Association* 4.1 (2015), e001513.

[80] MA Lillie and JM Gosline. "Limits to the durability of arterial elastic tissue". In: *Biomaterials* 28.11 (2007), pp. 2021–2031.

[81] Pu-Song Ma et al. "Asymptotic homogenization of phase-field fracture model: An efficient multiscale finite element framework for anisotropic fracture". In: *International Journal for Numerical Methods in Engineering* (2024), e7489.

[82] Nathan G March, Elliot J Carr, and Ian W Turner. "A fast algorithm for semi-analytically solving the homogenization boundary value problem for block locally-isotropic heterogeneous media". In: *Applied Mathematical Modelling* 92 (2021), pp. 23–43.

[83] Jean-Jacques Marigo, Corrado Maurini, and Kim Pham. "An overview of the modelling of fracture by gradient damage models". In: *Meccanica* 51 (2016), pp. 3107–3128.

[84] Jean-Jacques Marigo, Corrado Maurini, and Kim Pham. "Gradient damage models and their use in brittle fracture". In: *International Journal of Damage Mechanics* (2016).

[85] Takeo Matsumoto and Kazuaki Nagayama. "Tensile properties of vascular smooth muscle cells: bridging vascular and cellular biomechanics". In: *Journal of biomechanics* 45.5 (2012), pp. 745–755.

[86] Takeo Matsumoto, Shukei Sugita, and Kazuaki Nagayama. "Tensile properties of smooth muscle cells, elastin, and collagen fibers". In: *Vascular Engineering: New Prospects of Vascular Medicine and Biology with a Multidiscipline Approach* (2016), pp. 127–140.

[87] Rajendra H Mehta et al. "Predicting death in patients with acute type A aortic dissection". In: *Circulation* 105.2 (2002), pp. 200–206.

[88] Inga H Melvinsdottir et al. "The incidence and mortality of acute thoracic aortic dissection: results from a whole nation study". In: *European Journal of Cardio-Thoracic Surgery* 50.6 (2016), pp. 1111–1117.

[89] Christian Miehe and Steffen Mauthe. "Phase field modeling of fracture in multi-physics problems. Part III. Crack driving forces in hydro-poro-elasticity and hydraulic fracturing of fluid-saturated porous media". In: *Computer Methods in Applied Mechanics and Engineering* 304 (June 2016), pp. 619–655. ISSN: 00457825. DOI: 10.1016/j.cma.2015.09.021.

[90] Christian Miehe, Fabian Welschinger, and Martina Hofacker. "Thermodynamically consistent phase-field models of fracture: Variational principles and multi-field FE implementations". In: *International journal for numerical methods in engineering* 83.10 (2010), pp. 1273–1311.

[91] Russell B Millar. *Maximum likelihood estimation and inference: with examples in R, SAS and ADMB*. John Wiley & Sons, 2011.

[92] Laura Miller and Raimondo Penta. "Homogenization of a coupled electrical and mechanical bidomain model for the myocardium". In: *Mathematics and Mechanics of Solids* (2023), p. 10812865231207600.

[93] Laura Miller and Raimondo Penta. "Homogenized balance equations for nonlinear poroelastic composites". In: *Applied Sciences* 11.14 (2021), p. 6611.

[94] J Mo, G Milleret, and M Nagaraj. "Liquid crystal nanoparticles for commercial drug delivery". In: *Liquid crystals reviews* 5.2 (2017), pp. 69–85.

[95] Geza Mozes and Peter Gloviczki. *The Vein Book*. Academic Press, 2007. Chap. Chaptre 2 - Venous Embryology and Anatomy, pp. 15–25.

[96] Debabrata Mukherjee and Kim A. Eagle. "Aortic Dissection—An Update". In: *Current Problems in Cardiology* 30 (2005), pp. 287–325.

[97] Raymond W Ogden. *Non-linear elastic deformations*. Courier Corporation, 1997.

[98] RW Ogden. "Nearly isochoric elastic deformations: application to rubberlike solids". In: *Journal of the Mechanics and Physics of Solids* 26.1 (1978), pp. 37–57.

[99] Davide Pacini et al. "Acute aortic dissection: epidemiology and outcomes". In: *International journal of cardiology* 167.6 (2013), pp. 2806–2812.

[100] Linda A Pape et al. "Presentation, diagnosis, and outcomes of acute aortic dissection: 17-year trends from the International Registry of Acute Aortic Dissection". In: *Journal of the American College of Cardiology* 66.4 (2015), pp. 350–358.

[101] Achilles J. Pappano and Withrow Gil Wier. *Cardiovascular Physiology (Tenth Edition)*. Elsevier, 2013. Chap. 1 - Overview of the Circulation and Blood, pp. 1–9.

[102] R Penta, D Ambrosi, and A32772851307 Quarteroni. "Multiscale homogenization for fluid and drug transport in vascularized malignant tissues". In: *Mathematical Models and Methods in Applied Sciences* 25.01 (2015), pp. 79–108.

[103] Raimondo Penta and Alf Gerisch. "An introduction to asymptotic homogenization". In: *Multiscale Models in Mechano and Tumor Biology: Modeling, Homogenization, and Applications*. Springer. 2017, pp. 1–26.

[104] Raimondo Penta and Alf Gerisch. "The asymptotic homogenization elasticity tensor properties for composites with material discontinuities". In: *Continuum Mechanics and Thermodynamics* 29 (2017), pp. 187–206.

[105] Raimondo Penta et al. "Effective balance equations for elastic composites subject to inhomogeneous potentials". In: *Continuum Mechanics and Thermodynamics* 30.1 (Jan. 2018), pp. 145–163. ISSN: 09351175. DOI: 10.1007/s00161-017-0590-x.

[106] Luca Placidi. "A variational approach for a nonlinear 1-dimensional second gradient continuum damage model". In: *Continuum Mechanics and Thermodynamics* 27 (2015), pp. 623–638.

[107] William H Press et al. *Numerical recipes*. Cambridge University Press, London, England, 1988.

[108] Arun Raina and Christian Miehe. "A phase-field model for fracture in biological tissues". In: *Biomechanics and modeling in mechanobiology* 15.3 (2016), pp. 479–496.

[109] Ariel Ramirez-Torres et al. "Three scales asymptotic homogenization and its application to layered hierarchical hard tissues". In: *International Journal of Solids and Structures* 130 (2018), pp. 190–198.

[110] Margot R Roach and Alan C Burton. "The reason for the shape of the distensibility curves of arteries". In: *Canadian journal of biochemistry and physiology* 35.8 (1957), pp. 681–690.

[111] Anne M. Robertson and Paul N. Watton. "Chapter 8 - Mechanobiology of the Arterial Wall". In: *Transport in Biological Media*. Ed. by Sid M. Becker and Andrey V. Kuznetsov. Boston: Elsevier, 2013, pp. 275–347.

[112] Alejandro Roque-Piedra et al. "Effective properties of homogenised nonlinear viscoelastic composites". In: *Materials* 16.11 (2023), p. 3974.

[113] Pascale Royer. "Advection–diffusion in porous media with low scale separation: modelling via higher-order asymptotic homogenisation". In: *Transport in Porous Media* 128.2 (2019), pp. 511–551.

[114] Enrique Sánchez-Palencia. "Non-homogeneous media and vibration theory". In: *Lecture Note in Physics, Springer-Verlag* 320 (1980), pp. 57–65.

[115] Juan E Santos and Dongwoo Sheen. "Derivation of a Darcy's law for a porous medium composed of two solid phases saturated by a single-phase fluid: A homogenization approach". In: *Transport in porous media* 74 (2008), pp. 349–368.

[116] Frank W. Sellke et al. *Aortic dissection and acute aortic syndromes*. Springer, 2021. Chap. 1, pp. 3–17.

[117] Selda Sherifova and Gerhard A Holzapfel. "Biomechanics of aortic wall failure with a focus on dissection and aneurysm: A review". In: *Acta biomaterialia* 99 (2019), pp. 1–17.

[118] William M Sherk, Minhaj S Khaja, and David M Williams. "Anatomy, pathology, and classification of aortic dissection". In: *Techniques in Vascular and Interventional Radiology* 24.2 (2021), p. 100746.

[119] Javed Iqbal Siddique et al. "A review of mixture theory for deformable porous media and applications". In: *Applied Sciences* 7.9 (2017), p. 917.

[120] Eric M Siegal. "Acute aortic dissection". In: *Journal of Hospital Medicine* 1 (2006), pp. 94–105.

[121] William S Slaughter. *The linearized theory of elasticity*. Springer Science & Business Media, 2012.

[122] John D Smith. "Application of the method of asymptotic homogenization to an acoustic metafluid". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 467.2135 (2011), pp. 3318–3331.

[123] Jan A Staessen et al. "Essential hypertension". In: *The Lancet* 361.9369 (2003), pp. 1629–1641.

[124] Jialin Su et al. "Cocaine Induces Apoptosis in Primary Cultured Rat Aortic Vascular Smooth Muscle Cells: Possible Relationship to Aortic Dissection, Atherosclerosis, and Hypertension". In: *International journal of toxicology* 23 (2004), pp. 233–237.

[125] Chin-Teh Sun and Rajesh S Vaidya. "Prediction of composite properties from a representative volume element". In: *Composites science and Technology* 56.2 (1996), pp. 171–179.

[126] Amy SM Tam, M Catherine Sapp, and Margot R Roach. "The effect of tear depth on the propagation of aortic dissections in isolated porcine thoracic aorta". In: *Journal of biomechanics* 31.7 (1998), pp. 673–676.

[127] Thomas T Tsai et al. "Tear size and location impacts false lumen pressure in an ex vivo model of chronic type B aortic dissection". In: *Journal of vascular surgery* 47.4 (2008), pp. 844–851.

[128] Ramesh N Vaishnav, John T Young, and Dali J Patel. "Distribution of stresses and of strain-energy density through the wall thickness in a canine aortic segment". In: *Circulation research* 32.5 (1973), pp. 577–583.

[129] Augusto Visintin. "On the homogenization of visco-elastic processes". In: *The IMA Journal of Applied Mathematics* 77.6 (2012), pp. 869–886.

[130] KY Volokh. "Fung's model of arterial wall enhanced with a failure description". In: *Molecular & Cellular Biomechanics* 5.3 (2008), p. 207.

[131] KY Volokh. "Hyperelasticity with softening for modeling materials failure". In: *Journal of the Mechanics and Physics of Solids* 55.10 (2007), pp. 2237–2264.

[132] George Z Voyiadjis. *Handbook of damage mechanics: nano to macro scale for materials and structures*. Springer Reference, 2014.

[133] R Wait and AR Mitchell. *Finite element analysis and applications*. John Wiley and sons, 1986.

[134] Lei Wang et al. "Modelling of tear propagation and arrest in fibre-reinforced soft tissue subject to internal pressure". In: *Journal of Engineering Mathematics* 95.1 (2015), pp. 249–265.

[135] Lei Wang et al. "Modelling peeling-and pressure-driven propagation of arterial dissection". In: *Journal of engineering mathematics* 109 (2018), pp. 227–238.

[136] Hans W Weizsacker and John G Pinto. "Isotropy and anisotropy of the arterial wall". In: *Journal of biomechanics* 21.6 (1988), pp. 477–487.

[137] H Wolinsky and S Glagov. "A Lamellar Unit of Aortic Medial Structure and Function in Mammals". In: *Circulation Research* 20.1 (1967), pp. 99–111.

[138] Harvey Wolinsky and Seymour Glagov. "Structural basis for the static mechanical properties of the aortic media". In: *Circulation research* 14.5 (1964), pp. 400–413.

[139] Jian-Ying Wu. "A unified phase-field theory for the mechanics of damage and quasi-brittle failure". In: *Journal of the Mechanics and Physics of Solids* 103 (2017), pp. 72–99.

[140] Hua Yang et al. "Determination of metamaterial parameters by means of a homogenization approach based on asymptotic analysis". In: *Continuum mechanics and thermodynamics* 32 (2020), pp. 1251–1270.

[141] Y Yang et al. "Nonlinear asymptotic homogenization and the effective behavior of layered thermoelectric composites". In: *Journal of the Mechanics and Physics of Solids* 61.8 (2013), pp. 1768–1783.

[142] Yeong-Moo Yi, Sang-Hoon Park, and Sung-Kie Youn. "Asymptotic homogenization of viscoelastic composites with periodic microstructures". In: *International Journal of Solids and Structures* 35.17 (1998), pp. 2039–2055.

[143] Shi-Min Yuan. "Aortic dissection during pregnancy: a difficult clinical scenario". In: *Clinical Cardiology* 36.10 (2013), pp. 576–584.

[144] Alan T Zehnder. "Lecture notes on fracture mechanics". In: *Cornell University* 20 (2007), p. 22.

[145] Bo Zhang et al. "Imaging and analyzing the elasticity of vascular smooth muscle cells by atomic force acoustic microscope". In: *Ultrasound in medicine & biology* 38.8 (2012), pp. 1383–1390.