NTNU – Trondheim
Norwegian University of
Science and Technology

FEM Simluations of Shear Waves in the Heart Wall

Erik Løhre Grimsmo

Mechanical Engineering

Submission date:	June 2013
Supervisor:	Leif Rune Hellevik, KT
Co-supervisor:	Victorien Prot, KT

Norwegian University of Science and Technology
Department of Structural Engineering

Department of Structural Engineering
Faculty of Engineering Science and Technology
NTNU - Norwegian University of Science and Technology

MASTER THESIS 2013

for

Erik Løhre Grimsmo

FEM simulations of shear waves in the heart wall

Elementmetode-simuleringer av skjærbølger i hjerteveggen

Assignment text

The heart is a muscle, which is responsible for the blood circulation in the body. Some heart diseases cause the heart muscle to stiffen, and this creates problems with pumping of the blood. The consequence is that the heart pumps less efficiently, and that the patient's exercise capacity becomes worse. It is hard to find the stiffness of the heart muscle *in vivo*, but with new ultrasound technology it is possible to take pictures with an extreme frame rate. It is therefore possible to observe mechanical phenomenons that have not been observed before, e.g. fast deformation waves. Our hypothesis is that the velocity of these waves can say something about the stiffness of the heart muscle. We want to investigate details of the wave propagation with the help of the finite element method and relate the results to ultrasonic observations. We also want to investigate whether we can estimate the stiffness of the heart wall, which is modeled with a transversely isotropic material. The suggested content for this thesis is therefore to present:

- Theory of wave propagation in linear isotropic and anisotropic elastic materials.
- Finite element analyses of wave propagation in anisotropic media.
- Finite element analyses of a model that simulates the wave propagation in the heart wall.

The thesis should be organized according to existing guidelines.

Supervisors: Prof. Leif Rune Hellevik, Assoc. Prof. Victorien Prot

The thesis should be delivered at Department of Structural Engineering within the 10th of June.

Prof. Leif Rune Hellevik

Sammendrag

Hvis medisinske klinikere kan si noe om stivheten i hjertet til en pasient, kan de vurdere hjertets helsetilstand. Et hjerteanfall er en typisk årsak til at stivheten i hjertet endres. Med ny ultralydteknologi er det mulig å observere skjærbølger som forplanter seg i hjerteveggen på grunn av lukkingen av aortaklaffen. Vårt mål var å lage en elementmetode-modell, hvor vi kunne simulere denne skjærbølgeforplantningen langs hjerteveggen. Vi ønsket å undersøke muligheten for å estimere stivheten av en hjertemodell, hvor et transvers-isotropt materiale er implementert, basert på skjærbølgehastigheter. Vår tilnærming var å først lage følgende enkle elementmetode-modeller vi kunne validere med plan bølgeteori:

- Plane Wave Model - En enkel to-dimensjonal modell, med det transvers-isotrope materialet implementert, som simulerer plan bølgepropagasjon. Resultatet stemmer godt overens med teorien (mindre enn 2% avvik).
- Finite-sized Wave Model - En to-dimensjonal transvers-isotrop modell som har en bølgekilde av endelig størrelse. Vi observerte at plan bølgeteori ga tilfredstillende prediksjoner for hvordan bølger propagerer fra bølgekilden (mindre enn 6% avvik).

Deretter lagde vi modeller som kan simulere bølgepropagasjon ned langs hjerteveggen:

- Curved Model - En tre-dimensjonal modell som kan representere en del av hjerteveggen. Modellen er lagd for å undersøke effekten kurvatur har på skjærbølgehastigheten. Vi fant at endring i kurvaturen har liten påvirkning på bølgehastigheten, både når et isotropt materiale og det transvers-isotrope materialet er implementert.
- Truncated Ellipsoid Model - En tre-dimensjonal modell som er en enkel representasjon av venstre ventrikkel. Vi observerte at skjærbølgehastigheten varierte ($\pm 5\%$ fra en gjennomsnittsverdi) gjennom tykkelsen av hjerteveggen på grunn av anisotropi. Vi argumenterer for at dagens ultralydteknologi antageligvis ikke klarer å fange opp denne variasjonen basert på at måleusikkerheten er relativt høy. Videre foreslår vi at det å anta isotropi kan være akseptabelt selv om det transvers-isotrope materialet er implementert. Dette tillater oss å estimere stivheten av modellen.

Andre viktige resultater:

- Vi presenterer nye analytiske uttrykk for materialforskyvnings-vektorene (polariseringen) for plane bølger som propagerer i transvers-isotrope medier.
- En ny type av karakteristiske overflater som beskriver relasjonen mellom skjærbølgehastighet, propagasjonsretning og stivhet er også presentert.

Med denne oppgaven viser vi hvordan elementmetoden kan brukes til å undersøke detaljer i bølgepropagasjonen ned langs hjerteveggen.

Abstract

With knowledge of the stiffness of the heart wall, medical personnel can diagnose patients' heart condition. The stiffness of the heart can be altered, e.g. due to a heart attack. New ultrasound technology allows for *in vivo* observation of shear waves that propagate in the heart wall due to aortic-valve closure. Our main objective was to simulate this shear wave propagation along the heart wall with the help of the finite element method (FEM). We wanted to investigate whether we could obtain an estimate of the stiffness, based on measured shear wave speeds, in a heart wall modeled with a transversely isotropic material. To achieve this objective, we first made the following FEM models that we validated with plane wave theory:

- Plane Wave Model - A simple two-dimensional model with the transversely isotropic material implemented, which simulates plane wave propagation. The results agreed excellently with theory (less than 2% discrepancy).
- Finite-sized Wave Model - A two-dimensional transversely isotropic model that has a finite-sized source. We observed that plane wave theory gave satisfactory predictions for waves that propagate from the source (less than 6% discrepancy).

Then models that can simulate wave propagation down the heart wall were made:

- Curved Model - A three-dimensional model, which can represent a part of the heart wall. The model is made for investigation of the effect of curvature on wave speed, and we found that change in curvature only slightly affects the wave speed for both an isotropic and the transversely isotropic material.
- Truncated Ellipsoid Model - A three-dimensional model, which is a simple representation of the left ventricle. It was observed that the shear wave speed varied ($\pm 5\%$ from an average value) through the thickness of the model because of the transversely isotropic material implemented. We argue that current ultrasound technology is probably not able to capture this variation due to relatively high measuring uncertainty. Furthermore, we suggest that an assumption of isotropy can be acceptable even when the transversely isotropic material is implemented. This allows us to make an estimate of the stiffness of the model.

Other important results:

- We present novel analytical expressions for the material displacement vectors for plane waves that propagate in transversely isotropic media.
- A new type of characteristic surfaces that describe the relation between shear wave speed, propagation direction and stiffness is presented.

In conclusion, we show how we can investigate details of the wave propagation with the aid of the finite element method.

Preface

This master thesis is submitted to the Norwegian University of Science and Technology (NTNU) for the degree Master of Science. The work has been carried out at the Department of Structural Engineering, NTNU. My supervisors have been Prof. Leif Rune Hellevik and Assoc. Prof. Victorien Prot, both from the Department of Structural Engineering, NTNU. The topic of the thesis has been developed in collaboration with researchers at the university hospital St. Olav's at Trondheim, Norway.

Acknowledgements

I would like to thank my supervisor Prof. Leif Rune Hellevik and co-supervisor Assoc. Prof. Victorien Prot for help and guidance during the work of this thesis. Especially, I thank Prof. Leif Rune Hellevik for providing me with useful knowledge and feedback, and for pointing me in the right direction for this thesis. I would also like to express my gratitude towards Assoc. Prof. Victorien Prot for helping me with, for example, finite element modeling. It has been easy to arrange informal meetings such that we could discuss the work of my thesis continuously throughout the semester, and of that I am grateful.

I would also like to thank Dr. Brage H. Amundsen for providing me with knowledge and literature on the work performed by the researchers at St. Olav's hospital.

Finally, I thank my co-students and friends Lars Edvard Dæhli and Knut Andreas Kvåle for fruitful discussions and advice.

Trondheim, June 2013
Erik Grimsmo

Contents

List of Figures

List of Tables

Notation

Abbreviations

1D	One-dimensional
2D	Two-dimensional
3D	Three-dimensional
AVC	Aortic-Valve Closure
FEA	Finite Element Analysis/Analyses
FEM	Finite Element Method
NTNU	Norwegian University of Science and Technology
ODE	Ordinary Differential Equations
PDE	Partial Differential Equations
SH	Shear Horizontal
SV	Shear Vertical

The mathematical symbols used are listed below. Matrices and vectors are identified by boldface type in addition to brackets. The brackets are, however, left out of the figures to avoid clutter.

Mathematical symbols

$[\ ]$	Rectangular matrix or square matrix
$\{\ \}$	Column vector
$[\ ]^T$	Matrix transpose
$\|\|\ \|\|$	Norm of a matrix or vector
$\|\ \|$	Determinant of matrix
$\dot{}$	Time differentiation; for example, $\dot{u} = du/dt$
$'$	Differentiation with respect to the spatial variables; for example, $u'(x,y) = \partial u/\partial x$ or $u'(x,y) = \partial u/\partial y$

The Latin and Greek symbols are defined where they are first introduced in the text. Nevertheless, for the reader's convenience, the most important symbols are also listed below.

Latin symbols

$\{\mathbf{A}\}$	Material displacement vector; polarization vector
c	Wave velocity; phase velocity
$\{\mathbf{c}^e\}$	Energy velocity vector
$[\mathbf{C}]$	Material stiffness matrix; damping matrix
E	Young's Modulus; modulus of elasticity
G	Shear modulus
$\{\mathbf{n}\}$	Wave propagation direction; normal to phase plane
$\{\mathbf{T}\}$	Cauchy stress vector
t	Time
$\{\mathbf{u}\}$	Displacement vector

Greek symbols

$\{\epsilon\}$	Strain vector
$[\mathbf{\Gamma}]$	Acoustic tensor
λ	One of the Lamè constants
ν	Poisson's ratio
μ	One of the Lamè constants, equal to the shear modulus G
ψ	Skew angle
ρ	Mass density
ζ	Quasi angle

1 Introduction

Researchers have determined material properties from soft tissue imaging for several years. There are many techniques that can be mentioned: Palmeri *et al.* compare experimental results from *Acoustic Radiation Force Impulse* (ARFI) imaging with finite element analyses (FEA) [22]. They found that the shear wave speed agreed between the experimental and numerical analyses. Tabaru *et al.* discuss how ARF elastography can be used to determine shear and Young's moduli in soft tissues [25]. They also validate their result with FEA. Chen *et al.* use finite element method (FEM) modeling to validate the use of *Magnetic Resonance Elastography* (MRE) as a noninvasive diagnostic tool [7]. They compare the shear wave length obtained numerically with those found analytically and experimentally. They also look into the effects of material stiffness, density and excitation frequency on the shear wave length. Nenadic *et al.* determine viscoelastic material properties with a method called *Shearwave Dispersion Ultrasound Vibrometry* (SDUV), and they suggest that their technique can be used to determine material properties of the heart muscle [21]. Couade *et al.* use the technique called *Supersonic Shear Imaging* (SSI) to determine the elasticity of the heart wall of sheep *in vivo* [10]. They show how the stiffness of the heart varies at different times within a single heart cycle, and how the stiffness depends on the orientation of the ultrasound probe due to anisotropy in the heart wall. Even though soft tissue imaging is relevant for the topic at hand, it is not the focus of this thesis. We focus on wave propagation in the heart wall due to aortic-valve closure (AVC), not how it is observed experimentally.

At end-systole, the aortic valve closes and spontaneously excites small shock waves in the heart wall [6, 17]. Kanai was one of the first to exploit this wave propagation that comes from AVC [17]. He determined the elasticity and viscosity of the heart wall with the help of ultrasonic measurements. Brekke *et al.*, who are from the research group at St. Olav's Hospital, introduce a method they call *Ultra-high Frame Rate Tissue Doppler Imaging* (URF-TDI). This method can image two walls of the left ventricle at $\sim$1200 frames per second, and the high temporal resolution makes new information of the wave propagation from AVC available [6]. In particular, Brekke *et al.* have been able to measure the wave speed of the shear wave that propagates down *septum* due to AVC with the technique briefly described in Figure 1.1. Septum is the part of the heart wall that separates the atria and ventricles of the heart. As one can see later in this thesis, the shear wave speed is governed by the material properties, and can therefore be used to determine one of these properties, *e.g.* stiffness. Some heart diseases cause the heart muscle to become stiffer, which leads to lower exercise capacity, etc. Hence, with knowledge of the stiffness, medical doctors can diagnose patients' heart condition.

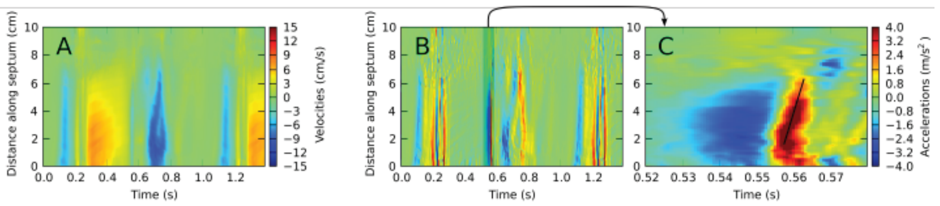


Figure 1.1: *This figure displays how Brekke et al. found the shear wave speed with the help of ultrasonic measurements. A: Velocity distribution along the septal wall from one cardiac cycle. B: Acceleration distribution along the septal wall, calculated from the velocities in A. C: A close-up of the time interval around AVC marked in B. A straight line has been aligned with the peak acceleration signal, and the slope of this line equaled the shear wave speed along septum. This figure is reprinted from [6].*

When the stiffness of the heart wall is determined from *in vivo* measurements, it is normal to assume isotropy in literature, see for example [10] and [17]. However, researchers generally agree that the heart wall is anisotropic due to muscle fiber orientation [14]. In this thesis, we therefore wanted to investigate if it is possible to estimate the stiffness of a model of the heart wall where a transversely isotropic material is implemented. And we wanted to relate the problem to the findings of Brekke *et al.* [6]. The main objective of the thesis was therefore to create an FEM model that simulates the wave propagation through the heart wall due to AVC. With an FEM model, one can investigate different aspects that affect the wave propagation, *e.g.* anisotropy of the heart wall. Due to the anisotropy, an understanding of wave propagation in anisotropic media is appropriate to acquire. A large part of this thesis is therefore dedicated to theory and simple FEM models that show how waves behave in anisotropic media.

The approach was to first create two simple two-dimensional transversely isotropic models that can be validated by plane wave theory. Then the observations and experience gained from these models was used to assert the results from more complicated models that were created to represent septum and the left ventricle. This was where one of the largest challenges lay: namely, to create a model that represented the heart wall in an accurate manner, but that could also be verified to some extent.

A pre-study project related to this thesis was performed during the fall semester of 2012 [12]. In this work, the author studied wave propagation in isotropic media, with focus on numerical accuracy and stability. It was observed that the wave speeds from theory agreed with those found from finite element analysis (FEA).

In this thesis, we show that from the models we could compare directly with theory, we obtained results that agreed well with the theory. For example, we saw that plane wave theory is a satisfactory tool for describing how waves propagate from finite-sized sources (a discrepancy of less than 6% was observed). With isotropic material implemented, the three-dimensional models gave results that also agreed well with plane wave theory. This gave confidence in these models and in the wave speed measurement procedure. On the other hand, with a transversely isotropic material implemented, plane wave theory was not sufficient to fully validate the results from the three-dimensional models. Nevertheless, we argue that the wave speeds obtained are reasonable. We observed that the shear wave speed varied through the thickness of the Truncated Ellipsoid Model due to layers with different material orientations. However, it is argued that the variation is probably too small to be captured by today's ultrasound technology, because Brekke *et al.* obtained a standard deviation of 25% from their ultrasonic experiments. And since the wave speed variation is only approximately $\pm5\%$ of the average value, it would be difficult to discern the variation with their measurement technique. Also, we suggest that this relatively low variation justifies an assumption of isotropy, which allows us to calculate an estimate of the stiffness, *i.e.* Young's modulus.

Theory used in the thesis is presented in Chapter 2. Here the reader can, for example, find theory on plane wave propagation in elastic media. In the same chapter, we present the methods used in this thesis, *e.g.* the details of the FEM modeling. In Chapter 3, two two-dimensional FEM models and two three-dimensional FEM models are presented and analyzed. We discuss general aspects of the results, limitations of the FEM models and the challenges for future work in Chapter 4. Finally, in Chapter 5 there are some concluding remarks.

2 Theory and methods

In this chapter, we present theory that we use to understand and analyze wave propagation in the heart wall. The first section is dedicated to explain some of the physical aspects of the heart. In the following sections we look into wave propagation in both isotropic and anisotropic media.

In the last two sections of this chapter, we discuss some of the methods used in this thesis. More specifically, we discuss some of the aspects of the softwares Maple and the FEM.

2.1 Physiological aspects of the heart

In this section we briefly discuss physiological features of the heart that are relevant for the work in this thesis.

2.1.1 The cause of stiffer heart wall

The theory in this subsection can be found in basic pathology books, see for example [3].

When a person suffers from heart attack, there is a sudden interruption of blood supply to a part of the heart. This leads to dead cells in that part of the heart wall. To maintain the level of blood flow to the body, the remaining functional parts of the heart must work harder. In other words, there is an increase in mechanical stress to the *myocardium*, the muscular tissue of the heart. The result is called *myocardial hypertrophy*, which means that the thickness of the heart wall is increased, see Figure 2.1. Many cardiac and metabolic diseases are caused by myocardial hypertrophy.

The increase in heart wall thickness does not only come from growth of new muscle fibers, but also from excessive formation of connective tissue, more commonly known as scar tissue, in the extracellular matrix. This phenomenon is called *fibrosis*, and results in a stiffer heart wall. The term fibrosis is used both for the case

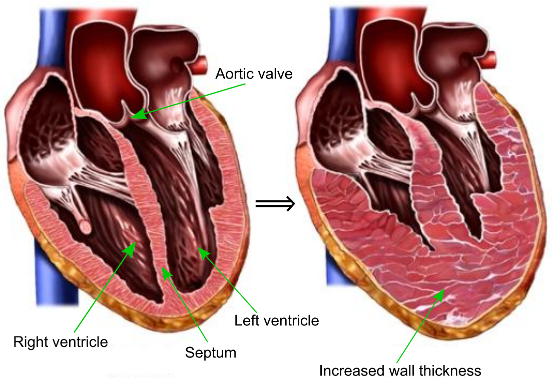


Figure 2.1: *Illustrations of heart cross sections; a normal heart (left) and a hypertrophic heart (right). The figure is adapted from: http://www.beliefnet.com/healthandhealing/getcontent.aspx?cid=615222.*

of localized formation of scar tissue and for the case when the fibrosis is diffusively distributed in the heart wall.

From an engineering point of view, we say that the material properties of the heart wall have changed due to the fibrosis. If we can determine the material properties, we can evaluate the condition of the heart.

2.1.2 Morphology and structure of left ventricle

There is an ongoing debate on the anisotropic microstructure of the heart [14]. Holzapfel *et al.* treat the left ventricle to be continuum composed of laminar sheets [14], which is the approach we implemented for two of the FEM models in this thesis. About 70 per cent of the sheet volume consists of parallel *myocytes*, more commonly known as muscle fibers [14]. The remaining volume consists of interstitial components. The fiber orientation varies smoothly through the wall thickness, which is shown in Figure 2.2. The inner layer of the heart wall is called *endocardium*, while the outer layer is called *epicardium*. The middle layer is the *myocardium*, which is often used synonymously with 'heart wall', because it is the dominant layer. We can note from this figure that a truncated ellipsoid is used to represent the left ventricle. This is a simple model that is commonly used in literature [14, 26]. An FEM model of the truncated ellipsoid was created and analyzed in this thesis, see Section 3.5.

With respect to the circumferential direction, the predominant muscle fiber direction varies linearly from $+50°$ to $+70°$ in the sub-epicardial region to $-50°$ to $-70°$ in the sub-endocardial region [14]. This means that the direction is nearly $0°$ at the mid-wall region.

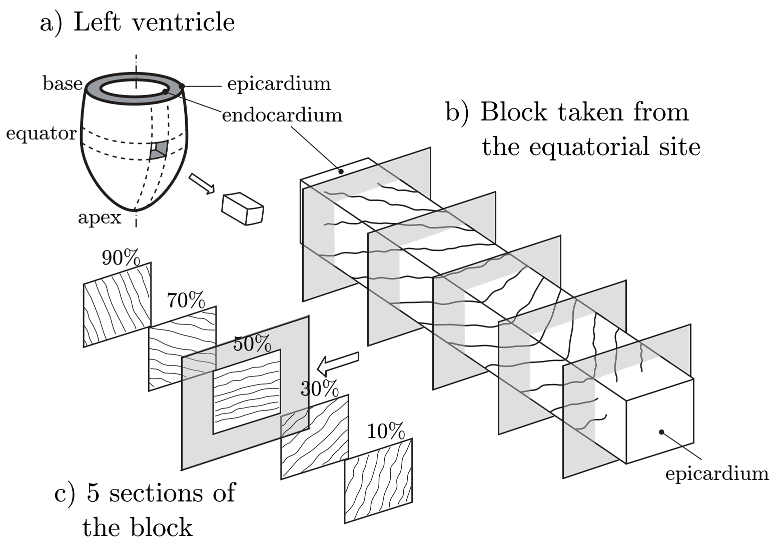


Figure 2.2: *Illustration of: a) a truncated ellipsoid that represents the left ventricle; b) cutout from the equator that shows the structure through the thickness from the epicardium to the endocardium; c) five cross-sections of the block at regular intervals from 10 to 90 per cent of the wall thickness. Observe the transmural variation of fiber orientation through the thickness. The figure is an adapted version of Figure 1 in [14].*

We discuss the structure of the heart wall further when we introduce anisotropic media in succeeding sections.

2.2 Unbounded elastic waves

In this section, the wave equation for waves in linearly elastic material are derived. These waves are more commonly known as *elastic waves.*

An example of a 1D displacement and stress wave is given in Figure 2.3. The relation between displacement and stress waves will be discussed later in this section, but for now, focus will be on the former type of waves.

The wave in Figure 2.3 has a certain velocity c and displacement $u(x,t)$. The differential equation for the wave is

$$c^2 \frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 u}{\partial t^2}, \tag{2.1}$$

which is the well-known one-dimensional wave equation. In [16], Equation (2.1) is

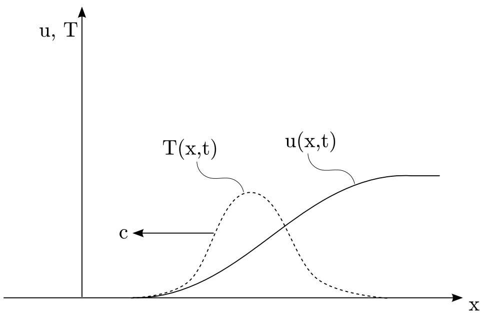


Figure 2.3: *Schematic illustration of a 1D wave in terms of displacement u and corresponding stress T propagating in the negative x-direction with wave speed c.*

derived from looking at longitudinal waves in cylindrical bars. In this case u is the axial displacement and c is given by

$$c = \sqrt{\frac{E}{\rho}},$$

where E is the Young's modulus and ρ is the mass density. This section, however, focuses on bulk wave propagation in two- or three-dimensional infinite (or semi-infinite) media. There are several approaches that give the wave equations we seek. Here, we start by presenting the constitutive relation for generally anisotropic media [23]:

$$T_{ij}(x) = C_{ijkl}(x)\epsilon_{kl}(x), \tag{2.2}$$

where T_{ij} is Cauchy stress, the coefficients in C_{ijkl} are the elastic moduli, which are functions of the position, and ϵ_{kl} is the strain. The Einstein summation convention is implied. Equation (2.2) is called the generalized Hooke's law and should be satisfied at every point in a continuum. The strain can be given by the following linear approximation:

$$\epsilon_{kl} = \frac{1}{2}\Big(\frac{\partial u_k}{\partial x_l} + \frac{\partial u_l}{\partial x_k}\Big). \tag{2.3}$$

Equation (2.3) inserted into Equation (2.2) gives

$$T_{ij} = \frac{1}{2}C_{ijkl}\Big(\frac{\partial u_k}{\partial x_l} + \frac{\partial u_l}{\partial x_k}\Big).$$

The elastic tensor C_{ijkl} has $3^4 = 81$ components, but symmetries of the stress and strain tensors, *i.e.* $T_{ij} = T_{ji}$ and $\epsilon_{kl} = \epsilon_{lk}$, allow us to write

$$C_{ijkl} = C_{jikl} = C_{ijlk},$$

which leaves 36 independent elastic coefficients. These symmetries are usually referred to as the *minor symmetries*. A material is said to be *hyperelastic* if there exists a strain-energy function $U_0 = U_0(\epsilon)$ such that

$$T_{ij} = \frac{\partial U_0}{\partial \epsilon_{ij}}.$$

The above equation in combination with Equation (2.2) gives

$$\frac{\partial T_{ij}}{\partial \epsilon_{kl}} = C_{ijkl} = \frac{\partial^2 U_0}{\partial \epsilon_{ij} \partial \epsilon_{kl}} = \frac{\partial^2 U_0}{\partial \epsilon_{kl} \partial \epsilon_{ij}} = C_{klij},$$

where we have used that the order of differentiation is interchangeable. The symmetry $C_{ijkl} = C_{klij}$ is called the *major symmetry*, and reduces the number of independent elastic coefficients from 36 to 21. To continue the derivation of the wave equation, we exploit the symmetry of C_{ijkl} and write

$$T_{ij} = \frac{1}{2} C_{ijkl} \Big(\frac{\partial u_k}{\partial x_l} + \frac{\partial u_l}{\partial x_k} \Big) = C_{ijkl} \frac{\partial u_k}{\partial x_l}. \tag{2.4}$$

This relation can be verified by simply writing out each term in the summations. The equations of motion for a continuum without body forces can be written in the form [23]

$$\frac{\partial T_{ij}}{\partial x_j} = \rho \frac{\partial^2 u_i}{\partial t^2}. \tag{2.5}$$

The divergence of the stress tensor, $\partial T_{ij}/\partial x_j$, can be found from Equation (2.4):

$$\frac{\partial T_{ij}}{\partial x_j} = \frac{\partial}{\partial x_j} \Big(C_{ijkl} \frac{\partial u_k}{\partial x_l} \Big) = \frac{\partial C_{ijkl}}{\partial x_j} \frac{\partial u_k}{\partial x_l} + C_{ijkl} \frac{\partial^2 u_k}{\partial x_j \partial x_l}.$$

Substitution of the expression above into Equation (2.5) gives the governing equation for wave propagation in inhomogeneous elastic media:

$$\frac{\partial C_{ijkl}}{\partial x_j} \frac{\partial u_k}{\partial x_l} + C_{ijkl} \frac{\partial^2 u_k}{\partial x_j \partial x_l} = \rho \frac{\partial^2 u_i}{\partial t^2}. \tag{2.6}$$

This equation can be simplified by the assumption of homogeneous material. Note that the material can still be anisotropic. C_{ijkl} is then independent of the position within the body, which leads us to the governing equation for waves in homogeneous elastic media:

$$C_{ijkl}\frac{\partial^2 u_k}{\partial x_j \partial x_l} = \rho \frac{\partial^2 u_i}{\partial t^2}. \tag{2.7}$$

The myocardium is inhomogeneous [14], but in the following paragraphs we assume homogeneity, because this obviously simplifies the problem. An important difference between Equation (2.6) and (2.7), is that the former gives dispersive solutions, while the latter does not [23]. A nondispersive solution is a solution which is not dependent on wave length or frequency. Note, however, that the discretization used, *i.e.* FEM and central difference, gives rise to what we call *numerical dispersion* [12, 19, 20].

Due to the symmetry of T_{ij} and ϵ_{kl}, and the corresponding symmetries of C_{ijkl}, it is possible to express the generalized Hooke's law in matrix form, *i.e.*

$$\{\mathbf{T}\} = [\mathbf{C}]\{\epsilon\},$$

where

$$\{\mathbf{T}\} = \begin{Bmatrix} T_{11} \\ T_{22} \\ T_{33} \\ T_{23} \\ T_{31} \\ T_{12} \end{Bmatrix}, \quad \{\epsilon\} = \begin{Bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{33} \\ 2\epsilon_{23} \\ 2\epsilon_{31} \\ 2\epsilon_{12} \end{Bmatrix},$$

$$[\mathbf{C}] = \begin{bmatrix} C_{1111} & C_{1122} & C_{1133} & C_{1123} & C_{1131} & C_{1112} \\ & C_{2222} & C_{2233} & C_{2223} & C_{2231} & C_{2212} \\ & & C_{3333} & C_{2333} & C_{3133} & C_{1233} \\ & & & C_{2323} & C_{2331} & C_{1223} \\ & sym. & & & C_{3131} & C_{1231} \\ & & & & & C_{1212} \end{bmatrix}. \tag{2.8}$$

Note that it is the minor symmetries that allow us to write the coefficients of C_{ijkl} in a 6-by-6 matrix $[\mathbf{C}]$ and it is the major symmetry that makes $[\mathbf{C}]$ symmetric. In the next subsection, we see how the form of $[\mathbf{C}]$ is determined by the assumption of isotropy, and how that leads to the wave equations in isotropic media.

2.2.1 Elastic waves in isotropic media

In this subsection, we derive the wave equations for isotropic media, but first we have to define C_{ijkl} in this type of media. An isotropic material has no preferred direction, which means that the elastic constants must be the same for an arbitrary rotation of the Cartesian coordinate system, *i.e.*

$$C^*_{ijkl} = C_{ijkl},$$

where C^*_{ijkl} and C_{ijkl} are the elastic tensors in coordinate systems with basis $(\mathbf{e}^*_i)$ and $(\mathbf{e}_i)$, respectively. Introducing the most general isotropic 4^{th} order tensor with minor and major symmetry [2]:

$$C_{ijkl} = \lambda\delta_{ij}\delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}), \tag{2.9}$$

where λ and μ are the Lamè elastic constants and δ_{ij} is the Kronecker delta. The relation between the Lamè constants and the engineering constants can be shown to be

$$\mu = G = \frac{E}{2(1+\nu)} \quad \text{and} \quad \lambda = \frac{\nu E}{(1+\nu)(1-2\nu)},$$

where G is the shear modulus, ν is the Poisson's ratio and E is the Young's modulus [15].

By inspection of the general stiffness matrix given in Equation (2.8), it can be seen that Equation (2.9) is a compact way of writing

$$[\mathbf{C}] = \begin{bmatrix} \lambda+2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix}.$$

As expected for isotropic elasticity, $[\mathbf{C}]$ leads to uncoupled shear and longitudinal terms. Insertion of Equation (2.9) into (2.7) gives

$$\begin{aligned} \left[\lambda\delta_{ij}\delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk})\right]\frac{\partial^2 u_k}{\partial x_j \partial x_l} &= \rho\frac{\partial^2 u_i}{\partial t^2}, \\ \Rightarrow \lambda\frac{\partial^2 u_k}{\partial x_i \partial x_k} + \mu\frac{\partial^2 u_i}{\partial x_j^2} + \mu\frac{\partial^2 u_k}{\partial x_k \partial x_i} &= \rho\frac{\partial^2 u_i}{\partial t^2}, \\ \Rightarrow (\lambda+\mu)\frac{\partial^2 u_k}{\partial x_k \partial x_i} + \mu\frac{\partial^2 u_i}{\partial x_j^2} &= \rho\frac{\partial^2 u_i}{\partial t^2}, \end{aligned}$$

which is commonly known as the *Navier equations of motion* (in the absence of body forces). Written in vector notation, the Navier equations become

$$(\lambda+\mu)\nabla\nabla\cdot\{\mathbf{u}\}+\mu\nabla^2\{\mathbf{u}\}=\rho\frac{\partial^2\{\mathbf{u}\}}{\partial t^2}, \tag{2.10}$$

where ∇ is the Del operator. To obtain the sought-after wave equations, we have to simplify the Navier equations, and we do that by the introduction of Helmholtz's decomposition theorem. The theorem implies that a vector field can be considered to be generated by a pair of potentials: a scalar potential Φ and a vector potential $\{\mathbf{H}\}$ [23], *e.g.*

$$\{\mathbf{u}\}=\nabla\Phi+\nabla\times\{\mathbf{H}\}, \quad \text{where} \quad \nabla\cdot\{\mathbf{H}\}=0. \tag{2.11}$$

In other words, $\{\mathbf{u}\}$ is decomposed into a curl-free component and a divergence-free component. Insertion of the decomposition into Equation (2.10), we obtain

$$(\lambda+\mu)\nabla\nabla\cdot(\nabla\Phi+\nabla\times\{\mathbf{H}\})+\mu\nabla^2(\nabla\Phi+\nabla\times\{\mathbf{H}\})=\rho\left(\nabla\frac{\partial^2\Phi}{\partial t^2}+\nabla\times\frac{\partial^2\{\mathbf{H}\}}{\partial t^2}\right).$$

Further, by introduction of the vector identity

$$\nabla^2\{\mathbf{u}\}=\nabla\nabla\cdot\{\mathbf{u}\}-\nabla\times\nabla\times\{\mathbf{u}\},$$

we find

$$\begin{aligned}\left[(\lambda+2\mu)\nabla\nabla\cdot(\nabla\Phi)-\rho\nabla\frac{\partial^2\Phi}{\partial t^2}\right]-\mu\nabla\times\nabla\times\nabla\Phi\\ +(\lambda+\mu)\nabla\nabla\cdot\nabla\times\{\mathbf{H}\}+\left[\mu\nabla^2\nabla\times\{\mathbf{H}\}-\nabla\times\frac{\partial^2\{\mathbf{H}\}}{\partial t^2}\right]=0.\end{aligned}$$

The above expression can be simplified by the following identities:

$$\underbrace{\nabla\cdot\nabla\Phi=\nabla^2\Phi}_{\substack{\text{Divergence of gradient of }\Phi\\ \text{equals Laplacian of }\Phi}}\quad,\quad\underbrace{\nabla\times\nabla\Phi=0}_{\substack{\text{Curl of gradient of }\Phi\\ \text{equals zero}}}\quad\text{and}\quad\underbrace{\nabla\cdot\nabla\times\{\mathbf{H}\}=0}_{\substack{\text{Divergence of curl of }\{\mathbf{H}\}\\ \text{equals zero}}}\;,$$

which leads to

$$\nabla\left[(\lambda+2\mu)\nabla^2\Phi-\rho\frac{\partial^2\Phi}{\partial t^2}\right]+\nabla\times\left[\mu\nabla^2\{\mathbf{H}\}-\frac{\partial^2\{\mathbf{H}\}}{\partial t^2}\right]=0. \tag{2.12}$$

For Equation (2.12) to be generally satisfied, both terms must vanish. Thus, we end up with two uncoupled wave equations:

$$\nabla^2\Phi = \frac{1}{c_l^2}\frac{\partial^2\Phi}{\partial t^2}, \tag{2.13}$$

$$\nabla^2\{\mathbf{H}\} = \frac{1}{c_t^2}\frac{\partial^2\{\mathbf{H}\}}{\partial t^2}, \tag{2.14}$$

where

$$c_l^2 = \frac{\lambda + 2\mu}{\rho} \quad \text{and} \quad c_t^2 = \frac{\mu}{\rho}.$$

The subscripts of the wave speeds, c_l and c_t, will become apparent later. For the case of a purely *dilatational* disturbance, which means $\nabla \times \{\mathbf{H}\} = 0$, the displacement field can be expressed as (see Equation (2.11))

$$\{\mathbf{u}_l\} = \nabla\Phi.$$

It is then easy to show that the wave equation (2.13) becomes

$$\nabla^2\{\mathbf{u}_l\} = \frac{1}{c_l^2}\frac{\partial^2\{\mathbf{u}_l\}}{\partial t^2}, \quad \text{where} \quad c_l^2 = \frac{\lambda + 2\mu}{\rho} = \frac{(1-\nu)E}{\rho(1-2\nu)(1+\nu)}, \tag{2.15}$$

which means that the dilatational disturbance propagates with the velocity c_l. Similarly, a displacement field with only a rotational part can be expressed as

$$\{\mathbf{u}_t\} = \nabla \times \{\mathbf{H}\}, \quad \text{where} \quad \nabla \cdot \{\mathbf{H}\} = 0.$$

Wave equation (2.14) then becomes

$$\nabla^2\{\mathbf{u}_t\} = \frac{1}{c_t^2}\frac{\partial^2\{\mathbf{u}_t\}}{\partial t^2}, \quad \text{where} \quad c_t^2 = \frac{\mu}{\rho} = \frac{E}{2\rho(1+\nu)}, \tag{2.16}$$

which says that rotational waves propagate with velocity c_t. It is apparent from Equation (2.15) and (2.16) that the dilatational and rotational waves are independent and propagate without interaction in unbounded media. The ratio between the two wave speeds can be expressed as

$$\frac{c_t}{c_l} = \frac{\sqrt{\frac{E}{2\rho(1+\nu)}}}{\sqrt{\frac{(1-\nu)E}{\rho(1-2\nu)(1+\nu)}}} = \sqrt{\frac{1-2\nu}{2(1-\nu)}},$$

which expresses that c_t/c_l is only dependent on the Poisson's ratio ν. In Figure 2.4 c_t/c_l is plotted for typical values of Poisson's ratio used in literature for biological tissues. We see from this figure that as $\nu \to 0.5$, then $c_t/c_l \to 0$, because $c_l \to \infty$. Hence, the limiting case of $\nu = 0.5$ should probably be avoided in FEM models.

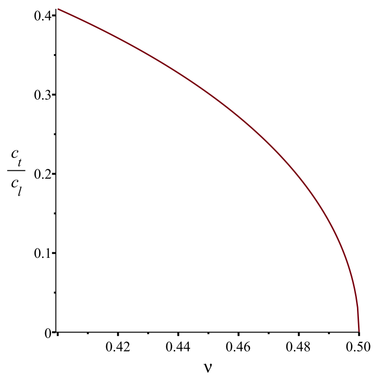
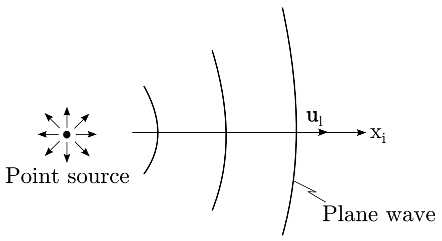
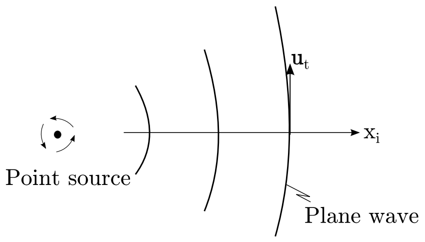


Figure 2.4: *Plot of the ratio c_t/c_l. We see that the ratio approaches zero for increase of ν. Note that $c_t < c_l$ for $0 < \nu < 0.5$.*

Figure 2.5 illustrates how dilatational and rotational waves develop from a point source. Figure 2.5a displays how the material displacement is parallel to the wave propagation direction for the case of a dilatational wave, and is thereby also called longitudinal wave. The rotational wave in Figure 2.5b has material displacement orthogonal to the wave propagation direction, and is therefore also called transverse wave. The figures also indicate that the waves can be considered *plane* when sufficiently distanced from the point source.



(a) *A point source that generates spherical longitudinal waves. The material displacement is parallel to the wave propagation direction.*



(b) *A point source that generates spherical transverse waves. The material displacement is orthogonal to the wave propagation direction.*

Figure 2.5: *Plane wave that develops in far field of the point source. The figure is inspired by Figure 7.7.5 in [16].*

Until now, only waves in terms of displacement have been discussed. It is interesting to find the relation between stresses and displacements. By insertion of

Equation (2.9) into the second expression of Equation (2.4), the relation is obtained:

$$T_{ij} = \lambda \frac{\partial u_k}{\partial x_k} \delta_{ij} + \mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right), \tag{2.17}$$

which is a form of Hooke's law for isotropic elastic materials. As an example, let us find the stresses when a plane *longitudinal* wave propagates in direction x_1. Then the displacement vector has the form

$$\{\mathbf{u}\} = [u_1(x_1, t), 0, 0]^T,$$

and the stresses can be found from Equation (2.17):

$$\begin{gathered} T_{11} = (\lambda + 2\mu) u_{1,1}, \quad T_{22} = T_{33} = \lambda u_{1,1}, \\ T_{12} = T_{13} = T_{23} = 0. \end{gathered}$$

Note that a longitudinal wave does not generate shear stresses. Note also that as long as μ differs from zero, longitudinal waves do not give rise to isotropic pressure. They are therefore not pressure waves as we know from fluid dynamics. An example of a plane *transverse* wave could be a wave that propagates in the x_1-direction with a displacement in the x_2 direction, which gives the following displacement vector:

$$\{\mathbf{u}\} = [0, u_2(x_1, t), 0]^T,$$

and the corresponding stresses are:

$$\begin{gathered} T_{11} = T_{22} = T_{33} = 0, \\ T_{12} = \mu u_{2,1}, \quad T_{13} = T_{23} = 0. \end{gathered}$$

In this case, only one shear stress component is unequal to zero. Transverse waves are therefore also called *shear waves.* Essentially, the stresses are proportional to the derivative of the displacements, which has already been visualized in Figure 2.3.

2.2.2 Elastic waves in anisotropic media

An elastic and isotropic material has a strain energy function and stiffness tensor C_{ijkl}, which are independent of orientation. If a material is not isotropic, then it is anisotropic. We have already seen that a general anisotropic material has 21 independent coefficients, but by the assumption of certain symmetries in the

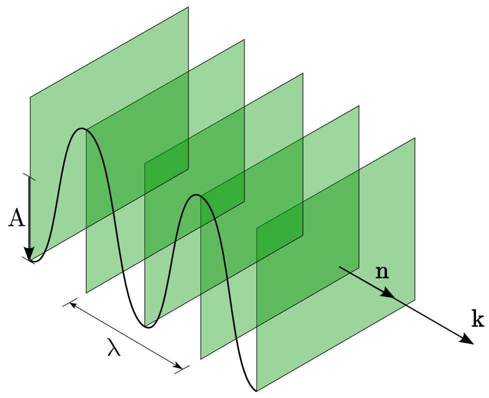


Figure 2.6: *Illustration of wavefronts of a plane wave that travels in 3D-space. The wave has a wavelength λ, constant amplitude A and propagates in the direction of the wave vector $\{\mathbf{k}\}$.*

material, the number is reduced significantly. Before we introduce symmetries of anisotropic materials, let us derive a general expression that allows for computation of the wave speeds and material displacements in anisotropic materials.

Assume a solution of the plane time-harmonic wave form

$$u_i = A_i cos(k_j x_j - \omega t), \tag{2.18}$$

where A_i is the amplitude, x_j is the position vector, k_j is the wave vector, ω is the frequency and t is the time. The wave vector $\{\mathbf{k}\}$ gives the direction of wave propagation, and differs only from the wave number k in that it has a direction as well as a magnitude. Thus, $||\{\mathbf{k}\}|| = k = 2\pi/\lambda$, where λ is the wave length. The wave fronts of a plane wave are surfaces of constant phase. These surfaces are infinite parallel planes with normal along the wave vector, see Figure 2.6.

By insertion of Equation (2.18) into (2.7), we obtain the well-known *Christoffel's equation*:

$$(C_{ijkl} k_j k_l - \rho\omega^2 \delta_{ik}) A_k = 0. \tag{2.19}$$

In the end we would like an expression for the wave speeds. We will therefore edit Christoffel's equation such that it contains wave speeds rather than frequencies. We express the wave vector $\{\mathbf{k}\}$ by the wave number k and its direction cosine $\{\mathbf{n}\}$:

$$k_j = kn_j, \quad k_l = kn_l. \tag{2.20}$$

The relation between wave number and wave speed is given by [23]

$$k = \frac{\omega}{c} \quad \Rightarrow \quad c = \frac{\omega}{k}. \tag{2.21}$$

By insertion of Equation (2.20) into (2.19) we therefore obtain

$$(C_{ijkl} n_j n_l - \rho c^2 \delta_{ik}) A_k = 0. \tag{2.22}$$

The expression above can be simplified further by the introduction of the acoustical tensor [23]

$$\Gamma_{ik} \equiv C_{ijkl} n_j n_l,$$

which allow us to write Equation (2.22) as

$$\boxed{(\Gamma_{ik} - \rho c^2 \delta_{ik}) A_k = 0.} \tag{2.23}$$

Equation (2.23) is an alternative form of Christoffel's equation, and it expresses an eigenvalue problem where ρc^2 are the eigenvalues and A_k are the eigenvectors. Since C_{ijkl} is symmetric, the acoustical tensor is symmetric as well, and it has therefore generally three distinct eigenvalues with corresponding orthogonal eigenvectors. The eigenvectors $\{\mathbf{A}\}$ express the material displacement of a plane wave that propagates in direction $\{\mathbf{n}\}$ and c is the corresponding wave speed, see Figure 2.7. Similarly as for dynamics, where we have vibration modes, we can term the pairs of wave speeds and material displacements as *wave modes*. The mode with material displacement $\{\mathbf{A}\}$ closest to $\{\mathbf{n}\}$ is termed quasi-longitudinal, and will usually have the largest wave speed. The two other modes are called quasi-transverse. It is important to understand that even though the material *displaces* in direction $\{\mathbf{A}\}$, the plane wave *propagates* in direction $\{\mathbf{n}\}$. Material displacement is physically interpreted and discussed further in Section (2.4).

Recall that in an isotropic medium we have two wave modes, *i.e.* longitudinal and transverse wave mode, that propagate independently of each other. The two wave modes have different material displacement vectors and wave speeds. Comparably, in an anisotropic medium, generally three wave modes can propagate in direction $\{\mathbf{n}\}$, each with its characteristic material displacement vector $\{\mathbf{A}\}$ and wave speed c. Note that, contrary to the isotropic case, the wave speeds depend on the propagation direction $\{\mathbf{n}\}$.

It is convenient to introduce a reduced notation for the indices: 11, 22, 33, 23, 13 and 12 are replaced by the single indices 1, 2, 3, 4, 5, and 6, respectively. This means that we can write the generalized Hooke's law as

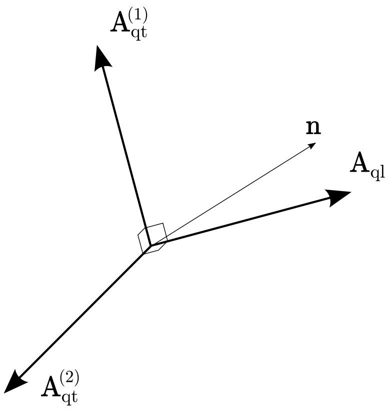

$\mathbf{A}_{\mathrm{qt}}^{(1)}$
$\mathbf{n}$
$\mathbf{A}_{\mathrm{ql}}$
$\mathbf{A}_{\mathrm{qt}}^{(2)}$

Figure 2.7: *Generally, three plane waves can propagate in the same direction* $\{\mathbf{n}\}$ *in an anisotropic medium, each with its characteristic velocity obtained from the eigenvalue problem. The corresponding eigenvectors* $\{\mathbf{A}\}$ *express the material displacement (polarization) of the wave and are mutually orthogonal. The subscripts 'ql' and 'qt' abbreviate quasi-longitudinal and quasi-transverse, respectively.*

$$\begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \end{Bmatrix} = \begin{bmatrix} C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\ & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\ & & C_{33} & C_{34} & C_{35} & C_{36} \\ & & & C_{44} & C_{45} & C_{46} \\ & sym. & & & C_{55} & C_{56} \\ & & & & & C_{66} \end{bmatrix} \begin{Bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ 2\epsilon_4 \\ 2\epsilon_5 \\ 2\epsilon_6 \end{Bmatrix}. \tag{2.24}$$

Γ_{ik} has only six independent components, because it is symmetric. With the reduced notation, the components of Γ_{ik} can be written as

$$\begin{aligned}
\Gamma_{11} &= C_{11}n_1^2 + C_{66}n_2^2 + C_{55}n_3^2 + 2C_{16}n_1n_2 + 2C_{56}n_2n_3 + 2C_{15}n_1n_3, \\
\Gamma_{22} &= C_{66}n_1^2 + C_{22}n_2^2 + C_{44}n_3^2 + 2C_{26}n_1n_2 + 2C_{24}n_2n_3 + 2C_{46}n_1n_3, \\
\Gamma_{33} &= C_{55}n_1^2 + C_{44}n_2^2 + C_{33}n_3^2 + 2C_{45}n_1n_2 + 2C_{34}n_2n_3 + 2C_{35}n_1n_3, \\
\Gamma_{12} &= C_{16}n_1^2 + C_{26}n_2^2 + C_{45}n_3^2 \\
&\qquad + (C_{12} + C_{66})n_1n_2 + (C_{25} + C_{46})n_2n_3 + (C_{14} + C_{56})n_1n_3, \\
\Gamma_{13} &= C_{15}n_1^2 + C_{46}n_2^2 + C_{35}n_3^2 \\
&\qquad + (C_{14} + C_{56})n_1n_2 + (C_{36} + C_{45})n_2n_3 + (C_{13} + C_{55})n_1n_3, \\
\Gamma_{23} &= C_{56}n_1^2 + C_{24}n_2^2 + C_{34}n_3^2 \\
&\qquad + (C_{25} + C_{46})n_1n_2 + (C_{23} + C_{44})n_2n_3 + (C_{36} + C_{45})n_1n_3.
\end{aligned} \tag{2.25}$$

In Section 2.4, Equation (2.25) is used when we solve Christoffel's equation (2.23) for transversely isotropic materials. Before we can do that, we need to define transverse isotropy. This is done in the next section, where we explain symmetries in anisotropic materials.

2.3 Symmetries in anisotropic materials

The symmetries in anisotropic materials can be explained by planes of material symmetry. Consider the plane P with normal $\{\mathbf{e}\}_1$ in Figure 2.8. A reflection with respect P can be described by the transformation

$$\{\mathbf{e}\}_1^* = -\{\mathbf{e}\}_1, \quad \{\mathbf{e}\}_2^* = \{\mathbf{e}\}_2, \quad \{\mathbf{e}\}_3^* = \{\mathbf{e}\}_3,$$

which more compactly can be written

$$\{\mathbf{e}\}_i^* = a_{ij}\{\mathbf{e}\}_j, \quad [\mathbf{a}] = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The plane P is said to be plane of material symmetry if the components of C_{ijkl}, expressed in the basis $(\{\mathbf{e}\}_1, \{\mathbf{e}\}_2, \{\mathbf{e}\}_3)$, remains unchanged when transformed by $[\mathbf{a}]$. The transformation rule for a 4^{th} order tensor is [15]

$$C_{ijkl}^* = a_{im}a_{jn}a_{kr}a_{ls}C_{mnrs}.$$

Thus, if $[\mathbf{a}]$ is defined by a plane of symmetry, the transformation becomes

$$C_{ijkl} = a_{im}a_{jn}a_{kr}a_{ls}C_{mnrs}. \tag{2.26}$$

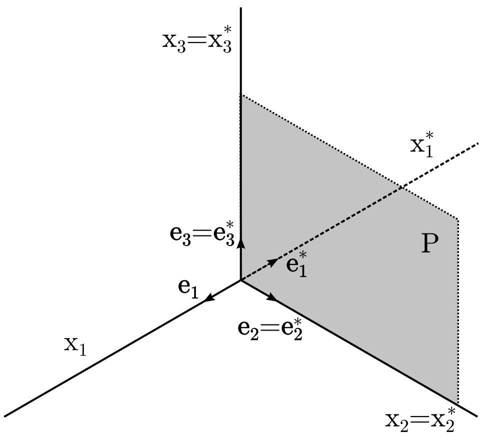


Figure 2.8: *P is a symmetry plane if the components of C_{ijkl} are invariant to the transformation from the un-starred to the starred set of base vectors. The figure is inspired by Figure 5-1 in [15].*

2.3.1 Orthotropic symmetry

A material is said to be orthotropic if there are three mutually orthogonal planes of symmetry, as illustrated in Figure 2.9. With the help of the restriction given by Equation (2.26), it can be shown that the elasticity tensor C_{ijkl} becomes

$$[\mathbf{C}] = \begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ C_{12} & C_{22} & C_{23} & 0 & 0 & 0 \\ C_{13} & C_{23} & C_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{66} \end{bmatrix},$$

for the case of orthotropic elasticity [15]. Now the number of independent elastic coefficients is reduced to 9. It should be noted that if one works with a coordinate system where the coordinate planes are not parallel to the planes of material symmetry, there will be more than 9 elastic coefficients in [**C**]. However, there will still be 9 *independent* elastic coefficients. Wood is an example of an orthotropic material, because it has different properties in three orthogonal directions, *e.g.* in axial, radial and circumferential direction.

According to [14], the myocardium has been modeled both as orthotropic and transversely isotropic in literature. In this thesis, we work with a transversely isotropic material, because elastic coefficients for the myocardium for that case could be found in literature.

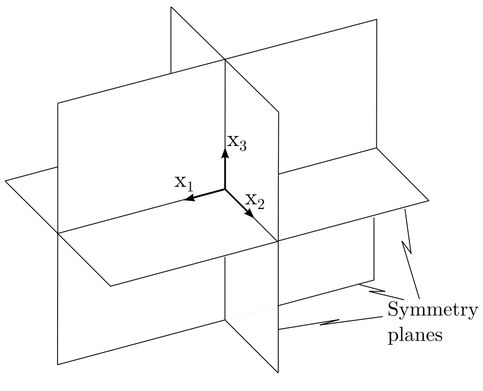


Figure 2.9: *An orthotropic material has three mutually orthogonal planes of symmetry. In this figure the coordinate planes are aligned with the symmetry planes, but this is not necessarily the case.*

2.3.2 Transversely isotropic symmetry

A material is transversely isotropic if through every particle there exists a symmetry axis for the material properties. A symmetry axis implies that every plane through such an axis is a plane of symmetry, see Figure 2.10. Thus, a plane *transverse* to the symmetry axis is a plane of isotropy. Hence, the name transverse isotropy is commonly used for this type of anisotropy. It can be noted that transverse isotropy implies orthotropy, but not the other way around.

Typically, fibrous materials are transversely isotropic. An example is skeletal muscles, where the direction of the fibers would be the symmetry axis. The myocardium is also fibrous, and is therefore often modeled as transversely isotropic [14]. It can be shown that this type of material has five independent coefficients in the stiffness matrix [16]:

$$[\mathbf{C}] = \begin{bmatrix} C_{11} & C_{12} & C_{13} & 0 & 0 & 0 \\ C_{12} & C_{11} & C_{13} & 0 & 0 & 0 \\ C_{13} & C_{13} & C_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{44} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2}(C_{11} - C_{12}) \end{bmatrix}.$$

Similarly as for the orthotropic case, the number of nonzero elements in this matrix depends on the orientation of the coordinate system. If one of the coordinate axes is aligned with the symmetry axis, the shape of the matrix is as shown above.

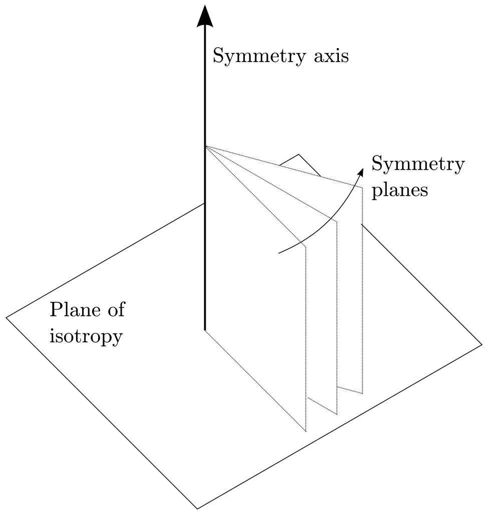


Figure 2.10: *A transversely isotropic material has a symmetry axis. Infinitely many symmetry planes go through that axis. A plane with its normal parallel to the symmetry axis is isotropic.*

2.4 Solving Christoffel's equation

With knowledge of the stiffness matrix [**C**], we can find expressions for the wave speeds from Christoffel's equation (2.23), which is restated here:

$$(\Gamma_{ik} - \rho c^2 \delta_{ik}) A_k = 0.$$

Before we solve the eigenvalue problem for a transversely isotropic medium, we briefly consider the limiting case of isotropy. Thus, we insert the stiffness matrix [**C**] from Equation (2.9). The wave propagation direction is defined in Figure 2.11.

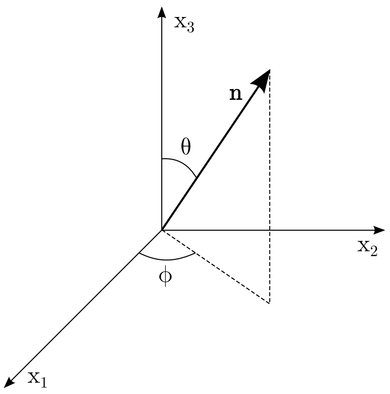


Figure 2.11: *The propagation direction* $\{\mathbf{n}\}$ *is specified by the polar angles* θ *and* ϕ*. The form of* $\{\mathbf{n}\}$ *is in general:* $\{sin\theta cos\phi, sin\theta sin\phi, cos\theta\}^T$*.*

With the help of the expressions in Equation (2.25), we can form the acoustic tensor Γ_{ik}:

$$[\mathbf{\Gamma}] = \begin{bmatrix} (\lambda+2\mu)n_1^2+\mu n_2^2+\mu n_3^2 & (\lambda+\mu)n_1n_2 & (\lambda+\mu)n_1n_3 \\ (\lambda+\mu)n_1n_2 & \mu n_1^2+(\lambda+2\mu)n_2^2+\mu n_3^2 & (\lambda+\mu)n_2n_3 \\ (\lambda+\mu)n_1n_3 & (\lambda+\mu)n_2n_3 & \mu n_1^2+\mu n_2^2+(\lambda+2\mu)n_3^2 \end{bmatrix},$$

$$n_1 = sin\theta cos\phi, \quad n_2 = sin\theta sin\phi, \quad n_3 = cos\theta.$$

Note the symmetry of [**Γ**]. A nontrivial solution to the eigenvalue problem requires that

$$\begin{vmatrix} (\lambda+2\mu)n_1^2+\mu n_2^2+\mu n_3^2-\rho c^2 & (\lambda+\mu)n_1n_2 & (\lambda+\mu)n_1n_3 \\ (\lambda+\mu)n_1n_2 & \mu n_1^2+(\lambda+2\mu)n_2^2+\mu n_3^2-\rho c^2 & (\lambda+\mu)n_2n_3 \\ (\lambda+\mu)n_1n_3 & (\lambda+\mu)n_2n_3 & \mu n_1^2+\mu n_2^2+(\lambda+2\mu)n_3^2-\rho c^2 \end{vmatrix} = 0.$$

We obtain three eigenvalues from this problem:

$$\rho c_1^2 = (\lambda + 2\mu) \quad \text{and} \quad \rho c_2^2 = \rho c_3^2 = \mu.$$

We see that there are only two *distinct* eigenvalues, and that we obtained exactly the same wave speeds as in Equation (2.15) and (2.16). In other words, $c_1 = c_l$ and $c_2 = c_3 = c_t$. Note that the eigenvalues are independent of the propagation direction. Also, be aware of that we have assumed plane waves to obtain the result above. This limitation is not present in Section 2.2.1, where we derived the same wave speeds. There we assumed that the displacement field can be decomposed into a scalar potential and a vector potential.

Let us now consider a transversely isotropic material with symmetry axis along the x_3 direction. We then know that the wave speeds do not vary in the x_1-x_2 plane, because it is a plane of isotropy. It is therefore more interesting to find expressions for the wave speeds in the x_1-x_3 plane. The propagation direction $\{\mathbf{n}\}$ has two nonzero components in this plane; $\{\mathbf{n}\} = \{sin(\theta), 0, cos(\theta)\}^T$. With the same procedure as for the isotropic case, we obtain the following eigenvalue problem on matrix form:

$$\begin{vmatrix} C_{11}n_1^2 + C_{44}n_3^2 - \rho c^2 & 0 & (C_{13}+C_{44})n_1n_3 \\ 0 & \frac{1}{2}(C_{11}-C_{12})n_1^2 + C_{44}n_3^2 - \rho c^2 & 0 \\ (C_{13}+C_{44})n_1n_3 & 0 & C_{44}n_1^2 + C_{33}n_3^2 - \rho c^2 \end{vmatrix} = 0.$$

$$n_1 = sin\theta, \quad n_3 = cos\theta.$$

Before we find the solutions to the eigenvalue problem, let us define the following variable:

$$\begin{aligned} B \quad \equiv \quad & \big(C_{11} - C_{44}\big)^2 + \\ & \big(-2C_{11}^2 - 2C_{11}C_{33} + 6C_{11}C_{44} + 2C_{33}C_{44} + 4C_{13}^2 + 8C_{13}C_{44}\big)cos^2\theta \\ & +\big(C_{11}^2 + 2C_{11}C_{33} + C_{33}^2 - 4C_{11}C_{44} - 4C_{33}C_{44} - 4C_{13}^2 - 8C_{13}C_{44}\big)cos^4\theta. \end{aligned}$$

This allows us to write the solutions to the eigenvalue problem in a compact form. There are three solutions, *i.e.* eigenvalues, to the problem:

$$\rho c_1^2 = \frac{1}{2}\big(C_{11}sin^2\theta + C_{33}cos^2\theta + C_{44} + \sqrt{B}\big), \tag{2.27}$$

$$\rho c_2^2 = \frac{1}{2}\big(C_{11}sin^2\theta + C_{33}cos^2\theta + C_{44} - \sqrt{B}\big), \tag{2.28}$$

$$\rho c_3^2 = C_{44}cos^2\theta + \frac{1}{2}\big(C_{11} - C_{12}\big)sin^2\theta. \tag{2.29}$$

Note that C_{12}, which is the stiffness coefficient that couples T_{11} with ϵ_{22} and T_{22} with ϵ_{11}, is absent in the expressions for c_1 and c_2. On the other hand, c_3 is independent of the stiffness coefficient C_{13}, which couples T_{11} and T_{22} with ϵ_{33}, and T_{33} with ϵ_{11} and ϵ_{22}. It is also independent of C_{33}, that couples T_{33} with ϵ_{33}. We can also find the corresponding normalized eigenvectors:

$$\{\mathbf{A}\}_1 = \left\{ \begin{matrix} \dfrac{2(C_{13}+C_{44})sin\theta cos\theta}{\left(\left(-C_{11}sin^2\theta+C_{33}cos^2\theta+C_{44}-2C_{44}cos^2\theta+\sqrt{B}\right)^2+\left(2(C_{13}+C_{44})sin\theta cos\theta\right)^2\right)^{1/2}} \\ 0 \\ \dfrac{1}{\left(1+\left(\frac{2(C_{13}+C_{44})sin\theta cos\theta}{-C_{11}sin^2\theta+C_{33}cos^2\theta+C_{44}-2C_{44}cos^2\theta+\sqrt{B}}\right)^2\right)^{1/2}} \end{matrix} \right\},$$

$$\{\mathbf{A}\}_2 = \left\{ \begin{matrix} \dfrac{2(C_{13}+C_{44})sin\theta cos\theta}{\left(\left(-C_{11}sin^2\theta+C_{33}cos^2\theta+C_{44}-2C_{44}cos^2\theta-\sqrt{B}\right)^2+\left(2(C_{13}+C_{44})sin\theta cos\theta\right)^2\right)^{1/2}} \\ 0 \\ \dfrac{1}{\left(1+\left(\frac{2(C_{13}+C_{44})sin\theta cos\theta}{-C_{11}sin^2\theta+C_{33}cos^2\theta+C_{44}-2C_{44}cos^2\theta-\sqrt{B}}\right)^2\right)^{1/2}} \end{matrix} \right\},$$

$$\{\mathbf{A}\}_3 = \left\{ \begin{matrix} 0 \\ 1 \\ 0 \end{matrix} \right\}.$$

Comment: *the derivation of the eigenvalues is inspired by the derivation found in [23]. In this reference, expressions for the eigenvalues for an orthotropic medium are derived. On the other hand, the author has not found general expressions for the eigenvectors, that are similar to those above, in literature. Therefore, to the best of our knowledge, these expressions are novel.*

We can see that $\{\mathbf{A}\}_3$ describes a material displacement (polarization) which is purely out of the x_1-x_3 plane for all values of θ. The dot product

$$\{\mathbf{A}\}_3 \cdot \{\mathbf{n}\} = 0,$$

shows that the polarization is orthogonal to the propagation direction. Thus, the third eigensolution can be defined as a pure transverse wave. In fact, one of the shear wave modes is always pure for any transversely isotropic material [23]. On the other hand, $\{\mathbf{A}\}_1$ and $\{\mathbf{A}\}_2$ have displacement components in the plane. By comparison of the wave speeds c_1 and c_2, we see that the former wave speed is highest and therefore corresponds to the quasi-longitudinal wave. Before we analyze the eigensolutions further, let us assume that $C_{11} > C_{44}$, $C_{11} > C_{66}$, $C_{33} > C_{44}$ and $C_{33} > C_{66}$. These are reasonable assumptions, because the longitudinal stiffness is larger than the shear stiffness for most soft tissues. The assumptions allow us to evaluate our expressions for certain θs, which we do in the following subsections.

2.4.1 Propagation along fibers

If we have a wave that propagates along the x_3-axis, *i.e.* $\theta = 0$, the expressions for the eigenvalues simplify to

$$\rho c_1^2 = \frac{1}{2}\left(C_{33} + C_{44} + \sqrt{(C_{33} - C_{44})^2}\right) = C_{33}, \tag{2.30}$$

$$\rho c_2^2 = \frac{1}{2}\left(C_{33} + C_{44} - \sqrt{(C_{33} - C_{44})^2}\right) = C_{44}, \tag{2.31}$$

$$\rho c_3^2 = C_{44}. \tag{2.32}$$

Note that we have used the assumption $C_{33} > C_{44}$ in the expressions above. Before we analyze the wave speeds, let us find the corresponding eigenvectors. A minor issue is that $\{\mathbf{A}\}_2$ is not defined for $\theta = 0$, because we get $0/0$ in the first component and $1/0$ in the third component of the vector. However, if we take the limit of $\{\mathbf{A}\}_2$ as θ approaches zero, the first component becomes unity and the third component becomes zero. For $\{\mathbf{A}\}_1$, we insert $\theta = 0$ directly into the vector. The result is:

$$\{\mathbf{A}\}_1 = \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix}, \quad \{\mathbf{A}\}_2 = \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix}, \quad \{\mathbf{A}\}_3 = \begin{Bmatrix} 0 \\ 1 \\ 0 \end{Bmatrix}.$$

From Equation 2.30 we see that c_1 is determined only by C_{33}, the longitudinal stiffness in the x_3-direction. Also, the polarization is parallel to $\{\mathbf{n}\}$, *i.e.*

$$\{\mathbf{A}\}_1 \times \{\mathbf{n}\} = 0$$

Thus, for the case $\theta = 0$, the first eigensolution represent a pure longitudinal wave. On the other hand, c_2 and c_3 are determined only by C_{44}, which is the shear stiffness that couples T_{23} with ϵ_{23} and T_{13} with ϵ_{13}. Similarly as for $\{\mathbf{A}\}_3$, the second eigenvector is now also orthogonal to $\{\mathbf{n}\}$, because

$$\{\mathbf{A}\}_2 \cdot \{\mathbf{n}\} = 0.$$

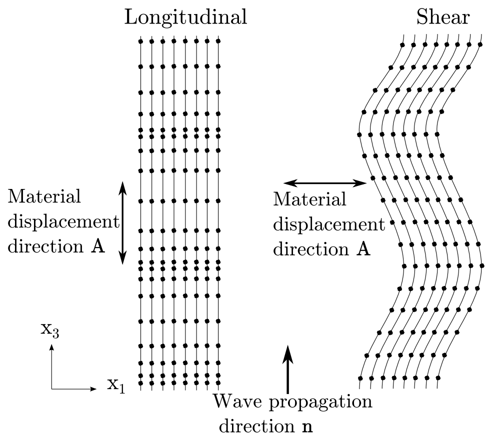

Longitudinal
Shear
Material displacement direction **A**
Material displacement direction **A**
x3
x1
Wave propagation direction **n**

Figure 2.12: *Wave propagation along fibers ($\{\mathbf{n}\} = \{0, 0, 1\}^T$). This figure illustrates only the shear wave with polarization in the plane. The shear wave with polarization out of the plane looks exactly the same. The figure is inspired by Figure 4-40 in [2].*

The second eigensolution is therefore, in this case, also a pure shear wave. Wave propagation along the x_3 axis can be visualized as in Figure 2.12.

2.4.2 Propagation transverse to fibers

Now, let the wave propagate in x_1 direction, which means $\theta = 90°$. The same procedure as in the previous case gives the following eigenvalues:

$$\rho c_1^2 = \frac{1}{2}\left(C_{11} + C_{44} + \sqrt{(C_{11} - C_{44})^2}\right) = C_{11}, \tag{2.33}$$

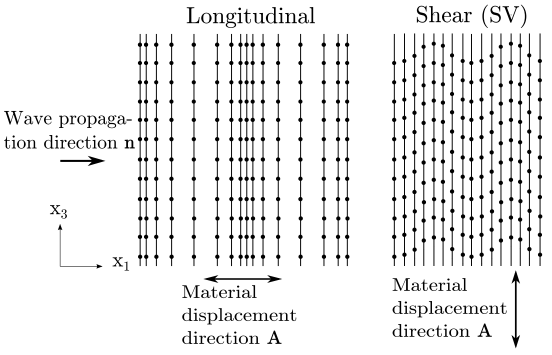
$$\rho c_2^2 = \frac{1}{2}\left(C_{11} + C_{44} - \sqrt{(C_{11} - C_{44})^2}\right) = C_{44}, \tag{2.34}$$

$$\rho c_3^2 = \frac{1}{2}(C_{11} - C_{12}). \tag{2.35}$$

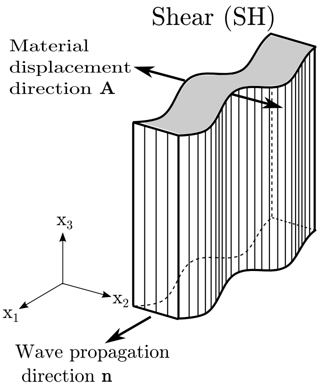
Again, note that we used the assumption $C_{11} > C_{44}$. For the case of $\theta = 90°$, the eigenvector $\{\mathbf{A}\}_2$ is defined, but $\{\mathbf{A}\}_1$ is not defined. The eigenvector $\{\mathbf{A}\}_1$ can be found by taking the limit of $\{\mathbf{A}\}_1$ as θ approaches $\pi/2$. The final result is:

$$\{\mathbf{A}\}_1 = \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix}, \quad \{\mathbf{A}\}_2 = \begin{Bmatrix} 0 \\ 0 \\ 1 \end{Bmatrix}, \quad \{\mathbf{A}\}_3 = \begin{Bmatrix} 0 \\ 1 \\ 0 \end{Bmatrix}.$$

As one might expect, the longitudinal wave speed, $c_1 = c_l$, is determined only by C_{11} and the corresponding polarization $\{\mathbf{A_1}\}$ is parallel to $\{\mathbf{n}\}$. Compared to the


Longitudinal
Shear (SV)
Wave propagation direction n
x3
x1
Material displacement direction A
Material displacement direction A

(a) *Longitudinal wave with 'horizontal' polarization and shear wave with 'vertical' polarization*


Shear (SH)
Material displacement direction A
x3
x1
x2
Wave propagation direction n

(b) *Shear wave with 'horizontal' polarization*

Figure 2.13: *Wave propagating transverse to fibers ($\{\mathbf{n}\} = \{1, 0, 0\}^T$). We define the 'vertical' direction as the fiber direction. The figure is inspired by Figure 4-40 in [2].*

previous case, we now have two distinct shear wave speeds rather than one, see Equation (2.34) and (2.35). This is due that the polarizations $\{\mathbf{A}\}_2$ and $\{\mathbf{A}\}_3$ represent two different mechanisms in the material, see Figure 2.13. In this figure we define vertically and horizontally polarized shear waves as SV and SH waves, respectively. From this figure it becomes apparent why $c_3 (= c_{SH})$ is determined by the shear stiffness which relates T_{12} and ϵ_{12}, namely $C_{66} = \frac{1}{2}(C_{11} - C_{12})$.

Comment: *a way to avoid limits in the calculation of the eigenvectors is to insert the value for θ before we find the eigensolutions from the eigenvalue problem.*

In this section and in Section 2.2.1 we have seen how we can find the relation between wave speeds and material constants, i.e. density and elastic coefficients. From ultrasonic measurements it is possible to obtain the wave speeds *in vivo*. Thus, with knowledge of the wave speeds, the elastic properties of the soft tissue can be determined. However, there are some challenges if this is to be performed. For example, the longitudinal wave speed is too large to be captured with ultrasonic measurements (normally $c_l \sim 1500$ m/s). We therefore need to rely on the observation of shear waves. Another challenge is that we need to obtain several wave speeds to fully determine $[\mathbf{C}]$. Even for the relatively simple case of transverse isotropy, where there are 5 unknowns in $[\mathbf{C}]$, we need at least 5 different shear wave speeds. As an example: Brekke *et al.* has been able to measure *one* shear wave speed that arises in the myocardium due to AVC [6].

We now have wave speeds expressed by θ and the material constants. The next section explains how we can use these expressions to analyze wave propagation in the material.

2.5 Characteristic surfaces

Characteristic surfaces are often used to help describe and predict propagation of elastic waves in anisotropic materials. Three of them are discussed in this section: velocity, slowness and wave surfaces. The velocity and slowness surfaces are relatively easily developed and understood. The wave surface requires more derivation and explanation.

A numerical example is provided throughout this section to help the reader with understanding the concepts. The mass density is set to 1060 kg/m^3, and the material is transversely isotropic with values for the elastic constants given in Table 2.1. The characteristic surfaces in this thesis are plotted by the Maple script in Appendix A.

Table 2.1: *Elastic stiffness coefficients used in the numerical example in this section. Note that C_{33} is significantly larger than the other coefficients.*

	C_{11}	C_{33}	C_{44}	C_{12}	C_{13}
[GPa]	2.46	10.00	$8.97 \cdot 10^{-3}$	2.4431	2.44

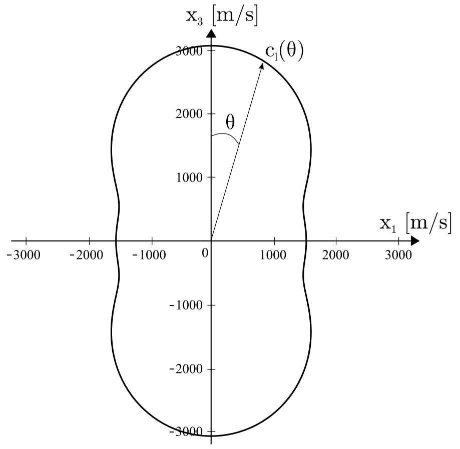
2.5.1 Velocity surfaces

We have seen how it is possible to develop an expression for the wave speed of the plane wave as a function of the direction cosine $\{\mathbf{n}\}$. As a reminder: the direction cosine gives the direction the wave propagates, and is expressed by angles with respect to the coordinate axes, see Figure 2.11. Plots of the wave speed for the quasi-longitudinal wave mode versus angles in the x_1-x_3 and x_1-x_2 plane are given in Figure 2.14. For simplicity, we call the quasi-longitudinal wave for longitudinal wave from now on.

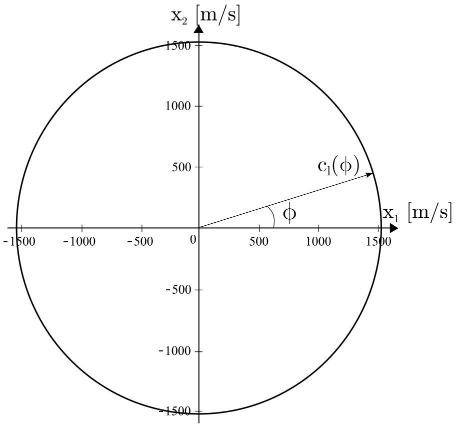
As we saw in Section 2.4, the longitudinal wave speed in the x_1, x_2 and x_3-directions are given by

$$c_l = \sqrt{\frac{C_{11}}{\rho}}, \quad c_l = \sqrt{\frac{C_{11}}{\rho}} \quad \text{and} \quad c_l = \sqrt{\frac{C_{33}}{\rho}},$$

respectively. It is thus obvious that c_l is largest in the x_3-direction since $C_{33} > C_{11}$. We can also see from Figure 2.14 that the velocity surface is perfectly circular in the isotropic x_1-x_2 plane. It is clear that velocity surfaces give a good qualitatively description on how wave speeds differ in various directions of a material.



(a) *Velocity surface in the x_1-x_3 plane. The velocity in x_3 direction is largest, because the material is stiffest in that direction.*



(b) *Velocity surface in the x_1-x_2 plane. Since this plane is isotropic, the velocity surface becomes circular.*

Figure 2.14: *Velocity surfaces of the longitudinal wave in a transversely isotropic material.*

2.5.2 Slowness surfaces

Slowness surfaces are the inverse of velocity surfaces. We can find the slowness surface of our example by plotting $1/c_l(\theta)$ versus θ, similar as for the velocity surface. The plot is given in Figure 2.15. Characteristic surfaces in the isotropic plane, x_1-x_2 plane, are not included in the remainder of this thesis, because they are always circular.

Figure 2.15 displays how the so-called *energy velocity* $\{\mathbf{c}^e\}$ is normal to the slowness surface and has a skew angle ψ with respect to the direction $\{\mathbf{n}\}$. In this text we use slowness surfaces to introduce energy velocity, but the surfaces can also be used to find reflection angles at interfaces [23]. To understand the concept of energy velocity, a derivation of it is given in the next subsection.

2.5.3 Energy velocity

Before we derive the energy velocity, the *Poynting vector* P_i is defined [24]:

$$P_i = -T_{ij}\frac{\partial u_j}{\partial t} \quad [W/m^2]. \tag{2.36}$$

The Poynting vector represents, in magnitude and direction, the instantaneous power per unit area transported by the wave. Or in other words, the rate of energy transfer per unit area. The energy velocity is, by definition, equal to the Poynting

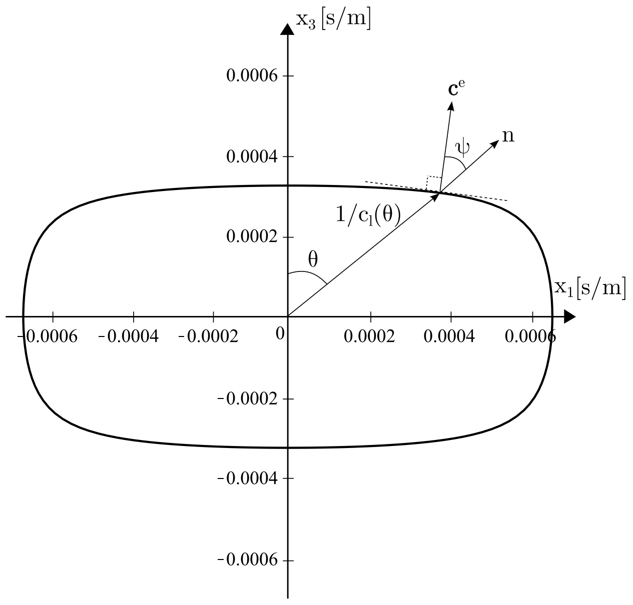


Figure 2.15: *Slowness surface obtained from the longitudinal wave speed c_l. This characteristic surface corresponds with the velocity surface in Figure 2.14. The energy velocity is normal to the surface, and has an angle ψ with respect to the wave propagation direction $\{\mathbf{n}\}$.*

vector divided by the energy per unit volume. The objective is therefore to find an expression for the energy of a plane wave. The derivation of the energy velocity in this subsection is based on the derivation found in [24], but it is modified such that it fits in with the rest of the thesis. With the solution of the form in Equation (2.18), *i.e.*

$$u_i = A_i cos(k_j x_j - \omega t),$$

the following identities can be developed:

$$\frac{\partial u_i}{\partial t} = \omega A_i sin(k_j x_j - \omega t), \quad \frac{\partial u_i}{\partial x_j} = -k_j A_i sin(k_j x_j - \omega t). \tag{2.37}$$

The kinetic energy density can be found from the well-known expression

$$e_k = \frac{1}{2}\rho\left(\frac{\partial u_i}{\partial t}\right)^2 = \frac{1}{2}\rho\omega^2 A_i^2 sin^2(k_j x_j - \omega t). \tag{2.38}$$

The potential energy density of a solid is equal to the elastic energy density, which is [24]

$$e_p = \frac{1}{2}C_{ijkl}\frac{\partial u_i}{\partial x_j}\frac{\partial u_k}{\partial x_l} = \frac{1}{2}C_{ijkl}k_jk_lA_iA_k sin^2(k_jx_j - \omega t).$$

From Christoffel's equation (2.19), we have that

$$C_{ijkl}k_jk_l = \rho\omega^2\delta_{ik},$$

which allows us to write the potential energy density as

$$e_p = \frac{1}{2}\rho\omega^2\delta_{ik}A_iA_k sin^2(k_jx_j - \omega t) = \frac{1}{2}\rho\omega^2A_i^2 sin^2(k_jx_j - \omega t). \tag{2.39}$$

By comparing Equation (2.38) and (2.39), we see that the kinetic and potential energy densities of a plane wave are equal, which is a classic result [24]. The total energy density is then

$$e = e_k + e_p = \rho\omega^2A_i^2 sin^2(k_jx_j - \omega t).$$

We now have an expression for the energy density, and the next step is to write the Poynting vector P_i on a similar form:

$$P_i = -T_{ij}\frac{\partial u_j}{\partial t} = -C_{ijkl}\frac{\partial u_k}{\partial x_l}\frac{\partial u_j}{\partial t},$$

where Equation (2.4) has been used. With the partial derivatives of $\{\mathbf{u}\}$ given in Equation (2.37), we obtain

$$P_i = C_{ijkl}A_kA_j\omega sin^2(k_jx_j - \omega t)k_l.$$

The energy velocity can now be found by dividing the Poynting vector with the energy density, giving

$$c_i^e = \frac{P_i}{e} = \frac{C_{ijkl}A_kA_j\omega sin^2(k_jx_j - \omega t)k_l}{\rho\omega^2A_i^2 sin^2(k_jx_j - \omega t)} = \frac{C_{ijkl}A_kA_jk_l}{\rho\omega A_i^2}.$$

Alternatively, with Equation (2.20) and (2.21), the energy velocity can be expressed as

$$\boxed{c_i^e = \frac{C_{ijkl}A_kA_jn_l}{\rho c A_i^2},} \tag{2.40}$$

where c is one of the wave speeds obtained from Christoffel's equation (2.23). The energy velocity c_i^e gives the direction of energy transport, which is the direction of

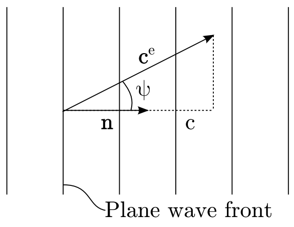


Figure 2.16: *Projection of the energy velocity $\{\mathbf{c}\}^e$ onto the propagation direction $\{\mathbf{n}\}$ gives the wave speed c. Note that c can be c_{ql} or c_{qt}.*

the *acoustic ray* [24]. When $\{\mathbf{c}^e\}$ is parallel to the direction of propagation $\{\mathbf{n}\}$, some authors in literature choose to define that case as a *pure* wave mode. Others define a pure wave as a wave with material displacements $\{\mathbf{A}\}$ perfectly parallel or perpendicular to $\{\mathbf{n}\}$ [24]. In this thesis, we have adopted the latter definition.

To better show the relation between $\{\mathbf{c}^e\}$ and c, we form the following scalar product:

$$\{\mathbf{c}^e\} \cdot \{\mathbf{n}\} = c_i^e n_i = \frac{C_{ijkl} A_k A_j n_l n_i}{\rho c A_i^2}.$$

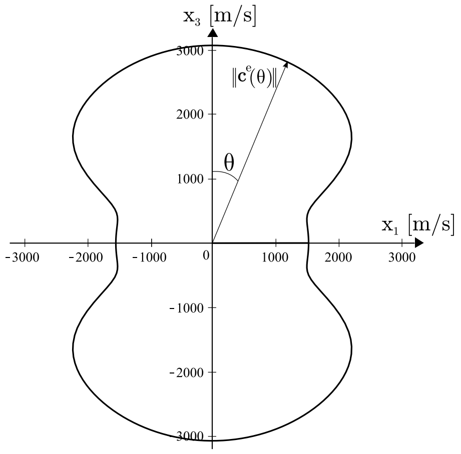
With the relation from Christoffel's equation (2.22), $C_{ijkl} n_j n_k = \rho c^2 \delta_{ik}$, we see that

$$\{\mathbf{c}^e\} \cdot \{\mathbf{n}\} = c, \tag{2.41}$$

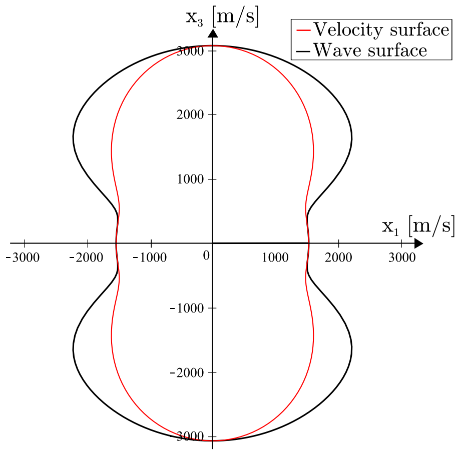
which is illustrated in Figure 2.16. It is possible to show that the energy velocity is, at all points, normal to the slowness surface [24]. Also, for lossless media, the energy velocity equals the group velocity [23, 24]. With knowledge of the energy velocity, we can now create the characteristic wave surfaces.

2.5.4 Wave surfaces

The wave surfaces can be created by plotting the locus of points traced by the end of the energy velocity $\{\mathbf{c}^e\}$. The surface can either be constructed geometrically with the help of Equation (2.41) (see [24] for more details), or it can be constructed directly from Equation (2.40). We have chosen the latter approach in our Maple script, see Appendix A. With our example values, we obtain the wave surface shown in Figure 2.17a. Note that the wave surface in this figure is based on the energy velocity related to the longitudinal wave mode, because we have inserted $\{\mathbf{A}\}_l$ and c_l into Equation (2.40). In the same manner, we could have found the energy velocity related to the shear wave modes by insertion of $\{\mathbf{A}\}_{SV}$ and c_{SV} or $\{\mathbf{A}\}_{SH}$ and c_{SH} into the same equation.



(a) *Plot of the norm of the energy velocity vector gives wave surface*



(b) *The velocity surface from Figure 2.14 plotted on top of the wave surface*

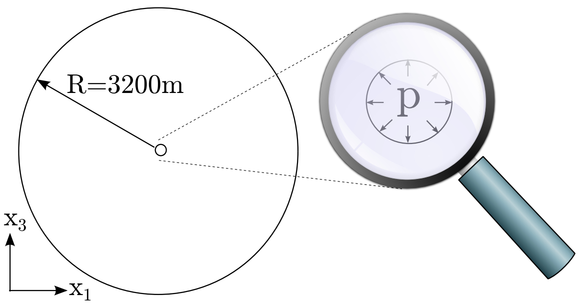
Figure 2.17: *This figure displays how the wave surface encloses the velocity surface, which means that the norm of the energy velocity vector will always be larger than the phase velocity. This is because* $\{\mathbf{c}^e\} \cdot \{\mathbf{n}\} = c$, *as seen in Equation* (2.41).

It can be observed from Figure 2.17b that the norm of the energy velocity $||\{\mathbf{c}\}^e||$ is equal to the longitudinal wave velocity c_l along the x_1 and x_3 directions. This is due to that the energy velocity vector is parallel to $\{\mathbf{n}\}$ in these directions, see Equation (2.41) and Figure 2.15. A physical interpretation of the wave surface is the following. A point source in the origin will emit a wave propagating in all directions. And the points reached by this wave, at unit time, represent the wave surface. In other words, the distance between a point on the surface and the origin, is the distance traveled by the energy in one second. This can be verified by creating a two-dimensional FEM model, as seen in Figure 2.18. We will not go into the details of this model, because it is used only for illustrative purposes. In Figure 2.18b the stress distribution, rather than the energy distribution, is plotted. This is acceptable, because stress and elastic energy is related through

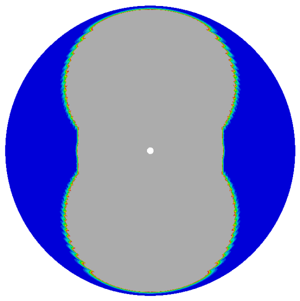
$$e_p = \frac{1}{2} T_{ij} \epsilon_{kl}.$$

The wave surface of an isotropic material will obviously be perfectly circular.

We have seen how the energy velocity $\{\mathbf{c}^e\}$ is useful for describing the behavior of plane waves in anisotropic media. In the next section, we will see how the energy velocity can be used to predict the propagation of *finite-sized* waves in anisotropic media.



(a) *2D circular FEM model. An internal pressure p is applied at time zero.*



(b) *Result from finite element analysis at unit time. The blue area is undisturbed, while the gray is disturbed.*

Figure 2.18: *A FEM model created to replicate the wave surface of Figure 2.17a. There is a clear similarity between the numerically and analytically obtained wave surfaces.*

2.6 Wave propagation from finite-sized sources

A wave propagating from a finite-sized source is not a proper plane wave. Nevertheless, the concepts from plane wave theory can be applied to predict some of the behavior of finite-sized wave propagation in anisotropic media. Rose [23] goes beyond plane wave analysis by solving the finite-size wave problem with the help of Green's function and numerical integration. This is beyond the scope of this thesis, and is therefore not discussed here. According to Rose, the theory we have at hand today represents only the beginning of fully understanding waves in anisotropic media.

An example of a finite-sized source is a transducer used for ultrasonic measurements. One of the major issues with ultrasonic wave propagation in anisotropic media, is the phenomenon called *beam skewing*, see Figure 2.19. Other phenomenons are beam focusing, diverging and splitting, see [23] for more information. If an imaging technique is developed for isotropic materials, corrections for the anisotropy must be made. If no corrections are made, there might occur serious errors in the results [23].

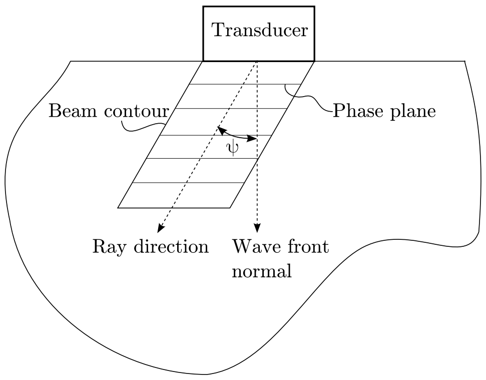


Figure 2.19: *The acoustic beam from a transducer might skew when applied to an anisotropic material. The ray direction is the direction of the energy velocity vector, and it has a skew angle ψ with respect to the wave front normal. The plane wave normal $\{\mathbf{n}\}$ from previous sections is similar to the wave front normal. Note that the energy velocity and skew angle in finite-sized wave problems are not necessarily the same as those obtained from plane wave theory.*

2.7 The use of Maple

The expressions for the eigenvalues and eigenvectors that we derived in Section 2.4 were obtained with the help of the software Maple 16.00 by Maplesoft, which is a commercial computer algebra system. Also, the characteristic surfaces that are included in this thesis were plotted in Maple. The Maple-scripts are made by the author and can be found in Appendix A.

In the book by Royer and Dieulesaint [24], many examples of characteristic surfaces are provided. We verified the correctness of the analytical expressions and the implementation into Maple by checking that we obtained the same characteristic surfaces as Royer and Dieulesaint.

2.8 Finite Element Method and Abaqus

In this thesis, the commercial FEA program Abaqus 6.11-1 by Dassault Systèmes Simulia Corp. was used to solve problems numerically. It is assumed that the reader is familiar with FEM. Nevertheless, some of the theory behind the method is presented here.

FEM is a numerical method that approximates partial differential equations (PDE) through spatial discretization. For problems with time-independent loading these PDEs are approximated with a set of algebraic equations, often on the form

$$[\mathbf{K}]\{\mathbf{u}\} = \{\mathbf{R}\},$$

where $[\mathbf{K}]$ is the global stiffness matrix, $\{\mathbf{u}\}$ is the displacement and $\{\mathbf{R}\}$ is the external load. For linear problems both $[\mathbf{K}]$ and $\{\mathbf{R}\}$ are independent of $\{\mathbf{u}\}$. Transient/dynamic problems are approximated with a set of ordinary differential equations (ODE) on the form

$$[\mathbf{M}]\{\ddot{\mathbf{u}}\} + [\mathbf{C}]\{\dot{\mathbf{u}}\} + [\mathbf{K}]\{\mathbf{u}\} = \{\mathbf{R}\}, \tag{2.42}$$

where the mass matrix $[\mathbf{M}]$ and damping matrix $[\mathbf{C}]$ have been introduced. We now have the inertia, damping and elastic forces that are proportional acceleration, velocity and displacement, respectively. Equation (2.42) represents spatially discretized equations of motion, which require equilibrium between the mentioned forces and the external forces $\{\mathbf{R}\}$. We would like solve for $\{\mathbf{u}\}$ in the ODE, and this requires numerical integration. With knowledge of $\{\mathbf{u}\}$, the stress and strain can be calculated, as seen in Section 2.2.

2.8.1 Dynamic analysis and numerical integration

A wave propagation problem is obviously a dynamic problem. There are several methods for performing dynamic analyses, *e.g.* implicit and explicit direct integration. For the latter method the equations of motion (2.42) can be written on the form [9]

$$[\mathbf{M}]\{\ddot{\mathbf{u}}\}_n + [\mathbf{C}]\{\dot{\mathbf{u}}\}_{n-1/2} + [\mathbf{K}]\{\mathbf{u}\}_n = \{\mathbf{R}\}_n.$$

The subscript n signals that the variables are evaluated at time step t_n. Note that the velocity $\{\dot{\mathbf{u}}\}_{n-1/2}$ is lagging by half a time step. This is because Abaqus uses the Half-Step Central Differences operator for explicit dynamic problems [9]:

$$\begin{aligned}
\{\ddot{\mathbf{u}}\}_n &= \frac{1}{\Delta t}\left(\{\dot{\mathbf{u}}\}_{n+1/2} - \{\dot{\mathbf{u}}\}_{n-1/2}\right) = \frac{\{\mathbf{u}\}_{n+1} - 2\{\mathbf{u}\}_n + \{\mathbf{u}\}_{n-1}}{\Delta t^2}, \\
\{\mathbf{u}\}_{n+1} &= \{\mathbf{u}\}_{n+1} + \Delta t_{n+1}\{\dot{\mathbf{u}}\}_{n+1/2} = \{\mathbf{u}\}_n + \Delta t\{\dot{\mathbf{u}}\}_{n-1/2} + \Delta t^2\{\ddot{\mathbf{u}}\}_n.
\end{aligned}$$

The operator is explicit in the sense that $\{\mathbf{u}\}_{n+1}$ is found from known values of $\{\mathbf{u}\}_n$, $\{\dot{\mathbf{u}}\}_{n-1/2}$ and $\{\ddot{\mathbf{u}}\}_n$. Note that we have assumed a fixed time step Δt, which simplify the expressions, but is not necessarily the case. With the central difference operator, the expression for the displacement at time step t_{n+1} is

$$\{\mathbf{u}\}_{n+1} = [\mathbf{K}^{\mathbf{eff}}]^{-1}\{\mathbf{R}^{\mathbf{eff}}\}_{n+1},$$

where

$$\begin{aligned}[\mathbf{K}^{\mathbf{eff}}] &= \frac{1}{\Delta t^2}[\mathbf{M}], \\ \{\mathbf{R}^{\mathbf{eff}}\}_{n+1} &= \{\mathbf{R}^{\mathbf{ext}}\}_n - [\mathbf{K}]\{\mathbf{u}\}_n + [\mathbf{M}]\frac{\{\mathbf{u}\}_n + \Delta t\{\dot{\mathbf{u}}\}_{n-1/2}}{\Delta t^2} - [\mathbf{C}]\{\dot{\mathbf{u}}\}_{n-1/2}.\end{aligned}$$

With a diagonal ('lumped') mass matrix, it is apparent that $\{\mathbf{u}\}_{n+1}$ is found without equation solving, and is therefore cheaply computed. It is here the strength of the explicit time integration algorithms becomes clear. Explicit integration is recommended for dynamic analyses if the wave propagation problem is created by blast or impact loading, because these problems require small time steps to capture relatively rapid changes in the load and response [9, 11]. In this thesis, we only considered impulse loading, and an explicit solver is therefore appropriate. A downside with the central difference operator, is that it has the stability limit

$$\Delta t \leq \frac{2}{\omega_{max}},$$

or with damping present

$$\Delta t \leq \frac{2}{\omega_{max}}(\sqrt{1+\xi_{max}^2} - \xi_{max}),$$

where ω_{max} is the highest natural frequency in the system and ξ_{max} is the fraction of critical damping in the mode with the highest frequency. An important notice is that there is no amplitude error introduced by central difference discretization as long as the chosen time step is below the stability limit [9]. However, no amplitude error does not mean that the computed amplitudes will be exact.

According to the Abaqus Documentation [11], the stable time increment can be estimated by the following ratio

$$\Delta t \approx \frac{\Delta x_{min}}{c_l}, \tag{2.43}$$

where Δx_{min} is the smallest element dimension in the mesh and c_l is the longitudinal wave speed found by Equation (2.15). This relation is similar to the so-called *Courant-Friedrichs-Lewy (CFL) condition*, which can be derived by performing a stability analysis on Equation (2.1) [1]. The CFL condition can be interpreted as follows: *the 'numerical velocity' must be greater or equal to the physical velocity, i.e.*

$$\frac{\Delta x_{min}}{\Delta t} \geq c_l.$$

Care must be taken when we apply Equation (2.43), because this estimate is not conservative for all element types [9]. One should also be aware of that decreasing Δt might lead to lower accuracy [9, 19]. A thorough study of accuracy and the CFL-condition was performed in the pre-study project [12]. In this project, it was also argued that Abaqus' automatic time-stepping algorithm produces satisfactory result. This algorithm is useful when we deal with anisotropic materials and more complex geometry, where a stability limit is not easily found. Automatic time-stepping was therefore chosen throughout this thesis.

2.8.2 Loading

We applied some sort of impulse load to all the models in this thesis, because we know that AVC causes a shock wave that propagates down septum. The amplitude and duration of the load can be found in Figure 2.20. Mathematically speaking, the load function $F(t)$ is given by

$$F(t) = \begin{cases} 1, & \text{if} \quad 1 \cdot 10^{-4} s < t < 2 \cdot 10^{-4} s, \\ 0, & \text{otherwise.} \end{cases}$$

We found no data on the duration of the 'load' caused by AVC in literature, and an assumption was therefore made. The amplitude of the load was irrelevant for our case, because we worked only with linear theory. The impulse load was expected to give rise to a stress wave of corresponding period that propagated through the models. To avoid potential difficulties with an infinite slope, as in the impulse load, the *Smooth Step* function in Abaqus was used. This made the edges of the load function in Figure 2.20 very slightly curved.

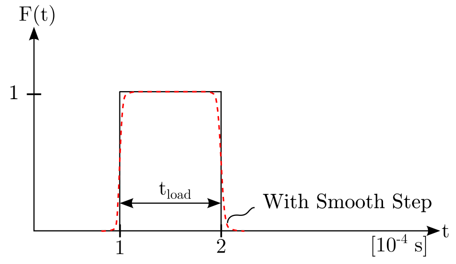


Figure 2.20: *Graphical representation of the load function $F(t)$. Abaqus' Smooth Step function removes sharp corners and infinite slopes.*

2.8.3 Choice of elements

Higher order elements are in general more accurate in static analyses, but for dynamic analyses they tend to produce more noise, because they have higher natural frequencies [9]. It is therefore best to avoid higher-order elements for our case. All models in this thesis have elements with a reduced number of integration points. This reduces computation time, but might increase dispersion effects [20]. Regardless, Abaqus does not allow elements with full integration in Abaqus/Explicit.

In the pre-study project [12], the size of the elements was calculated from the rule of thumb

$$L_{el} = \frac{t_{load} \cdot c}{N} \tag{2.44}$$

where L_{el} is the length of an element, t_{load} is the duration of the impulse load, c can be either c_l or c_t and N is a number chosen by the user. Abaqus Documentation suggests to set $N = 10$ for impact problems, which means that the loading takes place over the span of 10 elements [11]. In the pre-study project [12], we compared analytically and numerically obtained wave speeds, and errors of less than 5 percent were obtained with $N = 10$ and $c = c_l$ in Equation (2.44). We continued to use this rule of thumb for the models in this thesis.

2.8.4 Butterworth filter

From some of the analyses, we experienced large oscillations in the response. This made it difficult to find the shear wave speed. We therefore ran the response through a filter, which is built into Abaqus. The filter is called *Butterworth filter* and require that the user must specify a cutoff frequency. The filter basically accepts low-frequency data and rejects data above the cutoff frequency, see the Abaqus Documentation for more details [11]. An example of use of the Butterworth filter is shown in Figure 2.21.

Filtering the response does not necessarily give more physically correct results, but it makes it easier to work with the response. Brekke *et al.* used a Butterworth filter to remove clutter from their velocity curves [6].

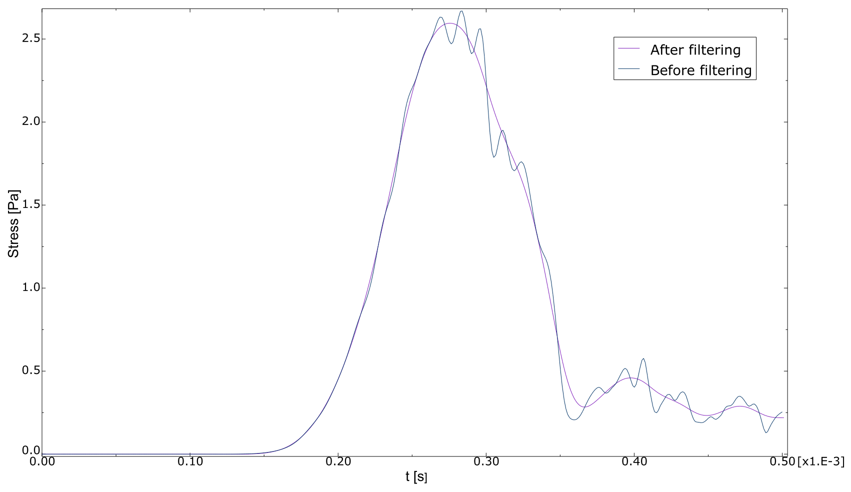


Figure 2.21: *The use of the Butterworth filter on the stress response at one node in a model that is introduced later. The cutoff frequency is here 20,000 Hz.*

3 Results

In this chapter, we first analyze a transversely isotropic material with the help of characteristic surfaces. In the following sections, we introduce various FEM models, and perform studies on them. The models are listed in order below with a short description:

1. Plane Wave Model - A simple two-dimensional model that simulates plane wave propagation.
2. Finite-sized Wave Model - A two-dimensional model that has a finite-sized source. The model displays the effect of energy transport direction well.
3. Curved Model - A three-dimensional model, which is made for investigation of the effect of curvature on wave speed. The model can represent the septum.
4. Truncated Ellipsoid Model - A three-dimensional model, which is a simple representation of the left ventricle.

To the best of our knowledge, models 1 to 3 are original. The Truncated Ellipsoid Model is based on models found literature [14, 26]. All the models are made from Python scripts developed by the author. Python is a general-purpose programming language. Abaqus can read the scripts and generate the model in the Abaqus graphical user interface. The advantage with scripts is that it is easy to perform various studies, *e.g.* parameter study and mesh study. The Python scripts can be found in Appendix B, and are written such that the user can easily change, for example, the geometry of the model. Also, the wave speed measurement procedures described later, are developed by the author.

All the field plots included, which are color plots that show for example the stress distribution, are extracted from Abaqus.

3.1 Analysis of a transversely isotropic material

It is interesting to investigate the properties of the material we implemented into our models. With the results from this section, we were able to better understand the results from the FEA that are presented later in the chapter. The material was analyzed by use of the tools developed in the Chapter 2.

3.1.1 Material constants

Throughout this thesis we worked with a transversely isotropic material that has the following stiffness coefficients:

Table 3.1: *The elastic stiffness coefficients obtained from a formalin fixed myocardium by Hoffmeister et al. [13].*

	C_{11}	C_{33}	C_{44}	C_{12}	C_{13}
[GPa]	2.46	2.53	$8.97 \cdot 10^{-3}$	2.4431	2.44

Hoffmeister *et al.* [13] did the stiffness measurements on a formalin fixed myocardium, which implies that they did the measurements *ex vivo*. We have not found, in literature, stiffness coefficients for a transversely isotropic material that were obtained from *in vivo* measurements of the myocardium. Hoffmeister *et al.* explain that soft tissues generally become stiffer when formalin fixed. They discuss, however, how the longitudinal wave speed is unaffected by the process. The effect on the shear wave speed is unknown, but they suggest that there is a significant change to it. Nevertheless, it was not important to produce realistic result from the models in this thesis. We were more interested in the *concepts* of wave propagation in anisotropic materials. With the Maple and Python scripts in the appendices, the user can easily modify the stiffness coefficients to his or hers own needs.

There are many reports on the density of soft tissues in literature. Soft tissues consist mainly of water, and have therefore density close to water. In this thesis we continued to use the mass density that was used in the pre-study project [12], *i.e.* $\rho = 1060$ kg/m^3.

3.1.2 Material displacement analysis

In Section 2.4, we derived expressions for the eigenvectors in the x_1-x_3 plane. As we know, these three eigenvectors express the material displacement direction for plane waves that propagate in direction $\{\mathbf{n}\}$. One of the eigenvectors showed that the shear horizontal wave mode is a pure transverse wave mode for all θs, and that its polarization is out of the plane. On the other hand, the two other eigenvectors express that the quasi-longitudinal and quasi-shear wave modes are generally not pure wave modes. It is therefore interesting to find out how 'quasi' these two wave modes are. In other words, we would like to find the angle between the wave propagation direction $\{\mathbf{n}\}$ and the material displacement vector $\{\mathbf{A}\}_i$, see Figure 3.1. The author has chosen to call this angle for the quasi angle ζ. Do not confuse ζ with the skew angle ψ, which is shown in for example Figure 2.15.

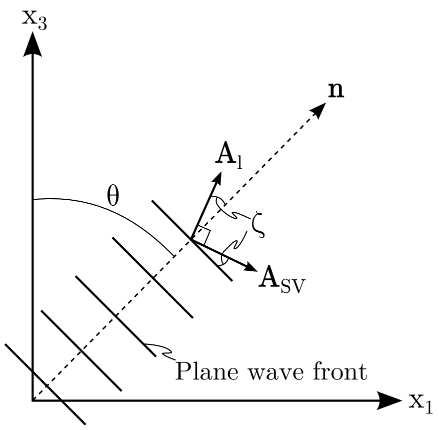


Figure 3.1: *Illustration of a plane wave that propagates in direction $\{\mathbf{n}\}$. The quasi-angle ζ expresses the angle between the wavefront normal $\{\mathbf{n}\}$ and the material displacement vector $\{\mathbf{A}\}_l$. Since $\{\mathbf{A}\}_{SV}$ is always orthogonal to $\{\mathbf{A}\}_l$, the same angle can be found between the wave front tangent and $\{\mathbf{A}\}_{SV}$.*

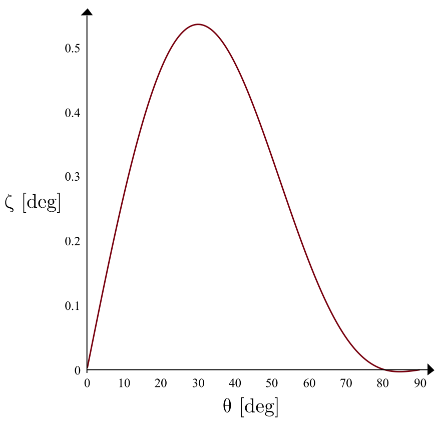
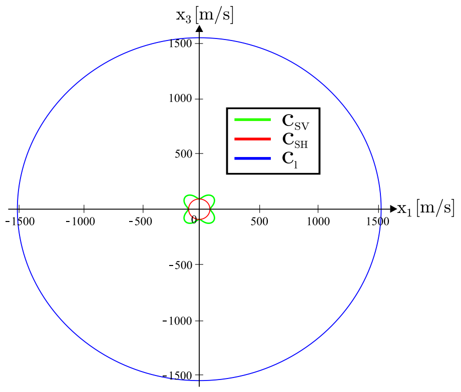


Figure 3.2: *The quasi-angle ζ is plotted for versus θ.*

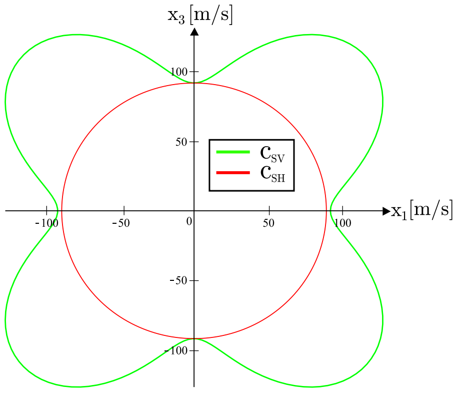
In Figure 3.2, we have found the quasi angle ζ as a function of θ. The plot was made in Maple, where we used the analytical expression for $\{\mathbf{A}\}_{SV}$ to find the angle ζ. As one might expect, the quasi angle varies for different θs. We can also see from this plot that the wave modes are *pure* longitudinal and shear for $\theta = 0$ and $\theta = 90°$, which has already been shown in Section 2.4. Note that ζ is small for all θs, even at its maximum. This indicates that the anisotropy is weak in the material (for the limiting case of isotropy, the quasi angle would be zero for all θs). We therefore neglect the quasi angle in the remainder of the thesis, and consider the wave modes as pure for all θs.

3.1.3 Characteristic surfaces

With knowledge of the stiffness coefficients and density, we can create the characteristic surfaces of the material. Figure 3.3 displays the velocity surfaces of the material. From inspection of this figure we see that c_l and c_{SH} generate almost circular velocity surfaces. This means that these wave speeds are close to independent of the propagation direction. Nevertheless, c_l is slightly larger along the x_3 axis compared to along the x_1 axis, because C_{33} is slightly larger than C_{11}. This is shown in Figure 3.4, a more quantitative presentation of the wave speeds. On the other hand, c_{SV} varies significantly with the propagation direction. It equals c_{SH} along the x_3 axis. This agrees with what was shown in Section 2.4.



(a) *The longitudinal velocity surface has a radius approximately 15 times larger than the shear velocity surfaces.*



(b) *Same plot as in a), but now with only the shear wave speeds.*

Figure 3.3: *The velocity surfaces of the material. Only the shear vertical (SV) shear wave speed shows a significant dependence on propagation direction.*

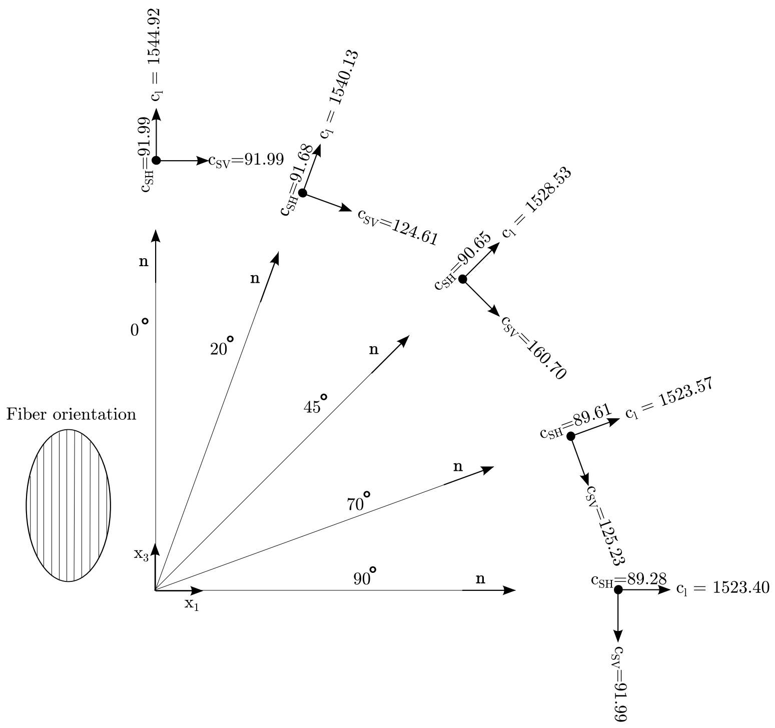


Figure 3.4: *An illustration on how the wave speeds quantitatively vary with θ. The arrows indicate the material displacement direction. Values are given in m/s.*

Since the velocity surface for c_l and c_{SH} are close to circular, and thereby not so interesting, we focus the remainder of this subsection on the shear vertical wave mode. According to Figure 3.3b it seems that c_{SV} has a maximum at approximately $\theta = 45°$. To investigate this further, we look back at Equation (2.28), *i.e.*

$$c_2 = c_{SV} = \left[\frac{1}{2\rho}\left(C_{11}sin^2\theta + C_{33}cos^2\theta + C_{44} - \sqrt{B}\right)\right]^{1/2},$$

where

$$\begin{aligned} B \;=\; & \left(C_{11} - C_{44}\right)^2 + \\ & \left(-2C_{11}^2 - 2C_{11}C_{33} + 6C_{11}C_{44} + 2C_{33}C_{44} + 4C_{13}^2 + 8C_{13}C_{44}\right)cos^2\theta \\ & +\left(C_{11}^2 + 2C_{11}C_{33} + C_{33}^2 - 4C_{11}C_{44} - 4C_{33}C_{44} - 4C_{13}^2 - 8C_{13}C_{44}\right)cos^4\theta. \end{aligned}$$

As mentioned before, c_{SV} is independent of C_{12}. We differentiate the expression for c_{SV} with respect to θ and set it equal to zero, and then solve for θ, which gives

$$\theta_{min} = 0 \quad \text{or} \quad \theta_{min} = 90° \quad \text{or} \quad \theta_{max} = cos^{-1}\left(\sqrt{\frac{C_{11} + C_{13}}{2C_{13} + C_{33} + C_{11}}}\right). \tag{3.1}$$

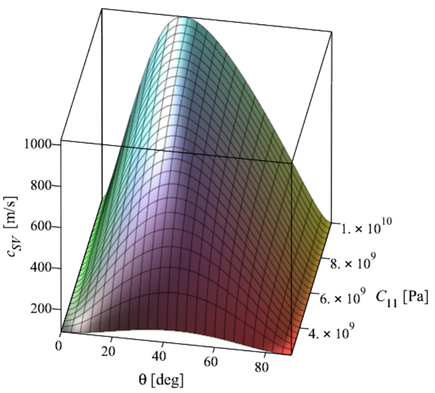
We have given these θs the subscripts 'min' and 'max', because they refer to the values of θ that give the extremal values for c_{SV}. One should note that θ_{min} is independent of the stiffness coefficients, while θ_{max} is not dependent of C_{44}. If we insert the expressions for θ_{min} and θ_{max} into the expression for c_{SV} we obtain

$$c_{SV,min} = \sqrt{\frac{C_{44}}{\rho}} \quad \text{and} \quad c_{SV,max} = \sqrt{\frac{C_{11}C_{33} - C_{13}^2}{\rho(C_{33} + C_{11} + 2C_{13})}}. \tag{3.2}$$

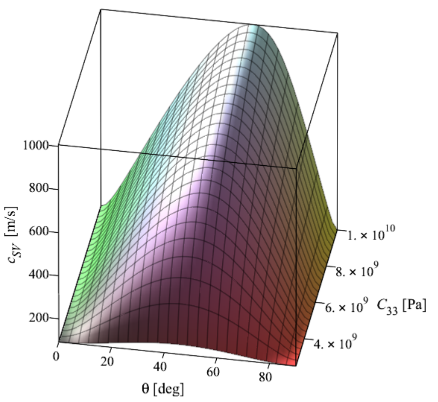
The expression for $c_{SV,min}$ has already been seen in Section 2.4. It it interesting to note that C_{44} solely governs $c_{SV,min}$, while it is absent in the expression for $c_{SV,max}$. With the stiffness coefficients introduced earlier, we find the extremal values to be

$$c_{SV,min} = 91.99 \quad [\text{m/s}], \qquad c_{SV,max} = 160.71 \quad [\text{m/s}] \quad \text{at} \quad \theta = 45.2°.$$

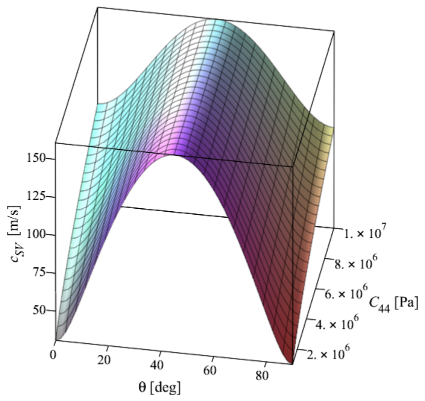
We see that the maximum shear wave speed is not necessarily at $\theta = 45.0°$. A graphical presentation of the relation between the stiffness coefficients and the wave speed c_{SV} is given in Figure 3.5.



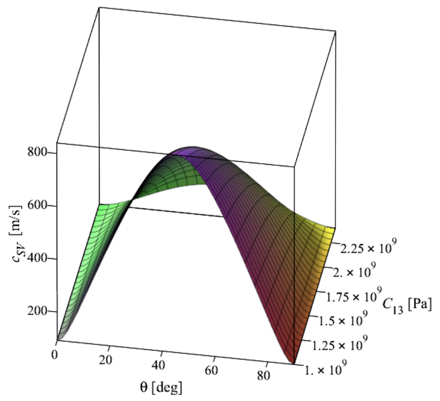
(a) *c_{SV} versus θ and C_{11}*



(b) *c_{SV} versus θ and C_{33}*



(c) *c_{SV} versus θ and C_{44}*



(d) *c_{SV} versus θ and C_{13}*

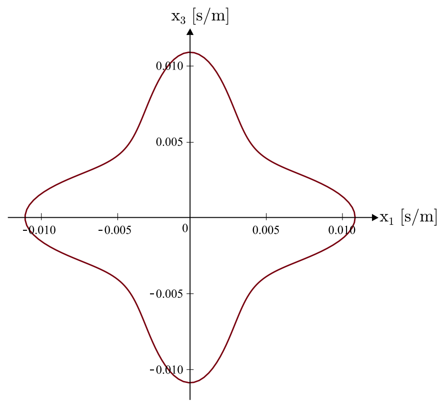
Figure 3.5: *Surface plots of the shear vertical wave speed c_{SV} versus θ and one stiffness coefficients. The other stiffness coefficients are kept constant. We clearly see that changes in stiffness leads to changes in wave speeds. Similar plots can obviously be made for the two other wave speeds as well.*

From Figure 3.5a and 3.5b we see that an increase of the longitudinal stiffness coefficients C_{11} and C_{33} generally leads to a higher wave speed, as one might expect from inspection of Equation (3.2). It is also clear that θ_{max} is shifted away from $\theta = 45°$ when C_{11} and C_{33} is increased. Note that the minimum value $c_{SV,min}$ is independent of these two stiffness coefficients, as already shown in Equation (3.2).

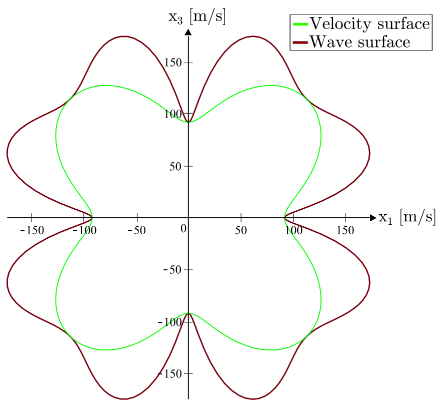
In Figure 3.5c we observe that an increase of C_{44} also generally leads to a higher wave speed, but neither θ_{max} nor $c_{SV,max}$ is affected. This last fact agrees with Equation (3.1) and (3.2). The plot in Figure 3.5d displays how an increase in the stiffness coefficient C_{13} generally leads to lower wave speed. This can be expected, because according to Equation (3.2) $c_{SV,max}$ decreases when C_{13} is increased. In Equation (3.1), it can be seen that also θ_{max} is dependent on C_{13}, even though this is not easily spotted from the angle in Figure 3.5d.

Comment: *to the best of our knowledge, the type of surfaces in Figure 3.5 is novel. The surfaces proved to be a good tool for a qualitative study of the stiffness dependence.*

The velocity, slowness, and wave surfaces are plotted in Figure 3.6. From this figure we see that also the energy velocity varies significantly with θ. As mentioned in Section 2.5 the energy velocity is always equal to or larger than the phase velocity. This means that the wave surface surrounds the velocity surface.



(a) *Slowness surface obtained from $1/c_{SV}$.*



(b) *Wave surface and velocity surface obtained from c^e_{SV} and c_{SV}, respectively.*

Figure 3.6: *The wave speed and energy velocity are equal along the x_1 and x_3 axis and at approximately $\theta = 45°$. This is due that the wave propagation direction $\{\mathbf{n}\}$ and the normal to the slowness surface are parallel at these positions, see Equation* (2.41).

3.2 Plane Wave Model

All the wave theory presented in Chapter 2, except for wave propagation in isotropic media, is based on the assumption of a plane wave solution, *i.e.* Equation (2.18). We refer to this as plane wave theory. It was therefore appropriate to model plane wave propagation with the help of the FEM and Abaqus. The goal was to obtain wave speeds from the FEM that matched the ones obtained from the analytical formulas. If they match, we can say that the model simulates plane wave propagation well, and that the FEM is able to handle wave propagation in anisotropic media.

We would like to model an infinite domain. One option is to use so-called infinite elements, which are supposed to provide a 'quiet' boundary. However, infinite elements were not used, because they did not seem to work well with anisotropic materials (the author experienced that the energy was not damped out properly). Also, the implementation of infinite elements is laborious. Another way to model plane waves is to create the model shown in Figure 3.7. A similar model gave accurate and satisfactory results in the pre-study project [12]. With the boundary conditions shown in the figure, this two-dimensional model simulates a one-dimensional problem. The time-dependency of the load is given in Section 2.8.2. The model generates a plane shear wave that propagates upwards.

We focused on the shear vertical wave mode, since this proved to be the most interesting mode in the previous section. The FEM model should be able to generate a plane wave that propagates in different wave propagation directions $\{\mathbf{n}\}$. One way to achieve this is to rotate the material orientation in Abaqus. Another way, is to simply rotate the whole model and keep the material orientation fixed, as in Figure 3.8. The latter option was chosen for the Plane Wave Model.

The element type used was *CPE4R*, four noded plane strain quadrilateral elements with reduced integration. A homogeneous mesh is preferred, because additional dispersion effects are then avoided [20]. The element size was calculated from Equation (2.44) with $N = 10$, which gave

$$L_{el} = \frac{t_{load} c_t}{10} = \frac{1 \cdot 10^{-4}\text{s} \cdot 100\text{m/s}}{10} = 0.001\text{m}. \tag{3.3}$$

This gave in total 40 000 elements. Note that we assumed the shear wave speed to be 100 m/s based on the velocity surfaces in Figure 3.3. We used $c_t = 100$ m/s to calculate the element size for the other models in thesis as well. Keep in mind that the theory presented in Chapter 2 applies for infinite media. However, a thin structure can, for all practical purposes, still be considered a semi-infinite media if the wavelength is small compared to the thickness of the object [23]. The thickness of the Plane Wave Model is one meter and the wavelength of the shear wave is approximately

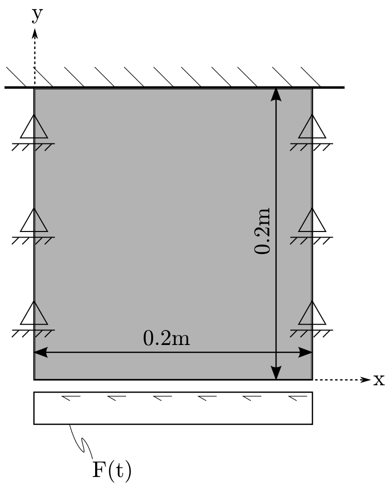


Figure 3.7: *An illustration of the Plane Wave Model. The traction load $F(t)$ is applied to the bottom boundary of the quadratic plate. This means that the stress $T_{xy}(x,0,t) = F(t)$. At the top boundary the plate is fixed, meaning $u_x(x,0.2,t) = u_y(x,0.2,t) = u'_x(x,0.2,t) = u'_y(x,0.2,t) = 0$. At the left and right boundaries, the boundary conditions $u_y(0,y,t) = u_y(0.2,y,t) = 0$ are applied, which are called rolling support boundary conditions.*

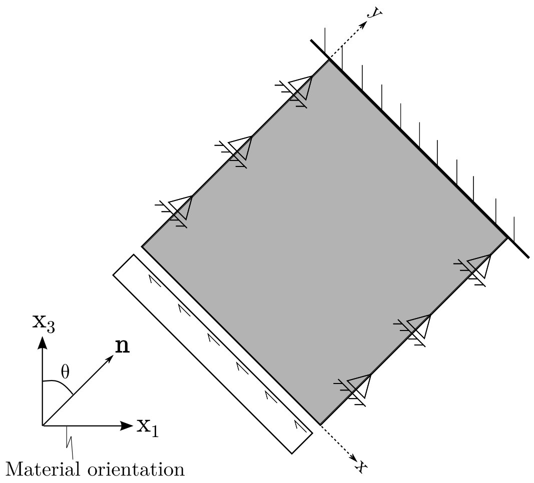


Figure 3.8: *An illustration of the same model as in Figure 3.7, only rotated. The plane wave will now propagate in the direction of $\{\mathbf{n}\}$ given by the angle θ. The material orientation is fixed.*

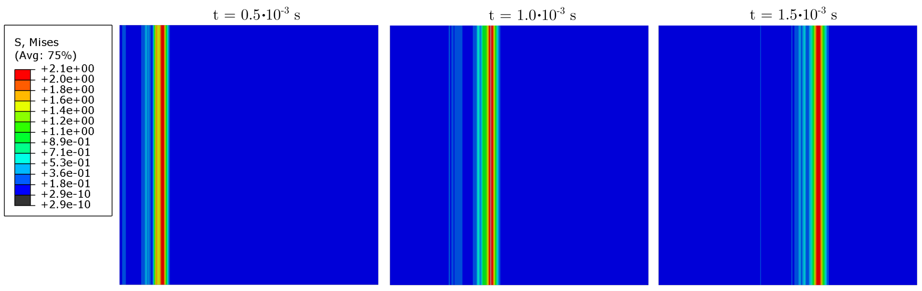


Figure 3.9: *Field plot of the Von Mises stress at three different time instances, extracted from Abaqus. The plots are from the plane wave model when $\theta = 90°$. One can clearly see that the wave propagates from left to right.*

$$\lambda_t \approx t_{load} \cdot c_t = 1 \cdot 10^{-4} s \cdot 100 m/s = 0.01 m. \tag{3.4}$$

Hence, we can regard the model as infinite in the thickness direction as well. In Equation (3.4), we have implied that the period of the wave is t_{load}, even though it is mathematically incorrect to say that an impulse response has a period.

In Figure 3.9, field plots of the stress are provided and a wave propagation can easily be observed. Note that the Von Mises stress is plotted rather than the shear stress. This is acceptable, because we are not interested in the values of the stress. The Von Mises stress is given by [16]

$$T_m^2 = \frac{1}{2}\left[(T_{11} - T_{22})^2 + (T_{22} - T_{33})^2 + (T_{11} - T_{33})^2 + 6(T_{23}^2 + T_{13}^2 + T_{12}^2)\right].$$

The Von Mises stress is independent of the orientation of the coordinate system, and is therefore handy when we rotate the model. Obviously, the shear stress components are the main contributors to the Von Mises stress for the Plane Wave Model, and that is why we can use the Von Mises stress to find the shear wave speed. Next, we discuss how to find the wave speeds from the model, and then a wave speed study is presented.

3.2.1 Wave speed study

To obtain the wave speed from the finite element model, we observed the stress at the node at the center of the plate. A minor issue is that the stress is evaluated at the elements' integration point. And since the integration point is at the center of the elements, the stress cannot be found directly at the center node, see Figure 3.10. However, Abaqus allows us to easily find the average of the four nearest integration points. Information to users that want to use this averaging: set the 'Average

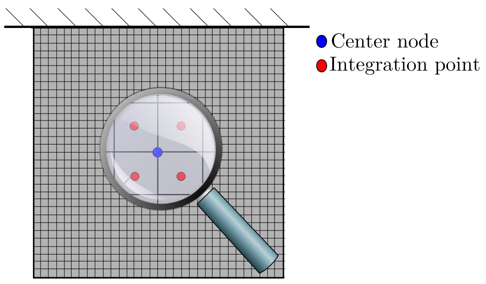


Figure 3.10: *A sketch of the mesh. The node at the center of the plate is magnified, and the surrounding integration points are shown. To find the stress at the center node, we average the stress of the four nearest integration points.*

threshold' to 100% (default is 75%) and uncheck the box 'Use region boundaries' to obtain smooth XY-plots.

The plot of stress versus time at the center node for two cases of orientation θ is shown in Figure 3.11.

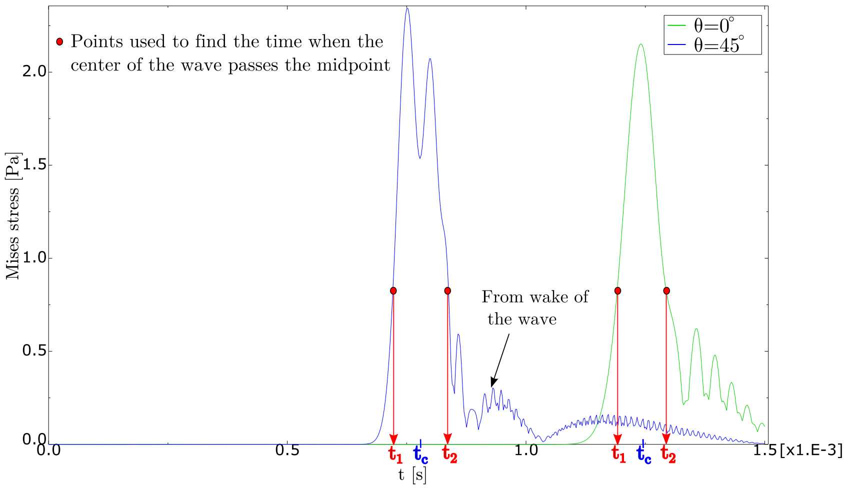


Figure 3.11: *Plot of Von Mises stress versus time obtained at the center node of the plate when the orientation is $\theta = 0°$ and $\theta = 45°$. The time t_1 and t_2 can be found with the help of 'Probe values' in Abaqus. The average of t_1 and t_2 gives the time t_c, which is used to calculate the wave speed. After the 'main' wave has passed the node, there are smaller disturbances that appear. These come from the wake of the wave. Since the wave speed at $\theta = 45°$ is higher, the wave arrives earlier at the center node than when $\theta = 0°$.*

We found the wave speed with knowledge of the length of the model and the time it takes for the wave to arrive at the center node. Note that the times t_c in Figure 3.11 were the times used in the calculation of the wave speeds. These times correspond to when the center of the waves pass the center node. We used the center of the wave to calculate the wave speeds, because we then avoided potential effects from smearing and attenuation of the wave. We can note from Figure 3.11 that the shape of the wave depends on the orientation θ. It is therefore better to use t_1 and t_2 to find when the center of the wave passes the center node, rather than, for example, the time of maximum stress. The wake of the wave is an effect that arises due to inertia and elasticity in the material; when the load is removed the material continues to oscillate. High-frequency oscillations in the response are expected when we work with impact loading, see for example Figure 11.12-4 in [9].

A study was performed where we extracted the shear wave speed from Abaqus at certain angles θ. The result is shown in Figure 3.12. We can see that the Plane Wave Model is able to accurately model the theoretical shear wave speed. The relative error in Figure 3.12b is calculated by the following formula:

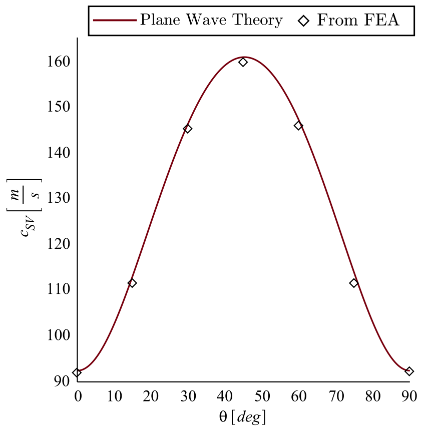
$$Error = \frac{c_{theory} - c_{FEA}}{c_{theory}} \cdot 100\%,$$

and we see that the error is less than 2% for all the selected values of θ. These errors are smaller than what we found in the pre-study project, where we reported the wave speeds from a similar model, which had an isotropic material implemented [12]. This can be explained by that we here use c_t rather than c_l in Equation (2.44) to calculate the element size. Based on the relatively low error, we can say that we are satisfied with the mesh density and the performance of the automatic time stepping algorithm. We study the mesh density in more detail in the next subsection. In the pre-study project it was also shown how the numerically obtained phase velocity is expected to be slightly less than the theoretical velocity due to numerical dispersion. We can see that this is also the case here.

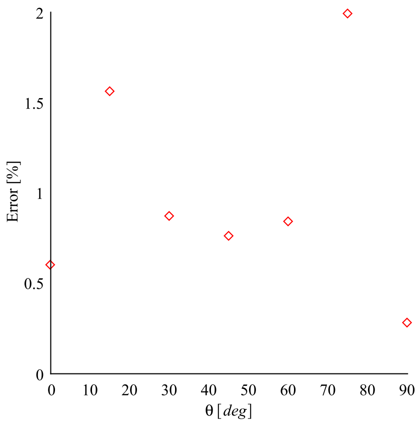
3.2.2 Mesh study

With a mesh study we can analyze the accuracy of the obtained shear wave speeds. In this study, we varied N in Equation (2.44), *i.e.* the element size, and extracted the corresponding shear wave speed. The result is shown in Figure 3.13. As one might expect the figure shows that more elements lead to a more accurate result with respect to theory. This gives confidence in the FEM model and the wave speed measurement procedure.

In the next section, we go one step further and study a two-dimensional model with finite-sized waves.



(a) *Wave speed c_{SV} versus the angle θ.*



(b) Relative error corresponding to a).

Figure 3.12: *Result from the wave speed study. The numerically obtained wave speeds are plotted together with the analytical solution, Equation* (2.28)*. The Plane Wave Model seems to accurately model the shear wave speed.*

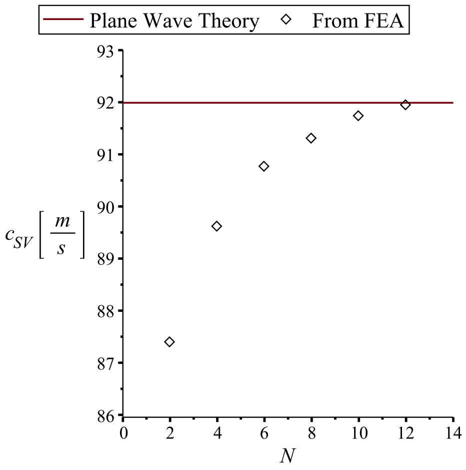


Figure 3.13: *The result from the mesh study performed on the Plane Wave Model with $\theta = 0°$. An increase in N means a finer mesh density. We can clearly see a tendency of convergence towards approximately the theoretical value for increase in N.*

3.3 Finite-sized Wave Model

With this finite element model we wanted to find out how well the plane wave theory developed in Chapter 2 can predict the behavior for finite-sized waves. Rose compares phase velocities, energy (group) velocities and skew angles obtained from plane wave theory with experimental data for a cast iron steel [23]. The experimental data was obtained with the help of a transducer, *i.e.* a finite source, and there was a good agreement with theory. We therefore expected the result obtained from FEA to also agree with plane wave theory. The present model resembles the Plane Wave Model, where the main difference is that we modeled some of the infinite domain, which surrounds the source, see Figure 3.14. In this way, a plane wave front (phase plane) of finite size propagates through the domain. Note that the rolling supports boundary conditions are absent in this model. Everything else, *e.g.* mesh density and load, is exactly the same as for the Plane Wave Model.

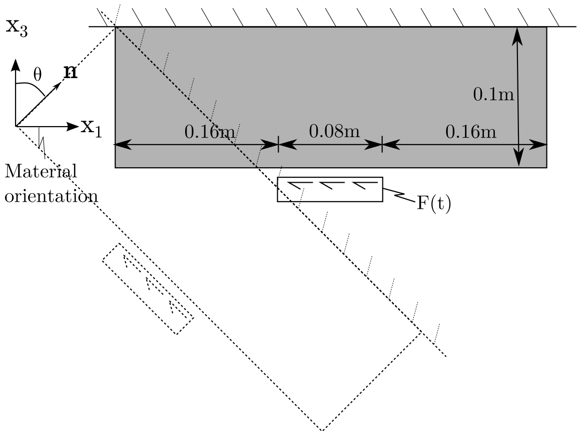
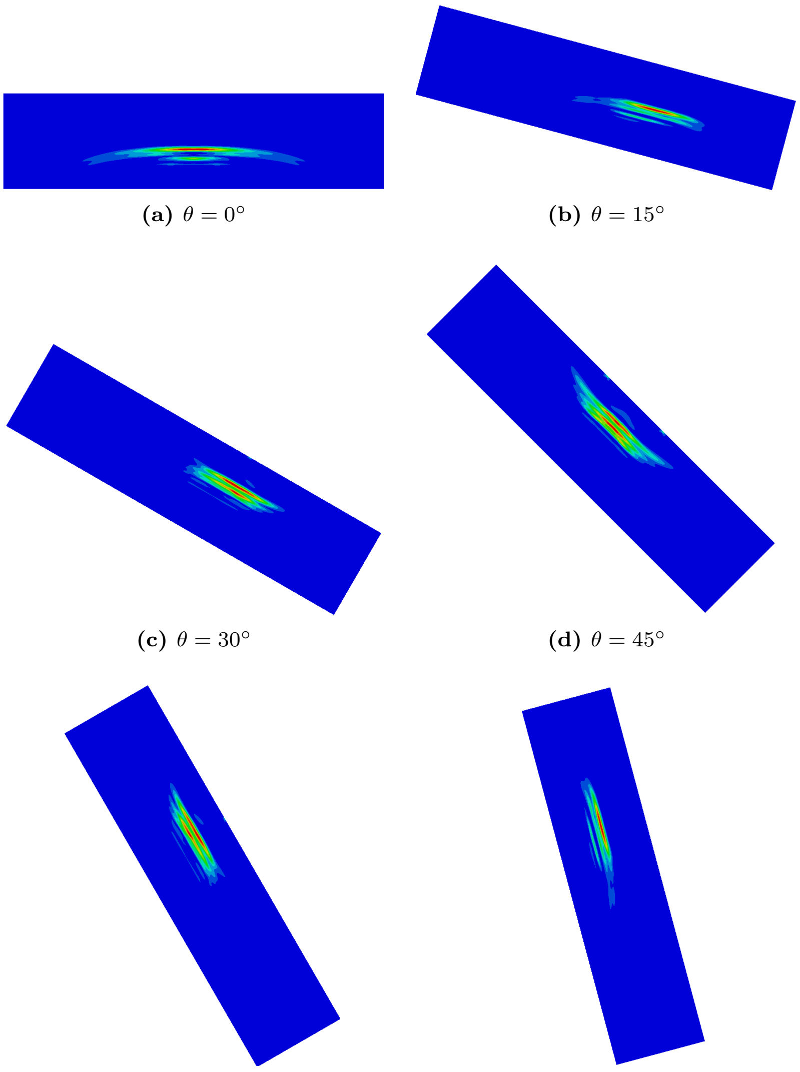


Figure 3.14: *Illustration of the Finite-sized Wave Model. Similarly to the Plane Wave Model, it can be rotated by an angle θ, which gives the wave propagation direction $\{\mathbf{n}\}$. Also here the material orientation is fixed.*

Before we look at the results from the Finite-sized Wave Model, let us briefly summarize some of the points presented in Chapter 2. There are three directions we have to keep in mind:

- $\{\mathbf{n}\}$ - Wave propagation direction. Normal to the wave front.
- $\{\mathbf{A}\}_i$ - Material displacement direction (polarization). It is orthogonal to $\{\mathbf{n}\}$ for a pure shear wave.
- $\{\mathbf{c}\}^e$ - Energy velocity. It gives the direction of energy transport.

In Figure 3.15 the strain energy density is plotted at a certain time instance for different θs. The strain energy density is plotted rather than the stress, because this gave more clear plots. However, the wave front could easily be seen also if the



(a) $\theta = 0°$ **(b)** $\theta = 15°$

(c) $\theta = 30°$ **(d)** $\theta = 45°$

(e) $\theta = 60°$ **(f)** $\theta = 75°$

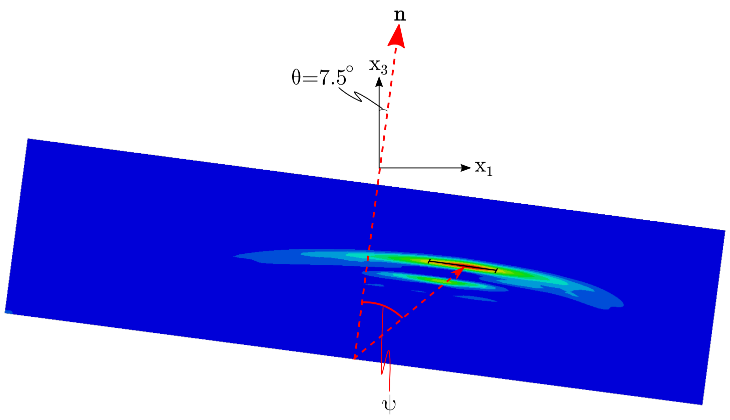
Figure 3.15: *Field plot of the strain energy density at* $t = 5.8 \cdot 10^{-4}$*s, obtained from Abaqus. Red color shows areas with the highest strain energy density. In the blue areas, the strain energy density is approximately zero. By comparison of the figures, it is clear that both the energy and phase velocity depend on* θ*.*

stress was plotted. We can see from Figure 3.15 that the energy velocity direction and the wave propagation direction are different for certain θs. The angle between them is the skew angle ψ.

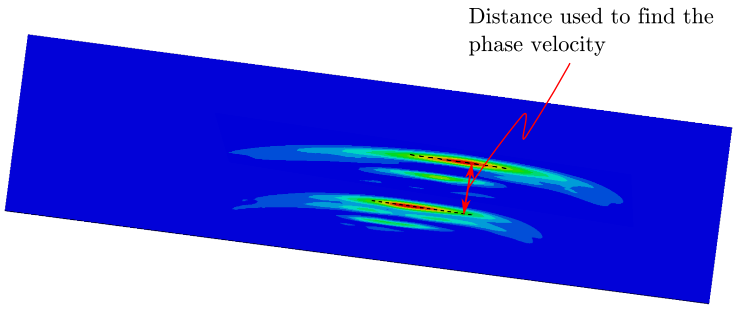
In the next subsection we find the phase velocity, energy velocity and skew angle and compare them with the analytical results.

3.3.1 Wave speed and skew angle study

Due to the nature of the Finite-sized Wave Model, we cannot use the same procedure as we did for the Plane Wave Model to find the wave speed. Also, we wanted to find the norm of the energy velocity $||\{\mathbf{c}\}^e||$ and the skew angle ψ. Let us denote the norm of the energy velocity as simply c^e from now on.



(a) *The skew angle ψ can be found by drawing a line between the center of the source and the center of the wave. The position of the center of the wave is assumed to be at approximately the point of maximum stress.*



(b) *The strain energy density at two time instances plotted on top of each other. As seen in the figure, we draw two dashed lines parallel to the bottom and top surface, and position them at the points of maximum stress. The distance traveled by the wave in direction $\{\mathbf{n}\}$ gives the phase velocity.*

Figure 3.16: *Field plots of the strain energy density for $\theta = 7.5°$. With knowledge of the skew angle ψ and the phase velocity c_{SV}, we find the energy velocity from $c^e = c_{SV}/cos(\psi)$.*

We found the sought-after values graphically, as indicated in Figure 3.16. The distances were measured in pixels in the graphics editor Inkscape 0.48, then converted to meters afterwards.

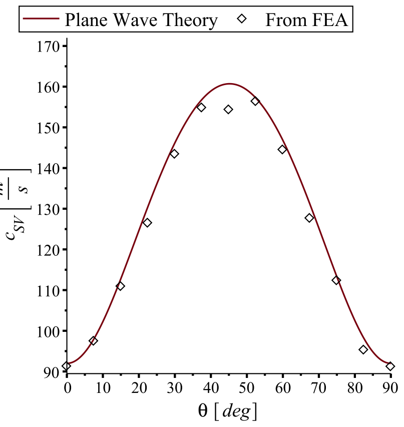
In Figure 3.17a and 3.17b the phase velocity c_{SV} obtained from the Finite-sized Wave Model is studied. Similarly as for the Plane Wave Model, we found that the wave speeds from FEA were slightly less than predicted by theory for all θs. This agrees with the result presented by Rose [23, Table 14-4]; he found that the experimental phase velocities were in general slightly lower than the theoretical ones. For the Plane Wave Model we obtained an average error of 1.0%, and for the current model we obtained an average error 1.8%. We can therefore say that the procedure we used for finding the phase velocity is satisfactory. However, we can see from Figure 3.17b that there is a good spread in the errors. This may be explained by the difficulty in being very accurate and consistent when finding the values graphically. Note that 'error' means disagreement between FEA results and the exact solution of the mathematical model, which in this case is plane wave theory. In other words, the error says nothing about how well the mathematical model represents reality.

The study of the energy velocity norm c^e is shown in Figure 3.17c and 3.17d. Again, the velocity obtained from FEA is in general lower than predicted by theory, as one might expect from numerical dispersion. As mentioned, we found c^e from the formula

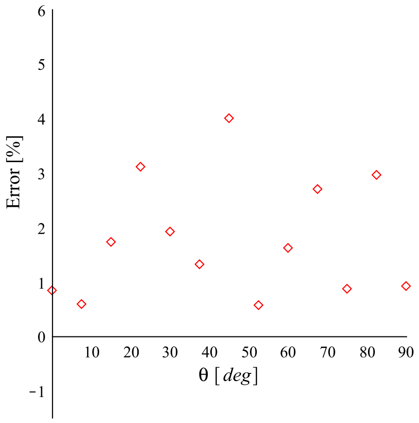
$$c^e = \frac{c_{SV}}{cos(\psi)}.$$

Since we introduce another source of error through ψ, it is not surprising that the average error of c^e (3.0%) is larger than for c_{SV}.

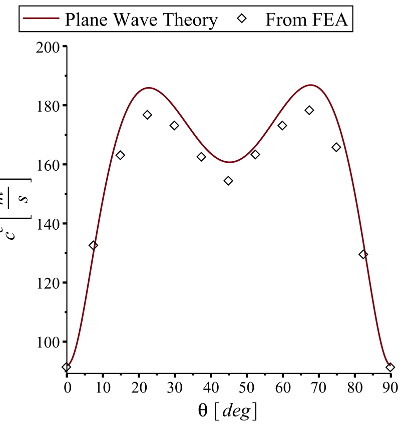
Finally, the study of the skew angle ψ is presented in Figure 3.18. We can see that the theory slightly over-predicts the absolute value of ψ for most of the θs. The source of this discrepancy has not been found. Nevertheless, the theory gives a good estimate of the skew angle.



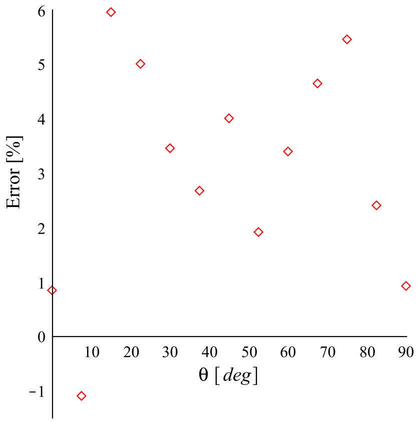
(a) *Wave speed c_{SV} versus the angle θ.*



(b) *Relative error corresponding to a).*

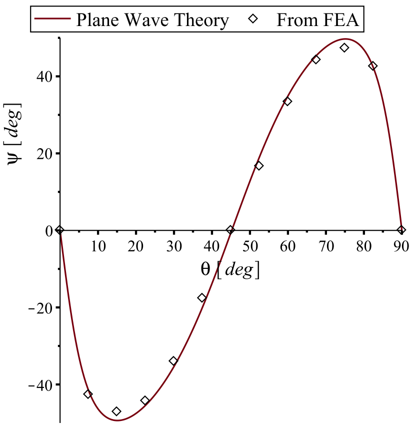


(c) *Energy velocity c^e versus the angle θ.*

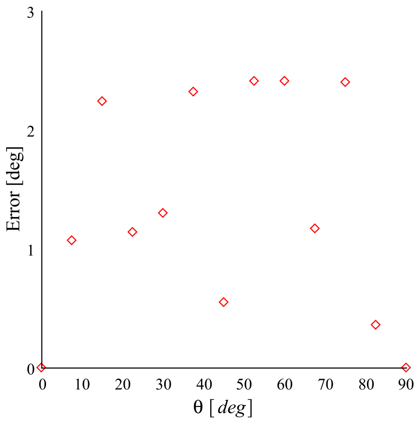


(d) *Relative error corresponding to d).*

Figure 3.17: *Result from the Finite-sized Wave Model. The numerically obtained wave speeds are plotted together with the analytical solutions from plane wave theory. The theory seems to give good estimates of c_{SV} and c^e, although slightly too high.*



(a) *Skew angle ψ versus the angle θ.*



(b) *Absolute error corresponding to a). Note that the error is calculated by $|\psi_{theory} - \psi_{FEA}|$.*

Figure 3.18: *Result from the Finite-sized Wave Model. The numerically obtained skew angles are plotted together with the analytical solution. A positive skew angle means that $\{\mathbf{c}\}^e$ is rotated counter-clockwise with respect to $\{\mathbf{n}\}$, see Figure 2.15. The plane wave theory predicts the skew angle well.*

3.4 Curved Model

With the model presented in this section, we are one step closer to the Truncated Ellipsoid Model, introduced in the next section. We named the present model the Curved Model, because we wanted to investigate the effects of curvature on shear wave speed. In this section, we first study the case of isotropic material, because we then can compare the results directly with plane wave theory. Next, we implement the transversely isotropic material, and study the effects of anisotropy. In the last subsection, we investigate the effects of different orientations of the load.

The Curved Model has a shape based on elliptical curves in the x-y plane, as seen in Figure 3.19. In the studies shown later in this section, we vary the dimension a to study the effect of curvature. Boundary conditions that are not shown in this figure are that the sides of the model are only allowed to translate in z direction. More specifically, $u_x(x, y, 0, t) = u_y(x, y, 0, t) = 0$ and $u_x(x, y, d, t) = u_y(x, y, d, t) = 0$. These boundary conditions were implemented in the spirit of the Plane Wave Model, where we also allowed material displacement in only one direction at the boundaries. As we have seen, this gave a clear and plane wave front for the Plane Wave Model, and we wanted the same for the current model.

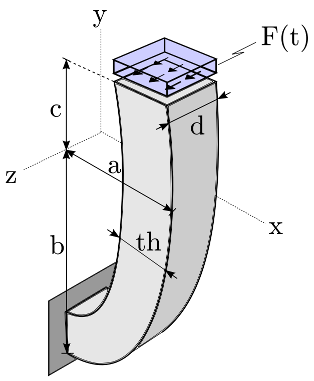


Figure 3.19: *An illustration of the Curved Model. The lengths a and b represent the minor and major axes, respectively, of the elliptical outer surface. The inner surface is parallel to the outer surface in the sense that is has the minor and major axes $a - th$ and $b - th$, respectively. The length c determines where the ellipse is truncated. The thickness th and depth d are constant. A uniformly distributed shear load is applied to the top surface, which is parallel to the x-z plane. In other words, $T_{yz}(x, c, z, t) = F(t)$. At the bottom surface the model is fixed, which means that $u_x(0, y, z, t) = u_y(0, y, z, t) = u_z(0, y, z, t) = 0$ and $u'_x(0, y, z, t) = u'_y(0, y, z, t) = u'_z(0, y, z, t) = 0$.*

It is useful to compare the results from the Curved Model with a model where curvature effects are absent. This helps us determine what is caused by curvature, and what is not. We therefore made the Straight Model shown in Figure 3.20.

In Section 2.1.2, we saw how the muscle fiber orientation varies through the heart

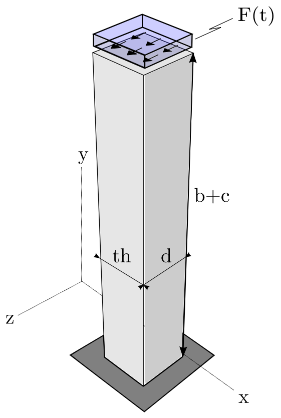


Figure 3.20: *This model was made to fully exclude potential effects from curvature. We will refer to this model as the Straight Model. The only difference with the Curved Model is that the elliptically shaped curves are now replaced by straight curves. Note that, similarly as for the Curved Model, the boundary conditions $u_x(x, y, 0, t) = u_y(x, y, 0, t) = 0$ and $u_x(x, y, d, t) = u_y(x, y, d, t) = 0$ are applied.*

Table 3.2: *Dimensions of the Curved Model. The dimensions correspond to those found in the Truncated Ellipsoid Model.*

	a	b	c	d	th
[mm]	10 to 30	45	15	10	10

wall. The Curved Model is therefore partitioned such that we can vary the fiber orientation through the thickness of the model, see Figure 3.21. We can see that there are 10 partitions through the thickness. However, with the appended Python script, the user can easily change the number of partitions. More layers make the material orientation transition smoother, but requires the model to have a finer mesh.

We used C3D8R elements, which are continuous three-dimensional eight-noded reduced integration elements. Thus, the elements only have one integration point. The arguments for reduced integration are the same as for the two-dimensional case. Obviously, a homogeneous mesh is impossible to achieve, and we therefore do not have only one element size. Nevertheless, a fine mesh density leads to a nearly homogeneous mesh. An approximate element size was calculated from Equation (2.44), and Abaqus meshed the geometry based on this size within certain tolerances.

The mass density, load amplitude and duration are the same as before. To study the effect of curvature on wave speed, we varied the minor axis a in Figure 3.19 and kept the other dimensions constant. The dimensions are listed in Table 3.2.

We first performed studies where an isotropic material was implemented, and then

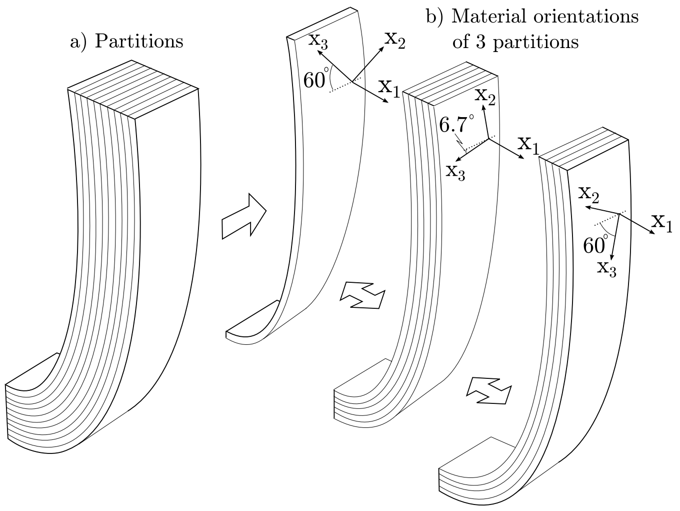


Figure 3.21: *Illustration of: a) the 10 partitions of the model; b) the material orientation that varies through the thickness of the model. As before, the x_3 direction gives the fiber direction. A distinct material orientation is assigned to each partition such that the fiber orientation varies from $+60°$ to $-60°$ with respect to the circumferential direction. The x_1 direction is normal to the surfaces at all points. The Straight Model is partitioned in a similar manner.*

we assigned the transversely isotropic material to the model.

3.4.1 Isotropic material

When an isotropic material is assigned to the whole model, the material orientations are obviously irrelevant. We wanted the isotropic material to give shear wave speeds in the same order as for the transverse isotropic material. Based on this, we used the following material properties:

- Young's Modulus - $E = 31.588$ [MPa]
- Poisson's ratio - $\nu = 0.49$
- Mass density - $\rho = 1060$ [kg/m^3]

With a Poisson's ratio close 0.5, the material is nearly incompressible. This is realistic, because biological tissues consist of mainly water, which is incompressible. The Young's modulus was chosen such that the shear wave speed for a plane wave became

$$c_t = \sqrt{\frac{E}{2\rho(1+\nu)}} = \sqrt{\frac{31.588 \cdot 10^6}{2 \cdot 1060 \cdot (1+0.49)}} = 100.0 \quad \text{[m/s]}.$$

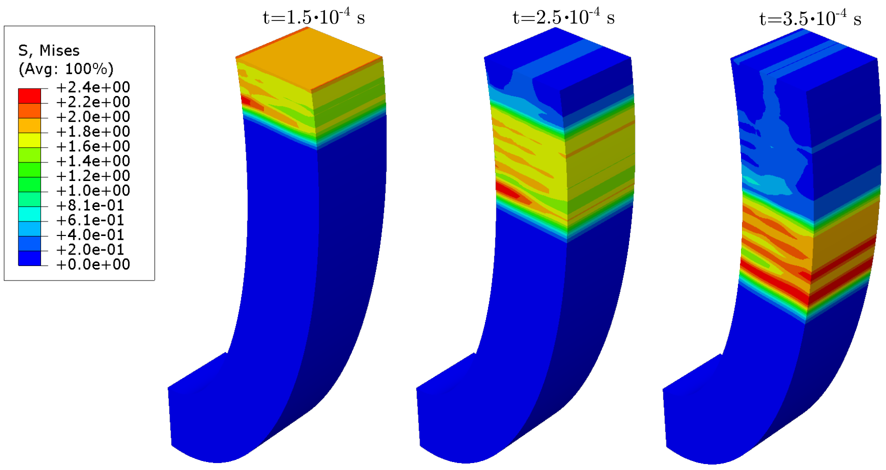


Figure 3.22: *The stress distribution at three time instances when the isotropic material is implemented. The wave propagation can be seen clearly. Here* $a = 25$ *mm.*

Field plots of the Mises stress at three time instances are given in Figure 3.22. Again, the Mises stress is useful, because we do not have to consider local and global coordinates. We only have to assume that the main contributors to the stress measure are the shear stress components, which is likely given the nature of the model. From the figure we can clearly see the stress wave propagating from the top surface and downwards. The wave front seems to remain plane through the domain of interest. With the result from the Plane Wave Model in mind, we can therefore expect the obtained wave speeds to be close to those found from plane wave theory. Thus, we can use the theory to validate our results.

Wave speed and mesh study

A similar procedure as for the Plane Wave Model was performed to find the wave speed; we observed the stress over time at two nodes. The position of the nodes is given in Figure 3.23. With two nodes we can measure the wave speed between the top surface and Node 1 and between Node 1 and Node 2.

We defined the obtained wave speed between the top surface and Node 1 as c_1, and the obtained wave speed between Node 1 and Node 2 as c_2. In the first study we used $N = 10$ in Equation (2.44), which gave the result shown in Figure 3.24a. We can see from this figure that the numerically obtained wave speeds are close to the analytical solution. However, there is a noticeable difference between c_1 and c_2. To investigate this further, we increased the mesh density by setting $N = 20$ and $N = 30$. The result from the studies with these mesh densities are shown in Figure 3.24b and 3.24c. The difference between c_1 and c_2 is clearly less for the case of increased mesh density. It is therefore likely that there is a difference only due

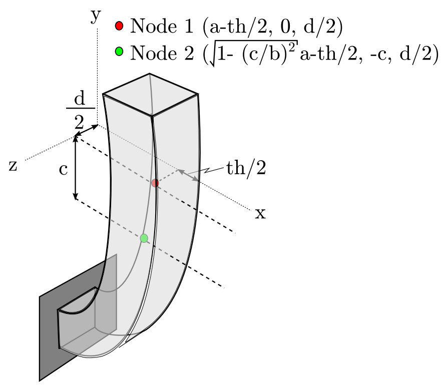
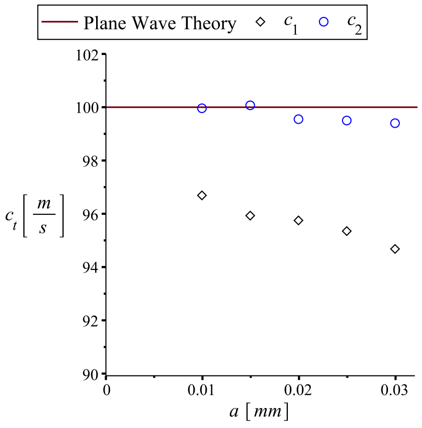

y
● Node 1 (a-th/2, 0, d/2)
● Node 2 ($\sqrt{1-(c/b)^2}$a-th/2, -c, d/2)
d/2
z
c
th/2
x

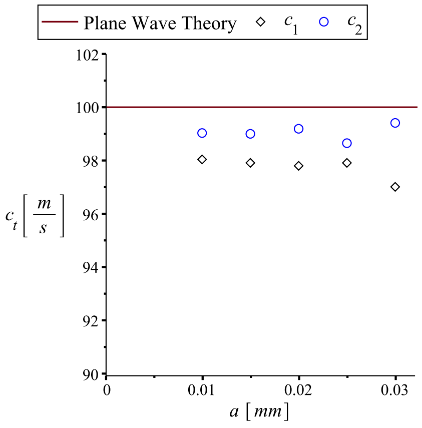
Figure 3.23: *The position of the nodes where we observed the stress over time when the isotropic material was implemented. The nodes are placed according to the coordinates shown in the figure.*

to numerical inaccuracy. Note that the wave speeds seem close to independent of a when $N = 20$ and $N = 30$.

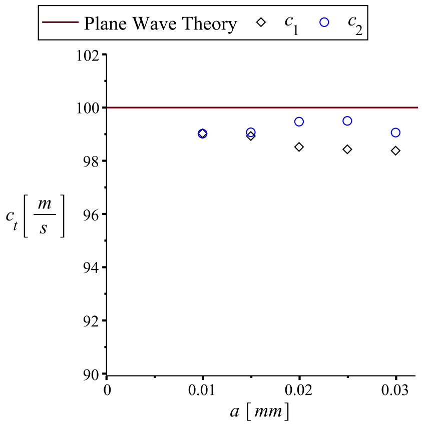
We implemented the isotropic material into the Straight model as well, and investigated if we obtained a difference between c_1 and c_2 without curvature. With $N = 20$ we obtained that $c_1 = 98.2$ m/s and $c_2 = 99.5$ m/s. We can thus conclude that c_1 and c_2 do not differ due to curvature. Nevertheless, it is useful to know that the wave speeds we find can depend on where we measure them.



(a) *N=10, which gives approximately 6 000 elements*



(b) *N=20, which gives approximately 45 000 elements*



(c) *N=30, which gives approximately 150 000 elements*

Figure 3.24: *The result from the wave speed study when an isotropic material is applied to the Curved 3D model. The difference between c_1 and c_2 is less when a finer mesh is implemented, and both wave speeds seem to converge to approximately 99 m/s for a increase in mesh density. Similarly as for the Plane Wave Model and the Finite-sized Wave Model, the measured wave speed is slightly lower than the theoretical one. The wave speeds seem to be less affected by a change in curvature for a finer mesh.*

3.4.2 Transversely isotropic material

With the transversely isotropic material, the shear wave speed varies through the thickness of the model. The traction load is directed such that mostly the shear vertical wave mode is excited. In the Section 3.4.3, we excite the shear horizontal wave mode. Field plots of the Mises stress is shown in Figure 3.25. Compared to the isotropic case, the wave propagation is now much more complex. Since we have several layers with different material orientations, we can say that the wave propagation consists of several waves that propagate with different speeds. The layers through the thickness create interfaces where refraction and reflection of the waves might occur. Also, at interfaces mode conversion can occur, which means that the energy of the waves can be converted into shear, longitudinal and interface wave modes [23]. Another aspect that complicates the behavior is that we have finite boundaries that create surfaces where surface waves can arise. These waves are commonly known as Rayleigh waves, which have their own wave speeds that also depend on propagation direction [23]. Nevertheless, we do not consider these effects in the remainder of the thesis, and assume that the shear vertical wave mode is dominant.

Obviously, the theory presented in Chapter 2 is not sufficient to accurately predict the wave propagation in the Curved Model. However, plane wave theory can be used to roughly estimate some of the behavior. We know that the material orientation varies from $-60°$ to $60°$ through the thickness of the material. A plot of the theoretical shear wave speed c_{SV} in this interval is provided in Figure 3.26. We can see the tendency of this wave speed variation through the thickness in Figure 3.25.

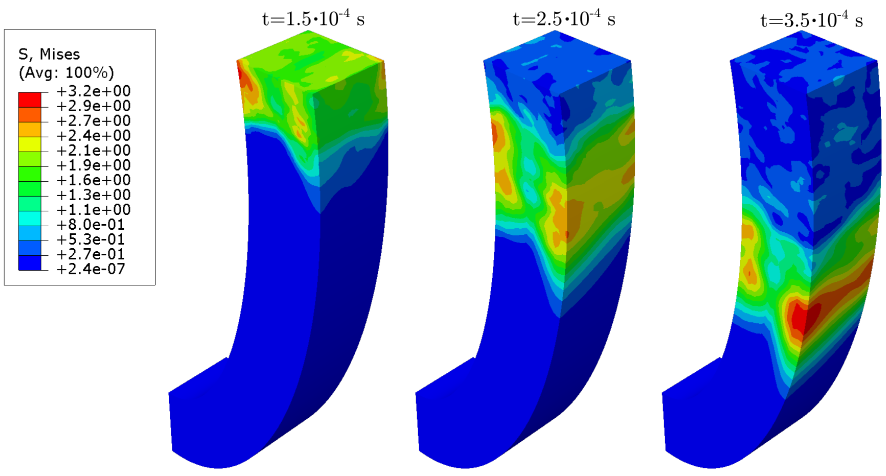


Figure 3.25: *The stress distribution at three time instances when the transversely isotropic material is implemented. The effect of the layers with different material orientations can be seen from that the wave speed varies through the thickness. Here* $a = 25$ *mm.*

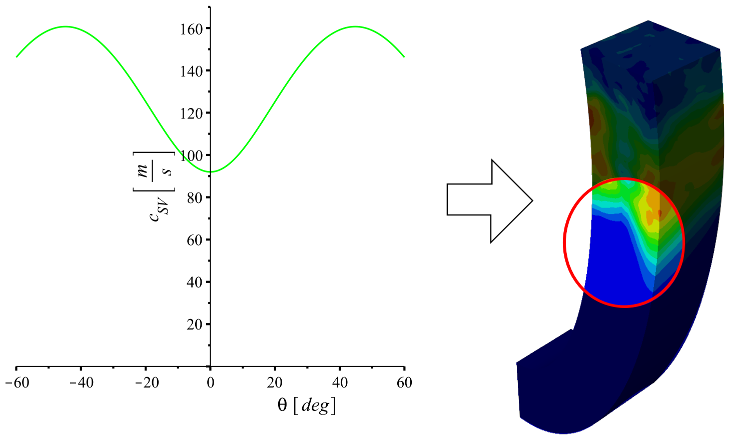


Figure 3.26: *Left: shear wave speed c_{SV} variation in the x_1-x_3 plane and the x_2-x_3 plane obtained from plane wave theory. The plot corresponds to the velocity surface in Figure 3.3. Right: one of the plots from Figure 3.25. We can see the resemblance between the left plot and the wave front in the right plot.*

We observe from the field plots in Figure 3.25 that the stresses are largest near the inner and outer surface. This can be explained by the following simple reasoning: higher stiffness causes higher wave speeds, and from structural mechanics, we know that the stress is normally largest where the structure is stiffest.

In the next paragraphs, we demonstrate how an average value of the shear wave speed can be estimated, and we perform again a study of the effect of curvature. It is useful to have knowledge about the dependence of the wave speed on curvature, because the Truncated Ellipsoid Model is also based one elliptical curves.

Wave speed study

We found the shear wave speeds by observation of the stress versus time at the nodes shown in Figure 3.27. We chose to observe 3 nodes in the thickness direction, because we know the wave speed varies in this direction. From the isotropic case, we saw that c_1 and c_2 could differ slightly. However, here we do not consider the effect that the measured wave speed can differ along the length.

The wave speed was calculated as before; we estimated when the center of the wave passed the observation nodes. And with knowledge of the distance traveled, we found the wave speeds. However, with the current model, it is slightly more difficult to determine the time when the center of the wave passes the nodes. A plot of the stress versus time at the three nodes is displayed in Figure 3.28a. We can see that there are relatively large oscillations in the stress. As have been discussed earlier, oscillations in the response are expected. See for example [18], where they study wave propagation in a circular ring. They obtained large oscillations in the response both from FEA and experimental analyses. In Figure 3.28b, the stress response has been run through the Butterworth filter with a cutoff frequency of 20,000 Hz. The cutoff frequency is based on the frequencies of the oscillations that we can see in Figure 3.28a. With the filtered plots, it is easier to be consistent with the wave speed estimation. Note, however, that the filtering does not necessarily give more correct wave speeds.

We termed the wave speeds c_a, c_b, and c_c and they correspond to the nodes Node 1a, Node 1b and Node 1c, respectively. We also calculated the average wave speed, which is termed c_{avg}. The result from the wave speed study is shown in Figure 3.29. We see that the wave speeds lie within the range 110 to 122 m/s. This agrees to some extent with what is predicted by plane wave theory, because in Figure 3.26 we

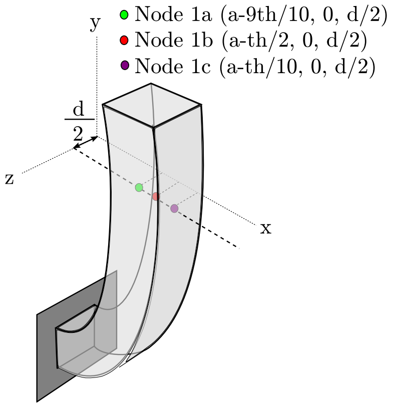
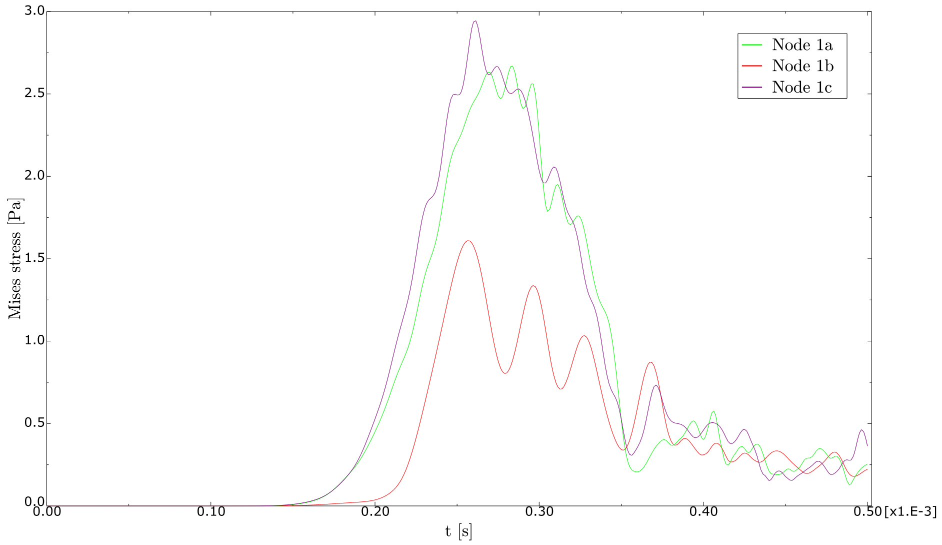
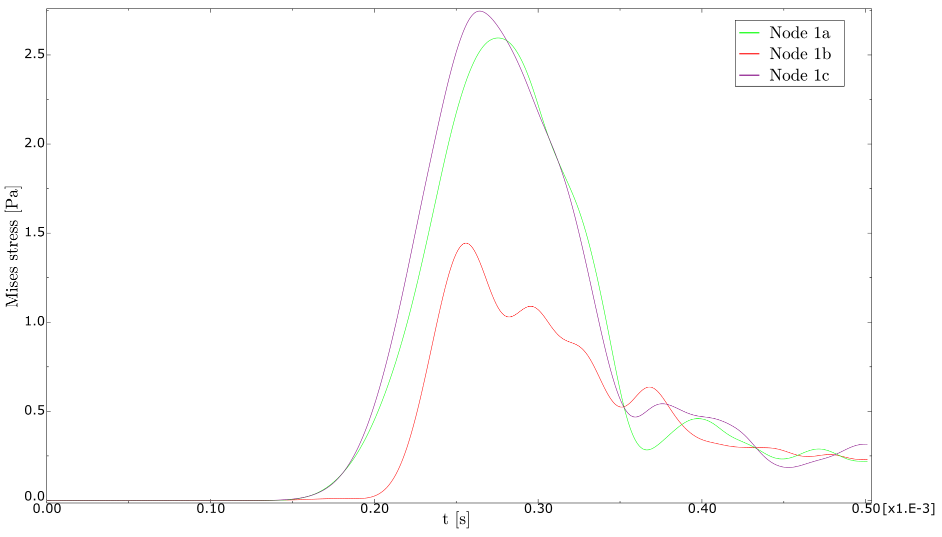

● Node 1a (a-9th/10, 0, d/2)
● Node 1b (a-th/2, 0, d/2)
● Node 1c (a-th/10, 0, d/2)
y
$\frac{d}{2}$
z
x

Figure 3.27: *The position of the nodes where we observed the stress over time when the transversely isotropic material was implemented. The nodes are placed according to the coordinates shown in the figure.*



(a) *Mises stress at the three nodes.*



(b) *Filtering applied to the plots in (a).*

Figure 3.28: *Stress at the three nodes versus time. We can find the time when the wave passes the nodes, and then we can calculate the wave speeds. Here $a = 25$ mm and $N = 20$.*

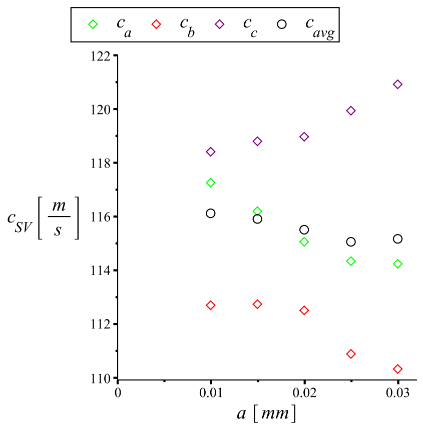


Figure 3.29: *A plot of the measured wave speeds for different values of a when the transversely isotropic material is implemented. There seem to be a slight dependence on the value of a. However, the average wave speed c_{avg} is nearly constant. Here we chose $N = 20$ based on that we observed a good accuracy with this mesh density when the isotropic material was implemented.*

see that the wave speed varies from 92 to 160 m/s. It is probably due to interaction between the waves of the different layers that there is less variation in the wave speed for the FEM model. In other words, we observe a smearing of the wave front. To investigate this further, we implemented the transversely isotropic material in the Straight Model as well. A summary of the result is given in Table 3.3. Note that since the observation nodes are placed between two layers, the theoretical wave speeds at those nodes are the average of the wave speeds from the two neighboring layers.

Table 3.3: *A comparison of shear wave speeds from the Curved Model, the Straight Model and plane wave theory for the case of transversely isotropic material.*

Models	c_a	c_b	c_c	c_{avg}	
Curved Model (a=25 mm)	115.0	111.3	118.6	114.9	[m/s]
Straight Model	118.7	113.2	118.7	116.8	[m/s]
Plane wave theory	153.7	96.8	153.7	134.7	[m/s]

There are several comments that can be made from Table 3.3:

- There is only a difference between c_a and c_c for the Curved Model. Thus, we can conclude that the curvature causes a difference between c_a and c_c.
- The Curved Model and the Straight Model gave relatively similar wave speeds, which means that the curvature affects the wave speeds only slightly. This is what we can conclude also from Figure 3.29.
- For both the FEM models, we see that the wave speed variation is far less

significant than predicted by plane wave theory. In other words, the difference between c_a, c_b and c_c, obtained from FEA, is not as pronounced as in theory. From this observation we conclude that plane wave theory is not sufficient to accurately predict wave propagation in anisotropic layered objects.

Since the wave speeds from the Curved Model agreed to some extent with plane wave theory, we can say that the model and the wave speed measurement procedure gave reasonable results. Due to the complexity of the model it is difficult to find a physical explanation of why c_a is measured lower than c_c. We see from Figure 3.29 that the difference increases for an increase in a. We can also note that the wave speeds are more dependent on a for the anisotropic case than for the isotropic case. Nevertheless, the average wave speed c_{avg} is nearly constant for variation in a. As seen in Section 3.5, this is a relevant observation for the Truncated Ellipsoid Model.

In the next subsection, mostly for illustrative purposes, we excite the shear horizontal wave mode rather than the shear vertical wave mode, and study the response.

3.4.3 Excitation of shear horizontal wave mode

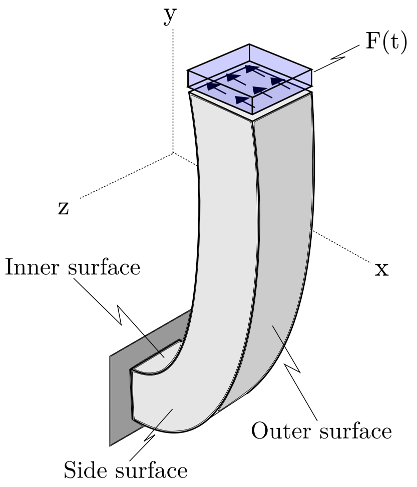
So far, we have only excited the shear vertical wave mode. In other words, the material displacement was mainly in the z-direction. In this subsection, the material is still transversely isotropic, but the orientation of the load is directed such that the shear horizontal wave mode is excited, *i.e.* the material displacement is mainly in the x-direction. Recall that the material displacement directions of the different wave modes are always mutually orthogonal. An illustration of the geometry and load is given in Figure 3.30 together with some analyses of the results.

Recall that previously the side surfaces were only allowed to translate in z-direction, *i.e.* $u_x(x,y,0,t) = u_y(x,y,0,t) = 0$ and $u_x(x,y,d,t) = u_y(x,y,d,t) = 0$. Since the orientation of the load is now changed, it is reasonable to change these boundary conditions accordingly. We therefore restrained the inner and outer surface to only move in the x-direction, *i.e.*

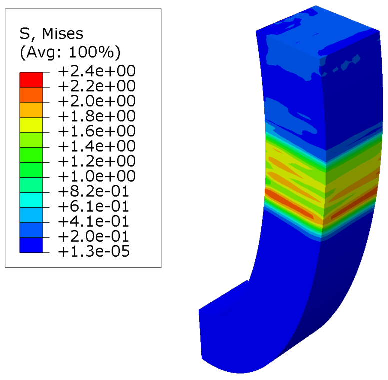
$$u_y\left(\sqrt{1-\left(\frac{y}{b}\right)^2}a, y, z, t\right) = u_z\left(\sqrt{1-\left(\frac{y}{b}\right)^2}a, y, z, t\right) = 0,$$

$$u_y\left(\sqrt{1-\left(\frac{y}{b-th}\right)^2}(a-th), y, z, t\right) = u_z\left(\sqrt{1-\left(\frac{y}{b-th}\right)^2}(a-th), y, z, t\right) = 0.$$

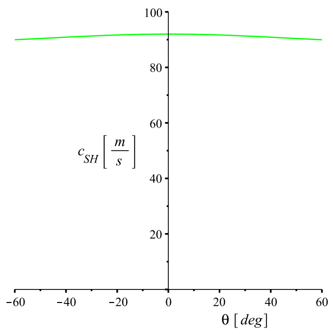
From Figure 3.30d and the wave speed measurement procedure we found the shear wave speeds to be $c_a = c_b = c_c = 90.5$ m/s. The theoretical average is 90.6 m/s. Thus, similarly as for when the isotropic material was implemented, we found a shear wave speed close to the theoretical value. This is expected, because there is almost no wave speed variation through the thickness for the shear horizontal wave mode, see Figure 3.30c.



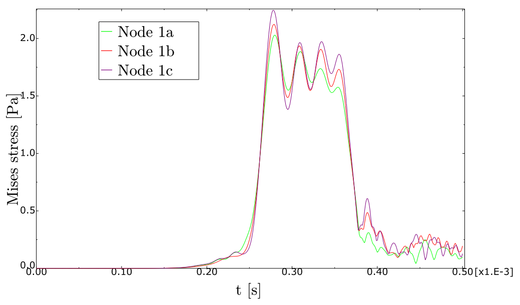
(a) *The Curved Model, where the orientation of the traction load is rotated 90° with respect to the case in Figure 3.19.*



(b) *Field plot of the Mises stress at $t = 3.5 \cdot 10^{-4}$. There is almost no wave speed variation visible through the thickness. We can expect this due to the plot in (c).*



(c) *Theoretical shear wave speed variation in the interval $\theta = -60°$ to $\theta = 60°$. The wave speed varies from 89.97 m/s to 91.99 m/s.*



(d) *The Mises stress at the three nodes. As we can see, the stress responses are nearly the same, and give therefore the same wave speeds.*

Figure 3.30: *The Curved Model with the transversely isotropic material and a load that excites the shear horizontal wave mode is shown in (a). A plot of the theoretical shear wave speed variation from plane wave theory is given in (c). The stress response is shown in (b) and (d).*

The Curved Model could represent the septum in the heart. In the next section, we introduce another model of the septum and the rest of the left ventricle. If we obtain similar wave speeds with this next model, we can assume that the Curved Model is a good representation of septum.

3.5 Truncated Ellipsoid Model

The left ventricle is often modeled as a truncated ellipsoid in literature [14, 26]. We therefore created the Truncated Ellipsoid Model with the dimensions shown in Figure 3.31. The model is partitioned in a similar manner as the Curved Model, see Figure 3.21. This means that the material orientation varies from $+60°$ in the epicardium to $-60°$ in the endocardium. Load duration and mass density are the same as before, and we have used N=10 in Equation (2.44) to find an approximate element size. With N=10, about 130 000 elements were generated. Based on the mesh study of the Curved Model, $N = 20$ or $N = 30$ would probably give more accurate results, but then the computation time becomes impractically long. The element type is the same as for the Curved Model.

With this model, we wanted to simulate the wave propagation down the septum due to AVC. There are obviously many simplifications in the Truncated Ellipsoid Model with respect to reality. Let us therefore briefly discuss some of the simplifications and assumptions. In reality, the load likely has components in all spatial directions. However, since wave modes propagate independently of each other, it is acceptable to have load that mainly excites the shear vertical wave mode down the septum. This makes it easier to identify the shear wave. Another simplification is that we have assumed that the thickness is constant. In [26], they show that the thickness varies, and this might affect the wave propagation. Nevertheless, we do not consider thickness variation, because it simplifies both the model and analysis. We have also assumed that the dimensions and stiffness is constant during the wave propagation. This is a fair assumption, because we extract the wave speed from a short period of time relative to the contraction time of the heart (this assumption is discussed in more detail in Section 4.2). Also, we saw in the previous section that it can be acceptable to neglect the effect of curvature on the wave speed.

The dimensions of the model can be varied easily with the appended Python script. However, the dimensions we have used to study this model are given in Table 3.4.

Table 3.4: *Dimensions of the Truncated Ellipsoid Model. The dimensions are based on those found in [26].*

	a	b	c	th
[mm]	25	45	15	10

We wanted to find an average shear wave speed such that we could make an estimate of the stiffness with the help of Equation (2.16), which is restated here:

$$c_t^2 = \frac{E}{2\rho(1+\nu)}.$$

In other words, we assumed isotropy when the stiffness of the heart was estimated. The density ρ and Poisson's ratio ν were assumed known, and thus the Young's

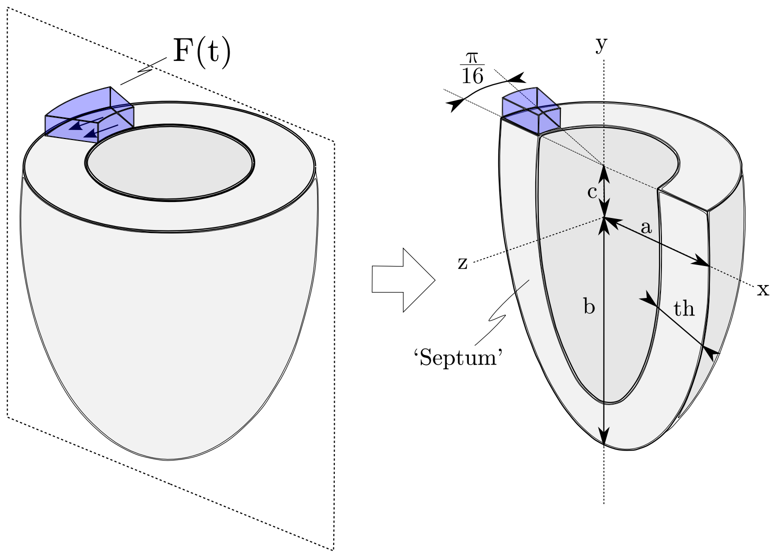


Figure 3.31: *The Truncated Ellipsoid Model. The figure on the right shows a cross-section of the model and the dimensions used. The thickness is constant. A uniformly distributed traction load is applied to a small part of the top surface, and the part beneath the load is the septum. To avoid potential rigid body motions, the boundary condition* $u_i(0, -b, 0, t) = u_i'(0, -b, 0, t) = 0$ *(*$i = 1, 2, 3$*) is applied.*

modulus E can be determined. We found the average wave speed with a similar approach as for the Curved Model; we observed the stress at the nodes displayed in Figure 3.32.

As we can see in Figure 3.33, the wave propagation is now very complex. In addition to the effects we discussed for the Curved Model, waves can now propagate in several directions, not only down the septum. We see in this figure that there are some small disturbances, represented by the dark blue color, that propagate relatively rapidly. This is the wave front of longitudinal waves, which propagate approximately 15 times faster than the shear waves. Another observation that can be made: the first time instance displays clearly the shear wave speed variation through the thickness, *i.e.* a shape that resembles Figure 3.26.

Due to the complexity of the problem, it was difficult to come up with a robust procedure that could be used to find the shear wave speeds. We could not use the Mises stress, because there were significant contributions to the measure from longitudinal stress components. We therefore observed the shear stress component T_{yz} (S_{23} in Abaqus) at the nodes. This is the largest shear stress component and it is computed with respect to the global coordinate axes. Note that the stress can now be both positive and negative, as opposed to the Mises stress. The response proved to be very oscillatory; an example is the stress response shown in Figure 3.34. It is obviously hard to discern when the center of the wave passes the

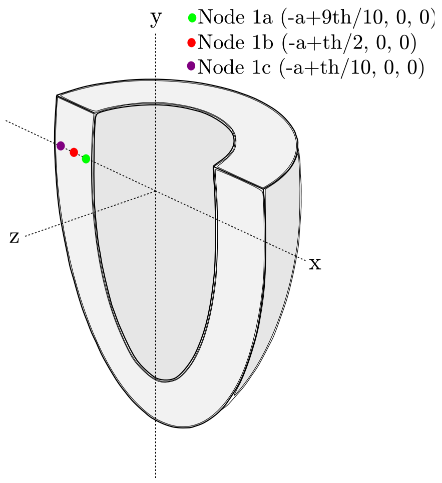


Figure 3.32: *The position of the nodes where we observed the stress over time. Three nodes allows us to calculate an estimate of the average wave speed through the thickness.*

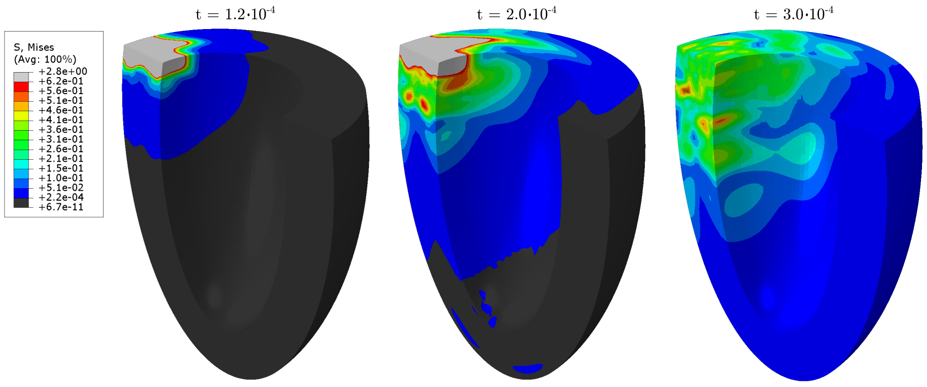


Figure 3.33: *A cross section of the model with field plots of the Mises stress at three time instances. The gray area is where the stress is 2.8 Pa or higher, while the black area is where the stress is* $6.7 \cdot 10^{-11}$ *Pa or lower. In other words, the black area is undisturbed area. The disturbance from the load propagates in all directions.*

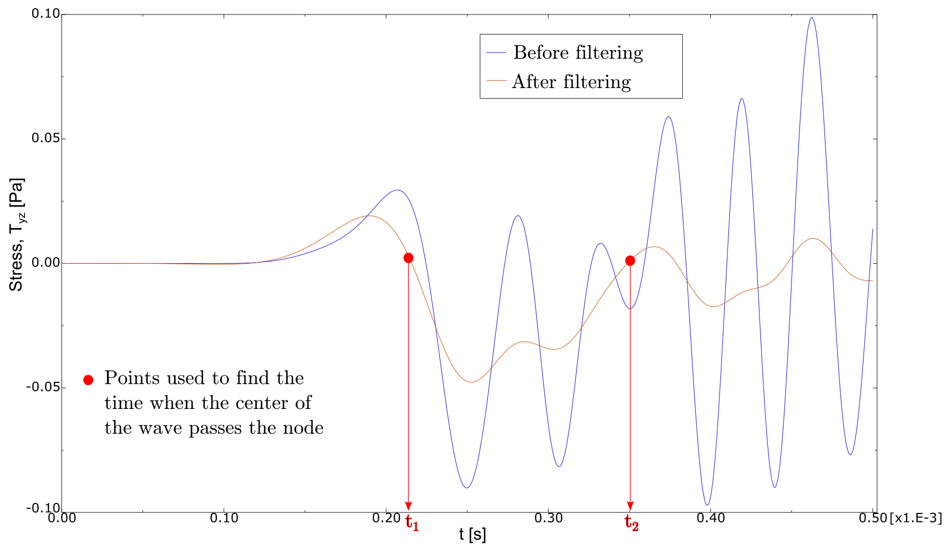


Figure 3.34: *The shear stress T_{yz} at Node 1b, before and after the Butterworth filter was applied. We find the time t_1 and t_2 from when the stress is approximately zero. As before, the average of t_1 and t_2 gives an estimate of when the center of the wave passes the node, which allows for calculation of the wave speed.*

Table 3.5: *Shear wave speeds obtained from the Truncated Ellipsoid Model. The result is compared with what was found from the Curved Model when $a = 25$ mm.*

Models	c_a	c_b	c_c	c_{avg}	
Truncated Ellipsoid Model	105.8	112.9	117.9	112.2	[m/s]
Curved Model (a=25 mm)	115.0	111.3	118.6	114.9	[m/s]

node, because the oscillations in the response are relatively large. However, the figure also displays how the Butterworth filter brings forth a shape that resembles an impulse response. The cutoff frequency was set to 10,000 Hz, based on that the load duration is $1 \cdot 10^{-4}$ seconds. With this cutoff frequency, we filtered out most of the response with a frequency higher than the impulse load.

An estimate of the shear wave speeds c_{SV} were found by the procedure described in the previous paragraph and Figure 3.34. The wave speeds are termed similarly as for the Curved Model. The result is shown in Table 3.5. We see from this table that the wave speeds c_b and c_c match well with those found from the Curved Model, and that c_a is noticeable lower for the Truncated Ellipsoid Model. We have not found a good explanation for why there is a relatively large discrepancy for only one of the wave speeds. Further investigation is required, but is not performed here.

Since the load is applied to a small area of the model, the problem is a finite-sized wave problem. We saw that the phase velocities were approximately the same for the Plane Wave Model and the Finite-sized Wave Model. Thus, it is reasonable that they should be nearly the same for the Curved Model and the Truncated Ellipsoid Model. With the result in Table 3.5, we can therefore assume that we obtained reasonable wave speeds. Since the wave speeds for the Curved Model and the Truncated Ellipsoid Model are nearly the same, we can conclude that the former model is a decent representation of the septum as well.

With an average wave speed, we can make an estimate for the Young's Modulus if we consider the material isotropic:

$$E = 2\rho(1+\nu)c_{avg}^2 = 2 \cdot 1060 \cdot (1+0.49) \cdot 112.2^2 = 39.765628 \cdot 10^6 \quad [\text{Pa}],$$

where we have used the same mass density and Poisson's ratio as in Section 3.4. It will be interesting to see whether we obtain a shear wave speed c_t that match c_{avg} when we implement the obtained E into our material. We therefore implemented an isotropic material with the material parameters from the equation above. With the same procedure as for the anisotropic case, we obtained $c_t = 111.4$ m/s. Again we see that the wave speed obtained from FEA is slightly lower than the theoretical value, which is in this case $c_{avg} = 112.2$ m/s. This final result gives confidence in the procedure that produced the wave speeds.

4 Discussion

The results have already been discussed to some extent in the previous chapter. In this chapter, we discuss some of the assumptions and simplifications we have made throughout the thesis. We also have some suggestion for future work and studies.

4.1 Material and nonlinearities

As already discussed in Section 3.1, the stiffness coefficients of the transversely isotropic material are found from a formalin fixed myocardium. Hoffmeister *et al.* suggest that there is a significant change in the shear wave speed due to formalin fixation [13]. We obtained shear wave speeds around 100 m/s with the transverse isotropic material, while Brekke *et al.* observed *in vivo* a wave speed down septum at approximately 5 m/s [6]. This indicates that the shear stiffness of the material we have used is too high. Nevertheless, as argued in Section 3.1, it was not important to produce realistic results. We were more interested in the implications that can be made from the results, *e.g.* that FEM can handle wave propagation in anisotropic media.

In this thesis, we have assumed linear elasticity, which is an acceptable assumption if the applied perturbations are relatively small. The stress and strain in the heart wall due to contraction of the muscle is likely relatively high compared to what we get from AVC. Thus, the small perturbation assumption is valid. In other words, we have assumed that the stress-strain relationship can be approximated linear for a small region in the nonlinear material curve, see Figure 4.1. Nevertheless, it might be interesting to implement nonlinear materials in future FEM models. Holzapfel *et al.* mentions several models available in literature of the *nonlinear* elasticity of the myocardium. They are isotropic, transversely isotropic or orthotropic [14]. Modeling anisotropy through defining strain-energy functions, which allow nonlinear material behavior, can be an alternative way to implement the material. This possibility should be considered if someone were to continue with the work in this thesis.

Another potential source of nonlinearity is geometrical nonlinearities. We know

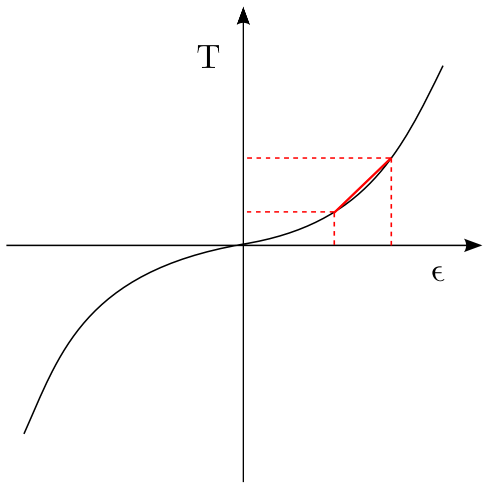


Figure 4.1: *A schematic illustration of a typical stress-strain relationship for biological tissue. The red line illustrates how we can assume linearity for a small portion of the curve, and still be fairly accurate.*

that biological tissues allow relatively large elastic deformations. It would therefore be relevant to look into the effects of geometrical nonlinearity. For example, if the deformations that come from the wave propagation are relatively large, then the effect might be relevant to consider. Another example is the pretension we have in the heart wall due to contraction, which can induce increased stiffness. Chen *et al.* show that pretension affects the shear wave when MRE measurements are performed *in vivo* [7].

An advantage with linear theory is that we do not have to consider the amplitude of the load. Throughout this thesis, we used a unit load, which means 1 N/m for the 2D models and 1 N/m^2 for the 3D models. If we increased the load, the response would have the same shape with respect to time. The only difference would be that the amplitude of the response increased with the same factor as the load, see Figure 4.2.

We could also consider implementation of viscoelasticity. Some authors choose to include viscoelasticity in their model of the myocardium, see for example [21]. Holzapfel *et al.* show how the myocardium has a viscoelastic response due to hysteresis effects [14]. However, Holzapfel *et al.* also argue that although the myocardium appears to be viscoelastic, it is less important from the point of view of mechanical modeling on the time scale of the cardiac cycle. They explain this with that the time scale of the cardiac cycle is small compared to the relaxation time of the viscoelastic response. The cardiac cycle is the sequence of mechanical and electrical events that repeats with every heart beat, and it has the time scale of about one second. If the shear wave speed is approximately 5 m/s and the length of septum is approximately 5 cm, then the time scale of the wave propagation down

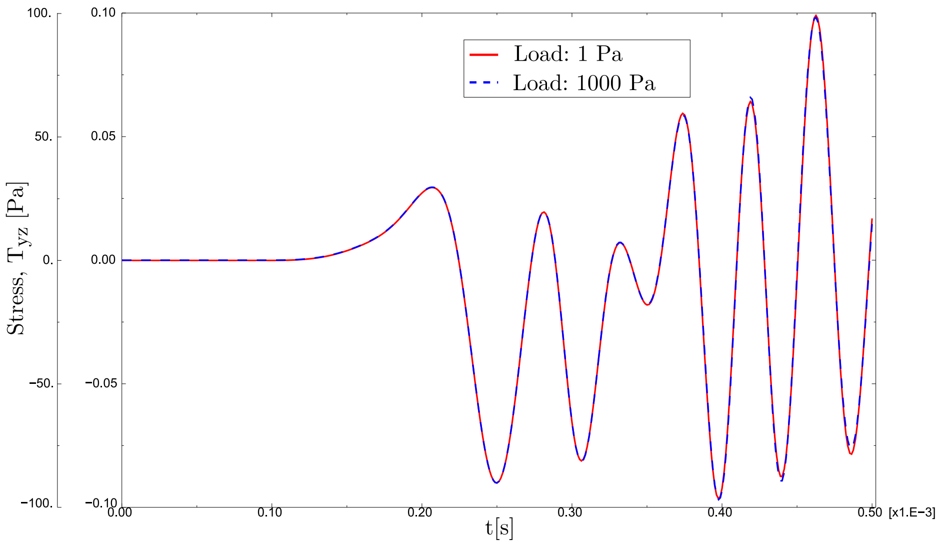


Figure 4.2: *This is the stress over time at Node 1b in the Truncated Ellipsoid Model. Due to linear theory, the shapes of the curves are identical for the two load cases. Note that there are two ordinate axes, and the leftmost axis is the rightmost axis scaled by the factor 1000. Obviously, the load case of 1000 Pa refers to the leftmost axis, and vice versa.*

septum is about:

$$\text{Time scale of wave propagation} = \frac{5\text{ cm}}{5\text{ m/s}} = 0.01\text{ s.}$$

Thus, according to Holzapfel *et al.*, we can neglect viscoelastic effects for the problem at hand. Also, it is not uncommon in literature to assume that dispersion induced by viscoelasticity is negligible, see for example [4]. This allows us to use the expressions for wave speeds in *elastic* media, *e.g.* Equation (2.15) and (2.16). Since we are most interested in modeling the wave speeds accurately, and not the amplitude, it seems fair to neglect viscoelasticity.

We have assumed that the stiffness is constant during the wave propagation. In [10], they show that stiffness, i.e. Young's modulus, varies significantly within a single cardiac cycle. More specifically, they show that it varies between approximately 15 and 105 kPa. At end of systole, when there is aortic-valve closure, they find that the stiffness is approximately 100 kPa. Based on the results from [10], and that the time scale of wave propagation is 0.01 seconds, we observe, however, that the stiffness hardly decreases more than 5 kPa in the time domain of interest. Thus, the assumption of constant stiffness is justified.

4.2 Determination of stiffness of heart wall

We saw in Section 3.5 that we can estimate a Young's Modulus E from knowledge of the wave propagation speed through the septum. In other words, if we know the shear wave speed through ultrasonic measurements we can make an estimate of the stiffness. However, to make this estimate, we have to assume isotropy and knowledge of the mass density ρ and Poisson's ratio ν. Let us first discuss the assumption of isotropy.

4.2.1 Assumption of isotropy

If there is almost no variation of the phase velocity through the thickness of the septum, then the assumption of isotropy is fair. Let us therefore discuss if it is acceptable to assume isotropy with respect to the results we obtained from the Truncated Ellipsoid Model. We saw that the shear wave speed varied when the transversely isotropic material was implemented into the model. More specifically, we found that c_{SV} varied from 105.8 to 117.9 m/s, and the average value was 112.2 m/s. Thus, a measure of the variation can be calculated:

$$\frac{117.9 - 112.2}{112.2} \cdot 100\% \approx 5\% \qquad \text{and} \qquad \frac{105.8 - 112.2}{112.2} \cdot 100\% \approx -6\%.$$

The assumption of isotropy can be acceptable based on the result above, but it would depend on the required accuracy. In relation to this discussion, it is relevant to question the wave speed obtained by the Brekke *et al.* [6]. Their study does not discuss if some variation of the shear wave speed through the thickness was observed. With their ultrasonic measurement technique they measured the shear wave speed along septum in ten healthy subjects. The result was a propagation velocity of 5.41 m/s with a standard deviation of 1.28 m/s (24%). It might be that some of the uncertainty in their measurements, *i.e.* standard deviation, comes from the variation of the wave speed through the thickness. Regardless, since the standard deviation of 24% is much larger than the wave speed variation of approximately ±5%, it is possible that today's ultrasound technology is not able to capture this variation. Thus, with the result from Brekke *et al.*, it would be natural to assume isotropy, and that gives the following Young's modulus E:

$$E = 2\rho(1+\nu)c_t^2 = 2 \cdot 1060(1+0.49) \cdot 5.41 \approx 92.5 \quad \text{[kPa]},$$

where we have assumed $\rho = 1060$ kg/m^3 and $\nu = 0.49$. This Young's modulus E agrees with what was found in [10]; they measured it to be 100 kPa. Kanai, on the other hand, found the E to be in the range 25-30 kPa [17]. More research on this topic is therefore in order.

4.2.2 Assumption of known density and Poisson's ratio

It is possible that when a heart suffers from fibrosis, there might be a change in both ρ and ν, not just E. These two parameters also affect the shear wave speed through

$$c_t = \sqrt{\frac{E}{2\rho(1+\nu)}}.$$

We cannot determine all *three* material parameters for an isotropic model purely through ultrasonic measurements, because we have only one wave speed available. Even if researchers are able to find the longitudinal wave speed in addition to the shear wave speed, then there is still one undetermined material parameter left. Nevertheless, it is not uncommon in literature to assume that biological tissues are practically incompressible, *i.e.* $\nu \approx 0.5$, see for example [4]. This gives

$$E \approx 3\rho c_t^2.$$

Also, since biological tissues in general consist mainly of water, it should be safe to assume that formation of scar tissue in myocardium leads to a negligible change in the density. Thus, with these assumptions, E can be found directly from ultrasound measurements. A more thorough study of this should be performed in future work.

4.3 Wave speed measurement procedures

With the wave speed measurement procedures presented in this thesis, we have experienced intraobserver errors. For example, the choice of where we found t_1 and t_2 in Figure 3.11 affected the wave speed noticeably. The relatively small length scales of the models can explain the sensitivity towards the time used to calculate the wave speeds.

For the Finite-size Wave Model we measured the wave speeds graphically. This was an elaborate procedure and there were many potential sources of intraobserver and interobserver errors. For example, the skew angles we found depended on what time instance we used for the measurements. Also, in Figure 3.16 we saw that the 'center' of the wave was used to measure the skew angle. Firstly, there is obviously a need for the user to interpret where the 'center' of the wave is. This can lead to interobserver errors if someone was to repeat this study. Secondly. we can question if this is the correct way to measure the skew angle. Figure 4.3 displays the difficulties discussed in this paragraph. In this figure, we see that there can be a few degrees discrepancy in the measurement of the skew angle, depending on where we do the measurement. The general tendency was that the obtained skew

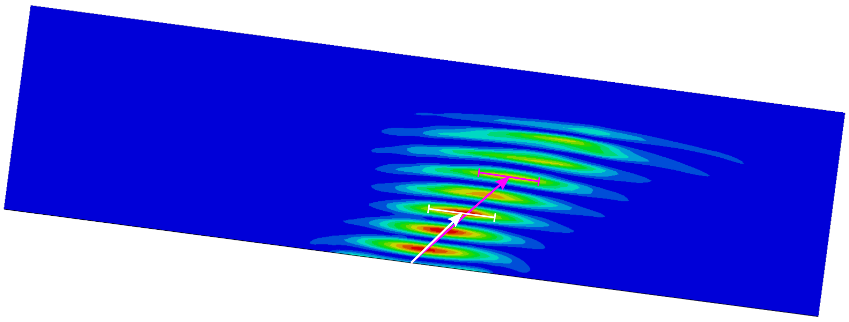


Figure 4.3: *Field plot of the strain energy density when a harmonic load is applied to the Finite-sized Wave Model with orientation $\theta = 7.5°$. The harmonic load has the same frequency as the impulse load. The two drawn-in arrows indicate where we interpret the center of the wave (phase plane) to be. As we can see, this leads to two slightly different measurements of the skew angle ψ. The pink arrow gives $\psi = -41.2°$, while the white arrow gives $\psi = -38.1°$. The analytical value is $\psi = -41.6°$.*

angle was closer to the theoretical value when the measurement was performed in a later time instance.

Nevertheless, for the models we could compare directly with theory, the errors were less than 6%. Since this error is much lower than the standard deviation of 24% obtained by Brekke *et al.* [6], we can be satisfied with the wave speed measurement procedures for these models.

For the Curved Model (with the transversely isotropic material) and Truncated Ellipsoid Model we saw that plane wave theory was not able to accurately describe the wave propagation due to the complexity of the models. It was therefore difficult to determine the validity of the results. However, as already discussed, the obtained wave speeds seemed reasonable, because they agreed to some extent with the theory. Also, with isotropic material implemented, the two models gave results that agreed excellently with the theory. If time would have allowed it, a study of waves in layered media and surface waves would have been performed, both theoretically and numerically. With knowledge of these wave types, we might have been able to better understand the wave propagation in the two mentioned models.

4.4 Truncated Ellipsoid Model as a model of left ventricle

As already mentioned in Section 3.5, a truncated ellipsoid has been commonly used in literature as a model of the left ventricle. It is obvious from Figure 2.2, which is a realistic illustration of the heart, that the Truncated Ellipsoid Model is a rather crude simplification of the left ventricle. Here we will discuss two aspects of this simplification.

4.4.1 Constant heart wall thickness

We have assumed that the left ventricle has constant thickness. Van den Broek *et al.* found that that the thickness was 1.49 cm at the equator and 0.48 cm at the apex during end-systole [26]. In other words, there is a significant variation of the thickness. The question is: does this variation affect the results enough to be considered? This question is left for future studies. In [8], they show that thickness variation can lead to wave mode conversions and reflections.

4.4.2 Boundary conditions

We have not focused on the boundary conditions of the Truncated Ellipsoid Model. All the surfaces are free, except for one point at the bottom of the apex where the model is fixed. This boundary condition was applied only to avoid potential rigid body motions. For the Plane Wave Model and the Curved Model, we experienced that the choice boundary conditions could affect the wave propagation. It is therefore probable that a change in the boundary conditions for the Truncated Ellipsoid Model can affect the obtained wave speed. The left ventricle is connected to, for example, the right ventricle, the atria and the aorta. These connections would restrain some of the motion from the wave propagation. In other words, these boundary conditions can add stiffness to the system and thereby increase the wave speed. Thus, it would be appropriate to consider this in future studies.

5 | Concluding remarks

In general, our results of wave propagation in transversely isotropic media agree well with plane wave theory. We base this conclusion on the results from the two-dimensional models, because we were able to validate these results with the theory. Plane wave theory proved to be an efficient tool for predicting even finite-sized wave propagation. With isotropic material, the three-dimensional models gave wave speeds that agreed excellently with plane wave theory as well. This experience gave confidence in the results we obtained when the transversely isotropic material was implemented, even though we were not able to properly validate these results with plane wave theory. To understand fully the wave propagation of the three-dimensional models with transversely isotropic materials, more advanced theory is probably needed.

Based on the results from the Truncated Ellipsoid Model, we have reason to believe that the muscle fiber orientation in the myocardium causes a small variation of the shear wave speed through the thickness. More specifically, we found that the wave speed varied 5 to 6 percent from an average value. Due to relatively high measurement inaccuracy it is likely that current ultrasound technology is not able to capture this variation. Based on this, we argue that it is reasonable to assume isotropy in the heart wall. This assumption allows us to calculate the stiffness of the heart wall if we assume that we know the mass density and Poisson's ratio. Thus, if researchers are able to obtain data on the mass density and Poisson's ratio in addition to the shear wave speed along septum, they can make an estimate of the stiffness in terms of the Young's modulus. The results in this thesis show that we can investigate details of wave propagation in the heart wall with the help of numerical simulations.

A | Maple scripts

On the following pages one can find two Maple scripts. The first script finds the phase velocity, energy velocity and skew angle for a wave mode chosen by the user. The script uses numerical values through all the lines. In other words, there are no analytical expressions. In the second script we find analytical expressions for the phase and energy velocities, and then plots of the characteristic surfaces are made.

Note that not all the plots made by Maple in this thesis can be found in the appended scripts. However, it should be easy to modify the scripts such that they give the desired plots.

For readers who are not familiar with Maple: remove the colon after a line if you want to print the result.

Numerical Values

*This Maple script is written such that it calculates the phase velocity, energy velocity and skew angle in a given wave propagation direction **n** (given by θ).*
Written by Erik Grimsmo.

Use 'restart' to delete all previously saved variables

```
> restart;
```

Include necessary packages

```
> with(LinearAlgebra) : with(plots) : with(VectorCalculus) :
```

The number of decimals to be displayed:

```
> interface(displayprecision = 2) :
```

Insert the theta (in degrees) that describes the propagation direction:

> $\theta := 20 : \theta := \frac{\theta \cdot \mathrm{Pi}}{180} : n := \langle convert(\sin(\theta), float), 0, convert(\cos(\theta), float)\rangle :$

Define the stiffness matrix for transverse isotropic material:

> $C := Matrix(6, 6, shape = symmetric) :$
> $C[1, 1] := 2.46E9 :$
> $C[2, 2] := 2.46E9 :$
> $C[3, 3] := 2.53E9 :$
> $C[4, 4] := 8.97E6 :$
> $C[5, 5] := 8.97E6 :$
> $C[6, 6] := \frac{1}{2}(C[1, 1] - C[1, 2]) :$
> $C[1, 2] := 2.4431E9 :$
> $C[1, 3] := 2.44E9 :$
> $C[2, 3] := 2.44E9 :$
> $evalm(C) :$

Define the mass density:

> $\rho := 1060 :$

Define the symmetrical acoustic tensor:

> $\mathrm{Gamma} := Matrix(3, 3, shape = symmetric) :$
> $\mathrm{Gamma}[1, 1] := C[1, 1] \cdot n[1]^2 + C[6, 6] \cdot n[2]^2 + C[5, 5] \cdot n[3]^2 :$
> $\mathrm{Gamma}[2, 2] := C[6, 6] \cdot n[1]^2 + C[2, 2] \cdot n[2]^2 + C[4, 4] \cdot n[3]^2 :$
> $\mathrm{Gamma}[3, 3] := C[5, 5] \cdot n[1]^2 + C[4, 4] \cdot n[2]^2 + C[3, 3] \cdot n[3]^2 :$
> $\mathrm{Gamma}[1, 2] := (C[1, 2] + C[6, 6]) \cdot n[1] \cdot n[2] :$
> $\mathrm{Gamma}[1, 3] := (C[1, 3] + C[5, 5]) \cdot n[1] \cdot n[3] :$
> $\mathrm{Gamma}[2, 3] := (C[2, 3] + C[4, 4]) \cdot n[2] \cdot n[3] :$
> $evalm(\mathrm{Gamma}) :$

Now, we can find the eigenvalues and eigenvectors of Gamma:

> $(\lambda, A) := Eigenvectors(convert(\mathrm{Gamma}, rational)) :$

Display the eigenvectors one by one:

> $Normalize(A[\,..,1], Euclidean) :$
> $Normalize(A[\,..,2], Euclidean) :$
> $Normalize(A[\,..,3], Euclidean) :$

Display the eigenvalues one by one:

> $simplify(\lambda[1]) :$
> $simplify(\lambda[2]) :$
> $simplify(\lambda[3]) :$

Finding the phase velocities (2 transverse, 1 longitudinal):

> $PhaseVelocities := \mathrm{sqrt}{\sim}\left(\frac{\lambda}{{\sim}\rho}\right) :$

Find energy velocity. Choose for which mode you want to find the energy velocity (i=1,2,3):

```
> i := 3 :
> EnergyVelocity1 := C[1, 1]·A[1, i]·A[1, i]·n[1] + C[6, 1]·A[2, i]·A[1, i]·n[1] + C[5, 1]·A[3, i]·A[1, i]
        ·n[1]
    + C[1, 6]·A[1, i]·A[2, i]·n[1] + C[6, 6]·A[2, i]·A[2, i]·n[1] + C[5, 6]·A[3, i]·A[2, i]·n[1]
    + C[1, 5]·A[1, i]·A[3, i]·n[1] + C[6, 5]·A[2, i]·A[3, i]·n[1] + C[5, 5]·A[3, i]·A[3, i]·n[1]
    + C[1, 6]·A[1, i]·A[1, i]·n[2] + C[6, 6]·A[2, i]·A[1, i]·n[2] + C[5, 6]·A[3, i]·A[1, i]·n[2]
    + C[1, 2]·A[1, i]·A[2, i]·n[2] + C[6, 2]·A[2, i]·A[2, i]·n[2] + C[5, 2]·A[3, i]·A[2, i]·n[2]
    + C[1, 4]·A[1, i]·A[3, i]·n[2] + C[6, 4]·A[2, i]·A[3, i]·n[2] + C[5, 4]·A[3, i]·A[3, i]·n[2]
    + C[1, 5]·A[1, i]·A[1, i]·n[3] + C[6, 5]·A[2, i]·A[1, i]·n[3] + C[5, 5]·A[3, i]·A[1, i]·n[3]
    + C[1, 4]·A[1, i]·A[2, i]·n[3] + C[6, 4]·A[2, i]·A[2, i]·n[3] + C[5, 4]·A[3, i]·A[2, i]·n[3]
    + C[1, 3]·A[1, i]·A[3, i]·n[3] + C[6, 3]·A[2, i]·A[3, i]·n[3] + C[5, 3]·A[3, i]·A[3, i]·n[3] :
> EnergyVelocity2 := C[6, 1]·A[1, i]·A[1, i]·n[1] + C[2, 1]·A[2, i]·A[1, i]·n[1] + C[4, 1]·A[3, i]·A[1, i]
        ·n[1]
    + C[6, 6]·A[1, i]·A[2, i]·n[1] + C[2, 6]·A[2, i]·A[2, i]·n[1] + C[4, 6]·A[3, i]·A[2, i]·n[1]
    + C[6, 5]·A[1, i]·A[3, i]·n[1] + C[2, 5]·A[2, i]·A[3, i]·n[1] + C[4, 5]·A[3, i]·A[3, i]·n[1]
    + C[6, 6]·A[1, i]·A[1, i]·n[2] + C[2, 6]·A[2, i]·A[1, i]·n[2] + C[4, 6]·A[3, i]·A[1, i]·n[2]
    + C[6, 2]·A[1, i]·A[2, i]·n[2] + C[2, 2]·A[2, i]·A[2, i]·n[2] + C[4, 2]·A[3, i]·A[2, i]·n[2]
    + C[6, 4]·A[1, i]·A[3, i]·n[2] + C[2, 4]·A[2, i]·A[3, i]·n[2] + C[4, 4]·A[3, i]·A[3, i]·n[2]
    + C[6, 5]·A[1, i]·A[1, i]·n[3] + C[2, 5]·A[2, i]·A[1, i]·n[3] + C[4, 5]·A[3, i]·A[1, i]·n[3]
    + C[6, 4]·A[1, i]·A[2, i]·n[3] + C[2, 4]·A[2, i]·A[2, i]·n[3] + C[4, 4]·A[3, i]·A[2, i]·n[3]
    + C[6, 3]·A[1, i]·A[3, i]·n[3] + C[2, 3]·A[2, i]·A[3, i]·n[3] + C[4, 3]·A[3, i]·A[3, i]·n[3]:
> EnergyVelocity3 := C[5, 1]·A[1, i]·A[1, i]·n[1] + C[4, 1]·A[2, i]·A[1, i]·n[1] + C[3, 1]·A[3, i]·A[1, i]
        ·n[1]
    + C[5, 6]·A[1, i]·A[2, i]·n[1] + C[4, 6]·A[2, i]·A[2, i]·n[1] + C[3, 6]·A[3, i]·A[2, i]·n[1]
    + C[5, 5]·A[1, i]·A[3, i]·n[1] + C[4, 5]·A[2, i]·A[3, i]·n[1] + C[3, 5]·A[3, i]·A[3, i]·n[1]
    + C[5, 6]·A[1, i]·A[1, i]·n[2] + C[4, 6]·A[2, i]·A[1, i]·n[2] + C[3, 6]·A[3, i]·A[1, i]·n[2]
    + C[5, 2]·A[1, i]·A[2, i]·n[2] + C[4, 2]·A[2, i]·A[2, i]·n[2] + C[3, 2]·A[3, i]·A[2, i]·n[2]
    + C[5, 4]·A[1, i]·A[3, i]·n[2] + C[4, 4]·A[2, i]·A[3, i]·n[2] + C[3, 4]·A[3, i]·A[3, i]·n[2]
    + C[5, 5]·A[1, i]·A[1, i]·n[3] + C[4, 5]·A[2, i]·A[1, i]·n[3] + C[3, 5]·A[3, i]·A[1, i]·n[3]
    + C[5, 4]·A[1, i]·A[2, i]·n[3] + C[4, 4]·A[2, i]·A[2, i]·n[3] + C[3, 4]·A[3, i]·A[2, i]·n[3]
    + C[5, 3]·A[1, i]·A[3, i]·n[3] + C[4, 3]·A[2, i]·A[3, i]·n[3] + C[3, 3]·A[3, i]·A[3, i]·n[3] :
```

$$> EnergyVelocity := \frac{1}{\rho \cdot PhaseVelocities[i] \cdot Norm(A[\,..,i])^2} \cdot \langle EnergyVelocity1, EnergyVelocity2, EnergyVelocity3 \rangle :$$

Below we calculate the phase velocity, energy velocity and skew angle for the wave mode given by i.

```
> evalf(PhaseVelocities[i]);
                                  124.6125030
> Norm(EnergyVelocity);
                                  184.5066087
```

$$> SkewAngle := -convert\left(convert\left(\arctan\left(EnergyVelocity[1], EnergyVelocity[3]\right), degrees\right) - \frac{\theta \cdot 180}{Pi} degrees, float\right);$$

$$SkewAngle := -47.52\ degrees$$

Analytical Expressions and Charactersitc Plots

This Maple script find the analytical expressions for the material displacements, phase velocities and energy velocities. Thereby the stiffness coefficients of ***C*** *are inserted, and the characterstic surface are plotted. Written by Erik Grimsmo.*

Use 'restart' to delete all previously saved variables

```
> restart;
```

```
> with(LinearAlgebra) : with(plots) : with(VectorCalculus) :
```

Use the following package if you want to work with degrees rather than radians. However, note that it does not work with polar plots.

```
> #my_trig := module()
   #option package;
   #export sin, cos;
     #sin := x → :-sin(Pi*x/180);
     #cos := x → :-cos(Pi*x/180);
   #end module:


   #with(my_trig) :
```

The number of decimals to be displayed:

```
> interface(displayprecision = 2) :
```

Insert the direction cosine of the plane wave (3-direction is the direction of the fibers):

> $n := \langle \cos(\theta), 0, \sin(\theta) \rangle$:

Define the stiffness matrix for transverse isotropic material:

> $C := Matrix(6, 6, shape = symmetric)$:
> $C[1, 1] := C_{11}$:
> $C[2, 2] := C_{11}$:
> $C[3, 3] := C_{33}$:
> $C[4, 4] := C_{44}$:
> $C[5, 5] := C_{44}$:
> $C[6, 6] := \frac{1}{2} \cdot (C_{11} - C_{12})$:
> $C[1, 2] := C_{12}$:
> $C[1, 3] := C_{13}$:
> $C[2, 3] := C_{13}$:
> $evalm(C)$:

Define the symmetrical acoustic tensor:

> $\mathrm{Gamma} := Matrix(3, 3, shape = symmetric)$:
> $\mathrm{Gamma}[1, 1] := C[1, 1] \cdot n[1]^2 + C[6, 6] \cdot n[2]^2 + C[5, 5] \cdot n[3]^2$:
> $\mathrm{Gamma}[2, 2] := C[6, 6] \cdot n[1]^2 + C[2, 2] \cdot n[2]^2 + C[4, 4] \cdot n[3]^2$:
> $\mathrm{Gamma}[3, 3] := C[5, 5] \cdot n[1]^2 + C[4, 4] \cdot n[2]^2 + C[3, 3] \cdot n[3]^2$:
> $\mathrm{Gamma}[1, 2] := (C[1, 2] + C[6, 6]) \cdot n[1] \cdot n[2]$:
> $\mathrm{Gamma}[1, 3] := (C[1, 3] + C[5, 5]) \cdot n[1] \cdot n[3]$:
> $\mathrm{Gamma}[2, 3] := (C[2, 3] + C[4, 4]) \cdot n[2] \cdot n[3]$:
> $evalm(\mathrm{Gamma})$:

Now, we can find the eigenvalues and eigenvectors of Gamma:

> $(\lambda, A) := Eigenvectors(convert(\mathrm{Gamma}, rational))$:

Display the eigenvectors one by one. The eigenvectors represent the material displacements (polarizations).

```
> Normalize(A[ .., 1], Euclidean) :
> Normalize(A[ .., 2], Euclidean) :
> Normalize(A[ .., 3], Euclidean) :
```

Display the eigenvalues one by one:

```
> simplify(λ[1]) :
> simplify(λ[2]) :
> simplify(λ[3]) :
```

Finding the phase velocities:

```
> PhaseVelocities := sqrt~(λ/~ρ) :
```

Find energy velocity:

```
> EnergyVelocities := Matrix(3, 3) :
> for i from 1 by 1 to 3 do  EnergyVelocities[1, i] := C[1, 1]·A[1, i]·A[1, i]·n[1] + C[6, 1]·A[2, i]·A[1, i]
        ·n[1] + C[5, 1]·A[3, i]·A[1, i]·n[1]
    + C[1, 6]·A[1, i]·A[2, i]·n[1] + C[6, 6]·A[2, i]·A[2, i]·n[1] + C[5, 6]·A[3, i]·A[2, i]·n[1]
    + C[1, 5]·A[1, i]·A[3, i]·n[1] + C[6, 5]·A[2, i]·A[3, i]·n[1] + C[5, 5]·A[3, i]·A[3, i]·n[1]
    + C[1, 6]·A[1, i]·A[1, i]·n[2] + C[6, 6]·A[2, i]·A[1, i]·n[2] + C[5, 6]·A[3, i]·A[1, i]·n[2]
    + C[1, 2]·A[1, i]·A[2, i]·n[2] + C[6, 2]·A[2, i]·A[2, i]·n[2] + C[5, 2]·A[3, i]·A[2, i]·n[2]
    + C[1, 4]·A[1, i]·A[3, i]·n[2] + C[6, 4]·A[2, i]·A[3, i]·n[2] + C[5, 4]·A[3, i]·A[3, i]·n[2]
    + C[1, 5]·A[1, i]·A[1, i]·n[3] + C[6, 5]·A[2, i]·A[1, i]·n[3] + C[5, 5]·A[3, i]·A[1, i]·n[3]
    + C[1, 4]·A[1, i]·A[2, i]·n[3] + C[6, 4]·A[2, i]·A[2, i]·n[3] + C[5, 4]·A[3, i]·A[2, i]·n[3]
    + C[1, 3]·A[1, i]·A[3, i]·n[3] + C[6, 3]·A[2, i]·A[3, i]·n[3] + C[5, 3]·A[3, i]·A[3, i]·n[3] end do:
> for i from 1 by 1 to 3 do EnergyVelocities[2, i] := C[6, 1]·A[1, i]·A[1, i]·n[1] + C[2, 1]·A[2, i]·A[1, i]
        ·n[1] + C[4, 1]·A[3, i]·A[1, i]·n[1]
    + C[6, 6]·A[1, i]·A[2, i]·n[1] + C[2, 6]·A[2, i]·A[2, i]·n[1] + C[4, 6]·A[3, i]·A[2, i]·n[1]
    + C[6, 5]·A[1, i]·A[3, i]·n[1] + C[2, 5]·A[2, i]·A[3, i]·n[1] + C[4, 5]·A[3, i]·A[3, i]·n[1]
    + C[6, 6]·A[1, i]·A[1, i]·n[2] + C[2, 6]·A[2, i]·A[1, i]·n[2] + C[4, 6]·A[3, i]·A[1, i]·n[2]
    + C[6, 2]·A[1, i]·A[2, i]·n[2] + C[2, 2]·A[2, i]·A[2, i]·n[2] + C[4, 2]·A[3, i]·A[2, i]·n[2]
    + C[6, 4]·A[1, i]·A[3, i]·n[2] + C[2, 4]·A[2, i]·A[3, i]·n[2] + C[4, 4]·A[3, i]·A[3, i]·n[2]
    + C[6, 5]·A[1, i]·A[1, i]·n[3] + C[2, 5]·A[2, i]·A[1, i]·n[3] + C[4, 5]·A[3, i]·A[1, i]·n[3]
    + C[6, 4]·A[1, i]·A[2, i]·n[3] + C[2, 4]·A[2, i]·A[2, i]·n[3] + C[4, 4]·A[3, i]·A[2, i]·n[3]
    + C[6, 3]·A[1, i]·A[3, i]·n[3] + C[2, 3]·A[2, i]·A[3, i]·n[3] + C[4, 3]·A[3, i]·A[3, i]·n[3] end do:
> for i from 1 by 1 to 3 do EnergyVelocities[3, i] := C[5, 1]·A[1, i]·A[1, i]·n[1] + C[4, 1]·A[2, i]·A[1, i]
        ·n[1] + C[3, 1]·A[3, i]·A[1, i]·n[1]
    + C[5, 6]·A[1, i]·A[2, i]·n[1] + C[4, 6]·A[2, i]·A[2, i]·n[1] + C[3, 6]·A[3, i]·A[2, i]·n[1]
    + C[5, 5]·A[1, i]·A[3, i]·n[1] + C[4, 5]·A[2, i]·A[3, i]·n[1] + C[3, 5]·A[3, i]·A[3, i]·n[1]
    + C[5, 6]·A[1, i]·A[1, i]·n[2] + C[4, 6]·A[2, i]·A[1, i]·n[2] + C[3, 6]·A[3, i]·A[1, i]·n[2]
    + C[5, 2]·A[1, i]·A[2, i]·n[2] + C[4, 2]·A[2, i]·A[2, i]·n[2] + C[3, 2]·A[3, i]·A[2, i]·n[2]
    + C[5, 4]·A[1, i]·A[3, i]·n[2] + C[4, 4]·A[2, i]·A[3, i]·n[2] + C[3, 4]·A[3, i]·A[3, i]·n[2]
    + C[5, 5]·A[1, i]·A[1, i]·n[3] + C[4, 5]·A[2, i]·A[1, i]·n[3] + C[3, 5]·A[3, i]·A[1, i]·n[3]
    + C[5, 4]·A[1, i]·A[2, i]·n[3] + C[4, 4]·A[2, i]·A[2, i]·n[3] + C[3, 4]·A[3, i]·A[2, i]·n[3]
    + C[5, 3]·A[1, i]·A[3, i]·n[3] + C[4, 3]·A[2, i]·A[3, i]·n[3] + C[3, 3]·A[3, i]·A[3, i]·n[3]  end do:
> for i from 1 by 1 to 3 do EnergyVelocities[ .., i] := 1/(ρ·PhaseVelocities[i]·Norm(A[ .., i])^2)
        ·EnergyVelocities[ .., i] end do:
```

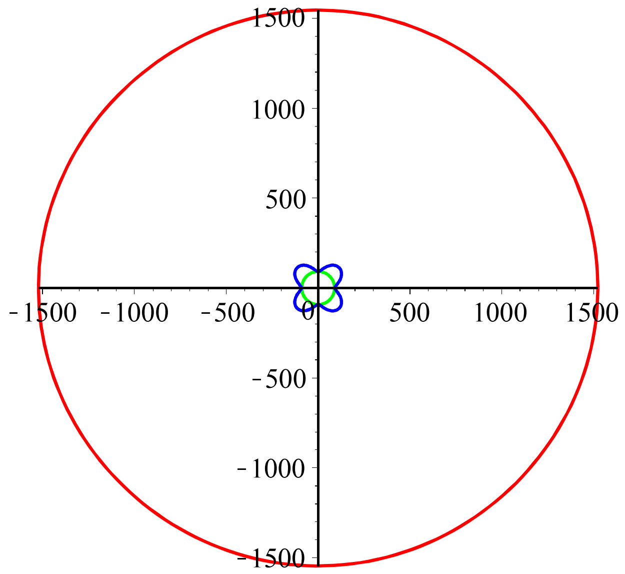
We now have analyitcal expressions for the phase velocities , material displacement and energy velocities.

Now we insert values for **C** and ρ such that characteristic surfaces can be found.

```
> C[11] := 2.46E9:
> C[33] := 2.53E9:
> C[44] := 8.97E6:
> C[12] := 2.4431E9 :
> C[13] := 2.44E9:
> ρ := 1060 :
> eval(PhaseVelocities) :
```

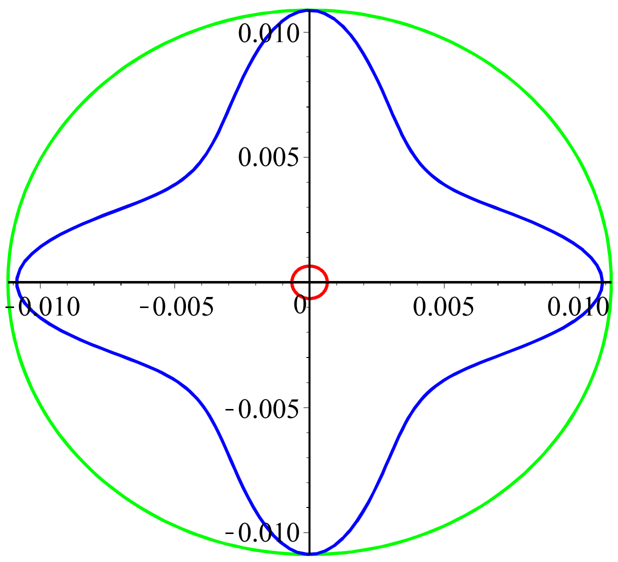
Now the characteristic surfaces can be plotted. We first plot the velocity surfaces:

```
> plota := plot(PhaseVelocities[1], θ = 0 .. 2· Pi, coords = polar, color = red) :
> plotb := plot(PhaseVelocities[2], θ = 0 .. 2 ·Pi, coords = polar, color = blue) :
> plotc := plot(PhaseVelocities[3], θ = 0 .. 2 ·Pi, coords = polar, color = green) :
> display({plota, plotb, plotc}, );
```



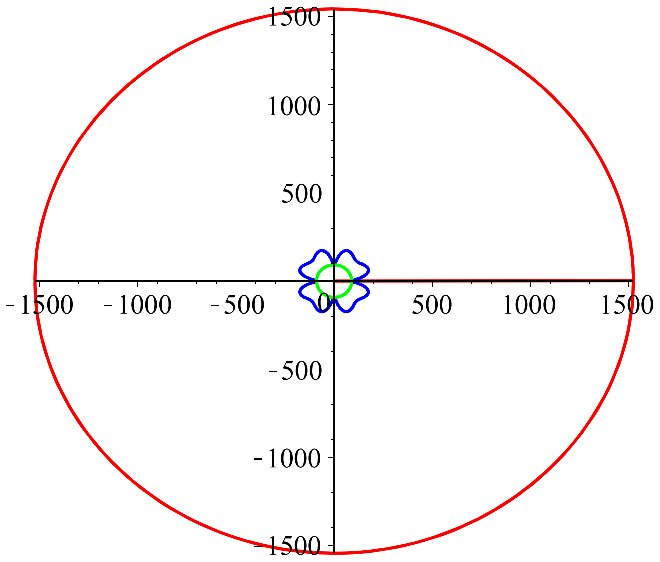
Plots of the slowness surfaces.

```
> plota := plot(1/PhaseVelocities[1], θ = 0 .. 2·Pi, coords = polar, color = red) :
> plotb := plot(1/PhaseVelocities[2], θ = 0 .. 2·Pi, coords = polar, color = blue) :
> plotc := plot(1/PhaseVelocities[3], θ = 0 .. 2·Pi, coords = polar, color = green) :
> display({plota, plotb, plotc});
```



Plots of the energy surfaces

```
> plota := plot(Norm(EnergyVelocities[ .., 1]), θ = 0 .. 2· Pi, coords = polar, color = red) :
> plotb := plot(Norm(EnergyVelocities[ .., 2]), θ = 0 ..  2 ·Pi, coords = polar, color = blue) :
> plotc := plot(Norm(EnergyVelocities[ .., 3]), θ = 0 ..  2 ·Pi, coords = polar, color = green) :
> display({plota, plotb, plotc});
```


1500
1000
500
-1500
-1000
-500
0
500
1000
1500
-500
-1000
-1500

B | Python scripts

In this chapter, one can find the Python scripts that generate the four models discussed in this thesis. We have also appended the model, which was briefly discussed in Figure 2.18. One procedure to run the script, is to open Abaqus CAE (Abaqus' graphical user interface), click on 'File' and then 'Run script...'.

The scripts are written such that there are some variables in the beginning of the script that the user can edit. These variables are typically the dimensions of the geometry, mesh density, load amplitude etc. In this way, the user does not have to read and understand all the code to be able to work with the scripts.

Comment: *we have experienced that Abaqus' has a tendency to crash when we run the scripts due to the applied traction load. We have not found any report on this bug on the World Wide Web. Nevertheless, a way to avoid this problem is to comment out the lines in the code where the traction load is applied. Next: run the script (now, no load is applied to the model), and then uncomment the lines previously commented out. Finally, run the script again, and everything should then be in order.*

Plane Wave Model

```
#This is a python script that creates an FEM model referred to as the Plane Wave Model
#in the master thesis. Under the section "Define variables", the user can specify for example
#the geometry or mesh density.

#Written by Erik Grimsmo

#------------------------------------------------------Initialize---------------------------------------------------------------

from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *

session.journalOptions.setValues(replayGeometry=COORDINATE,recoverGeometry=COORDINATE)

#Change work directory (use your personal directory, and uncomment the line below)
#os.chdir(r'D:\abaqusworkdirectory')

#----------------------------------------------Define variables--------------------------------------------------------------

a = 0.2 #Height of plate
b = 0.2 #Length of plate

theta = 45 #Orientation of plate. 0 degrees = n aligned with fibre direction

rho = 1060.0 #Density

F = 1.0 #The amplitude of the applied traction load

t = 0.0015 #Period of time integration
c_t = 100.0 #Approx. transverse wave speed, used to find appropriate element size
loadTime = 1E-4 #Duration of blast load
N = 10 #Number of elements over which the load will take place
elSize = loadTime*c_t/n #Element size

#Do not alter the script after this section,
#unless you are familiar with python scripting in Abaqus

#----------------------------------------------Create model--------------------------------------------------------------
```

Plane Wave Model

```
theta = 90-theta #The model was initially made with  theta
#equal to zero along x1-direction. This is fixed with this line.

mdb.Model(name='PWM')
mdb.models['PWM'].setValues(noPartsInputFile=ON)

#--------------------------------------------Create sketch-------------------------------------------------------------

mdb.models['PWM'].ConstrainedSketch(name='plateSketch', sheetSize=200.0)
mdb.models['PWM'].sketches['plateSketch'].rectangle(point1=(0.0, 0.0),
   point2=(a, b))
mdb.models['PWM'].Part(dimensionality=TWO_D_PLANAR, name='plate',
   type=DEFORMABLE_BODY)
mdb.models['PWM'].parts['plate'].BaseShell(sketch=
   mdb.models['PWM'].sketches['plateSketch'])

#--------------------------------------------Define material---------------------------------------------------------

#Isotropic material
mdb.models['PWM'].Material(name='isotropic')
mdb.models['PWM'].materials['isotropic'].Density(table=((1060.0, ), ))
mdb.models['PWM'].materials['isotropic'].Elastic(table=((39765628.0, 0.49), ))

#Transverse isotropic material
mdb.models['PWM'].Material(name='anisotropic')
mdb.models['PWM'].materials['anisotropic'].Elastic(table=((2460000000.0,
   2440000000.0, 2530000000.0, 2443100000.0, 2440000000.0, 2460000000.0, 0.0,
   0.0, 0.0, 8970000.0, 0.0, 0.0, 0.0, 0.0, 8450000.0, 0.0, 0.0, 0.0, 0.0,
   0.0, 8970000), ), type=ANISOTROPIC)
mdb.models['PWM'].materials['anisotropic'].Density(table=((rho, ), ))
mdb.models['PWM'].parts['plate'].MaterialOrientation(
   additionalRotationField='', additionalRotationType=ROTATION_ANGLE, angle=
   0, axis=AXIS_3, fieldName='', localCsys=None, orientationType=SYSTEM,
   region=Region(
   faces=mdb.models['PWM'].parts['plate'].faces.getByBoundingBox(
   -1,-1,-1,a+1,b+1,1) ,), stackDirection=STACK_3)

#--------------------------------------------Create section-------------------------------------------------------------

#Choose here what material you want to work with
mdb.models['PWM'].HomogeneousSolidSection(material='anisotropic', name=
   'Section-1', thickness=None)
mdb.models['PWM'].parts['plate'].SectionAssignment(offset=0.0,
   offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
   faces=mdb.models['PWM'].parts['plate'].faces.getSequenceFromMask(
   mask=('[#1 ]', ), )), sectionName='Section-1', thicknessAssignment=
   FROM_SECTION)
```

Plane Wave Model

```
#----------------------------------------------Partition face---------------------------------------------------------------

#Here we partition the face that allow defining a set at the  center of the model

mdb.models['PWM'].ConstrainedSketch(gridSpacing=0.001, name='__profile__',
   sheetSize=0.044, transform=
   mdb.models['PWM'].parts['plate'].MakeSketchTransform(
   sketchPlane=mdb.models['PWM'].parts['plate'].faces.findAt((
   a/2, b/2, 0.0), (0.0, 0.0, 1.0)), sketchPlaneSide=SIDE1,
   sketchOrientation=RIGHT, origin=(0.0, 0.0, 0.0)))
mdb.models['PWM'].parts['plate'].projectReferencesOntoSketch(filter=
   COPLANAR_EDGES, sketch=mdb.models['PWM'].sketches['__profile__'])
mdb.models['PWM'].sketches['__profile__'].Line(point1=(0.0, b/2), point2=(
   a, b/2))
mdb.models['PWM'].sketches['__profile__'].Line(point1=(a/2, 0.0), point2=(
   a/2, b))
mdb.models['PWM'].parts['plate'].PartitionFaceBySketch(faces=
   mdb.models['PWM'].parts['plate'].faces.findAt(((a/2,
   b/2, 0.0), )), sketch=mdb.models['PWM'].sketches['__profile__'])
del mdb.models['PWM'].sketches['__profile__']


#----------------------------------------------Create assembly---------------------------------------------------------------

mdb.models['PWM'].rootAssembly.DatumCsysByDefault(CARTESIAN)
mdb.models['PWM'].rootAssembly.Instance(dependent=ON, name='plate-1',
   part=mdb.models['PWM'].parts['plate'])

#-----------------------------------------------Create step-----------------------------------------------------------------

mdb.models['PWM'].ExplicitDynamicsStep(name='dynamic',previous=
   'Initial', timePeriod=t,  quadBulkViscosity=0.0, linearBulkViscosity=0.06)

#----------------------------------------Apply BCs and load------------------------------------------------------------

#Fixed end
mdb.models['PWM'].DisplacementBC(amplitude=UNSET, createStepName='Initial',
   distributionType=UNIFORM, fieldName='', localCsys=None, name='fixedEnd',
   region=Region(
   edges=mdb.models['PWM'].rootAssembly.instances['plate-1'].edges.findAt(
   ((a, b/4, 0.0), ), ((a,3*b/4,0.0),), )), u1=SET, u2=SET, ur3=SET)

#Determine the load shape in time
mdb.models['PWM'].SmoothStepAmplitude(data=((0.0, 0.0), (loadTime-1E-5, 0.0), (
   loadTime, 1.0), (2*loadTime, 1.0), (2*loadTime+1E-5, 0.0)), name='impulse_smooth', timeSpan=STEP)

#Apply surface traction
```

Plane Wave Model

```
mdb.models['PWM'].SurfaceTraction(amplitude='impulse_smooth', createStepName=
    'dynamic', directionVector=(
    mdb.models['PWM'].rootAssembly.instances['plate-1'].InterestingPoint(
    mdb.models['PWM'].rootAssembly.instances['plate-1'].edges.findAt(
    (a, 0.0, 0.0), ), MIDDLE),
    mdb.models['PWM'].rootAssembly.instances['plate-1'].vertices.findAt(
    (a, b/2, 0.0), )), distributionType=UNIFORM, field='', localCsys=None,
    magnitude=F, name='traction', region=Region(
    side1Edges=mdb.models['PWM'].rootAssembly.instances['plate-1'].edges.findAt(
    ((0, b/4, 0.0), ), ((0, 3*b/4, 0.0), ), )), resultant=ON)

#--------------------------------------------Generate mesh--------------------------------------------------------

mdb.models['PWM'].parts['plate'].seedPart(deviationFactor=0.1,
    minSizeFactor=0.1, size=elSize)
mdb.models['PWM'].parts['plate'].setMeshControls(elemShape=QUAD,
    regions=mdb.models['PWM'].parts['plate'].faces.findAt(((a/4,
    b/4, 0.0),), ((3*a/4,b/4,0),), ((a/4,3*b/4,0),), ((3*a/4,3*b/4,0),) ), technique=STRUCTURED)
mdb.models['PWM'].parts['plate'].setElementType(elemTypes=(ElemType(
    elemCode=CPE4R, elemLibrary=STANDARD, secondOrderAccuracy=OFF,
    hourglassControl=DEFAULT, distortionControl=DEFAULT), ElemType(
    elemCode=CPE4R, elemLibrary=STANDARD)), regions=(
    mdb.models['PWM'].parts['plate'].faces.findAt(((a/4,
    b/4, 0.0),), ((3*a/4,b/4,0),), ((a/4,3*b/4,0),), ((3*a/4,3*b/4,0),),),))
mdb.models['PWM'].parts['plate'].generateMesh()

#---------------------------Rotate plate and apply boundary condititions--------------------------------------

mdb.models['PWM'].rootAssembly.rotate(angle=theta, axisDirection=(0.0, 0.0,
    1.0), axisPoint=(0.0, 0.0, 0.0), instanceList=('plate-1', ))

#Apply boundary conditions that can only be applied after rotation
theta = theta*pi/180

myDatum = mdb.models['PWM'].rootAssembly.DatumCsysByThreePoints(coordSysType=CARTESIAN
    , name='Datum csys-2', origin=
    mdb.models['PWM'].rootAssembly.instances['plate-1'].vertices.findAt(
    (0.0, 0.0, 0.0), ), point1=
    mdb.models['PWM'].rootAssembly.instances['plate-1'].vertices.findAt(
    (a*cos(theta), a*sin(theta), 0.0), ), point2=
    mdb.models['PWM'].rootAssembly.instances['plate-1'].vertices.findAt(
    (-b*sin(theta), b*cos(theta), 0.0), ))

mdb.models['PWM'].DisplacementBC(amplitude=UNSET, createStepName='Initial',
    distributionType=UNIFORM, fieldName='',
    localCsys=mdb.models['PWM'].rootAssembly.datums[myDatum.id], name='fixedEnd',
    region=Region(
    edges=mdb.models['PWM'].rootAssembly.instances['plate-1'].edges.findAt(
    ((a*cos(theta)-0.25*b*sin(theta), a*sin(theta)+0.25*b*cos(theta), 0.0), ),
    ((a*cos(theta)-0.75*b*sin(theta), a*sin(theta)+0.75*b*cos(theta), 0.0), ), )),
    u1=SET, u2=SET, ur3=SET)

mdb.models['PWM'].DisplacementBC(amplitude=UNSET, createStepName='Initial',
    distributionType=UNIFORM, fieldName='', localCsys=
    mdb.models['PWM'].rootAssembly.datums[myDatum.id], name='rolling support', region=Region(
    edges=mdb.models['PWM'].rootAssembly.instances['plate-1'].edges.findAt(
    ((0.25*a*cos(theta), 0.25*a*sin(theta), 0.0), ),
    ((0.75*a*cos(theta), 0.75*a*sin(theta), 0.0), ),
    ((0.25*a*cos(theta)-b*sin(theta), 0.25*a*sin(theta)+b*cos(theta), 0.0), ),
    ((0.75*a*cos(theta)-b*sin(theta),  0.75*a*sin(theta)+b*cos(theta), 0.0), ),
    )), u1=SET, u2=UNSET, ur3=UNSET)

#--------------------------------------------------Request outputs--------------------------------------------------------

#Define set at center and at load of plate
mdb.models['PWM'].parts['plate'].Set(name='midNode', vertices=
    mdb.models['PWM'].parts['plate'].vertices.findAt(((a/2, b/2,
    0.0), )))

#History output at midnode
mdb.models['PWM'].historyOutputRequests['H-Output-1'].setValues(frequency=1,
    rebar=EXCLUDE, region=
    mdb.models['PWM'].rootAssembly.instances['plate-1'].sets['midNode']
    , sectionPoints=DEFAULT, variables=('UT', 'S12', ))

#Field output
mdb.models['PWM'].fieldOutputRequests['F-Output-1'].setValues(numIntervals=
    300,  directions=OFF, variables=('S', ))

#---------------------------------------------Create job -------------------------------------------------------------------

mdb.models['PWM'].rootAssembly.regenerate()
mdb.Job(activateLoadBalancing=False, atTime=None, contactPrint=OFF,
    description='', echoPrint=OFF, explicitPrecision=SINGLE, historyPrint=OFF,
    model='PWM', modelPrint=OFF, multiprocessingMode=DEFAULT, name=
    'Plane_Wave_Model', nodalOutputPrecision=SINGLE, numCpus=1, numDomains=
    1, parallelizationMethodExplicit=DOMAIN, queue=None, scratch='', type=
    ANALYSIS, userSubroutine='', waitHours=0, waitMinutes=0)
```

Finite-sized Wave Model

```
#This is a python script that creates an FEM model referred to as the Finite-sized Wave Model
#in the master thesis. Under the section "Define variables", the user can specify for example
#the geometry or mesh density.

#Written by Erik Grimsmo

#----------------------------------------------------Initialize----------------------------------------------------------------

from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *

session.journalOptions.setValues(replayGeometry=COORDINATE,recoverGeometry=COORDINATE)

#Change work directory (use your personal directory, and uncomment the line below)
#os.chdir(r'D:\abaqusworkdirectory')

#---------------------------------------------Define variables---------------------------------------------------------------

a = 0.1#Height of plate
b = 0.4 #Length of plate
c = 0.04 #Length of half  the domain where the load is applied

theta = 45  #Orientation of plate. 0 degrees = n aligned with fibre direction

rho = 1060.0 #Density

F = 1.0 #The amplitude of the applied traction load

t = 0.001 #Period of time integration
c_t = 100 #Approx.  shear wave speed, used to find appropriate element size
loadTime = 1E-4 #Duration of blast load
N = 10 #Number of elements over which the load will take place
elSize = loadTime*c_t/n #Element size

#Do not alter the script after this section,
#unless you are familiar with python scripting in Abaqus

#---------------------------------------------Create model---------------------------------------------------------------

eps = 0.0001 #A variable that helps with modeling
theta = 90-theta #The model was initially made with  theta
#equal to zero along x1-direction. This is fixed with this line.

mdb.Model(name='FSWM')
mdb.models['FSWM'].setValues(noPartsInputFile=ON)
```

```
#---------------------------------------------Create sketch------------------------------------------------------------

mdb.models['FSWM'].ConstrainedSketch(name='plateSketch', sheetSize=200.0)
mdb.models['FSWM'].sketches['plateSketch'].rectangle(point1=(0.0, 0.0),
   point2=(a, b))
mdb.models['FSWM'].Part(dimensionality=TWO_D_PLANAR, name='plate',
   type=DEFORMABLE_BODY)
mdb.models['FSWM'].parts['plate'].BaseShell(sketch=
   mdb.models['FSWM'].sketches['plateSketch'])

#---------------------------------------------Define material--------------------------------------------------------

#Isotropic material
mdb.models['FSWM'].Material(name='isotropic')
mdb.models['FSWM'].materials['isotropic'].Density(table=((1060.0, ), ))
mdb.models['FSWM'].materials['isotropic'].Elastic(table=((39765628.0, 0.49), ))

#Transverse isotropic material
mdb.models['FSWM'].Material(name='anisotropic')
mdb.models['FSWM'].materials['anisotropic'].Elastic(table=((2460000000.0,
   2440000000.0, 2530000000.0, 2443100000.0, 2440000000.0, 2460000000.0, 0.0,
   0.0, 0.0, 8970000.0, 0.0, 0.0, 0.0, 0.0, 8450000.0, 0.0, 0.0, 0.0, 0.0,
   0.0, 8970000), ), type=ANISOTROPIC)
mdb.models['FSWM'].materials['anisotropic'].Density(table=((rho, ), ))
mdb.models['FSWM'].parts['plate'].MaterialOrientation(
   additionalRotationField='', additionalRotationType=ROTATION_ANGLE, angle=
   0, axis=AXIS_3, fieldName='', localCsys=None, orientationType=SYSTEM,
   region=Region(
   faces=mdb.models['FSWM'].parts['plate'].faces.getByBoundingBox(
   -1,-1,-1,a+1,b+1,1) ,), stackDirection=STACK_3)

#---------------------------------------------Create section-----------------------------------------------------------

#Choose here what material you want to work with
mdb.models['FSWM'].HomogeneousSolidSection(material='anisotropic', name=
   'Section-1', thickness=None)
mdb.models['FSWM'].parts['plate'].SectionAssignment(offset=0.0,
   offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
   faces=mdb.models['FSWM'].parts['plate'].faces.getSequenceFromMask(
   mask=('[#1 ]', ), )), sectionName='Section-1', thicknessAssignment=
   FROM_SECTION)

#---------------------------------------------Partition face-----------------------------------------------------------

#We apply the partitions in the section, because want the partition where the load is to be
#applied.

mdb.models['FSWM'].ConstrainedSketch(gridSpacing=0.001, name='__profile__',
   sheetSize=0.044, transform=
   mdb.models['FSWM'].parts['plate'].MakeSketchTransform(
   sketchPlane=mdb.models['FSWM'].parts['plate'].faces.findAt((
   a/2, b/2, 0.0), (0.0, 0.0, 1.0)), sketchPlaneSide=SIDE1,
   sketchOrientation=RIGHT, origin=(0.0, 0.0, 0.0)))
mdb.models['FSWM'].parts['plate'].projectReferencesOntoSketch(filter=
   COPLANAR_EDGES, sketch=mdb.models['FSWM'].sketches['__profile__'])
```

Finite-sized Wave Model

```
mdb.models['FSWM'].sketches['__profile__'].Line(point1=(0.0, b/2), point2=(
    a, b/2))
mdb.models['FSWM'].sketches['__profile__'].Line(point1=(0.0, b/2+c), point2=(
    a, b/2+c))
mdb.models['FSWM'].sketches['__profile__'].Line(point1=(a, b/2-c), point2=(
    0.0, b/2-c))
mdb.models['FSWM'].parts['plate'].PartitionFaceBySketch(faces=
    mdb.models['FSWM'].parts['plate'].faces.findAt(((a/2,
    b/2, 0.0), )), sketch=mdb.models['FSWM'].sketches['__profile__'])
del mdb.models['FSWM'].sketches['__profile__']


#--------------------------------------------Create assembly------------------------------------------------------------

mdb.models['FSWM'].rootAssembly.DatumCsysByDefault(CARTESIAN)
mdb.models['FSWM'].rootAssembly.Instance(dependent=ON, name='plate-1',
    part=mdb.models['FSWM'].parts['plate'])

#-----------------------------------------------Create step------------------------------------------------------------------

mdb.models['FSWM'].ExplicitDynamicsStep(name='dynamic',previous=
    'Initial', timePeriod=t,  quadBulkViscosity=0.0)

#----------------------------------------Define load and BCs-----------------------------------------------------------

#Determine the load shape in time
mdb.models['FSWM'].SmoothStepAmplitude(data=((0.0, 0.0), (loadTime-1E-5, 0.0), (
    loadTime, 1.0), (2*loadTime, 1.0), (2*loadTime+1E-5, 0.0)), name='impulse_smooth', timeSpan=STEP)

#Fixed end
mdb.models['FSWM'].DisplacementBC(amplitude=UNSET, createStepName='Initial',
    distributionType=UNIFORM, fieldName='', localCsys=None, name='fixedEnd',
    region=Region(
    edges=mdb.models['FSWM'].rootAssembly.instances['plate-1'].edges.findAt(
    ((a, b-eps, 0.0), ), ((a,eps,0.0),),((a, b/2-eps, 0.0), ), ((a,b/2+eps,0.0),),
    )), u1=SET, u2=SET, ur3=SET)

#Apply surface traction
mdb.models['FSWM'].SurfaceTraction(amplitude='impulse_smooth', createStepName=
    'dynamic', directionVector=(
    mdb.models['FSWM'].rootAssembly.instances['plate-1'].InterestingPoint(
    mdb.models['FSWM'].rootAssembly.instances['plate-1'].edges.findAt(
    (a, 0.0, 0.0), ), MIDDLE),
    mdb.models['FSWM'].rootAssembly.instances['plate-1'].vertices.findAt(
    (a, b/2, 0.0), )), distributionType=UNIFORM, field='', localCsys=None,
    magnitude=F, name='traction', region=Region(
    side1Edges=mdb.models['FSWM'].rootAssembly.instances['plate-1'].edges.findAt(
    ((0, b/2+eps, 0.0), ), ((0, b/2-eps, 0.0), ), )), resultant=ON)

#--------------------------------------------Generate mesh----------------------------------------------------------

mdb.models['FSWM'].parts['plate'].seedPart(deviationFactor=0.1,
    minSizeFactor=0.1, size=elSize)
mdb.models['FSWM'].parts['plate'].setMeshControls(elemShape=QUAD,
    regions=mdb.models['FSWM'].parts['plate'].faces.getByBoundingBox(
    -1,-1,-1,a+1,b+1,2 ), technique=STRUCTURED)
```

```
mdb.models['FSWM'].parts['plate'].setElementType(elemTypes=(ElemType(
   elemCode=CPE4R, elemLibrary=STANDARD, secondOrderAccuracy=OFF,
   hourglassControl=DEFAULT, distortionControl=DEFAULT), ElemType(
   elemCode=CPE4R, elemLibrary=STANDARD)), regions=(
   mdb.models['FSWM'].parts['plate'].faces.getByBoundingBox(
   -1,-1,-1,a+1,b+1,2 ),))
mdb.models['FSWM'].parts['plate'].generateMesh()

#----------------------------------------------Rotate plate----------------------------------------------------------------

mdb.models['FSWM'].rootAssembly.rotate(angle=theta, axisDirection=(0.0, 0.0,
   1.0), axisPoint=(0.0, 0.0, 0.0), instanceList=('plate-1', ))

#-------------------------------------------------Request outputs--------------------------------------------------------

#Field output
mdb.models['FSWM'].fieldOutputRequests['F-Output-1'].setValues(numIntervals=
   300,  directions=OFF, variables=('S', 'ENER','SVAVG', 'U', 'V', 'A'))

#---------------------------------------------Create job -----------------------------------------------------------------

mdb.models['FSWM'].rootAssembly.regenerate()
mdb.Job(activateLoadBalancing=False, atTime=None, contactPrint=OFF,
   description='', echoPrint=OFF, explicitPrecision=SINGLE, historyPrint=OFF,
   model='FSWM', modelPrint=OFF, multiprocessingMode=DEFAULT, name=
   'Finite-sized_Wave_Model', nodalOutputPrecision=SINGLE, numCpus=1, numDomains=
   1, parallelizationMethodExplicit=DOMAIN, queue=None, scratch='', type=
   ANALYSIS, userSubroutine='', waitHours=0, waitMinutes=0)
```

Curved Model

```
#This is a python script that creates an FEM model referred to as the Curved Model
#in the master thesis. Under the section "Define variables", the user can specify for example
#the geometry or mesh density.

#Written by Erik Grimsmo

#-----------------------------------------------------Initialize---------------------------------------------------------------


from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *

session.journalOptions.setValues(replayGeometry=COORDINATE,recoverGeometry=COORDINATE)

#Change work directory (use your personal directory, and uncomment the line below)
#os.chdir(r'D:\abaqusworkdirectory')

#---------------------------------------------Define variables---------------------------------------------------------------

a = 0.025 #Short axis of ellipse
b = 0.045 #Long axis of ellipse
c = 0.015 #Dimension used to define height of ellipse
th = 0.01 #Thickness (of heart wall)
d = 0.01 #Depth of extrusion

numberOfPartitions = 10 #Number of partitions of the myocardium
#through it thickness (select even number)

rho = 1060.0 #Density

F = 1.0 #The amplitude of the applied traction load

t = 0.0005 #Period of time integration
c_t = 100 #Approx longitudinal wave speed
loadTime = 1E-4 #Duration of blast load
N = 20 #Number of elements over which the load will take place
elSize = loadTime*c_t/n #Element size

#Do not alter the script after this section,
#unless you are familiar with python scripting in Abaqus

#---------------------------------------------Create model--------------------------------------------------------------

eps = 0.0001 #A variable that helps with modeling
r = sqrt(1-(c/b)**2)*a #A dimension that is used a lot
```

Curved Model

```
mdb.Model(name='CM')
mdb.models['CM'].setValues(noPartsInputFile=ON)

#--------------------------------------------Create sketch-----------------------------------------------------------

mdb.models['CM'].ConstrainedSketch(name='__profile__', sheetSize=1.0)
mdb.models['CM'].sketches['__profile__'].ConstructionLine(point1=(0.0,
   -0.5), point2=(0.0, 0.5))

mdb.models['CM'].sketches['__profile__'].EllipseByCenterPerimeter(
   axisPoint1=(0.0, -b), axisPoint2=(a, 0.0), center=(0.0, 0.0))
mdb.models['CM'].sketches['__profile__'].EllipseByCenterPerimeter(
   axisPoint1=(0.0, -b+th), axisPoint2=(a-th, 0.0), center=(0.0, 0.0))
mdb.models['CM'].sketches['__profile__'].Line(point1=(0.0,
   -b+th), point2=(0.0, -b))
mdb.models['CM'].sketches['__profile__'].Line(point1=(
   sqrt(1-(c/(b-th))**2)*(a-th), c), point2=(r, c))

mdb.models['CM'].sketches['__profile__'].geometry.findAt((-a, 0.0))
mdb.models['CM'].sketches['__profile__'].autoTrimCurve(curve1=
   mdb.models['CM'].sketches['__profile__'].geometry.findAt((-a,
   0.0), ), point1=(-a, 0.0))
mdb.models['CM'].sketches['__profile__'].geometry.findAt((-a+th, 0.0))
mdb.models['CM'].sketches['__profile__'].autoTrimCurve(curve1=
   mdb.models['CM'].sketches['__profile__'].geometry.findAt((-a+th, 0.0), ),
   point1=(-a+th, 0.0))
mdb.models['CM'].sketches['__profile__'].geometry.findAt((0, b))
mdb.models['CM'].sketches['__profile__'].autoTrimCurve(curve1=
   mdb.models['CM'].sketches['__profile__'].geometry.findAt((0, b), ),
   point1=(0, b))
mdb.models['CM'].sketches['__profile__'].geometry.findAt((0, b-th))
mdb.models['CM'].sketches['__profile__'].autoTrimCurve(curve1=
   mdb.models['CM'].sketches['__profile__'].geometry.findAt((0, b-th), ),
   point1=(0, b-th))

mdb.models['CM'].Part(dimensionality=THREE_D, name='Part-1', type=
   DEFORMABLE_BODY)
mdb.models['CM'].parts['Part-1'].BaseSolidExtrude(depth=d, sketch=
   mdb.models['CM'].sketches['__profile__'])
del mdb.models['CM'].sketches['__profile__']

#--------------------------------------------Define material---------------------------------------------------------

#Isotropic material
mdb.models['CM'].Material(name='isotropic')
mdb.models['CM'].materials['isotropic'].Density(table=((1060.0, ), ))
mdb.models['CM'].materials['isotropic'].Elastic(table=((31588000.0, 0.49), ))

#Transverse isotropic material
mdb.models['CM'].Material(name='anisotropic')
mdb.models['CM'].materials['anisotropic'].Elastic(table=((2460000000.0,
   2443100000.0, 2460000000.0, 2440000000.0, 2440000000.0, 2530000000.0, 0.0,
   0.0, 0.0, 8970000.0, 0.0, 0.0, 0.0, 0.0, 8970000.0, 0.0, 0.0, 0.0, 0.0,
   0.0, 8450000.0), ), type=ANISOTROPIC)
mdb.models['CM'].materials['anisotropic'].Density(table=((rho, ), ))
```

Curved Model

```
#---------------------------------------------Create section---------------------------------------------------------------

#Choose here what material you want to work with
mdb.models['CM'].HomogeneousSolidSection(material='anisotropic', name=
   'Section-1', thickness=None)
mdb.models['CM'].parts['Part-1'].SectionAssignment(offset=0.0, offsetField=''
   , offsetType=MIDDLE_SURFACE, region=Region(
   cells=mdb.models['CM'].parts['Part-1'].cells.findAt(((a, 0,
   0), ), )), sectionName='Section-1', thicknessAssignment=
   FROM_SECTION)

#-----------------------------------Partition thickness of the model-------------------------------------------------

mdb.models['CM'].ConstrainedSketch(gridSpacing=0.02, name='__profile__',
   sheetSize=1.07, transform=
   mdb.models['CM'].parts['Part-1'].MakeSketchTransform(
   sketchPlane=mdb.models['CM'].parts['Part-1'].faces.findAt((a-th/2,
   0.0, 0.0), ), sketchPlaneSide=SIDE1,
   sketchUpEdge=mdb.models['CM'].parts['Part-1'].edges.findAt((0.0, -b+th/2,
   0.0), ), sketchOrientation=RIGHT, origin=(0.0, 0.0, 0.0)))
mdb.models['CM'].parts['Part-1'].projectReferencesOntoSketch(filter=
   COPLANAR_EDGES, sketch=mdb.models['CM'].sketches['__profile__'])

l1 = th/numberOfPartitions #Length of one partition

for i in range(numberOfPartitions-1):
   mdb.models['CM'].sketches['__profile__'].EllipseByCenterPerimeter(
      axisPoint1=(0.0, b-(i+1)*l1), axisPoint2=(a-(i+1)*l1, 0.0), center=(0.0, 0.0))

mdb.models['CM'].parts['Part-1'].PartitionFaceBySketch(faces=
   mdb.models['CM'].parts['Part-1'].faces.findAt(((a-th/2, 0.0,
   0.0), )), sketch=mdb.models['CM'].sketches['__profile__'],
   sketchUpEdge=mdb.models['CM'].parts['Part-1'].edges.findAt((0.0, -b+th/2,
   0.0), ))

for i in range(numberOfPartitions-1):
   mdb.models['CM'].parts['Part-1'].PartitionCellBySweepEdge(cells=
      mdb.models['CM'].parts['Part-1'].cells.findAt(((a, 0.0,
      d/2), )), edges=(mdb.models['CM'].parts['Part-1'].edges.findAt((
      a-th+(i+1)*l1, 0.0, 0), ), ), sweepPath=
      mdb.models['CM'].parts['Part-1'].edges.findAt((r, c,
      d/2), ))

#Finally, material orientation can be given to each cell
for i in range(numberOfPartitions):
   mdb.models['CM'].parts['Part-1'].MaterialOrientation(
      additionalRotationField='', additionalRotationType=ROTATION_ANGLE, angle=
      60-i*120/(numberOfPartitions-1), axis=AXIS_1, flipNormalDirection=False,
      flipPrimaryDirection=False,
      normalAxisDefinition=SURFACE, normalAxisDirection=AXIS_1, normalAxisRegion=
      Region(side1Faces=mdb.models['CM'].parts['Part-1'].faces.findAt(((
      a, 0.0, d/2), ), )), orientationType=DISCRETE,
      primaryAxisDefinition=EDGE, primaryAxisDirection=AXIS_3, primaryAxisRegion=
      Region(edges=mdb.models['CM'].parts['Part-1'].edges.findAt(((0.0,
      -b, d/2), ), )), region=Region(
    cells=mdb.models['CM'].parts['Part-1'].cells.findAt(((a-i*l1-eps, 0.0,
    d/2), ), )), stackDirection=STACK_3)


#--------------------------------------------Other partitions----------------------------------------------------------
#The following partitions are created such that we can define sets where the stress
#vs. time will be extracted

myDatum1 = mdb.models['CM'].parts['Part-1'].DatumPlaneByPrincipalPlane(offset=0.0,
   principalPlane=XZPLANE)
myDatum2 = mdb.models['CM'].parts['Part-1'].DatumPlaneByPrincipalPlane(offset=-c,
   principalPlane=XZPLANE)
myDatum3 = mdb.models['CM'].parts['Part-1'].DatumPlaneByPrincipalPlane(offset=d/2,
   principalPlane=XYPLANE)

mdb.models['CM'].parts['Part-1'].PartitionCellByDatumPlane(cells=
   mdb.models['CM'].parts['Part-1'].cells.getByBoundingBox(
   -1,-b-1,-1,a+1,c+1,d+1), datumPlane=
   mdb.models['CM'].parts['Part-1'].datums[myDatum1.id])
mdb.models['CM'].parts['Part-1'].PartitionCellByDatumPlane(cells=
   mdb.models['CM'].parts['Part-1'].cells.getByBoundingBox(
   -1,-b-1,-1,a+1,c+1,d+1), datumPlane=
   mdb.models['CM'].parts['Part-1'].datums[myDatum2.id])
mdb.models['CM'].parts['Part-1'].PartitionCellByDatumPlane(cells=
   mdb.models['CM'].parts['Part-1'].cells.getByBoundingBox(
   -1,-b-1,-1,a+1,c+1,d+1), datumPlane=
   mdb.models['CM'].parts['Part-1'].datums[myDatum3.id])

#Define the sets
mdb.models['CM'].parts['Part-1'].Set(name='node1b', vertices=
   mdb.models['CM'].parts['Part-1'].vertices.findAt(((a-th/2, 0.0, d/2),
   )))
mdb.models['CM'].parts['Part-1'].Set(name='node1c', vertices=
   mdb.models['CM'].parts['Part-1'].vertices.findAt(((a-l1, 0.0, d/2),
   )))
mdb.models['CM'].parts['Part-1'].Set(name='node1a', vertices=
   mdb.models['CM'].parts['Part-1'].vertices.findAt(((
   a-(numberOfPartitions-1)*l1, 0.0, d/2),  )))
mdb.models['CM'].parts['Part-1'].Set(name='node2', vertices=
   mdb.models['CM'].parts['Part-1'].vertices.findAt(((
   sqrt(1-(c/(b-numberOfPartitions/2*l1))**2)*(a-numberOfPartitions/2*l1), -c,
   d/2), )))

#--------------------------------------------Create assembly-----------------------------------------------------------

mdb.models['CM'].rootAssembly.DatumCsysByDefault(CARTESIAN)
mdb.models['CM'].rootAssembly.Instance(dependent=OFF, name='CM-1',
    part=mdb.models['CM'].parts['Part-1'])


#----------------------------------------------Create step-------------------------------------------------------------

mdb.models['CM'].ExplicitDynamicsStep(name='dynamic', previous=
     'Initial', timePeriod=t,  quadBulkViscosity=1.2)

#---------------------------------------Define load and BCs-----------------------------------------------------------
```

Curved Model

```
#Fixed end
mdb.models['CM'].EncastreBC(createStepName='Initial', localCsys=None, name=
   'encastre', region=Region(
   faces=mdb.models['CM'].rootAssembly.instances['CM-1'].faces.getByBoundingBox(
   -eps,-b-eps,-eps,eps,-b+th+eps,d+eps)))

#Fix sides
mdb.models['CM'].DisplacementBC(amplitude=UNSET, createStepName='Initial',
   distributionType=UNIFORM, fieldName='', localCsys=None, name='fix U1 and U2 1',
   region=Region(
   faces=mdb.models['CM'].rootAssembly.instances['CM-1'].faces.getByBoundingBox(
   -1,-b-1,-eps,a+1,c+1,eps)), u1=SET, u2=SET, u3=UNSET, ur1=UNSET,
   ur2=UNSET, ur3=UNSET)
mdb.models['CM'].DisplacementBC(amplitude=UNSET, createStepName='Initial',
   distributionType=UNIFORM, fieldName='', localCsys=None, name='fix U1 and U2 2',
   region=Region(
   faces=mdb.models['CM'].rootAssembly.instances['CM-1'].faces.getByBoundingBox(
   -1,-b-1,d-eps,a+1,c+1,d+eps)), u1=SET, u2=SET, u3=UNSET, ur1=UNSET,
   ur2=UNSET, ur3=UNSET)

 #Determine the load shape in time
mdb.models['CM'].SmoothStepAmplitude(data=((0.0, 0.0), (1E-4-1E-5, 0.0), (
     1E-4, 1.0), (1E-4+loadTime, 1.0), (1E-4+loadTime+1E-5, 0.0)), name='impulse_smooth', timeSpan=
     STEP)

#Apply surface traction
mdb.models['CM'].SurfaceTraction(amplitude='impulse_smooth',
   createStepName='dynamic', directionVector=(
   mdb.models['CM'].rootAssembly.instances['CM-1'].vertices.findAt((
   a, 0.0, 0.0), ),
   mdb.models['CM'].rootAssembly.instances['CM-1'].vertices.findAt((
   a, 0.0, d), )), distributionType=UNIFORM, field='', localCsys=None
   , magnitude=F, name='shear', region=Region(
   side1Faces=mdb.models['CM'].rootAssembly.instances['CM-1'].faces.getByBoundingBox(
   0,c-eps,-1,a+1,c+eps,d+1 )))

#---------------------------------------------Generate mesh-------------------------------------------------------------

mdb.models['CM'].rootAssembly.seedPartInstance(deviationFactor=0.1,
   minSizeFactor=0.1, regions=(
   mdb.models['CM'].rootAssembly.instances['CM-1'], ), size=elSize)
mdb.models['CM'].rootAssembly.setElementType(elemTypes=(ElemType(
   elemCode=C3D8R, elemLibrary=EXPLICIT, secondOrderAccuracy=OFF,
   kinematicSplit=AVERAGE_STRAIN, hourglassControl=DEFAULT,
   distortionControl=DEFAULT), ElemType(elemCode=C3D6, elemLibrary=EXPLICIT),
   ElemType(elemCode=C3D4, elemLibrary=EXPLICIT)), regions=(
   mdb.models['CM'].rootAssembly.instances['CM-1'].cells.getByBoundingBox(
   -1,-b-1,-1,a+1,c+1,d+1), ))
mdb.models['CM'].rootAssembly.generateMesh(regions=(
   mdb.models['CM'].rootAssembly.instances['CM-1'],))

#------------------------------------------------Request outputs-----------------------------------------------------------

mdb.models['CM'].fieldOutputRequests['F-Output-1'].setValues(directions=OFF,
   numIntervals=400, variables=('S', 'SVAVG',  'U','ENER' ,))
```

```
#-----------------------------------------------Create job ----------------------------------------------------------------------

mdb.models['CM'].rootAssembly.regenerate()
mdb.Job(activateLoadBalancing=False, atTime=None, contactPrint=OFF,
    description='', echoPrint=OFF, explicitPrecision=SINGLE, historyPrint=OFF,
    model='CM', modelPrint=OFF, multiprocessingMode=DEFAULT, name=
    'Curved_Model', nodalOutputPrecision=SINGLE, numCpus=1, numDomains=1,
    parallelizationMethodExplicit=DOMAIN, queue=None, scratch='', type=ANALYSIS
    , userSubroutine='', waitHours=0, waitMinutes=0)
```

Truncated Ellipsoid Model

```
#This is a python script that creates an FEM model referred to as the Truncated Ellipsoid Model
#in the master thesis. Under the section "Define variables", the user can specify for example
#the geometry or mesh density.

#Written by Erik Grimsmo

#----------------------------------------------------Initialize-----------------------------------------------------------

from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *

session.journalOptions.setValues(replayGeometry=COORDINATE,recoverGeometry=COORDINATE)

#Change work directory (use your personal directory, and uncomment the line below)
#os.chdir(r'D:\abaqusworkdirectory')

#---------------------------------------------Define variables------------------------------------------------------------

a = 0.025 #Short axis of ellipsoid
b = 0.045 #Long axis of ellipsoid
c = 0.015 #Dimension used to define height of ellipsoid
th = 0.01 #Thickness of heart wall

numberOfPartitions = 10 #Number of partitions of the myocardium
#through it thickness (select even number)

rho = 1060.0 #Density

F = 1.0 #The amplitude of the applied traction load

t = 0.0005 #Period of time integration
c_t= 100 #Approx transverse wave speed
loadTime = 1E-4 #Duration of blast load
N = 10 #Number of elements over which the load will take place
elSize = loadTime*c_t/n #Element size

#Do not alter the script after this section,
#unless you are familiar with python scripting in Abaqus

#---------------------------------------------Create model-------------------------------------------------------------

eps = 0.00001 #A variable that helps with modeling

mdb.Model(name='TEM')
mdb.models['TEM'].setValues(noPartsInputFile=ON)
```

Truncated Ellipsoid Model

```
#---------------------------------------------Create sketch-----------------------------------------------------------

mdb.models['TEM'].ConstrainedSketch(name='__profile__', sheetSize=1.0)
mdb.models['TEM'].sketches['__profile__'].ConstructionLine(point1=(0.0,
   -0.5), point2=(0.0, 0.5))

mdb.models['TEM'].sketches['__profile__'].EllipseByCenterPerimeter(
   axisPoint1=(0.0, -b), axisPoint2=(a, 0.0), center=(0.0, 0.0))
mdb.models['TEM'].sketches['__profile__'].EllipseByCenterPerimeter(
   axisPoint1=(0.0, -b+th), axisPoint2=(a-th, 0.0), center=(0.0, 0.0))
mdb.models['TEM'].sketches['__profile__'].Line(point1=(0,
   -b), point2=(0, -b+th))
mdb.models['TEM'].sketches['__profile__'].Line(point1=(
   sqrt(1-(c/(b-th))**2)*(a-th), c), point2=(sqrt(1-(c/b)**2)*a, c))

mdb.models['TEM'].sketches['__profile__'].geometry.findAt((-a, 0.0))
mdb.models['TEM'].sketches['__profile__'].autoTrimCurve(curve1=
   mdb.models['TEM'].sketches['__profile__'].geometry.findAt((-a,
   0.0), ), point1=(-a, 0.0))
mdb.models['TEM'].sketches['__profile__'].geometry.findAt((-a+th, 0.0))
mdb.models['TEM'].sketches['__profile__'].autoTrimCurve(curve1=
   mdb.models['TEM'].sketches['__profile__'].geometry.findAt((-a+th, 0.0), ),
   point1=(-a+th, 0.0))
mdb.models['TEM'].sketches['__profile__'].geometry.findAt((eps, b))
mdb.models['TEM'].sketches['__profile__'].autoTrimCurve(curve1=
   mdb.models['TEM'].sketches['__profile__'].geometry.findAt((eps, b), ),
   point1=(eps, b))
mdb.models['TEM'].sketches['__profile__'].geometry.findAt((eps, b-th))
mdb.models['TEM'].sketches['__profile__'].autoTrimCurve(curve1=
   mdb.models['TEM'].sketches['__profile__'].geometry.findAt((eps, b-th), ),
   point1=(eps, b-th))

mdb.models['TEM'].Part(dimensionality=THREE_D, name='ellipsoid', type=
   DEFORMABLE_BODY)
mdb.models['TEM'].parts['ellipsoid'].BaseSolidRevolve(angle=360.0,
   flipRevolveDirection=OFF, sketch=
   mdb.models['TEM'].sketches['__profile__'])
del mdb.models['TEM'].sketches['__profile__']

#---------------------------------------------Define material---------------------------------------------------------

#Isotropic material
mdb.models['TEM'].Material(name='isotropic')
mdb.models['TEM'].materials['isotropic'].Density(table=((1060.0, ), ))
mdb.models['TEM'].materials['isotropic'].Elastic(table=((39765628.0, 0.49), ))

#Anisotropic material
mdb.models['TEM'].Material(name='anisotropic')
mdb.models['TEM'].materials['anisotropic'].Elastic(table=((2460000000.0,
   2443100000.0, 2460000000.0, 2440000000.0, 2440000000.0, 2530000000.0, 0.0,
   0.0, 0.0, 8970000.0, 0.0, 0.0, 0.0, 0.0, 8970000.0, 0.0, 0.0, 0.0, 0.0,
   0.0, 8450000.0), ), type=ANISOTROPIC)
mdb.models['TEM'].materials['anisotropic'].Density(table=((rho, ), ))

#---------------------------------------------Create section-----------------------------------------------------------
```

Truncated Ellipsoid Model

```
#Choose here what material you want to work with
mdb.models['TEM'].HomogeneousSolidSection(material='anisotropic', name=
    'Section-1', thickness=None)
mdb.models['TEM'].parts['ellipsoid'].SectionAssignment(offset=0.0, offsetField=''
    , offsetType=MIDDLE_SURFACE, region=Region(
    cells=mdb.models['TEM'].parts['ellipsoid'].cells.findAt(((a, 0,
    0), ), )), sectionName='Section-1', thicknessAssignment=
    FROM_SECTION)

#----------------Create the partitions that are to have different material orientations-------------------

#Partition faces
mdb.models['TEM'].parts['ellipsoid'].PartitionCellByPlaneThreePoints(cells=
    mdb.models['TEM'].parts['ellipsoid'].cells.findAt(((a, 0.0,
    0.0), )), point1=mdb.models['TEM'].parts['ellipsoid'].vertices.findAt((
    sqrt(1-(c/(b-th))**2)*(a-th), c, 0.0), ), point2=
    mdb.models['TEM'].parts['ellipsoid'].vertices.findAt((sqrt(1-(c/b)**2)*a, c, 0.0), )
    , point3=mdb.models['TEM'].parts['ellipsoid'].vertices.findAt((0.0, -b,
    0.0), ))
mdb.models['TEM'].parts['ellipsoid'].PartitionCellByPlaneThreePoints(cells=
    mdb.models['TEM'].parts['ellipsoid'].cells.findAt(((0.0, 0.0,
    a), ), ((0.0, 0.0, -a), )), point1=
    mdb.models['TEM'].parts['ellipsoid'].InterestingPoint(
    mdb.models['TEM'].parts['ellipsoid'].edges.findAt((0.0, c, sqrt(1-(c/(b-th))**2)*(a-th)),
    ), MIDDLE), point2=mdb.models['TEM'].parts['ellipsoid'].InterestingPoint(
    mdb.models['TEM'].parts['ellipsoid'].edges.findAt((0.0, c, -sqrt(1-(c/(b-th))**2)*(a-th)
    ), ), MIDDLE), point3=
    mdb.models['TEM'].parts['ellipsoid'].vertices.findAt((0.0, -b, 0.0), ))

mdb.models['TEM'].ConstrainedSketch(gridSpacing=0.03, name='__profile__',
    sheetSize=1.56, transform=
    mdb.models['TEM'].parts['ellipsoid'].MakeSketchTransform(
    sketchPlane=mdb.models['TEM'].parts['ellipsoid'].faces.findAt((a-th/2,
    0.0, 0.0), ), sketchPlaneSide=SIDE1,
    sketchUpEdge=mdb.models['TEM'].parts['ellipsoid'].edges.findAt((sqrt(1-(c/b)**2)*a-th/2,
    c, 0.0), ), sketchOrientation=TOP, origin=(0.0, 0.0, 0.0)))
mdb.models['TEM'].parts['ellipsoid'].projectReferencesOntoSketch(filter=
    COPLANAR_EDGES, sketch=mdb.models['TEM'].sketches['__profile__'])

l1 = th/numberOfPartitions

for i in range(numberOfPartitions-1):
    mdb.models['TEM'].sketches['__profile__'].EllipseByCenterPerimeter(axisPoint1=(
        0.0, -b+(i+1)*l1), axisPoint2=(a-(i+1)*l1, 0.0), center=(0.0, 0.0))

mdb.models['TEM'].parts['ellipsoid'].PartitionFaceBySketch(faces=
    mdb.models['TEM'].parts['ellipsoid'].faces.findAt(((a-th/2, 0.0,
    0.0), ), ), sketch=
    mdb.models['TEM'].sketches['__profile__'], sketchOrientation=TOP,
    sketchUpEdge=mdb.models['TEM'].parts['ellipsoid'].edges.findAt((sqrt(1-(c/b)**2)*a-th/2,
    c, 0.0), ))
del mdb.models['TEM'].sketches['__profile__']

#Partition cells
for i in range(numberOfPartitions-1):
    mdb.models['TEM'].parts['ellipsoid'].PartitionCellBySweepEdge(cells=
        mdb.models['TEM'].parts['ellipsoid'].cells.findAt(((eps, 0.0,
        -a+th), )), edges=(mdb.models['TEM'].parts['ellipsoid'].edges.findAt((
        a-(i+1)*l1, 0.0, 0.0), ), ), sweepPath=
        mdb.models['TEM'].parts['ellipsoid'].edges.findAt((eps, c,-sqrt(1-(c/b)**2)*a
        ), ))
    mdb.models['TEM'].parts['ellipsoid'].PartitionCellBySweepEdge(cells=
        mdb.models['TEM'].parts['ellipsoid'].cells.findAt(((eps, 0.0,
        a-th), )), edges=(mdb.models['TEM'].parts['ellipsoid'].edges.findAt((
        a-(i+1)*l1, 0.0, 0.0), ), ), sweepPath=
        mdb.models['TEM'].parts['ellipsoid'].edges.findAt((eps, c,sqrt(1-(c/b)**2)*a
        ), ))
    mdb.models['TEM'].parts['ellipsoid'].PartitionCellBySweepEdge(cells=
        mdb.models['TEM'].parts['ellipsoid'].cells.findAt(((-eps, 0.0,
        -a+th), )), edges=(mdb.models['TEM'].parts['ellipsoid'].edges.findAt((
        0.0, 0.0, -a+(i+1)*l1), ), ), sweepPath=
        mdb.models['TEM'].parts['ellipsoid'].edges.findAt((-eps, c,-sqrt(1-(c/b)**2)*a
        ), ))
    mdb.models['TEM'].parts['ellipsoid'].PartitionCellBySweepEdge(cells=
        mdb.models['TEM'].parts['ellipsoid'].cells.findAt(((-eps, 0.0,
        a-th), )), edges=(mdb.models['TEM'].parts['ellipsoid'].edges.findAt((
        0.0, 0.0, a-(i+1)*l1), ), ), sweepPath=
        mdb.models['TEM'].parts['ellipsoid'].edges.findAt((-eps, c,sqrt(1-(c/b)**2)*a
        ), ))

#-------------------------------------Assign material orientations---------------------------------------------------

for i in range(numberOfPartitions):
    mdb.models['TEM'].parts['ellipsoid'].MaterialOrientation(
        additionalRotationField='', additionalRotationType=ROTATION_ANGLE, angle=
        -30-i*120/(numberOfPartitions-1), axis=AXIS_1, flipNormalDirection=False, flipPrimaryDirection=
        False,
        normalAxisDefinition=SURFACE, normalAxisDirection=AXIS_1, normalAxisRegion=
        Region(side1Faces=mdb.models['TEM'].parts['ellipsoid'].faces.findAt(((
        a, 0.0, -eps), ), )), orientationType=DISCRETE,
        primaryAxisDefinition=EDGE, primaryAxisDirection=AXIS_2, primaryAxisRegion=
        Region(edges=mdb.models['TEM'].parts['ellipsoid'].edges.findAt(((sqrt(1-(c/b)**2)*a,
        c, -eps), ), )), region=Region(
        cells=mdb.models['TEM'].parts['ellipsoid'].cells.findAt(((a-i*l1-eps,
        0.0, -eps), ), )), stackDirection=STACK_3)
    mdb.models['TEM'].parts['ellipsoid'].MaterialOrientation(
        additionalRotationField='', additionalRotationType=ROTATION_ANGLE, angle=
        -30-i*120/(numberOfPartitions-1), axis=AXIS_1, flipNormalDirection=False, flipPrimaryDirection=
        False,
        normalAxisDefinition=SURFACE, normalAxisDirection=AXIS_1, normalAxisRegion=
        Region(side1Faces=mdb.models['TEM'].parts['ellipsoid'].faces.findAt(((
        a, 0.0, eps), ), )), orientationType=DISCRETE,
        primaryAxisDefinition=EDGE, primaryAxisDirection=AXIS_2, primaryAxisRegion=
        Region(edges=mdb.models['TEM'].parts['ellipsoid'].edges.findAt(((sqrt(1-(c/b)**2)*a,
        c, eps), ), )), region=Region(
        cells=mdb.models['TEM'].parts['ellipsoid'].cells.findAt(((a-i*l1-eps,
        0.0, eps), ), )), stackDirection=STACK_3)
    mdb.models['TEM'].parts['ellipsoid'].MaterialOrientation(
        additionalRotationField='', additionalRotationType=ROTATION_ANGLE, angle=
        -30-i*120/(numberOfPartitions-1), axis=AXIS_1, flipNormalDirection=False, flipPrimaryDirection=
        False,
        normalAxisDefinition=SURFACE, normalAxisDirection=AXIS_1, normalAxisRegion=
        Region(side1Faces=mdb.models['TEM'].parts['ellipsoid'].faces.findAt(((
        -a, 0.0, -eps), ), )), orientationType=DISCRETE,
        primaryAxisDefinition=EDGE, primaryAxisDirection=AXIS_2, primaryAxisRegion=
        Region(edges=mdb.models['TEM'].parts['ellipsoid'].edges.findAt(((-sqrt(1-(c/b)**2)*a,
        c, -eps), ), )), region=Region(
        cells=mdb.models['TEM'].parts['ellipsoid'].cells.findAt(((-a+i*l1+eps,
        0.0, -eps), ), )), stackDirection=STACK_3)
    mdb.models['TEM'].parts['ellipsoid'].MaterialOrientation(
        additionalRotationField='', additionalRotationType=ROTATION_ANGLE, angle=
        -30-i*120/(numberOfPartitions-1), axis=AXIS_1, flipNormalDirection=False, flipPrimaryDirection=
        False,
        normalAxisDefinition=SURFACE, normalAxisDirection=AXIS_1, normalAxisRegion=
        Region(side1Faces=mdb.models['TEM'].parts['ellipsoid'].faces.findAt(((
        -a, 0.0, eps), ), )), orientationType=DISCRETE,
        primaryAxisDefinition=EDGE, primaryAxisDirection=AXIS_2, primaryAxisRegion=
        Region(edges=mdb.models['TEM'].parts['ellipsoid'].edges.findAt(((-sqrt(1-(c/b)**2)*a,
        c, eps), ), )), region=Region(
        cells=mdb.models['TEM'].parts['ellipsoid'].cells.findAt(((-a+i*l1+eps,
        0.0, eps), ), )), stackDirection=STACK_3)


#----------------------------------------------Create assembly-------------------------------------------------------------

mdb.models['TEM'].rootAssembly.DatumCsysByDefault(CARTESIAN)
mdb.models['TEM'].rootAssembly.Instance(dependent=OFF, name='ellipsoid-1',
    part=mdb.models['TEM'].parts['ellipsoid'])

#------------------------------------------------Create step-------------------------------------------------------------------

mdb.models['TEM'].ExplicitDynamicsStep(name='dynamic', previous=
    'Initial', timePeriod=t,  quadBulkViscosity=1.2)

#----------------------------------------Define load and BCs-------------------------------------------------------------

#BC
mdb.models['TEM'].DisplacementBC(amplitude=UNSET, createStepName='Initial',
    distributionType=UNIFORM, fieldName='', localCsys=None, name='fix', region=
    Region(
    vertices=mdb.models['TEM'].rootAssembly.instances['ellipsoid-1'].vertices.findAt(
    ((0.0, -b, 0.0), ), )), u1=SET, u2=SET, u3=SET, ur1=SET, ur2=SET, ur3=
    SET)

#Create partitions necessary to be able to apply load and at the same time maintain a structured mesh
myDatum1 = mdb.models['TEM'].rootAssembly.DatumPlaneByPrincipalPlane(offset=0.0,
    principalPlane=XZPLANE)
myDatum3 = mdb.models['TEM'].rootAssembly.DatumPlaneByPrincipalPlane(offset=-c,
    principalPlane=XZPLANE)
mdb.models['TEM'].rootAssembly.PartitionCellByDatumPlane(cells=
    mdb.models['TEM'].rootAssembly.instances['ellipsoid-1'].cells.getByBoundingBox(
    -2*b,-2*b,-2*b,2*b,2*b,2*b), datumPlane=mdb.models['TEM'].rootAssembly.datums[myDatum1.id])
mdb.models['TEM'].rootAssembly.PartitionCellByDatumPlane(cells=
    mdb.models['TEM'].rootAssembly.instances['ellipsoid-1'].cells.getByBoundingBox(
    -2*b,-2*b,-2*b,2*b,2*b,2*b), datumPlane=mdb.models['TEM'].rootAssembly.datums[myDatum3.id])

#Create surface partition where the load is to be applied
```

Truncated Ellipsoid Model

```
mdb.models['TEM'].ConstrainedSketch(gridSpacing=0.02, name='__profile__',
   sheetSize=1.0, transform=mdb.models['TEM'].rootAssembly.MakeSketchTransform(
   sketchPlane=mdb.models['TEM'].rootAssembly.instances['ellipsoid-1'].faces.findAt(
   (sqrt(1-(c/b)**2)*a-th/2, c, -eps), ), sketchPlaneSide=SIDE1,
   sketchUpEdge=mdb.models['TEM'].rootAssembly.instances['ellipsoid-1'].edges.findAt(
   (sqrt(1-(c/b)**2)*a, c, -eps), ), sketchOrientation=RIGHT, origin=(0.0, c,
   0.0)))
mdb.models['TEM'].rootAssembly.projectReferencesOntoSketch(filter=COPLANAR_EDGES
   , sketch=mdb.models['TEM'].sketches['__profile__'])
mdb.models['TEM'].sketches['__profile__'].Line(point1=(0.0, 0.0), point2=(
   -sqrt(1-(c/b)**2)*a*cos(pi/16), sqrt(1-(c/b)**2)*a*sin(pi/16)))
mdb.models['TEM'].sketches['__profile__'].Line(point1=(0.0, 0.0), point2=(
   -sqrt(1-(c/b)**2)*a*cos(pi/16), -sqrt(1-(c/b)**2)*a*sin(pi/16)))
mdb.models['TEM'].rootAssembly.PartitionFaceBySketch(faces=
   mdb.models['TEM'].rootAssembly.instances['ellipsoid-1'].faces.getByBoundingBox(
   -a,c-eps,-a,a,c+eps,a ), sketch=
   mdb.models['TEM'].sketches['__profile__'], sketchUpEdge=
   mdb.models['TEM'].rootAssembly.instances['ellipsoid-1'].edges.findAt((
   sqrt(1-(c/b)**2)*a, c, -eps), ))
del mdb.models['TEM'].sketches['__profile__']

 #Determine the load shape in time
mdb.models['TEM'].SmoothStepAmplitude(data=((0.0, 0.0), (1E-4-1E-5, 0.0), (
   1E-4, 1.0), (1E-4+loadTime, 1.0), (1E-4+loadTime+1E-5, 0.0)), name='impulse_smooth', timeSpan=STEP)

#Apply surface traction
mdb.models['TEM'].SurfaceTraction(amplitude='impulse_smooth', createStepName=
   'dynamic', directionVector=(
   mdb.models['TEM'].rootAssembly.instances['ellipsoid-1'].vertices.findAt((
   0.0, 0.0, -a), ),
   mdb.models['TEM'].rootAssembly.instances['ellipsoid-1'].vertices.findAt((
   0.0, 0.0, a), )), distributionType=UNIFORM, field='',
   localCsys=None, magnitude=F, name='shear', region=Region(
   side1Faces=mdb.models['TEM'].rootAssembly.instances['ellipsoid-1'].faces.getByBoundingBox(
   -a,c-eps,-a,-eps,c+eps,a )))

#---------------------------------------------Generate mesh-------------------------------------------------------------

mdb.models['TEM'].rootAssembly.seedPartInstance(deviationFactor=0.1,
   minSizeFactor=0.1, regions=(
   mdb.models['TEM'].rootAssembly.instances['ellipsoid-1'], ), size=elSize)
mdb.models['TEM'].rootAssembly.generateMesh(regions=(
   mdb.models['TEM'].rootAssembly.instances['ellipsoid-1'], ))

#-------------------------------------------------Request outputs----------------------------------------------------------

#Field output
mdb.models['TEM'].fieldOutputRequests['F-Output-1'].setValues(directions=OFF,
   numIntervals=400, variables=('S', 'SVAVG', 'ENER', ))

#Define sets
mdb.models['TEM'].rootAssembly.Set(name='node1c', vertices=
   mdb.models['TEM'].rootAssembly.instances['ellipsoid-1'].vertices.findAt(((
   -a+l1, 0.0, 0.0), )))
mdb.models['TEM'].rootAssembly.Set(name='node1b', vertices=
   mdb.models['TEM'].rootAssembly.instances['ellipsoid-1'].vertices.findAt(((
   -a+th/2, 0.0, 0.0), )))
mdb.models['TEM'].rootAssembly.Set(name='node1a', vertices=
   mdb.models['TEM'].rootAssembly.instances['ellipsoid-1'].vertices.findAt(((
   -a+(numberOfPartitions-1)*l1, 0.0, 0.0), )))

#History output
mdb.models['TEM'].HistoryOutputRequest(createStepName='dynamic', frequency=10,
   name='H-Output-1', rebar=EXCLUDE, region=
   mdb.models['TEM'].rootAssembly.sets['node1b'], sectionPoints=
   DEFAULT, variables=('U',))

#-----------------------------------------------Create job -----------------------------------------------------------------------

mdb.models['TEM'].rootAssembly.regenerate()
mdb.Job(activateLoadBalancing=False, atTime=None, contactPrint=OFF,
   description='', echoPrint=OFF, explicitPrecision=SINGLE, historyPrint=OFF,
   model='TEM', modelPrint=OFF, multiprocessingMode=DEFAULT, name=
   'Truncated_Ellipsoid_Model', nodalOutputPrecision=SINGLE, numCpus=1, numDomains=1,
   parallelizationMethodExplicit=DOMAIN, queue=None, scratch='', type=ANALYSIS
   , userSubroutine='', waitHours=0, waitMinutes=0)
```

Wave Surface Model

```
#This is a python script that creates the FEM model that illustrates the wave surface in the
#theory capter in the master thesis. Under the section "Define variables", the user can specify
#for example the geometry or mesh density. Note that this model has a different material
#than the other models

#Written by Erik Grimsmo

#----------------------------------------------------Initialize-----------------------------------------------------------------

from part import *
from material import *
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from mesh import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from connectorBehavior import *

session.journalOptions.setValues(replayGeometry=COORDINATE,recoverGeometry=COORDINATE)

#Change work directory (use your personal directory, and uncomment the line below)
#os.chdir(r'D:\abaqusworkdirectory')

#---------------------------------------------Define variables-----------------------------------------------------------------

inner = 75 #inner radius
outer = 3300 #outer radius

rho = 1060.0 #Density

F = 100 #Amplitude of applied pressure

loadTime = 2.0 #Duration of load
t = 1.1 #Period of time integration
n = 30 #Number of elements in seed

#---------------------------------------------Create model-----------------------------------------------------------------

eps = 0.00001 #A variable that helps with modeling

mdb.Model(name='WSM')
mdb.models['WSM'].setValues(noPartsInputFile=ON)

#---------------------------------------------Create sketch-----------------------------------------------------------------

mdb.models['WSM'].ConstrainedSketch(name='__profile__', sheetSize=
   1.0)
mdb.models['WSM'].sketches['__profile__'].CircleByCenterPerimeter(
   center=(0.0, 0.0), point1=(inner, 0.0))
mdb.models['WSM'].sketches['__profile__'].CircleByCenterPerimeter(
   center=(0.0, 0.0), point1=(outer, 0.0))
```

Wave Surface Model

```
mdb.models['WSM'].Part(dimensionality=TWO_D_PLANAR, name='circularPlate',
   type=DEFORMABLE_BODY)
mdb.models['WSM'].parts['circularPlate'].BaseShell(sketch=
   mdb.models['WSM'].sketches['__profile__'])
del mdb.models['WSM'].sketches['__profile__']

#---------------------------------------------Define material---------------------------------------------------------

mdb.models['WSM'].Material(name='anisotropic')
mdb.models['WSM'].materials['anisotropic'].Elastic(table=((2460000000.0,
      2440000000.0, 10000000000.0, 2443100000.0, 2440000000.0, 2460000000.0, 0.0,
      0.0, 0.0, 8970000.0, 0.0, 0.0, 0.0, 0.0, 8450000.0, 0.0, 0.0, 0.0, 0.0,
      0.0, 8970000), ), type=ANISOTROPIC)
mdb.models['WSM'].materials['anisotropic'].Density(table=((rho, ), ))
mdb.models['WSM'].parts['circularPlate'].MaterialOrientation(
   additionalRotationField='', additionalRotationType=ROTATION_ANGLE, angle=
   0, axis=AXIS_3, fieldName='', localCsys=None, orientationType=SYSTEM,
   region=Region(
   faces=mdb.models['WSM'].parts['circularPlate'].faces.findAt(((outer-eps,
   0.0, 0.0), ), )), stackDirection=STACK_3)

#---------------------------------------------Create section---------------------------------------------------------

mdb.models['WSM'].HomogeneousSolidSection(material='anisotropic', name=
   'Section-1', thickness=None)
mdb.models['WSM'].parts['circularPlate'].SectionAssignment(offset=0.0,
   offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
   faces=mdb.models['WSM'].parts['circularPlate'].faces.getSequenceFromMask(
   mask=('[#1 ]', ), )), sectionName='Section-1', thicknessAssignment=
   FROM_SECTION)

#---------------------------------------------Partition face---------------------------------------------------------

mdb.models['WSM'].ConstrainedSketch(gridSpacing=0.001, name=
   '__profile__', sheetSize=0.056, transform=
   mdb.models['WSM'].parts['circularPlate'].MakeSketchTransform(
   sketchPlane=mdb.models['WSM'].parts['circularPlate'].faces.findAt(
   (inner, inner, 0.0), (0, 0, 1.0)), sketchPlaneSide=SIDE1,
   sketchOrientation=RIGHT, origin=(0.0, 0.0, 0.0)))
mdb.models['WSM'].parts['circularPlate'].projectReferencesOntoSketch(
   filter=COPLANAR_EDGES, sketch=
   mdb.models['WSM'].sketches['__profile__'])
mdb.models['WSM'].sketches['__profile__'].Line(point1=(inner, 0.0),
   point2=(outer, 0.0))
mdb.models['WSM'].sketches['__profile__'].Line(point1=(-inner, 0.0),
   point2=(-outer, 0.0))
mdb.models['WSM'].sketches['__profile__'].Line(point1=(0.0, inner),
   point2=(0, outer))
mdb.models['WSM'].sketches['__profile__'].Line(point1=(0.0, -inner),
   point2=(0, -outer))

l1 = inner + (outer-inner)/10 #Length used in next partitions

mdb.models['WSM'].sketches['__profile__'].ArcByCenterEnds(center=(
   0.0, 0.0), direction=COUNTERCLOCKWISE, point1=(l1, 0), point2=(0,
   l1))
```

```
mdb.models['WSM'].sketches['__profile__'].ArcByCenterEnds(center=(
   0.0, 0.0), direction=COUNTERCLOCKWISE, point1=(0, l1), point2=(-l1,
   0))
mdb.models['WSM'].sketches['__profile__'].ArcByCenterEnds(center=(
   0.0, 0.0), direction=COUNTERCLOCKWISE, point1=(-l1, 0), point2=(0,
   -l1))
mdb.models['WSM'].sketches['__profile__'].ArcByCenterEnds(center=(
   0.0, 0.0), direction=COUNTERCLOCKWISE, point1=(0.0, -l1), point2=(l1,
   0))
mdb.models['WSM'].parts['circularPlate'].PartitionFaceBySketch(
   faces=mdb.models['WSM'].parts['circularPlate'].faces.findAt(((
   inner, inner, 0.0), )), sketch=
   mdb.models['WSM'].sketches['__profile__'])
del mdb.models['WSM'].sketches['__profile__']

#--------------------------------------------Create assembly------------------------------------------------------------

mdb.models['WSM'].rootAssembly.DatumCsysByDefault(CARTESIAN)
mdb.models['WSM'].rootAssembly.Instance(dependent=ON, name='circularPlate-1',
   part=mdb.models['WSM'].parts['circularPlate'])

#----------------------------------------------Create step---------------------------------------------------------------

mdb.models['WSM'].ExplicitDynamicsStep(name='dynamic', previous=
   'Initial', timePeriod=t,  quadBulkViscosity=0.0)

#-----------------------------------------Apply BCs and load-------------------------------------------------------------

#Determine the load shape in time
mdb.models['WSM'].SmoothStepAmplitude(data=((0.0, 0.0), (1E-4-1E-5, 0.0), (
    1E-4, 1.0), (1E-4+loadTime, 1.0), (1E-4+loadTime+1E-5, 0.0)), name='impulse_smooth',
    timeSpan=STEP)

#Apply pressure
mdb.models['WSM'].Pressure(amplitude='impulse_smooth', createStepName=
   'dynamic', distributionType=UNIFORM, field='', magnitude=F, name=
   'pressure', region=Region(
   side1Edges=mdb.models['WSM'].rootAssembly.instances['circularPlate-1'].edges.findAt(
   ((inner/sqrt(2), inner/sqrt(2), 0.0), ), ((-inner/sqrt(2), inner/sqrt(2), 0.0), ),
   ((-inner/sqrt(2), -inner/sqrt(2), 0.0), ), ((inner/sqrt(2), -inner/sqrt(2), 0.0), ),)))


#Fixed outer boundary
mdb.models['WSM'].DisplacementBC(amplitude=UNSET, createStepName=
   'Initial', distributionType=UNIFORM, fieldName='', localCsys=None, name=
   'fixed1', region=Region(
   edges=mdb.models['WSM'].rootAssembly.instances['circularPlate-1'].edges.findAt(
   ((-outer/sqrt(2), outer/sqrt(2), 0.0), ), )), u1=SET, u2=SET, ur3=SET)
mdb.models['WSM'].DisplacementBC(amplitude=UNSET, createStepName=
   'Initial', distributionType=UNIFORM, fieldName='', localCsys=None, name=
   'fixed2', region=Region(
   edges=mdb.models['WSM'].rootAssembly.instances['circularPlate-1'].edges.findAt(
   ((outer/sqrt(2), outer/sqrt(2), 0.0), ), )), u1=SET, u2=SET, ur3=SET)
mdb.models['WSM'].DisplacementBC(amplitude=UNSET, createStepName=
   'Initial', distributionType=UNIFORM, fieldName='', localCsys=None, name=
   'fixed3', region=Region(
   edges=mdb.models['WSM'].rootAssembly.instances['circularPlate-1'].edges.findAt(
   ((outer/sqrt(2), -outer/sqrt(2), 0.0), ), )), u1=SET, u2=SET, ur3=SET)
mdb.models['WSM'].DisplacementBC(amplitude=UNSET, createStepName=
   'Initial', distributionType=UNIFORM, fieldName='', localCsys=None, name=
   'fixed4', region=Region(
   edges=mdb.models['WSM'].rootAssembly.instances['circularPlate-1'].edges.findAt(
   ((-outer/sqrt(2), -outer/sqrt(2), 0.0), ), )), u1=SET, u2=SET, ur3=SET)


#----------------------------------------------Generate mesh---------------------------------------------------------------

mdb.models['WSM'].parts['circularPlate'].seedEdgeByNumber(
        constraint=FINER, edges=
        mdb.models['WSM'].parts['circularPlate'].edges.findAt(
        ((inner/sqrt(2), inner/sqrt(2), 0.0), ),((-inner/sqrt(2), inner/sqrt(2), 0.0), ),
        ((-inner/sqrt(2), -inner/sqrt(2), 0.0), ),((inner/sqrt(2), -inner/sqrt(2), 0.0), ),
        ((l1/sqrt(2), l1/sqrt(2), 0.0), ),((-l1/sqrt(2), l1/sqrt(2), 0.0), ),
        ((-l1/sqrt(2), -l1/sqrt(2), 0.0), ),((l1/sqrt(2), -l1/sqrt(2), 0.0), ),
        ((outer/sqrt(2), outer/sqrt(2), 0.0), ),((-outer/sqrt(2), outer/sqrt(2), 0.0), ),
        ((-outer/sqrt(2), -outer/sqrt(2), 0.0), ),((outer/sqrt(2), -outer/sqrt(2), 0.0), ),        ), number=n)
mdb.models['WSM'].parts['circularPlate'].setMeshControls(elemShape=
   QUAD, regions=
   mdb.models['WSM'].parts['circularPlate'].faces.getByBoundingSphere(
   center=(0,0,0), radius=2*outer), technique=STRUCTURED)
mdb.models['WSM'].parts['circularPlate'].setElementType(elemTypes=(
   ElemType(elemCode=CPE4R, elemLibrary=EXPLICIT, secondOrderAccuracy=OFF,
   hourglassControl=DEFAULT, distortionControl=DEFAULT), ElemType(
   elemCode=CPE3, elemLibrary=EXPLICIT)), regions=(
   mdb.models['WSM'].parts['circularPlate'].faces.getByBoundingSphere(
   center=(0,0,0), radius=2*outer), ))
mdb.models['WSM'].parts['circularPlate'].generateMesh()

#------------------------------------------------Request outputs-------------------------------------------------------------

#Create interesting sets
mdb.models['WSM'].parts['circularPlate'].Set(name='horizontalPoint'
   , vertices=
   mdb.models['WSM'].parts['circularPlate'].vertices.findAt(((
   l1, 0.0, 0.0), )))
mdb.models['WSM'].parts['circularPlate'].Set(name='verticalPoint',
   vertices=
   mdb.models['WSM'].parts['circularPlate'].vertices.findAt(((0.0,
   l1, 0.0), )))

#Field output
mdb.models['WSM'].fieldOutputRequests['F-Output-1'].setValues(numIntervals=
   400, variables=('S', 'SVAVG', 'SENER' ))

#History output
mdb.models['WSM'].historyOutputRequests['H-Output-1'].setValues(
   frequency=10, rebar=EXCLUDE, region=
   mdb.models['WSM'].rootAssembly.instances['circularPlate-1'].sets['horizontalPoint']
   , sectionPoints=DEFAULT, variables=('S11', 'S12', 'MISES', 'UT'))
mdb.models['WSM'].HistoryOutputRequest(createStepName='dynamic',
   frequency=10, name='H-Output-2', rebar=EXCLUDE, region=
   mdb.models['WSM'].rootAssembly.instances['circularPlate-1'].sets['verticalPoint']
   , sectionPoints=DEFAULT, variables=('S11', 'S12', 'MISES', 'UT'))

#----------------------------------------------Create job -------------------------------------------------------------------

mdb.models['WSM'].rootAssembly.regenerate()
mdb.Job(activateLoadBalancing=False, atTime=None, contactPrint=OFF,
   description='', echoPrint=OFF, explicitPrecision=SINGLE, historyPrint=OFF,
   model='WSM', modelPrint=OFF, multiprocessingMode=DEFAULT, name=
   'Wave_Surface_Model', nodalOutputPrecision=SINGLE, numCpus=1, numDomains=
   1, parallelizationMethodExplicit=DOMAIN, queue=None, scratch='', type=
   ANALYSIS, userSubroutine='', waitHours=0, waitMinutes=0)
```

Bibliography

[1] J. B. Aarseth. *Numeriske beregningsmetoder.* 2011.

[2] B. Angelsen. *Ultrasound imaging. 1. Basic principles wave generation, propagation, and beamforming in homogeneous tissue.* Ultrasound Imaging: Waves, Signals, and Signal Processing. Emantec, 2000.

[3] J. C. Aster, A. K. Abbas, N. Fausto, and V. Kumar. *Robbins and Cotran, Pathalogic Basis of Disease.* 8th edition, 2009.

[4] J. Bercoff, M. Tanter, and M. Fink. Supersonic shear imaging: A new technique for soft tissue elasticity mapping. *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, 51(4):396–409, 2004. Cited By (since 1996):409.

[5] A. F. Bower. Applied mechanics of solids @ONLINE. http://www.solidmechanics.org/, 2009.

[6] B. Brekke, L. C. L. Nilsen, J. Lund, H. Torp, T. Bjastad, B. H. Amundsen, A. Stoylen, and S. A. Aase. Ultra-high frame rate tissue doppler imaging. *Submitted to Ultrasound in Medicine and Biology*, 2013.

[7] Qingshan Chen, Stacie I. Ringleb, Armando Manduca, Richard L. Ehman, and Kai-Nan An. A finite element model for analyzing shear wave propagation observed in magnetic resonance elastography. *Journal of Biomechanics*, 38(11):2198 – 2203, 2005.

[8] Y. Cho. Estimation of ultrasonic guided wave mode conversion in a plate with thickness variation. *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, 47(3):591–603, 2000.

[9] R. D. Cook, D. S. Malkus, M. E. Plesha, and R. J. Witt. *Concepts and Applications of Finite Element Analysis.* John Wiley & Sons, Inc, 4 edition, 2002.

[10] M. Couade, M. Pernot, M. Tanter, E. Messas, A. Bel, M. Ba, A. . Hagège, and M. Fink. Quantitative imaging of myocardium elasticity using supersonic shear imaging. In *Proceedings - IEEE Ultrasonics Symposium*, 2009.

[11] Dassault Systèmes. *Abaqus 6.11 Online Documentation*, 2011.

[12] E. L. Grimsmo. Fem simulations of shear waves in heart wall (pre-study project related to master thesis), 2012.

[13] B. K. Hoffmeister, S. E. Gehr, and J. G. Miller. Anisotropy of the transverse mode ultrasonic properties of fixed tendon and fixed myocardium. *Journal of the Acoustical Society of America*, 99(6):3826–3836, 1996.

[14] Gerhard A. Holzapfel and Ray W. Ogden. Constitutive modelling of passive myocardium: a structurally based framework for material characterization. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1902):3445–3475, 2009.

[15] O.S. Hopperstad and T. Boervik. *Lecture Notes, TKT4135 Mechanics of Materials.* 2012.

[16] F. Irgens. *Continuum Mechanics.* Springer-Verlag Berlin Heidelberg, 2008.

[17] H. Kanai. Propagation of spontaneously actuated pulsive vibration in human heart wall and in vivo viscoelasticity estimation. *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, 52(11):1931–1942, 2005.

[18] G. Liu and J. Qu. Transient wave propagation in a circular annulus subjected to transient excitation on its outer surface. *Journal of the Acoustical Society of America*, 104(3 I):1210–1220, 1998.

[19] Kurt J. Marfurt. Accuracy of finite-difference and finite-element modeling of the scalar and elastic wave equations. *Geophysics*, 49(5):533–549, 1984.

[20] R. Mullen and T. Belytschko. Dispersion analysis of finite element semidiscretizations of the two-dimensional wave equation. *International journal for Numerical Methods in Engineering*, 18:11–29, 1982.

[21] I.Z. Nenadic, M.W. Urban, S.A. Mitchell, and J.F. Greenleaf. Lamb wave dispersion ultrasound vibrometry (lduv) method for quantifying mechanical properties of viscoelastic solids. *Physics in Medicine and Biology*, 56(7):2245–2264, 2011.

[22] M. L. Palmeri, A. C. Sharma, R. R. Bouchard, R. W. Nightingale, and K. R. Nightingale. A finite-element method model of soft tissue response to impulsive acoustic radiation force. *IEEE transactions on ultrasonics, ferroelectrics, and frequency control*, 52(10):1699–1712, 2005.

[23] J. L. Rose. *Ultrasonic Waves in Solid Media.* Cambridge University Press., 1999.

[24] D. Royer, E. Dieulesaint, and D.P. Morgan. *Elastic Waves in Solids I: Free and Guided Propagation.* Advanced Texts in Physics. Springer, 2000.

[25] M. Tabaru, T. Azuma, and K. Hashiba. Measurement of elastic properties of tissue by shear wave propagation generated by acoustic radiation force. *Japanese Journal of Applied Physics*, 49(7 PART 2), 2010.

[26] J. H. J. M. Van Den Broek and M. H. L. M. Van Den Broek. Application of an ellipsoidal heart model in studying left ventricular contractions. *Journal of Biomechanics*, 13(6):493–503, 1980.