NTNU – Trondheim
Norwegian University of
Science and Technology

FEM simulations of an Acoustic Radiation Force Impulse applied to a Soft Tissue with a Tumor Inclusion

Lars Edvard Dæhli

Mechanical Engineering
Submission date: June 2013
Supervisor: Leif Rune Hellevik, KT

Norwegian University of Science and Technology
Department of Structural Engineering

Institutt for konstruksjonsteknikk
Fakultet for ingeniørvitenskap og teknologi
NTNU- Norges teknisk- naturvitenskapelige universitet

TILGJENGELIGHET

Åpen

MASTEROPPGAVE 2013

FAGOMRÅDE:	DATO:	ANTALL SIDER: 164
Beregningsmekanikk	10.06.2013	17+90+57

TITTEL:
FEM-simuleringer av en svulst i et bløtt vev utsatt for bølgeindusert impulslast

UTFØRT AV:

Lars Edvard Bryhni Dæhli

SAMMENDRAG:

De elastiske egenskapene til bløtt vev kan estimeres ved å indusere lokale forskyvninger og skjærbølger. Vi har laget en plan tøynings FEM-modell for å simulere en stiv elastisk svulst inne i et bløtt vev. Det bløte vevet er påført en bølgeindusert impulslast. Vi har brukt et vev-etterlignende materiale for å representere de viskoelastiske materialegenskapene til bløtt vev, og kalibrert en Maxwell-modell ut i fra eksperimentelle data. Bølgelasten som genereres av en fokusert ultralydprobe ble beregnet i en ultralydsimulering. En gaussisk funksjon ble tilpasset lastfordelingen fra simuleringen og implementert i FEM-modellen som en impulslast.

Fra FEM-simuleringene kunne vi se at den påførte impulslasten induserte lokale aksielle forskyvninger i fokalpunktet, som videre dannet en skjærbølge som forflyttet seg vekk fra fokalpunktet. Basert på den tidsavhengige forskyvningen i fokalpunktet og skjærbølgepropagasjonen gjennom det heterogene vevet, så har vi undersøkt tre forskjellige metoder for å estimere den elastiske stivheten: (i) ved å bruke skjærbølgehastigheten; (ii) ved å betrakte skjærbølgerefleksjonene fra svulsten; (iii) ved å måle tiden det tar før vi oppnår maksimal forskyvning i fokalpunktet.

Skjærbølgehastigheten ble eksakt gjengitt i det bløte vevet, men skjærbølgehastigheten i svulsten var avhengig av størrelsen og formen til svulsten, noe som resulterte i upålitelige estimater av stivheten til svulsten. Skjærbølgerefleksjonene fra svulsten var kompliserte, og refleksjonsfaktoren var avhengig av formen til svulsten. Vi må vite stivheten til det bløte vevet på forhånd, for å kunne bestemme stivheten til svulsten. Ved å måle tiden før maksimal forskyvning inntraff, så vi at denne tiden var relatert til stivheten. Vesentlige problemer ved denne metoden er at den var avhengig av varigheten på impulslasten, at den bare kan brukes for en perfekt Gaussisk ultralydbølge og at det kan være vanskelig å observere når maksimalforskyvningen inntreffer, på grunn av begrensede målefrekvenser på ultralydproben.

FAGLÆRER: Professor Leif Rune Hellevik

VEILEDER(E): Professor Leif Rune Hellevik, Førsteamanuensis Victorien Prot,
Post.doc Abigail Swillens (Universitetet i Gent)

UTFØRT VED: Institutt for konstruksjonsteknikk

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

MASTER THESIS 2013

for

Lars Edvard Bryhni Dæhli

FEM simulations of an Acoustic Radiation Force Impulse applied to a Soft Tissue with a Tumor Inclusion

FEM-simuleringer av en svulst i et bløtt vev utsatt for bølgeindusert impulslast

Soft tissues consist mainly of water and the remaining part is made up of cellular structures. The material properties of soft tissues are consequently quite similar to those of water, in terms of mass density and volume elasticity. Water does not have any resistance to shear deformations, but the cellular structure gives the soft tissue shear stiffness. Tumors have higher stiffness than the surrounding soft tissue and they can be felt by manual palpation of the tissue. However, this is not possible for tumors that are small in size or located deep within the tissue.

Due to the relatively low shear stiffness, ultrasound induced shear waves travel at a much lower speed than the ultrasound compression waves. The shear wave speed is therefore easier to measure than the compression wave speed. Modern ultrasound techniques takes advantage of this and measures the shear wave speed in the tissue to determine the elastic stiffness. In this thesis the relation between measurements and material properties should be assessed by means of FEM simulations. Suggested content for the thesis are:

- A study of relevant theory, such as ultrasound and wave propagation.
- Finite element simulations of wave propagation in a soft tissue.

The thesis should be organized according to existing guidelines.

Supervisors: Prof. Leif Rune Hellevik (NTNU), Ass.Prof. Victorien Emile Prot (NTNU), Post.Doc. Abigail Swillens (UGhent)

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

Leif Rune Hellevik
Main supervisor

Abstract

The elastic stiffness properties of soft tissues can be estimated by the use of locally induced displacements and shear waves. In this work, we have made a two-dimensional plane strain finite element model to simulate a soft tissue with a stiffer elastic inclusion. The soft tissue was subjected to an acoustic radiation force impulse. The elastic inclusion represents a potential tumor within the healthy tissue. We have used a tissue-mimicking gel-agar phantom to represent the viscoelastic material properties of soft tissue and calibrated a three-element Maxwell model based on stress relaxation data from an experiment carried out on the gel-agar phantom. The calibrated Maxwell model was verified in a finite element simulation of the stress relaxation test. The acoustic radiation force generated by a focused linear array transducer was determined from an ultrasound pressure field simulation. A three-element Gaussian function was fitted to the resulting acoustic radiation force field and implemented as a body force in the finite element model.

From the finite element analyses of the soft tissue with an inclusion, we found that the applied body force induced a local axial displacement in the focal region, which gave rise to a shear wave propagating away from the region of excitation. Based on the time dependent axial displacement profile in the focal region and the shear wave propagation through the heterogeneous tissue, we have examined three different ways of estimating the elastic stiffness: (i) using the shear wave speeds; (ii) using shear wave reflection factor values; (iii) using the time to peak displacement in the focal region.

We found that the shear wave speed was accurately (< 0.15 % deviance) represented in the soft tissue and could be used to estimate the elastic stiffness in this region. However, the shear wave speed in the tumor was dependent upon the size and shape of the tumor, which resulted in unreliable stiffness estimates. The shear wave reflections from the tumor were rather complex and the reflection factor was highly dependent upon the shape of the tumor. Also, we must know the elastic stiffness value of the healthy tissue in advance, since the shear wave reflection only provides information about the relative stiffness difference between the healthy tissue and the tumor. Thus, this method may be used to locate an inclusion, but cannot be used to quantify the stiffness of neither the surrounding tissue nor the inclusion. The time to peak displacement was inversely related to the stiffness and independent of the load magnitude, which is favorable for medical imaging application. However, the time to peak displacement was dependent upon the impulse time of the applied load and can only be directly related to the elastic stiffness for a perfectly Gaussian ultrasound beam. Also, limitations of the pulse repetition frequency can make it difficult to detect the peak displacement.

The results in this thesis indicate that stiffness estimation methods based on shear wave speed measurements are most reliable.

Preface

This work is conducted as my master thesis concluding my M.Sc degree in Mechanical Engineering at Norwegian University of Science and Technology (NTNU). The thesis was written in the period between 14^{th} of January and 10^{th} of June 2013 at the Department of Structural Engineering.

Cancer, referred to as malignant tumors, is the cause of about 13 % of all human deaths worldwide [1]. If these malignant tumors are detected at an early stage, the number of deaths may be reduced. Today, different imaging techniques have been developed in order to detect tumors within healthy tissue. In regard to this, the Finite Element Method (FEM) may be used to examine the reliability of such imaging techniques. I was intrigued by this application of FEM and decided to dedicate my master thesis to this topic.

Acknowledgments

First, I would like to thank my main supervisor Professor Leif Rune Hellevik at the Department of Structural Engineering for his valuable guidance throughout the semester. I would also like to thank co-supervisor Associate Professor Victorien Emile Prot for sharing his thoughts and knowledge. I would like to express my appreciation to co-supervisor Post-Doctoral Researcher Abigail Swillens at Ghent University for answering an enormous amount of questions and showing interest in my work. Also, I would like to thank Annette Caenan for providing experimental data.

In addition, I think it is in order to thank my peers Arne, Erik, Johan and Knut for all the fruitful discussions and the great atmosphere in the office.

Lastly, thanks to my partner in crime, Caroline, for letting me spend most of the time at the university writing this thesis.

Contents

List of Figures

List of Tables

Notation

Abbreviations

ARF	Acoustic Radiation Force
ARFI	Acoustic Radiation Force Impulse
CFL	Courant-Friedrich-Levy
FEA	Finite Element Analysis
FEM	Finite Element Method
FNM	Fast Nearfield Method
PE	Piezoelectric
PRF	Pulse Repetition Frequency
ROE	Region of Excitation
SWEI	Shear Wave Elasticity Imaging
SSI	Supersonic Shear Imaging
TTP	Time to Peak
VA	Vibro-Acoustography

Mathematical symbols

$[\,]$	Rectangular matrix
$\{\}$	Column vector
$\lvert\,\rvert$	Magnitude
$\dot{x}$	Time differentiation of variable x
$x_{i,ij}$	Partial differential of x_i w.r.t i and j
δ_{ij}	Kroenecker delta
ϵ_{ijk}	Levi-Civita tensor
∇	Differential operator
∇^2	Laplacian operator

Latin symbols

A	Cross sectional area
c	Speed of sound
c_l	Longitudinal wave speed
c_t	Shear wave speed
C_{ijkl}	Elastic tensor
d_l	Longitudinal damping coefficient
d_s	Shear damping coefficient
D_A	Aperture size
E	Young's modulus
E_0	Instantaneous Young's Modulus
f	Frequency
F	Force
$F_\#$	Focal configuration of transducer
g	Relaxation function
G	Shear modulus
G_0	Instantaneous shear modulus
I	Intensity
K	Bulk modulus
K_0	Instantaneous bulk modulus
L	Length of specimen
L_F	Focal length
P	Root-mean square pressure
R	Reflection factor or radius
R_1	Radius along x_1-axis in ellipse
R_2	Radius along x_2-axis in ellipse
t	Time
T_a	Analysis time in FEA
T_i	Impulse time of load in FEA
u_i	Displacement in direction i
U_0	Potential energy function
V	Volume
Z	Acoustic impedance
$[\mathbf{B}]$	Strain-displacement matrix
$[\mathbf{C}]$	Damping matrix or elasticity matrix
$[\mathbf{K}]$	Stiffness matrix

$[\mathbf{M}]$	Mass matrix
$[\mathbf{N}]$	Interpolation function
$\{\mathbf{D}\}$	Global nodal displacements
$\{\mathbf{R}\}^{\text{int}}$	Internal forces in element
$\{\mathbf{R}\}^{\text{ext}}$	External nodal loads on element
$\{\mathbf{W}\}$	Vector potential

Greek symbols

α	Attenuation coefficient
β	Refracted wave angle
θ	Incident wave angle
μ	Lamè coefficient
λ	Lamè coefficient or wave length
η	Viscosity constant
σ_{ij}	Stress tensor
σ'_{ij}	Deviatoric stress tensor
σ_H	Hydrostatic stress
ε_{ij}	Strain tensor
ε'_{ij}	Deviatoric strain tensor
ε_V	Volumetric strain
σ	Uniaxial stress
ε	Uniaxial strain
σ_t	Cauchy stress
ε_l	Logarithmic
ρ	Density
ρ_0	Density for viscoelastic material
ν	Poisson's ratio
ν_0	Poisson's ratio for viscoelastic material
τ	Relaxation time
ψ	Scalar potential
ω	Frequency
ξ	Damping ratio
Υ	Threshold value
ϵ	Error measure

1 Introduction

Soft tissues consist mainly of water and the remaining part is made up of cellular structures. The material properties of soft tissues are consequently quite similar to those of water, in terms of mass density and incompressibility. Water has no resistance to shear deformations, which means that it has no shear stiffness. As opposed to water, soft tissues are able to withstand some shear deformation because the cellular structures give rise to shear stiffness. A tumor has significantly higher shear stiffness than the surrounding healthy tissue [2]. Consequently, the relative stiffness difference between a healthy tissue and a tumor can be used to detect tumors within the soft tissue.

Manual palpation has been used to qualitatively determine the elasticity of soft tissues and to detect tumors within the healthy tissue for centuries and is still used in physical examinations [3]. However, this technique suffers from several limitations. Firstly, it is only possible to detect inclusions that are quite large in size, have significantly larger stiffness than the surrounding tissue and are located close to the skin surface [3]. Secondly, it is a subjective measurement and provides only qualitative elasticity data [3].

Elastography, also referred to as elasticity imaging, is a non-invasive imaging technique that can be used to assess the elastic stiffness properties of tissue [4, 5]. There are many different types of elastography procedures and the differences between them relies on whether the applied loading is static or dynamic, the loading is applied externally or internally and how the displacements in the tissue are measured [3, 5]. Static elastography is based on externally applied compressive forces and cannot provide quantitative information due to the unknown stresses in the tissue. Dynamic elastography uses externally or internally applied dynamic forces and can be used to extract quantitative information regarding tissue elasticity. In general, elastography can be performed with MRI, CT and ultrasound. However, due to the simple instrumentation and the low cost, ultrasound elastography procedures are usually the preferred choice [6].

There are several types of ultrasound elastography methods and many are beyond the scope of this thesis. However, dynamic methods such as Vibro-Acoustography (VA) [3], Shear Wave Elasticity Imaging (SWEI) [3, 7], Acoustic Radiation Force Impulse (ARFI) imaging [3, 8] and Supersonic Shear Imaging (SSI) [9, 10] are all based on the use of radiation force to induce local displacements and shear waves within the tissue. The radiation force is generated by a transfer of momentum from the ultrasound waves to the tissue, arising from reflection and absorption of the waves. The elastic stiffness of a

soft tissue can be estimated based on the local displacements and shear wave propagation within the tissue.

A malignant tumor is in general stiffer than a benign tumor [2]. Even though ultrasound elastography is widely used to detect inclusions within a healthy tissue, biopsy is the current method to determine whether a tumor is malignant or benign [11]. Since biopsy is both an expensive and a painful process, it is desirable to accurately determine the stiffness of a detected inclusion with ultrasound elastography. The accuracy of ultrasound elastography methods can be examined by the use of Finite Element Method (FEM) simulations.

The purpose of the work conducted in this thesis is to examine different stiffness estimation methods and to account for the reliability of these methods by the use of Finite Element Analysis (FEA). The work is inspired by Palmeri et al. [12] and Lee et al. [5]. Palmeri et al. used the FEM to study the dynamic mechanical response of an elastic spherical inclusion subjected to an impulsive acoustic radiation force excitation by considering the displacement magnitude, Time to Peak (TTP) displacement and recovery time. Lee et al. used FEM simulations to analyze shear wave propagation through a viscoelastic tissue-mimicking phantom with a stiffer cylindrical inclusion, and compared the resulting shear waves with experimental data.

In the present work, we have examined whether shear wave speed, shear wave reflections from the tumor boundary and TTP displacement in the focal point can be used to estimate the elastic stiffness of a soft tissue and an inclusion within the tissue. We have also discussed the reliability of these methods. The use of TTP displacement to determine the stiffness of tissue was suggested by Sarvazyan et al. [7]. In order to study these methods, we made a two-dimensional plane strain FEM model of a soft tissue with a tumor inside. We applied an Acoustic Radiation Force (ARF) impulse to induce a local axial displacement and a shear wave in the soft tissue. The ARF was determined from an ultrasound pressure field simulation of a focused linear array transducer using the ultrasound simulation tool FOCUS [13]. We used a tissue-mimicking gel-agar phantom to represent the soft tissue. The material properties of the gel-agar phantom were determined from a stress relaxation test carried out at Ghent University in collaboration with Institut Langevin in Paris [14]. They have also provided us with experimental shear wave speed data from an SSI experiment on the same gel-agar phantom.

The axial displacement and the shear wave propagation in the FEM simulations were used to calculate the numerical shear wave speed in the soft tissue and the tumor, the numerical shear wave reflection factor and the TTP displacement in the focal point. The numerical shear wave speeds and shear wave reflection factors have further been compared to analytical values for verification of the estimation method. We have also compared the numerical shear wave speed in the soft tissue with the experimental shear wave speed data.

Chapter 2 covers relevant theory of ultrasound, linear elastic and linear viscoelastic materials, waves in unbounded linear elastic and linear viscoelastic materials, plane shear wave reflection and FEM.

Chapter 3 deals with the calibration of a generalized Maxwell viscoelastic material model based on stress relaxation data of a tissue-mimicking gel-agar phantom provided by Ghent University. This resulted in a three-element Maxwell model. In order to verify the calibrated material model, we made an additional FEM model that replicated the experimental setup from the stress relaxation test.

Chapter 4 treats the simulation of the 2D ultrasound pressure field arising from a focused linear array transducer. We used a continuous wave excitation of the transducer elements and found the corresponding time independent pressure field. The pressure field was further used to determine the ARF field. The simulation was performed in FOCUS [13].

Chapter 5 deals with the FEM simulations used to examine the different stiffness estimation methods. We modeled a two-dimensional plane strain model to represent the cross-section along the ultrasound imaging axis in a soft tissue infected by a tumor. We used the three-element Maxwell model from Chapter 3 to represent the soft tissue material properties. The ARF found in Chapter 4 was applied as a body force for a short time, in order to generate an impulse load. We were able to induce a local axial displacement in the vicinity of the ultrasound focal point, which resulted in a shear wave propagating away from the Region of Excitation (ROE).

Chapter 6 contains concluding remarks about the results we obtained in the preceding chapters and highlights the most important findings in our work.

Chapter 7 points out possible improvements of the work that is carried out and gives suggestions for further work.

2 Theory

This chapter will cover relevant background theory for the work carried out in this thesis. The theory serves as an introduction to important concepts regarding ultrasound and wave propagation in soft tissues. This is essential in order to understand the models that are analyzed and discussed later on. Some background theory on the Finite Element Method is also included.

2.1 Ultrasound in Medical Imaging

The theory regarded in this chapter is adapted from [3], [7], [15] and [16]. *Ultrasonic imaging* can provide information about the mechanical properties of soft tissue and can be used to visualize potential tumors within the tissue. Ultrasonic imaging is based upon the use of an *ultrasound transducer* to generate pressure waves in the tissue, referred to as *ultrasound waves*. The ultrasonic images are made from echoes owing to reflection and scattering of the ultrasound waves as they encounter boundaries between materials with different *acoustic impedance*. The acoustic impedance is given as

$$Z = \rho c_l \tag{2.1}$$

where ρ is the density and c_l is the longitudinal velocity of the material.

In order to determine the elastic material properties, we must excite the tissue, detect the displacements and then process the information to produce the images. The excitation of the tissue can be done by either mechanical forces or radiation forces and can result in either static or dynamic tissue displacements. We will focus on the use of Acoustic Radiation Force (ARF) and dynamic tissue displacements that generate shear waves in the tissue. The displacements generated by the excitation can be detected in several ways, such as using the Doppler effect or the pulse-echo method. We will consider the pulse-echo method, only. The last task is to display the received information as images in terms of the spatial distribution of displacements, strains, stresses, shear waves or stiffness.

2.1.1 Ultrasound Wave Propagation

One cycle of an ultrasound wave contains a zone of compression followed by a zone of rarefaction, as illustrated in Figure 2.1. The compression zones of the ultrasound wave are referred to as the *wave fronts*. If the ultrasound source is large compared to the wave length of the ultrasound wave (λ), the wave fronts are represented by straight lines. These waves are referred to as *plane waves*. However, if the ultrasound wave originates from a point source, the wave fronts are represented by spheres. These waves are referred to as *spherical waves*. A plane wave and a spherical wave are shown in Figure 2.1. Both the plane wave and the spherical wave propagate in the normal direction of the wave front.

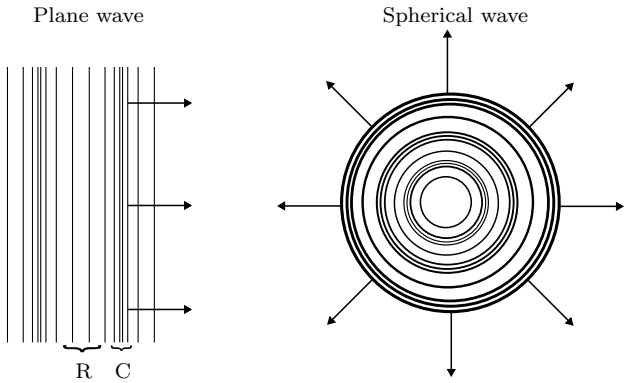


Figure 2.1: A source with large dimensions generates plane waves. A source with small dimensions generates spherical waves. C and R denotes zone of compression and zone of rarefaction, respectively.

As the ultrasound wave propagates through the tissue, it will be affected by reflections and scattering. *Reflection* is an orderly deflection of the ultrasound beam and occurs when the ultrasound beam encounters an obstacle which is much greater than the ultrasound wavelength. The ultrasound wave retains its integrity as it changes direction. In addition to the reflected part, some part of the ultrasound beam will be transmitted through the material. *Refraction* refers to the directional change of the ultrasound beam as it crosses the boundary between two materials with different sound speed. This means that the transmitted wave propagates at a different angle than the incident wave. Reflection and refraction are shown in Figure 2.2. Based on the acoustic impedance of the two materials (Z_A and Z_B), the incident angle (θ) and the refraction angle (β) we can determine the amplitude of the reflected wave and the transmitted wave as

$$R = \frac{Z_B cos\theta - Z_A cos\beta}{Z_B cos\theta + Z_A cos\beta} \quad \text{and} \quad T = \frac{2Z_B cos\theta}{Z_B cos\theta + Z_A cos\beta} \tag{2.2}$$

R and T are called the *reflection factor* and the *transmission factor*, respectively. *Scattering* is a less orderly deflection of the ultrasound beam which occurs as the size of the obstacle becomes less than or comparable to the ultrasound wavelength. The ultrasound wave will

be reflected in many different directions and loses its integrity when scattering occurs. Reflection and scattering of ultrasound waves are of great importance because they form the basis of ultrasound imaging. Both reflection and scattering are illustrated in Figure 2.2.

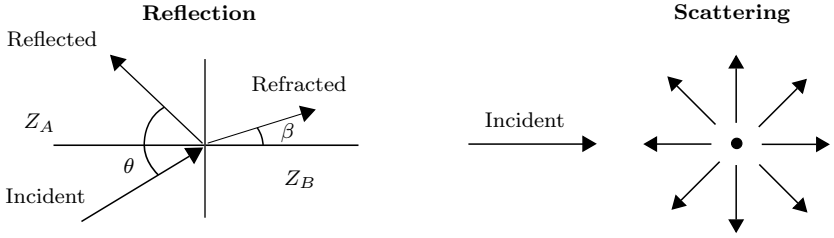


Figure 2.2: Reflection and scattering of an incident ultrasound wave.

Attenuation refers to any mechanism that removes energy from the ultrasound beam and converts it to energy of another form. The greatest contribution to attenuation is absorption, which is conversion of wave motion energy into heat. Relaxation processes are believed to be the main reason for absorption. Relaxation refers to the time it takes for a particle to return to its original position after being displaced to a new position by the ultrasound wave. Other contributions to attenuation are reflection, refraction, scattering, interference and diffraction. Because of attenuation, the amplitude of the ultrasound wave decreases as it propagates through the tissue. The attenuation coefficient α is dependent upon both frequency of the ultrasound wave and the distance the wave has travelled. This means that we can image tissue at different depths by changing the frequency of the ultrasound waves. High frequency ultrasound waves are more heavily attenuated than low frequency ultrasound waves. Hence, low frequencies must be used to image the tissue far from the skin surface, while higher frequencies can be used to image the tissue closer to the skin surface. Usually, the frequency range for medical ultrasound imaging is 2-20 MHz. A common value for the attenuation coefficient in soft tissue is $\alpha \approx 0.7 \ \frac{\text{dB}}{\text{cmMHz}}$.

2.1.2 Ultrasound Transducer

An *ultrasound transducer* is a device that converts mechanical vibrations into electrical signals or electrical signals into mechanical vibrations by the use of *piezoelectric elements*. Applying a pressure to the piezoelectric (PE) elements develops a voltage across opposite surfaces which can be converted into electrical signals. Hence, this effect can be used to detect reflected ultrasound waves in the tissue. Similarly, we can apply a voltage that causes the crystals in the PE elements to deform. This effect can be used to induce ultrasound waves in the tissue we wish to image.

Modern transducers are made up from several PE elements in an array, as illustrated in Figure 2.3 where the transducer is seen from above. The gap in between the different PE elements is called *kerf*, while the distance between the centers of neighboring elements is called *pitch*. The *axial direction* of a transducer is the imaging direction perpendicular to

the transducer surface. The direction along the transducer surface is the *lateral direction*. The axial and lateral direction spans a 2D-plane which defines the *imaging plane*. The direction perpendicular to the imaging plane is called the *elevation direction*. The arrays may consist of several elements in the elevation direction, which is referred to as either 1.5D arrays or 2D arrays. A 1D array has only one element in the elevation direction.

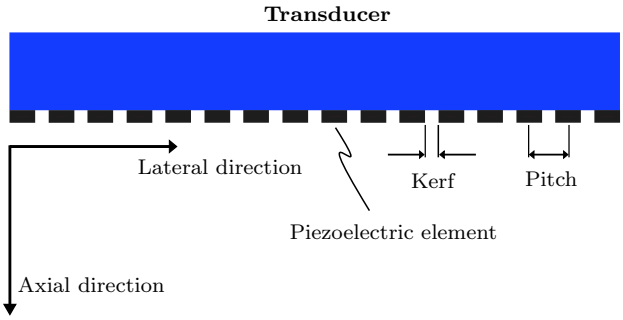


Figure 2.3: Illustration of a linear array transducer seen from above. *Kerf* is the distance between the PE elements and *pitch* is the center distance between the PE elements.

A *linear array transducer* consists of a 1D array and can excite the PE elements in groups. Each group excitation results in a *scan line* and the groups can be excited sequentially, in order to produce multiple scan lines. The time between sequential excitations is called the pulse repetition time. Hence, the rate of excitations is called Pulse Repetition Frequency (PRF) and is an important parameter in order to obtain sufficient frame rates in the images. The ultrasound images are made up from backscattered information in each scan line.

When the PE elements are excited they generate pressure waves that propagate into the tissue. The transducer is made from many small PE elements. Hence, we regard the transducer as a collection of point sources. Each PE element corresponds to a point source that radiates spherical waves into the tissue. The wave fronts from different PE elements may intersect and cause interference. With constructive interference, the wave fronts reinforce one another and the total amplitude at the region of intersection is the sum of the amplitudes contributed from each wave front. Destructive interference refers to a region where a wave front intersects a zone of rarefaction. When many PE elements are excited simultaneously, we get many regions of constructive and destructive interference in the tissue. This is illustrated in Figure 2.4. The interference is most noticeable close to the transducer and diminishes farther from the transducer. The region closest to the transducer is called the *near field* (Fresnel zone) while the region farther from the transducer is called the *far field* (Fraunhofer zone).

The ultrasound beam resulting from exciting the PE elements may have a lateral beam profile corresponding to Figure 2.5. The 6-dB response width corresponds to the lateral width where the normalized pressure ($\frac{P}{P_0}$) is equal to 0.5. This means that everywhere inside the 6-dB margin

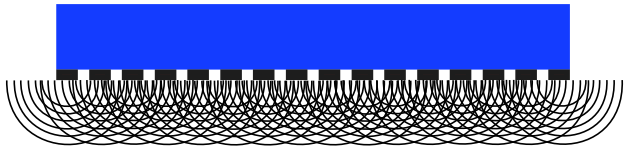


Figure 2.4: Pressure waves originating from the PE elements. Wave fronts from different elements creates constructive interference while wave fronts and zones of rarefactions creates destructive interference.

$$\frac{P}{P_0} \geq 0.5 \tag{2.3}$$

As the PE elements vibrate, they also generate transversal wave modes. These wave modes result in beams called *side lobes*, which have much less intensity than the main ultrasound beam, referred to as the *main lobe*. The side lobes are illustrated in Figure 2.5. Side lobes are not desirable in medical imaging, because we want to be certain that the received echoes arise from a target along the centerline. Echoes from a side lobe will give false information about the tissue and disturb the image. A common way to eliminate side lobes is to apply more voltage to the transducer elements in the center than to the elements on the edge, which causes the intensity to decrease gradually from the center to the edge. This is called *apodization*.

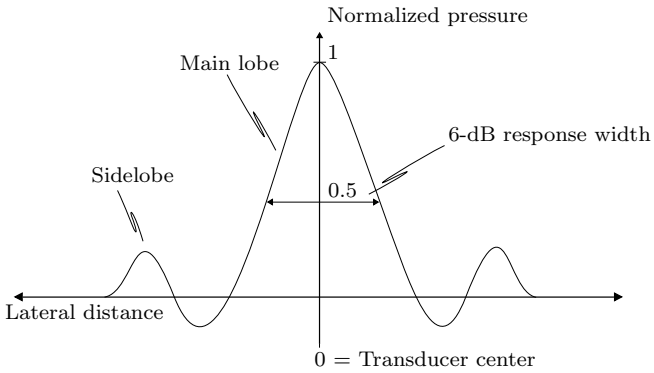


Figure 2.5: Illustration of a possible ultrasound beam profile in the lateral direction. The figure indicates the 6-dB response margin and two side lobes originating from vibrations of the transducer.

The ultrasound beam can be focused due to diffraction focusing and geometric focusing. *Diffraction focusing* is a natural focusing process that occurs because of diffraction effects in the ultrasound field. Diffraction focusing causes the ultrasound beam to converge in the axial direction, which means that the beam is narrower at some distance from the transducer than at the transducer surface. Diffraction focusing occurs regardless of other focusing processes. Figure 2.6 shows an unfocused transducer affected by diffraction focusing. The near field corresponds to the ultrasound field closest to the transducer surface

where diffraction effects are prominent, which cause an oscillating pressure amplitude. In the far field, the wave fronts from the PE elements have formed a plane wave and the pressure amplitude decreases at a fixed rate.

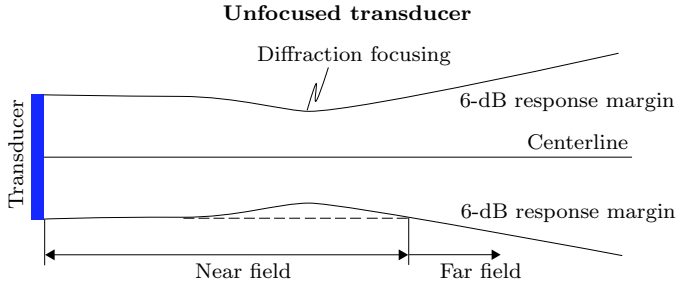


Figure 2.6: An unfocused linear array transducer experiencing diffraction focusing. The near field and far field are indicated. The figure was adapted from [16].

Diffraction focusing is not sufficient for medical imaging applications. Hence, we usually apply *geometric focusing*. A geometric focused ultrasound transducer can heighten the intensity by more than 100 times in the focal zone compared to outside the focal zone. This means that received signals from a reflecting object within the focal region are much larger and consequently easier to detect. The ultrasound beam may be focused using lenses, curved transducer elements and electronic delays of the transducer elements. Henceforth, we consider the electronic delay method. If the PE elements close to the center of the transducer are most delayed, the spherical waves will interfere and create a focal point along the centerline. The total focus of the ultrasound transducer will be a combination of diffraction and geometric focusing. A focused transducer is shown in Figure 2.7. The F-number ($F_\#$) is an important dimensionless parameter that describes the beam shape of a focused ultrasound transducer. It is defined as

$$F_\# = \frac{L_F}{D_A} \tag{2.4}$$

where L_F is the axial distance to the focal point and D_A is the size of the transducer aperture. $F_\#$ is usually on the range $1 \leq F_\# \leq 2$ for medical applications. In Figure 2.7, D_F refers to the beam width at the focal point.

2.1.3 Acoustic Radiation Force

An *Acoustic Radiation Force* (ARF) is generated based on the attenuation of the ultrasound wave. As the ultrasound wave gets absorbed or reflected by the tissue, momentum is transferred from the wave to the tissue. This momentum transfer induces a force in the direction of the wave propagation. If an amount α of the ultrasound wave is absorbed or reflected, this induces an ARF of magnitude

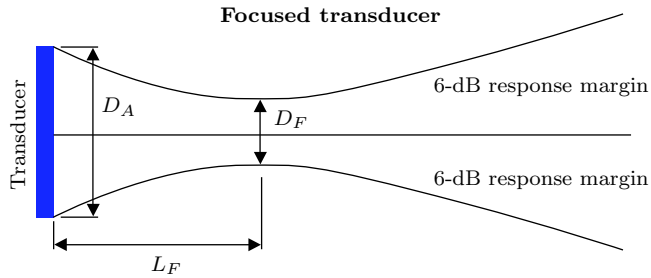


Figure 2.7: A geometric focused linear array transducer. L_F refers to the focal depth, D_A to the transducer aperture size and D_F to the ultrasound beam width at the focal point. The figure was adapted from [16].

$$|F| = \frac{2\alpha I}{c} \tag{2.5}$$

in the direction of the wave propagation. Here, $\alpha\,[\frac{\text{dB}}{\text{mMHz}}]$ is the attenuation coefficient, $I\,[\frac{\text{W}}{\text{m}^2}]$ is the temporal average intensity and $c\,[\frac{\text{m}}{\text{s}}]$ is the sound of speed in the tissue. I may be found from the pressure field as

$$I = \frac{P^2}{2\rho c} \tag{2.6}$$

where P represents the root-mean square pressure.

Based on the ARF, we are able to induce shear waves locally inside the tissue by the use of a focused ultrasound transducer. The transducer generates a short-duration ultrasound beam, referred to as a *pushing pulse*. The pushing pulse displaces the tissue as it propagates. After the pushing pulse has passed some region in the tissue, this region will restore the elastic energy and move towards its original position. The tissue is displaced more extensively inside the focal region than outside the focal region, due to the high ultrasound intensity in the focal region. This generates local displacements in the proximity of the focal region. Since the tissue outside the focal region is initially undisturbed, shear stresses develop in the region between the displaced and the undisturbed tissue. With time, the deformations and shear stresses propagate outward from the focal region as shear waves. The use of ARF to induce shear waves locally in the tissue is illustrated in Figure 2.8.

The induced shear waves can be detected using the same excitation transducer or additional ultrasound transducers. Information regarding the shear wave propagation can further be used to determine the elastic properties of tissue. For instance, the shear stiffness of the tissue can be estimated based on shear wave speed using Equation 2.66. The relations between elastic material parameters and wave speeds are discussed in detail in Chapter 2.4.1 and 2.4.2.

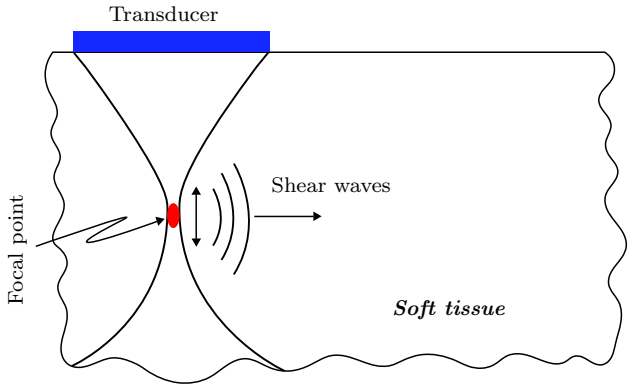


Figure 2.8: Illustration of how an ultrasound transducer can be used to generate an ARF in the soft tissue. The ARF induces shear waves that can be used to determine the stiffness properties of the soft tissue.

2.1.4 Pulse-Echo Imaging

Pulse-echo imaging is a standard method for generation of ultrasound images from backscattered information. The PE elements in the ultrasound transducer are excited, in order to produce a pushing pulse that propagates into the tissue. If the pushing pulse experiences a change in acoustic impedance, some part of the pulse will be reflected and another transmitted through the boundary between the two materials. This is illustrated in Figure 2.9 where I indicates the incident wave, R the reflected wave and T the transmitted wave. The reflected part and the transmitted part can be determined from Equation 2.2. The reflection wave may be detected by the transducer. Further, the distance from the transducer surface to the reflecting object can be calculated as

$$d = \frac{ct}{2} \tag{2.7}$$

t refers to the time from the pressure pulse was initiated until the reflection reached back to the transducer surface, while c is the wave speed of the material. When we receive information from many reflected pressure pulses, we are able to determine the shape of any inhomogeneities.

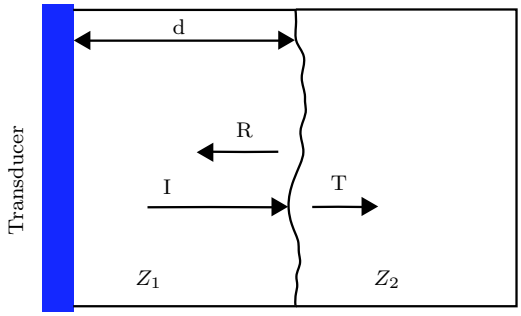


Figure 2.9: Illustration of the pulse-echo imaging method. I indicates the incident wave, R the reflected wave and T the transmitted wave. The figure was adapted from [16].

2.2 Elastic Materials

The theory presented in this chapter is adapted from [17]. If a material is reversible and path independent, it is said to be an *elastic* material. This means that a deformed body recovers to its initial configuration when the forces causing the deformation is removed and that there is a unique relation between stresses and strains in the body. Further, we can define the material as *hyperelastic* if there exists an elastic potential function such that

$$U_0 = U_0(\varepsilon_{ij}) \Rightarrow \sigma_{ij} = \frac{\partial U_0}{\partial \varepsilon_{ij}} \tag{2.8}$$

where U_0 denotes the potential energy function. Most materials are considered hyperelastic, and in the case of soft tissues we can assume that the hyperelastic material definition is appropriate.

2.2.1 Linear Elastic Materials

An elastic material is said to be *linear elastic* if the stress-strain relation is linear. A linear elastic material can be described using the generalized Hooke's law on tensor form

$$\sigma_{ij} = C_{ijkl}\varepsilon_{kl} \tag{2.9}$$

where C_{ijkl} is a fourth order tensor governing the elastic material constants and has 81 components. σ_{ij} and ε_{kl} are the second order stress and strain tensors, respectively. Due to static equilibrium of an infinitesimal material element, the stress tensor becomes symmetric. Using the Green strain tensor and assuming infinitesimal strains, we arrive at the following expression for strains in the body

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) = \frac{1}{2}(u_{i,j} + u_{j,i}) \tag{2.10}$$

where u_i (u_j) denotes the displacement in the direction of the coordinate axis x_i (x_j). From Equation 2.10 we see that the strain tensor is symmetric. Since both the stresses and strains are symmetric, the elastic tensor C_{ijkl} has 36 independent coefficients.

$$\sigma_{ij} = \sigma_{ji} \quad \Rightarrow \quad C_{ijkl} = C_{jikl} \tag{2.11}$$
$$\varepsilon_{kl} = \varepsilon_{lk} \quad \Rightarrow \quad C_{ijkl} = C_{ijlk} \tag{2.12}$$

The symmetries indicated in Equations 2.11 and 2.12 are referred to as the *minor symmetries*. By considering the stress-strain relation using the elastic potential function U_0, we find that

$$\sigma_{ij} = \frac{\partial U_0}{\partial \varepsilon_{ij}} = C_{ijkl}\varepsilon_{kl} \tag{2.13}$$

If we differentiate σ_{ij} with respect to ε_{kl}, we can write

$$C_{ijkl} = \frac{\partial^2 U_0}{\partial \varepsilon_{ij}\partial \varepsilon_{kl}} = \frac{\partial^2 U_0}{\partial \varepsilon_{kl}\partial \varepsilon_{ij}} = C_{klij} \tag{2.14}$$

This symmetry is referred to as the *major symmetry* of the elastic tensor and the number of independent elastic coefficients is now reduced to 21. If we further assume the material to be isotropic, we can describe the elastic coefficients using only two material constants. Introducing the Lamè elastic constants λ and μ, we may write the elastic coefficients as

$$C_{ijkl} = \lambda\delta_{ij}\delta_{kl} + \mu(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}) \tag{2.15}$$

Inserting this into Equation 2.9, we get

$$\sigma_{ij} = \lambda\varepsilon_{kk} + 2\mu\varepsilon_{ij} \tag{2.16}$$

The relation between the Lamè constants and the well-known Young's modulus (E) and the Poisson's ratio (ν) is given by

$$\mu = G = \frac{E}{2(1+\nu)} \tag{2.17}$$
$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)} \tag{2.18}$$

We can now write the relation between strains and stresses as

$$\varepsilon_{ij} = -\frac{\nu}{E}\sigma_{kk}\delta_{ij} + \frac{1+\nu}{E}\sigma_{ij} \tag{2.19}$$

and by inverting this expression we get the a relation between stresses and strains on the form

$$\sigma_{ij} = \frac{E\nu}{(1+\nu)(1-2\nu)}\varepsilon_{kk}\delta_{ij} + \frac{E}{1+\nu}\varepsilon_{ij} \tag{2.20}$$

It is clearly seen that the number of unknown elastic coefficients is now reduced to two, namely E and ν. The stress and strain tensor can be decomposed into deviatoric and dilatational contributions. For the stress tensor, the decomposition reads

$$\sigma_{ij} = \sigma'_{ij} + \frac{1}{3}\sigma_{kk}\delta_{ij} = \sigma'_{ij} + \sigma_H\delta_{ij} \tag{2.21}$$

whereas the strain decomposition reads

$$\varepsilon_{ij} = \varepsilon'_{ij} + \frac{1}{3}\varepsilon_{kk}\delta_{ij} = \varepsilon'_{ij} + \frac{1}{3}\varepsilon_v\delta_{ij} \tag{2.22}$$

From this we can further define relation between deviatoric stresses and strains and the dilatational stress and strain as

$$\sigma'_{ij} = 2G\varepsilon'_{ij} \quad \text{and} \quad \sigma_H = K\varepsilon_v \tag{2.23}$$

where the constants

$$G = \frac{E}{2(1+\nu)} \quad \text{and} \quad K = \frac{E}{3(1-2\nu)} \tag{2.24}$$

can be derived by inserting Equation 2.21 into Equation 2.19, or alternatively Equation 2.22 into Equation 2.20. The hydrostatic stress (σ_H) and the volumetric strain (ε_v) represents the isotropic part of the stress and strain tensor, respectively. The deviatoric stress (σ'_{ij}) and strain tensor (ε'_{ij}) represents states of stress and strain with zero dilatation, which means that the volume is preserved. This decomposition becomes important when we introduce a viscoelastic material definition because the time dependent response is assumed to be governed by the deviatoric stress.

In order for the potential energy function (U_0) to be a positive definite function of the strains, the elastic coefficients must be given by the following conditions

$$E > 0 \quad \text{and} \quad -1 < \nu < 0.5 \tag{2.25}$$

A Poisson's ratio of 0.5 corresponds to an incompressible material. Soft tissues are nearly incompressible and have Poisson's ratio values close to 0.5.

For linear isotropic elastic materials, we are able to determine the material properties (E and ν) from uniaxial material testing. The initial length of the specimen (L_0) must be

measured, along with the load (F) and the current length (L) during the test. If we further assume that the Poisson's ratio of the material is known in advance, the elastic modulus is the only parameter left to determine. If the deformations are of considerable magnitude, we must account for the change in cross-sectional area and length by relating stress and strain to the current configuration. Assuming that the material remains isotropic and linear elastic during deformation and that the total volume of the specimen is preserved, we can use the Cauchy stress measure (σ_t) and the logarithmic strain measure (ε_l) to calculate E from

$$E = \frac{\sigma_t}{\varepsilon_l} = \frac{\frac{F}{A}}{ln(\frac{L}{L_0})} = \frac{FL}{A_0 L_0 ln(1+\varepsilon)} \tag{2.26}$$

A_0 and A are the initial and current area, respectively. ε denotes the engineering strain measure, which is given by $\varepsilon = \frac{\Delta L}{L_0}$ where ΔL is the change in length during deformation. We have assumed that $AL = A_0 L_0$ in the derivation of Equation 2.26 due to volume preservation.

2.3 Viscoelastic Materials

The theory presented in this chapter is adapted from [17]. The elastic properties of soft tissues can change depending on the rate of loading and deformation, which means that soft tissues are rate dependent. Also, for a given constant load or deformation, we may experience creep or stress relaxation which means that the elastic properties of soft tissues are time dependent even when the rate of loading or deformation is zero. This rate- and time dependence is caused by internal friction, such as friction between cells in the soft tissue. In order to capture the rate- and time dependent response of the soft tissue, we need to include *viscoelastic* material properties. In general, the viscoelastic relation between stresses and strains can be nonlinear. However, we limit the discussion to *linear viscoelasticity*.

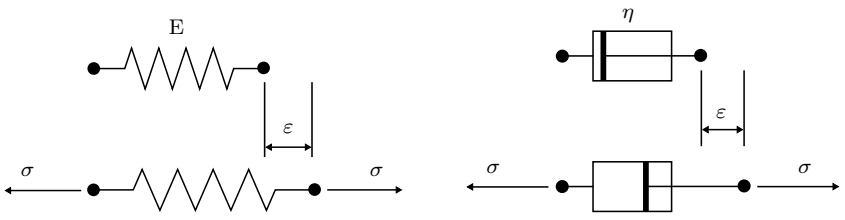
2.3.1 Linear Viscoelastic Materials

In the case of a *linear viscoelastic material*, the relation between state-of-stress and state-of-strain in the material is linear. *Rheological models* can be used to describe viscoelastic materials. The mechanical elements used in a viscoelastic rheological model are linear springs and viscous dashpots, which are shown in Figures 2.10(a) and 2.10(b).

The stress-strain relation for the two rheological elements can be written as

$$\sigma = E\varepsilon \tag{2.27}$$

$$\sigma = \eta\dot{\varepsilon} \tag{2.28}$$



(a) Linear spring element. (b) Linear viscous dashpot element.

Figure 2.10: This figure shows the rheological elements used to describe a viscoelastic material.

where Equations 2.27 and 2.28 refer to the linear elastic spring and the linear viscous dashpot, respectively. Here, E is the spring constant governing the stiffness of the spring and η is the viscosity constant governing the resistance in the dashpot. By combining the rheological elements in various ways, we are able to establish different material models.

2.3.2 The Maxwell Model

The *Maxwell model* is a combination of a linear spring in series with a linear viscous dashpot, as shown in Figure 2.11. The differential equation governing the uniaxial (one-dimensional) stress-strain relation of the Maxwell model can be written

$$\frac{\dot{\sigma}}{E} + \frac{\sigma}{\eta} = \dot{\varepsilon} \tag{2.29}$$

The strain is decomposed into elastic and inelastic terms according to Figure 2.11(a), such that the total strain and strain rate can be written

$$\varepsilon = \varepsilon^e + \varepsilon^i \Rightarrow \dot{\varepsilon} = \dot{\varepsilon}^e + \dot{\varepsilon}^i \tag{2.30}$$

The elastic and inelastic strain contributions can be expressed as

$$\dot{\varepsilon}^e = \frac{\dot{\sigma}}{E} \quad \text{and} \quad \dot{\varepsilon}^i = \frac{\sigma}{\eta} \tag{2.31}$$

since the stress (σ) is the same in both the spring and the dashpot. From this we see that the model is capable of both describing *stress relaxation* ($\dot{\varepsilon} = 0$) and *creep* ($\dot{\sigma} = 0$). The expressions for stress relaxation and creep can be found as

$$\sigma(t) = E\varepsilon_0 e^{-\frac{t}{\tau}} \tag{2.32}$$

$$\varepsilon(t) = \frac{\sigma_0}{E} + \frac{\sigma_0}{\eta} t = \frac{\sigma_0}{E}\left(1 + \frac{t}{\tau}\right) \tag{2.33}$$

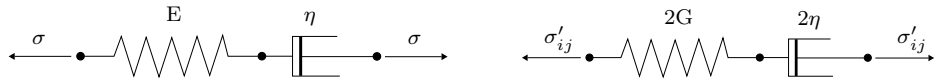
where $\tau = \frac{\eta}{E}$ is the *relaxation time*. In an elastic material, all energy is stored as strain energy during deformation. This is not the case for a viscoelastic material. While the elastic contribution is stored as strain energy, the viscous contribution related to the dashpots of the model is dissipated as heat.

Considering the viscoelastic material without limiting the theory to a uniaxial case, the rheological model can be set up according to Figure 2.11(b). We can write the strain decomposition as

$$\varepsilon_{ij} = \varepsilon^e_{ij} + \varepsilon^i_{ij} \tag{2.34}$$

and the strain rate decomposition as

$$\dot{\varepsilon_{ij}} = \dot{\varepsilon}^e_{ij} + \dot{\varepsilon}^i_{ij} \tag{2.35}$$



(a) Rheological model of the uniaxial Maxwell viscoelastic material model.

(b) Rheological model of the 3D Maxwell viscoelastic material model.

Figure 2.11: This figure describes the rheological Maxwell model in both the uniaxial case and the full three-dimensional case.

Further, we assume that the volumetric response is purely elastic such that it is only the deviatoric response that leads to dissipation. Hence, the rate and time dependent response of the material is governed by the deviatoric response and we may write the total stress as

$$\sigma_{ij}(t) = \sigma'_{ij}(t) + \sigma_H \delta_{ij} \tag{2.36}$$

From Equation 2.23 we get the following expressions for the elastic strains

$$\varepsilon'^e_{ij} = \frac{1}{2G}\sigma'_{ij} \quad \text{and} \quad \varepsilon_v = \frac{\sigma_H}{K} \tag{2.37}$$

In order to include the dissipative effects, we must account for the inelastic strain rate governed by the viscous dashpot

$$\dot{\varepsilon}^i_{ij} = \dot{\varepsilon}'^i_{ij} = \frac{1}{2\eta}\sigma'_{ij} \tag{2.38}$$

Hence, the total strain rate is given by

$$\dot{\varepsilon}_{ij} = \dot{\varepsilon}'_{ij} = \dot{\varepsilon}'^e_{ij} + \dot{\varepsilon}'^i_{ij} = \frac{1}{2G}\dot{\sigma}'_{ij} + \frac{1}{2\eta}\sigma'_{ij} \tag{2.39}$$

The stress relaxation and creep response is found in a similar manner as for the one-dimensional case. The stress relaxation is expressed as

$$\sigma'_{ij}(t) = 2G\varepsilon'^{0}_{ij} e^{-\frac{t}{\tau}} \tag{2.40}$$

while the creep strain is expressed as

$$\varepsilon'_{ij}(t) = \frac{1}{2G}\sigma'^{0}_{ij}\left(1 + \frac{t}{\tau}\right) \tag{2.41}$$

where ε'^{0}_{ij} and σ'^{0}_{ij} are the constant deviatoric strain and deviatoric stress, respectively. The constant $\tau = \frac{\eta}{G}$ is the *relaxation time*, as for the one-dimensional case, but it is governed by the shear stiffness (G) rather than Young's modulus (E).

2.3.3 Generalized Maxwell Model

A *generalized Maxwell model* is a combination of several Maxwell elements and a spring in parallel. The viscoelastic response of soft tissues is dependent upon the frequency of the loading. Hence, it is important to include Maxwell elements that can account for both the high-frequency response and the low-frequency response. A general higher-order Maxwell model is illustrated in Figure 2.12, where N denotes the number of Maxwell elements used in the model. We note that a generalized Maxwell model will be included in the Finite Element Method (FEM) model in Chapter 5. Thus, a brief discussion of the generalized Maxwell model is appropriate.

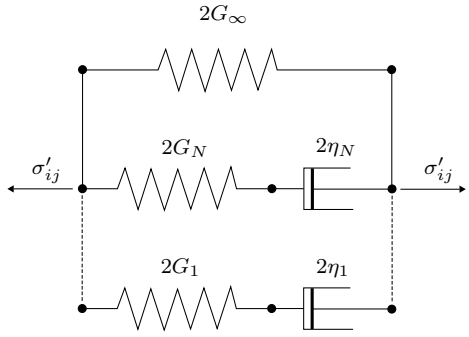


Figure 2.12: Rheological model of a generalized Maxwell material model. N is the total number of Maxwell elements used in the model.

Since the elements are connected in parallel, the strain is equal in all of the elements, and the total deviatoric stress $\sigma'_{ij}(t)$ of the model is a sum of the individual deviatoric stresses from the elements $\sigma'^{(n)}_{ij}(t)$ such that

$$\sigma'_{ij} = \sigma'^{(\infty)}_{ij} + \sum_{n=1}^{N} \sigma'^{(n)}_{ij} \tag{2.42}$$

The individual deviatoric stress terms can be written on the form

$$\sigma'^{(\infty)}_{ij} = 2G_\infty \varepsilon'_{ij} \tag{2.43}$$

$$\sigma'^{(n)}_{ij} = 2G_n \varepsilon'_{ij} = 2\eta_n \dot{\varepsilon}'_{ij} \tag{2.44}$$

As given in Lee et al. [5], this result can be expressed as a convolution integral on the form

$$\sigma'_{ij}(t) = \int_0^t 2G(t-\tau)\frac{d\varepsilon'_{ij}}{d\tau} d\tau \tag{2.45}$$

The shear modulus $G(t)$ can further be defined in terms of a prony series

$$G(t) = G_\infty + \sum_{n=1}^{N} G_n e^{-\frac{t}{\tau_n}} \tag{2.46}$$

and the integral in Equation 2.45 can be written

$$\sigma'_{ij}(t) = 2G_0 \int_0^t g(t-\tau)\frac{d\varepsilon'_{ij}}{d\tau} d\tau \tag{2.47}$$

where G_0 is the instantaneous shear modulus given as

$$G_0 = G_\infty + \sum_{n=1}^{N} G_n \tag{2.48}$$

From Figure 2.12 we may readily verify that this indeed will be the stiffness when time approaches zero, since the dashpots has not yet had the time to dissipate any energy. $g(t)$ is the *relaxation function*, which is defined as the ratio between $G(t)$ and G_0 and consequently given by the prony series expression

$$g(t) = 1 - \sum_{n=1}^{N} g_n \left(1 - e^{-\frac{t}{\tau_n}}\right) \tag{2.49}$$

where $\tau_n = \frac{\eta_n}{G_n}$ and $g_n = \frac{G_n}{G_0}$ are two constants governing the relaxation response. The prony series in Equation 2.49 is important for the numerical implementation because ABAQUS calculates the stresses based on the integral Equation 2.47.

2.4 Wave Propagation

Wave propagation is a direct result of a mediums ability to transmit localized disturbance to other parts of the medium. This phenomena can be discovered both for fluids and solids, but we limit the discussion to wave propagation in solid materials. The material is considered in a continuous manner, such that the theory of continuum mechanics is applicable and material properties are regarded as continuous functions of microscopic quantities.

In solid materials, we can experience two distinct types of wave behavior. The first is related to compressive and tensile stress, and will cause particles to move in the direction of the wave motion. These waves will affect the volume of the body and are called *volumetric* or *dilatational* waves. The second is due to the fact that the solid is able to transmit shear disturbance, which causes particles to move transverse to the wave motion. These waves will not affect the volume, hence they are volume preserving. The latter form of wave propagation is called *shear* or *distortional* waves.

2.4.1 Waves in Linear Elastic Solids

In the general case, a localized disturbance in an infinitely large body will cause deformation waves that propagate outward in all three dimensions with the velocity c. The governing differential equation for wave propagation in an unbounded, three-dimensional, isotropic, linear elastic solid can be derived from the Cauchy equation and the Hookean elasticity relations. The theory presented is adapted from [18] and [19]. The *Cauchy equation* represents the balance of momentum and can be written

$$\nabla \cdot [\sigma] + \rho\{\mathbf{b}\} = \rho\{\ddot{\mathbf{u}}\} \quad \Leftrightarrow \quad \sigma_{ij,j} + \rho b_i = \rho \ddot{u}_i \tag{2.50}$$

where b_i are the body forces. Using the stress-strain relations for Hookean materials given in Section 2.2 in Equation 2.16, the infinitesimal strain definition from Equation 2.10 and neglecting body forces we arrive at the *Navier equation*

$$(\lambda + \mu)\frac{\partial^2 u_j}{\partial x_i \partial x_j} + \mu \frac{\partial^2 u_i}{\partial {x_j}^2} = \rho \frac{\partial^2 u_i}{\partial t^2} \tag{2.51}$$

Writing this on shorthand notation, we get

$$(\lambda + \mu)u_{j,ij} + \mu u_{i,jj} = \rho \ddot{u}_i \tag{2.52}$$

Now, we introduce the differential operator, the displacement vector and the Laplacian operator

$$\nabla = \begin{Bmatrix} \frac{\partial}{\partial x_1} \\ \frac{\partial}{\partial x_2} \\ \frac{\partial}{\partial x_3} \end{Bmatrix}, \quad \{\mathbf{u}\} = \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} \quad \text{and} \quad \nabla^2 = \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} + \frac{\partial^2}{\partial x_3^2} \tag{2.53}$$

such that we can write Equation 2.51 as

$$(\lambda + \mu)\nabla\nabla \cdot \{\mathbf{u}\} + \mu\nabla^2\{\mathbf{u}\} = \rho\frac{\partial^2\{\mathbf{u}\}}{\partial t^2} \tag{2.54}$$

The Laplacian of the displacement field can alternatively be written

$$\nabla^2\{\mathbf{u}\} = \nabla\nabla \cdot \{\mathbf{u}\} - \nabla \times \nabla \times \{\mathbf{u}\} \tag{2.55}$$

and introducing this into Equation 2.54 then gives

$$(\lambda + 2\mu)\nabla\nabla \cdot \{\mathbf{u}\} - \mu\nabla \times \nabla \times \{\mathbf{u}\} = \rho\frac{\partial^2\{\mathbf{u}\}}{\partial t^2} \tag{2.56}$$

which can be rewritten in terms of a *Helmholtz* decomposition of the displacement vector

$$\{\mathbf{u}\} = \nabla\psi + \nabla \times \{\mathbf{W}\} \tag{2.57}$$

where ψ and $\{\mathbf{W}\}$ are scalar and vector potentials, respectively. Substituting this expression into Equation 2.56 and using that

$$\nabla \cdot \nabla\psi = \nabla^2\psi, \quad \nabla \times \nabla \times \nabla\psi = 0 \quad \text{and} \quad \nabla \cdot \nabla \times \{\mathbf{W}\} = 0 \tag{2.58}$$

we find that

$$\underbrace{\nabla\left[(\lambda + 2\mu)\nabla^2\psi - \rho\frac{\partial^2\psi}{\partial t^2}\right]}_{\text{Dilatation}} + \underbrace{\nabla \times \left[\mu\nabla^2\{\mathbf{W}\} - \rho\frac{\partial^2\{\mathbf{W}\}}{\partial t^2}\right]}_{\text{Distortion}} = 0 \tag{2.59}$$

In order for this relation to hold, both the dilatation term and the distortion term must equal zero, which leads to two differential equations governing wave propagation in isotropic elastic materials

$$(\lambda + 2\mu)\nabla^2\psi = \rho\frac{\partial^2\psi}{\partial t^2} \quad \Rightarrow \quad \nabla^2\psi = \frac{\rho}{\lambda + 2\mu}\frac{\partial^2\psi}{\partial t^2} \tag{2.60}$$

$$\mu\nabla^2\{\mathbf{W}\} = \rho\frac{\partial^2\{\mathbf{W}\}}{\partial t^2} \quad \Rightarrow \quad \nabla^2\{\mathbf{W}\} = \frac{\rho}{\mu}\frac{\partial^2\{\mathbf{W}\}}{\partial t^2} \tag{2.61}$$

Equations 2.60 and 2.61 are *wave equations* which can be written on the familiar form

$$\nabla^2\psi = \frac{1}{c_l^2}\frac{\partial^2\psi}{\partial t^2} \tag{2.62}$$

$$\nabla^2\{\mathbf{W}\} = \frac{1}{c_t^2}\frac{\partial^2\{\mathbf{W}\}}{\partial t^2} \tag{2.63}$$

where we recognize the expressions for the longitudinal wave velocity (c_l) and the transversal wave velocity (c_t) as

$$c_l = \sqrt{\frac{\lambda + 2\mu}{\rho}} \quad \text{and} \quad c_t = \sqrt{\frac{\mu}{\rho}} \tag{2.64}$$

Using Equations 2.17 and 2.18 we can express the two wave velocities in Equation 2.64 with the more familiar elastic constants E, G and ν

$$c_l = \sqrt{\frac{E(1-\nu)}{\rho(1+\nu)(1-2\nu)}} \tag{2.65}$$

and

$$c_t = \sqrt{\frac{G}{\rho}} = \sqrt{\frac{E}{2\rho(1+\nu)}} \tag{2.66}$$

These two expressions can be used to determine the analytical wave speed in an unbounded isotropic linear elastic material. The ratio between transversal and longitudinal wave speed can further be written

$$\frac{c_t}{c_l} = \frac{\sqrt{1-2\nu}}{\sqrt{2(1-\nu)}} \tag{2.67}$$

From this we see that for materials having Poisson's ratio in the range of $0 \leq \nu < 0.5$ the transversal wave speed is always smaller than the longitudinal wave speed. This is shown in Figure 2.13. We can also see that the ratio is more sensitive to a change in Poisson's ratio and decreases rapidly towards zero as the Poisson's ratio is in the vicinity of 0.5. This is because c_l increases rapidly towards infinity as $\nu \to 0.5$, while c_t is not greatly affected by changes in Poisson's ratio and remains fairly constant.

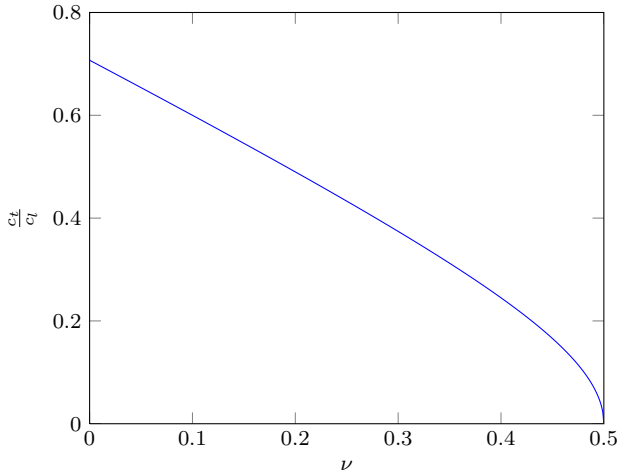


Figure 2.13: Plot illustrating the ratio $\frac{c_t}{c_l}$ for $0 \leq \nu \leq 0.5$. The ratio is very sensitive to Poisson's ratio as the material is nearly incompressible.

2.4.2 Waves in Linear Viscoelastic Materials

We will now move from the linear elastic material and over to the linear viscoelastic material. The material is assumed isotropic and infinitely large, such that the influence from material boundaries is of no importance. The theory presented in this chapter is adapted from [20] and [21].

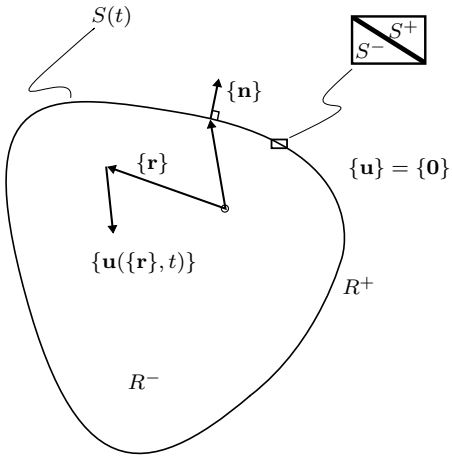
The general constitutive equation for an isotropic linear viscoelastic material may be written as a convolution integral on the form

$$\sigma_{ij}(t) = \int_0^t \left[2G(t-\tau)\dot{\varepsilon}'_{ij}(\tau) + K(t-\tau)\dot{\varepsilon}_{kk}(\tau)\delta_{ij}\right] d\tau \tag{2.68}$$

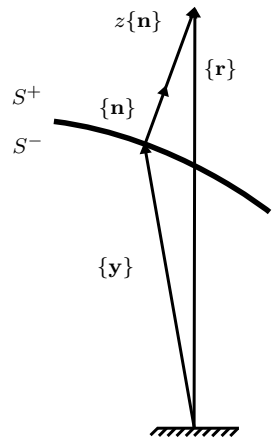
where the $K(t)$ is the time dependent bulk modulus, $G(t)$ is the time dependent shear modulus, $\dot{\varepsilon}'_{ij}$ is the deviatoric strain rate w.r.t τ and $\dot{\varepsilon}_{kk}$ is the volumetric strain rate w.r.t τ. However, as mentioned in Chapter 2.3.2, we often assume that the inelastic behavior is governed by the deviatoric response. Hence, the bulk modulus is commonly approximated as the linear elastic bulk modulus (K), which is independent of time. We note that the shear stiffness ($G(t)$) includes both elastic stiffness from springs and the dissipation governed by viscous dashpots, referring to the rheological elements described in Chapter 2.3.1.

A region defined by a closed surface S is in motion, while the medium outside of S remains undisturbed. The region inside S is denoted R^-, while the region outside S is R^+. The surface S describes the wave front and is a function of time, $S(t)$. The side of the surface

facing the region R^- is denoted S^-, and the side facing the region R^+ is denoted S^+. This is shown in Figure 2.14(a).



(a) Illustration of the wave surface S(t) representing the wave front. The regions inside (R^-) and outside (R^+) the wave front, the sides of the wave front facing the inner region (S^-) and the outer region (S^+) are shown on the figure. $\{\mathbf{n}\}$ represents a unit normal vector to the wave front.



(b) Illustration of the wave front propagation. $\{\mathbf{y}\}$ is the position vector and $\{\mathbf{n}\}$ is the normal vector. $\{\mathbf{r}\}$ represents a position in the material.

Figure 2.14: Illustration of wave propagation in a viscoelastic material. The wave surface and an arbitrary position in the material are given.

We assume small displacements and deformations and that both displacements and velocities are continuous functions. Furthermore, we assume that the displacements and velocities satisfy boundary conditions on the form

$$\{\mathbf{u}(\{\mathbf{r}\},t)\} = \{\mathbf{0}\} \quad \text{and} \quad \{\dot{\mathbf{u}}(\{\mathbf{r}\},t)\} = \{\mathbf{0}\} \quad \text{in } R^+ \text{ and on } S^- \tag{2.69}$$

The wave front propagation through the medium can be described with a Cartesian coordinate system, as shown in Figure 2.14(b). The position vector $\{\mathbf{y}(q,t)\}$ is introduced to represent an arbitrary curve on the wave front S where q is a curve parameter. The wave velocity c is the velocity of the wave front in the normal direction and is written

$$c = n_i(q,t)\frac{\partial y_i(q,t)}{\partial t} \tag{2.70}$$

Further, we let z denote a coordinate along the direction of the wave propagation. An arbitrary position in the material $\{\mathbf{r}\}$ can then be written as

$$\{\mathbf{r}\} = \{\mathbf{y}\} + z\{\mathbf{n}\} \Leftrightarrow r_i(q,t,z) = y_i(q,t) + zn_i(q,t) \tag{2.71}$$

Hence, the unit normal vector can be found by differentiating the new position vector

$$n_i = \frac{\partial r_i}{\partial z} \tag{2.72}$$

The partial derivatives of a field function $f(\{\mathbf{r}, t)$ can then calculated as

$$f_{,z} = \frac{\partial f}{\partial z} = \frac{\partial f}{\partial r_i}\frac{\partial r_i}{\partial z} = f_{,i}\, n_i \quad \text{and} \quad f_{,zz} = \frac{\partial^2 f}{\partial r_i \partial r_j}\frac{\partial r_i}{\partial z}\frac{\partial r_j}{\partial z} = f_{,ij}\, n_i n_j \tag{2.73}$$

Since the field function is a function of both $\{\mathbf{r}(q, t, z)\}$ and time itself, the time derivative of f will have two terms according to the chain rule

$$f_{,t}\,|_{q,z} = f_{,t} + f_{,i}\,\frac{\partial r_i}{\partial t} \tag{2.74}$$

If we further assume that the field function is a *level function*, which means that it is constant on the wave front for any time t, the partial derivatives of $f_{,i}$ represent a normal vector to the wave front. This is explained in Chapter 2.4 in [20]. Thus, we can write

$$f(\{\mathbf{r}\}, t) = f_0 \quad \text{on S for arbitrary } t \quad \Rightarrow \quad f_{,i} = f_{,z}\, n_i \quad \text{on S} \tag{2.75}$$

Using Equations 2.70 and 2.75 we obtain

$$f_{,i}\,\frac{\partial y_i}{\partial t} = f_{,z}\, n_i \frac{\partial y_i}{\partial t} = f_{,z}\, c = c f_{,i}\, n_i \tag{2.76}$$

We use Equations 2.74 and 2.76 to express a differential equation on the form

$$f_{,t} + c f_{,z} = 0 \quad \text{on S} \tag{2.77}$$

We can now use the results from Equation 2.75 and 2.77 and apply it to $\{\mathbf{u}\}$ and $\{\dot{\mathbf{u}}\}$. We also use the boundary conditions from Equation 2.69 to obtain the following conditions

$$\{\mathbf{u}(\{\mathbf{r}\}, t)\} = \{\mathbf{0}\} \quad \text{on } S^- \Rightarrow \{\mathbf{u}\}_{,i} = \{\mathbf{u}\}_{,z}\, n_i; \quad \dot{\{\mathbf{u}\}} + c\{\mathbf{u}\}_{,i}\, n_i = \{\mathbf{0}\} \quad \text{on } S^- \tag{2.78}$$

$$\{\dot{\mathbf{u}}(\{\mathbf{r}\}, t)\} = \{\mathbf{0}\} \quad \text{on } S^- \Rightarrow \{\dot{\mathbf{u}}\}_{,i} = \{\dot{\mathbf{u}}\}_{,z}\, n_i; \quad \ddot{\{\mathbf{u}\}} + c\dot{\{\mathbf{u}\}}_{,i}\, n_i = \{\mathbf{0}\} \quad \text{on } S^- \tag{2.79}$$

These relations imply that

$$\{\mathbf{u}\}_{,z} = \{\mathbf{0}\} \Rightarrow \{\dot{\mathbf{u}}\}_{,z} + c\{\mathbf{u}\}_{,zz} = \{\mathbf{0}\} \quad \text{on } S^- \tag{2.80}$$

which further gives the result

$$\{\ddot{\mathbf{u}}\} = c^2\{\mathbf{u}\}_{,zz} \quad \text{and} \quad \{\dot{\mathbf{u}}\}_{,i} = -c\{\mathbf{u}\}_{,zz}\, n_i \quad \text{on } S^- \tag{2.81}$$

Finally, we write the *kinematic conditions* at the wave front as

$$\{\ddot{\mathbf{u}}\} = c^2\{\mathbf{a}\} \quad \text{and} \quad \{\dot{\mathbf{u}}\}_{,i} = -c\{\mathbf{a}\}n_i \quad \text{on } S^- \tag{2.82}$$

where $\{\mathbf{a}\} = \{\mathbf{u}\}_{,zz} = \{\mathbf{u}\}_{,ij}\, n_i n_j$ is the *amplitude* of the wave front. We may now write

$$\{\mathbf{u}\}_{,ij} = \{\mathbf{a}\}n_i n_j \tag{2.83}$$

The reader is referred to Chapter 9.5.3 in [20] for a more detailed derivation of the kinematic conditions. In order to obtain expressions for the wave velocities in a linear viscoelastic material, we use conservation of linear momentum. For convenience, we write the Cauchy equation once more neglecting the body forces

$$\sigma_{ij,j} = \rho\ddot{u}_i \tag{2.84}$$

Further, we consider the constitutive equation for the linear viscoelastic material given in Equation 2.68 and take into account that the displacements are dependent upon the position $\{\mathbf{r}\}$. Hence, the strains are also dependent upon the position in addition to time $\{\varepsilon(\{\mathbf{r}\}, t)\}$. Integrating Equation 2.68 by parts gives

$$\begin{aligned}\sigma_{ij}(t) \quad = \quad & \left[2G(t-\tau)\varepsilon'_{ij}(\{\mathbf{r}\},\tau) + K(t-\tau)\varepsilon_v(\{\mathbf{r}\},\tau)\right]_0^t - \\ & \int_0^t \left[\frac{2dG(t-\tau)}{d\tau}\varepsilon'_{ij}(\{\mathbf{r}\},\tau) + \frac{dK(t-\tau)}{d\tau}\varepsilon_v(\{\mathbf{r}\},\tau)\delta_{ij}\right] d\tau\end{aligned} \tag{2.85}$$

If we further use the substitution

$$s = t - \tau \quad \Rightarrow \quad d\tau = -ds \tag{2.86}$$

we can write the derivatives of G and K as

$$\frac{dG(t-\tau)}{d\tau} = -\frac{dG(s)}{ds} \quad \text{and} \quad \frac{dK(t-\tau)}{d\tau} = -\frac{dK(s)}{ds} \tag{2.87}$$

The upper and lower integration limits change according to

$$s(0) = t \quad \text{and} \quad s(t) = 0 \tag{2.88}$$

The stress can then be written as

$$\begin{aligned}\sigma_{ij}(t) &= 2G_0\varepsilon'_{ij}(\{\mathbf{r}\},t) + K_0\varepsilon_v(\{\mathbf{r}\},t)\delta_{ij} + \\ &\int_0^t \left[2\frac{dG(s)}{ds}\varepsilon'_{ij}(\{\mathbf{r}\},t-s) + \frac{dK(s)}{ds}\varepsilon_v(\{\mathbf{r}\},t-s)\delta_{ij}\right] ds \end{aligned} \tag{2.89}$$

where G_0 is the instantaneous shear modulus and K_0 is the instantaneous bulk modulus. Since the wave front is moving, the material at position $\{\mathbf{r}\}$ will only be subjected to deformation for a time Δt. Δt represents the time from when the wave front reaches $\{\mathbf{r}\}$ to the present time, t. Hence, we can rewrite Equation 2.89 by letting $t = \Delta t$ in the upper integration limit

$$\begin{aligned}\sigma_{ij}(t) &= 2G_0\varepsilon'_{ij}(\{\mathbf{r}\},t) + K_0\varepsilon_v(\{\mathbf{r}\},t)\delta_{ij} + \\ &\int_0^{\Delta t} \left[2\frac{dG(s)}{ds}\varepsilon'_{ij}(\{\mathbf{r}\},t-s) + \frac{dK(s)}{ds}\varepsilon_v(\{\mathbf{r}\},t-s)\delta_{ij}\right] ds \end{aligned} \tag{2.90}$$

We will now include the infinitesimal strain from Equation 2.10, the kinematic conditions from Equation 2.82, Equation 2.83, the strain decomposition from Equation 2.22, let $\Delta t \to 0$ and $\{\mathbf{r}\} \to \{\mathbf{y}\}$. With reference to [20], we may find the divergence of the stress tensor as

$$\sigma_{ij,j} = G_0 u_{i,jj} + \left[K_0 + \frac{1}{3}G_0\right] u_{j,ij} = G_0 a_i + \left[K_0 + \frac{1}{3}G_0\right] a_j n_i n_j \quad \text{on } S^- \tag{2.91}$$

which further gives the Cauchy equation as

$$G_0 a_i + \left[K_0 + \frac{1}{3}G_0\right] a_j n_i n_j = \rho c^2 a_i \quad \text{on } S^- \tag{2.92}$$

The scalar product and vector product of Equation 2.92 then leads to the following expressions for the longitudinal velocity and the shear velocity

$$\left[K_0 + \frac{4}{3}G_0 - \rho c^2\right] a_i n_i = 0 \Rightarrow c_l = \sqrt{\frac{K_0 + \frac{4}{3}G_0}{\rho}} \tag{2.93}$$

$$\left[G_0 - \rho c^2\right] \epsilon_{ijk} a_i n_j \mathbf{e}_k = 0 \Rightarrow c_t = \sqrt{\frac{G_0}{\rho}} \tag{2.94}$$

where we have used that $n_i n_i = 1$ for the scalar product, ϵ_{ijk} is the permutation symbol (Levi-Civita tensor) and that $\epsilon_{ijk} n_i n_j = 0$ for the vector product. Note the similarities between the expressions for elastic wave velocities and viscoelastic wave velocities.

2.5 Plane Shear Wave Reflection

Reflection occurs in the boundary between two materials with different impedance. A tumor has different elastic properties than the surrounding soft tissue and this will cause wave reflections at the boundary between the healthy tissue and the tumor. Hence, it is necessary to understand the theory governing wave reflections. In this chapter we cover the theory of *plane shear wave reflection*.

In the subsequent equations c_t is denoted c, x_1 is denoted x and u_2 denoted u. The following is adapted from [18] where the reflection of a longitudinal wave is considered. We have used a similar approach to derive the reflection factor in the case of a shear wave.

The wave equation for the shear wave has a general solution on the form

$$u = Ae^{i(kx-\omega t)} + Be^{-i(kx+\omega t)} \tag{2.95}$$

where the first term represents a forward propagating shear wave, and the second term represents a backward propagating shear wave. The displacement function for the forward propagating incident shear wave moving towards the reflection surface is written

$$u_I = A_I e^{i(k^a x-\omega t)} \tag{2.96}$$

Here, it is used that

$$k^j = \frac{\omega}{c^j} \quad \text{for} \quad j = a, b \tag{2.97}$$

where the super-indices denote the material (a or b). As shown in Figure 2.15, the incident shear wave splits into two parts when moving into the material boundary; one that transmits through and into the material b, and one that reflects back into material a.

This results in the following equations for the reflected shear wave and the transmitted shear wave

$$u_R = A_R e^{-i(k^a x+\omega t)} \tag{2.98}$$

$$u_T = A_T e^{i(k^b x-\omega t)} \tag{2.99}$$

Superposition gives the entire displacement field as

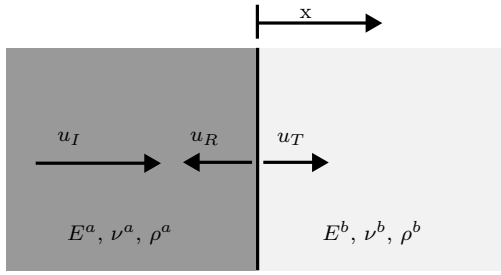


Figure 2.15: A forward propagating incident shear wave (u_I) encounters a change in material properties. As a result of the different material properties some part of the shear wave reflects (u_R) while the rest transmits into the neighboring material (u_T).

$$u^a = u_I^a + u_R^a = A_I e^{i(k^a x - \omega t)} + A_R e^{-i(k^a x + \omega t)} \tag{2.100}$$
$$u^b = u_T^b = A_T e^{i(k^b x - \omega t)} \tag{2.101}$$

Using Hooke's law, the shear stress can be written

$$\sigma_{12} = 2G\varepsilon_{12} \tag{2.102}$$

We can write the shear strain ε_{12} as

$$\varepsilon_{12} = \frac{1}{2}(\frac{\partial u_2}{\partial x_1} + \frac{\partial u_1}{\partial x_2}) = \frac{1}{2}\frac{\partial u_2}{\partial x_1} = \frac{1}{2}\frac{\partial u}{\partial x} \tag{2.103}$$

since $u_1 = 0$. The shear stress may now be written

$$\sigma_{12} = \sigma = \frac{\partial u}{\partial x} G \tag{2.104}$$

If we differentiate u in both materials, we get

$$\frac{\partial u}{\partial x}^a = iA_I k^a e^{i(k^a x - \omega t)} - iA_R k^a e^{-i(k^a x + \omega t)} \tag{2.105}$$
$$\frac{\partial u}{\partial x}^b = iA_T k^b e^{i(k^b x - \omega t)} \tag{2.106}$$

which further gives

$$\sigma^a = G^a(iA_I k^a e^{i(k^a x - \omega t)} - iA_R k^a e^{-i(k^a x + \omega t)}) \tag{2.107}$$
$$\sigma^b = G^b iA_T k^b e^{i(k^b x - \omega t)} \tag{2.108}$$

We require that the transversal displacements and the shear stresses are continuous, also at the boundary between the different materials defined as $x = 0$ in Figure 2.15. Hence, the following boundary conditions must apply

$$u^a(x=0) = u^b(x=0) \tag{2.109}$$
$$\sigma^a(x=0) = \sigma^b(x=0) \tag{2.110}$$

By substituting Equations 2.100 and 2.101 into Equation 2.109 we get

$$A_I e^{-i\omega t} + A_R e^{-i\omega t} = A_T e^{-i\omega t} \Rightarrow A_I + A_R = A_T \tag{2.111}$$

Further, by substituting Equations 2.107 and 2.108 into Equation 2.110 we get

$$G^a(iA_I k^a - iA_R k^a)e^{-i\omega t} = G^b i A_T k^b e^{-i\omega t} \Rightarrow G^a k^a (A_I - A_R) = G^b k^b A_T \tag{2.112}$$

If we combine Equations 2.111 and 2.112, we can solve for A_R and A_T

$$A_R = \frac{G^a k^a - G^a k^b}{G^a k^a + G^b k^b} A_I \tag{2.113}$$
$$A_T = \frac{2G^a k^a}{G^a k^a + G^b k^b} A_I \tag{2.114}$$

By imposing both Equations 2.66 and 2.97, this may alternatively be written

$$A_R = \frac{\sqrt{G^a \rho^a} - \sqrt{G^b \rho^b}}{\sqrt{G^a \rho^a} + \sqrt{G^b \rho^b}} A_I \tag{2.115}$$
$$A_T = \frac{2\sqrt{G^a \rho^a}}{\sqrt{G^a \rho^a} + \sqrt{G^b \rho^b}} A_I \tag{2.116}$$

The next step is to introduce A_R and A_T into the Equations 2.107 and 2.108, in order to obtain

$$\sigma_R = -iA_R k^a e^{i(k^a x - \omega t)} G^a \tag{2.117}$$
$$\sigma_I = iA_I k^a e^{i(k^a x - \omega t)} G^a \tag{2.118}$$

Finally, we may readily define the reflection factor as the ratio between the reflected stress wave and the initial stress wave at the material boundary, which gives

$$R = \frac{\sigma_R(x=0)}{\sigma_I(x=0)} = \frac{\sqrt{G^b\rho^b} - \sqrt{G^a\rho^a}}{\sqrt{G^a\rho^a} + \sqrt{G^b\rho^b}} \tag{2.119}$$

By introducing E and ν, and assuming that ρ and ν is constant we get

$$R = \frac{\sqrt{E^b} - \sqrt{E^a}}{\sqrt{E^a} + \sqrt{E^b}} \tag{2.120}$$

We note that the same result can be obtained using the expressions regarding reflected displacement and initial displacement, Equations 2.98 and 2.96 respectively.

2.6 Finite Element Analysis

The Finite Element Method (FEM) is a numerical approach to solve field problems and can be employed in calculation of structural response for both static and dynamic problems. It is extensively used in both industry and academy and is applicable for the problem at hand. We will emphasize the use for dynamic mechanical problems in this report, but some basic principles of the FEM approach is presented for clarity. The theory presented in this section is adapted from [22].

2.6.1 Overview

In the FEM, the governing mathematical model is spatially discretized by many finite elements. The displacement field ($\{\mathbf{u}\}$) in each element is described by interpolation functions ($[\mathbf{N}]$) and nodal displacement values ($\{\mathbf{d}\}$) from

$$\{\mathbf{u}\} = [\mathbf{N}]\{\mathbf{d}\} \tag{2.121}$$

In the case of small deformations, the strains can further be found from Equation 2.10 as

$$\{\varepsilon\} = [\partial]\{\mathbf{u}\} = [\partial][\mathbf{N}]\{\mathbf{d}\} = [\mathbf{B}]\{\mathbf{d}\} \tag{2.122}$$

where $[\mathbf{B}]$ is the so-called strain-displacement matrix. For a linear elastic material, we may readily calculate the stresses from Hookes law

$$\{\sigma\} = [\mathbf{C}]\{\varepsilon\} \tag{2.123}$$

where $[\mathbf{C}]$ is the elasticity matrix. The relation between external forces on the nodes $\{\mathbf{r}^{ext}\}$ and nodal displacements $\{\mathbf{d}\}$ are given by

$$[\mathbf{k}]\{\mathbf{d}\} = \{\mathbf{r}^{\mathbf{ext}}\} \tag{2.124}$$

where $[\mathbf{k}]$ is the element stiffness matrix. When establishing the element matrices in the element equation system, integration over the volume of the elements is performed. From the principle of virtual displacements, the stiffness matrix in linear elastic FEM reads out

$$[\mathbf{k}] = \int_V [\mathbf{B}]^T [\mathbf{C}][\mathbf{B}] dV \tag{2.125}$$

We refer to [22] for a derivation of the stiffness matrix expression. The numerical integration of Equation 2.125 is performed using Gauss quadrature rule. *Full integration* refers to a quadrature rule that is able to exactly integrate all stiffness coefficients k_{ij} of an undistorted element. This may lead to an overstiff response due to the assumed displacement fields. *Reduced integration* implies that a quadrature rule of less than full order is used. In addition to reduce the computational time, this may also increase the accuracy of the computed FEA results because the underintegration partly compensates for the overstiffness of the assumed displacement fields. However, reduced integration may introduce spurious modes. A remedy for this is to enable hourglass control in the FEM simulation.

2.6.2 Dynamic Equilibrium Equation

Direct integration uses step-by-step integration in time, without rewriting the equilibrium equations. Using a finite element approach, the equation of motion can be written

$$[\mathbf{M}]\{\ddot{\mathbf{D}}\}_n + [\mathbf{C}]\{\dot{\mathbf{D}}\}_n + \{\mathbf{R}^{int}\}_n = \{\mathbf{R}^{ext}\}_n \tag{2.126}$$

where $[\mathbf{M}]$ and $[\mathbf{C}]$ are the mass and damping matrices, respectively, and $\{\ddot{\mathbf{D}}\}$ and $\{\dot{\mathbf{D}}\}$ represents nodal accelerations and velocities. $\{\mathbf{R}^{int}\}$ represents the internal forces, which in the case of linear FEM is governed by the stiffness and nodal displacements and can be written

$$\{\mathbf{R}^{int}\} = [\mathbf{K}]\{\mathbf{D}\}_n \tag{2.127}$$

Equation 2.126 represents the dynamic equilibrium in step n. Discretization of the differential equation is carried out using finite difference approximations of the time derivatives.

The direct integration methods calculate conditions at time step n+1 from Equation 2.126, a difference scheme and known conditions at preceding time steps. The algorithms used are either *implicit* or *explicit*, but we will limit the discussion to an explicit method since this is preferred for wave propagation problems. The explicit method uses only historical information to calculate the response in time step $n + 1$. Hence, it is only conditionally stable and the time steps must be kept small in order to achieve stability. An explicit difference scheme is presented in the subsequent chapter.

2.6.3 Abaqus/Explicit

The ABAQUS/Explicit package [21] uses a half-step central difference method to solve Equation 2.126. The velocities and accelerations are approximated by the equations

$$\{\dot{\mathbf{D}}\}_{n-\frac{1}{2}} = \frac{1}{\Delta t}\left(\{\mathbf{D}\}_n - \{\mathbf{D}\}_{n-1}\right) \tag{2.128}$$

$$\{\dot{\mathbf{D}}\}_{n+\frac{1}{2}} = \frac{1}{\Delta t}\left(\{\mathbf{D}\}_{n+1} - \{\mathbf{D}\}_n\right) \tag{2.129}$$

$$\{\ddot{\mathbf{D}}\}_n = \frac{1}{\Delta t^2}\left(\{\mathbf{D}\}_{n+1} - 2\{\mathbf{D}\}_n + \{\mathbf{D}\}_{n-1}\right) \tag{2.130}$$

The equation of motion 2.126 is rewritten with velocity lagging half a time step

$$[\mathbf{M}]\{\ddot{\mathbf{D}}\}_n + [\mathbf{C}]\{\dot{\mathbf{D}}\}_{n-\frac{1}{2}} + \{\mathbf{R}^{int}\}_n = \{\mathbf{R}^{ext}\}_n \tag{2.131}$$

By combining the above expressions we arrive at

$$\frac{1}{\Delta t^2}[\mathbf{M}]\{\mathbf{D}\}_{n+1} = \{\mathbf{R}^{ext}\}_n - \{\mathbf{R}^{int}\}_n + \frac{1}{\Delta t^2}[\mathbf{M}]\left(\{\mathbf{D}\}_n + \Delta t\{\dot{\mathbf{D}}\}_{n-\frac{1}{2}}\right) - [\mathbf{C}]\{\dot{\mathbf{D}}\}_{n-\frac{1}{2}} \tag{2.132}$$

which may readily be solved for the nodal displacements $\{\mathbf{D}\}_{n+1}$.

The central difference method utilizes lumped mass matrices, which means that there is no need to calculate $[\mathbf{M}]^{-1}$. Hence, we only need to solve N independent equations corresponding to the N degrees of freedom in the model, which can be seen from Equation 2.132. Thus, this method is computationally efficient. However, the time increments must be kept very small in order to obtain a stable solution. This means that we need a large number of time steps in the analysis. It can be shown [22] that the stability limit is given by the expression

$$\Delta t \leq \frac{2}{\omega_{\max}} \tag{2.133}$$

where ω_{max} is the highest natural frequency of the structure. We should also take into account the effects of damping. Damping can be introduced both as material damping and numerical damping, and the latter will be present in nearly any analysis. The stability limit will be reduced as the damping is increased according to

$$\Delta t \leq \frac{2}{\omega_{\max}}(\sqrt{1+\xi_{\max}^2} - \xi_{\max}) \tag{2.134}$$

where ξ_{max} denotes the damping ratio of the highest frequency mode. In the explicit FEA we can approximate this stability limit, and introduce a *critical time step* which defines the upper bound for applicable time step values

$$\Delta t_{\text{cr}} \approx \frac{L_e^{\min}}{c_l} \tag{2.135}$$

L_e^{min} is the size of the smallest element in the model and c_l is the longitudinal (dilatational) wave speed, which is given by Equation 2.65. The interpretation of this approximation is that information cannot be able to travel across more than one element during a single time step. This must be true, since the deformation cannot occur at a greater speed than the speed of the wave which carries the information regarding the deformation. A way to measure the time step is the *Courant number*. This is defined as the ratio between the applied time step and the critical time step

$$C_n = \frac{\Delta t}{\Delta t_{\text{cr}}} = c_l \frac{\Delta t}{L_e^{\min}} \tag{2.136}$$

For the half-step central difference scheme, we must ensure that the Courant number is smaller than unity. Otherwise, the computations become unstable and eventually fail. This is also referred to as the *CFL-condition*. However, C_n should also be kept close to unity, in order to obtain accuracy.

ABAQUS/Explicit offers two different ways of choosing the time step. We can use *fixed time incrementation* or *automatic time incrementation*. The fixed time incrementation method uses time increments of a predefined value during the entire step. This requires the user to have knowledge about the critical time step, and how this is related to both the element size and the size of the chosen time increment. The time increment will not change during the time step. The automatic time incrementation requires no intervention from the user and is in most cases the preferred method. It can be based on either element-by-element estimation or global estimation of the time step. The element-by-element estimation calculates the critical time step based on current the dilatational wave speed (c_l) in each element. The global estimation on the other hand, estimates the highest natural frequency ($\omega_{\max}$) for the entire model based on the current dilatational wave speed. The global time increment estimator is the default method in Abaqus. The time increments may change during a time step, which makes it easier to keep C_n close to unity for all time increments.

2.6.4 Infinite Elements

The theory is adapted from [21]. *Infinite elements* can be used in problems where the region of interest is small compared to the surrounding medium. If we use finite elements and model only the small region that we are interested in, we will get wave reflections from the boundaries of the model. One way to handle this is to extend the mesh far away from the region of interest, in order to reduce the influence of the surrounding medium.

However, this can be very inefficient since we would need a great number of elements. A better approach is to use infinite elements to damp out reflections.

We only consider waves along the x_1-axis, but the approach is similar for waves along the x_2- and x_3-axis. At the boundary of the finite element domain we introduce damping on the form

$$\sigma_{11} = -d_l \dot{u}_1 \tag{2.137}$$
$$\sigma_{12} = -d_t \dot{u}_2 \tag{2.138}$$
$$\sigma_{13} = -d_t \dot{u}_3 \tag{2.139}$$

The damping coefficients are chosen in order to avoid reflection of longitudinal and shear wave energy. We assume that the incident and reflected waves are plane, such that

$$u_1^{\mathrm{I}} = f_1(x_1 - c_l t) \quad \text{and} \quad u_1^{\mathrm{R}} = f_2(x_1 + c_l t) \tag{2.140}$$

If we use superposition of the displacements, we get that $u_1 = f_1 + f_2$. We may further write the longitudinal stress as

$$\sigma_{11} = (\lambda + 2\mu)\frac{\partial u_1}{\partial x_1} = (\lambda + 2\mu)(f_1' + f_2') \tag{2.141}$$

and the velocity as

$$\dot{u}_1 = -c_l(f_1' - f_2') \tag{2.142}$$

Now, inserting Equations 2.141 and 2.142 into Equation 2.137 we get

$$(\lambda + 2\mu - d_l c_l)f_1' + (\lambda + 2\mu + d_l c_l)f_2' = 0 \tag{2.143}$$

In order to damp out reflections, we must have $f_2 = 0$ which means that $f_2' = 0$. If Equation 2.143 is satisfied, then

$$d_l = \frac{\lambda + 2\mu}{c_l} = \rho c_l \tag{2.144}$$

We may find the damping coefficient d_t from a similar approach with the shear stresses

$$d_t = \frac{\mu}{c_t} = \rho c_t \tag{2.145}$$

We note that these damping coefficients will damp out all reflections as long as the material close to the boundary is linear elastic and the incident waves are plane and normal to the boundary. However, they work quite well also with non-plane waves.

3 Calibration of the Viscoelastic Material Model

We wanted to use a gelatin-based tissue-mimicking material to represent the material properties of a soft tissue. In order to describe the material behavior, we must fit a mathematical relation to given experimental data. This is referred to as calibration of a material model.

3.1 Method

We calibrated a generalized Maxwell viscoelastic material model, presented in Chapter 2.3.3, based on experimental data from a stress relaxation test on a gel-agar phantom. The experiment was carried out at Ghent University in collaboration with Institut Langevin in Paris [14]. In a stress relaxation test, the specimen is extended a certain amount after which the displacements are kept constant. Thus, the strain rate is zero ($\dot{\varepsilon} = 0$) and the stress decreases due to energy dissipation. The test specimen was a gel-agar phantom loaded in uniaxial tension. The initial geometry of the test specimen, the fastened end and the loading direction are illustrated in Figure 3.1. From this figure we see that the specimen was initially 3.5 mm in *thickness*, 57.4 mm in *width* and 63.97 mm in *length*. The longitudinal extension of the specimen and the force applied on the specimen was measured at different times during the experiment. The force is plotted in Figure 3.2(a) while the deformation is plotted in Figure 3.2(b). From the experimental data, we calculated the uniaxial strain and uniaxial stress, which was further used to calculate the elastic modulus at each time instance. We assumed a linear relation between stress and strain and referred the deformations to the current configuration, such that the Cauchy stress and logarithmic strain measured were used.

The relaxation coefficients (g_n and τ_n) from Equation 2.49 must be estimated, in order to implement the viscoelastic material model in Abaqus. There are several possible ways to determine these parameters, for instance by the use of a least-square estimation method in Matlab. However, Abaqus has a built-in least-square estimation routine to determine the relaxation coefficients based on stress relaxation or creep test data. We relied on the

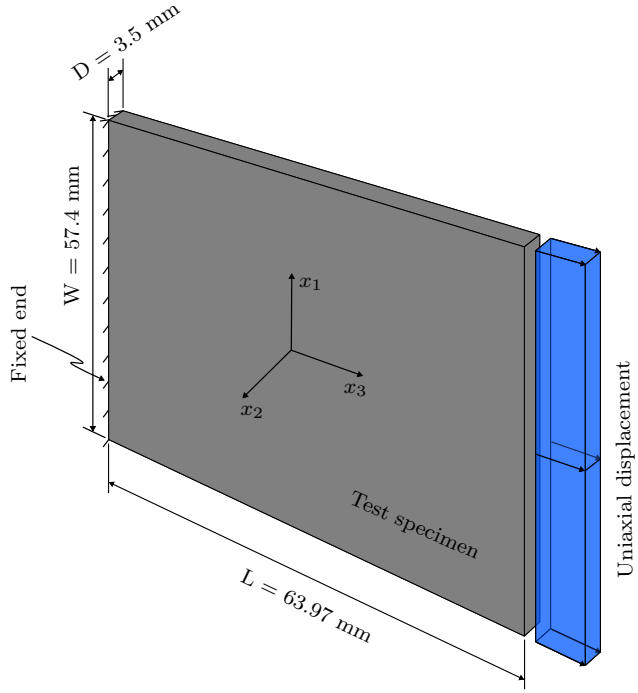
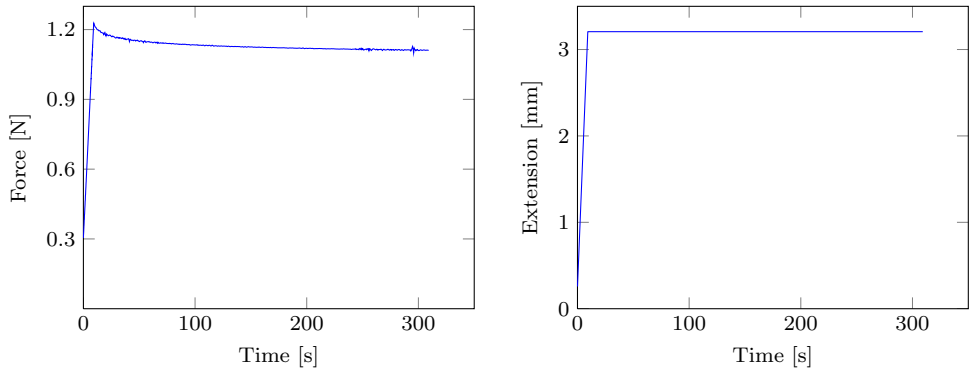


Figure 3.1: The test specimen used in the uniaxial relaxation test. The figure indicates the geometry of the specimen, the boundary conditions and the loading direction. D is the thickness, W is the width and L is the length of the specimen.



(a) Force data from experiment.

(b) Extension data from experiment.

Figure 3.2: The experimental data from the uniaxial tension test of a gel-agar phantom. The experiment was carried out at Ghent University.

estimation method provided in ABAQUS for the calibration of the generalized Maxwell material model.

In order to use the built-in estimation method in ABAQUS, we must first define an elastic material and provide experimental data. ABAQUS uses the elastic material as a reference material to describe either the instantaneous response (G_0) or the long-term response (G_∞), based on what the user chooses. The time dependent material response is governed by the relaxation coefficients (g_n and τ_n) estimated from the material test data.

Elastic Material Parameters

We defined the elastic material parameters in terms of the instantaneous elastic modulus (E_0) and the corresponding Poisson's ratio (ν_0) and density (ρ_0). We further assumed that the material is nearly incompressible, such that $V(t) \approx V_0$. From Figures 3.2(a) and 3.2(b), we see that the specimen was slightly deformed when the first measurement was made, since the first force and displacement values are different from zero. Thus, E_0 cannot be estimated directly from the stress and strain values at the onset of stress relaxation ($t = 9.739$ s). In order to account for the pre-tension, we determined E_0 from Equation 2.26 by using the difference between the measurement at onset of relaxation and the first measurement

$$E_0 = E(t = 9.739) = \frac{\sigma_t|_{t=9.739} - \sigma_t|_{t=0}}{\varepsilon_l|_{t=9.739} - \varepsilon_l|_{t=0}} = \frac{1}{A_0 L_0} \left(\frac{(FL)|_{t=9.739} - (FL)|_{t=0}}{ln(\frac{L}{L_0})|_{t=9.739} - ln(\frac{L}{L_0})|_{t=0}} \right) \quad (3.1)$$

Then we calculated the instantaneous shear modulus (G_0) based on the relation between E and G given in Equation 2.24

$$G_0 = \frac{E_0}{2(1 + \nu_0)} \quad (3.2)$$

We note that E_0 was determined from the current configuration using Cauchy stress and logarithmic strain. The reason for this is that the gel-agar phantom is quite flexible and we believe that the assumption of small deformations is not valid. In ABAQUS it is possible to include non-linear geometry in the analysis (NLGEOM=ON), in order to account for large deformations.

Viscoelastic Material Parameters

We defined the inelastic part of the viscoelastic material using the ratio between current shear modulus and the instantaneous shear modulus ($\frac{G(t)}{G_0}$). We assumed that the inelastic response is governed by deviatoric stresses. Hence, we only accounted for the shear modulus while the bulk modulus was disregarded. The Poisson's ratio was assumed constant ($\nu(t) = \nu_0$), which means that the ratio between shear moduli is equal to the ratio between the elastic moduli

$$g(t) = \frac{G(t)}{G_0} = \frac{E(t)}{E_0} \tag{3.3}$$

where $g(t)$ is referred to as the relaxation function. The time dependent elastic modulus in the relaxation phase was found from the experimental data using Equation 2.26, accounting for the pre-tension of the specimen

$$E(t_i) = \frac{\sigma_t(t_i)|_{t_i>9.739} - \sigma_t(t=0)}{\varepsilon_l^0 - \varepsilon_l(t=0)} \tag{3.4}$$

ε_l^0 denotes the constant logarithmic strain in the relaxation phase ($t_i > 9.739$) s. Afterwards, these values were used to calculate $g(t_i)$ for the discrete times (t_i) in the experimental data using Equation 3.3. The values of $g(t_i)$ and t_i were further implemented in Abaqus and the relaxation coefficients (g_n and τ_n) were determined from a least-square estimation method by minimizing the difference between the prony series in Equation 2.49 and the relaxation values $g(t_i)$ from the experimental data.

Also, we defined a tolerance level (LStol in Appendix B.1) and how many relaxation elements to include for the estimation. One relaxation element corresponds to one Maxwell element, shown in Figure 2.12. The default number of relaxation elements is 13, which also is the maximum number. Abaqus first tries to fit the model to the stress relaxation data with only one element (g_1 and τ_1). If the given tolerance level is not met, an additional element is included in the next fit. This is repeated until the response is within the tolerance level or the maximum number of elements is reached.

Verification of the Material Parameters

The elastic material constants and the relaxation coefficients we found from the least-square estimation were further implemented in an FEM model for verification. The model contained the same geometry, boundary conditions and loading as in the experiment. This is shown in Figure 3.1. The approximate global element size was set to 1.1 mm and the total number of elements in the model was 12064. We used 20-node, quadratic brick elements with reduced integration (C3D20R). The meshed model is shown in Figure 3.3. We included non-linear geometry in the FEM model, such that the deformations were referred to the current configuration. In order to account for the pre-tension of the specimen, we assigned a prescribed stress corresponding to the initial stress $\sigma_t(t=0)$ from the experimental data. Further, we applied a time dependent analysis step, named *visco* in Abaqus, for both the loading and the stress relaxation phase. A visco step is essentially a quasi-static analysis procedure that takes the time dependent material response into account, and is suited to solve creep and stress relaxation problems [21]. We extracted the uniaxial stress and the total reaction force from the FEA results and compared with the Cauchy stress and the measured force data from the stress relaxation experiment. The FEM model can be imported in Abaqus from the script in Appendix B.1.

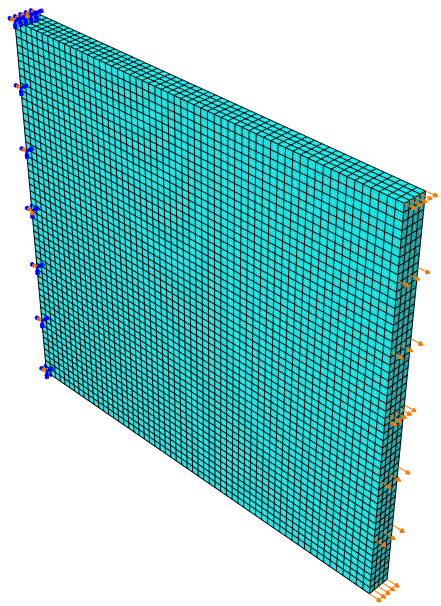


Figure 3.3: The FEM model used to simulate the experiment on the gel-agar phantom. The geometry, boundary conditions and loading is the same as for the experiment, given in Figure 3.1. The total number of elements in the model was 12064 with an approximate element size of 1.1 mm.

3.2 Results

We used the MATLAB script given in Appendix A.1 to extract the experimental displacement and force data from a spreadsheet, calculate the Cauchy stress, logarithmic strain, elastic modulus and relaxation values, and make comparisons with FEA results from ABAQUS.

Elastic Material Parameters

We calculated the logarithmic strain (ε_l) and the Cauchy stress (σ_t) from the stress relaxation data. These values were used to estimate E_0 and G_0 using Equations 3.1 and 3.2, respectively. Further, we assumed values for ν_0 and ρ_0 that are representative to the properties of soft tissues. The resulting elastic material parameters are given in Table 3.1.

Viscoelastic Material Parameters

We performed a least-square estimation of the relaxation coefficient (g_n and τ_n) in ABAQUS, based on the experimental relaxation data ($g(t_i)$) and the prony series in Equation 2.49. We included three relaxation elements and set the tolerance level (LStol in B.1) for

Table 3.1: The elastic material parameters for the gel-agar phantom. The values of ν_0 and ρ_0 are representative to the properties of soft tissues. E_0 was estimated from experimental data.

Material parameter	Value	Unit
E_0	107585	Pa
G_0	35886	Pa
ν_0	0.499	-
ρ_0	1060	$\frac{\text{kg}}{\text{m}^3}$

the least-square fit to 0.0019, which was the lowest possible tolerance level for a three-element Maxwell model with the given experimental relaxation values. This resulted in the relaxation coefficients given in Table 3.2. The relaxation function defined by the three-element prony series is plotted against the experimental relaxation values in Figure 3.4.

Table 3.2: Relaxation coefficients from the least-square estimation of the prony series.

n	g_n	τ_n [s]
1	0.00094575	1.5178
2	0.00384291	11.541
3	0.00646134	93.977

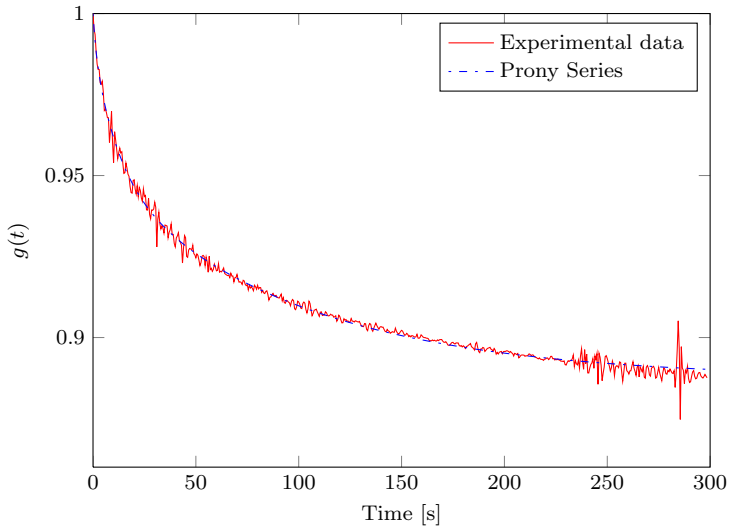
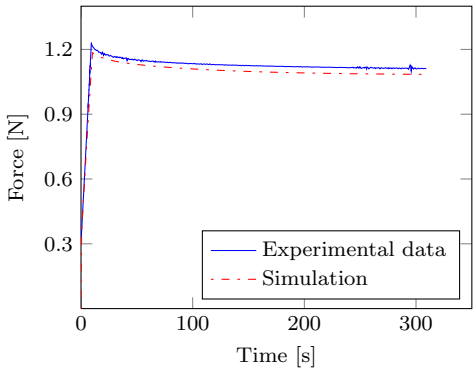


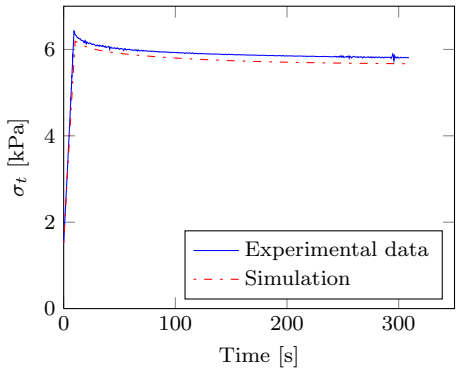
Figure 3.4: The three-element prony series is plotted against the relaxation values extracted from the experimental data.

Verification of the Material Parameters

We implemented the elastic material parameters and the relaxation coefficients from Table 3.2 in the FEM model presented in Chapter 3.1. From the FEA results, we extracted the total reaction force in the fastened end and the uniaxial stress in the model. These values were plotted against the measured force data and the uniaxial stress in the experiment. This is shown in Figures 3.5(a) and 3.5(b), respectively. Figure 3.6 shows the deviance between the uniaxial stress from the experimental data and the FEM simulation. We have expressed the deviance as the natural logarithm of the percentage error between the FEA results (σ^{sim}) and the experimental data (σ^{exp}).



(a) This graph shows the force values from the FEA for the calibrated viscoelastic material model plotted against the values extracted from the experimental data.



(b) This graph shows the uniaxial stress values from the FEA for the calibrated viscoelastic material model plotted against the values extracted from the experimental data.

Figure 3.5: The force and uniaxial stress from the FEM simulation are compared to the experimental data from the gel-agar phantom.

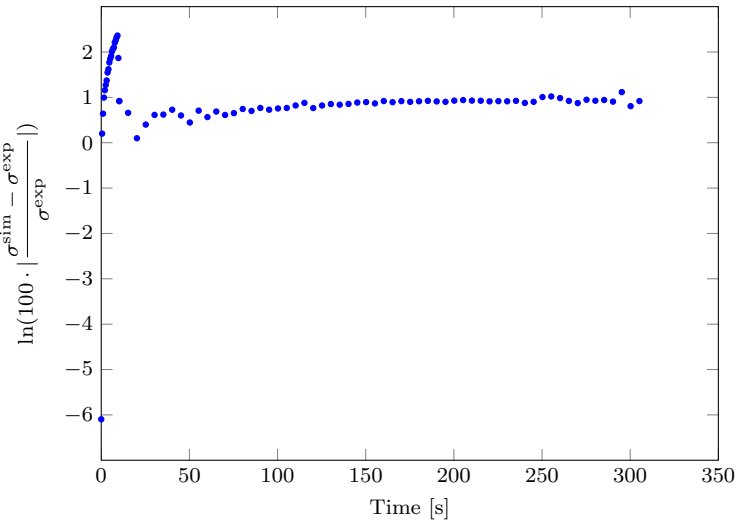


Figure 3.6: The graph shows the natural logarithm of the relative error between the numerical stress values extracted from FEA results and the stress values from the experimental data.

3.3 Discussion

In Chapter 3.2, we estimated the elastic and inelastic material parameters from experimental stress relaxation data. In order to determine E_0 from the experiment, we assumed a constant stress-strain ratio during loading, as indicated in Equation 3.1. From Table 3.2 we see that the smallest estimated relaxation time was $\tau_1 = 1.5178$ s. Since the duration of the loading phase was approximately 10 s, the stress in the first relaxation element ($\sigma^{(1)}$) would already have been reduced to about 0.0014 of its initial stress at the onset of the relaxation phase. From Equation 2.32, we may readily verify this

$$\sigma^{(1)}(t = 10) = \sigma_0^{(1)} e^{-\frac{10}{1.5178}} \approx 1.4 \cdot 10^{-3} \sigma_0^{(1)} \tag{3.5}$$

Also, the stress in the second relaxation element would have been reduced to less than half of its initial value due to a relaxation time of $\tau_2 = 11.541$ s, according to

$$\sigma^{(2)}(t = 10) = \sigma_0^{(2)} e^{-\frac{10}{11.541}} \approx 0.42 \sigma_0^{(2)} \tag{3.6}$$

Hence, the displacement and force measurements in the experiment were already affected by the time dependent material behavior and the assumption of constant stress-strain ratio may not be entirely correct. However, if we take this time dependency into account, we would have to separate the elastic and inelastic parts of the response during loading, in order to determine the proper value of E_0 and the relaxation values $g(t_i)$ from the experimental data. This is rather difficult and would require a tedious trial-and-error approach, since the elastic and inelastic parts of the response are not known in advance.

Consequently, we chose to use the assumption of constant stress-strain ratio and we considered it as sufficient for the purpose of the material calibration in this thesis.

The reaction force and uniaxial stress from the experimental data and the FEA results were compared in Figures 3.5(a) and 3.5(b). We see that the FEA results are comparable to the experimental data. This is also verified by Figure 3.6, which indicates that the discrepancy in uniaxial stress is on the range of

$$-6.1 < \ln \epsilon < 2.3 \quad \Rightarrow \quad 0.00224\% < \epsilon < 9.975\% \tag{3.7}$$

ϵ represents the absolute error between the FEA results and the experimental data and is given by

$$\epsilon = 100\,\% \cdot \left| \frac{\sigma^{\text{sim}} - \sigma^{\text{exp}}}{\sigma^{\text{exp}}} \right| \tag{3.8}$$

We observe that the smallest deviance was obtained immediately after the onset of loading ($\dot{\varepsilon} > 0$). The reason for this is that we applied the measured initial uniaxial stress value from the experiment ($\sigma_t(t = 0)$) as a pre-stress in the FEM simulation. Consequently, the first few stress values from the FEA results correspond to the experimental values. However, the deviance increases quite rapidly during the loading phase and reaches a maximum of $\epsilon_{\max} \approx 10\%$ at the onset of the relaxation phase ($\dot{\varepsilon} = 0$). We believe that this stems from underestimated displacement values in the analysis. In the experiment, the initial uniaxial displacement (u_0^{exp}) and force were different from zero, which gave rise to initial stress and strain. However, prescribing a stress in the FEM simulation does not introduce an initial displacement, such that $u_0^{\text{sim}} = 0$. Consequently, the displacement in the FEA was smaller by an amount $\Delta u = u_0^{\text{exp}}$ throughout the analysis, which in turn caused underestimated strain values. Thus, from Equation 2.123 we readily see that this causes underestimated stress values. We note that we could have prescribed the displacement rather than stress in order to obtain the correct displacement history in the FEM simulation. However, when such a simulation was carried out, we found that the uniaxial stress values were even lower. We are not aware of the reasons for this, but it might be related to an underestimated value of the instantaneous elastic modulus (E_0) from the experimental data. Also, it might be possible that the displacement and force measurements in the beginning of the experiment are somewhat inaccurate, since the initial displacement and force magnitudes are relatively small and difficult to measure. This would give initial conditions that do not reflect the true material behavior and eventually cause erroneous FEA results. Due to this, we chose to use a prescribed stress, since this gave a closer replica of the experiment.

Referring to Figure 3.5, we see that the force and uniaxial stress in the FEA are slightly underestimated also in relaxation phase ($t > 9.739$ s). We believe that the reason for this discrepancy is mostly due to the initial error from the loading phase. Since the stress is underestimated during loading, we never reach a stress level as high as in the experiment. In turn, this causes an initial offset of the FEA results in the relaxation phase, which can be seen in both Figure 3.5(a) and 3.5(b). We also observe that the deviance remains fairly

constant in the relaxation phase and that the average logarithmic error is approximately $\ln \epsilon \approx 0.85$, which corresponds to an average absolute error of $\epsilon \approx 2.35\%$. This further implies that the initial offset is the main source of error in the relaxation part. This observation also agrees with the results from Figure 3.4, where we see that the prony series fits perfectly to the experimental relaxation data. Hence, the relaxation behavior observed in the experiment is accurately described by the three-element relaxation function with the coefficients from Table 3.2.

Even though the relaxation behavior from the experiment seems to be well captured by the three-element Maxwell model, we have relied on the relaxation data from a single experiment in the calibration. If this single experiment is biased, the estimated material parameters do not reflect the real material behavior. Thus, we should probably have included test data from several relaxation experiments in the estimation procedure, in order to increase the reliability of the calibration. We note that data from creep experiments could also have been used, in order to obtain more reliable values of the material parameters.

4 ARF Simulation

As explained in Chapter 2.1.3, the Acoustic Radiation Force (ARF) generated by a focus ultrasound transducer can be used to induce local displacements and shear waves within the soft tissue. We wanted to simulate the pressure field from an ultrasound transducer and use this to determine the ARF field. A Supersonic Shear Imaging (SSI) experiment on a gel-agar phantom had already been carried out at Ghent University, in collaboration with Institut Langevin in Paris [14]. We used the transducer setup from this experiment as a basis for the ultrasound transducer in the simulation.

4.1 Method

We made a model of a linear array ultrasound transducer in MATLAB using the FOCUS toolbox [13]. FOCUS is an ultrasound simulation tool that can be used to model various kinds of ultrasound transducers and to calculate the resulting pressure field for a given material. We modeled the transducer with 40 transducer elements. The elements were 170 μm in width and the kerf was 30 μm. Thus, the transducer width was approximately 8 mm. We used a focal depth of 10 mm in the axial direction, which gives $F_{\#} \approx 1.25$ using Equation 2.4. The material properties in the simulation were set to match those of soft tissues and are given in Table 4.1. The center frequency (f_0) of the transducer was 8 MHz, which corresponds to the center frequency of the transducer used in the SSI experiment on the gel-agar phantom. We used apodization in the analysis by using a sine function to define when the piezoelectric (PE) elements were excited. Thus, the PE elements on the transducer edges were excited first, and the PE elements in the center were excited last, according to a sine shape. Further, we used a continuous wave excitation, which means that the voltage signal applied to the PE elements is a continuous sinusoid signal.

The pressure field in the plate was determined based on a Fast Nearfield Method (FNM) [23] in FOCUS. Then, we calculated the average pressure at each axial position from the lateral pressure amplitudes exceeding a given threshold value, Υ. For instance, the average pressure in axial position z_i was found by averaging all the pressure values $P(x, z_i)$ in the lateral direction x within the range $[\Upsilon \cdot P_{\max}(x, z_i) \leq P(x, z_i) \leq P_{\max}(x, z_i)]$. Further, we used the pressure field along the centerline to calculate the intensity and ARF fields from Equations 2.6 and 2.5, respectively. We used the same equations to determine the

averaged intensity and ARF fields from the averaged pressure field. Lastly, the averaged ARF field from the FOCUS simulation was fitted to a Gaussian function of the form

$$f(z) = \sum_{i=1}^{N} a_i e^{-\left(\frac{b_i z}{c_i}\right)^2} \tag{4.1}$$

This Gaussian function was further used in the FEM model in Chapter 5 to represent the ARF from a linear array ultrasound transducer.

Table 4.1: The material parameters used in the FOCUS simulation.

Material parameter	Value	Unit
c	1540	$\frac{\text{m}}{\text{s}}$
ρ_0	1060	$\frac{\text{kg}}{\text{m}^3}$
α	0.7	$\frac{\text{dB}}{\text{cmMHz}}$

4.2 Results

We used the MATLAB script in Appendix A.2 to calculate the 2D pressure, intensity and ARF fields caused by a linear array ultrasound transducer. Further, this script was used to plot the pressure, intensity and ARF fields and to fit the Gaussian function to the ARF field.

Figure 4.1 shows the pressure field as a colorplot in the imaging plane. The red color indicates large pressure amplitudes whereas blue color refers to the undisturbed soft tissue. The range of the colorbar is $0 \leq P \leq 6.26$ MPa. With reference to Figure 4.1, the transducer was placed at the lower edge and extended approximately 4 mm to each side in the lateral direction. Figure 4.2 displays the lateral beam profile for various axial positions, z.

The pressure along the centerline is plotted against the averaged pressure field in Figure 4.3(a). The threshold value for the average pressure was set to 80 %, which corresponds to $\Upsilon = 0.8$. The intensity along the centerline and the averaged intensity field are given in Figure 4.3(b).

We determined the ARF field from the intensity field using Equation 2.5. Further, we fitted a Gaussian function on the form of Equation 4.1 to the ARF field. We found that three terms gave reasonable accuracy with a *goodness-of-fit ratio* corresponding to $R^2_{fit} = 0.9905$. R^2_{fit} indicates how well the function is fitted to the input data. $R^2_{fit} = 1$ corresponds to a perfect fit, while $R^2_{fit} = 0$ means that the function cannot be fitted to the given data set. The three-element Gaussian function can be written

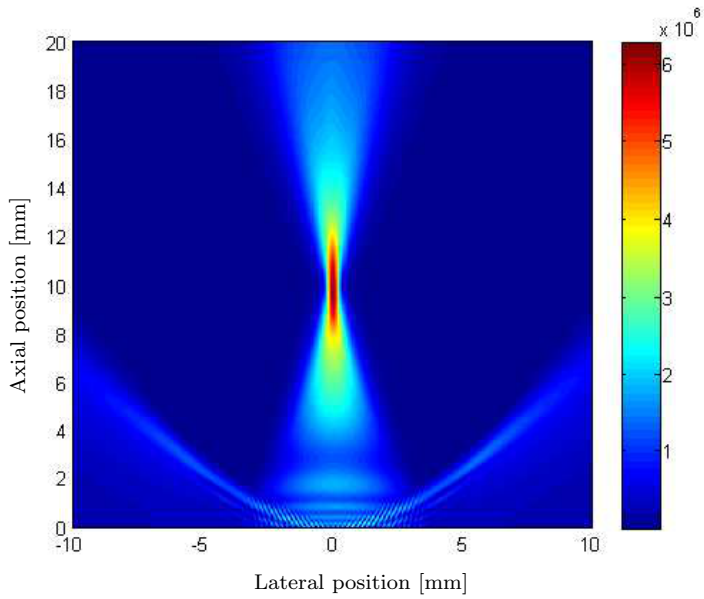


Figure 4.1: The 2D pressure field from a linear array transducer simulated in FOCUS [13]. The transducer was made up of 40 transducer elements with a center frequency of 8 MHz. The width of each element was 170 μm and the kerf was 30 μm. The material parameters correspond to soft tissue. Side lobes are seen in the near field. The transducer was located at the lower edge and extended about 4 mm to each side from the center.

$$f(z) = a_1 e^{-\left(\frac{b_1 z}{c_1}\right)^2} + a_2 e^{-\left(\frac{b_2 z}{c_2}\right)^2} + a_3 e^{-\left(\frac{b_3 z}{c_3}\right)^2} \tag{4.2}$$

The three-element Gaussian function and the averaged ARF field from the FOCUS simulation are plotted in Figure 4.4. The coefficients estimated in the curve fit are given in Table 4.2.

Table 4.2: The coefficients found from the curve fit of a three-element Gaussian function to the ARF field calculated in FOCUS [13].

i	a_i	b_i	c_i
1	$6.564 \cdot 10^5$	0.009823	0.001613
2	$-1.494 \cdot 10^5$	0.015060	0.003940
3	$3.417 \cdot 10^5$	0.011480	0.007131

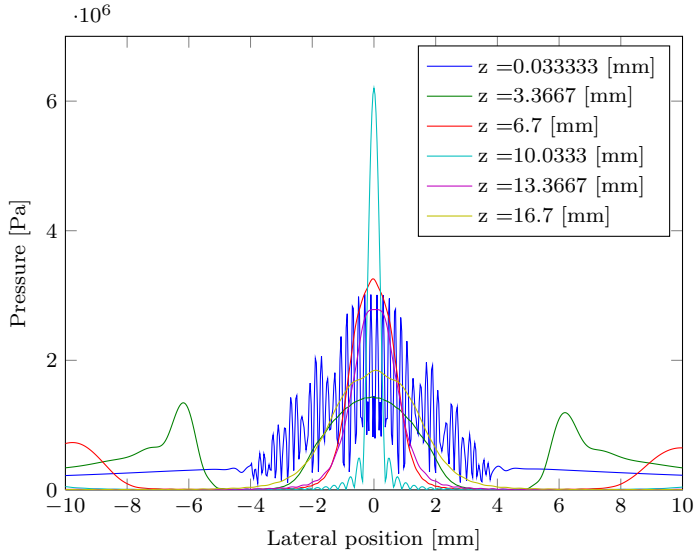
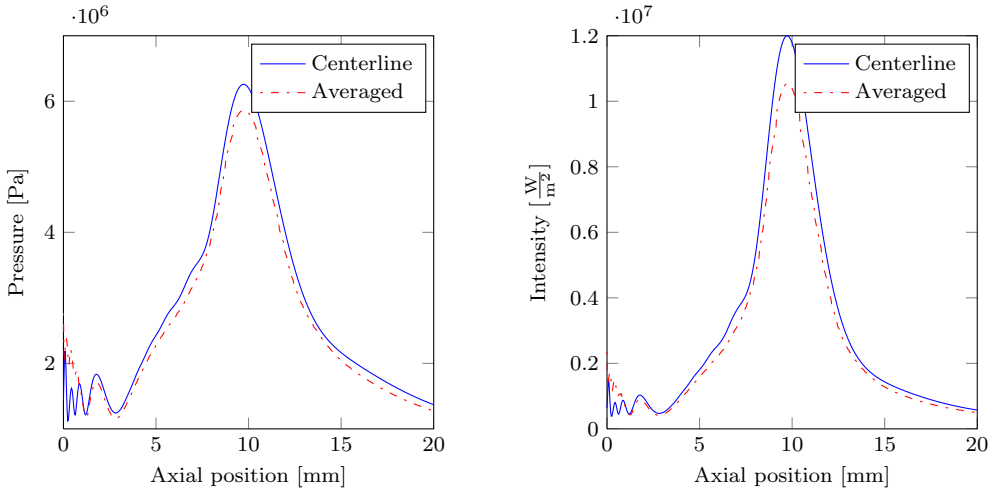


Figure 4.2: These graphs show the lateral profile of the ultrasound pressure field for different axial positions, z. We can clearly see the side lobes, especially for $z = 3.3667$ mm and $z = 6.7$ mm.



(a) Ultrasound pressure distribution in the axial direction.

(b) Ultrasound intensity distribution in the axial direction.

Figure 4.3: These graphs show the pressure distribution and the intensity distribution in the axial direction. The pressure field was calculated with the MATLAB toolbox FOCUS [13]. The distribution was found from the centerline in the axial direction and from the laterally averaged values along the axial direction.

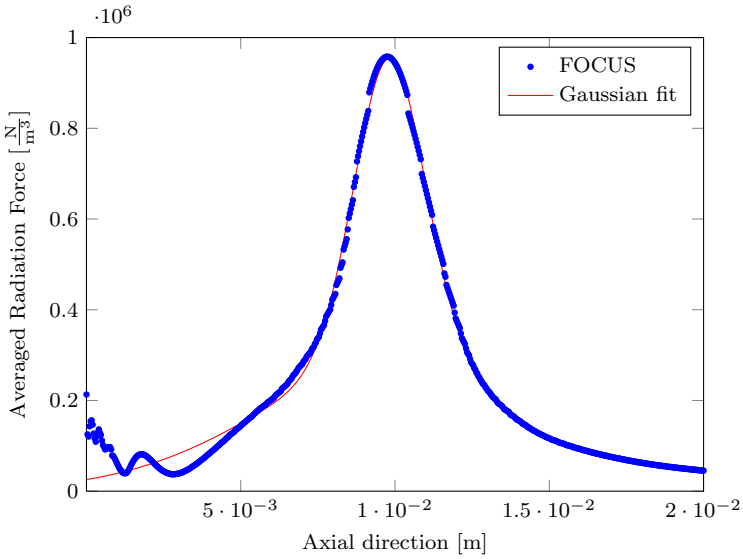


Figure 4.4: A three-element Gaussian function was fitted to the averaged ARF field from the FOCUS simulation. The Gaussian function was further implemented as an analytical body force field in ABAQUS.

4.3 Discussion

The purpose of the FOCUS simulation was to obtain realistic values for the ARF caused by a linear array ultrasound transducer. Physical ultrasound transducers induce three-dimensional pressure fields, which in turn result in three-dimensional intensity and ARF fields. However, since the FEM model in Chapter 5 is two-dimensional, we need to define a two-dimensional ARF field. Consequently, we would benefit from simulating the pressure field in 2D rather than 3D. Consider a plane that extends in the axial and lateral direction and cuts through the middle of the focal point. Such a plane is similar to the imaging plane in Figure 2.3. We believe that the 3D pressure field in this plane is fairly well represented by the 2D pressure field originating from a linear array transducer. Hence, we decided to use a linear array transducer in the simulation, which also gives the benefits of substantially reduced computational time and easier implementation.

We chose the number of PE elements in order to achieve a proper value of $F_\#$ for a focal depth of 10 mm. The resulting $F_\#$ was approximately 1.25, which seems realistic since $F_\#$ usually is between 1 and 2 for medical applications. We chose the focal depth in the simulation in order to match the focal depth from the SSI experiment on the gel-agar phantom, which was about 10 mm. We used a continuous wave excitation of the transducer, which causes a time independent pressure field. In real-life, the pressure field are generally time dependent, because the PE elements are usually subjected to pulse excitations. Since the ultrasound waves propagate away from the transducer surface through the medium with velocity c, the ultrasound beam will not generate the desired ARF field instantly. However, we believe that the continuous wave excitation is a fair approximation since the ultrasound waves reach the focal point after

$$t = \frac{L_f}{c_l} = \frac{10 \cdot 10^{-3}\,\mathrm{m}}{1540\,\frac{\mathrm{m}}{\mathrm{s}}} \approx 6.5\,\mu\mathrm{s} \tag{4.3}$$

which is significantly less than the impulse times (T_i) of the ARF impulse used in the FEM simulation in Chapter 5.

From Figure 4.1 we see that the transducer indeed is focused at an axial depth of 10 mm, which is indicated by the red region. Figures 4.3(a) and 4.3(b) show the axial pressure and intensity profiles. We observe a rapid increase in amplitude as we approach the focal point, which means that the pressure waves originating from the transducer elements gradually interfere with one another in a constructive manner. Also, the pressure and intensity profiles are relatively smooth. Consequently, we believe that the results from the FOCUS simulation are reliable and free of significant numerical errors. However, we observe some diffraction effects close to the transducer, which is seen as oscillations in the pressure distribution. From the theory presented in Chapter 2.1.2, we know that diffraction effects are prominent in the near field and dies out in the far field where the pressure waves are approximately plane. Consequently, we will always experience some oscillation just ahead of the transducer. However, we observed less oscillation when we kept $F_\#$ constant and increased the size of the transducer. This has to do with the ratio between the transducer size and the ultrasound wave length (λ). Increasing the transducer size-to-wave length

ratio reduces the amount of diffraction. Thus, we could have used a larger transducer in the simulation, in order to further reduce the amount of oscillation. This would have lead to a greater focal distance (L_F) due to the restriction that $F_\#$ should be kept between 1 and 2. Since we wanted to replicate the SSI experiment, with a focal depth of about 10 mm, L_F was already predetermined. This put an obvious restriction on the transducer size. We also noticed that the amount of oscillation was dependent upon the height of the transducer elements, even though the simulation was 2D. In an effort to minimize the amount of oscillation, we carried out simulations with different element height. We found that an element height of about 3 mm gave the best results. Hence, we used this element height in the final simulation. We are not yet aware of the reason for this behavior.

We note that many ultrasound transducers used in medical applications consist of 192 piezoelectric (PE) elements. With this number of elements and the same element size and kerf as we used in the simulation, the transducer width would have been approximately 4 cm. This further implies that the axial distance to the focal point (L_F) had to be between 4 cm and 8 cm in order to keep $1 \leq F_\# \leq 2$, which is substantially higher than what was used in the SSI experiment. However, most transducers excites only some of the elements simultaneously, which justifies the choice of 40 elements and the use of a smaller transducer in the simulation.

The amount of diffraction was also dependent on the center frequency (f_0). We used center frequencies between 1 and 20 MHz and observed that lower center frequencies in general induced more oscillations in the pressure field. This can be explained using the same transducer size-to-wave length argument. The ultrasound wave length is given as

$$\lambda = \frac{c}{f_0} \tag{4.4}$$

where c is the wave speed of the tissue and f_0 the center frequency of the transducer. Hence, we see that lower frequencies result in larger wave lengths, which in turn cause smaller transducer size-to-wave length ratios. From this we readily understand that increasing the transducer width or the center frequency have the same effect on the amount of diffraction and thus, amount of oscillation. The magnitude of the ARF is also dependent upon the frequency. Ultrasound waves of higher frequencies are more heavily attenuated, which causes larger radiation force magnitudes. This makes it important to use the same value as in the experiment, in order to obtain similar radiation force magnitudes. Hence, we used a center frequency of 8 MHz in order to match the SSI experiment.

Figure 4.2 shows the lateral beam profile. We clearly see how the ultrasound beam narrows down towards the focal point before it reaches maximum pressure, indicated by the spike at $z = 10.0333$ mm. The beam takes the form of half a sine wave close to the transducer surface, which is due to the applied apodization. We also observe that the lateral pressure field is affected by rapid oscillations close to the transducer, which can be seen for $z = 0.033$ mm. This stems from the alternating constructive and destructive interference of wave fronts originating from the transducer elements, which was briefly explained in Chapter 2.1.2. Farther away from the transducer, the wave fronts gradually merge and form a plane wave without oscillations.

We readily see from Figures 4.1 and 4.2 that the simulated transducer gave rise to side lobes, despite the fact that the transducer was apodized. We believe that the appearance of these side lobes is related to center frequency and transducer width. Firstly, we noticed that the side lobes became apparent for center frequencies larger than about 7 MHz and that they were more pronounced when the center frequency was increased. Secondly, we observed that the side lobes were less pronounced when we increased the width of the transducer. As explained in Chapter 2.1.2, the side lobes stems from transversal vibration modes. We think that these transversal modes are less significant as the number of elements and the lateral size of the transducer increases, which in turn explains why the appearance of side lobes was dependent on the transducer size. Even though side lobes are undesired in medical imaging, they do not represent a problem in our simulation since we are concerned with the incident ultrasound wave rather than echoes from reflection and backscatter. Hence, we did not put in additional effort to get rid of the side lobes.

From Figure 4.2 we see that the pressure is not constant in the lateral direction, which implies that the intensity field and the ARF field are not constant in the lateral direction. We also observe that the intensity and ARF values are greatest along the centerline. Since the ARF field from the entire main lobe of the ultrasound beam was applied as a body force in the FEM model, we had to take this lateral change of amplitude into account. The total force applied to the tissue in the FEM simulation would otherwise have been overestimated. Hence, we averaged the pressure field in the lateral direction, which further resulted in averaged intensity and ARF fields. The averaging threshold parameter Υ defined in Chapter 4.1 was used to define which lateral pressure values that should be taken into account. We chose a value of 0.8, which means that the lateral pressure values that are greater than 80 % of the maximum lateral pressure are used to calculate the averaged pressure field. From Figure 4.3 we see that the averaged pressure and intensity values are slightly smaller than the values along the centerline and that the shape of the two curves is similar. The exception is for axial positions very close to the transducer, where the averaged pressure is somewhat larger than the centerline pressure. This is caused by oscillations in the lateral pressure distribution close to the transducer, which make the pressure magnitudes highly dependent upon the lateral position. The centerline pressure is in this case lower than the average pressure because the wave fronts interfere destructively along the centerline. This may be seen in Figure 4.2 for $z = 0.0333$ mm. We could argue that a threshold value of 0.5 would have been better suited, since this value also refers to the 6-dB response margin that is used to define the ultrasound beam width. This was outlined in Chapter 2.1.2.

Lastly, we fitted a Gaussian function to the averaged ARF field. We chose a Gaussian function because the averaged ARF field nearly has a Gaussian profile, with a distinct peak at the focal point and gradually decreasing amplitude away from the focal point. The Gaussian function and the averaged ARF field from the FOCUS simulation were plotted in Figure 4.4. We readily see that the Gaussian function gives an exact representation for axial positions greater than about 5 mm. This is not a surprising result, since the ARF distribution is approximately Gaussian in this region. However, closer to the transducer surface we see that there is a mismatch between the two curves. We believe that this has little impact on the FEA results since the ARF magnitude in this area is relatively small

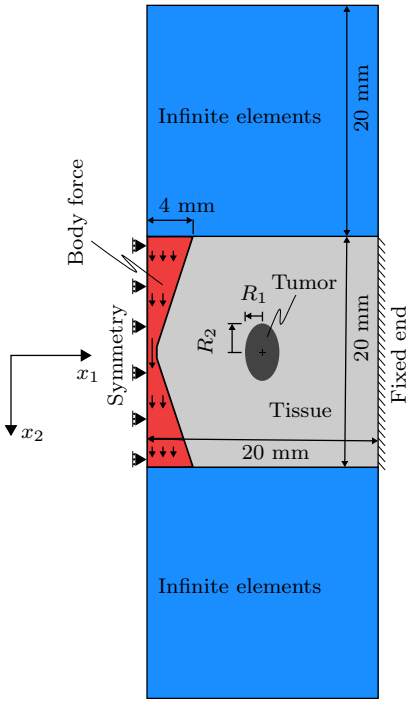
compared to the focal point. Also, the ARF values from the FOCUS simulation may not be entirely correct in this region, due to diffraction. Hence, we are not concerned about the deviance this close to the transducer.

5 An FEM model of a Soft Tissue with a Tumor inclusion

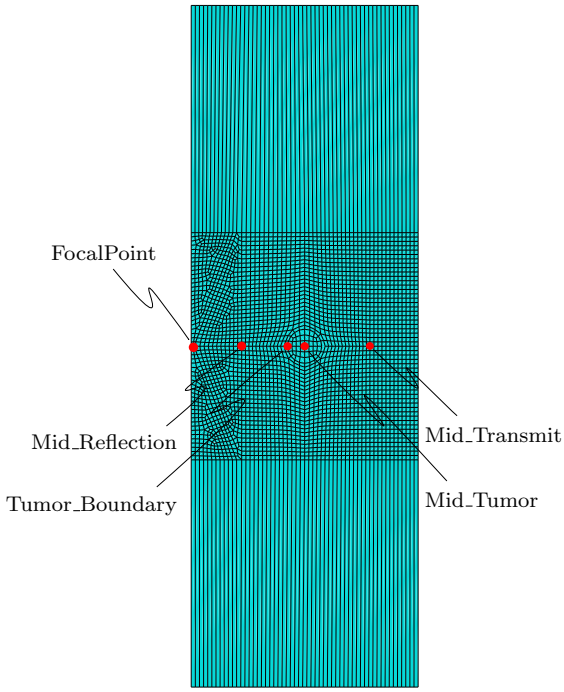
We wanted to employ the FEM to simulate a local axial displacement remotely induced by the ARF impulse from a focused ultrasound transducer and investigate the resulting shear wave propagation in a soft tissue with a tumor inside. The axial displacement and the shear wave were further used to examine different ways of estimating the elastic stiffness of the soft tissue and tumor.

5.1 Method

We made a two-dimensional plane strain plate model in Abaqus that represents a soft tissue segment cut out along the imaging plane of the ultrasound transducer, illustrated in Figure 2.3. The model is shown in Figure 5.1. Figure 5.1(a) indicates the relevant dimensions of the model. The depth of the tissue was 20 mm, which corresponds to the axial length ahead of the linear array transducer from the Focus simulation in Chapter 4. The lateral width was also set to 20 mm. We modeled the tumor as an elliptical shaped region with the possibility of changing the radius along the major and minor axes. The two radii are given as R_1 and R_2 in Figure 5.1(a). The default values of the radii are $R_1 = R_2 = 1.5$ mm which correspond to a circular inclusion. The x_1-axis corresponds to the transversal direction, while the x_2-axis corresponds to the axial direction ahead of the ultrasound transducer. The plate was fixed at the right edge and symmetric along the x_2-axis at the left edge ($u_1 = 0$). The ARF impulse was modeled as a body force impulse by implementing the Gaussian function from Equation 4.2 with the parameters from Table 4.2 as an analytical field. This analytical field was further assigned to a region of the plate that represents the shape of the focused ultrasound beam, indicated by the red colored region in Figure 5.1(a), and applied for a given impulse time (T_i). The shape of the focused ultrasound beam corresponds to the shape of the pressure field in Figure 4.1. We note that only half of the ARF field was modeled since the ARF field was symmetric about the centerline. The ARF field is illustrated in the red region of Figure 5.1(a) where we have indicated that the body force increases towards the focal point and that the half-width of the transducer was 4 mm.



(a) An illustration of the two-dimensional plane strain model. The plate is symmetric along the x_2-axis at the left end, fixed at the right end and connected to regions with infinite elements at the upper and lower surface. The red colored region represents the ARF from the focused ultrasound beam. The tumor is shown as an elliptical shaped region with minor and major radii R_1 and R_2.



(b) The ABAQUS model is shown with element size corresponding to n=4. The element sets are highlighted with a red dot and are used to extract relevant results from the FEA. From left-to-right, the elements sets are; FocalPoint, Mid_Reflection, Tumor_Boundary, Mid_Tumor and Mid_Transmit.

Figure 5.1: The two-dimensional plane strain model implemented in ABAQUS. Relevant geometry, boundary and loading conditions, tissue, tumor and infinite sections are shown in the left figure. The right figure illustrates a meshed model and highlights the relevant element sets.

We divided the model into three different sections corresponding to the *soft tissue*, the *tumor* and an *infinite region* that was used to prevent reflections. We used 4-node, linear, plane strain elements with reduced integration (CPE4R) in the tissue and tumor sections and assigned infinite plane strain elements (CINPE4) to the infinite section. We used hourglass control in order to prevent zero-energy deformation modes. The infinite elements were used to damp out reflections from the upper and lower boundaries in the tissue section, as briefly explained in Chapter 2.6.4. We applied a free meshing technique in the infinite section and body force region and a structured meshing technique in the tumor and remaining part of the tissue section, indicated by the grey region in Figure 5.1(a).

We chose free meshing in the body force region such that the elements were better shaped. The approximate global element size in the soft tissue section was determined based on the shear wave speed of the tissue ($c_t^{(\text{tissue})}$), the impulse time of the ARF (T_i) and the number of elements the impulse load should span (n). The lateral distance travelled by the shear wave through the tissue within the impulse time, referred to as the *impulse length*, is given by

$$\lambda_I^{(\text{tissue})} = c_t^{(\text{tissue})} \cdot T_i \tag{5.1}$$

which further can be used to determine the approximate element size in the tissue section as

$$\Delta x^{(\text{tissue})} = \frac{\lambda_I^{(\text{tissue})}}{n} \tag{5.2}$$

where I denotes *impulse*. Since the infinite elements share nodes with adjacent finite elements, their width correspond to the width of the element which they are connected to. We used slightly larger elements in the tumor section in order to prevent that very small elements were generated at the boundary between the tumor and tissue. Very small elements at the boundary would reduce the Courant number (C_n) in the remaining parts of the FEM model since Δt_{cr} is calculated based on the smallest element in the mesh. This can be seen from Equations 2.136 and 2.135. Values of C_n that are much lower than unity reduce the accuracy of the simulation. Hence, the approximate element in the tumor section was

$$\Delta x^{(\text{tumor})} \approx 1.3 \cdot \Delta x^{(\text{tissue})} \tag{5.3}$$

We wanted the possibility to change between a purely elastic material and a viscoelastic material in the soft tissue section, in order to examine differences between the two material definitions. We used the elastic material parameters E_0, ν_0 and ρ_0 from Table 3.1 as *default values* for both the purely elastic material and the elastic part of the viscoelastic material. The inelastic part of the viscoelastic material was defined by the relaxation coefficients found from the calibration of the Maxwell model in Chapter 3.2, given in Table 3.2. In the infinite section, we used the same elastic material parameters as in the soft tissue region. Both because a solid material section with linear elastic behavior must be used to define the infinite elements and because the infinite elements should have the same material properties as the adjacent finite element, in order to damp out reflections properly. The latter can be seen from the definition of the damping coefficients in Equations 2.144 and 2.145. The tumor was also modeled as an elastic material with the same values of ν and ρ (Table 3.1), but with a larger E-modulus than in the soft tissue region.

We used explicit time integration with automatic time incrementation and set the analysis time $T_a = 4$ ms as default value, in order for the shear wave to traverse the entire plate. The impulse time of the ARF was $T_i = 250\,\mu$s by default. We used a smooth step amplitude to apply the ARF impulse in order to avoid numerical instabilities. The impulse

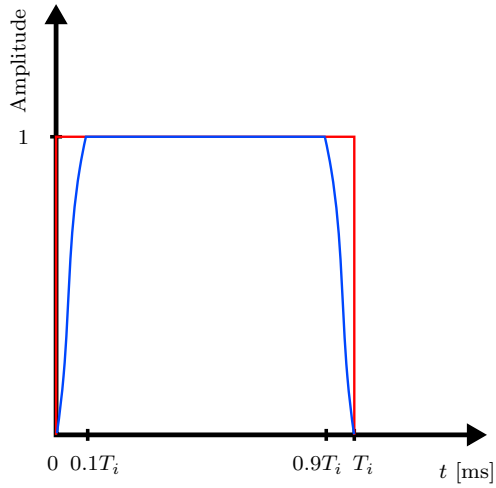


Figure 5.2: The smooth step amplitude applied to the body force. Red lines indicate a square pulse with impulse time T_i. Blue lines indicate the applied smoothed pulse.

shape is given in Figure 5.2. The displacements and Cauchy stresses were extracted as field output for the entire model. This made it possible to visualize the shear wave propagation. We created node sets at selected locations in the FEM model from which we extracted numerical values for shear stress (σ_{12}) and axial displacement (u_2). These node sets are referred to as *FocalPoint*, *Mid_Reflection*, *Tumor_Boundary*, *Mid_Tumor* and *Mid_Transmit* in Figure 5.1(b).

The plane strain model will be used in all subsequent analyses and is the basis for studies on mesh convergence, shear wave speed estimation, shear wave reflection and Time to Peak (TTP) displacement in the focal point. The FEM model can be loaded in Abaqus from the python script in Appendix B.2.

Mesh convergence study

In order to examine the mesh dependency of the model and determine a proper element size for future simulations, we ran five simulations with different element size. We changed the value of n to obtain various mesh densities. The approximate element size in the tissue and tumor sections was further calculated from Equations 5.2 and 5.3, respectively. The values of n, $\Delta x^{(\text{tissue})}$ and the total number of elements $\#^{el}$ used in the study are given in Table 5.1. We used the viscoelastic material definition with default elastic material properties in all simulations. We used the Matlab script in Appendix A.4 to extract the shear stress $\sigma_{12}(t)$ and the axial displacement $u_2(t)$ from the node sets *Mid_Reflection* and *Tumor_Boundary* and make plots for different mesh densities.

Table 5.1: The number of elements the impulse load should span (n), the corresponding element size in the soft tissue section ($\Delta x^{(\text{tissue})}$) and the number of elements ($\#^{el}$) used in the FEM simulations.

n	$\Delta x^{(\text{tissue})}$ [mm]	$\#^{el}$
6	0.24243	7167
8	0.18183	12280
10	0.14546	19230
12	0.12122	27795
14	0.1142	38029

Wave speed study

In Chapter 2.4 derived the theory of shear wave propagation in both linear elastic and linear viscoelastic materials. From this theory, we know that the shear wave speed in such materials is related to the shear stiffness through the simple Equations 2.66 and 2.94 for elastic and viscoelastic materials, respectively. In this study, we wanted to examine whether the shear wave speed found in the FEM simulations could be used to estimate the stiffness of the soft tissue and the tumor.

We were provided with shear wave speed data from the Supersonic Shear Imaging (SSI) experiment carried out at Ghent University in collaboration with Institut Langevin in Paris [14]. They used a SuperSonic Imagine Aixplorer ultrasound system [24] to visualize the shear wave propagation in a gel-agar phantom similar to that of the relaxation test in Chapter 3.1. Detailed information about SSI imaging can be found in [9, 10]. The focal depth (L_F) of the ultrasound transducer was changed in order to induce shear waves at different tissue depths. The shear wave speed was determined and plotted against the focal depth, referred to as *tissue depth* in Figure 5.3.

We wanted to determine the numerical shear wave speed in the soft tissue and tumor sections of our plane strain model. In the soft tissue section, we used both the elastic and the viscoelastic material definition with the default elastic material parameters from Table 3.1. The resulting numerical shear wave speed values were compared to the analytical shear wave speeds found from Equations 2.66 and 2.94 with $G = G_0 = 35885.6$ Pa. In theory, the shear wave speeds for the elastic and the viscoelastic material are the same when the elastic parameters G and ρ correspond to the viscoelastic parameters G_0 and ρ_0. The shear wave speed in the soft tissue section was also used in a comparison with the experimental results from Figure 5.3.

In order to calculate the shear wave speed in the soft tissue and tumor sections from the FEA results, we measured the time between shear stress peaks in different nodes. Then we determined the shear wave speed from

$$c_t = \frac{\delta x_1}{\Delta t} \tag{5.4}$$

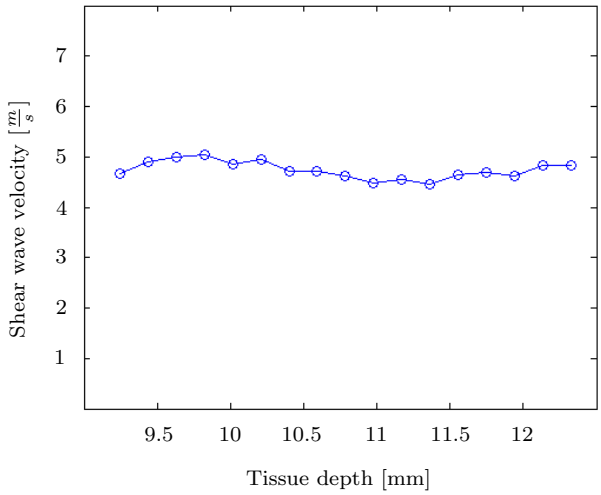


Figure 5.3: The shear wave speed determined in the SSI experiment with a SuperSonic Imagine Aixplorer ultrasound system [24]. The depth refers to the axial position of the focal point in the imaging plane. The shear wave speed data are provided by Ghent University.

where δx_1 refers to the lateral distance between the nodes we extract the shear stress from and Δt is the time between the stress peaks. We used node sets *Mid_Reflection* and *Tumor_Boundary* to determine $c_t^{(\text{tissue})}$ and node sets *Tumor_Boundary* and *Mid_Tumor* to determine $c_t^{(\text{tumor})}$. The distance between the nodes and the time at which the shear wave reaches the different nodes are indicated in Figure 5.4. The shear wave reaches *Mid_Reflection* at time t_1, *Tumor_Boundary* at t_2 and *Mid_Tumor* at t_3. Hence, we find the time between the shear stress peaks in the tissue and tumor sections from

$$\Delta t^{(\text{tissue})} = t_2 - t_1 \quad \text{and} \quad \Delta t^{(\text{tumor})} = t_3 - t_2 \tag{5.5}$$

The shear wave front is indicated with blue color and represents the shear stress peak at times t_1, t_2 and t_3. The MATLAB script in Appendix A.5 was used extract the FEA results. The peak times and peak stress values were determined in each of the three node sets. The distance between the nodes ($\delta x_1^{(\text{tissue})}$ and $\delta x_1^{(\text{tumor})}$) were known in advance from the FEM model. Further, the peak times were used to calculate the shear wave speed in the tissue and tumor sections from Equation 5.4.

Reflection study

In Chapter 2.5 we presented the theory of plane shear wave reflections from plane material boundaries. From this we readily understand that the reflected shear wave carries information regarding the stiffness ratio between the soft tissue and an inclusion with different elastic stiffness.

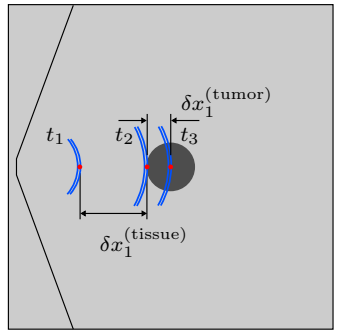


Figure 5.4: The shear wave speed was determined by measuring how long it took the wave to travel between two nodes. The shear wave front is indicated by the blue curved lines and refers to the stress peak from the FEM simulations.

In this study, we wanted to examine whether the ratio between the amplitudes of the reflected shear wave and the incident shear wave calculated from the FEA results could be used to estimate the elastic stiffness of the tumor. In order to do this, we varied the stiffness of the tumor while keeping the stiffness of the soft tissue constant. We used the viscoelastic material definition with the default elastic material properties in the soft tissue section. The stiffness of the tumor was changed as an integer multiple of the tissue stiffness, according to the stiffness ratio values in Table 5.2. The shape of the tumor was initially circular with $R_1 = R_2 = 1.5$ mm.

The numerical reflection factor could be determined from the expression

$$R^{\text{num}} = \frac{|\sigma_{12}^{\text{peak}}|_R}{|\sigma_{12}^{\text{peak}}|_I} \tag{5.6}$$

where the subindices R and I denote the *reflected* and *incident* waves, respectively. We calculated the analytical reflection factor from Equation 2.120 for every distinct value of $E^{(\text{tumor})}$ and used these as reference values for comparison with the numerical results. However, we note that this equation is only valid for plane shear waves and plane material boundaries. Even though we can locally approximate the shear waves as plane, the material boundary between the soft tissue and the tumor is certainly not plane. In an effort to approximate plane material boundary conditions in our model, we carried out simulations with an elliptical shaped tumor. The stiffness of the tumor and the minor radius were kept constant, $E^{(\text{tumor})} = 8E^{(\text{tissue})}$ and $R_1 = 1.5$ mm, while R_2 was increased according to Table 5.2.

The MATLAB script in Appendix A.7 was used to load the files from the FEM simulation, determine the numerical reflection factor values and compare these with the analytical reflection factor values.

Table 5.2: The stiffness ratio $\frac{E^{(\text{tumor})}}{E^{(\text{tissue})}}$ and the major radius R_2 used in the FEA. $R_1 = 1.5$ mm in all simulations.

Stiffness ratio	2	3	4	5	6	7	8	8	8	8	8
R_2 [mm]	1.5	1.5	1.5	1.5	1.5	1.5	1.5	3	5	7	9

Time to Peak Displacement

As previously discussed by Sarvazyan et al. [7] and Palmeri et al. [12], Time to Peak (TTP) displacement is related to the elastic stiffness of the soft tissue. The tissue responds quicker when the stiffness increases, due to greater elastic restorative forces that oppose the inertia forces. This means that we reach maximum displacement sooner in a stiffer tissue. Hence, the TTP displacement is inversely related to the stiffness. Sarvazyan et al. have given a relation between the shear stiffness and the TTP displacement as

$$G = \rho \left(\frac{aD}{t_{\max}} \right)^2 \quad \Rightarrow \quad t_{\max} = \sqrt{\frac{\rho}{G}} aD = \frac{aD}{c_t} \tag{5.7}$$

where a and D are the Gaussian profile parameter and a dimensionless diffraction parameter, respectively, and $t_{\max}$ is the TTP displacement. The parameters a and D are dependent upon the size and curvature of the transducer elements and the frequency of the ultrasound wave, which can be seen in Sarvazyan et al. [7]. This means that the TTP displacement is also dependent upon these parameters, since $t_{\max}$ is seen to be proportional to a and D in Equation 5.7. This is also discussed in Palmeri et al. [12], where it is shown that the TTP displacement is dependent upon the focal configuration ($F_\#$) and impulse time (T_i), which can be linked to a and D. However, Equation 5.7 is only valid for a perfect Gaussian ultrasound beam. The reader is referred to [7] for details.

In order to find out whether the TTP displacement is a reliable estimate of the elastic stiffness, we examined how the elastic stiffness and impulse time affects the TTP displacement in the focal point. We used the viscoelastic material definition in all subsequent simulations. First, we varied the stiffness of the tissue and used a constant impulse time, $T_i = 250\,\mu$s. Afterwards, we carried out simulations with different values of T_i while the elastic stiffness was constant, $E_0^{(\text{tissue})} = 107585$ Pa. The analysis time was set to $T_a = 2$ ms, since we are only concerned with the temporal axial displacement profile in the focal point before any reflection waves returns from the tumor.

The temporal axial displacement distribution $u_2(t)$ was extracted from node set *FocalPoint* and further loaded in MATLAB. The TTP displacement was determined through the script in Appendix A.8.

Table 5.3: The soft tissue stiffness, impulse time and analysis time values used in the TTP displacement simulations.

$E_0^{(\text{tissue})}$ [kPa]	T_i [μs]	T_a [ms]
60	250	2
80	250	2
100	250	2
107.585	50	2
107.585	100	2
107.585	150	2
107.585	200	2
107.585	250	2
120	250	2
140	250	2
160	250	2

5.2 Results

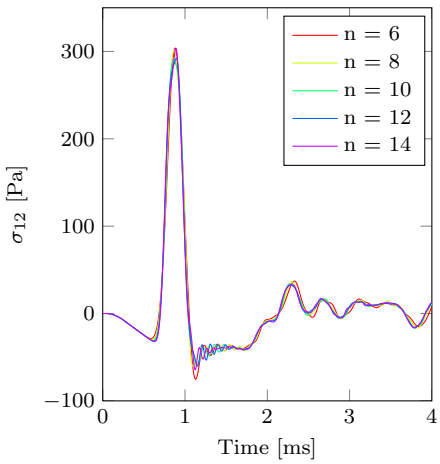
The ARF field approximated by the Gaussian function in Equation 4.2 with the coefficients from Table 4.2 was implemented in the two-dimensional plane strain model given in Appendix B.2. We used the plane strain model explained in Chapter 5.1 in all subsequent analyses.

Mesh Study

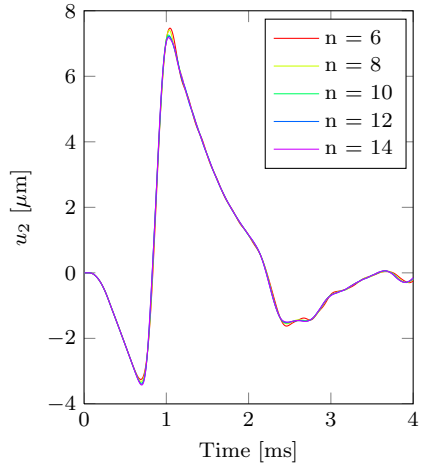
We used n, $\Delta x^{(\text{tissue})}$ and $\#^{el}$ corresponding to Table 5.1 in the FEM simulations. The temporal shear stress $\sigma_{12}(t)$ and axial displacement $u_2(t)$ distributions in node sets *Mid_Reflection* and *Tumor_Boundary* were extracted from the FEA results and are plotted in Figures 5.5 and 5.6, respectively. We note that the difference between the curves are small, even though the number of elements ranges from 7167 to 38029.

Wave Speed Study

We extracted the temporal shear stress distribution $\sigma_{12}(t)$ from the node sets *Mid_Reflection*, *Tumor_Boundary* and *Mid_Tumor* and found the peak time and peak stress values. Figures 5.7 and 5.8 show the shear stress distribution in all three node sets for the viscoelastic and elastic material definition, respectively. The peak stress values are indicated with a red dot and the corresponding peak time values were found by recognizing the time of the peak stress. We note that the curves appear to be similar for both the viscoelastic and the elastic material definition.

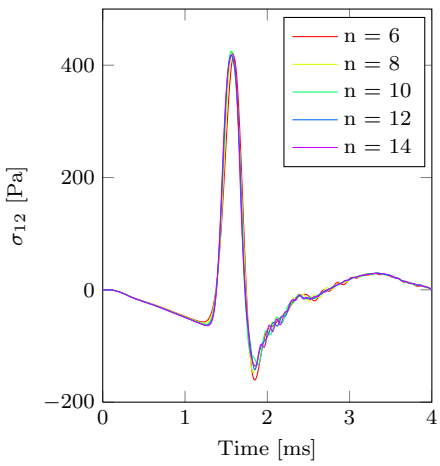


(a) Shear stress $\sigma_{12}(t)$ extracted from the node set *Mid_Reflection*.

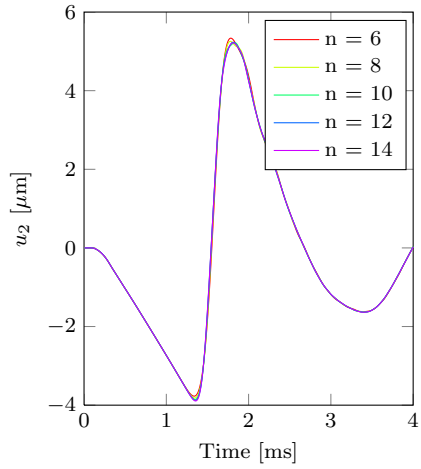


(b) Axial displacement $u_2(t)$ extracted from the node set *Mid_Reflection*.

Figure 5.5: The shear stress and axial displacement in node set *Mid_Reflection* are plotted against time for different mesh densities. n corresponds to the number of elements that the shear wave traverses during the impulse time.



(a) Shear stress $\sigma_{12}(t)$ extracted from the node set *Tumor_Boundary*.



(b) Axial displacement $u_2(t)$ extracted from the node set *Tumor_Boundary*.

Figure 5.6: The shear stress and axial displacement in node set *Tumor_Boundary* are plotted against time for different mesh densities. n corresponds to the number of elements that the shear wave traverses during the impulse time.

Table 5.4: The numerical shear wave speed for the viscoelastic and the elastic material definitions with the same elastic material parameters (E, ν, ρ). The numerical shear wave speed was found from the peak shear stress values. The analytical shear wave speed was calculated from Equation 2.66. The deviance between numerical results and analytical values are given in %.

	Tissue	Tumor
Analytical c_t $[\frac{\text{m}}{\text{s}}]$	5.8184	10.0778
Numerical c_t $[\frac{\text{m}}{\text{s}}]$	5.8098	8.7525
Deviance [%]	-0.1493	-13.1513

We determined $\Delta t^{(\text{tissue})}$ and $\Delta t^{(\text{tumor})}$ from Equation 5.5 and used these values to calculate $c_t^{(\text{tissue})}$ and $c_t^{(\text{tumor})}$ from Equation 5.4. The resulting c_t values for the viscoelastic and the elastic material definition were equal and are both given in Table 5.4. The analytical shear wave speed for the elastic and viscoelastic material definition were calculated from Equation 2.66, since the elastic material parameters were similar in both material definitions. The deviance between numerical and analytical values are also given in Table 5.4. We note that the material in the tumor section was purely elastic in both analyses.

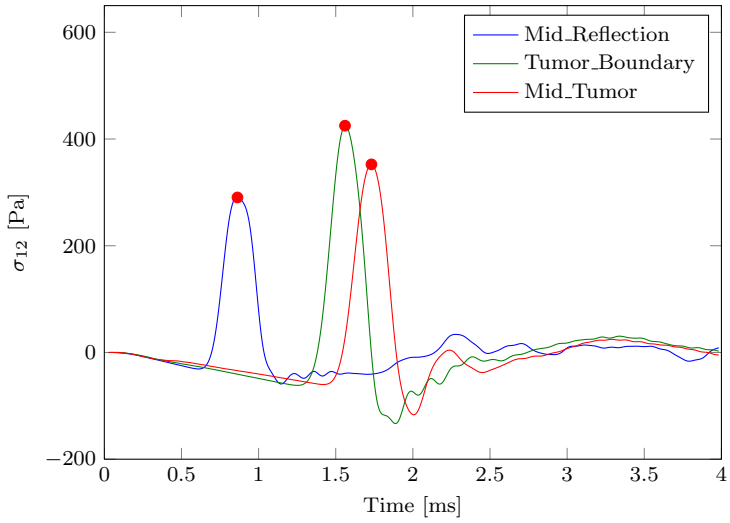


Figure 5.7: The shear stress distribution from node sets *Mid_Reflection, Tumor_Boundary* and *Mid_Tumor* for the viscoelastic material definition. The shear stress peaks are indicated with red dots. The time difference $\Delta t^{(\text{tissue})}$ between the first and second peak was used to estimate $c_t^{(\text{tissue})}$. The time difference $\Delta t^{(\text{tumor})}$ between the second and third peak was used to determine $c_t^{(\text{tumor})}$.

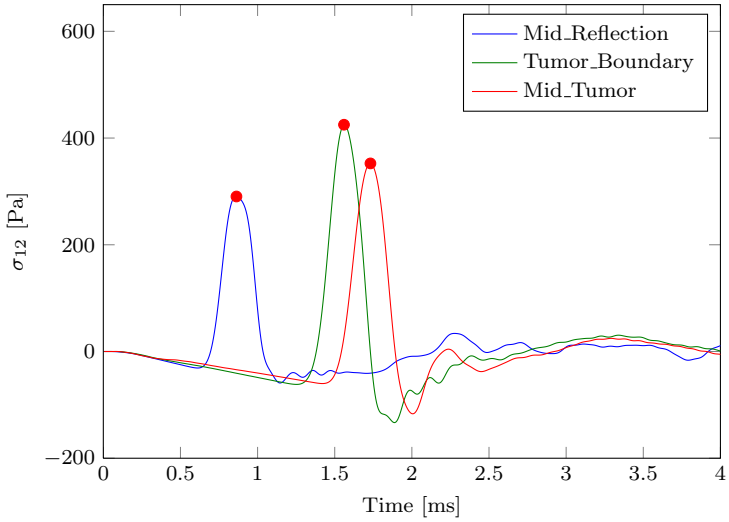


Figure 5.8: The shear stress distribution from node sets *Mid_Reflection, Tumor_Boundary* and *Mid_Tumor* for the elastic material definition. The shear stress peaks are indicated with red dots. The time difference $\Delta t^{(\text{tissue})}$ between the first and second peak was used to estimate $c_t^{(\text{tissue})}$. The time difference $\Delta t^{(\text{tumor})}$ between the second and third peak was used to determine $c_t^{(\text{tumor})}$.

Reflection Study

The temporal shear stress distribution $\sigma_{12}(t)$ in the node set *Mid_Reflection* was extracted from the FEA results. Further, the peak shear stress of the incident and reflected shear waves were found. This is shown in Figure 5.9, where the blue and the green dot indicate the incident shear wave peak and the reflected shear wave peak, respectively. From this figure, we also see that $\sigma_{12} < 0$ before the incident and reflected waves arrived at the node, which is indicated by the yellow and the magenta dot. This means that $\sigma_{12} = 0$ cannot be used as a reference value for determining the stress peaks. Consequently, we used the shear stress values at the yellow dot and the magenta dot as reference values for the incident and reflected shear wave, respectively. Thus, rather than using Equation 5.6 directly, we found the numerical reflection factor from

$$R^{\text{num}} = \frac{\sigma_{12}^{(\text{green})} - \sigma_{12}^{(\text{magenta})}}{\sigma_{12}^{(\text{blue})} - \sigma_{12}^{(\text{yellow})}} \tag{5.8}$$

where $\sigma_{12}^{(\text{magenta})}$ and $\sigma_{12}^{(\text{yellow})}$ represent the reference stress level for the reflected and incident shear wave, respectively. $\sigma_{12}^{(\text{green})}$ and $\sigma_{12}^{(\text{blue})}$ are the peak shear stress values for the reflected and the incident shear wave, respectively. The resulting values of R^{num} and the analytical reflection factor (R^{ana}) for different stiffness ratios are plotted in Figure 5.10(a), while the deviance between R^{num} and R^{ana} is given in Figure 5.10(b).

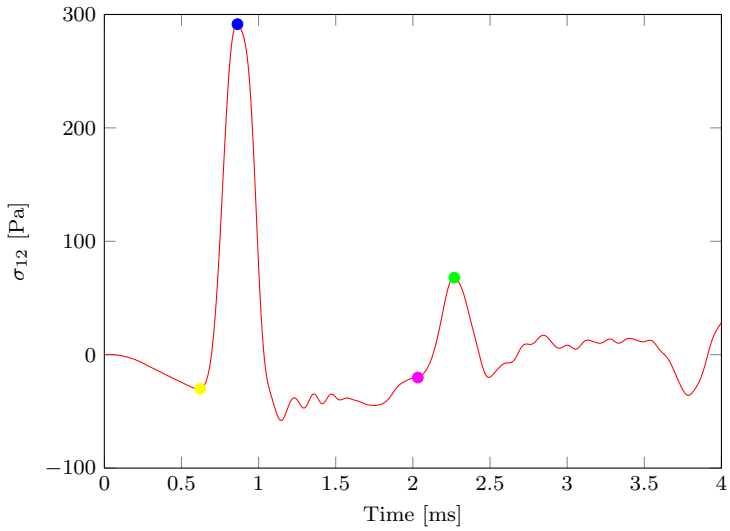
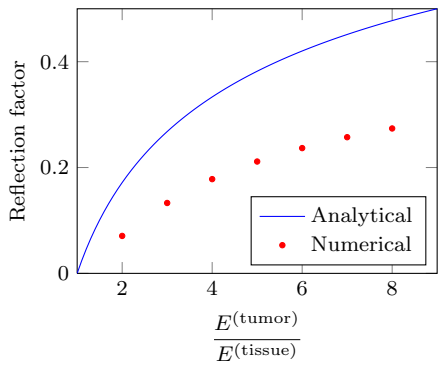


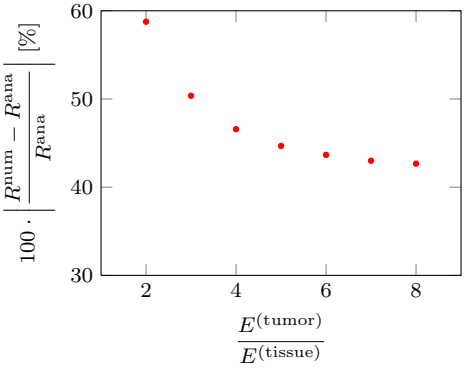
Figure 5.9: The temporal shear stress distribution for $E^{(\text{tumor})} = 8E^{(\text{tissue})}$ in node set *Mid_Reflection*. The amplitudes used to determine the numerical reflection factor (R^{num}) are indicated by blue and green dots. R^{num} was found as the ratio between the reflected amplitude and the incident amplitude. The tumor was modeled as a circular inclusion with $R = 1.5$ mm in this figure. The yellow dot and the magenta dot indicate the stress level as the incident and reflected waves arrived, and was used as reference stress values.

We also determined R^{num} for increasing values of the major radius R_2 while the stiffness ratio was kept constant, $E^{(\text{tumor})} = 8E^{(\text{tissue})}$. The resulting values of R^{num} are shown in Figure 5.11(a) where R^{ana} is plotted for comparison. The deviance between R^{num} and R^{ana} is given in Figure 5.11(b). We note that the deviance decreases as the major radius increases.

Figure 5.12 shows field plots of the shear stress at four different times to demonstrate the shear wave propagation and the reflections from the tumor boundary. The tumor was circular with $R = 1.5$ mm and the stiffness of the tumor was set to $E^{(\text{tumor})} = 8E^{(\text{tissue})}$ in order to clearly visualize the reflections. The light grey regions have stress levels above 100 Pa, while the black region have stress levels below -100 Pa. Green color corresponds to approximately zero stress. Red, orange and yellow colors are intermediate and decreasing positive stress levels, respectively. Blue colors correspond to intermediate and negative stress levels. The infinite elements are indicated by the light grey sections on the upper and lower boundaries. We note that the infinite elements are not shown in their full size.

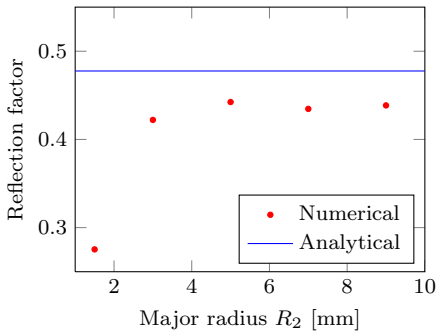


(a) The numerical reflection factor found from the FEA are compared to the analytical expression. R^{num} were found from the shear stress in node set *Mid_Reflection*. The reflection factor is underestimated in the FEA.

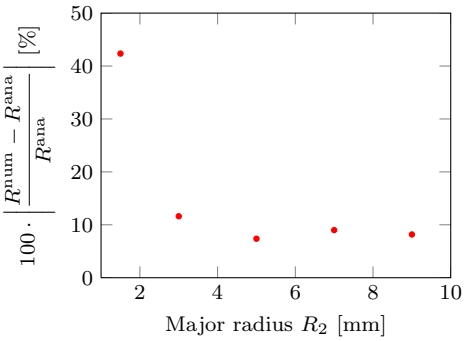


(b) The deviance between the numerical vales and the analytical values are given in %. The discrepancy is seen to remain fairly constant for the different elasticity ratios.

Figure 5.10: The reflection factor was found numerically by determining the ratio between the reflected shear stress amplitude and the incident shear stress amplitude in the node set *Mid_Reflection*. Both the reflection factor values and the deviance is plotted against the ratio between tumor stiffness and tissue stiffness, $\frac{E^{(\text{tumor})}}{E^{(\text{tissue})}}$.



(a) Major radius R_2 was increased and the numerical reflection factor was determined from the incident and reflected shear stress peak in node set *Mid_Reflection*. The analytical value from plane shear wave reflection theory is indicated by the blue line.



(b) The deviance between the numerical values and the plane wave theory is given in %. The discrepancy is seen to decrease as R_2 is increased. R^{num} is always lower than R^{ana}.

Figure 5.11: The numerical reflection factor was determined from the shear stress in node set *Mid_Reflection* for several major radius values R_2, while the stiffnes of the tumor was kept constant at $E^{(\text{tumor})} = 8E^{(\text{tissue})}$. The deviance between the numerical values and the plane wave theory is seen to decrease as the major radius increases.

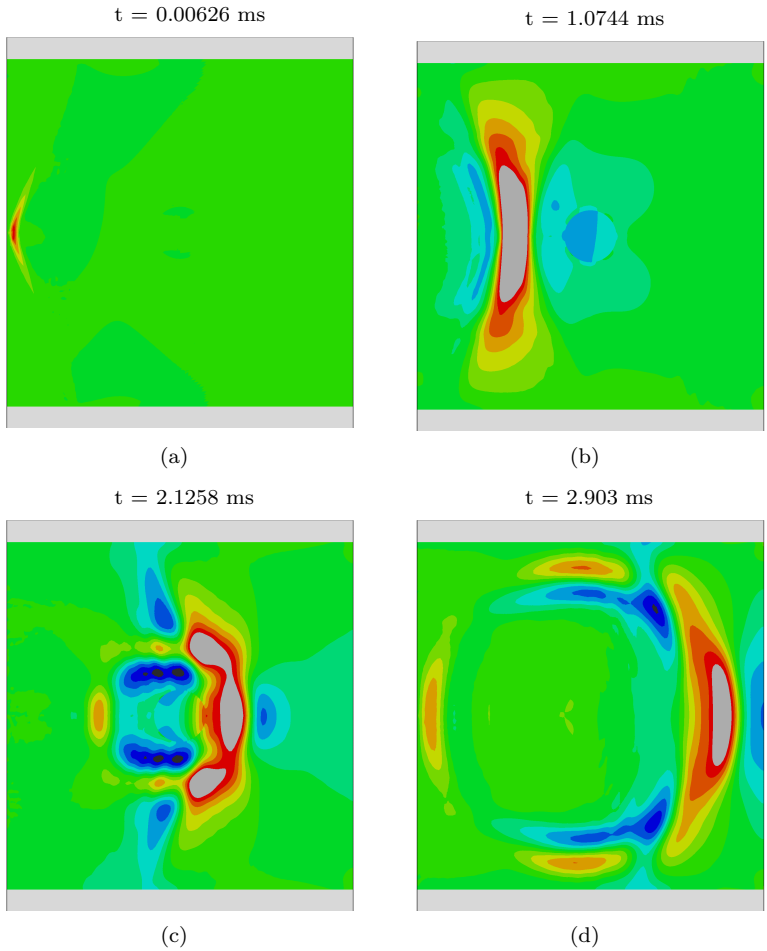


Figure 5.12: Field plots of σ_{12} at times $t = 0.00629$ ms, 1.0744 ms, 2.1258 ms and 2.903 ms. The light grey color indicates stress levels higher than 100 Pa, while the dark grey color indicates stress levels lower than -100 Pa. The tumor is circular with $R = 1.5$ mm and is located in the middle of the soft tissue section. The tumor may be seen ahead of the shear wave front in Figure 5.12(b).

Time to Peak Displacement

We extracted the temporal axial displacement profile ($u_2(t)$) in the focal point from the node set *FocalPoint*. The temporal axial displacement for the different soft tissue stiffness values are plotted in Figure 5.13. The red dots indicate the location of the peak displacement in each curve. Figure 5.14 shows a plot of the TTP displacement against the soft tissue stiffness. We note that the TTP displacement is reduced for increased values of $E^{(\text{tissue})}$.

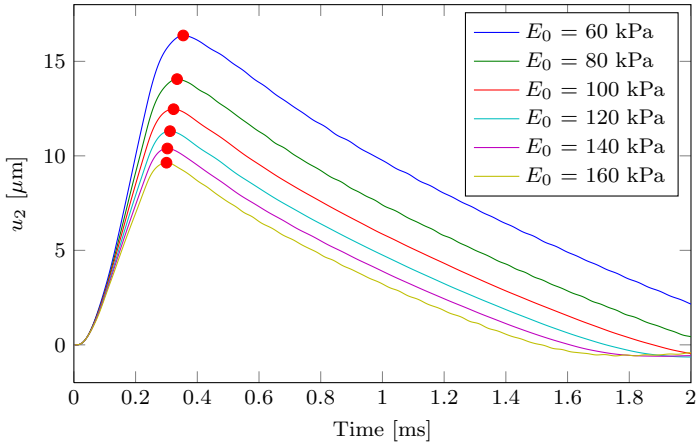


Figure 5.13: Axial displacement profile in the focal point for different soft tissue stiffness values. The red dots indicate the peak displacement. u_2 is given in μm, while the time is given in ms.

Figure 5.15 shows the temporal axial displacement profile for different impulse times (T_i), whereas Figure 5.16 shows the TTP displacement as a function of the impulse time. We see that the TTP displacement increases for increasing impulse times.

Figure 5.17 shows a field plot of the axial displacement for $E^{(\text{tissue})} = 100$ kPa and at time $t = 6.3 \cdot 10^{-5}$ s. The blue color indicates the focal region, where the axial displacement magnitude is largest. We note that only half of the modeled plate is shown in the field plot, since the relevant information is the axial displacement in the focal point. The location of the tumor is indicated by the half circle. The upper and lower boundary between the tissue and infinite sections are indicated by the black lines. The infinite elements are not shown in their full size in Figure 5.17.

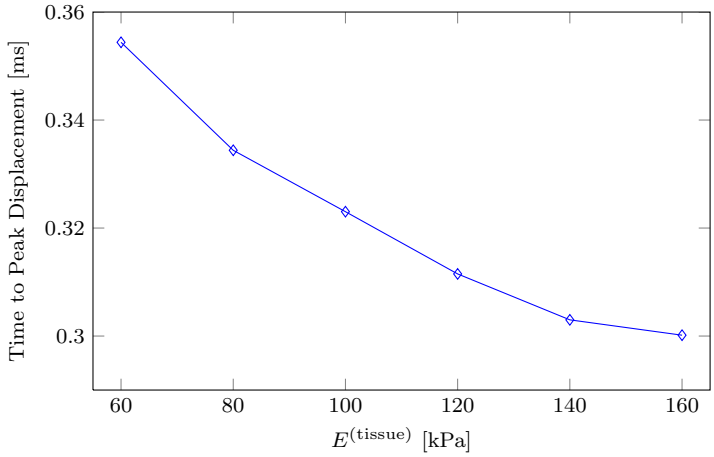


Figure 5.14: The TTP displacement is plotted as a function of soft tissue stiffness $E^{(\text{tissue})}$. TTP displacement is given in ms, while $E^{(\text{tissue})}$ is given in kilopascals. It is seen that TTP displacement decreases as the stiffness increases.

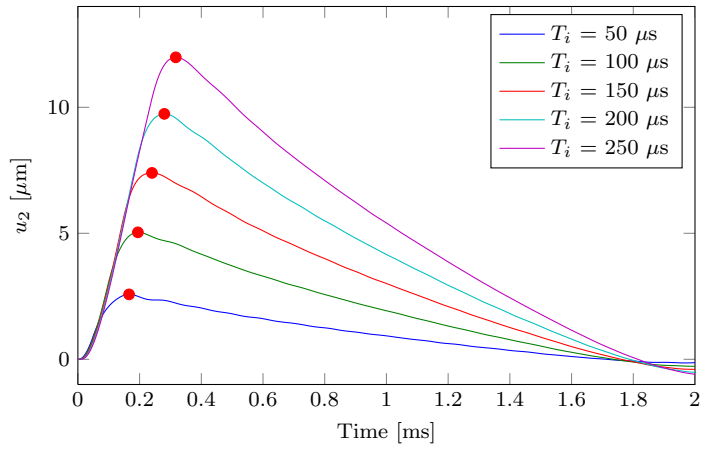


Figure 5.15: Axial displacement profile in the focal point for different impulse times (T_i). The red dots indicate the peak displacement. u_2 is given in μm, while the time is given in ms.

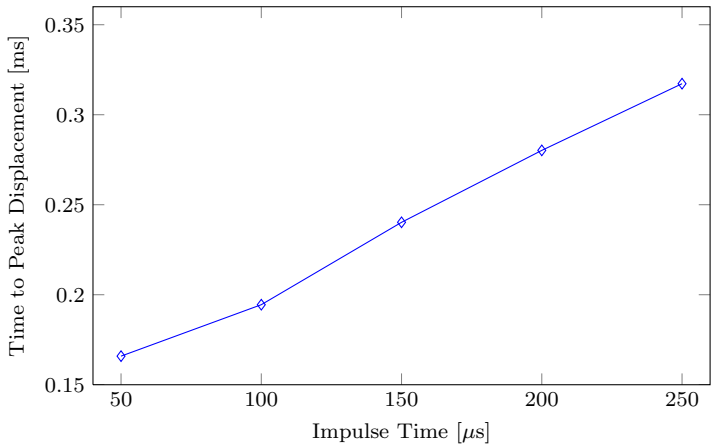


Figure 5.16: The TTP displacement is plotted as a function of impulse time (T_i). TTP displacement is given in ms, while impulse time is given in μs. It is seen that the TTP displacement increases as the impulse time increases.

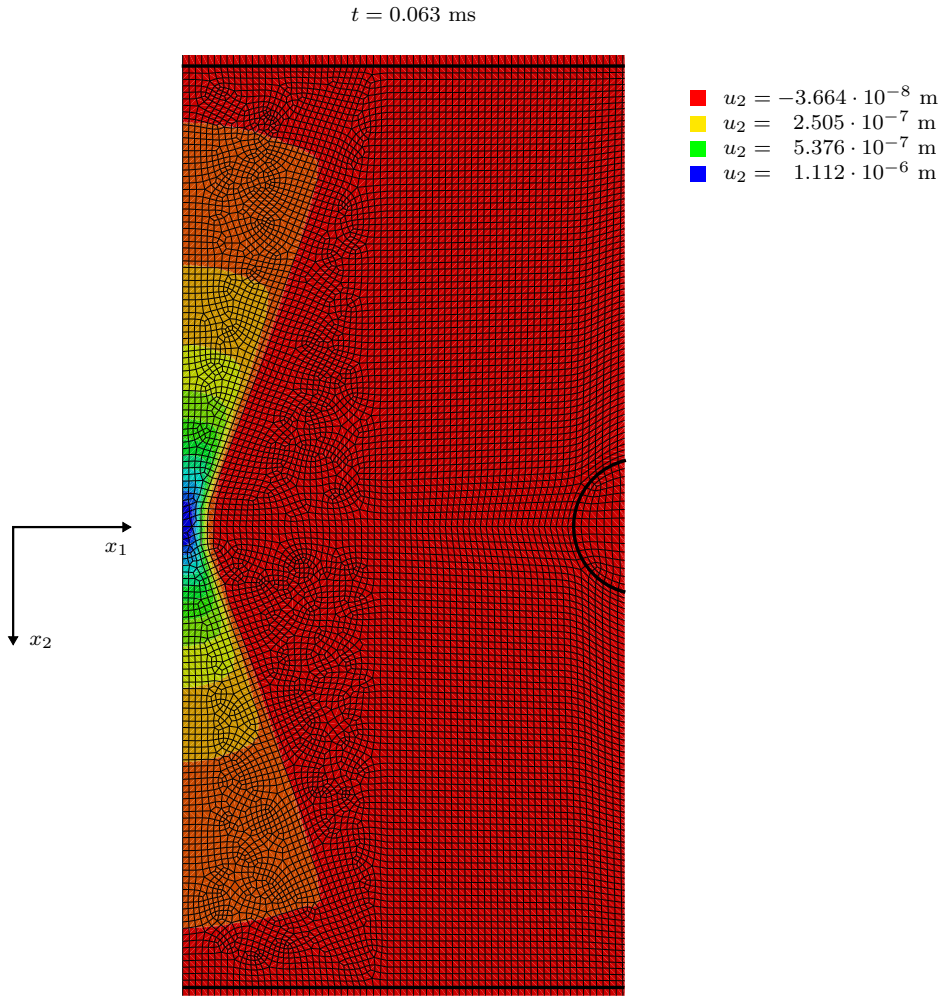


Figure 5.17: Field plot of the axial displacement in a region corresponding to half the plate. The tumor is indicated by the half circle to the right in the figure. $E^{(\text{tissue})} = 100$ kPa and $t = 6.3 \cdot 10^{-5}$ s. Blue color corresponds to large displacement magnitude and red color corresponds to small displacement magnitude.

5.3 Discussion

Modeling considerations

We modeled the soft tissue as a two-dimensional plane strain plate in Abaqus. We believe that this is a fair approximation of the physical three-dimensional gel-agar phantom since the imaging direction of the ultrasound transducer was in the thickness direction (D), referring to Figure 3.1, which we see is much smaller than both the width (W) and the length (L). Hence, the out-of-plane strains on a cross-section in the thickness direction may be neglected. In physical ultrasound examinations, the large amount of tissue surrounding the focal point reduces the mobility of the tissue normal to the imaging direction. Consequently, when we consider a cross-section along the axial direction of the transducer, we may assume that the out-of-plane strains are negligible since $u_1 \approx 0$ and $u_3 \approx 0$, while $u_2 \neq 0$.

We chose a Poisson's ratio of 0.499 as a trade-off between computational time and accuracy. The analysis run time is very sensitive to changes in Poisson's ratio when we approach incompressible conditions, $\nu = 0.5$. This is because the longitudinal (dilatational) wave velocity is highly susceptible to changes in Poisson's ratio for nearly incompressible materials, which can be seen from Equation 2.65. Palmeri et al. [25] found that the analysis run time was about 3.3 times longer when the value of ν changed from 0.499 to 0.4999. A Poisson's ratio of 0.499 does not support a longitudinal velocity of 1540 $\frac{\text{m}}{\text{s}}$, which is the approximate value of c_l for tissue. In order to obtain such a high value of c_l we would need $\nu \approx 0.49999$, according to Lee et al. [5]. However, we believe that the slight difference in ν has less impact on the FEA results, since the physics in the FEM model is dominated by shear waves. This is also argued by Palmeri et al. [12]. The shear wave speed (c_t) is minimally affected by this slight underestimation in ν, which can be seen from Equation 2.66. However, the peak displacement decreases as ν increases, as discussed in Palmeri et al. [25]. Hence, $\nu = 0.499$ causes slightly overestimated peak displacement values. This is shown in Figure 5.18, which is taken from Palmeri et al. [25].

We see from Table 3.2 that the relaxation times τ_n are quite large compared to the analysis time (T_a). This implies that viscoelasticity is not important in our FEM simulations. We may readily verify this from the shear stress profiles in Figures 5.7 and 5.8, which are seen to coincide. Consequently, we could have disregarded the viscoelastic material in the FEM simulations and still obtain the same results, since we used such small analysis times.

We note that we have applied the steady-state ARF field from Chapter 4.2 as an impulse load for a time (T_i). This is not entirely correct, since it would take some time for the ultrasound waves to reach the focal point and generate the desired ARF push. However, referring to Chapter 4.3, the ultrasound waves reach the focal point 6.5 μs after the ultrasound transducer was excited. Consequently, it takes 13 μs before the ultrasound waves reach the lower boundary of the model and to obtain a steady-state ARF in the modeled tissue. We assume that this travel time is sufficiently small to be neglected, compared to the impulse time of the ARF. Hence, we believe that the assumption of a time independent ARF field is representative of the physical time dependent ARF field

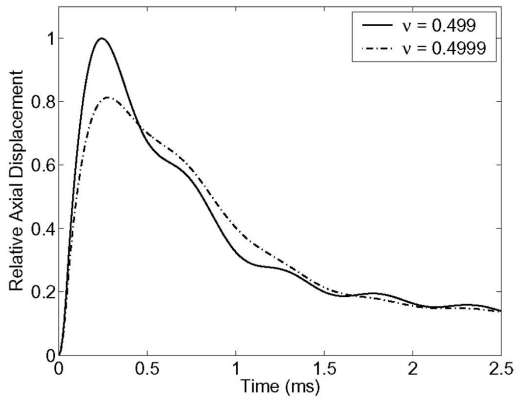


Figure 5.18: Figure 1 in Palmeri et al. [25]. Comparison of the simulated axial displacement at the focal point for $\nu = 0.499$ and $\nu = 0.4999$. The ARF was applied for 45 μs.

originating from a ultrasound transducer.

Even though the material behavior of soft tissues is generally non-linear, we have used linear elastic and linear viscoelastic material definitions in the FEM simulations. However, since the deformations caused by ultrasound waves are rather small, at least for medical applications, we assume that this non-linearity is negligible and that a linear material model will suffice.

Lastly, we could have used infinite elements on the right edge in Figure 5.1(a), in order to damp out reflections from this boundary. However, our interest lay in the reflections from the tumor and we did not use any FEA results that could have been affected by reflections from the right edge of the plate. Hence, we chose to disregard infinite elements on this edge and rather use finite elements for simpler implementation.

Mesh Dependency

The temporal shear stress and axial displacement distributions from the node sets *Mid_Reflection* and *Tumor_Boundary* were plotted in Figures 5.5 and 5.6. We observe that the differences between the various σ_{12}-profiles and u_2-profiles for different values of n are rather small. The axial displacement profiles are seen to coincide for almost all time values, except in the proximity of the displacement peaks. From Figures 5.5(b) and 5.6(b) we see that the maximum displacement is slightly larger for $n = 6$ and $n = 8$ than for the higher values of n. The trend is opposite for the preceding negative displacement peak, where the magnitude of the axial displacement is slightly smaller for the lower n-values than for the higher n-values. We also observe that the axial displacements are negative before the arrival of the shear wave, $t \approx 0.9$ ms in Figure 5.5(b) and $t \approx 1.7$ ms in Figure 5.6(b). We believe that this is due to volume preservation. As the focal region is pushed downwards in the positive axial direction, the tissue outside the focal region is pushed upwards in the negative axial direction. This is also discussed in Lee et al. [5].

We observe a larger discrepancy in the shear stress profiles than in the axial displacement profiles, as we can see in Figures 5.5(a) and 5.6(a). The shear stress tends to oscillate after the incident stress peak, and the amount and frequency of the oscillation are not the same for the different mesh densities. We believe the reason for this behavior is that the stresses are calculated from the displacement gradient, which can be seen from Equations 2.122 and 2.123. In general, errors are amplified during differentiation. A slight deviance between the axial displacement profiles may consequently cause a more pronounced deviance between the shear stress profiles.

Referring to Figure 5.5(a), we get some indications of diffusive effects in the stress peak at $t \approx 0.9$ ms, since the smaller values of n produce slightly wider shear stress waves. We observe the same for the stress peak at $t \approx 1.7$ ms in Figure 5.6(a). This could mean that the initial impulse shape is somewhat smeared out due to numerical diffusion. We also observe some numerical dispersion, which is indicated by an increased time difference between the incident stress peak and the reflected stress peak for decreasing values of n. This can be seen in Figure 5.5(a). The incident shear stress peaks for all values of n arrive at $t \approx 0.9$ ms. The reflected shear stress peak for $n = 14, 12$ and 10 are seen to arrive at $t \approx 2.4$ ms, while the arrival time increases slightly for the lower values of n. This indicates that numerical dispersion is more apparent when n decreases, hence when the element size increases. Numerical diffusion and dispersion effects are discussed in Marfurt [26], where it is argued that more accurate solutions are obtained as the mesh is refined. Also, it was mentioned that high frequency response ($\lambda < \Delta x$) is more affected by numerical diffusion and dispersion than low frequency response ($\lambda > \Delta x$). Since our impulse time was 0.25 ms, the dominant frequency is 4000 Hz, which gives an impulse length of $\lambda_I^{(\text{tissue})} = 1.45$ mm. This suggests that our simulations are of low frequency, according to the element sizes ($\Delta x^{(\text{tissue})}$) given in Table 5.1, and should not be greatly affected by numerical diffusion or dispersion. This is in agreement with the aforementioned observation of slight numerical diffusion and dispersion in our FEA results and that these errors increase as the element size increases.

Even though we have pointed out some minor differences between the various mesh densities, the simulated response curves are in general quite similar. Thus, we could probably have used either of the mesh densities and still obtain approximately the same accuracy. However, we chose $n = 10$ as the default mesh density. This value was also used in an example in the ABAQUS documentation [21]. Treating $n = 14$ as a reference curve, we see that both the amplitude and the arrival of the stress wave are in agreement for $n = 10$. Hence, the degree of numerical diffusion and dispersion for $n = 10$ and $n = 14$ should be on the same order. Also, $n = 10$ reduces the total number of elements compared to $n = 14$ and 12, which is beneficial for computational time.

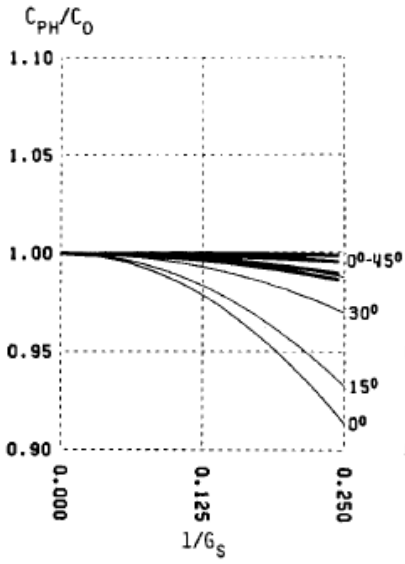


Figure 5.19: Adapted from Figure 9 in Marfurt [26]. The figure shows the numerical dispersion curves for the explicit finite-element method for different element angles. The ratio $\frac{c_{ph}}{C_0}$ is the ratio between numerical and analytical wave speed while $\frac{1}{G_S}$ is equivalent to $\frac{1}{n}$. Marfurt used $\nu = 0.4$ and square elements to obtain the results in this figure.

Wave Speed Study

The numerical shear wave speed in the soft tissue and tumor were presented in Table 5.4 along with the analytical values and the deviance between numerical and analytical shear wave speed. We see that the shear wave speed in the soft tissue is in accordance with the analytical solution, with an error of only $\epsilon \approx -0.15\%$. This slight deviance could be due to numerical dispersion, which is discussed in Marfurt [26]. Referring to Figure 5.19, taken from Marfurt, which shows numerical dispersion for square elements with $\nu = 0.4$, we see that the dispersion error is about 2 % for $n = 10$. This means that we get better results than what is expected. However, we cannot use Figure 5.19 directly in comparison with our results since numerical dispersion is dependent upon both the Poisson's ratio and the angle of the elements in the mesh, as argued in Marfurt. We used $\nu = 0.499$ whereas the results in Figure 9 in Marfurt stems from $\nu = 0.4$. Also, the elements in our FEM model are not uniformly oriented or entirely square, as opposed to those used in Marfurt.

If we compare the numerical shear wave speed in the soft tissue with the measured shear wave speed from the SSI experiment on the gel-agar phantom, the deviance is much more pronounced than for the analytical shear wave speed comparison. The focal depth in the FEM simulations was 10 mm and from Figure 5.3 we see that the experimental shear wave speed at this depth correspond to $c_t^{\text{exp}} \approx 5\frac{m}{s}$. This gives a deviance of

$$\epsilon = \frac{5.8098 - 5}{5} \cdot 100\,\% \approx 16\,\% \tag{5.9}$$

We believe this discrepancy stems from an incorrect estimate of E_0 in the viscoelastic material calibration from Chapter 3. A shear wave speed of 5 $\frac{\mathrm{m}}{\mathrm{s}}$ correspond to an instantaneous elastic stiffness of

$$E_0 = 2\rho_0(1+\nu_0)(c_t^{\exp})^2 = 2 \cdot 1060 \cdot (1 + 0.499) \cdot 5^2 = 79477\,\mathrm{Pa} \tag{5.10}$$

From the calibration we carried out in Chapter 3, we found $E_0 = 107585$ Pa, which is significantly larger. This indicates that our calibration was not entirely correct and that additional test data should have been used in order to obtain a better estimate of E_0, as previously discussed in Chapter 3.3.

Even though the shear wave speed in the soft tissue was accurately replicated in the FEM simulation for the estimated value of E_0, with an error of only -0.15 %, this is not the case for the shear wave speed in the tumor. Referring to Table 5.4, we see that the deviance between the numerical value and the analytical value is $\epsilon \approx -13.15$ %. We suspect this discrepancy has to do with the larger element size used in the tumor or with boundary effects from the boundary between the soft tissue and the tumor. In order to examine this a bit further, we ran two additional simulations. We used an elliptical shaped tumor with $R_2 = 5$ mm in the first simulation, in order to increase the most relevant dimension of the tumor. In the second simulation, we reduced the element size in the tumor ($\Delta x^{(\mathrm{tumor})}$) to match the element size of the tissue section ($\Delta x^{(\mathrm{tissue})}$). The shear stress in the node sets *Tumor_Boundary* and *Mid_Tumor* were extracted, since we were only concerned about the shear wave speed in the tumor section. The resulting shear wave speeds are given in Table 5.5.

From Table 5.5, we observe that reducing the element size in the tumor section has only minor effects on the numerical shear wave speed, since the deviance is reduced by an amount $\Delta\epsilon = 1.5$ % compared to the result in Table 5.4. From Equations 5.1 and 5.2, and using that

$$n^{(\mathrm{tumor})} = \frac{c_t^{(\mathrm{tumor})} \cdot T_i}{\Delta x^{(\mathrm{tumor})}} = \frac{\sqrt{3} c_t^{(\mathrm{tissue})} \cdot T_i}{\Delta x^{(\mathrm{tumor})}} \tag{5.11}$$

we may argue that as the element size in the tumor section is reduced by an amount $\frac{1}{1.3}$, the value of $n^{(\mathrm{tumor})}$ is increased by an amount of 1.3, from $n^{(\mathrm{tumor})} \approx 13.3$ to $n^{(\mathrm{tumor})} \approx 17.3$ when $n^{(\mathrm{tissue})} = 10$. Referring to Figure 5.19, we observe that the numerical dispersion decreases as n increases, however slowly for such large values of n. This could explain the minor improvement of the shear wave speed in the tumor when we reduce the element size.

Again referring to Table 5.5, we see that increasing R_2 has a much more pronounced effect on the calculated numerical shear wave speed than reducing the element size. We believe this is due to the ratio between the size of the tumor and the impulse length of the shear wave in the tumor. The impulse length in the tumor is given as

Table 5.5: The shear wave speed in the tumor when considering an elliptical shaped tumor with $R_2 = 5$ mm or using the same element size in both tissue and tumor. The deviance was found relative to the analytical solution, $c_t^{(\text{ana})} = 10.0778 \frac{\text{m}}{\text{s}}$.

Possible improvement	c_t $[\frac{\text{m}}{\text{s}}]$	Deviance [%]
Elliptical shaped tumor	10.2887	2.09
Smaller element size	8.9021	-11.66

$$\lambda_I^{(\text{tumor})} = c_t^{(\text{tumor})} \cdot T_i = \sqrt{3} \cdot \lambda_I^{(\text{tissue})} \tag{5.12}$$

The impulse length in the tumor is about 2.5 mm, which is nearly the same as the diameter of the circular tumor, $D = 3$ mm. When R_2 is increased, the shear wave propagation along the centerline of the tumor is less affected by boundary effects between the tumor and the soft tissue. One type of boundary effect is interface waves, also referred to as *Stoneley waves* for a solid-solid interface. These waves are guided along the material interface, and cause the shear wave to move slower close to the boundary between the soft tissue and the tumor. Since the distance from the boundary to the center of the tumor increases when R_2 is increased, the influence of Stoneley waves is reduced in the center of the tumor, which causes a higher shear wave speed. We will not go into further detail about interface waves, but we refer the reader to Rose [18] for more information. Also, both the angle of the elements and the shape of the elements are better for the elliptical inclusion, which in turn reduces the dispersion error, referring to Marfurt [26]. In Table 5.5, we see that the deviance is about 2 %, which is acceptable. However, the numerical shear wave speed is larger than the analytical. This is not what we expect, since the numerical dispersion for quadrilateral finite elements with lumped mass (CPE4R/Explicit) should lead to reduced wave speeds, which can be seen in Figure 5.20, taken from Belytschko et al. [27]. We are not aware of the reason for this, but it may be due to inaccurate measurements of the actual wave front. However, we have not yet figured out a better way to determine the shear wave front than to consider the shear stress peak, as indicated in Figure 5.7.

We have seen that the shear wave speed in the soft tissue can be accurately estimated by detecting the shear wave front at different locations within the soft tissue, and measuring the time it takes the shear wave to travel between these locations. Consequently, we believe that the shear wave speed can provide an accurate estimate of the elastic stiffness in a homogeneous tissue, using ultrasound to track the shear wave propagation. In the tumor, we observed that the estimated shear wave speed was highly dependent upon the shape and size of the tumor due to boundary effects from the interface between the soft tissue and the tumor. Thus, we cannot be certain that the shear wave speed reflects the true elastic stiffness of the tumor. However, if the focal point of the ultrasound transducer was located within the tumor, the shear wave speed might have been more accurately estimated since boundary effects probably would have been less pronounced. Although, this could be difficult in reality since the shear wave speed in the tumor is relatively high and the size of the tumor is quite small, which would require a very high Pulse Repetition

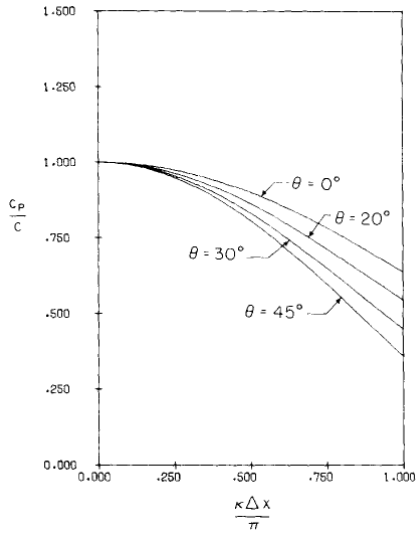


Figure 5.20: Figure 4 in Belytschko et al. [27]. The figure shows the numerical dispersion of underintegrated bilinear quadrilateral element with lumped mass as a function of the relative wave number. The ratio $\frac{c_p}{c_0}$ is the ratio between numerical and analytical wave speeds and the relative wave number corresponds to $\frac{1}{n}$.

Frequency (PRF) of the transducer in order to track the shear wave propagation.

Reflection Study

From Figure 5.10 we observe that the discrepancy between the plane shear wave reflection factor from Equation 2.120 and the numerical values obtained from the FEA results is prominent. The numerical reflection factors are much lower than the analytical values. However, this comes as no surprise due to the circular shape of the tumor, which causes a non-plane boundary between the soft tissue and the tumor. From the field plots in Figure 5.12, we readily observe that the incident shear wave is reflected in many directions, referred to as *scattering* in Chapter 2.1.1. Hence, the amount of the reflections that actually reach back to the node *Mid_Reflection* is much smaller than in the case of a plane material boundary. This causes reduced reflection factors. From Figure 5.10(b), we also see that the deviance decreases as the stiffness ratio between the tumor and the soft tissue increases. We believe the reason for this is that larger stiffness ratios increase the amplitude of the reflected shear wave, which makes the numerical reflection factor less susceptible to small differences in the reference shear stress value, indicated by the magenta dot in Figure 5.9.

Figure 5.11 shows the effects of increasing the value of R_2 relative to R_1, thus making the material boundary facing the Region of Excitation (ROE) more plane. We observe that the deviance in Figure 5.11(b) is reduced from $\epsilon \approx 40\%$ to $\epsilon \approx 10\%$ as the ratio between R_2

and R_1 is increased from 1 to 5. As $\frac{R_2}{R_1}$ increases, the boundary facing the ROE gradually becomes more plane, which reduces the scattering of the incident wave and increases the magnitude of the reflected wave. However, we observe that the deviance is not reduced to zero, which indicates that the plane shear wave reflection theory is never valid. This could be due to several reasons. Firstly, even though we increased $\frac{R_2}{R_1}$ to a maximum value of 9, the material boundary is still not entirely plane. Secondly, we must keep in mind that the shear wave generated by the ARF is not truly plane, since it originates from a small region. Lastly, we are not certain that the reference shear stress values used to determine the numerical reflection factors are correct, which could affect the deviance. Also, we observe that the lowest deviance is obtained for $\frac{R_2}{R_1} = 5$. This is rather counter-intuitive, since the larger radii ratios should approximate a plane material boundary even better. We find it difficult to point out any obvious reasons for this occurrence, but it might be due to mesh effects since the mesh changed slightly for the different radii ratios.

The tumors in real life are usually irregularly shaped and the assumption of plane material boundaries is not valid. This makes it difficult to determine the stiffness of the tumor based on the reflected shear wave, which reduces the reliability of the method. For instance, if we would estimate the tumor stiffness based on the numerical reflection factor for a stiffness ratio of 2 from Figure 5.10, we would obtain

$$R^{\text{num}} = 0.0709 \quad \Rightarrow \quad \frac{E^{(\text{tumor})}}{E^{(\text{tissue})}} \approx 1.1526 \tag{5.13}$$

while the true value is $\sqrt{3}$. This gives an error of $\epsilon \approx -33.5\%$.

Another limitation of this method is due to the definition of the boundary between the soft tissue and the tumor. We have assumed a discrete material boundary in both the FEM model and in the derivation of the analytical shear wave reflection factor in Equation 2.120. However, it is more likely that the stiffness changes more gradually for real life tumors. This reduces the impedance mismatch, which in turn causes the reflection magnitudes to decrease. Thus, the simple analytical relation in Equation 2.120 can no longer be used to determine the stiffness of the tumor, regardless of whether the material boundaries are plane or not. This is discussed in Palmeri et al. [12]. Also, since the reflection factor only provides information about the relative stiffness difference between the soft tissue and the tumor, we must know the stiffness of the soft tissue in advance, in order to estimate the stiffness of the tumor. However, reflection waves give qualitative information of inclusions within the soft tissue and may still be appropriate to determine the approximate shape of a potential tumor, even though the exact stiffness is difficult to predict.

TTP Displacement Study

Figure 5.13 shows the axial displacement profile in the node set *FocalPoint* for different values of E_0. We observe that the magnitude of the axial displacement decreases as the elastic stiffness of the soft tissue increases. This is because the applied body force remains constant in all simulations, while the counteracting elastic forces increase due to increasing elastic stiffness. Based on these observations, we may think that axial displacement

magnitudes alone could be used to estimate the elastic stiffness. However, as indicated by Figure 5.21 taken from Palmeri et al. [12], the axial displacement profiles estimated by ultrasound are affected by considerable jitter and underestimation of displacement magnitudes. This reduces the utility of using displacement magnitudes to estimate the elastic stiffness of soft tissues. However, the relative difference between estimated displacement magnitudes still reflects elastic stiffness differences within the tissue. Hence, we are still able to image a heterogeneous tissue by measuring the displacement magnitudes with ultrasound, even if exact determination of the elastic stiffness properties is not feasible.

From Figures 5.13 and 5.14, we readily observe that the TTP displacement decreases as the stiffness of the tissue increases. We also see that the decay of the curve in Figure 5.14 indicates that the TTP displacement is inversely related to the stiffness, as proposed in Chapter 5.1. Based on these observations, TTP displacement appears as an adequate measure of elastic stiffness properties, which is also suggested by Sarvazyan et al. [7]. The TTP displacement is independent of displacement magnitude and consequently also independent of the applied force, as discussed in Palmeri et al. [12]. This is advantageous in medical applications since the magnitude of the ARF is difficult to determine due to unknown attenuation in the tissue.

Even though the TTP displacement seems to be a promising measure of stiffness, there are some additional aspects that have not yet been brought to light. Firstly, the impulse time and the focal configuration ($F_\#$) have an impact on the TTP displacement. From Figures 5.15 and 5.16, we observe that the TTP displacement increases as the impulse time (T_i) increases. This is in accordance with what was found in Palmeri et al. [12]. Also, in Figure 5.22 from Palmeri et al. [12], the TTP displacement is seen to increase for larger values of $F_\#$. Both T_i and $F_\#$ are chosen by the person who performs the ultrasound examination, and thus it is very important that these effects are accounted for. Secondly, both the density and the Poisson's ratio has an impact on the TTP displacement. In Figure 5.23 from Palmeri et al. [12], we see that the TTP displacement increases almost proportionally to the increase in density. From Figure 5.18, taken from Palmeri et al. [25], we observe that the TTP displacement is dependent upon Poisson's ratio. The TTP displacement is seen to increase slightly as ν is changed from 0.499 to 0.4999. Thus, both density and Poisson's ratio affects the TTP displacement in an opposite manner to that of elasticity. However, if we compare the elastic properties of tumors and healthy tissues, we presume that the differences in density and Poisson's ratio are rather small compared to the difference in stiffness. Thus, these effects are assumed negligible compared to those of the elastic stiffness. Lastly, limitations in the Pulse Repetition Frequency (PRF) can make experimental stiffness estimation based on the TTP displacement difficult. Since the TTP displacement values are rather short, we would need very high PRFs in order to detect the displacement peak. From Figure 5.16, we see that the TTP displacement can be as low as 0.15 ms. This means that the PRF must be substantially larger than 6667 Hz, in order to obtain sufficient resolution to detect the displacement peak.

In order to use the TTP displacement to estimate the stiffness of the tumor, we need to excite the tumor directly by locating the focal point inside the tumor. However, measuring the TTP displacement might be difficult. Since the stiffness of the tumor is higher than that of the soft tissue, we would need even higher PRFs in order to detect the peak

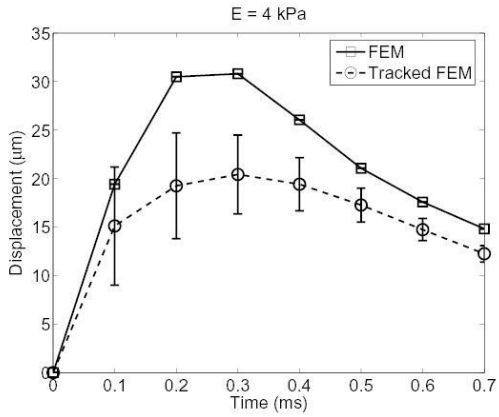


Figure 5.21: Figure 3 in Palmeri et al. [12]. Comparison of the simulated axial displacement at the focal point for FEM and tracked FEM data. The data correspond to a PRF of 10 kHz.

displacement. Also, since the size of the tumor is relatively small, shear wave reflections from the tumor boundary could possibly reach back to the ROE and cause constructive interference. This would make it difficult to separate the actual displacement peak from the reflected shear wave peak, and could result in erroneous values of the TTP displacement. This is discussed in Palmeri et al. [12].

Even though Equation 5.7 provides an analytical expression to calculate the shear modulus (G) based on the TTP displacement ($t_{\max}$), this expression is only valid for a perfectly Gaussian ultrasound beam. From Figure 4.3(a), we clearly see that this is not the case. As argued in Bercoff et al. [9], techniques that rely on the measurement of the axial displacement in the focal point are not quantitative when the ultrasound beam is not perfectly Gaussian. Hence, we cannot use the TTP displacement to provide reliable estimates of the stiffness in neither the soft tissue nor the tumor, since the ultrasound beams are rarely perfect Gaussian. Even if they were, the TTP displacement would not have been well-suited to estimate the stiffness of the tumor, due to the limited PRF values and the possibility of constructive interference from the shear wave reflections.

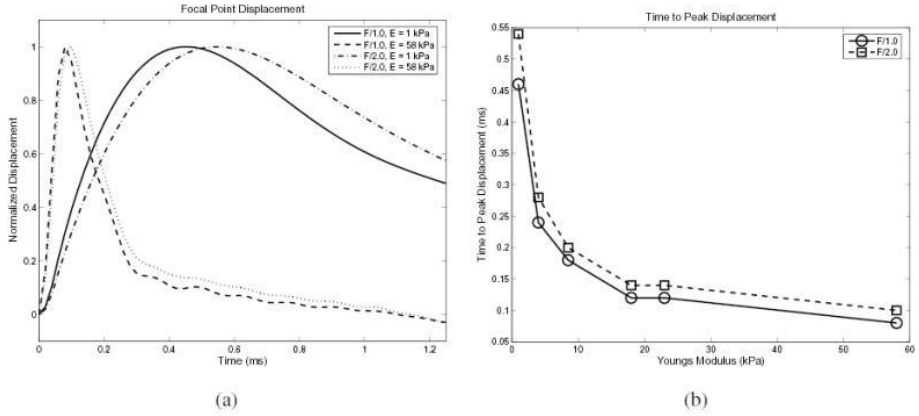


Figure 5.22: Figure 4 in Palmeri et al. [12]. Figure A shows normalized axial displacement profiles for two different stiffness values (1 kPa and 58 kPa) and two different focal configurations ($F_{\#} = 1$ and $F_{\#} = 2$). Figure B shows the TTP displacement as a function of tissue stiffness for both focal configurations.

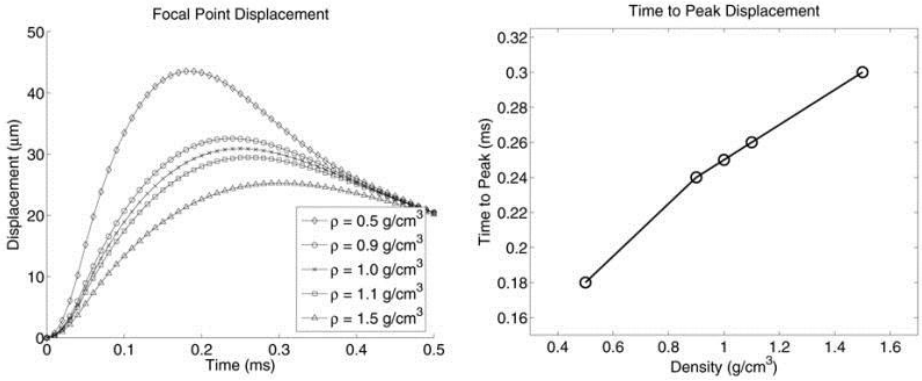


Figure 5.23: Figure 5 in Palmeri et al. [12]. Figure A shows the axial displacement profiles at the focal point for different density values, ρ. The elastic stiffness was 4 kPa. Figure B shows the TTP displacement at the focal point as a function of density.

6 Concluding Remarks

The work conducted in this thesis has been concerned with the use of FEM to simulate a locally induced axial displacement in a soft tissue by the use of an ARF impulse. Due to the impulse load and the restorative forces of the tissue, the axial displacement results in a shear wave that propagates away from the ROE. Many modern elastography methods relies on the use of locally induced displacements and shear waves to determine the elastic properties of soft tissue and to locate possible tumors within the tissue. Even though relative stiffness difference can be used for imaging, it is important to quantify elastic stiffness because malignant tumors are in general stiffer than benign tumors. We have carried out FEM analyses that can be used to verify the accuracy of stiffness estimation procedures based on shear wave speed, shear wave reflection and TTP displacement.

The stress relaxation behavior of the viscoelastic tissue-mimicking gel-agar phantom was accurately described by a three-element Maxwell model. However, from the FEM simulation of the stress relaxation test we saw that the uniaxial stress was slightly underestimated compared to the experimental data. We assumed that this was related to the use of a prescribed initial stress in the FEM model, inaccurate estimate of E_0 or inaccurate displacement and force measurements in the experiment. We also argued that more experimental test data should have been included in order to increase the reliability of the estimated material parameters E_0, g_n, and τ_n.

We have been able to simulate the ARF field originating from a linear array transducer and shown how this ARF field can be implemented as a body force in an FEM model by the use of a mathematical function. The ARF field was well represented by a three-element Gaussian function.

We obtained similar FEA results for both the elastic and the viscoelastic material definitions. Thus, the FEM simulations were unaffected by the relaxation behavior of the calibrated three-element Maxwell model. We argued that this was due to the large relaxation times (τ_n) found from the material calibration in Chapter 3.2 and the relatively short analysis time (T_a).

The shear wave speed was accurately represented in the soft tissue compared to the analytical value for the estimated E_0. However, we observed a discrepancy between the numerical shear wave speed and the experimental shear wave speed found from the SSI experiment. This deviance was assumed to stem from inaccurate estimation of E_0 in Chapter 3, and suggests that the estimated value of E_0 from the relaxation test is too

large. The numerical shear wave speed in the tumor was dependent upon the size and shape of the tumor. In the case of a circular tumor with $R = 1.5$ mm, we saw that the numerical shear wave speed was underestimated by 13.15 %. We argued that this was due to boundary effects, since the impulse length (λ_I) was comparable to the diameter of the tumor. When we increased R_2 relative to R_1, we saw that the numerical shear wave speed was more in accordance with the analytical value. Since the shear wave speed of the tumor is dependent upon the size and shape of the tumor, it may be difficult to use this to determine the elastic stiffness. A possible remedy is to excite the tumor directly, but this has not been investigated.

We have seen that the shear wave reflection factor from the FEA is not suited to determine the proper stiffness ratio between the soft tissue and a tumor. Since the boundaries between soft tissues and tumors are rarely plane, and the shear wave caused by the ARF is not entirely plane, the reflection factors determined from the reflected shear wave cannot be used to estimate the stiffness of the tumor. This method also requires that we know the stiffness of the soft tissue in advance, in order to estimate the stiffness of the tumor. However, as indicated in Chapter 5.3, the reflected wave may still be used to image an inclusion within the soft tissue and to determine the shape of the inclusion.

We have seen that the TTP displacement is inversely related to the elastic stiffness of the soft tissue. The TTP displacement is independent of the applied force, which is beneficial in medical imaging where the exact value of the load magnitude is difficult to determine, due to the unknown attenuation. However, the TTP displacement must be used cautiously because factors such as impulse time and transducer size will affect the TTP displacement values and may cause erroneous estimates of the elastic stiffness. Also, the TTP displacement within a tumor can provide false information because of reflections from the material boundary surrounding the ROE and limited PRFs. Lastly, the TTP displacement can only be linked to the shear modulus for a perfect Gaussian ultrasound beam, which is rarely the case.

7 Further Work

We have experienced that much of the work carried out in this thesis can be improved and taken one step further. Also, we have used some assumptions that should be examined in more detail. We will end this thesis by pointing out some improvements that could be made and some assumptions that should be checked:

- More experimental data from stress relaxation and creep tests should be used in the calibration, such that the elastic stiffness (E_0) is estimated more accurately.
- Carry out transient pressure field simulations and check the validity of the continuous wave assumption.
- Make a three-dimensional FEM model with a spherical inclusion inside and check the validity of the plane strain approximation.
- Make an FEM model with the possibility to place the focal point inside the tumor, such that the shear wave speed of the tumor can be estimated with the least amount of boundary effects. Also, this model could be used to see to what extent the TTP displacement is affected by shear wave reflections.
- Use differently shaped inclusions. Malignant tumors are rarely smooth, such that it might be of interest to examine inclusions that are unevenly shaped.

References

[1] Ahmedin Jemal, Freddie Bray, Melissa M. Center, Jacques Ferlay, Elizabeth Ward, and David Forman. Global cancer statistics. *CA: A Cancer Journal for Clinicians*, 2011.

[2] Elisa E. Konofagou, Tim Harrigan, and Jonathan Ophir. Shear strain estimation and lesion mobility assessment in elastography. *Ultrasonics*, 2000.

[3] Peter N.T Wells and Hai-Dong Liang. Medical ultrasound: Imaging of soft tissue strain and elasticity. *Journal of the Royal Society Interface*, 2011.

[4] Daniel T. Ginat, Stamatia V. Destounis, Richard G. Barr, Benjamin Castaneda, John G. Strang, and Deborah J. Rubens. Us elastography of breast and prostate lesions. *The journal of continuing medical education in radiology*, 2009.

[5] Kristen H. Lee, Benjamin A. Szajewski, Zaegyoo Hah, Kevin J. Parker, and Antoinette M. Maniatty. Modeling shear waves through a viscoelastic medium induced by acoustic radiation force. *International Journal for Numerical Methods in Biomedical Engineering*, 2012.

[6] SonoWorld. http://sonoworld.com/Client/MiniSites/MiniSiteDetails.aspx?MiniSiteId=1.

[7] Armen P. Sarvazyan, Oleg V. Rudenko, Scott D. Swanson, J. Brian Fowlkes, and Stanislav Y. Emelianov. Shear wave elasticity imaging: A new ultrasonic technology of medical diagnostics. *Ultrasound in Medicine and Biology*, 1998.

[8] Kathryn Nightingale, Mary Scott Soo, Roger Nightingale, and Gregg Trahey. Acoustic radiation force imaging: In vivo demonstration of clinical feasibility. *Ultrasound in Medicine and Biology*, 2001.

[9] Jéremy Bercoff, Mickaël Tanter, and Mathias Fink. Supersonic shear imaging: A new technique for soft tissue elasticity mapping. *IEEE Trans Ultrason Ferroelectr Freq Control*, 2004.

[10] Jeremy Bercoff. Shear wave elastography. 2008.

[11] Elizabeth S. Burnside, Timothy J. Hall, Amy M. Sommer, Gina K. Hesley, Gale A. Sisney, William E. Svensson, Jason P. Fine, Jinfeng Jiang, and Nicholas J. Hangiandreou. Differentiating benign from malignant solid breast masses with us strain imaging. *Radiology*, 2007.

[12] Mark L. Palmeri, Stephen A. McAleavey, Kelly L. Fong, Gregg E. Trahey, and Kathryn R. Nightingale. Dynamic mechanical response of elastic spherical inclusions to impulsive acoustic radiation force excitation. *IEEE Trans Ultrason Ferroelectr Freq Control*, 2006.

[13] Robert J. McGough. *FOCUS Documentation*. Michigan State University, 2013.

[14] Institut Langevin. http://www.institut-langevin.espci.fr/Langevin-Institute.

[15] William R. Hendee and E. Russell Ritenour. *Medical Imaging Physics*. Wiley-Liss, Inc., 2002.

[16] Abigail Swillens. *A Multiphysics Model for Improving the Ultrasonic Assessment of Large Arteries*. PhD thesis, UGent, 2009-2010.

[17] O.S.Hopperstad and T.Børvik. *Material Mechanics*. 2013.

[18] Joseph L. Rose. *Ultrasonic Waves in Solid Media*. The Press Syndicate of the University of Cambridge, University of Cambridge, Cambridge, UK, 1999.

[19] Karl F. Graff. *Wave Motion in Elastic Solids*. Dover Publications, Inc., New York, 1991.

[20] Fritjov Irgens. *Continuum Mechanics*. Springer-Verlag Berlin Heidelberg, Fridtjov Irgens, Wolffsgate 12, 5006 Bergen, Norway, 2008.

[21] SIMULIA. *Abaqus 6.11 Documentation*, 2011.

[22] Robert D. Cook, David S. Malkus, Michael E. Plesha, and Robert J. Witt. *Concepts and applications of finite element analysis*. John Wiley & Sons, Inc, University of Wisconsin - Madison, 2002.

[23] D. Chen and R. J. McGough. A 2d fast near-field method for calculating near-field pressures generated by apodized rectangular pistons. *Journal of the Acoustical Society of America*, 124(5):1526–1537, 2008.

[24] SupersonicImagine. http://www.supersonicimagine.com, 2012.

[25] Mark L. Palmeri, Amy C. Sharma, Richard R. Bouchard, Roger W. Nightingale, and Kathryn R. Nightingale. A finite-element method model of soft tissue response to impulsive acoustic radiation force. *IEEE Trans Ultrason Ferroelectr Freq Control*, 2005.

[26] Kurt J. Marfurt. Accuracy of finite-difference and finite-element modeling of the scalar and elastic wave equations. *Geophysics*, Vol. 49.

[27] Robert Mullen and Ted Belytschko. Dispersion analysis of finite element semidiscretizations of the two-dimensional wave equation. *International Journal for Numerical Methods in Engineering*, Vol. 18.

A MATLAB scripts

- *experimentalDataRetriever.m* was used to extract experimental data from a spreadsheet and calculate uniaxial strain and stress and the elastic modulus. Further, we used this script to make plots for comparison between experimental data and the analysis in Abaqus.
- *twoDimLinarray.m* was used to calculate the pressure field generated by a linear array ultrasound transducer. We also used it to plot pressure, intensity and radiation force fields.
- *createFit.m* was used to create the curve fit of the three-element Gaussian function to the radiation force distribution.
- *meshStudy.m* was used to load and plot the temporal shear stress and axial displacement distributions from the FEA results in the mesh study.
- *waveSpeeds.m* was used to determine the wave speed from the temporal shear stress distribution extracted from the FEA results in the wave speed study.
- *extractData.m* was used to load .dat-files from the FEA results.
- *reflectionStudy.m* was used to determine the shear stress peaks and the numerical reflection from FEA results in the reflection study. We also used it to plot the shear stress, reflection factor comparison and the deviance.
- *TTP.m* was used to determine the TTP displacement for different values of E_0. We also used it to plot the TTP displacement and the temporal axial displacement distribution.

A.1 experimentalDataRetriever.m

```
1 % Writen by: Lars Edvard Bryhni Daehli
2 % ------------------------------------------------------------------
3 % Script that imports the excel files with experimental data
4 % and plots relevant graphs.
5 % Will also perform least-square estimation of the viscoelastic
6 % material model.
```

```
7  % ------------------------------------------------------------------
8
9  clear all; clc; clf;
10
11 path = 'D:\NTNU\Master\Excel\';
12 name = 'Relaxation_Data_ThinPhantom.xlsx';
13 file = fullfile(path,name);
14
15 [N,T,R] = xlsread(file);
16
17 %The first 20 rows does not contain any relevant information. We will
18 %leave these rows out.
19
20 N = N(20:end,:);
21
22 %Time, extension and force is found as column 2,3 and 4.
23
24 t = N(:,1);
25 d = N(:,2);
26 f = N(:,3);
27
28 %Initial values of the test specimen
29 thickness = 3.5e-03;
30 width = 57.4e-03;
31 L0 = 63.97;
32 A0 = thickness*width;
33
34 %Experimental stress and strain values
35 strain_e = d./L0;
36 strain_l = log(1+strain_e);
37 stress_e = f./A0;
38 stress_t = stress_e.*exp(strain_l);
39
40 %Stress and strain values from the (simple support) Abaqus analysis
41 %based on the relaxation coefficients estimated by Abaqus
42 t_aba = N(:,16);
43 force_aba = N(:,17);
44 strain_aba = N(:,18);
45 stress_aba = N(:,19);
46
47 % Plotting the experimental data and the results from the analyses
48
49 %% Force-time plot
50
51 figure(1)
52 plot(t,f,'b')
53 xlabel('Time [s]')
54 ylabel('Force [N]')
55 %legend('Force data','Location','SouthEast')
56 %title('Force-Time')
57 axis([0 350 0 1.3])
58 matlab2tikz('force_time_experiment.tikz','height','\figureheight',...
59     'width','\figurewidth','showInfo', false)
60
61 %% Displacement-time plot
62
63 figure(2)
```

```
64 plot(t,d)
65 xlabel('Time [s]')
66 ylabel('Extension [mm]')
67 %legend('Extension data','Location','SouthEast')
68 %title('Extension-Time')
69 matlab2tikz('extension_time_experiment.tikz','height','\figureheight',...
       ...
70     'width','\figurewidth','showInfo', false)
71
72 %Determine the temporal elasticity based on E(t)=s(t)-s1/(e(t)-e1)
73
74 for i = 2:length(f)
75     E(i) = (stress_t(i)-stress_t(1))/(strain_l(i)-strain_l(1));
76 end
77
78 %Finding the onset of stress relaxation
79 for i = 1:length(d)
80     if d(i+1) == d(i)
81         i_relax = i;
82         break
83     end
84 end
85
86 %Defining the instantaneous elastic modulus as E(t) when the stress
87 %relaxation phase begin.
88 E0 = E(i_relax);
89 t_relax = t(i_relax);
90
91 %Initialize the shear modulus ratio (relaxation function)
92 nn = length(d)-i_relax;
93 g = zeros(N,1);
94 t_red = g;
95
96 %Determine the relaxation function values from the experimental data ...
       set
97 for n = 1:nn
98     j = (i_relax-1)+n;
99     g(n) = E(j)/E0;
100    t_red(n) = t(j)-t_relax;
101 end
102
103 %% Plot relaxation function
104
105 figure(3)
106 plot(t_red,g)
107 xlabel('Time [s]')
108 ylabel('g(t) = $\frac{G(t)}{G_0}$')
109 %legend('Relaxation data','Location','NorthEast')
110 %title('Shear modulus ratio')
111 matlab2tikz('relaxation_experiment.tikz','height','\figureheight',...
112     'width','\figurewidth','showInfo', false,'parseStrings',false)
113
114 %The model to be fitted is a Maxwell model of N elements. N can be ...
       set to
115 %a number of different integers. Remove the % ahead of cftool to ...
       display
116 %the curve fit and make any adjustments.
```

```
117
118 %cftool('two_element_fit.sfit')
119
120 %Find the non-zero entries of the "Abaqus vectors"
121 t_aba = t_aba(1:82);
122 strain_aba = strain_aba(1:82);
123 stress_aba = stress_aba(1:82);
124 force_aba = force_aba(1:82);
125
126 s_aba = stress_aba/1000;
127 s_t = stress_t/1000;
128
129 %% Plot stress comparison
130
131 figure(4)
132 plot(t,s_t,'b-',t_aba,s_aba,'r-.')
133 xlabel('Time [s]')
134 ylabel('$\sigma$ [kPa]')
135 legend('Experimental data','Simulation','Location','SouthEast')
136 %title('Stress comparison')
137 axis([0 350 0 7])
138 matlab2tikz('stress_comparison.tikz','height','\figureheight',...
139     'width','\figurewidth','showInfo', false,'parseStrings',false)
140
141 %% Plot force comparison
142
143 figure(5)
144 plot(t,f,'b-',t_aba,force_aba,'r-.')
145 xlabel('Time [s]')
146 ylabel('Force [N]')
147 legend('Experimental data','Simulation','Location','SouthEast')
148 %title('Force comparison')
149 matlab2tikz('force_comparison.tikz','height','\figureheight',...
150     'width','\figurewidth','showInfo', false)
151
152 %% Check the percentage difference between simulation and experiment
153
154 for i = 1:length(t_aba)
155
156     tol = 0.03;   %How large +/- tolerance between t_aba(i+1) and ...
        t_aba(i)
157     s = 0;
158     teller = 0;
159
160     %Average of experimental stress values close to Abaqus times
161     for j = 1:length(t)
162         if t(j)≤ (t_aba(i)+tol) && t(j) ≥ (t_aba(i)-tol)
163            s = s+stress_t(j)
164            teller = teller+1;
165         end
166     end
167         if s>0
168             S(i)= s/teller;
169             s_diff(i) = 100*abs((stress_aba(i)-S(i))/S(i));
170             ls_diff(i) = log(s_diff(i));
171             time(i) = t_aba(i);
172         end
```

```
173
174
175 end
176
177 %% Plot error between Abaqus and experiment
178
179 figure(6)
180 plot(time,ls_diff,'.')
181 xlabel('Time [s]')
182 ylabel(...
183 'ln($100\cdot\lvert\dfrac{\sigma_{aba}-\sigma_{exp}}{\sigma_{exp}}\...
    rvert$)')
184 %legend('Relative logarithmic error')
185 %title('Relative error between experiment and Abaqus analysis')
186 matlab2tikz('deviance_calibration.tikz','height','\figureheight',...
187     'width','\figurewidth','showInfo', false,'parseStrings',false)
188
189 %% Plot the least-square estimation for g(t) and the "real" g(t)
190
191 g1 = 0.00945749;
192 g2 = 0.0384291;
193 g3 = 0.0646134;
194 tau1 = 1.5178;
195 tau2 = 11.541;
196 tau3 = 93.977;
197 g_est = zeros(size(t_red));
198
199 for i = 1:length(t_red)
200     g_est(i) = 1-g1*(1-exp(-(t_red(i)/tau1)))-...
201         g2*(1-exp(-(t_red(i)/tau2)))-g3*(1-exp(-(t_red(i)/tau3)));
202 end
203
204 figure(7)
205 plot(t_red,g,'r',t_red,g_est,'b-.')
206 %title('Relaxation function comparison')
207 xlabel('Time [s]')
208 ylabel('g(t)')
209 legend('Experimental data','Prony Series','Location','NorthEast')
210 matlab2tikz('relaxation_comparison.tikz','height','\figureheight',...
211     'width','\figurewidth','showInfo', false,'parseStrings',false)
```

A.2 twoDimLinarray.m

```
1 % Written by: Lars Edvard Bryhni Daehli
2 % -------------------------------------------------------------------
3 % Script that generates a pressure field in a plate of similar
4 % dimension as that of the FEM-simulation for a 2D-plate with
5 % infinite elements. Extract the pressure field to calculate
6 % the intensity field values afterwards such that the force
7 % field can be calculated.
8 % -------------------------------------------------------------------
9 clear all; clf; clc;
```

```
10 cd('D:\NTNU\Master\Matlab\Ultrasound_Simulations');
11
12 %Total dimensions of plate given in [m]
13 x_total = 20e-3;
14 y_total = 2e-3;
15 z_total = 20e-3;
16
17 % Use apodization if apod = 1
18 apod = 1;
19 % Perform curve fit if fit = 1
20 fit = 0;
21
22 %Values from Abigail regarding transducer size
23 pitch = 200e-6;                %Total width of both elements and
24 kerf = 30e-6;                  %Spacing between elements, kerf
25 w = pitch-kerf;                %Width of elements [m]
26 h = y_total;                   %Height of elements [m]
27 el_x = 40;                     %Number of elements in x-direction
28 el_y = 1;                      %Number of elements in y-direction
29 width = el_x*pitch - kerf;     %Total width of transducer
30
31 %Active elements, if desired to change the number of active elements
32 el_x = el_x;
33 width = el_x*pitch - kerf;
34
35 transducer_array = create_rect_planar_array(el_x, el_y,...
36     w, h, kerf,0);
37
38 figure(1)
39 draw_array(transducer_array,[0 0 1]);
40
41 %Show the transducer geometry
42 title('Transducer Geometry')
43 xlabel('x-axis [m]')
44 ylabel('y-axis [m]')
45 zlabel('z-axis [m]')
46
47 %Define the appropriate media for the wave propagation
48 define_media();
49 medium = set_medium('lossless');     %Could also use water -> ...
       attenuation
50 medium.soundspeed = 1540;           %To match the soft tissue ...
       properties
51 medium.density = 1060;              %of both speed and density
52 medium.attenuation = 0.7;
53 c = medium.soundspeed;
54 rho = medium.density;
55 f0 = 8e6;                           %Center frequency of the ...
       transducer
56 lambda = medium.soundspeed/f0;      %Wave length
57
58 %Define the minimum and maximum values of the x,y, and z-axis
59 xmin = -x_total/2;
60 xmax = -xmin;
61 ymin = 0;
62 ymax = 0;
63 zmin = 0;
```

```
64 zmax = z_total;
65
66 %Define the transducer focus
67 focus_x = 0;
68 focus_y = 0;
69 focus_z = zmax/2;                    %Defined as a given point
70
71 %Calculate the F-number in order to control that it is on the
72 %range 1 -> 2
73 F = focus_z/width;
74 disp(['F# = ', num2str(F)]);
75
76 if  F<1 || F>2
77     errorMsg = sprintf('Warning: F-number is outside the valid range!...
       ');
78     uiwait(warndlg(errorMsg));
79     return;
80 end
81
82 %Number of points to subdivide the grid into
83 xpoints = 600;
84 ypoints = 1;
85 %zpoints = 200;                   %Is not used, because dz is set to ...
     dx
86
87 %Size of elements in grid
88 dx = (xmax-xmin)/xpoints;
89 dy = (ymax-ymin)/ypoints;
90 dz = dx;                          %Give the same increment in x and z
91 %dz = (zmax-zmin)/zpoints;
92
93 %Find the vectors defining the coordinates to be covered in the
94 %calculation and make the coordinate grid
95 x = xmin:dx:xmax;
96 y = ymin:dy:ymax;
97 z = zmin:dz:zmax;
98 ? = [dx dy dz];
99 cg = set_coordinate_grid(?,xmin,xmax,ymin,ymax,zmin,zmax);
100
101 %Display relevant information regarding the analysis
102 disp(['Focus point at (', num2str(focus_x), ', ', num2str(focus_y), ...
      ...
103 ', ', num2str(focus_z), ')']);
104
105 %% FOCUS
106 %Focus the transducer at the given focus point
107 num_abscissa = 200;           %Number of abscissas used for ...
      calculation
108 transducer_array = find_single_focus_phase(transducer_array,focus_x, ...
      ...
109 focus_y, focus_z, medium, f0, num_abscissa);
110
111 %% APODIZATION
112 if apod == 1
113 apo = zeros(el_x,1);
114 j = 40;
115 for i = 1:el_x
```

```
116     apo(i) = sin((pi*i)/el_x);
117 end
118 transducer_array = set_apodization(transducer_array, apo);
119 end
120
121
122 %% PRESSURE X-Z COLORPLOT
123 %Calculate the 3D pressure distribution with fnm_call. The pressure
124 %distribution is complex, and we need to find the absolute value for
125 %plotting. This is done with P = abs(p).
126
127 ndiv=10;
128 tic();
129 disp('Calculating pressure field...');
130 p = fnm_call(transducer_array, cg, medium, ndiv, f0, 0);
131 disp(['Simulation complete in ', num2str(toc()), ' seconds.'])
132 figure(2)
133 h = pcolor(x*1000,z*1000,rot90(squeeze(abs(p)),3));
134 set(h,'edgecolor','none');
135 colorbar
136 % title('Pressure field at y = 0 cm');
137 % xlabel('x [mm]');
138 % ylabel('z [mm]');
139 %axis([-10,10,0,20])
140
141 Nx = round(length(x));
142 Nz = round(length(z));
143 Nx_middle = round(Nx/2);
144 P = abs(p);                    %Find the pressure amplitude
145
146 %% BEAM PROFILE
147 %Plot the pressure distribution along the lateral (x-axis) direction
148 %for different places along the axial (z-axis) direction.
149 factor = 6;
150 teller = 0;
151 dNz = floor(Nz/factor);
152 figure(3)
153
154 for i = 1:dNz:(factor-1)*dNz+1
155     teller = teller + 1;
156     p_x(:,teller) = P(:,1,i);
157     leg{teller} = strcat('z = ', num2str(1000*i*dz),' [mm]');
158 end
159
160 xx = 1000*x;    %Just for the axis label to be in mm
161
162 plot(xx',p_x)
163 xlabel('Lateral position [mm]')
164 ylabel('Pressure [Pa]')
165 legend(leg)
166 matlab2tikz('ultrasound_beamprofile.tikz','height','\figureheight'...
      ,...
167   'width','\figurewidth','showInfo', false,'parseStrings',false)
168
169 %% PRESSURE ALONG AXIAL POSITION
170 %Find the average pressure for each z-increment. The pressure is
171 %different along the x-direction, dependent on which axial position
```

```
%(z-axis) we are looking at. Also, the pressure will naturally ...
    deviate
%in the lateral (x-axis) direction due to destructive and ...
    constructive
%interference of the waves excerted by each transducer element.
%The following for-loop addes the pressure values that are larger 80 ...
    %
%of the maximum pressure in that z-increment and takes the average of
%these.

%Determine which x-points to include in the average calculation.
%Only interested in the points that are located within the width
%of the transducer.

numPoints = round((width/x_total)*xpoints);
Xmin = round((xpoints-numPoints)/2);
Xmax = Xmin + numPoints;

for i = 1:Nz
    Psum = 0;
    ant = 0;
    Pmax = max(P(Xmin:Xmax,1,i));
    for j = Xmin:Xmax
        if P(j,1,i) > 0.80*Pmax
            Psum = Psum + P(j,1,i);
            ant = ant + 1;
        else
            Psum = Psum;
            ant = ant;
        end
    end
    Pavg(i) = Psum/ant;
end

P = P(Nx_middle,1,:);           %Calculate pressure in middle section
P = squeeze(P);                 %Create a 2D-array pressure (x,z)
X = x*1000;                     %Convert to [mm]
Z = z*1000;
I = P.^2/(2*rho*c);             %Calculate intensity
Iavg = Pavg.^2/(2*rho*c);       %Intensity using averaged pressure

figure(4)
plot(Z',P,'b',Z',Pavg,'r-.')
%title('Pressure distribution')
xlabel('Axial position [mm]')
ylabel('Pressure [Pa]')
legend('Centerline','Averaged')
matlab2tikz('ultrasound_pressure.tikz','height','\figureheight',...
  'width','\figurewidth','showInfo', false,'parseStrings',false)
%% PLOT INTENSITY
figure(5)
plot(Z,I,'b',Z,Iavg,'r-.')
%title('Intensity')
xlabel('Axial position [mm]')
ylabel('Intensity [$\frac{W}{m^2}$]')
legend('Centerline','Averaged')
matlab2tikz('ultrasound_intensity.tikz','height','\figureheight',...
```

```
    'width','\figurewidth','showInfo', false,'parseStrings',false)

%% PLOT BODY FORCE
% Now, calculate the body force [N/m^3]
alpha = 0.7*100;
F = (2*alpha*I)/c;
Favg = (2*alpha*Iavg)/c;

figure(6)
plot(Z,F,'b',Z,Favg,'r-.')
%title('Radiation force')
xlabel('Axial position [mm]')
ylabel('Body force [$\frac{N}{m^3}$]')
legend('Centerline','Averaged')
matlab2tikz('radiation_force.tikz','height','\figureheight',...
    'width','\figurewidth','showInfo', false,'parseStrings',false)

%% CURVE FIT
%Curve fitting in order to find the appropriate body force field
%to implement in Abaqus. createFit.m finds constants for a three-...
    element
%gaussian function.

if fit == 1
[fit,gof] = createFit(z,Favg);
end
```

A.3 createFit.m

```
function [fitresult, gof] = createFit(z, Favg)
%CREATEFIT(Z,FAVG)
%  Create a fit.
%
%  Data for 'untitled fit 4' fit:
%      X Input : z
%      Y Output: Favg
%  Output:
%      fitresult : a fit object representing the fit.
%      gof : structure with goodness-of fit info.
%
%  See also FIT, CFIT, SFIT.

%  Auto-generated by MATLAB on 26-Apr-2013 11:12:07


%% Fit: 'untitled fit 4'.
[xData, yData] = prepareCurveData( z, Favg );

% Set up fittype and options.
ft = fittype( 'gauss3' );
opts = fitoptions( ft );
opts.Display = 'Off';
```

```
opts.Lower = [-Inf -Inf 0 -Inf -Inf 0 -Inf -Inf 0];
opts.StartPoint = [1864369.83586478 0.0101 0.000547727483697369...
    1098616.03363283 0.0109666666666667 0.000778445275184035...
    1083955.92275907 0.0093 0.000944554623995073];
opts.Upper = [Inf Inf Inf Inf Inf Inf Inf Inf Inf];

% Fit model to data.
[fitresult, gof] = fit( xData, yData, ft, opts );

x = xData*1000; %Convert to mm

% Plot fit with data.
figure();
h = plot( fitresult, xData, yData );
title = ('Radiation force curve fit')
legend( h, 'FOCUS', 'Gaussian fit', 'Location', 'NorthEast' );
% Label axes
xlabel( 'Axial direction [m]' );
ylabel( 'Averaged Radiation Force $\frac{N}{m^3}$' );
matlab2tikz('radiation_force_fit.tikz','height','\figureheight',...
  'width','\figurewidth','showInfo', false,'parseStrings',false)
```

A.4 meshStudy.m

```
% Written by: Lars Edvard Bryhni Daehli
%
% Used to extract the shear stress and transversal displacement for
% different mesh sizes.

clear all; clf; clc;

folder = 'D:\NTNU\Master\Matlab\Extracted\2D_PlaneStrain\MeshStudy\';
cd(folder);          %Make sure to put files in correct folder
splot = 1;           %Plot the shear stress
uplot = 1;           %Plot the transversal displacement
reflected = 1;       %Use "reflection" node from Abaqus model
boundary = 1;        %Use "boundary" node from Abaqus model
E = 107585;                     %Elasticity of tissue [Pa]
rho = 1060;                     %Density of tissue [kg/m^3]
nu = 0.499;                     %Poisson's ratio of tissue []
c_t = sqrt(E/(2*rho*(1+nu)));   %Shear wave speed [m/s]
T = 250e-06;                    %Impulse time [s]
%% Which n to include. Must match the files in the folder ...\...
    MeshStudy

n = [6,8,10,12,14]';            %Number of elements impulse loading ...
    spans
dx = 1000*c_t*T./n;             %Element size [mm]
col = hsv(length(n));           %Provides a unique color for each "n"

%% Plot the shear stress (S12) for different mesh sizes.
%Can choose whether to use reflection node or boundary node, or both.
```

```
27
28 if splot == 1
29     if reflected == 1
30         figure(1)
31         hold on;
32         for i = 1:length(n);
33             tofind = (['*stress_reflect_n=',num2str(n(i)),'.dat']);
34             file = fullfile(folder,tofind);
35             datfile = dir(file);
36             name = datfile.name;
37             fullName = fullfile(folder,name);
38             imported = importdata(fullName,' ',4);
39             t = imported.data(:,1);
40             s = imported.data(:,2);
41             leg{i} = ['n = ', num2str(n(i))];
42             plot(1000*t,s,'color',col(i,:));
43         end
44         xlabel('Time [ms]')
45         ylabel('$\sigma_{12}$ [Pa]')
46         legend(leg);
47         matlab2tikz('meshstudy_stress_reflection.tikz','height',...
48             '\figureheight', 'width','\figurewidth','showInfo',...
49             false,'parseStrings',false)
50     end
51     if boundary == 1
52         figure(2)
53         hold on;
54         for i = 1:length(n);
55             tofind = (['*stress_boundary_n=',num2str(n(i)),'.dat']);
56             file = fullfile(folder,tofind);
57             datfile = dir(file);
58             name = datfile.name;
59             fullName = fullfile(folder,name);
60             imported = importdata(fullName,' ',4);
61             t = imported.data(:,1);
62             s = imported.data(:,2);
63             leg{i} = ['n = ', num2str(n(i))];
64             plot(1000*t,s,'color',col(i,:));
65         end
66         xlabel('Time [ms]')
67         ylabel('$\sigma_{12}$ [Pa]')
68         legend(leg);
69         matlab2tikz('meshstudy_stress_boundary.tikz','height',...
70             '\figureheight', 'width','\figurewidth','showInfo',...
71             false,'parseStrings',false)
72     end
73 end
74
75 %% Plot transversal displacement (u2) for different mesh sizes.
76 %Can choose between reflection node and boundary node, or both. Plot ...
       in
77 %micro meter.
78
79 if uplot == 1
80     if reflected == 1
81         figure(3)
82         hold on;
```

```
83          for i = 1:length(n);
84              tofind = (['*u2_reflect_n=',num2str(n(i)),'.dat']);
85              file = fullfile(folder,tofind);
86              datfile = dir(file);
87              name = datfile.name;
88              fullName = fullfile(folder,name);
89              imported = importdata(fullName,' ',4);
90              t = imported.data(:,1);
91              u = -10^6*(imported.data(:,2));
92              %leg{i} = ['$\Delta{x}$ = ', num2str(dx(i)), ' [mm]'];
93              leg{i} = ['n = ', num2str(n(i))];
94              plot(1000*t,u,'color',col(i,:));
95          end
96          xlabel('Time [ms]')
97          ylabel('$u_2$ [$\mu$m]')
98          legend(leg);
99          matlab2tikz('meshstudy_displacement_reflection.tikz','height'...
                ,...
100             '\figureheight', 'width','\figurewidth','showInfo',...
101             false,'parseStrings',false)
102     end
103     if boundary == 1
104         figure(4)
105         hold on;
106         for i = 1:length(n);
107             tofind = (['*u2_boundary_n=',num2str(n(i)),'.dat']);
108             file = fullfile(folder,tofind);
109             datfile = dir(file);
110             name = datfile.name;
111             fullName = fullfile(folder,name);
112             imported = importdata(fullName,' ',4);
113             t = imported.data(:,1);
114             u = -10^6*(imported.data(:,2));
115             %leg{i} = ['$\Delta{x}$ = ', num2str(dx(i)), ' [mm]'];
116             leg{i} = ['n = ', num2str(n(i))];
117             plot(1000*t,u,'color',col(i,:));
118         end
119         xlabel('Time [ms]')
120         ylabel('$u_2$ [$\mu$m]')
121         legend(leg);
122         matlab2tikz('meshstudy_displacement_boundary.tikz','height'...
                ,...
123             '\figureheight', 'width','\figurewidth','showInfo',...
124             false,'parseStrings',false)
125     end
126 end
127
128 %% Calculate mean deviance between different mesh sizes.
129 % Use n=14 as reference. Can be considered in the report.
130 tofind = (['*stress_reflect_n=14.dat']);
131 file = fullfile(folder,tofind);
132 datfile = dir(file);
133 name = datfile.name;
134 fullName = fullfile(folder,name);
135 imported = importdata(fullName,' ',4);
136 stress_ref = imported.data(3:end,2);
137
```

```
for i = 1:length(n)-1
    tofind = (['*stress_reflect_n=',num2str(n(i)),'.dat']);
    file = fullfile(folder,tofind);
    datfile = dir(file);
    name = datfile.name;
    fullName = fullfile(folder,name);
    imported = importdata(fullName,' ',4);
    t = imported.data(3:end,1);
    s = imported.data(3:end,2);
    diff(:,i) = s-stress_ref;
    dev(:,i) = abs(diff(:,i)./stress_ref);
    ldev(:,i) = log(dev(:,i));
    mean_dev(i) = sum(dev(:,i))/length(t);
    leg{i} = ['n = ', num2str(n(i))];
end
```

A.5 waveSpeeds.m

```
% Written by: Lars Edvard Bryhni Daehli
%
% Used to determine the shear wave speed for soft tissue and tumor ...
    based on
% the FEA results from planeStrainPlate with n=10.

clear all; clf; clc;
folder = ...
'D:\NTNU\Master\Matlab\Extracted\2D_PlaneStrain\WaveSpeeds\';
cd(folder);         %Make sure to put files in correct folder

%% Tissue values
E0 = 107585;                            %Elasticity of tissue [Pa]
rho = 1060;                             %Density of tissue [kg/m^3]
nu = 0.499;                             %Poisson's ratio of tissue []
ct_ti = sqrt(E0/(2*rho*(1+nu)));        %Shear wave speed [m/s]
T = 250e-06;                            %Impulse time [s]
n = 10;                                 %Number of elements to span
dx = 1000*ct_ti*T./n;                   %Element size [mm]

disp(['Analytical shear wave speed in tissue: ', num2str(ct_ti), ' [m...
    /s]'])

%% Tumor values
E = 3*E0;                               %Elasticity of tissue [Pa]
rho = 1060;                             %Density of tissue [kg/m^3]
nu = 0.499;                             %Poisson's ratio of tissue []
ct_tu = sqrt(E/(2*rho*(1+nu)));         %Shear wave speed [m/s]

disp(['Analytical shear wave speed in tumor: ', num2str(ct_tu), ' [m/...
    s]'])

%% Load files

```

```
% Mid_Reflection
name = ('*stress_reflect.dat');
file = fullfile(folder,name);
datfile = dir(file);
datname = datfile.name;
fullName = fullfile(folder,datname);
imported = importdata(fullName,' ',4);
stress_reflect = imported.data(:,2);
t = imported.data(:,1);

% Tumor_Boundary
name = ('*stress_boundary.dat');
file = fullfile(folder,name);
datfile = dir(file);
datname = datfile.name;
fullName = fullfile(folder,datname);
imported = importdata(fullName,' ',4);
stress_boundary = imported.data(:,2);


% Mid_Tumor
name = ('*stress_tumor.dat');
file = fullfile(folder,name);
datfile = dir(file);
datname = datfile.name;
fullName = fullfile(folder,datname);
imported = importdata(fullName,' ',4);
stress_tumor = imported.data(:,2);

figure(1)
plot(t,stress_reflect,t,stress_boundary,t,stress_tumor)
xlabel('Time [s]');
ylabel('$\sigma_{12}$ [Pa]')
legend('Mid$\_$Reflection','Tumor$\_$Boundary','Mid$\_$Tumor')
matlab2tikz('stress_waveSpeedStudy.tikz','height','\figureheight',...
    'width','\figurewidth','showInfo', false,'parseStrings',false)


%% Define a region for the maxima search

[T,s] = ginput(2);      % Use T to define where to search for maxima

t_min = T(1); t_max = T(2);

j=1;

for i = 1:length(t)
    if t(i) > t_min && t(i) < t_max
        tau(j) = t(i);
        S_r(j) = stress_reflect(i);
        S_b(j) = stress_boundary(i);
        S_t(j) = stress_tumor(i);
        j = j+1;
    end
end

%% Find maxima for each of the node sets
```

```
89 [max_r,n_r] = max(S_r); [max_b,n_b] = max(S_b); [max_t,n_t] = max(S_t...
       );
90
91 % Find the corresponding values of tau
92 tau_r = tau(n_r); tau_b = tau(n_b); tau_t = tau(n_t);
93
94 % Find the time it takes to travel between node sets
95 dt_tissue = tau_b-tau_r; dt_tumor = tau_t-tau_b;
96
97 %Distance between node sets (found from planeStrainPlate.py)
98 dx_tissue = 4.05e-03; dx_tumor = 1.5e-03;
99
100 ct_tissue = dx_tissue/dt_tissue;
101 ct_tumor = dx_tumor/dt_tumor;
102
103 disp(['Numerical shear wave speed in tissue: ',...
104     num2str(ct_tissue), ' [m/s]'])
105 disp(['Numerical shear wave speed in tumor: ',...
106     num2str(ct_tumor), ' [m/s]'])
107
108 dev_tissue = 100*(ct_tissue-ct_ti)/ct_ti;
109 dev_tumor = 100*(ct_tumor-ct_tu)/ct_tu;
110
111 disp(['Deviation in tissue: ',...
112     num2str(dev_tissue), ' %'])
113 disp(['Deviation in tumor: ',...
114     num2str(dev_tumor), ' %'])
115
116 figure(2)
117 plot(1000*tau,S_r,1000*tau,S_b,1000*tau,S_t)
118 xlabel('Time [ms]');
119 ylabel('$\sigma_{12}$ [Pa]')
120 legend('Mid$\_$Reflection','Tumor$\_$Boundary','Mid$\_$Tumor')
121 axis([0 1000*max(t) -200 650]);
122 hold on
123 plot(1000*tau_r,max_r,'r.',1000*tau_b,max_b,'r.',1000*tau_t,max_t,'r....
       ')
124 hold off
125 matlab2tikz('maxPoints_waveSpeedStudy.tikz','height','\figureheight'...
       ,...
126     'width','\figurewidth','showInfo', false,'parseStrings',false)
127
128 %% Elastic analysis
129
130 % Mid_Reflection
131 name = ('*elastic_stress_reflect.dat');
132 file = fullfile(folder,name);
133 datfile = dir(file);
134 datname = datfile.name;
135 fullName = fullfile(folder,datname);
136 imported = importdata(fullName,' ',4);
137 stress_reflectEl = imported.data(:,2);
138 t_El = imported.data(:,1);
139 % Tumor_Boundary
140 name = ('*elastic_stress_boundary.dat');
141 file = fullfile(folder,name);
142 datfile = dir(file);
```

```
datname = datfile.name;
fullName = fullfile(folder,datname);
imported = importdata(fullName,' ',4);
stress_boundaryEl = imported.data(:,2);
% Mid_Tumor
name = ('*elastic_stress_tumor.dat');
file = fullfile(folder,name);
datfile = dir(file);
datname = datfile.name;
fullName = fullfile(folder,datname);
imported = importdata(fullName,' ',4);
stress_tumorEl = imported.data(:,2);

%% Find maxima for each of the node sets
[max_rEl,n_rEl] = max(stress_reflectEl);
[max_bEl,n_bEl] = max(stress_boundaryEl);
[max_tEl,n_tEl] = max(stress_tumorEl);

% Find the corresponding values of tau
tau_rEl = t_El(n_rEl); tau_bEl = t_El(n_bEl); tau_tEl = t_El(n_tEl);

% Find the time it takes to travel between node sets
dt_tissueEl = tau_bEl-tau_rEl; dt_tumorEl = tau_tEl-tau_bEl;

%Distance between node sets (found from planeStrainPlate.py)
dx_tissue = 4.05e-03; dx_tumor = 1.5e-03;

ct_tissueEl = dx_tissue/dt_tissueEl;
ct_tumorEl = dx_tumor/dt_tumorEl;

disp(['Numerical shear wave speed in tissue: ',...
    num2str(ct_tissueEl), ' [m/s]'])
disp(['Numerical shear wave speed in tumor: ',...
    num2str(ct_tumorEl), ' [m/s]'])

dev_tissueEl = 100*(ct_tissueEl-ct_ti)/ct_ti;
dev_tumorEl = 100*(ct_tumorEl-ct_tu)/ct_tu;

disp(['Deviation in tissue: ',...
    num2str(dev_tissueEl), ' %'])
disp(['Deviation in tumor: ',...
    num2str(dev_tumorEl), ' %'])

figure(3)
plot(1000*t_El,stress_reflectEl,1000*t_El,stress_boundaryEl,...
    1000*t_El,stress_tumorEl)
xlabel('Time [ms]');
ylabel('$\sigma_{12}$ [Pa]')
legend('Mid$\_$Reflection','Tumor$\_$Boundary','Mid$\_$Tumor')
axis([0 1000*max(t_El) -200 650]);
hold on
plot(1000*tau_rEl,max_rEl,'r.',1000*tau_bEl,max_bEl,'r.',...
    1000*tau_tEl,max_tEl,'r.')
hold off
matlab2tikz('maxPointsElastic_waveSpeedStudy.tikz','height',...
    '\figureheight', 'width','\figurewidth','showInfo',...
    false,'parseStrings',false)
```

A.6 extractData.m

```
% Written by: Lars Edvard Bryhni Daehli

function [t,r] = extractData(name,folder)

file = fullfile(folder,name);
datfile = dir(file);
datname = datfile.name;
fullName = fullfile(folder,datname);
imported = importdata(fullName,' ',4);
t = imported.data(:,1);
r = imported.data(:,2);

end
```

A.7 reflectionStudy.m

```
% Written by: Lars Edvard Bryhni Daehli
%
% Find the reflection coefficient from the analysis and compare with ...
    the
% analytical values found from plane shear wave reflection theory.

clear all; clf; clc;
folder = ...
'D:\NTNU\Master\Matlab\Extracted\2D_PlaneStrain\Reflection\';
cd(folder);          %Make sure to put files in correct folder

%% Load files
j=1;
for i = 2:8
    name{j} = ['*Refl_E=',num2str(i),'.dat'];
    [T,S] = extractData(name{j},folder);
    t(:,j) = T;      %T should be the same for all files!
    s(:,j) = S;      %S will change.
    j = j+1;
end

%% Define the appropariate window to search for the maximum values.
% This is done with the ginput(4). This gives 4 different time values...
    .
% The two first time values should be used to find the first maxima. ...
    The
% two last time values are used to find the second maximum.
figure(1)
plot(t,s)
xlabel('Time [s]')
ylabel('Stress [Pa]')

```

```
30 [tau,amp] = ginput(4);
31
32 t1_min = tau(1); t1_max = tau(2);
33 t2_min = tau(3); t2_max = tau(4);
34
35 %% Find the first maximum
36 for j = 1:7
37     k = 1;
38     for i = 1:length(t(:,1))
39         if t(i,j) ≥ t1_min && t(i,j) ≤ t1_max
40             s1(k,j) = s(i,j);
41             S_diff(1,j) = s1(1,j);
42             k = k+1;
43         end
44     end
45 end
46
47 [amp1,n1] = max(s1);
48 s1_offset = amp1-S_diff;
49
50 %% Find the second maximum
51 for j = 1:7
52     k = 1;
53     for i = 1:length(t(:,1))
54         if t(i,j) ≥ t2_min && t(i,j) ≤ t2_max
55             s2(k,j) = s(i,j);
56             s_diff(1,j) = s2(1,j);
57             k = k+1;
58         end
59     end
60 end
61
62 [amp2,n2] = max(s2);
63 s2_offset = amp2-s_diff;
64
65 %% Find the ratios between amp1 and amp2 and define the analytical
66 %  expression
67 E_numRat = 2:1:8;
68 R_num = s2_offset./s1_offset;
69 E_rat = 1:0.1:9;
70 R_ana = (sqrt(E_rat)-1)./(sqrt(E_rat)+1);
71
72 figure(2)
73 plot(E_rat,R_ana)
74 xlabel('$\dfrac{E^{\text{(tumor)}}}{E^{\text{(tissue)}}}$')
75 ylabel('R')
76 hold on
77 plot(E_numRat,R_num,'r.')
78 legend('Analytical', 'Numerical','Location','SouthEast')
79 hold off
80 matlab2tikz('reflectionFactor.tikz','height',...
81     '\figureheight', 'width','\figurewidth','showInfo',...
82     false,'parseStrings',false)
83
84
85 %% Find the deviance
86 for i = 1:length(R_num)
```

```
87      for j = 1:length(R_ana)
88          if E_rat(j) == E_numRat(i)
89              dev(i) = abs(100*((R_num(i)-R_ana(j))/R_ana(j)));
90          end
91      end
92  end
93
94  figure(3)
95  plot(E_numRat,dev,'r.')
96  xlabel('$\dfrac{E^{\text{(tumor)}}}{E^{\text{(tissue)}}}$')
97  ylabel('$100\cdot\left|\dfrac{R^{num}-R^{ana}}{R^{ana}}\right|$ [\%]'...
        )
98  axis([1 9 30 60])
99  matlab2tikz('devianceReflection.tikz','height',...
100     '\figureheight', 'width','\figurewidth','showInfo',...
101     false,'parseStrings',false)
102
103
104 %% Illustrate how to find R for E_tumor = 8*E_tissue
105 A1 = amp1(7); A2 = amp2(7);
106
107 for i = 1:length(t(:,7))
108     if s(i,7) == A1
109         N1 = i;
110     end
111     if s(i,7) == A2
112         N2 = i;
113     end
114 end
115
116 T1 = t(N1,7);
117 T2 = t(N2,7);
118
119 figure(4)
120 plot(1000*t(:,7),s(:,7),'r')
121 xlabel('Time [ms]')
122 ylabel('$\sigma_{12}$ [Pa]')
123 hold on
124 plot(1000*T1,A1,'b.',1000*T2,A2,'g.',1000*t2_min,s_diff(7),'m.',...
125     1000*t1_min,S_diff(7),'y.')
126 matlab2tikz('amplitudeExtraction.tikz','height',...
127     '\figureheight', 'width','\figurewidth','showInfo',...
128     false,'parseStrings',false)
129
130 %% Elliptical inclusion
131 j=1;
132 for i = 3:2:9
133     name{j} = ['*Refl_E=8_R=',num2str(i),'.dat'];
134     [TT,SS] = extractData(name{j},folder);
135     tt(:,j) = TT;       %T should be the same for all files!
136     ss(:,j) = SS;       %S will change.
137     j = j+1;
138 end
139
140 for j = 1:4
141     k = 1;
142     for i = 1:length(tt(:,j))
```

```
143         if tt(i,j) ≥ t1_min && tt(i,j) ≤ t1_max
144             s3(k,j) = ss(i,j);
145             SS_diff(1,j) = s3(1,j);
146             k = k+1;
147         end
148     end
149 end
150
151 [amp3,n3] = max(s3);
152 s3_offset = amp3-SS_diff;
153
154 for j = 1:4
155     k = 1;
156     for i = 1:length(tt(:,j))
157         if tt(i,j) ≥ t2_min && tt(i,j) ≤ t2_max
158             s4(k,j) = ss(i,j);
159             ss_diff(1,j) = s4(1,j);
160             k = k+1;
161         end
162     end
163 end
164
165 r = 1:1:10;
166
167 [amp4,n4] = max(s4);
168 s4_offset = amp4-ss_diff;
169
170 R8num = s4_offset./s3_offset;
171 R8num = [R_num(7) R8num];
172 R8ana = 0.4775922*ones(size(r));
173 Ry = [1.5 3 5 7 9];
174
175 figure(5)
176 plot(Ry,R8num,'r.',r,R8ana,'b')
177 xlabel('Major radius $R_2$ [mm]')
178 ylabel('R')
179 axis([1 10 0.25 0.55])
180 legend('Numerical','Plane wave theory','Location','SouthEast')
181 matlab2tikz('majorRadiusDependence.tikz','height',...
182     '\figureheight', 'width','\figurewidth','showInfo',...
183     false,'parseStrings',false)
184
185 for i = 1:5
186     Dev(i) = 100*abs((R8num(i)-R8ana(1))/R8ana(1));
187 end
188
189 figure(6)
190 plot(Ry,Dev,'r.')
191 xlabel('Major radius $R_2$ [mm]')
192 ylabel('Deviance [\%]')
193 axis([1 10 0 50])
194 matlab2tikz('deviance_majorRadiusDependence.tikz','height',...
195     '\figureheight', 'width','\figurewidth','showInfo',...
196     false,'parseStrings',false)
197
198 figure(7)
199 hold on
```

```
plot(1000*tt,ss)
xlabel('Time [s]')
ylabel('$\sigma_{12}$')
plot(1000*t2_min,ss_diff,'m.',...
    1000*t1_min,SS_diff,'y.')
hold off
matlab2tikz('amplitudeExtraction_ellipse.tikz','height',...
    '\figureheight', 'width','\figurewidth','showInfo',...
    false,'parseStrings',false)
```

A.8 TTP.m

```
% Written by: Lars Edvard Bryhni Daehli
%
% Find the TTP displacement from the analysis

clear all; clf; clc;
folder = ...
'D:\NTNU\Master\Matlab\Extracted\2D_PlaneStrain\TTP\';
cd(folder);          %Make sure to put files in correct folder

%% Load files for elasticity study
col = hsv(6);
j=1;
for i = 60:20:160
    name{j} = ['*displacement_E=',num2str(i),'.dat'];
    [T,D] = extractData(name{j},folder);
    t(:,j) = T;          %T should be the same for all files!
    d(:,j) = -D;         %D will change.
    leg{j} = ['E = ', num2str(i), ' kPa'];
    E(j) = i;
    j = j+1;
end

%% Find maximum displacement for each value of E
[nc,nr] = size(d);

figure(1)
hold on
tt = 1000.*t;
dd = 10^6.*d;
plot(tt,dd)
xlabel('Time [ms]')
ylabel('$u_2$ [$\mu$m]')
for i = 1:nr
    [amp(i),n(i)] = max(d(:,i));
    ttp(i) = t(n(i),i);
    plot(1000*ttp(i),10^6*amp(i),'r.')
end
legend(leg)
matlab2tikz('TTPdisplacement.tikz','height',...
    '\figureheight', 'width','\figurewidth','showInfo',...
```

```
41      false,'parseStrings',false)
42
43  ttp_ms = 1000*ttp;
44  figure(2)
45  plot(E,ttp_ms,'bd',E,ttp_ms,'b')
46  xlabel('E [kPa]')
47  ylabel('Time to Peak Displacement [ms]')
48  axis([55 165 0.29 0.36]);
49  matlab2tikz('TTP.tikz','height',...
50      '\figureheight', 'width','\figurewidth','showInfo',...
51      false,'parseStrings',false)
52
53  %% Load files for pulse duration study
54  j=1;
55  for i = 50:50:250
56      Name{j} = ['*ttp=',num2str(i),'.dat'];
57      [T,D] = extractData(Name{j},folder);
58      t_p(:,j) = T;           %T should be the same for all files!
59      d_p(:,j) = -D;          %D will change.
60      Leg{j} = ['$T_i$ = ', num2str(i), ' $\mu$s'];
61      j = j+1;
62  end
63
64  figure(3)
65  hold on
66  TT = 1000.*t_p;
67  DD = 10^6.*d_p;
68  plot(TT,DD)
69  xlabel('Time [ms]')
70  ylabel('$u_2$ [$\mu$m]')
71  for i = 1:5
72      [Amp(i),N(i)] = max(d_p(:,i));
73      ttp_p(i) = t_p(N(i),i);
74      plot(1000*ttp_p(i),10^6*Amp(i),'r.')
75      pulse(i) = 50*i;
76  end
77  axis([0 2 -1 14]);
78  legend(Leg)
79  matlab2tikz('TTPdisplacement_pulse.tikz','height',...
80      '\figureheight', 'width','\figurewidth','showInfo',...
81      false,'parseStrings',false)
82
83  TTP_ms = 1000*ttp_p;
84  figure(4)
85  plot(pulse,TTP_ms,'bd',pulse,TTP_ms,'b')
86  xlabel('Impulse Time [$\mu$s]')
87  ylabel('Time to Peak Displacement [ms]')
88  axis([40 260 0.15 0.36]);
89  matlab2tikz('TTP_pulse.tikz','height',...
90      '\figureheight', 'width','\figurewidth','showInfo',...
91      false,'parseStrings',false)
```

B | PYTHON scripts

The scripts given below were used to automatize the modelling, simulation and data retrieving in Abaqus - and to be able to perform parametric studies in an easy fashion.

- *uniaxial.py* was used to simulate the experiment on the gel-agar phantom in Abaqus.
- *planeStrainPlate.py* was the main FEM model in this thesis. It contains a two-dimensional plane strain plate with an inclusion. Infinite elements are used in the model. It is possible to change relevant modelling features, such as; element size, material properties, elastic or viscoelastic analysis, type of element, analysis time, impulse time and tumor size.

B.1 uniaxial.py

```
# Written by: Lars Edvard Bryhni Daehli

# ---------- Initializing ----------

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)

workDirectory = r'D:\NTNU\Master\Simulations\Uniaxial_Model'
caeName = 'Uniaxial.cae'
jobName = 'Uniaxial_Test'
```

```
25
26 os.chdir(workDirectory)
27
28 # ---------- Parameters ----------
29 # Naming convention according to the experiments carried out at UGent
30
31 a = 63.97E-03 #Length (uniaxial test direction)
32 b = 57.4E-03  #Width
33 c = 3.5E-03 #Thickness
34
35 #Compensate for the initial stress -> Smaller area!
36 red = sqrt(1-0.004027357)
37 b = b*red
38 c = c*red
39
40 saveModel = 1 #Set to 1 if you want to save the CAE-model.
41 viscoLoad = 1 #1 if viscoelastic step during loading.
42 clamped = 0 #1 if specimen is modelled as clamped. 0 if
43       #simply supported and prevented from moving in y and z.
44
45 E0 = 107585.6   #Instantaneous elastic modulus from experimental data...
     .
46 nu0 = 0.499    #Chosen Poisson's ratio.
47 rho = 1060     #Chosen density.
48 LStol = 0.0019     #Tolerance criteria for lest square fit of ...
     relaxation data.
49 Nmax = 13    #Maximum number of terms in the prony series.
50
51 elSize = 1.1E-03  #Approximate global elemenent size.
52 elType = C3D20R   #Choose element type. Check Abaqus manual for names...
     .
53
54 timeLoading = 10.239  #The time it takes to reach desired ...
     displacement.
55 timeRelax = 299.98  #The total relaxation time.
56 if viscoLoad == 0:
57  dtLoading = 0.5*timeLoading  #Time steps for the loading part if ...
      elastic.
58 if viscoLoad == 1:
59  dtLoading = 0.05*timeLoading #Time steps for the loading part if ...
      viscoelastic.
60
61 dtRelax = 5      #Time steps for the relaxation part.
62 preDisp = 0.25763E-03   #Offset of the relaxation data
63 dispLoad = 3.20633E-03-preDisp  #Total displacement when relaxation ...
     begins.
64 preStress = 1520.5     #Pre-stress of specimen
65
66
67 CEtol = 0.05     #Tolerance for viscoelasticity (on the order of max ...
     strain).
68
69 percMemory = 80    #How many percentage of the CPU maximum for ...
     analysis and
70       #preprocessing
71
72 prony = 1   #If you want to use the relaxation coefficients
```

```
73        #given or implement your own, set prony = 1.
74        #prony = 0 -> Use the relaxation values implemented from
75        #relaxation test.
76 # Relaxation cefficients
77 g1 = 9.45749E-03
78 g2 = 3.84291E-02
79 g3 = 6.46134E-02
80 t1 = 1.5178
81 t2 = 11.541
82 t3 = 93.977
83 # ---------------- Create model ------------------
84
85 mdb.Model(modelType=STANDARD_EXPLICIT, name='Uniaxial')
86
87 # ------------ Creating geometry ----------------.
88
89 mdb.models['Uniaxial'].ConstrainedSketch(name='__profile__', ...
       sheetSize=2*a)
90 mdb.models['Uniaxial'].sketches['__profile__'].sketchOptions....
       setValues(
91       decimalPlaces=3)
92 mdb.models['Uniaxial'].sketches['__profile__'].rectangle(point1=(0, b...
       )
93       , point2=(a, 0))
94 mdb.models['Uniaxial'].Part(dimensionality=THREE_D, name='Part-1', ...
       type=
95       DEFORMABLE_BODY)
96 mdb.models['Uniaxial'].parts['Part-1'].BaseSolidExtrude(depth=c, ...
       sketch=
97       mdb.models['Uniaxial'].sketches['__profile__'])
98
99 # ---------------- Material -----------------------
100
101 mdb.models['Uniaxial'].Material(name='Viscoelastic')
102 mdb.models['Uniaxial'].materials['Viscoelastic'].Elastic(moduli=...
       INSTANTANEOUS,
103       table=((E0, nu0), ))
104 mdb.models['Uniaxial'].materials['Viscoelastic'].Density(table=((rho,...
        ), ))
105 mdb.models['Uniaxial'].materials['Viscoelastic'].Viscoelastic(domain=...
       TIME,
106       errtol=LStol, nmax=Nmax, table=(), time=RELAXATION_TEST_DATA)
107
108 if prony == 0:
109  mdb.models['Uniaxial'].materials['Viscoelastic'].viscoelastic....
        shearTestData.setValues(
110       shrinf=0.8875, table=((0.995151231, 0.5), (0.993538583, 1.0), ...
          (0.988378965,
111       1.5), (0.984403849, 2.0), (0.982747908, 2.5), (0.982737084, 3.0),...
           (
112       0.97859182, 3.5), (0.978158894, 4.0), (0.979111331, 4.5), ...
          (0.975831918,
113       5.0), (0.970084828, 5.5), (0.970528577, 6.0), (0.969478732, 6.5),...
           (
114       0.967876907, 7.0), (0.967833614, 7.5), (0.960214121, 8.0), ...
          (0.964294446,
```

```
115     8.5), (0.969825073, 9.0), (0.960116712, 9.5), (0.954001636, 10.0)...
        , (
116     0.963526003, 10.5), (0.960993387, 11.0), (0.958071138, 11.5), ...
            (0.955192182,
117     12.0), (0.957042939, 12.5), (0.958460771, 13.0), (0.956988824, ...
            13.5), (
118     0.957335164, 14.0), (0.953720234, 14.5), (0.950668107, 15.0), ...
            (0.95088457,
119     15.5), (0.953871758, 16.0), (0.952324048, 16.5), (0.950419175, ...
            17.0), (
120     0.949758963, 17.5), (0.94668519, 18.0), (0.944747847, 18.5), ...
            (0.944563854,
121     19.0), (0.948438539, 19.5), (0.948687472, 20.0), (0.945786869, ...
            20.5), (
122     0.944271629, 21.0), (0.946847537, 21.5), (0.943514009, 22.0), ...
            (0.94627391,
123     22.5), (0.945397236, 23.0), (0.943059437, 23.5), (0.942042061, ...
            24.0), (
124     0.945851808, 24.5), (0.943524832, 25.0), (0.943838703, 25.5), ...
            (0.939412037,
125     26.0), (0.939347098, 26.5), (0.942962028, 27.0), (0.938015851, ...
            27.5), (
126     0.936662958, 28.0), (0.939368744, 28.5), (0.939282159, 29.0), ...
            (0.939660969,
127     29.5), (0.941749836, 30.0), (0.940223773, 30.5), (0.927971973, ...
            31.0), (
128     0.936598019, 31.5), (0.938676063, 32.0), (0.935071956, 32.5), ...
            (0.933513423,
129     33.0), (0.934509152, 33.5), (0.933816471, 34.0), (0.932225469, ...
            34.5), (
130     0.934714792, 35.0), (0.934595737, 35.5), (0.934422567, 36.0), ...
            (0.930840106,
131     36.5), (0.932474401, 37.0), (0.933253668, 37.5), (0.935028663, ...
            38.0), (
132     0.931814984, 38.5), (0.931835836, 39.0), (0.929032641, 39.5), ...
            (0.927993619,
133     40.0), (0.930775167, 40.5), (0.931348794, 41.0), (0.932355347, ...
            41.5), (
134     0.929941785, 42.0), (0.929314043, 42.5), (0.928491484, 43.0), ...
            (0.92300415,
135     43.5), (0.927668925, 44.0), (0.931478672, 44.5), (0.93081846, ...
            45.0), (
136     0.925038901, 45.5), (0.924768322, 46.0), (0.927019536, 46.5), ...
            (0.927495755,
137     47.0), (0.927344231, 47.5), (0.927073652, 48.0), (0.928480661, ...
            48.5), (
138     0.925915576, 49.0), (0.925515119, 49.5), (0.924595152, 50.0), ...
            (0.925807344,
139     50.5), (0.925363595, 51.0), (0.923848355, 51.5), (0.92221406, ...
            52.0), (
140     0.923144851, 52.5), (0.925082193, 53.0), (0.925352772, 53.5), ...
            (0.925028078,
141     54.0), (0.92224653, 54.5), (0.924865731, 55.0), (0.923404606, ...
            55.5), (
142     0.920168486, 56.0), (0.925082193, 56.5), (0.920991045, 57.0), ...
            (0.921380678,
```

```
143     57.5), (0.920439064, 58.0), (0.921045161, 58.5), (0.922657809, ...
            59.0), (
144     0.921672903, 59.5), (0.920601412, 60.0), (0.920731289, 60.5), ...
            (0.922776864,
145     61.0), (0.920190132, 61.5), (0.920449888, 62.0), (0.920276717, ...
            62.5), (
146     0.920850344, 63.0), (0.921478086, 63.5), (0.919800499, 64.0), ...
            (0.919508274,
147     64.5), (0.919280988, 65.0), (0.919118641, 65.5), (0.919259342, ...
            66.0), (
148     0.918501722, 66.5), (0.919843791, 67.0), (0.918988763, 67.5), ...
            (0.91897794,
149     68.0), (0.917202944, 68.5), (0.918480075, 69.0), (0.918545014, ...
            69.5), (
150     0.919454158, 70.0), (0.917993034, 70.5), (0.91708389, 71.0), ...
            (0.917192121,
151     71.5), (0.917040597, 72.0), (0.916402032, 72.5), (0.917668339, ...
            73.0), (
152     0.917473523, 73.5), (0.916423678, 74.0), (0.916012398, 74.5), ...
            (0.916326269,
153     75.0), (0.914702798, 75.5), (0.916109807, 76.0), (0.915287248, ...
            76.5), (
154     0.91574182, 77.0), (0.915135724, 77.5), (0.914443042, 78.0), ...
            (0.916153099,
155     78.5), (0.915687704, 79.0), (0.914908438, 79.5), (0.916131453, ...
            80.0), (
156     0.915016669, 80.5), (0.915449595, 81.0), (0.915406302, 81.5), ...
            (0.913956001,
157     82.0), (0.914551274, 82.5), (0.913761184, 83.0), (0.913588014, ...
            83.5), (
158     0.914226579, 84.0), (0.914432219, 84.5), (0.913295789, 85.0), ...
            (0.911574909,
159     85.5), (0.912419114, 86.0), (0.912559815, 86.5), (0.913176734, ...
            87.0), (
160     0.913620483, 87.5), (0.912949448, 88.0), (0.91288451, 88.5), ...
            (0.91302521,
161     89.0), (0.912386645, 89.5), (0.912289237, 90.0), (0.912191828, ...
            90.5), (
162     0.912787101, 91.0), (0.91340402, 91.5), (0.911325977, 92.0), ...
            (0.912321706,
163     92.5), (0.910763173, 93.0), (0.910590003, 93.5), (0.911488324, ...
            94.0), (
164     0.911347623, 94.5), (0.91095799, 95.0), (0.910709057, 95.5), ...
            (0.90978909,
165     96.0), (0.911185276, 96.5), (0.911748079, 97.0), (0.911390916, ...
            97.5), (
166     0.908879946, 98.0), (0.908609367, 98.5), (0.909529335, 99.0), ...
            (0.911228568,
167     99.5), (0.910947167, 100.0), (0.910286955, 100.5), (0.909085586, ...
            101.0), (
168     0.908934062, 101.5), (0.910514241, 102.0), (0.91095799, 102.5), (
169     0.910568357, 103.0), (0.908598544, 103.5), (0.907602815, 104.0), ...
            (
170     0.908793361, 104.5), (0.911185276, 105.0), (0.910535887, 105.5), ...
            (
171     0.908522782, 106.0), (0.90675861, 106.5), (0.907418822, 107.0), (
```

```
172     0.908425374, 107.5), (0.908533605, 108.0), (0.908760891, 108.5), ...
            (
173     0.907970802, 109.0), (0.907256474, 109.5), (0.906542147, 110.0), ...
            (
174     0.907830101, 110.5), (0.907938332, 111.0), (0.906812725, 111.5), ...
            (
175     0.906910134, 112.0), (0.906953426, 112.5), (0.907959979, 113.0), ...
            (
176     0.907667754, 113.5), (0.906845195, 114.0), (0.906682848, 114.5), ...
            (
177     0.906780256, 115.0), (0.907202359, 115.5), (0.90693178, 116.0), (
178     0.90693178, 116.5), (0.906293214, 117.0), (0.906401446, 117.5), (
179     0.906715317, 118.0), (0.90503773, 118.5), (0.905633003, 119.0), (
180     0.906888487, 119.5), (0.906466385, 120.0), (0.905427363, 120.5), ...
            (
181     0.904518219, 121.0), (0.904583158, 121.5), (0.905806173, 122.0), ...
            (
182     0.905351601, 122.5), (0.905535594, 123.0), (0.90503773, 123.5), (
183     0.905113492, 124.0), (0.904691389, 124.5), (0.904864559, 125.0), ...
            (
184     0.905860289, 125.5), (0.905059376, 126.0), (0.905026907, 126.5), ...
            (
185     0.904788797, 127.0), (0.905167607, 127.5), (0.905059376, 128.0), ...
            (
186     0.904886206, 128.5), (0.904788797, 129.0), (0.904972791, 129.5), ...
            (
187     0.90424764, 130.0), (0.903554959, 130.5), (0.903414258, 131.0), (
188     0.903370965, 131.5), (0.904312579, 132.0), (0.904258463, 132.5), ...
            (
189     0.902505114, 133.0), (0.902104658, 133.5), (0.90366319, 134.0), (
190     0.904334225, 134.5), (0.904096116, 135.0), (0.902461821, 135.5), ...
            (
191     0.90194231, 136.0), (0.902894747, 136.5), (0.903706483, 137.0), (
192     0.904442457, 137.5), (0.90366319, 138.0), (0.903089564, 138.5), (
193     0.902602522, 139.0), (0.903219441, 139.5), (0.903392612, 140.0), ...
            (
194     0.903186972, 140.5), (0.903208618, 141.0), (0.902515937, 141.5), ...
            (
195     0.901899018, 142.0), (0.90176914, 142.5), (0.902505114, 143.0), (
196     0.902072188, 143.5), (0.901920664, 144.0), (0.901314568, 144.5), ...
            (
197     0.901541854, 145.0), (0.901823256, 145.5), (0.901671732, 146.0), ...
            (
198     0.901639262, 146.5), (0.90290557, 147.0), (0.901953134, 147.5), (
199     0.901401153, 148.0), (0.90101152, 148.5), (0.901357861, 149.0), (
200     0.901866548, 149.5), (0.902223712, 150.0), (0.901985603, 150.5), ...
            (
201     0.901292922, 151.0), (0.900708472, 151.5), (0.900827526, 152.0), ...
            (
202     0.901292922, 152.5), (0.901000697, 153.0), (0.900308016, 153.5), ...
            (
203     0.90060024, 154.0), (0.900210607, 154.5), (0.900611064, 155.0), (
204     0.900513655, 155.5), (0.900297192, 156.0), (0.900113199, 156.5), ...
            (
205     0.90022143, 157.0), (0.900989874, 157.5), (0.89984262, 158.0), (
206     0.899756035, 158.5), (0.900059083, 159.0), (0.900827526, 159.5), ...
            (
```

```
207     0.900502832, 160.0), (0.900113199, 160.5), (0.899983321, 161.0), ...
            (
208     0.899983321, 161.5), (0.899810151, 162.0), (0.899788505, 162.5), ...
            (
209     0.899636981, 163.0), (0.899052531, 163.5), (0.899052531, 164.0), ...
            (
210     0.898987592, 164.5), (0.899258171, 165.0), (0.899020061, 165.5), ...
            (
211     0.899182409, 166.0), (0.898955122, 166.5), (0.898944299, 167.0), ...
            (
212     0.899279817, 167.5), (0.899052531, 168.0), (0.899344756, 168.5), ...
            (
213     0.899377225, 169.0), (0.899388048, 169.5), (0.89891183, 170.0), (
214     0.898576312, 170.5), (0.898511374, 171.0), (0.898727836, 171.5), ...
            (
215     0.898294911, 172.0), (0.897699638, 172.5), (0.897591406, 173.0), ...
            (
216     0.897602229, 173.5), (0.89815421, 174.0), (0.898468081, 174.5), (
217     0.898511374, 175.0), (0.89815421, 175.5), (0.898013509, 176.0), (
218     0.898349026, 176.5), (0.89832738, 177.0), (0.898338203, 177.5), (
219     0.897580583, 178.0), (0.897320828, 178.5), (0.897526467, 179.0), ...
            (
220     0.897104365, 179.5), (0.897461529, 180.0), (0.897201773, 180.5), ...
            (
221     0.89663897, 181.0), (0.897223419, 181.5), (0.897331651, 182.0), (
222     0.897634699, 182.5), (0.897883631, 183.0), (0.89815421, 183.5), (
223     0.89715848, 184.0), (0.896942018, 184.5), (0.896584854, 185.0), (
224     0.896141105, 185.5), (0.898089271, 186.0), (0.89850055, 186.5), (
225     0.896942018, 187.0), (0.895664887, 187.5), (0.895340192, 188.0), ...
            (
226     0.896465799, 188.5), (0.896487446, 189.0), (0.896942018, 189.5), ...
            (
227     0.896790494, 190.0), (0.895719002, 190.5), (0.896076166, 191.0), ...
            (
228     0.896498269, 191.5), (0.896714732, 192.0), (0.896833786, 192.5), ...
            (
229     0.896974487, 193.0), (0.896379214, 193.5), (0.896249336, 194.0), ...
            (
230     0.895783941, 194.5), (0.8966065, 195.0), (0.896108635, 195.5), (
231     0.894874797, 196.0), (0.895405131, 196.5), (0.894928913, 197.0), ...
            (
232     0.895437601, 197.5), (0.895621594, 198.0), (0.895708179, 198.5), ...
            (
233     0.895004675, 199.0), (0.895654063, 199.5), (0.895719002, 200.0), ...
            (
234     0.895719002, 200.5), (0.895913819, 201.0), (0.895881349, 201.5), ...
            (
235     0.895686533, 202.0), (0.895405131, 202.5), (0.895686533, 203.0), ...
            (
236     0.895275253, 203.5), (0.89509126, 204.0), (0.894311993, 204.5), (
237     0.895112906, 205.0), (0.894950559, 205.5), (0.894744919, 206.0), ...
            (
238     0.895069614, 206.5), (0.89374919, 207.0), (0.894052238, 207.5), (
239     0.89450681, 208.0), (0.894625865, 208.5), (0.894604218, 209.0), (
240     0.894571749, 209.5), (0.894214585, 210.0), (0.894387755, 210.5), ...
            (
```

```
241     0.894311993, 211.0), (0.894571749, 211.5), (0.894961382, 212.0), ...
            (
242     0.894788212, 212.5), (0.894420225, 213.0), (0.894658334, 213.5), ...
            (
243     0.894593395, 214.0), (0.894182116, 214.5), (0.893911537, 215.0), ...
            (
244     0.893857421, 215.5), (0.893586843, 216.0), (0.893413672, 216.5), ...
            (
245     0.893727544, 217.0), (0.893327087, 217.5), (0.893251325, 218.0), ...
            (
246     0.893208033, 218.5), (0.892948277, 219.0), (0.89333791, 219.5), (
247     0.893381203, 220.0), (0.893662605, 220.5), (0.893695074, 221.0), ...
            (
248     0.89319721, 221.5), (0.893500258, 222.0), (0.894182116, 222.5), (
249     0.894192939, 223.0), (0.893467788, 223.5), (0.893532727, 224.0), ...
            (
250     0.893067332, 224.5), (0.893272972, 225.0), (0.893132271, 225.5), ...
            (
251     0.893392026, 226.0), (0.893067332, 226.5), (0.892980747, 227.0), ...
            (
252     0.892677699, 227.5), (0.892731814, 228.0), (0.892601937, 228.5), ...
            (
253     0.892796753, 229.0), (0.891638677, 229.5), (0.891508799, 230.0), ...
            (
254     0.892298889, 230.5), (0.892850869, 231.0), (0.892645229, 231.5), ...
            (
255     0.892904985, 232.0), (0.892883338, 232.5), (0.893489434, 233.0), ...
            (
256     0.894615042, 233.5), (0.891075873, 234.0), (0.892840046, 234.5), ...
            (
257     0.891887609, 235.0), (0.891562915, 235.5), (0.893024039, 236.0), ...
            (
258     0.890534716, 236.5), (0.894398579, 237.0), (0.896530738, 237.5), ...
            (
259     0.892493705, 238.0), (0.890220845, 238.5), (0.896270983, 239.0), ...
            (
260     0.891920079, 239.5), (0.894907266, 240.0), (0.889095238, 240.5), ...
            (
261     0.889062768, 241.0), (0.890599655, 241.5), (0.891692793, 242.0), ...
            (
262     0.892158188, 242.5), (0.893478611, 243.0), (0.89051307, 243.5), (
263     0.894355286, 244.0), (0.893208033, 244.5), (0.895026321, 245.0), ...
            (
264     0.885577716, 245.5), (0.895102083, 246.0), (0.893933183, 246.5), ...
            (
265     0.891487153, 247.0), (0.886681677, 247.5), (0.889041122, 248.0), ...
            (
266     0.890870234, 248.5), (0.894128, 249.0), (0.894409402, 249.5),
267     (0.893500258,
268     250.0), (0.892277242, 250.5), (0.891216574, 251.0),
269     (0.889647218, 251.5), (
270     0.890653771, 252.0), (0.892753461, 252.5), (0.891530445, 253.0), ...
            (
271     0.888196917, 253.5), (0.89085941, 254.0), (0.891292336, 254.5), (
272     0.891194928, 255.0), (0.890707886, 255.5), (0.892482882, 256.0), ...
            (
```

```
273     0.892298889, 256.5), (0.889452402, 257.0), (0.886540976, 257.5), ...
            (
274     0.889062768, 258.0), (0.892071603, 258.5), (0.892093249, 259.0), ...
            (
275     0.889744626, 259.5), (0.889452402, 260.0), (0.888727251, 260.5), ...
            (
276     0.88972298, 261.0), (0.889766273, 261.5), (0.891292336, 262.0), (
277     0.892526175, 262.5), (0.890567185, 263.0), (0.88951734, 263.5), (
278     0.888337618, 264.0), (0.890794472, 264.5), (0.891465506, 265.0), ...
            (
279     0.887839753, 265.5), (0.890902703, 266.0), (0.890220845, 266.5), ...
            (
280     0.889441578, 267.0), (0.891010934, 267.5), (0.88779646, 268.0), (
281     0.887460943, 268.5), (0.89085941, 269.0), (0.891335629, 269.5), (
282     0.890534716, 270.0), (0.888467495, 270.5), (0.888088685, 271.0), ...
            (
283     0.890545539, 271.5), (0.89085941, 272.0), (0.891119166, 272.5), (
284     0.890339899, 273.0), (0.887471766, 273.5), (0.887688229, 274.0), ...
            (
285     0.889181823, 274.5), (0.890794472, 275.0), (0.889041122, 275.5), ...
            (
286     0.88782893, 276.0), (0.8866925, 276.5), (0.88913853, 277.0),
287     (0.890632124,
288     277.5), (0.890188375, 278.0), (0.888846305, 278.5),
289     (0.888683958, 279.0), (
290     0.889831212, 279.5), (0.889452402, 280.0), (0.890253314, 280.5), ...
            (
291     0.889647218, 281.0), (0.889062768, 281.5), (0.888922067, 282.0), ...
            (
292     0.889766273, 282.5), (0.884614456, 283.0), (0.889235939, 283.5), ...
            (
293     0.896097812, 284.0), (0.905156784, 284.5), (0.89640086, 285.0), (
294     0.874722102, 285.5), (0.897180127, 286.0), (0.890599655, 286.5), ...
            (
295     0.889852858, 287.0), (0.885805002, 287.5), (0.889874504, 288.0), ...
            (
296     0.889127707, 288.5), (0.891129989, 289.0), (0.887363535, 289.5), ...
            (
297     0.886129696, 290.0), (0.886346159, 290.5), (0.888045393, 291.0), ...
            (
298     0.889398286, 291.5), (0.889777096, 292.0), (0.887991277, 292.5), ...
            (
299     0.887266126, 293.0), (0.887471766, 293.5), (0.888575727, 294.0), ...
            (
300     0.889160177, 294.5), (0.889203469, 295.0), (0.888175271, 295.5), ...
            (
301     0.887753168, 296.0), (0.887504236, 296.5), (0.887969631, 297.0), ...
            (
302     0.888608196, 297.5), (0.888099509, 298.0), (0.887579998, 298.5), ...
            (
303     0.887569174, 298.98)))
304
305 if prony == 1:
306  mdb.models['Uniaxial'].materials['Viscoelastic'].viscoelastic....
        setValues(domain=
307   TIME, table=((g1, 0.0, t1), (g2, 0.0, t2), (g3, 0.0, t3)), time=...
        PRONY)
```

```
308
309 # ------------- Create and assign section -------------------------
310
311 mdb.models['Uniaxial'].HomogeneousSolidSection(material='Viscoelastic...
        ', name=
312     'Phantom', thickness=None)
313
314 mdb.models['Uniaxial'].parts['Part-1'].SectionAssignment(offset=0.0,
315     offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
316     cells=mdb.models['Uniaxial'].parts['Part-1'].cells.findAt(((0.5*a...
            , 0.5*b,
317     0),),)), sectionName='Phantom', thicknessAssignment=...
            FROM_SECTION)
318
319 # ---------------- Create partitions ------------------------------
320
321 mdb.models['Uniaxial'].ConstrainedSketch(gridSpacing=0.004, name='...
        __profile__',
322     sheetSize=a, transform=
323     mdb.models['Uniaxial'].parts['Part-1'].MakeSketchTransform(
324     sketchPlane=mdb.models['Uniaxial'].parts['Part-1'].faces.findAt...
            ((0.0,
325     0.5*b, 0.5*c),), sketchPlaneSide=SIDE1,
326     sketchUpEdge=mdb.models['Uniaxial'].parts['Part-1'].edges.findAt...
            ((0.0,
327     b, 0.25*c),), sketchOrientation=TOP, origin=(0.0, 0.5*b,
328     0.5*c)))
329 mdb.models['Uniaxial'].sketches['__profile__'].sketchOptions....
        setValues(
330     decimalPlaces=3)
331 mdb.models['Uniaxial'].parts['Part-1'].projectReferencesOntoSketch(...
        filter=
332     COPLANAR_EDGES, sketch=mdb.models['Uniaxial'].sketches['...
            __profile__'])
333 mdb.models['Uniaxial'].sketches['__profile__'].Line(point1=(0.0, 0.5*...
        b),
334     point2=(0.0, -0.5*b))
335 mdb.models['Uniaxial'].sketches['__profile__'].Line(point1=(-0.5*c, ...
        0.0),
336     point2=(0.5*c, 0.0))
337 mdb.models['Uniaxial'].parts['Part-1'].PartitionFaceBySketch(faces=
338     mdb.models['Uniaxial'].parts['Part-1'].faces.findAt(((0.0, 0.5*b,
339     0.5*c),)), sketch=mdb.models['Uniaxial'].sketches['__profile__'...
            ],
340     sketchOrientation=TOP, sketchUpEdge=
341     mdb.models['Uniaxial'].parts['Part-1'].edges.findAt((0.0, b,
342     0.25*c),))
343
344 mdb.models['Uniaxial'].ConstrainedSketch(gridSpacing=0.004, name='...
        __profile__',
345     sheetSize=a, transform=
346     mdb.models['Uniaxial'].parts['Part-1'].MakeSketchTransform(
347     sketchPlane=mdb.models['Uniaxial'].parts['Part-1'].faces.findAt((...
            a,
348     0.5*b, 0.5*c),), sketchPlaneSide=SIDE1,
349     sketchUpEdge=mdb.models['Uniaxial'].parts['Part-1'].edges.findAt...
            ((a,
```

```
350     b, 0.25*c), ), sketchOrientation=TOP, origin=(0.0, 0.5*b,
351     0.5*c)))
352 mdb.models['Uniaxial'].sketches['__profile__'].sketchOptions....
        setValues(
353     decimalPlaces=3)
354 mdb.models['Uniaxial'].parts['Part-1'].projectReferencesOntoSketch(...
        filter=
355     COPLANAR_EDGES, sketch=mdb.models['Uniaxial'].sketches['...
            __profile__'])
356 mdb.models['Uniaxial'].sketches['__profile__'].Line(point1=(0.0, 0.5*...
        b),
357     point2=(0.0, -0.5*b))
358 mdb.models['Uniaxial'].sketches['__profile__'].Line(point1=(-0.5*c, ...
        0.0),
359     point2=(0.5*c, 0.0))
360 mdb.models['Uniaxial'].parts['Part-1'].PartitionFaceBySketch(faces=
361     mdb.models['Uniaxial'].parts['Part-1'].faces.findAt(((a, 0.5*b,
362     0.5*c), )), sketch=mdb.models['Uniaxial'].sketches['__profile__'...
            ],
363     sketchOrientation=TOP, sketchUpEdge=
364     mdb.models['Uniaxial'].parts['Part-1'].edges.findAt((a, b,
365     0.25*c), ))
366
367 # ------------------- Create sets -----------------------------------
368
369 mdb.models['Uniaxial'].parts['Part-1'].Set(faces=
370     mdb.models['Uniaxial'].parts['Part-1'].faces.getByBoundingBox...
            (-0.1*a, -0.1*b,
371     -0.1*c, 0.1*a, 1.1*b, 1.1*c), name='MountedEnd')
372
373 mdb.models['Uniaxial'].parts['Part-1'].Set(faces=
374     mdb.models['Uniaxial'].parts['Part-1'].faces.getByBoundingBox...
            (0.95*a, -0.1*b,
375     -0.1*c, 1.05*a, 1.1*b, 1.1*c), name='DisplacedEnd')
376
377 mdb.models['Uniaxial'].parts['Part-1'].Set(name='MidNode_DisplacedEnd...
        ', vertices=
378     mdb.models['Uniaxial'].parts['Part-1'].vertices.findAt(((a, 0.5*b...
            ,
379     0.5*c), )))
380
381 mdb.models['Uniaxial'].parts['Part-1'].Set(name='MidNode_MountedEnd',...
         vertices=
382     mdb.models['Uniaxial'].parts['Part-1'].vertices.findAt(((0, 0.5*b...
            ,
383     0.5*c), )))
384
385 # ------------------- Create assembly -------------------------------
386
387 mdb.models['Uniaxial'].rootAssembly.DatumCsysByDefault(CARTESIAN)
388 mdb.models['Uniaxial'].rootAssembly.Instance(dependent=ON, name='Part...
        -1-1',
389     part=mdb.models['Uniaxial'].parts['Part-1'])
390
391 # ------------------- Mesh ------------------------------------------
392
393 mdb.models['Uniaxial'].parts['Part-1'].seedPart(deviationFactor=0.1,
```

```
394     minSizeFactor=0.1, size=elSize)
395 mdb.models['Uniaxial'].parts['Part-1'].generateMesh()
396
397 mdb.models['Uniaxial'].parts['Part-1'].setElementType(elemTypes=(...
    ElemType(
398     elemCode=elType, elemLibrary=STANDARD), ElemType(elemCode=C3D15,
399     elemLibrary=STANDARD), ElemType(elemCode=C3D10, elemLibrary=...
        STANDARD)),
400     regions=(mdb.models['Uniaxial'].parts['Part-1'].cells.findAt...
        (((0.5*a,
401     0.5*b, 0), )), ))
402
403 # ------------------ Create Steps ------------------------------------
404
405 mdb.models['Uniaxial'].Stress(distributionType=UNIFORM, name='...
    OffsetStress',
406     region=Region(
407     cells=mdb.models['Uniaxial'].rootAssembly.instances['Part-1-1'].....
        cells.findAt(
408     ((0.0, 0.009567, 0.001167), ), )), sigma11=preStress, sigma12...
        =0.0, sigma13=0.0
409     , sigma22=0.0, sigma23=0.0, sigma33=0.0)
410
411 if viscoLoad == 1:
412  mdb.models['Uniaxial'].ViscoStep(description=
413     'This step is for the loading of the specimen up until the ...
        relaxation strain is reached.'
414     , cetol=CEtol, initialInc=dtLoading, maxInc=dtLoading,
415     maxNumInc=1000, minInc=0.001*dtLoading, name='Loading', nlgeom=ON...
        , previous='Initial',
416     timePeriod=timeLoading)
417
418
419 if viscoLoad == 0:
420  mdb.models['Uniaxial'].StaticStep(description=
421     'This step is for the loading of the specimen up until the ...
        relaxation strain is reached.'
422     , initialInc=dtLoading, maxInc=dtLoading, maxNumInc=1000, minInc...
        =0.001*dtLoading,
423     name='Loading', nlgeom=ON, previous='PreLoad', timePeriod=...
        timeLoading)
424
425 mdb.models['Uniaxial'].ViscoStep(cetol=CEtol, description=
426     'This step is for the relaxation of the stress as the ...
        longitudinal strain is kept constant',
427     initialInc=dtRelax, maxInc=dtRelax, maxNumInc=10000,
428     minInc=0.00001, name='Relaxation', previous='Loading', timePeriod...
        =timeRelax)
429
430 # ------------------ Output values -----------------------------------
431
432 del mdb.models['Uniaxial'].fieldOutputRequests['F-Output-1']
433 del mdb.models['Uniaxial'].historyOutputRequests['H-Output-1']
434
435 mdb.models['Uniaxial'].FieldOutputRequest(createStepName='Loading', ...
    name=
436     'Loading', variables=('S', 'U', 'RF'))
```

```
437 mdb.models['Uniaxial'].FieldOutputRequest(createStepName='Relaxation'...
      , name=
438     'Relaxation', variables=('S', 'U', 'RF'))
439
440 mdb.models['Uniaxial'].HistoryOutputRequest(createStepName='Loading',...
      name=
441     'DisplacedEnd', rebar=EXCLUDE, region=
442     mdb.models['Uniaxial'].rootAssembly.instances['Part-1-1'].sets['...
          DisplacedEnd']
443     , sectionPoints=DEFAULT, variables=('S11', 'E11', 'LE11', 'CEEQ')...
          )
444
445 mdb.models['Uniaxial'].HistoryOutputRequest(createStepName='Loading',...
      name=
446     'MountedEnd', rebar=EXCLUDE, region=
447     mdb.models['Uniaxial'].rootAssembly.instances['Part-1-1'].sets['...
          MountedEnd']
448     , sectionPoints=DEFAULT, variables=('RF1',))
449
450 mdb.models['Uniaxial'].HistoryOutputRequest(createStepName='Loading',...
      name=
451     'MidNode_DisplacedEnd', rebar=EXCLUDE, region=
452     mdb.models['Uniaxial'].rootAssembly.instances['Part-1-1'].sets['...
          MidNode_DisplacedEnd']
453     , sectionPoints=DEFAULT, variables=('U1',))
454
455 # ---------- Loading and BC's ----------
456
457 if clamped == 0:
458  mdb.models['Uniaxial'].DisplacementBC(amplitude=UNSET, ...
      createStepName='Initial'
459     , distributionType=UNIFORM, fieldName='', localCsys=None, name=
460     'FasteningPoints', region=
461     mdb.models['Uniaxial'].rootAssembly.instances['Part-1-1'].sets['...
          MountedEnd']
462     , u1=SET, u2=UNSET, u3=UNSET, ur1=SET, ur2=SET, ur3=SET)
463
464 if clamped == 0:
465  mdb.models['Uniaxial'].DisplacementBC(amplitude=UNSET, ...
      createStepName='Initial'
466      , distributionType=UNIFORM, fieldName='', localCsys=None, name='...
          Fixed_u3',
467      region=Region(
468      vertices=mdb.models['Uniaxial'].rootAssembly.instances['Part-1-1...
          '].vertices.findAt(
469      ((0.0, b, 0.5*c), ), ((0.0, 0.0, 0.5*c), ), )), u1=UNSET, u2=...
          UNSET,
470     u3=SET, ur1=UNSET, ur2=UNSET, ur3=UNSET)
471
472 if clamped == 0:
473  mdb.models['Uniaxial'].DisplacementBC(amplitude=UNSET, ...
      createStepName='Initial'
474      , distributionType=UNIFORM, fieldName='', localCsys=None, name='...
          Fixed_u2',
475      region=Region(
476      vertices=mdb.models['Uniaxial'].rootAssembly.instances['Part-1-1...
          '].vertices.findAt(
```

```
477       ((0.0, 0.5*b, 0), ), ((0.0, 0.5*b, c), ), )), u1=UNSET, u2=SET,
478      u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=UNSET)
479
480 if clamped == 1:
481  mdb.models['Uniaxial'].DisplacementBC(amplitude=UNSET, ...
         createStepName='Initial'
482      , distributionType=UNIFORM, fieldName='', localCsys=None, name=
483      'FasteningPoints', region=
484      mdb.models['Uniaxial'].rootAssembly.instances['Part-1-1'].sets['...
             MountedEnd']
485      , u1=SET, u2=SET, u3=SET, ur1=SET, ur2=SET, ur3=SET)
486
487 mdb.models['Uniaxial'].DisplacementBC(amplitude=UNSET, createStepName...
        ='Loading'
488      , distributionType=UNIFORM, fieldName='', fixed=OFF, localCsys=...
             None, name=
489      'UniaxialLoading', region=
490      mdb.models['Uniaxial'].rootAssembly.instances['Part-1-1'].sets['...
             DisplacedEnd']
491      , u1=dispLoad, u2=UNSET, u3=UNSET, ur1=UNSET, ur2=UNSET, ur3=...
             UNSET)
492
493 # ------------------- Create job ------------------------------------
494
495 mdb.models['Uniaxial'].rootAssembly.regenerate()
496
497 if clamped == 0:
498  jobName = jobName + '_simplesupport'
499
500 if clamped == 1:
501  jobName = jobName + '_clamped'
502
503 mdb.Job(atTime=None, contactPrint=OFF, description='', echoPrint=OFF,
504      explicitPrecision=SINGLE, getMemoryFromAnalysis=True, ...
             historyPrint=OFF,
505      memory=percMemory, memoryUnits=PERCENTAGE, model='Uniaxial', ...
             modelPrint=OFF,
506      multiprocessingMode=DEFAULT, name=jobName, nodalOutputPrecision=
507      SINGLE, numCpus=1, queue=None, scratch='', type=ANALYSIS, ...
             userSubroutine=''
508      , waitHours=0, waitMinutes=0)
509
510 mdb.models['Uniaxial'].keywordBlock.synchVersions(...
        storeNodesAndElements=False)
511 mdb.models['Uniaxial'].keywordBlock.replace(38,
512      '\n*Initial Conditions, type=STRESS, UNBALANCED STRESS=STEP\...
             n_PickedSet4, 1520.5, 0., 0., 0., 0., 0.')
513
514 if saveModel == 1:
515  mdb.saveAs(pathName=workDirectory + '/' + caeName)
```

B.2 planeStrainPlate.py

```
# Written by: Lars Edvard Bryhni Daehli

# ----- Initializing ----- #

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 ------------------

workDirectory = r'D:\NTNU\Master\Simulations\2D_PlaneStrain'
caeName = 'planeStrainPlate.cae'
modelName = 'planeStrainPlate'

os.chdir(workDirectory)

# ----------------- Defining analysis ---------------------

makeJob_1 = 1      # 1 -> Creates the first job to write input from
makeJob_2 = 1    # 1 -> Creates the second job to change stack ...
    direction
includeShear = 0  # 1 -> Generates a shear force. Abaqus tends to ...
    shut down if
      #       this is included the first time the script is started.
includeARF = 1     # 1 -> Generates an acoustic radiation force ...
    excitation
saveModel = 1    # 1 -> CAE-model saved with given caeName
rollingUpper = 0  # 1 -> Roller supports on upper boundary
rollingLower = 0  # 1 -> Roller supports on lower boundary
viscoElastic = 1  # 1 -> Include a viscoelastic material definition ...
    based on
      #       time-domain prony coefficients.

# NB: Cannot use both implicit and explicit!

explicit = 1     # 1 -> Explicit analysis
implicit = 0     # 1 -> Implicit analysis

# ------------------ Defining variables -------------------
# |--- REMARK! ---|
# If the geometry is changed, then the element stack direction will ...
    not
# change automatically by running the script. I could not figure out ...
    a smart
```

```
52 # way to handle this, and would not spend too much time on it. So, if... any of
53 # the constants a,b,c,d,f,E0,nu0,rho0,n,Xe or Ye are changed, the ... infinite
54 # elements will no longer be automatically generated by the script.
55 # Then you have to change the mesh stack direction manually in #Edit:
56 # Mesh Stack Orientation". Make it point outwards from the tissue ... region. Load the
57 # input file in e.g TextPad, and change elements CPS4R to CINPE4. ... Import the
58 # input file to Abaqus and create a job with the input file you just ... imported!
59
60 # Geometry
61 a = 20E-03    #Width [m]
62 b = 60E-03    #Height [m] (Including the infinite elements)
63 c = 10E-03    #Distance from centre to outer partition
64       #NB: Make sure that the length of the infinite elements
65       #are as long as the height of the finite region, b. This
66       #is ensured as long as b = 6*c.
67 d = 8E-03   #Width of TOTAL transducer (Only modelling half)
68 f = 0.05*d    #Focus size
69 Xe = 1.5E-03    #X-value of ellipse [m]
70 Ye = 1.5E-03    #Y-value of ellipse [m]
71
72 n = 10      #Controls mesh size. It is the number of elements
73       #the impulse load should span. Set n=6,7,.. or 10.
74 # Material Constants
75 E0 = 107585   #Instantaneous E-modulus of soft tissue [Pa]
76 nu0 = 0.499   #Poisson's ratio of soft tissue [Pa]
77 rho0 = 1060   #Density of Soft Tissue
78 g1 = 9.45749E-03    #Prony (relaxation) coefficients
79 g2 = 3.84291E-02    #for the viscoelastic material
80 g3 = 6.46134E-02        #are g1,g2,g3,tau1,tau2 and tau3
81 tau1 = 1.5178
82 tau2 = 11.541
83 tau3 = 93.977
84 E_t = 3*E0    #Elastic modulus of tumor [m]
85 nu_t = nu0    #Poisson's ratio of tumor [m]
86 rho_t = rho0    #Density of tumor
87
88 # Step values
89 Ta = 4E-3   #Analysis time [s]
90 Ti = 250E-6   #Impulse time [s]
91 linBulk = 0.06    #Linear bulk viscosity parameter
92 quadBulk = 0.0    #Quadratic bulk viscosity parameter
93
94 shearForce = 1    #Magnitude of shear traction force [Pa].
95       #Valid for both ARF and edge shear force!
96 numField = 700    #Number of field output frames
97 numHist = 700   #Number of history output
98
99 elType = CPE4R    #Choose element type for explicit analysis
100 elTypeImplicit = CPE8 #Choose element type for implicit analysis
101 meshTech = STRUCTURED #Choose meshing technique (FREE/STRUCTURED (... default))
102 meshTech_ARF = FREE #Choose meshing technique for the ARF-region
```

```
103 minSize = 0.95          #Minimum size factor for elements
104
105 # -- Curve fitting of the body force from ultrasound field simulation...
        --
106
107 #Using a Gaussian function to fit the body force distribution
108 #from the ultrasound field cftool(x,y) in Matlab is used for the
109 #curve fitting. This is done in the matlab script
110 #two_dim_linarray.m.
111 #
112 #F(Y) = a1*exp(-((b1*Y)/c1)^2) + ... + a3*exp(-((b3*Y)/c3)^2)
113
114 a1 = 6.564E5
115 b1 = 0.009823
116 c1 = 0.001613
117 a2 = -1.494E5
118 b2 = 0.01506
119 c2 = 0.00394
120 a3 = 3.417E5
121 b3 = 0.01148
122 c3 = 0.007131
123
124 # ---------------- Calculating central constants -------
125
126 #Longitudinal velocity of soft tissue [m/s]
127 cl_1 = sqrt(E0*(1-nu0)/(rho0*(1-2*nu0)*(1+nu0)))
128 #Transversal velocity of soft tissue [m/s]
129 ct_1 = sqrt(E0/(2*rho0*(1+nu0)))
130 #Longitudinal velocity of tumor [m/s]
131 cl_2 = sqrt(E_t*(1-nu_t)/(rho_t*(1-2*nu_t)*(1+nu_t)))
132 #Transversal velocity of tumor [m/s]
133 ct_2 = sqrt(E_t/(2*rho_t*(1+nu_t)))
134
135 # ---------------- Defining the mesh density ----------
136
137 # The script is already optimized for n = 3 -> n = 10. If any of ...
        these values,
138 # or the ones in between, are selected the script
139 #  makes sure that the elements governing the inifnite elements are
140 # correctly oriented according to the definition in the Abaqus manual...
        .
141 # The infinite elements must "point" in the infinite
142 # direction - This is not the case!
143 #
144 # Remember to open the Job_Inputfile_ANALYSISNAME.inp and find
145 # the elements defined as *Element, type=CPS3R and change them
146 # to type=CINPE4. This must be done in order to use
147 # infinite elements.
148
149 #Distance travelled by the shear wave within the impulse time [m]
150 dx = ct_1 * Ti
151 #Approximate global element size [m]
152 elSize = dx/n      #Changes mesh density with Ti,E0...
153 #elSize = 0.14E-03    #El.size for n=10,E0=107585,Ti=250E-6
154 elSize_inclusion = 1.3*elSize
155 #Approximate element size for inclusion [m]
156
```

```
157 # ------------------ Time incrementation ------------------
158
159 #Only to be used in implicit analysis
160
161 dt = 2E-05
162 dt_min = 1.0E-07
163
164 print '-------------------------------------------------------'
165 print ' * Width = ' + repr(1000*a) + ' mm'
166 print ' * Height = ' + repr(1000*(2*c)) + ' mm'
167 print ' * Ellipse X-value = ' + repr(1000*Xe) + ' mm'
168 print ' * Ellipse Y-value = ' + repr(1000*Ye) + ' mm'
169 print ' * Approximate element size = ' + repr(1000*elSize) + ' mm'
170 print ' * Elastic Modulus of Tissue = ' + repr(E0/1000) + ' kPa'
171 print ' * Elastic Modulus of Tumor = ' + repr(E_t/1000) + ' kPa'
172 print ' * Shear speed in tissue = ' + repr(ct_1) + ' m/s'
173 print ' * Longitudinal speed in tissue = ' + repr(cl_1) + ' m/s'
174 print ' * Shear speed in tumor = ' + repr(ct_2) + ' m/s'
175 print ' * Longitudinal speed in tumor = ' + repr(cl_2) + ' m/s'
176 print '-------------------------------------------------------'
177
178 # ------------------------ Creating model ------------------
179 mdb.Model(name=modelName)
180 mdb.models[modelName].setValues(noPartsInputFile=ON)
181
182 # ---------------------- Creating the geometry ---------------
183
184 mdb.models[modelName].ConstrainedSketch(name='__profile__',
185     sheetSize=2*a)
186 mdb.models[modelName].sketches['__profile__'].rectangle(point1=(-0.5*...
    a,
187     -0.5*b), point2=(0.5*a, 0.5*b))
188 mdb.models[modelName].Part(dimensionality=TWO_D_PLANAR, name='Plate',
189     type=DEFORMABLE_BODY)
190 mdb.models[modelName].Part(dimensionality=TWO_D_PLANAR, name='Plate',
191     type=DEFORMABLE_BODY)
192 mdb.models[modelName].parts['Plate'].BaseShell(sketch=
193     mdb.models[modelName].sketches['__profile__'])
194
195 # ---- Creating an elliptical partition with lines ----
196
197 mdb.models[modelName].ConstrainedSketch(gridSpacing=0.1*Xe, name=
198     '__profile__', sheetSize=2*a, transform=
199     mdb.models[modelName].parts['Plate'].MakeSketchTransform(
200     sketchPlane=mdb.models[modelName].parts['Plate'].faces.findAt((
201     -a/6, -b/6, 0.0), (0.0, 0.0, 1.0)), sketchPlaneSide=SIDE1,
202     sketchOrientation=RIGHT, origin=(0.0, 0.0, 0.0)))
203 mdb.models[modelName].parts['Plate'].projectReferencesOntoSketch(
204     filter=COPLANAR_EDGES, sketch=
205     mdb.models[modelName].sketches['__profile__'])
206 mdb.models[modelName].sketches['__profile__'].
    EllipseByCenterPerimeter(
207     axisPoint1=(Xe, 0.0), axisPoint2=(0.0, Ye), center=(0.0, 0.0))
208 mdb.models[modelName].sketches['__profile__'].Line(point1=(-Xe,
209     0.0), point2=(Xe, 0.0))
210 mdb.models[modelName].sketches['__profile__'].Line(point1=(0.0,
211     Ye), point2=(0.0, -Ye))
```

```
212 mdb.models[modelName].parts['Plate'].PartitionFaceBySketch(faces=
213     mdb.models[modelName].parts['Plate'].faces.findAt(((-a/6,
214     -b/6, 0.0), )), sketch=
215     mdb.models[modelName].sketches['__profile__'])
216
217 # ---------- Partitioning the soft tissue ----------
218
219 # 1: Choosing the face to partition
220
221 mdb.models[modelName].ConstrainedSketch(gridSpacing=0.1*Xe, name=
222     '__profile__', sheetSize=2*a, transform=
223     mdb.models[modelName].parts['Plate'].MakeSketchTransform(
224     sketchPlane=mdb.models[modelName].parts['Plate'].faces.findAt((
225     0.45*a, -0.8*c, 0.0), (0.0, 0.0, 1.0)), sketchPlaneSide=SIDE1,
226     sketchOrientation=RIGHT, origin=(0.0, 0.0, 0.0)))
227 mdb.models[modelName].parts['Plate'].projectReferencesOntoSketch(
228     filter=COPLANAR_EDGES, sketch=
229     mdb.models[modelName].sketches['__profile__'])
230
231 # 2: Create partitions for finite and infinite elements
232
233 mdb.models[modelName].sketches['__profile__'].Line(point1=(-0.5*a,
234     c), point2=(0.5*a, c))
235 mdb.models[modelName].sketches['__profile__'].Line(point1=(-0.5*a,
236     -c), point2=(0.5*a, -c))
237
238 mdb.models[modelName].sketches['__profile__'].Line(point1=(0,
239     c), point2=(0, Ye))
240 mdb.models[modelName].sketches['__profile__'].Line(point1=(0,
241     -c), point2=(0, -Ye))
242
243 mdb.models[modelName].sketches['__profile__'].Line(point1=(-0.5*a,
244     0), point2=(-Xe, 0))
245 mdb.models[modelName].sketches['__profile__'].Line(point1=(Xe,
246     0), point2=(0.5*a, 0))
247
248 mdb.models[modelName].sketches['__profile__'].Line(point1=
249     ((-0.5*a)+(0.5*d), c), point2=(-0.5*a+f, f))
250
251 mdb.models[modelName].sketches['__profile__'].Line(point1=
252     (-0.5*a+f, f), point2=(-0.5*a+f, -f))
253
254 mdb.models[modelName].sketches['__profile__'].Line(point1=
255     (-0.5*a+f, -f), point2=((-0.5*a)+(0.5*d), -c))
256
257 mdb.models[modelName].sketches['__profile__'].Line(point1=
258     (-0.5*(0.5*a+Xe-f), c), point2=(-0.5*(0.5*a+Xe-f), -c))
259
260 mdb.models[modelName].sketches['__profile__'].Line(point1=
261     (0.5*(0.5*a+Xe), c), point2=(0.5*(0.5*a+Xe), -c))
262
263 mdb.models[modelName].parts['Plate'].PartitionFaceBySketch(faces=
264      mdb.models[modelName].parts['Plate'].faces.findAt(((0.45*a,
265      -0.8*c, 0.0), )), sketch=
266     mdb.models[modelName].sketches['__profile__'])
267
268 mdb.models[modelName].parts['Plate'].PartitionEdgeByPoint(edge=
```

```
269     mdb.models[modelName].parts['Plate'].edges.findAt((-0.5*(0.5*a+Xe...
            -f),
270     0.0, 0.0),), point=
271     mdb.models[modelName].parts['Plate'].InterestingPoint(
272     mdb.models[modelName].parts['Plate'].edges.findAt((-0.5*(0.5*a+Xe...
            -f),
273     0.0, 0.0),), MIDDLE))
274
275 mdb.models[modelName].parts['Plate'].PartitionEdgeByPoint(edge=
276     mdb.models[modelName].parts['Plate'].edges.findAt((0.5*(0.5*a+Xe)...
            ,
277     0.0, 0.0),), point=
278     mdb.models[modelName].parts['Plate'].InterestingPoint(
279     mdb.models[modelName].parts['Plate'].edges.findAt((0.5*(0.5*a+Xe)...
            ,
280     0.0, 0.0),), MIDDLE))
281
282 mdb.models[modelName].parts['Plate'].PartitionEdgeByPoint(edge=
283     mdb.models[modelName].parts['Plate'].edges.findAt((-0.5*a+0.5*f, ...
            0.0,
284     0.0),), point=
285     mdb.models[modelName].parts['Plate'].InterestingPoint(
286     mdb.models[modelName].parts['Plate'].edges.findAt((-0.5*a+0.5*f, ...
            0.0,
287     0.0),), MIDDLE))
288
289 # ------------------------ Material ------------------------
290
291 # 1:Soft Tissue
292
293 if viscoElastic == 0:
294  mdb.models[modelName].Material(name='SoftTissue')
295  mdb.models[modelName].materials['SoftTissue'].Elastic(moduli=...
        INSTANTANEOUS,
296  table=((E0, nu0),))
297  mdb.models[modelName].materials['SoftTissue'].Density(table=((rho0,
298      ),))
299
300 if viscoElastic == 1:
301  mdb.models[modelName].Material(name='SoftTissue')
302  mdb.models[modelName].materials['SoftTissue'].Elastic(moduli=...
        INSTANTANEOUS,
303  table=((E0, nu0),))
304  mdb.models[modelName].materials['SoftTissue'].Density(table=((rho0,
305      ),))
306  mdb.models[modelName].materials['SoftTissue'].Viscoelastic(
307     domain=TIME, table=((g1, 0.0, tau1), (g2, 0.0, tau2),
308     (g3, 0.0, tau3)), time=PRONY)
309
310 # 2:Tumor
311
312 mdb.models[modelName].Material(name='Tumor')
313 mdb.models[modelName].materials['Tumor'].Elastic(moduli=INSTANTANEOUS...
        ,
314 table=((E_t, nu_t),))
315 mdb.models[modelName].materials['Tumor'].Density(table=((rho_t,
316     ),))
```

```
317
318 # 3:Material in the infinite element region must be elastic and
319 #   match the elastic part of the viscoelastic material.
320 mdb.models[modelName].Material(name='InfiniteMaterial')
321 mdb.models[modelName].materials['InfiniteMaterial'].Elastic(moduli=...
      INSTANTANEOUS,
322  table=((E0, nu0), ))
323 mdb.models[modelName].materials['InfiniteMaterial'].Density(table=((...
      rho0,
324      ), ))
325
326 # ---------------------- Creating sections -----------------
327
328 mdb.models[modelName].HomogeneousSolidSection(material='SoftTissue',
329      name='SoftTissue', thickness=None)
330 mdb.models[modelName].HomogeneousSolidSection(material='Tumor',
331      name='Tumor', thickness=None)
332 mdb.models[modelName].HomogeneousSolidSection(material='...
      InfiniteMaterial',
333      name='Infinite', thickness=None)
334
335 # ---------------------- Assign sections -----------------
336
337 mdb.models[modelName].parts['Plate'].SectionAssignment(offset=0.0,
338      offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
339      faces=mdb.models[modelName].parts['Plate'].faces.getByBoundingBox...
          (
340     -0.55*a,-1.1*c,-0.1,0.55*a,1.1*c,1.1), ), sectionName='SoftTissue'...
          ,
341      thicknessAssignment=FROM_SECTION)
342
343 mdb.models[modelName].parts['Plate'].SectionAssignment(offset=0.0,
344      offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
345      faces=mdb.models[modelName].parts['Plate'].faces.getByBoundingBox...
          (
346     -1.1*Xe,-1.1*Ye,-0.1,1.1*Xe,1.1*Ye,1.1), ), sectionName='Tumor',
347      thicknessAssignment=FROM_SECTION)
348
349 mdb.models[modelName].parts['Plate'].SectionAssignment(offset=0.0,
350      offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
351      faces=mdb.models[modelName].parts['Plate'].faces.getByBoundingBox...
          (
352     -0.55*a,-0.55*b,-0.1,0.55*a,-0.95*c,1.1), ), sectionName='Infinite...
          ',
353      thicknessAssignment=FROM_SECTION)
354
355 mdb.models[modelName].parts['Plate'].SectionAssignment(offset=0.0,
356      offsetField='', offsetType=MIDDLE_SURFACE, region=Region(
357      faces=mdb.models[modelName].parts['Plate'].faces.getByBoundingBox...
          (
358     -0.55*a,0.95*c,-0.1,0.55*a,0.55*b,1.1), ), sectionName='Infinite',
359      thicknessAssignment=FROM_SECTION)
360
361 # ---------------------- Creating assembly -----------------
362
363 mdb.models[modelName].rootAssembly.DatumCsysByDefault(CARTESIAN)
364 mdb.models[modelName].rootAssembly.Instance(dependent=ON, name=
```

```
365     'Plate-1', part=mdb.models[modelName].parts['Plate'])
366
367 # ---------------------- Create sets ----------------------
368 mdb.models[modelName].parts['Plate'].Set(edges=
369     mdb.models[modelName].parts['Plate'].edges.findAt(((a/2, 0.5*c,
370     0.0), ), ((a/2, -0.5*c, 0.0), ), ), name='Fixed')
371 mdb.models[modelName].parts['Plate'].Set(edges=
372     mdb.models[modelName].parts['Plate'].edges.findAt(((-a/2, 0.5*c,
373     0.0), ), ((-a/2, -0.5*c, 0.0), ), ), name='Loading')
374 mdb.models[modelName].parts['Plate'].Set(edges=
375     mdb.models[modelName].parts['Plate'].edges.findAt(((a/4, c,
376     0.0), ), ((-a/4, c, 0.0), ), ), name='Upper')
377 mdb.models[modelName].parts['Plate'].Set(edges=
378     mdb.models[modelName].parts['Plate'].edges.findAt(((a/4, -c,
379     0.0), ), ((-a/4, -c, 0.0), ), ), name='Lower')
380 mdb.models[modelName].parts['Plate'].Set(edges=
381     mdb.models[modelName].parts['Plate'].edges.findAt(((-0.45*a, 0.0,
382     0.0), ), ((-1.1*Xe, 0.0, 0.0), ), ((-0.5*a+0.55*d, 0.0, 0.0), ),
383     ((-0.5*a+0.51*d, 0.0, 0.0), ), ), name='Reflection_Line')
384 mdb.models[modelName].parts['Plate'].Set(edges=
385     mdb.models[modelName].parts['Plate'].edges.findAt(((0.45*a, 0.0,
386     0.0), ), ((1.1*Xe, 0.0, 0.0), ), ), name='Transmission_Line')
387 mdb.models[modelName].parts['Plate'].Set(edges=
388     mdb.models[modelName].parts['Plate'].edges.findAt(((0.5*a,
389     1.5*c, 0.0), ), ((-0.5*a, 1.5*c, 0.0), ), ((-0.5*a, -1.5*c, 0.0),...
            ),
390     ((0.5*a, -1.5*c, 0.0), ), ), name='Infinite_1el')
391 mdb.models[modelName].parts['Plate'].Set(name='Mid_Reflection',
392     vertices=mdb.models[modelName].parts['Plate'].vertices.findAt(((
393     -0.5*(0.5*a+Xe-f), 0.0, 0.0), )))
394 mdb.models[modelName].parts['Plate'].Set(name='Mid_Transmitted',
395     vertices=mdb.models[modelName].parts['Plate'].vertices.findAt(((
396     (0.25*a+0.5*Xe), 0.0, 0.0), )))
397 mdb.models[modelName].parts['Plate'].Set(name='Mid_Tumor',
398     vertices=mdb.models[modelName].parts['Plate'].vertices.findAt(((
399     0.0, 0.0, 0.0), )))
400 mdb.models[modelName].parts['Plate'].Set(name='FocalPoint',
401     vertices=mdb.models[modelName].parts['Plate'].vertices.findAt(((
402     -0.5*a+0.5*f, 0.0, 0.0), )))
403 mdb.models[modelName].rootAssembly.Set(faces=
404     mdb.models[modelName].rootAssembly.instances['Plate-1'].faces....
          findAt(
405     ((-0.5*a+f, -0.5*c, 0.0), ), ((-0.5*a+f, 0.5*c, 0.0), ), ), name=
406     'BodyForce')
407 mdb.models[modelName].parts['Plate'].Set(name='...
      Reflection_Tumor_Boundary',
408     vertices=mdb.models[modelName].parts['Plate'].vertices.findAt(((
409     -Xe , 0.0, 0.0), )))
410 mdb.models[modelName].parts['Plate'].Set(name='...
      Transmission_Tumor_Boundary',
411     vertices=mdb.models[modelName].parts['Plate'].vertices.findAt(((
412     Xe , 0.0, 0.0), )))
413
414 # Separate the elements that should be assigned different stack ...
      orientation
415
416 mdb.models[modelName].parts['Plate'].Set(faces=
```

```
417     mdb.models[modelName].parts['Plate'].faces.getByBoundingBox(
418    -0.55*a,0.9*c,-0.1,0.55*a,0.55*b,1.1), name='Infinite_Upper')
419
420 mdb.models[modelName].parts['Plate'].Set(faces=
421     mdb.models[modelName].parts['Plate'].faces.getByBoundingBox(
422    -0.55*a,-0.55*b,-0.1,0.55*a,-0.9*c,1.1), name='Infinite_Lower')
423
424 # ---------- Datum Plane for the ultrasound field --------
425
426 myDatum = mdb.models[modelName].rootAssembly.DatumCsysByThreePoints(
427     coordSysType=CARTESIAN, name='CSYS_BodyForceField', origin=
428     mdb.models[modelName].rootAssembly.instances['Plate-1'].vertices....
            findAt(
429     (-0.5*a, c, 0.0), ), point1=
430     mdb.models[modelName].rootAssembly.instances['Plate-1']....
            InterestingPoint(
431     mdb.models[modelName].rootAssembly.instances['Plate-1'].edges....
            findAt(
432     (-0.1*a, c, 0.0), ), MIDDLE), point2=
433     mdb.models[modelName].rootAssembly.instances['Plate-1']....
            InterestingPoint(
434     mdb.models[modelName].rootAssembly.instances['Plate-1'].edges....
            findAt(
435     (-0.5*a, 0.1*c, 0.0), ), MIDDLE))
436
437 # ---------- Creating and generating mesh ----------
438
439 # 1: Meshing technique and element choice
440
441 mdb.models[modelName].parts['Plate'].setMeshControls(elemShape=QUAD,...
      regions=
442     mdb.models[modelName].parts['Plate'].faces.getByBoundingBox(
443    -0.5*a+0.45*d,-1.1*c,-0.1,0.55*a,1.1*c,1.1),
444     technique=meshTech)
445
446 mdb.models[modelName].parts['Plate'].setMeshControls(elemShape=QUAD,...
      regions=
447     mdb.models[modelName].parts['Plate'].faces.getByBoundingBox(
448    -0.55*a,-1.1*c,-0.1,-0.5*a+0.6*d,1.1*c,1.1),
449     technique=meshTech_ARF)
450
451 if explicit == 1:
452  mdb.models[modelName].parts['Plate'].setElementType(elemTypes=(
453     ElemType(elemCode=elType, elemLibrary=EXPLICIT, ...
            secondOrderAccuracy=OFF,
454     hourglassControl=DEFAULT, distortionControl=DEFAULT), ElemType(
455     elemCode=CPE3, elemLibrary=EXPLICIT)), regions=(
456     mdb.models[modelName].parts['Plate'].faces.getByBoundingBox(
457    -0.55*a,-1.1*c,-0.1,0.55*a,1.1*c,1.1),
458     ))
459
460 if implicit == 1:
461  mdb.models[modelName].parts['Plate'].setElementType(elemTypes=(
462     ElemType(elemCode=elTypeImplicit, elemLibrary=STANDARD,
463     secondOrderAccuracy=OFF,
464     hourglassControl=DEFAULT, distortionControl=DEFAULT), ElemType(
465     elemCode=CPE6M, elemLibrary=STANDARD)), regions=(
```

```
466     mdb.models[modelName].parts['Plate'].faces.getByBoundingBox(
467    -0.55*a,-1.1*c,-0.1,0.55*a,1.1*c,1.1),
468     ))
469  mdb.models[modelName].parts['Plate'].setElementType(elemTypes=(
470     ElemType(elemCode=CPS8, elemLibrary=STANDARD,
471     secondOrderAccuracy=OFF,
472     hourglassControl=DEFAULT, distortionControl=DEFAULT), ElemType(
473     elemCode=CPS6M, elemLibrary=STANDARD)), regions=(
474     mdb.models[modelName].parts['Plate'].faces.getByBoundingBox(
475    -0.55*a,0.95*c,-0.1,0.55*a,0.55*b,1.1),
476     ))
477  mdb.models[modelName].parts['Plate'].setElementType(elemTypes=(
478     ElemType(elemCode=CPS8, elemLibrary=STANDARD,
479     secondOrderAccuracy=OFF,
480     hourglassControl=DEFAULT, distortionControl=DEFAULT), ElemType(
481     elemCode=CPS6M, elemLibrary=STANDARD)), regions=(
482     mdb.models[modelName].parts['Plate'].faces.getByBoundingBox(
483    -0.55*a,-0.55*b,-0.1,0.55*a,-0.95*c,1.1),
484     ))
485
486
487 # 2: Mesh seed
488
489 mdb.models[modelName].parts['Plate'].seedPart(deviationFactor=0.1,
490     minSizeFactor=minSize, size=elSize)
491
492 mdb.models[modelName].parts['Plate'].seedEdgeByNumber(constraint=
493     FINER, edges=mdb.models[modelName].parts['Plate'].edges.findAt(((
494     0.5*a, 1.5*c, 0.0),), ((-0.5*a, 1.5*c, 0.0),), ((-0.5*a, -1.5*c...
          ,
495     0.0),), ((0.5*a, -1.5*c, 0.0),),), number=1)
496
497 mdb.models[modelName].parts['Plate'].seedEdgeBySize(constraint=FINER,
498     deviationFactor=0.1, edges=
499     mdb.models[modelName].parts['Plate'].edges.getByBoundingBox(-1.1*...
          Xe,
500     -1.1*Ye, -0.1, 1.1*Xe, 1.1*Ye, 1.1), minSizeFactor=minSize,
501     size=elSize_inclusion)
502
503 # 3: Generate mesh
504 mdb.models[modelName].parts['Plate'].generateMesh()
505
506
507 # ---------------- Create loading step ----------------
508
509 # Comment out the step to exclude from the analysis
510
511 # Step 1: Explicit Dynamic
512 if explicit == 1:
513  mdb.models[modelName].ExplicitDynamicsStep(name='Dynamic_Explicit',
514     previous='Initial', timePeriod=Ta, linearBulkViscosity=linBulk,
515     quadBulkViscosity=quadBulk)
516
517
518 # Step 2: Implicit Dynamic
519 if implicit == 1:
520  mdb.models[modelName].ImplicitDynamicsStep(timePeriod=Ta,
```

```
521  initialInc= dt, minInc = dt_min, maxInc=dt ,maxNumInc=10000000,
522  name='Dynamic_Implicit',
523  nlgeom=ON, noStop=OFF, nohaf=OFF, previous='Initial',
524  timeIncrementationMethod=AUTOMATIC)
525
526 # ------------  Field and history output ---------------
527
528 del mdb.models[modelName].fieldOutputRequests['F-Output-1']
529 del mdb.models[modelName].historyOutputRequests['H-Output-1']
530
531 if explicit == 1:
532  mdb.models[modelName].FieldOutputRequest(createStepName=
533      'Dynamic_Explicit', name='WholeModel', numIntervals=numField,
534      variables=('S','UT','LE'))
535  mdb.models[modelName].HistoryOutputRequest(createStepName=
536      'Dynamic_Explicit', name='Mid_Reflected', numIntervals=numHist, ...
            rebar=EXCLUDE,
537       region=
538       mdb.models[modelName].rootAssembly.instances['Plate-1'].sets['...
            Mid_Reflection']
539      , sectionPoints=DEFAULT, variables=('U1','U2'))
540  mdb.models[modelName].HistoryOutputRequest(createStepName=
541      'Dynamic_Explicit', name='Mid_Transmitted', numIntervals=numHist,...
            rebar=EXCLUDE,
542      region=
543      mdb.models[modelName].rootAssembly.instances['Plate-1'].sets['...
            Mid_Transmitted']
544      , sectionPoints=DEFAULT, variables=('U1','U2'))
545  mdb.models[modelName].HistoryOutputRequest(createStepName=
546      'Dynamic_Explicit', name='Mid_Tumor', numIntervals=numHist, rebar...
            =EXCLUDE,
547      region=
548      mdb.models[modelName].rootAssembly.instances['Plate-1'].sets['...
            Mid_Tumor']
549      , sectionPoints=DEFAULT, variables=('U1','U2'))
550  mdb.models[modelName].HistoryOutputRequest(createStepName=
551      'Dynamic_Explicit', name='FocalPoint', numIntervals=numHist, ...
            rebar=EXCLUDE,
552      region=
553      mdb.models[modelName].rootAssembly.instances['Plate-1'].sets['...
            FocalPoint']
554      , sectionPoints=DEFAULT, variables=('U1','U2'))
555  mdb.models[modelName].HistoryOutputRequest(createStepName=
556      'Dynamic_Explicit', name='Reflection_Tumor', numIntervals=numHist...
            , rebar=EXCLUDE,
557      region=
558      mdb.models[modelName].rootAssembly.instances['Plate-1'].sets['...
            Reflection_Tumor_Boundary']
559      , sectionPoints=DEFAULT, variables=('U1','U2'))
560  mdb.models[modelName].HistoryOutputRequest(createStepName=
561      'Dynamic_Explicit', name='Tumor_Transmission', numIntervals=...
            numHist, rebar=EXCLUDE,
562      region=
563      mdb.models[modelName].rootAssembly.instances['Plate-1'].sets['...
            Transmission_Tumor_Boundary']
564      , sectionPoints=DEFAULT, variables=('U1','U2'))
565
```

```
566 if implicit == 1:
567   mdb.models[modelName].FieldOutputRequest(createStepName=
568       'Dynamic_Explicit', name='WholeModel', frequency=1,
569     variables=('S','UT','E'))
570   mdb.models[modelName].HistoryOutputRequest(createStepName=
571    'Dynamic_Implicit', name='Mid_Tumor', frequency=1, rebar=EXCLUDE,
572    region=
573    mdb.models[modelName].rootAssembly.instances['Plate-1'].sets['...
         Mid_Tumor']
574    , sectionPoints=DEFAULT, variables=('U1','U2'))
575   mdb.models[modelName].HistoryOutputRequest(createStepName=
576    'Dynamic_Implicit', name='Mid_Reflection', frequency=1, rebar=...
         EXCLUDE,
577    region=
578    mdb.models[modelName].rootAssembly.instances['Plate-1'].sets['...
         Mid_Reflection']
579    , sectionPoints=DEFAULT, variables=('U1','U2'))
580   mdb.models[modelName].HistoryOutputRequest(createStepName=
581    'Dynamic_Implicit', name='Mid_Transmitted', frequency=1, rebar=...
         EXCLUDE,
582    region=
583    mdb.models[modelName].rootAssembly.instances['Plate-1'].sets['...
         Mid_Transmitted']
584    , sectionPoints=DEFAULT, variables=('U1','U2'))
585   mdb.models[modelName].HistoryOutputRequest(createStepName=
586    'Dynamic_Implicit', name='FocalPoint', frequency=1, rebar=EXCLUDE,
587    region=
588    mdb.models[modelName].rootAssembly.instances['Plate-1'].sets['...
         FocalPoint']
589    , sectionPoints=DEFAULT, variables=('U1','U2'))
590
591 # ------------------ Amplitude function ------------------
592
593 mdb.models[modelName].SmoothStepAmplitude(data=((0.0, 0.0), (0.1*Ti,
594     1.0), (0.9*Ti, 1.0), (Ti, 0.0)), name='SmoothStep', timeSpan=STEP...
          )
595
596 # ------------ Create ultrasound loading field ----------
597
598 mdb.models[modelName].ExpressionField(description='',
599     expression=
600     repr(a1) + '*exp ( -pow ( ((Y-' + repr(b1) + ')/' + repr(c1) + ')...
          ,2 ) ) + '
601     + repr(a2) + '*exp ( -pow ( ((Y-' + repr(b2) + ')/' + repr(c2) + ...
          '),2 ) ) + '
602     + repr(a3) + '*exp ( -pow ( ((Y-' + repr(b3) + ')/' + repr(c3) + ...
          '),2 ) )',
603     localCsys=mdb.models[modelName].rootAssembly.datums[myDatum.id]
604     , name='BodyForce')
605
606 # ------------ Loading and boundary conditions ----------
607
608 # Clamp at the right edge and symmetry at left edge
609
610 mdb.models[modelName].XsymmBC(createStepName='Initial',
611     localCsys=None, name='Symmetry', region=Region(
```

```
612     edges=mdb.models[modelName].rootAssembly.instances['Plate-1'].... edges.findAt(
613     ((-0.5*a, -0.5*c, 0.0), ), ((-0.5*a, 0.5*c, 0.0), ), )))
614
615 mdb.models[modelName].rootAssembly.regenerate()
616 mdb.models[modelName].DisplacementBC(amplitude=UNSET, createStepName=
617     'Initial', distributionType=UNIFORM, fieldName='', localCsys=None... , name=
618     'Clamped', region=
619     mdb.models[modelName].rootAssembly.instances['Plate-1'].sets['... Fixed']
620     , u1=SET, u2=SET, ur3=SET)
621
622 # Roller support on upper and lower boundaries
623
624 if rollingLower == 1:
625  mdb.models[modelName].DisplacementBC(amplitude=UNSET, createStepName... =
626     'Initial', distributionType=UNIFORM, fieldName='', localCsys=None... , name=
627     'RollingSupport_Lower', region=
628     mdb.models[modelName].rootAssembly.instances['Plate-1'].sets['... Lower']
629     , u1=SET, u2=UNSET, ur3=UNSET)
630
631 if rollingUpper == 1:
632  mdb.models[modelName].DisplacementBC(amplitude=UNSET, createStepName... =
633     'Initial', distributionType=UNIFORM, fieldName='', localCsys=None... , name=
634     'RollingSupport_Upper', region=
635     mdb.models[modelName].rootAssembly.instances['Plate-1'].sets['... Upper']
636     , u1=SET, u2=UNSET, ur3=UNSET)
637
638 # Shear force on the left edge
639
640 if includeShear == 1:
641  mdb.models[modelName].SurfaceTraction(amplitude='SmoothStep',
642     createStepName='Dynamic_Explicit', directionVector=(
643     mdb.models[modelName].rootAssembly.instances['Plate-1'].vertices.... findAt(
644     (-0.5*a, 0.0, 0.0), ),
645     mdb.models[modelName].rootAssembly.instances['Plate-1'].... InterestingPoint(
646     mdb.models[modelName].rootAssembly.instances['Plate-1'].edges.... findAt(
647     (-0.5*a, 0.25*b, 0.0), ), MIDDLE)), distributionType=UNIFORM, ... field='',
648     localCsys=None, magnitude=shearForce, name='ShearTraction', ... region=Region(
649     side1Edges=mdb.models[modelName].rootAssembly.instances['Plate-1'... ].edges.findAt(
650     ((-0.5*a, 0.5*c, 0.0), ), ((-0.5*a, -0.5*c, 0.0), ), )))
651
652 # Radiation force in the ultrasound pressure field
653
```

```
654 if includeARF == 1:
655  if explicit == 1:
656   mdb.models[modelName].BodyForce(amplitude='SmoothStep',
657     comp2=-1.0, createStepName='Dynamic_Explicit', distributionType=...
          FIELD,
658     field='BodyForce', name='BodyForce', region=Region(
659     faces=mdb.models[modelName].rootAssembly.instances['Plate-1'].....
          faces.findAt(
660     ((-0.5*a+0.5*f, -0.1*c, 0.0), (0.0, 0.0, 1.0)), ((-0.5*a+0.5*f, ...
          0.1*c, 0.0),
661     (0.0, 0.0, 1.0)), )))
662  if implicit == 1:
663   mdb.models[modelName].BodyForce(amplitude='SmoothStep',
664     comp2=-1.0, createStepName='Dynamic_Implicit', distributionType=...
          FIELD,
665     field='BodyForce', name='BodyForce', region=Region(
666     faces=mdb.models[modelName].rootAssembly.instances['Plate-1'].....
          faces.findAt(
667     ((-0.5*a+0.5*f, -0.1*c, 0.0), (0.0, 0.0, 1.0)), ((-0.5*a+0.5*f, ...
          0.1*c, 0.0),
668     (0.0, 0.0, 1.0)), )))
669 # ------------------ Create job ------------------------
670
671 jobName = modelName + '_n=' + repr(int(
672 round(n,0))) + '_Job'
673
674 if viscoElastic == 1:
675  jobName = 'V_' + modelName + '_T=' + repr(int(
676  round(1E6*Ti,0))) +'_Job'
677 #+ '_a=' + repr(a) + '_b=' + repr(b) + '_Xe=' + repr(
678 #Xe) + '_Ye=' + repr(Ye) + '_v=' + repr(nu1)
679
680 jobName = jobName.replace(".",",")
681
682 inputName = 'Infinite' + '_n=' + repr(int(round(n,0)))
683
684 if viscoElastic == 1:
685  inputName = 'Infinite_V' + '_T=' + repr(int(round(1E6*Ti,0)))
686
687 inputName = inputName.replace(".",",")
688
689 inputJobName = inputName + '_Job'
690
691 inputJobName = inputJobName.replace(".",",")
692
693 if makeJob_1 == 1:
694  mdb.Job(activateLoadBalancing=False, atTime=None, contactPrint=OFF,
695     description='', echoPrint=OFF, explicitPrecision=SINGLE, ...
          historyPrint=OFF,
696     model=modelName, modelPrint=OFF, multiprocessingMode=DEFAULT,
697     name=jobName, nodalOutputPrecision=SINGLE, numCpus=1,
698     numDomains=1, parallelizationMethodExplicit=DOMAIN, queue=None, ...
          scratch='',
699     type=ANALYSIS, userSubroutine='', waitHours=0, waitMinutes=0)
700
701 # Writing out an input file and importing it back to Abaqus CAE
702
```

```
703 if makeJob_1 == 1:
704  mdb.jobs[jobName].writeInput()
705  mdb.ModelFromInputFile(inputFileName=
706      'D:/NTNU/Master/Simulations/2D_PlaneStrain/' + jobName + '.inp',
707      name=inputName)
708
709 # Creating the second job that governs the infinite elements input ...
        file
710
711 if makeJob_2 == 1:
712  mdb.Job(activateLoadBalancing=False, atTime=None, contactPrint=OFF,
713      description='', echoPrint=OFF, explicitPrecision=SINGLE, ...
            historyPrint=OFF,
714      model=inputName, modelPrint=OFF, multiprocessingMode=
715      DEFAULT, name=inputJobName, nodalOutputPrecision=
716      SINGLE, numCpus=1, numDomains=1, parallelizationMethodExplicit=...
            DOMAIN,
717      queue=None, scratch='', type=ANALYSIS, userSubroutine='', ...
            waitHours=0,
718      waitMinutes=0)
719
720 if n == 6:
721  mdb.models[inputName].parts['PART-1'].orientElements(
722      pickedElements=
723      mdb.models[inputName].parts['PART-1'].elements[5631:5713],
724      referenceRegion=
725      mdb.models[inputName].parts['PART-1'].elemEdges[182145])
726
727 if n == 8:
728  mdb.models[inputName].parts['PART-1'].orientElements(
729       pickedElements=
730       mdb.models[inputName].parts['PART-1'].elements[9705:9814],
731       referenceRegion=
732      mdb.models[inputName].parts['PART-1'].elemEdges[312961])
733
734 if n == 10:
735  mdb.models[inputName].parts['PART-1'].orientElements(
736       pickedElements=
737       mdb.models[inputName].parts['PART-1'].elements[16514:16655]
738       , referenceRegion=
739       mdb.models[inputName].parts['PART-1'].elemEdges[531777])
740
741 if n == 12:
742  mdb.models[inputName].parts['PART-1'].orientElements(
743       pickedElements=
744       mdb.models[inputName].parts['PART-1'].elements[21905:22068]
745       , referenceRegion=
746      mdb.models[inputName].parts['PART-1'].elemEdges[704641])
747
748 if n == 14:
749  mdb.models[inputName].parts['PART-1'].orientElements(
750       pickedElements=
751       mdb.models[inputName].parts['PART-1'].elements[30049:30238],
752       referenceRegion=
753      mdb.models[inputName].parts['PART-1'].elemEdges[966273])
754
755 if makeJob_2 == 1:
```

```
756  mdb.jobs[inputJobName].writeInput()
757
758 if saveModel == 1:
759  mdb.saveAs(pathName=workDirectory + '/' + caeName)
```