Delft University of Technology

Bachelor thesis

Bachelor Applied Physics & Applied Mathematics

Solving The Westervelt Equation With Losses Using First And Second Order Finite Element Method

Author:
Marcella Zijta

Supervisors:
Dr. Domenico J. P. Lahaye
Dr. ir. Martin D. Verweij

Delft University of Technology,

Faculty of Applied Sciences &
Faculty of Electrical Engineering, Mathematics and Computer Science,

Section of Acoustical Wavefield Imaging &
Department of Numerical Analysis

July 20, 2017

Summary

In this Bachelor thesis we researched the second order finite element method as a method to solve the one-dimensional Westervelt wave equation. The Westervelt equation is a wave equation that describes the propagation of a nonlinear plane wave. The goal was to use a first order finite element method and a second order finite element method to solve the Westervelt equation, research the difference between these methods and known solutions and simulate the effect of certain changes in parameters. First we developed an understanding of the nonlinear wave propagation by analyzing the Burgers equation, which is an approximation to the Westervelt equation without attenuation term in another coordinate system. This analysis explains the influence of nonlinearity on the frequency spectrum of the solution. By looking at the derivation of the Westervelt equation changes in wave form during propagation are explained.
We used our own finite element method to make implementations in Matlab to solve the linear wave equation, the Westervelt equation without attenuation term and the complete Westervelt equation. To deduce the implementations, we assumed that our unknown solution was a linear combination of first order basis functions. We then wrote the term of the equations into these linear combination and paraphrased them as matrix vector multiplications. Finally, a backwards differential method was used to solves these matrix vector multiplications iteratively in time.
We compared the solutions of the linear wave equation using FEM to the known analytic solution and compared the solutions of the Westervelt equation without attenuation term to the solution of the Burgers equation. The finite element method cannot not be completely accurate because it depends on a mesh size that is not infinitely small. This means we always use an approximation in space. There is also an error in time, because we use backwards difference scheme to solve the equation iteratively in time. To approve accuracy we should put research into an adaptive mesh, where we choose a mesh with a lot more points around the peaks.
The difference between the first and second basis functions of the finite element method was for the linear wave equation only caused by the small phase shift of the solutions. For the Westervelt equation with and without attenuation term the difference was larger, because of the effect of the nonlinear term on the Westervelt equation. The nonlinearity caused the slope of the wave to steepen when going from a maximum to a minimum. This effect eventually caused the formation of shock waves. The attenuation term inhibited the effect of the nonlinearity on the form of the wave during propagation and thus made the slope of the wave less steep. As a result, no shock waves were formed. The formation of shock waves was immediately clear, when we looked at the solutions at two times the shock wave distance. We concluded from this that the finite element method is not usable after shock waves are formed. We have also shown the frequency spectrum of the solutions and saw the transfer of energy to higher harmonics for the solutions that depended on nonlinearity. The attenuation term damped the transfer of energy and indeed in the spectra we saw less peaks in the higher harmonics. Lastly, we discussed a sudden change in velocity of the wave and calculated the amplitude of the reflected and transmitted wave. We simulated these waves with our implementation in Matlab and concluded that the theory about reflection and transmission agrees with our simulations.

Contents

1 Introduction

1.1 Medical Acoustic Imaging

Through the late 19th and 20th centuries there was a deep desire to look inside the body. This caused many scientists to develop diagnostic and treatment devices that used all kinds of waves. Röntgen's discovery of the X-rays is just one example of this. Interestingly not only the desire to see the insides of a live body caused this vast spur in development. Also the sinking of the Titanic in 1912 triggered people to discover applications to search the bottom of the ocean. This search for discoveries was stimulated by the French government, because they wanted a device that was able to detect enemy submarines during the first World War. The basis of medical ultrasound was formed in that device but it was not till 1940 that the first ultrasonic echo imaging device was made. Karl Theo Dussik was the first person to make an image of the human body using ultrasonic echo.
Now medical acoustic imaging is a standard in the medical diagnostic practice. The most known use is obstretric ultrasonography during pregnancy. This creates the real-time image of the fetus in its mother's womb. This image is used to monitor the health of the fetus and detect potential problems with the fetus. The gender of the fetus can also be discovered by imaging. Beside obstretric ultrasonography, acoustic imaging is also used in many other applications. In cardiography, for example, it has given the cardiologists a way to diagnose several heart diseases and abnormalities.

Figure 1.1: *An image of a fetus in the womb of a mother made with obstretic ultrasonography.*

Ultrasonic images are made by sending ultrasound pulses into the body using a probe. The sound echoes off the tissue. These echoes contain spatial and contrast information. The probe records the different echoes and sends them to a computer that processes the echoes and displays the image. With obstretric ultrasonography the fetus will reflect an echo that has a different intensity and shape than the echo reflected by womb tissue. So the computer will process them differently and they will have a different color in the image. The fetus is solid, so the wave will bounce off and the computer will show this as a light-colored image. The tissue of the uterus is a lot less compact, so most of the sound wave will pass through and the echoes will be few and faint. In the image this region will stay black.

There are a lot of advantages using ultrasound imaging compared to other imaging procedures. Firstly, ultrasound uses non-ionizing sound waves. These waves are not associated with the formation of cancer, while other imaging procedures can only be used a maximum amount of times on a person's body because of radiation exposure. That is one of the reasons why ultrasound imaging is used to make an image of a fetus in an uterus, so the fetus does not get exposed to radiation. Another advantage of ultrasound imaging is that it is far less expensive than a CT-scan or a MRI. Additionally, with ultrasound imaging a real-time image can be made. So when making an image of a heart, the heartbeat and its influence on the heart tissue are visible. So, for example, a cardiologist can see

abnormalities during the closing of the heart valve when the heart is beating.
Since 1940 the ultrasound imaging has been revolutionized. One sizable improvement is the adding of a spatial element. Multiple 2D-images at different angles can be converted to a 3D-image which gives doctors an advantage when looking for abnormalities. The first 3D-image of a fetus was made by processing raw 2D-images in 1986 [1]. Another improvement is the adding of contrast agents. The contrast agents are bubbles filled with gasses, composed of soft- or hard-shell material. On the shell of the bubbles specific ligands are coupled, which bind to specific molecules in tissue. Thus the bubbles improve the sensitivity for tumor detection by enhancing the contrast between different structures.[2] Finally, the utilization of the nonlinear distortion of the ultrasound field has resulted in significant improvement of the image quality. The utilization of nonlinear distortion is elaborated in [3] and [4]. In the next section this nonlinear distortion will be explained in more detail.

1.2 Westervelt equation

When working in the field of acoustics the propagation of the sound waves have to be described by a nonlinear wave equation. The nonlinearity is caused by field dependent medium behavior that shows up in the derivatives as well as in the medium parameters of the wave equation. This nonlinearity only has significant effect at large acoustic pressures or frequencies, so most of the time it is sufficient to use the linear wave equation to model wave propagation. This works especially when attenuation is taking into account as well, because it damps the effect of the nonlinear term. The solution to the linear wave equation is well known, so it is easy to get information about the propagation. However the linear wave equation is not usable for all applications of acoustics, like medical acoustic imaging. In medical acoustic imaging ultrasound waves are used. These waves have high frequencies and high amplitudes so the nonlinearity becomes significant.
In this thesis we will state an equation which describes the nonlinear wave equation in general terms. This equation is known as the Westervelt equation and is given by

$$\nabla^2 p - \frac{1}{c_0^2}\frac{\partial^2 p}{\partial t^2} + \frac{\delta}{c_0^4}\frac{\partial^3 p}{\partial t^3} = -\frac{\beta}{\rho_0 c_0^4}\frac{\partial^2 p^2}{\partial t^2}, \tag{1.1}$$

where p is the unknown pressure field, β is the coefficient of nonlinearity, δ is the sound diffusivity, ρ_0 is the ambient density of mass and c_0 is the small signal sound speed. The subscript 0 emphasizes (for all materials) that we mean the *small signal* sound propagation speed in case of c_0 and the *ambient* density of mass in case of ρ_0, which must not be confused with the local sound propagation speed and the local density of mass. δ is a coefficient with SI-units m^2/s that accounts for the attenuation of the wave. The acoustic energy losses are mostly caused by absorption and scattering. The coefficient depends on the shear viscosity, the buld viscosity, thermal conductivity and specific heat. More discussion on this coefficient can be found in [4]. β is a dimensionless material property that accounts for the nonlinear effects. It is theoretically derived from the equation of states and its values are usually obtained empirically. More discussion on this coefficient can be found in [3] or [5].
If we set $\beta = 0$ and $\delta = 0$ in the Westervelt equation, the linear wave equation is left. The term with β is clearly nonlinear, because the second derivative of p^2 is taken. So it is understandable that when the frequency and wave amplitude are considerable the partial derivative becomes very large, which causes nonlinear propagation.
In this thesis only the one-dimensional version of the Westervelt equation is used. So ∇^2 is replaced with $\frac{\partial^2}{\partial x^2}$ with x the spatial coordinate.

1.3 Finite Element Method

The Westervelt equation (1.1) does not have a known analytic solution. This is because of the nonlinearity, superposition of wave fields does not apply. With linear waves the wavefield of a combination of several sources is equal to a superposition of the wavefields of several sources. Because of the nonlinearity the pulse shape of the wave deforms during propagation thus the principle of superposition cannot be used. So this equation has to been solved numerically. There are a lot of different common methods to solve the Westervelt equation. Because of the large amount of computation time that is

needed these models have been limited to 1D or 2D cartesian of cylindrical implementations. The first method to solve the Westervelt equation is the use of finite difference method. This is the most straightforward way. The derivatives in the equation are approximated by finite difference schemes. The new equations that are derived from this are solved iteratively in time. Sparrow and Raspet [6] studied nonlinear effects in air both with a finite difference solution of a set of first-order conservation laws. Solovchuk and Sheu used an implicit finite difference time domain method to solve the nonlinear Westervelt equation to investigate the influence of blood flow on temperature distribution during high-intensity ultrasound[7]. Another method to solve the Westervelt equation is with the use of Greens function. It is used to solve the Westervelt equation iteratively and the nonlinear term is viewed as a source function to the linear wave equation. Jing, Tao and Clement proposed an implicit solution for the nonlinear term by employing the Greens function and used this to solve the nonlinear wave equation [8]. Another method is the finite element method (FEM). For this method the Westervelt equation first has to be written into its weak formulation. This weak formulation is then solved for given boundary conditions and a specific test function. This new equation is then solved using specific basis functions. For the time domain the approximation of the derivatives with difference schemes is used.
In this thesis the applicability of FEM to the Westervelt equation is researched. We want to research FEM as a method to solve the Westervelt equation, because it is a method that is not commonly used to solve the Westervelt equation. So there is still research to be done about solving the Westervelt equation using FEM. Another reason why we have chosen the FEM method is the fact that this thesis is actually a continuation of Bas Dirkse's Thesis [9]. In his thesis the applicability of first order FEM to the Westervelt equation is researched. He discovered that his own implementation of first order FEM was more accurate than more general build-in time solvers. Also was concluded that FEM had an advantage over other methods when solving equations with inhomogeneous domains. He concluded that it is more advantageous to use higher-order basis functions, since they reduce the relative error by a larger factor than the low-order basis functions for certain refinement. The higher-order FEM use more auxiliary points in between every two element points than lower-order FEM, which should give a better approximation. To test this theory, a first order FEM and a second order FEM are used to solve the Westervelt equation in this thesis. In the thesis of Bas the Westervelt equation is taken without attenuation term. In this thesis the effect of the attenuation term on the solution and the efficiency of the solving scheme is researched as well. The values of the parameters of the surroundings are also researched on the effect that they have on the solution.
To compare the accuracy of the solution by using FEM, the solution to the Burgers equation is used. The Burgers equation is an equation derived from the one-dimensional lossless Westervelt equation that can be solved implicitly, because it neglected some terms. The derivation of the Burgers equation and the information on the propagation it contains are described in Section 2.1.

1.4 Structure of this thesis

In Chapter 2 we study the derivation of the Burgers equation along with its implicit solution. In Matlab we use interpolation to turn this implicit solution into an explicit solution to compare to solutions made with the finite element method. In Chapter 2 we also study wave propagation given by the Westervelt equation. This propagation tells us where and why shock wave formation occurs. We also discuss the effect of the attenuation term on the shock wave formation. In this Chapter the frequency spectrum of the solution to the Burgers equation is debated, as well. It explains that through nonlinearity the energy will transfer to higher harmonics. Finally, we discuss the effect of a change in parameters during the propagation of the wave. If we change the small signal sound speed of the wave during propagation, reflection and transmission of the wave will occur. Formulas to compute the amplitude of the reflected and transmitted wave are given.
In Chapter 3 we discuss solving the Westervelt equation using the finite element method. Firstly, we write the Westervelt equation into its weak form using a test function v. The unknown pressure wave function and the test function are both linear combinations of basis functions. In this Chapter we discuss first order and second order basis function in space. First we use first order basis functions in space. With these basis functions we write out the terms of the equation in basis functions and we

determine which terms of the summation are zero. The rest of the terms are now smartly written into a matrix vector multiplication. After we have solved the equation in space, we explain how to solve the equation iteratively in time using difference schemes. We do the same for the Westervelt equation without attenuation term and with attenuation term. We repeat this notion using second order basis functions. We implement the matrices and difference schemes that we got in Matlab.
In Chapter 4 we discuss the results obtained by the Matlab implementations from Chapter 3. We put a source function at $x = 0$ into the wave equation. For the linear wave we know the analytic solution, so we implement this into the figure with the solutions of the linear wave equation of the first and second order FEM. Into the figure with the solutions of the Westervelt equation without attenuation term of the first and second order FEM, we put the explicit solution of the Burgers equation to compare. All the figures are plotted at the shock wave formation distance. For the figures of the linear wave equation and the Westervelt equation with/without term of attenuation we also show the difference between the solution of the first order FEM and the second order FEM. We discuss if the figures agree with the theory. We plot the solutions to the different wave equations at two times the shock wave formation distance and discuss the changes. Thereafter, the frequency spectra of the solutions are shown to make the nonlinearity of the solutions of the Westervelt equation clearly visible and the effect of damping on the nonlinearity. Finally, solutions are plotted for a sudden change in small signal sound speed in the middle of the travel length. This sudden change of parameter causes reflection and transmission waves to arise. For different changes in small signal sound speed the reflection and transmission coefficients are calculated. These coefficients are linked back to the figure and agreements are discussed.

2 Theory

2.1 Burgers equation

2.1.1 Derivation of the Burgers equation

To derive the Burgers equation the comoving or retarded time, $\tau = t - \frac{x}{c_0}$ has to be introduced into the lossless Westervelt equation. We change the coordinate system from (x,t) to (x,τ). So p, the unknown pressure field, now is a function of $p(x,\tau)$. The first order derivatives of the new coordinate system are

$$\begin{array}{rclcrcl} \frac{\partial}{\partial t} & = & \frac{\partial}{\partial \tau}\frac{\partial \tau}{\partial t} + \frac{\partial}{\partial x}\frac{\partial x}{\partial t} & = & 1\cdot\frac{\partial}{\partial \tau} + 0\cdot\frac{\partial}{\partial x} & = & \frac{\partial}{\partial \tau}, \\ \frac{\partial}{\partial x} & = & \frac{\partial}{\partial \tau}\frac{\partial \tau}{\partial x} + \frac{\partial}{\partial x}\frac{\partial x}{\partial x} & = & \frac{-1}{c_0}\cdot\frac{\partial}{\partial \tau} + 1\cdot\frac{\partial}{\partial x} & = & \frac{\partial}{\partial \tau} - \frac{1}{c_0}\frac{\partial}{\partial \tau}. \end{array} \tag{2.1}$$

So if we use the new derivatives to rewrite all the term of the lossless Westervelt equation we get the following equation.

$$\frac{\partial^2 p}{\partial x^2} - \frac{2}{c_0}\frac{\partial^2 p}{\partial x \partial t} = \frac{\beta}{\rho_0 c_0^4} p \frac{\partial^2 p^2}{\partial \tau^2}. \tag{2.2}$$

Within the comoving time frame, the variation of the pressure with respect to x small. So using this we neglect the second order derivative with respect to x and then we integrate the function with respect to τ. Finally we multiply this final equation with the factor $-\frac{c_0}{2}$ and this gives us the lossless Burgers equation [4].

$$\frac{\partial p}{\partial x} = \frac{\beta}{\rho_0 c_0^3} p \frac{\partial p}{\partial \tau} \tag{2.3}$$

For this Burgers equation an implicit solution is known as derived in [4] and [10]. It is of the form

$$p(x,\tau) = f(\phi),\ \phi = \tau + \frac{\beta}{\rho_0 c_0^3}(x - x_0)p(x,\tau) \tag{2.4}$$

where $p(x_0,\tau)$ is the source pressure as a function of the comoving time at $x = x_0$. This solution can be used to compare to the solution obtained by FEM.

2.1.2 Numerical approximation to the implicit Burgers solution

If we define $x = x_0 + \Delta x$ and subsitute the Taylor expansion of $p(x_0 + \Delta x, \tau)$ around x_0 into ϕ from (2.4) and neglect the higher-order terms, we obtain [10]

$$p(x_0 + \Delta x, \tau) \simeq p(x_0, \tau + \frac{\beta}{\rho_0 c_0^3}\Delta x p(x,\tau)) \tag{2.5}$$

This approximation is valid only if $p(x,\tau)$ varies slowly over Δx. So with a source function at $x = 0$ we can get the explicit solution at every x of our domain length iteratively. To compare this to a FEM solution the coordinate system has to be changed back so we change back the time basis for every x

$$p(x,t) = p(x, \tau + \frac{x}{c_0}) \tag{2.6}$$

A implementation of this explicit scheme can be found in Appendix A

2.2 Nonlinear wave propagation using the Westervelt equation

In this thesis we did not discuss the derivation of the Westervelt equation, but we just gave the general form. However, the derivation gives us a lot of information about the propagation of the wave. In [3] and [4] the derivation is explained. It tells us for the lossless Westervelt equation that at a point of constant pressure the velocity at that point is

$$\frac{dx}{dt}\Big|_p = c_0 + \frac{\beta}{\rho_0 c_0} p \tag{2.7}$$

So every point of a wave moves with the constant small signal sound speed and an extra nonlinearity factor that depends on the pressure at that point. This means that when we are discussing a linear wave all the points of the wave travel with the same speed c_0. But when discussing a nonlinear wave this means that the point of the wave at maximum pressure travels a lot faster than the point at minimum pressure.

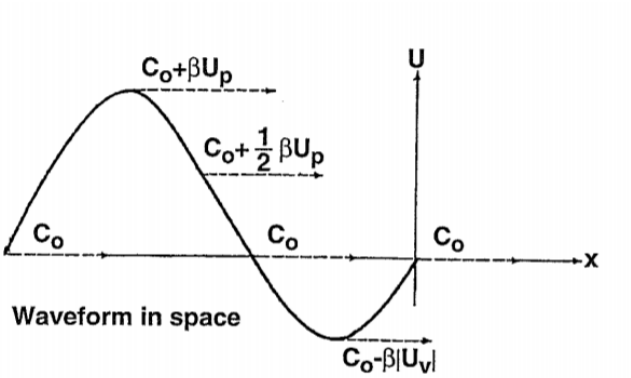

Figure 2.1: *Every point of a wave moves with the constant small signal sound speed and an extra nonlinearity factor that depends on the pressure at that point. [4]*

So when traveling in time there will be a point that the higher pressure point takes over the lower pressure point. This is where $\frac{\partial p}{\partial x} \to -\infty$. At this point a shock wave is formed and we call this point the shock formation distance. This shock formation distance is derived in [4] and is given by

$$x_{sh} = \frac{\rho_0 c_0^3}{\beta f'_{max}} \tag{2.8}$$

where f'_{max} is the maximum of the derivative of the source function f to the variable that it depends on. If we assume that the source is a single frequency sine wave ($f(t) = P_0 sin(2\pi f_0 t)$, with f_0 the frequency and P_0 the amplitude of the wave), then the derivative with respect to t is $f'(t) = 2\pi f_0 P_0 cos(2\pi f_0 t)$. Thus $f'_{max} = 2\pi P_0 f_0$ and using this in Equation 2.8 we get

$$x_{sh} = \frac{\rho_0 c_0^3}{\beta P_0 2\pi f_0} \tag{2.9}$$

The lossless Burgers solution is valid only on the area in front of the shock formation distance. This is because in the derivation it is assumed that the variation in pressure with respect to x is small and at the shock formation distance the variation goes to infinity.
Now we have not taken into account the effect of the attenuation term on the nonlinear Westervelt equation. The attenuation term damps the amplitude of the wave and damps the velocity of the wave, especially at the points of high pressure. So the attenuation term can retain the beginning of the shock wave. If the coefficient of attenuation is very small it will slow the steepening of the wave, but it will not stop the forming of a shock wave. If the coefficient of attenuation has a greater influence than the nonlinear term no shock wave will form. So the point of high pressure will never overtake the point of low pressure. Because of the effect of attenuation on the amplitude of the wave, this wave will just eventually damp out.

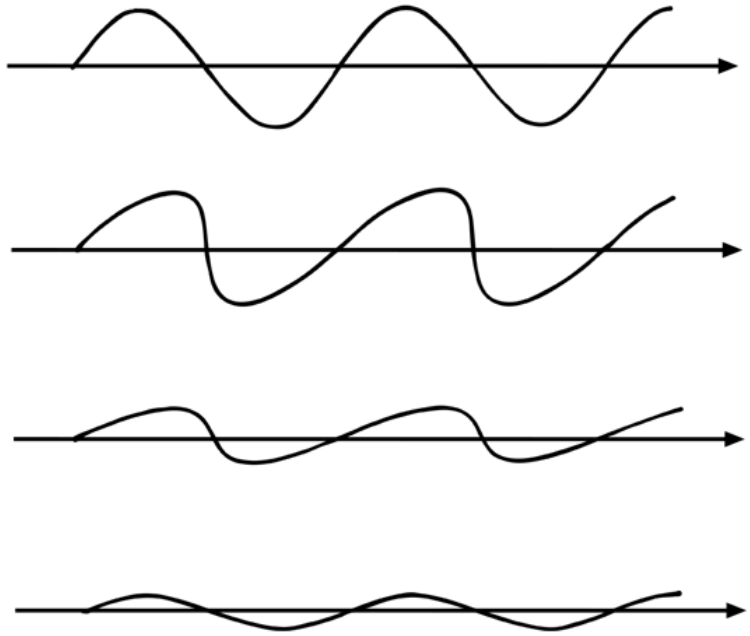

Figure 2.2: *First the wave is a normal sinus. The wave undergoes nonlinear propagation and it slope steepens. But because of attenuation also working on the wave, the wave will not become a shock wave but will remain propagating with a shrinking amplitude.*

2.3 Fourier Spectrum

In this section we will explain what happens to the Fourier spectrum of nonlinear wave propagation. For a single frequency source there is also an explicit solution known to the Burgers equation (2.3) found by Fubini in 1935 [4]. For this solution we assume the source has the form of

$$p(0,t) = P_0 \sin(\omega t),$$

Fubini found that given this source the solution to the Burgers equation is

$$p(\sigma,\tau) = P_0 \sum_{n=1}^{\infty} \frac{2}{n\sigma} J_n(n\sigma)\sin(n\omega\tau), \tag{2.10}$$

where $\sigma = x/x_{sh}$ is the dimensionless measure of length and J_n is the ordinary Bessel function of order n. Because of our derivation of the Burgers equation it is only valid for $0 < \sigma < 1$ [11]. This solution shows that energy is passed over to higher harmonics as the wave travels along x. So at small σ the energy is still based in one peak at the source frequency. When x travels to x_{sh} the energy of the first peaks is passed over to higher harmonics. So the first peak will lose part of its amplitude and more peaks will arise. It is important to remember that this happens for the solution of the Burgers equation and this is a consequence of the nonlinearity of the Burgers equation. For the linear wave equation the energy will not transfer to higher harmonics so in the frequency spectrum there is only one peak.
This is an important observation that does not only apply to single frequency sources. For each frequency component in the source function, higher-order harmonics will form as the wave progresses. In this thesis, however, we will keep our research to a single frequency source function.

2.4 Reflection and Transmission Coefficients

We know from the Introduction that the Westervelt equation is used in medical ultrasound to make an image of the insights of a body. To get this image the wave has to travel through a lot of different

tissues in the body. These tissues all have distinct parameters. These changes in parameters cause the arising of acoustic reflection and transmission. Now in this thesis we want to simulate this on a smaller scale. We assume that somewhere in the travel length the medium changes. We also assume that the wave is incident to the interface between the two media. We know that the displacement and the stresses of the incident wave plus the reflected wave have to be the continuous before and after the interface with the transmitted wave. We define the acoustic impedance as [12]

$$Z = \rho c \tag{2.11}$$

with ρ the ambient density of the medium and c the small signal sound speed of the wave in the medium. This gives us the following equations for the reflection and transmission coefficients [12]:

$$R = \frac{Z_2 - Z_1}{Z_1 + Z_2} \tag{2.12}$$

$$T = \frac{2Z_2}{Z_1 + Z_2} \tag{2.13}$$

The index of the impedance is used to describe the medium in which the impedance act. The index 1 is given to the medium in front of the interface and the index 2 is given to the medium after the interface. These reflection and transmission coefficients describe the amplitudes of the reflected and transmitted waves .

3 Using the Finite Element Method for solving the Westervelt equation

In this section we formulate a first order and second order finite element method to solve the one-dimensional Westervelt equation. We do this to analyze the difference between the numerical solutions of the first order finite element method and the second order finite element method. We want to research FEM as a method to solve the Westervelt equation, because it is a method that is not commonly used to solve the Westervelt equation. Another reason why we have chosen the FEM method is the fact that this thesis is actually a continuation of Bas Dirkse's Thesis [9]. In his thesis the applicability of first order FEM to the Westervelt equation is researched. He discovered that his own implementation of first order FEM was more accurate than more general build-in time solvers and it was very applicable for inhomogeneous domains. He concluded that it is more advantageous to use higher-order basis functions, since they reduce the relative error by a larger factor than the low-order basis functions for certain refinement. The higher-order FEM uses more auxiliary points in between every two element points than lower-order FEM, which should give a better approximation. To test this theory, we formulate a first order FEM and a second order FEM to solve the Westervelt equation. Another important change from the thesis of Bas is researching the effect of the attenuation term on the solution. That is why in this chapter the attenuation term is a part of the eventual finite element scheme that we deduce.
First, we start this chapter by writing the Westervelt equation into its weak form using a test function. Then we introduce our first order basis functions in space. The unknown solution and the test function can both be written as linear combinations of the basis functions. Next every term of the Westervelt equation is written into the linear combinations. The terms of the summations that are zero are subsequently determined and the summations are written into matrix vector multiplications. This is done in succession for the terms of the linear wave equation, the linear wave equation with the nonlinear term and the complete Westervelt equation. For these three equations we also determine how to solve them iteratively in time using a backwards differential scheme and successive approximation. The same approach is used for second order basis functions. So eventually we have for the first order and second order FEM three backward differential scheme to solve iteratively in time; one for the linear wave equation, one for the Westervelt equation without attenuation term and one for the complete Westervelt equation.

3.1 Weak Formulation

We want to cast the one-dimensional Westervelt equation into its weak formulation. To do so, we start with the Westervelt equation accompanied by suitable initial and boundary conditions:

$$\begin{cases} \frac{\partial^2 p}{\partial x^2} - \frac{1}{c_0^2}\frac{\partial^2 p}{\partial t^2} + \frac{\delta}{c_0^4}\frac{\partial^3 p}{\partial t^3} = -\frac{\beta}{\rho_0 c_0^4}\frac{\partial^2 p^2}{\partial t^2} \\ p(0,t) = f(t) \\ p(L,t) = 0 \\ p(x,0) = 0 \\ \left.\frac{\partial p}{\partial t}\right|_{t=0} = 0. \end{cases} \tag{3.1}$$

Note that $f(t)$ is the source function at $x = 0$. Then we multiply the PDE by a (arbitrary) test function $v(x)$ and integrate over the domain to get

$$\int_0^L \frac{\partial^2 p}{\partial x^2} v(x)dx - \frac{1}{c_0^2}\int_0^L \frac{\partial^2 p}{\partial t^2} v(x)dx + \frac{\delta}{c_0^4}\int_0^L \frac{\partial^3 p}{\partial t^3} v(x)dx = -\frac{\beta}{\rho_0 c_0^4}\int_0^L \frac{\partial^2 p^2}{\partial t^2} v(x)dx, \tag{3.2}$$

which is the first weak formulation. We need to add a constraint to (3.2) in order to make it equivalent to (3.1). We require that v is sufficiently smooth and satisfies $v(0) = v(L) = 0$, since p has essential boundary conditions at $x = 0$ and $x = L$ [13]. Now if we integrate the first term by parts to reduce the highest order spatial derivative present in the weak formulation, we get

$$\int_0^L \frac{\partial^2 p}{\partial x^2} v(x) dx = \left[\frac{\partial p}{\partial x} v(x)\right]_0^L - \int_0^L \frac{\partial p}{\partial x}\frac{dv}{dx} dx = -\int_0^L \frac{\partial p}{\partial x}\frac{dv}{dx} dx, \tag{3.3}$$

where the last equality follows if we note that v vanishes at $x = 0$ and $x = L$, because of the homogeneous boundary conditions. Plugging (3.3) into (3.2) and multiplying by -1 we get the desired weak formulation

$$\int_0^L \frac{\partial p}{\partial x}\frac{dv}{dx} dx + \frac{1}{c_0^2}\int_0^L \frac{\partial^2 p}{\partial t^2} v(x) dx - \frac{\delta}{c_0^4}\int_0^L \frac{\partial^3 p}{\partial t^3} v(x) dx = \frac{\beta}{\rho_0 c_0^4}\int_0^L \frac{\partial^2 p^2}{\partial t^2} v(x) dx, \tag{3.4}$$

which is now only valid for all smooth v satisfying $v(0) = v(L) = 0$, since we used this condition in (3.3).

3.2 First order FEM

3.2.1 Discretization of the linear wave equation

We start by taking a look at discretizing the linear wave equation, i.e. $\beta = 0$ and $\delta = 0$ in (3.1).

Discretization in space We can discretize the domain into n elements, and hence $n+1$ nodes in a one-dimensional problem, where $x_0 = 0$ and $x_n = L$. The unknown solution p can then be expanded as a linear combination

$$p(x,t) \approx \sum_{i=0}^{n} u_i(t)\phi_i(x) = u_0(t)\phi_0(x) + u_n(t)\phi_n(x) + \sum_{i=1}^{n-1} u_i(t)\phi_i(x), \tag{3.5}$$

where $\{\phi_i : i = 0, 1, ..., n\}$ is a set of suitable basis functions around x_i. Note that u_0 and u_n are known from the Dirichlet boundary conditions. We choose ϕ_i to be a piecewise linear function such that

$$\phi_i(x_j) = \begin{cases} 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases}. \tag{3.6}$$

In Figure 3.1 we have shown two of such basis functions.

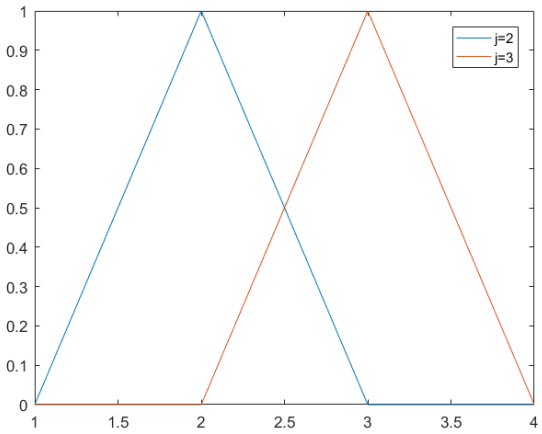

Figure 3.1: *Example of two adjacent basis functions*

In the figure above and the Matlab scripts we have used the basis functions that are defined as followed:

$$\phi_i(x) = \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}}, & \text{if } x \in [x_{i-1}, x_i] \\ \frac{x - x_{i+1}}{x_i - x_{i+1}}, & \text{if } x \in [x_i, x_{i+1}] \\ 0 & else \end{cases} \tag{3.7}$$

If we assume that the test function v can also be represented by (linear combinations of) these set of basis function $\{\phi_i : i = 0, 1, ..., n\}$ [13], we obtain the following finite element representation of the weak form:

$$\begin{cases} \text{Find the set } \{u_1(t), ..., u_{n-1}(t)\} \text{ such that,} \\ \sum_{i=0}^{n} \left(u_i \int_0^L \phi_i' \phi_j' dx + \frac{1}{c_0^2} \ddot{u}_i \int_0^L \phi_i \phi_j dx \right) = 0 \qquad \forall j \in \{1, ..., n-1\}. \end{cases} \tag{3.8}$$

It is important to note that $\int_0^L \phi_i' \phi_j' dx$ and $\int_0^L \phi_i \phi_j dx$ vanish if $|j - i| > 1$, because of the choice of the basis functions.

Looking more closely to the case $j = 1$ in (3.8), we see that there is a non-vanishing term containing u_0, which accounts for the (time-dependent) Dirichlet boundary condition. Since $\phi_0(0) = 1$ it follows that $u_0(t) = f(t)$, the source function which is known. Hence, we can bring that term to the right-hand-side of the equation and obtain

$$\sum_{i=1}^{n} \left(u_i \int_0^L \phi_i' \phi_1' dx + \frac{1}{c_0^2} \ddot{u}_i \int_0^L \phi_i \phi_1 dx \right) = -u_0 \int_0^L \phi_0' \phi_1' dx - \frac{1}{c_0^2} \ddot{u}_0 \int_0^L \phi_0 \phi_1 dx. \tag{3.9}$$

Of course when $u_0(t)$ is known, $\ddot{u}_0$ can either be calculated analytically or can be approximated numerically to an arbitrary accuracy.

The case $j = n - 1$ is similar to $j = 1$, where a non-vanishing term containing u_n exists. u_n accounts for the homogeneous Dirichlet boundary condition at $x = L$ and therefore $u_n(t) = 0$. So that term can just be eliminated from the set of equations.

Recognizing that the set of equations described by (3.8) can be written in matrix vector notation, we get

$$\mathbf{K}\mathbf{u} + \mathbf{M}\ddot{\mathbf{u}} = \mathbf{g}, \tag{3.10}$$

where the elements of the mass matrix $\mathbf{M}$ and the stiffness matrix $\mathbf{K}$ are

$$\begin{aligned} \mathbf{K}(i,j) &= \int_0^L \phi_i' \phi_j' dx \qquad \text{and} \\ \mathbf{M}(i,j) &= \frac{1}{c_0^2} \int_0^L \phi_i \phi_j dx \quad \forall i, j \in \{1, ..., n-1\} \end{aligned} \tag{3.11}$$

respectively.

Note again that $\mathbf{M}(i,j)$ and $\mathbf{K}(i,j)$ equal zero if $|j - i| > 1$ and thus $\mathbf{M}$ and $\mathbf{K}$ are very sparse $(n-1) \times (n-1)$ matrices. If we use the basis functions given by formula 4.7 we get the following values for the matrices.
For matrix K all possibilities are

$$\begin{aligned} \mathbf{K}(i,i) &= \int_0^L \phi_i' \phi_i' dx = \int_{x_{i-1}}^{x_i} \frac{1}{(x_j - x_{j-1})^2} dx + \int_{x_i}^{x_{i+1}} \frac{1}{(x_j - x_{j+1})^2} dx \\ &= \frac{1}{x_i - x_{i-1}} + \frac{1}{x_{i+1} - x_i} \\ \mathbf{K}(i,i+1) &= \int_0^L \phi_i' \phi_{i+1}' dx = \int_{x_i}^{x_{i+1}} \frac{1}{x_i - x_{i+1}} \frac{1}{x_{i+1} - x_i} dx \\ &= -\frac{1}{x_{i+1} - x_i} \\ \mathbf{K}(i+1,i) &= \int_0^L \phi_{i+1}' \phi_i' dx = -\frac{1}{x_{i+1} - x_i} \end{aligned} \tag{3.12}$$

For matrix M all possibilities are

$$\begin{aligned} \mathbf{M}(i,i) &= \frac{1}{c_0^2} \int_0^L \phi_i \phi_i dx = \frac{1}{c_0^2} \int_{x_{i-1}}^{x_i} \left(\frac{x - x_{i-1}}{x_i - x_{i-1}}\right)^2 dx + \frac{1}{c_0^2} \int_{x_i}^{x_{i+1}} \left(\frac{x - x_{i+1}}{x_i - x_{i+1}}\right)^2 dx \\ &= \frac{1}{3c_0^2}(x_i - x_{i-1}) + \frac{1}{3c_0^2}(x_{i+1} - x_i) \\ \mathbf{M}(i,i+1) &= \frac{1}{c_0^2} \int_0^L \phi_i \phi_{i+1} dx = \frac{1}{c_0^2} \int_{x_i}^{x_{i+1}} \frac{x - x_{i+1}}{x_i - x_{i+1}} \frac{x - x_i}{x_{i+1} - x_i} dx \\ &= \frac{1}{6c_0^2}(x_{i+1} - x_i) \\ \mathbf{M}(i+1,i) &= \frac{1}{c_0^2} \int_0^L \phi_{i+1} \phi_i dx = \frac{1}{6c_0^2}(x_{i+1} - x_i) \end{aligned} \tag{3.13}$$

and the elements of $\mathbf{g}$ are given by

$$\mathbf{g}(j) = \begin{cases} -u_0(t)\int_0^L \phi_0'\phi_1' dx - \frac{1}{c_0^2}\ddot{u}_0(t)\int_0^L \phi_0\phi_1 dx, & \text{if } j = 1 \\ 0, & \text{if } j \neq 1. \end{cases} \tag{3.14}$$

The right-hand-side vector $\mathbf{g}$ is a column vector of length $n-1$ and only the first value is not equal to zero because of the boundary condition.

Discretization in time The semi-discretized system (3.43) still needs to be discretized in time. We can do that by use of finite difference methods. We would like to use a backward differential formula accurate up to $\mathcal{O}\big((\Delta t)^2\big)$. This is given by (see Appendix B for derivation)

$$\mathbf{K}\mathbf{u^k} + \mathbf{M}\frac{2\mathbf{u^k} - 5\mathbf{u^{k-1}} + 4\mathbf{u^{k-2}} - \mathbf{u^{k-3}}}{(\Delta t)^2} = \mathbf{g^k}, \tag{3.15}$$

where $\mathbf{u^k}$ is the vector containing the coefficients $\{u_1, ..., u_{n+1}\}$ at time t_k. Solving (3.15) for $\mathbf{u^k}$, we arrive at

$$((\Delta t)^2\mathbf{K} + 2\mathbf{M})\mathbf{u^k} = 5\mathbf{M}\mathbf{u^{k-1}} - 4\mathbf{M}\mathbf{u^{k-2}} + \mathbf{M}\mathbf{u^{k-3}} + (\Delta t)^2\mathbf{g^k}, \tag{3.16}$$

which can be solved quickly for $\mathbf{u^k}$ by Matlab's backslash operator without inverting the matrix $((\Delta t)^2\mathbf{K} + 2\mathbf{M})$. So $\mathbf{u^k}$ can be calculated from the values of $\mathbf{u}$ at times t_{k-1}, t_{k-2} and t_{k-3}. Using iteration the solution can be solved in evolving time.

We identify $\mathbf{u^0}$ as the initial condition, discretized in space. But in order to initiate the above scheme, we need an $\mathbf{u^1}$ and a $\mathbf{u^2}$. We can get $\mathbf{u^1}$ from the initial condition on the first time derivative of the pressure field. If we discretize $\frac{\partial p}{\partial t}|_{t=0}$ as $\dot{\mathbf{u}}^\mathbf{0}$, then we can estimate $\mathbf{u^1}$ by

$$\mathbf{u^1} = \mathbf{u^0} + \Delta t\dot{\mathbf{u}}^\mathbf{0}. \tag{3.17}$$

To solve for $\mathbf{u^2}$, we start with one time step of first-order accuracy (i.e. use the fifth equation in (B.6)), which only depends on the previous two time steps. So using this scheme the $\mathbf{u^2}$ can be found. After that the backward differential formula with second order accuracy will be used.

A Matlab code implementing this finite element scheme can be found in Appendix C.1.

3.2.2 Discretization of the nonlinear term

Discretization in space We start by expanding the time derivative in the right-hand-side term of (3.4), which gives us

$$\frac{\partial^2 p^2}{\partial t^2} = \frac{\partial}{\partial t}\left(\frac{\partial}{\partial t}p^2\right) = \frac{\partial}{\partial t}\left(2p\frac{\partial p}{\partial t}\right) = 2\left(\frac{\partial p}{\partial t}\right)^2 + 2p\frac{\partial^2 p}{\partial t^2}. \tag{3.18}$$

Using the same basis function and linear expansion of $p(x,t)$ as in Section 3.2.1, we get the following set of terms on the right-hand-side of (3.8)

$$\frac{2\beta}{\rho_0 c_0^4}\int_0^L \Big(\sum_{i=0}^n \dot{u}_i\phi_i\Big)\Big(\sum_{m=0}^n \dot{u}_m\phi_m\Big)\phi_j dx + \frac{2\beta}{\rho_0 c_0^4}\int_0^L \Big(\sum_{i=0}^n \ddot{u}_i\phi_i\Big)\Big(\sum_{m=0}^n u_m\phi_m\Big)\phi_j dx, \tag{3.19}$$

for the j^{th} equation, where $j \in \{1, 2, ..., n-1\}$. Interchanging summation over i and integration, we get

$$\sum_{i=0}^n \dot{u}_i\frac{2\beta}{\rho_0 c_0^4}\int_0^L \Big(\sum_{m=0}^n \dot{u}_m\phi_m\Big)\phi_i\phi_j dx + \sum_{i=0}^n \ddot{u}_i\frac{2\beta}{\rho_0 c_0^4}\int_0^L \Big(\sum_{m=0}^n u_m\phi_m\Big)\phi_i\phi_j dx. \tag{3.20}$$

Now if we recognize that both integrals vanish if $m \neq j-1, j, j+1$ (because of the choice of the basis functions), we can eliminate the summation over m to arrive at

$$\begin{array}{l}\sum_{i=0}^{n}\dot{u}_i\left(\dot{u}_{j-1}\frac{2\beta}{\rho_0 c_0^4}\int_0^L\phi_{j-1}\phi_i\phi_j dx+\dot{u}_j\frac{2\beta}{\rho_0 c_0^4}\int_0^L\phi_j\phi_i\phi_j dx+\dot{u}_{j+1}\frac{2\beta}{\rho_0 c_0^4}\int_0^L\phi_{j+1}\phi_i\phi_j dx\right)\quad+\\ \sum_{i=0}^{n}\ddot{u}_i\left(u_{j-1}\frac{2\beta}{\rho_0 c_0^4}\int_0^L\phi_{j-1}\phi_i\phi_j dx+u_j\frac{2\beta}{\rho_0 c_0^4}\int_0^L\phi_j\phi_i\phi_j dx+u_{j+1}\frac{2\beta}{\rho_0 c_0^4}\int_0^L\phi_{j+1}\phi_i\phi_j dx\right),\end{array}\tag{3.21}$$

for the j^{th} equation, where again $j \in \{1, ..., n-1\}$. (3.21) is the term that should be added to the right-hand-side of (3.8).

Identifying three new matrices $\mathbf{C_1}$, $\mathbf{C_2}$ and $\mathbf{C_3}$ with elements

$$\begin{array}{rcll}\mathbf{C_1}(i,j) &=& \frac{2\beta}{\rho_0 c_0^4}\int_0^L\phi_{j-1}\phi_i\phi_j dx, & \\ \mathbf{C_2}(i,j) &=& \frac{2\beta}{\rho_0 c_0^4}\int_0^L\phi_j\phi_i\phi_j dx & \text{and}\\ \mathbf{C_3}(i,j) &=& \frac{2\beta}{\rho_0 c_0^4}\int_0^L\phi_{j+1}\phi_i\phi_j dx & \forall i,j\in\{1,...,n-1\}\end{array}\tag{3.22}$$

respectively, we can express (3.21) more compactly as

$$\begin{array}{l}\sum_{i=0}^{n}\dot{u}_i\left(\dot{u}_{j-1}\mathbf{C_1}(i,j)+\dot{u}_j\mathbf{C_2}(i,j)+\dot{u}_{j+1}\mathbf{C_3}(i,j)\right)\quad+\\ \sum_{i=0}^{n}\ddot{u}_i\left(u_{j-1}\mathbf{C_1}(i,j)+u_j\mathbf{C_2}(i,j)+u_{j+1}\mathbf{C_3}(i,j)\right).\end{array}\tag{3.23}$$

In each of the equations $j = 1, ..., n-1$ where the term between brackets does not vanish for $i = 0$ and $i = n$ (this is only the case for $j = 1$ and $j = n-1$ because of the choice of basis functions), those contributions to the summation, which account for the Dirichlet boundary conditions, are separated and absorbed into the $\mathbf{g}$ vector as mentioned in Section 3.2.1.

Now if we look carefully at the first term of (3.23) we see that for a fixed j^{th} equation, all elements $\mathbf{C_1}(i,j) \quad \forall i \in \{1, ..., n-1\}$ are multiplied by $\dot{u}_{j-1}$. All elements $\mathbf{C_2}(i,j) \quad \forall i \in \{1, ..., n-1\}$ are multiplied by $\dot{u}_j$ and all elements $\mathbf{C_3}(i,j) \quad \forall i \in \{1, ..., n-1\}$ are multiplied by $\dot{u}_{j+1}$. The second term of (3.23) is similar except each row of a $\mathbf{C}$ matrix is multiplied by u_{j-1}, u_j and u_{j+1} respectively. This applies for all $j = 1, ..., n-1$. Therefore (3.23) can be written in full matrix form as

$$\begin{array}{l}\left(\begin{bmatrix}\dot{u}_0 & 0 & \cdots & 0\\ 0 & \dot{u}_1 & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ 0 & \cdots & 0 & \dot{u}_{n-2}\end{bmatrix}\mathbf{C_1}+\begin{bmatrix}\dot{u}_1 & 0 & \cdots & 0\\ 0 & \dot{u}_2 & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ 0 & \cdots & 0 & \dot{u}_{n-1}\end{bmatrix}\mathbf{C_2}+\begin{bmatrix}\dot{u}_2 & 0 & \cdots & 0\\ 0 & \dot{u}_3 & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ 0 & \cdots & 0 & \dot{u}_n\end{bmatrix}\mathbf{C_3}\right)\dot{\mathbf{u}}+\\ \left(\begin{bmatrix}u_0 & 0 & \cdots & 0\\ 0 & u_1 & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ 0 & \cdots & 0 & u_{n-2}\end{bmatrix}\mathbf{C_1}+\begin{bmatrix}u_1 & 0 & \cdots & 0\\ 0 & u_2 & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ 0 & \cdots & 0 & u_{n-1}\end{bmatrix}\mathbf{C_2}+\begin{bmatrix}u_2 & 0 & \cdots & 0\\ 0 & u_3 & \ddots & \vdots\\ \vdots & \ddots & \ddots & 0\\ 0 & \cdots & 0 & u_n\end{bmatrix}\mathbf{C_3}\right)\ddot{\mathbf{u}}.\end{array}\tag{3.24}$$

Note that all explicitly given matrices in (3.50) are diagonal matrices, such that when matrix multiplication is used, the the first term on the diagonal is multiplied by the first column of $\mathbf{C_x}$ and so on. Also note that we know u_0 and u_n from the given Dirichlet boundary conditions and there derivatives can therefore be calculated analytically or numerically.

Now name the matrices in the first row of (3.50) from left to right $\mathbf{H_{-1}}(\dot{\mathbf{u}})$, $\mathbf{H_0}(\dot{\mathbf{u}})$, $\mathbf{H_1}(\dot{\mathbf{u}})$ and in the second row $\mathbf{I_{-1}}(\mathbf{u})$, $\mathbf{I_0}(\mathbf{u})$, $\mathbf{I_1}(\mathbf{u})$, where the bracket notation means that $\mathbf{H}$ is a function of $\dot{\mathbf{u}}$ and $\mathbf{I}$ is a function of $\mathbf{u}$.

Defining

$$\begin{aligned} \mathbf{N_1}(\dot{\mathbf{u}}) &= \mathbf{H_{-1}}(\dot{\mathbf{u}})\mathbf{C_1} + \mathbf{H_0}(\dot{\mathbf{u}})\mathbf{C_2} + \mathbf{H_1}(\dot{\mathbf{u}})\mathbf{C_3} \quad \text{and} \\ \mathbf{N_2}(\mathbf{u}) &= \mathbf{I_{-1}}(\mathbf{u})\mathbf{C_1} + \mathbf{I_0}(\mathbf{u})\mathbf{C_2} + \mathbf{I_1}(\mathbf{u})\mathbf{C_3} \quad , \end{aligned} \tag{3.25}$$

we can finally state the semi-discrete matrix form of the finite element formulation of the one-dimensional Westervelt equation:

$$\mathbf{Ku} + \mathbf{M}\ddot{\mathbf{u}} = \mathbf{N_1}(\dot{\mathbf{u}})\dot{\mathbf{u}} + \mathbf{N_2}(\mathbf{u})\ddot{\mathbf{u}} + \mathbf{g}, \tag{3.26}$$

where $\mathbf{g}$ now also includes the Dirichlet boundary condition contributions of the nonlinear term:

$$\mathbf{g}(j) = \begin{cases} -\int_0^L \phi_0'\phi_1' dx - \frac{1}{c_0^2}\ddot{u}_0 \int_0^L \phi_0\phi_1 dx \\ +\dot{u}_0 \left(\dot{u}_0 \frac{2\beta}{\rho_0 c_0^4} \int_0^L \phi_0\phi_0\phi_1 dx + \dot{u}_1 \frac{2\beta}{\rho_0 c_0^4} \int_0^L \phi_1\phi_0\phi_1 dx \right) \\ +\ddot{u}_0 \left(u_0 \frac{2\beta}{\rho_0 c_0^4} \int_0^L \phi_0\phi_0\phi_1 dx + u_1 \frac{2\beta}{\rho_0 c_0^4} \int_0^L \phi_1\phi_0\phi_1 dx \right), & \text{if } j = 1 \\ 0, & \text{if } j \neq 1. \end{cases} \tag{3.27}$$

Discretization in time Now we focus on the time discretization of (3.52). As in the linear case, we use a second-order backward differential scheme. But for these equations we also need first order backward differential scheme. The derivation of expressions for these derivatives of first order and second order accuracy can be found in Appendix B. Applying them to (3.52) gives us

$$\begin{aligned} &\mathbf{Ku^k} + \mathbf{M}\frac{\mathbf{2u^k - 5u^{k-1} + 4u^{k-2} - u^{k-3}}}{(\Delta t)^2} = \\ &\mathbf{N_1}\left(\frac{\mathbf{3u^k - 4u^{k-1} + u^{k-2}}}{\mathbf{2\Delta t}}\right)\frac{\mathbf{3u^k - 4u^{k-1} + u^{k-2}}}{2\Delta t} + \mathbf{N_2}(\mathbf{u^k})\frac{\mathbf{2u^k - 5u^{k-1} + 4u^{k-2} - u^{k-3}}}{(\Delta t)^2} + \mathbf{g^k}, \end{aligned} \tag{3.28}$$

where just as before we find $\mathbf{u^1}$ from the given initial condition $\mathbf{u^0}$ and its derivative $\dot{\mathbf{u}}^\mathbf{0}$, and find $\mathbf{u^2}$ from a single first-order BDF time step.

(3.28) is an implicit system of nonlinear equations in unknown $\mathbf{u^k}$ which can not be made explicit in a straight forward way. Therefore we solve it by the method of successive approximation. Because the time step Δt is small, we expect this method to converge in just a few iterations.

During each time step we start by estimating $\mathbf{u^k}$, i.e. by ${}^\mathbf{0}\mathbf{u^k} = \mathbf{u^k_{linear}}$, where $\mathbf{u^k_{linear}}$ is the solution to (3.16) and the left superscript 0 means that it is an initial guess for $\mathbf{u^k}$. We can then, based on this guess, calculate the matrices $\mathbf{N_1}$ and $\mathbf{N_2}$. Then we can solve (3.28) and obtain ${}^\mathbf{1}\mathbf{u^k}$. Now we can repeat calculating $\mathbf{N_1}$ and $\mathbf{N_2}$, and solving (3.28) until $\max(|{}^\mathbf{r}\mathbf{u^k} - {}^{\mathbf{r-1}}\mathbf{u^k}|) < \epsilon$, where ϵ is a small number defining the allowed tolerance. Then we set $\mathbf{u^k} = {}^\mathbf{r}\mathbf{u^k}$ and we have finished this time step. This process, which is called successive approximation, is summarized step by step below:

1. Initiate by guessing ${}^\mathbf{0}\mathbf{u^k} = \mathbf{u^k_{linear}}$.
2. Calculate $\mathbf{N_1}$ and $\mathbf{N_2}$.
3. Solve (3.28) for ${}^\mathbf{r}\mathbf{u^k}$
4. If $\max(|{}^\mathbf{r}\mathbf{u^k} - {}^{\mathbf{r-1}}\mathbf{u^k}|) < \epsilon$ go to 5.
   Otherwise go to 2.
5. set $\mathbf{u^k} = {}^\mathbf{r}\mathbf{u^k}$.

3.2.3 Discretization of the attenuation term

Discretization in space In the former sections we forgot about one the term, the term of attenuation. Now we want to bring that term into our numeric scheme as well. To make it easier we will forget about the nonlinear term for now.
First we assume that we use the same first order basis functions as we did in (3.7). The unknown solution p and the test function v can be represented by (a linear combinations of) these set of basis functions. Using this we obtain the following finite element representation of the weak form:

$$\begin{cases} \text{Find the set } \{u_1(t), ..., u_{n-1}(t)\} \text{ such that,} \\ \sum_{i=0}^{n} \left(u_i \int_0^L \phi_i' \phi_j' dx + \frac{1}{c_0^2} \ddot{u}_i \int_0^L \phi_i \phi_j dx - \frac{\delta}{c_0^4} \dddot{u}_i \int_0^L \phi_i \phi_j dx \right) = 0 \qquad \forall j \in \{1, ..., n-1\}. \end{cases} \tag{3.29}$$

It is important to note that $\int_0^L \phi_i' \phi_j' dx$ and $\int_0^L \phi_i \phi_j dx$ vanish if $|j - i| > 1$, because of the choice of the basis functions.
Also for $j = 1$ and $j = n-1$ the Dirichlet boundary conditions still have the same influence, but there is now an extra term. So for $j = 1$ we obtain the following equation

$$\begin{aligned} \sum_{i=1}^{n} \left(u_i \int_0^L \phi_i' \phi_1' dx + \frac{1}{c_0^2} \ddot{u}_i \int_0^L \phi_i \phi_j dx - \frac{\delta}{c_0^4} \dddot{u}_i \int_0^L \phi_i \phi_j dx \right) = \\ -u_0 \int_0^L \phi_0' \phi_1' dx - \frac{1}{c_0^2} \ddot{u}_0 \int_0^L \phi_0 \phi_1 dx + \frac{\delta}{c_0^4} \dddot{u}_0 \int_0^L \phi_0 \phi_1 dx \end{aligned} \tag{3.30}$$

Of course when $u_0(t)$ is known, $\ddot{u}_0$ and $\dddot{u}_0$ can either be calculated analytically or can be approximated numerically to an arbitrary accuracy. Just as before for $j = n-1$ the term of u_n can be eliminated. Namely because of the homogeneous Dirichlet boundary condition at $x = L$ the value of $u_n(t)$ is always zero.
Recognizing that the set of equations described by (3.29) can be written in matrix vector notation, we get

$$\mathbf{K}\mathbf{u} + \mathbf{M}\Big(\frac{1}{c_0^2} \ddot{\mathbf{u}} - \frac{\delta}{c_0^4} \dddot{\mathbf{u}} \Big) = \mathbf{g}, \tag{3.31}$$

where the elements of the mass matrix $\mathbf{M}$ and the stiffness matrix $\mathbf{K}$ are

$$\begin{aligned} \mathbf{K}(i,j) &= \int_0^L \phi_i' \phi_j' dx \quad \text{and} \\ \mathbf{M}(i,j) &= \int_0^L \phi_i \phi_j dx \quad \forall i, j \in \{1, ..., n-1\} \end{aligned} \tag{3.32}$$

respectively.
Because of the chosen basis functions $\mathbf{M}$ and $\mathbf{K}$ are very sparse $(n-1) \times (n-1)$ matrices. Their values are given by (3.32). $\mathbf{g}$ is a column vector of length $n-1$ and only the first value is not equal because of the boundary condition. The elements of $\mathbf{g}$ are given by

$$\mathbf{g}(j) = \begin{cases} -u_0(t) \int_0^L \phi_0' \phi_1' dx - \frac{1}{c_0^2} \ddot{u}_0(t) \int_0^L \phi_0 \phi_1 dx + \frac{\delta}{c_0^4} \dddot{u}_0(t) \int_0^L \phi_0 \phi_1 dx, & \text{if } j = 1 \\ 0, & \text{if } j \neq 1. \end{cases} \tag{3.33}$$

If the non-linear term is taken into account as well, the set of equations that describes the system can be described as followed

$$\mathbf{K}\mathbf{u} + \mathbf{M}\Big(\frac{1}{c_0^2} \ddot{\mathbf{u}} - \frac{\delta}{c_0^4} \dddot{\mathbf{u}} \Big) = \mathbf{N_1}(\dot{\mathbf{u}})\dot{\mathbf{u}} + \mathbf{N_2}(\mathbf{u})\ddot{\mathbf{u}} + \mathbf{g}, \tag{3.34}$$

$\mathbf{N_1}$ and $\mathbf{N_2}$ are the same as in the previous section. $\mathbf{M}$ and $\mathbf{K}$ are the same as defined as in (3.32).

Only the vector $\mathbf{g}$ has changed.

$$\mathbf{g}(j) = \begin{cases} -\int_0^L \phi_0'\phi_1' dx - \frac{1}{c_0^2}\ddot{u}_0 \int_0^L \phi_0\phi_1 dx + \frac{\delta}{c_0^4}\ddot{u}_0(t) \int_0^L \phi_0\phi_1 dx \\ +\dot{u}_0 \left(\dot{u}_0 \frac{2\beta}{\rho_0 c_0^4} \int_0^L \phi_0\phi_0\phi_1 dx + \dot{u}_1 \frac{2\beta}{\rho_0 c_0^4} \int_0^L \phi_1\phi_0\phi_1 dx \right) \\ +\ddot{u}_0 \left(u_0 \frac{2\beta}{\rho_0 c_0^4} \int_0^L \phi_0\phi_0\phi_1 dx + u_1 \frac{2\beta}{\rho_0 c_0^4} \int_0^L \phi_1\phi_0\phi_1 dx \right), & \text{if } j = 1 \\ 0, & \text{if } j \neq 1. \end{cases} \tag{3.35}$$

Discretization in time The semi-discretized system (3.31) still needs to be discretized in time. We can do that by use of finite difference methods. We would like to use a backward differential formula accurate up to $\mathcal{O}\big((\Delta t)^2\big)$. This is given by (see Appendix B for derivation)

$$\mathbf{K}\mathbf{u}^{\mathbf{k}} + \mathbf{M}\Big(\frac{1}{c_0^2}\frac{2\mathbf{u}^{\mathbf{k}} - 5\mathbf{u}^{\mathbf{k-1}} + 4\mathbf{u}^{\mathbf{k-2}} - \mathbf{u}^{\mathbf{k-3}}}{(\Delta t)^2} - \frac{\delta}{c_0^4}\frac{2.5\mathbf{u}^{\mathbf{k}} - 9\mathbf{u}^{\mathbf{k-1}} + 12\mathbf{u}^{\mathbf{k-2}} - 7\mathbf{u}^{\mathbf{k-3}} + 1.5\mathbf{u}^{\mathbf{k-4}}}{(\Delta t)^3}\Big) = \mathbf{g}^{\mathbf{k}}, \tag{3.36}$$

where $\mathbf{u}^{\mathbf{k}}$ is the vector containing the coefficients $\{u_1, ..., u_{n+1}\}$ at time t_k. Solving (3.36) for $\mathbf{u}^{\mathbf{k}}$, we arrive at

$$\begin{aligned} \Big((\Delta t)^3\mathbf{K} + (\frac{2\Delta t}{c_0^2} - \frac{2.5\delta}{c_0^4})\mathbf{M}\Big)\mathbf{u}^{\mathbf{k}} = \frac{5\Delta t}{c_0^2}\mathbf{M}\mathbf{u}^{\mathbf{k-1}} - \frac{4\Delta t}{c_0^2}\mathbf{M}\mathbf{u}^{\mathbf{k-2}} + \\ \frac{\Delta t}{c_0^2}\mathbf{M}\mathbf{u}^{\mathbf{k-3}} - \frac{9\delta}{c_0^4}\mathbf{u}^{\mathbf{k-1}} + \frac{12\delta}{c_0^4}\mathbf{u}^{\mathbf{k-2}} - \frac{7\delta}{c_0^4}\mathbf{u}^{\mathbf{k-3}} + \frac{1.5\delta}{c_0^4}\mathbf{u}^{\mathbf{k-4}} + (\Delta t)^3\mathbf{g}^{\mathbf{k}}, \end{aligned} \tag{3.37}$$

which can be solved quickly for $\mathbf{u}^{\mathbf{k}}$ by Matlab's backslash operator without inverting the matrix $((\Delta t)^2\mathbf{K} + 2\mathbf{M})$. So $\mathbf{u}^{\mathbf{k}}$ can be calculated from the values of $\mathbf{u}$ at times t_{k-1}, t_{k-2}, t_{k-3} and t_{k-4}. Using iteration the solution can be solved in evolving time.

We identify $\mathbf{u}^{\mathbf{0}}$ as the initial condition, discretized in space. But in order to initiate the above scheme, we need an $\mathbf{u}^{\mathbf{1}}$ and a $\mathbf{u}^{\mathbf{2}}$. We can get $\mathbf{u}^{\mathbf{1}}$ from the initial condition on the first time derivative of the pressure field. If we discretize $\frac{\partial p}{\partial t}|_{t=0}$ as $\dot{\mathbf{u}}^{\mathbf{0}}$, then we can estimate $\mathbf{u}^{\mathbf{1}}$ by

$$\mathbf{u}^{\mathbf{1}} = \mathbf{u}^{\mathbf{0}} + \Delta t\dot{\mathbf{u}}^{\mathbf{0}}. \tag{3.38}$$

To derive the value of $\mathbf{u}^{\mathbf{2}}$ we use the sixth equation in Appendix B. For this differential formula we need to know virtual element $\mathbf{u}^{-\mathbf{1}}$. Because of the initial conditions we derive that this is equal to $\mathbf{u}^{\mathbf{1}}$. We also need element $\mathbf{u}^{\mathbf{3}}$ before we can use equation 3.37. These can also be found using the lower accuracy order backward differential formula(use the sixth equation in Appendix B).

If we also take the nonlinear term into account, we get the following scheme:

$$\begin{aligned} \mathbf{K}\mathbf{u}^{\mathbf{k}} + \mathbf{M}\frac{2\mathbf{u}^{\mathbf{k}} - 5\mathbf{u}^{\mathbf{k-1}} + 4\mathbf{u}^{\mathbf{k-2}} - \mathbf{u}^{\mathbf{k-3}}}{c_0^2 * (\Delta t)^2} - \frac{\delta}{c_0^4}\mathbf{M}\frac{-2.5\mathbf{u}^{\mathbf{k}} + 9\mathbf{u}^{\mathbf{k-1}} - 12\mathbf{u}^{\mathbf{k-2}} + 7\mathbf{u}^{\mathbf{k-3}} - 1.5\mathbf{u}^{\mathbf{k-4}}}{(\Delta t)^2} = \\ \mathbf{N_1}(\frac{\mathbf{3u^k - 4u^{k-1} + u^{k-2}}}{\mathbf{2\Delta t}})\frac{3\mathbf{u}^{\mathbf{k}} - 4\mathbf{u}^{\mathbf{k-1}} + \mathbf{u}^{\mathbf{k-2}}}{2\Delta t} + \mathbf{N_2}(\mathbf{u}^{\mathbf{k}})\frac{2\mathbf{u}^{\mathbf{k}} - 5\mathbf{u}^{\mathbf{k-1}} + 4\mathbf{u}^{\mathbf{k-2}} - \mathbf{u}^{\mathbf{k-3}}}{(\Delta t)^2} + \mathbf{g}^{\mathbf{k}}, \end{aligned} \tag{3.39}$$

The first values can be found the same way as described above. The rest of the values are found using the method of successive approximation as is described in 3.2.2. A Matlab code implementing this finite element scheme can be found in Appendix C.3.

3.3 Second order FEM

3.3.1 Discretization of the linear wave equation

Now that we have solved the Westervelt equation using the first order FEM, we would like to solve it using a second order FEM. We want to use second order FEM because they converge faster to the

eventual solution than the first order FEM. This happens because the second order basis functions are dependent on more points in the domain.
Now we are going to do the same for second order FEM as first order FEM. We divide the domain again into n+1 elements. Each element e_j has two element points (x_j and x_{j+1}) and one auxilary point ($x_{j+\frac{1}{2}}$). So actually the domain is divided into $2n+1$ points. We can still expand the unknown solution of p using a linear combination of basis functions, but now the basis functions are of the second order.
The second order basis functions are defined as follows

$$\phi_i(x) = \begin{cases} \frac{(x-x_{i-1})(x-x_{i-\frac{1}{2}})}{(x_i-x_{i-1})(x_i-x_{i-\frac{1}{2}})}, & \text{if } x \in [x_{i-1}, x_i] \\ \frac{(x-x_{i+1})(x-x_{i+\frac{1}{2}})}{(x_i-x_{i+1})(x_i-x_{i+\frac{1}{2}})}, & \text{if } x \in [x_i, x_{i+1}] \\ 0 & else \end{cases} \tag{3.40}$$

In Figure 3.2 an example of a second order basis function is shown that satisfies the conditions above.

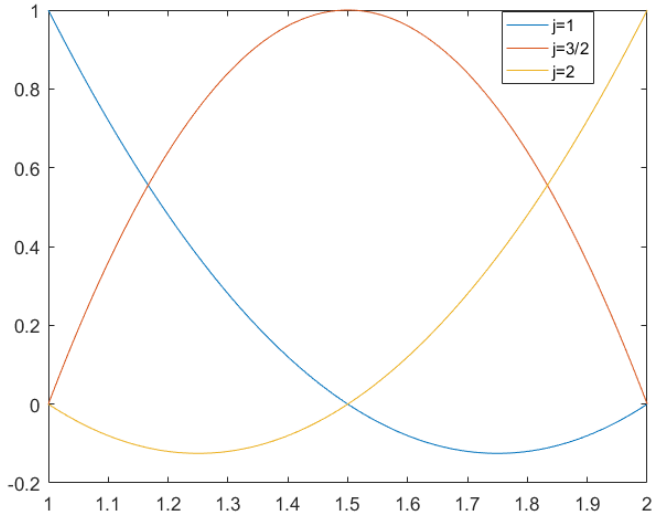

Figure 3.2: *Example of a quadratic basis function*

We can also expand the test function v in the above basis functions. If we do this we get the following finite element representation of the weak form:

$$\begin{cases} \text{Find the set } \{u_{\frac{1}{2}}(t), u_1(t), ..., u_{n-1}(t), u_{n-\frac{1}{2}}(t)\} \text{ such that,} \\ \sum_{i=0,\frac{1}{2},1,...}^{N} \left(u_i \int_0^L \phi_i' \phi_j' dx + \frac{1}{c_0^2} \ddot{u}_i \int_0^L \phi_i \phi_j dx \right) = 0 \qquad \forall j \in \{\frac{1}{2}, ..., n-\frac{1}{2}\} \end{cases} \tag{3.41}$$

It is important to note that $\int_0^L \phi_i' \phi_j' dx$ and $\int_0^L \phi_i \phi_j dx$ vanish if $|j-i| > 1$, because of the choice of the basis functions.

Looking more closely to the case $j = 1$ in (3.41), we see that there is a non-vanishing term containing u_0, which accounts for the (time-dependent) Dirichlet boundary condition. Different to the first order basis equation there is also a non-vanishing term containing u_0 in the case of $j = \frac{1}{2}$. Since $\phi_0(0) = 1$ it follows that $u_0(t) = f(t)$, the source function which is known. Hence, we can bring the terms to the right-hand-side of the equations and obtain

$$\begin{aligned} \sum_{i=\frac{1}{2}}^{n} \left(u_i \int_0^L \phi_i' \phi_{\frac{1}{2}}' dx + \frac{1}{c_0^2} \ddot{u}_i \int_0^L \phi_i \phi_{\frac{1}{2}} dx \right) &= -u_0 \int_0^L \phi_0' \phi_{\frac{1}{2}}' dx - \frac{1}{c_0^2} \ddot{u}_0 \int_0^L \phi_0 \phi_{\frac{1}{2}} dx \\ \sum_{i=\frac{1}{2}}^{n} \left(u_i \int_0^L \phi_i' \phi_1' dx + \frac{1}{c_0^2} \ddot{u}_i \int_0^L \phi_i \phi_1 dx \right) &= -u_0 \int_0^L \phi_0' \phi_1' dx - \frac{1}{c_0^2} \ddot{u}_0 \int_0^L \phi_0 \phi_1 dx \end{aligned} \tag{3.42}$$

Of course when $u_0(t)$ is known, $\ddot{u}_0$ can either be calculated analytically or can be approximated numerically to an arbitrary accuracy.

The case $j = n-1$ and $j = n-\frac{1}{2}$ is similar to $j = 1$ and $j = \frac{1}{2}$, where a non-vanishing term containing u_n exists. u_n accounts for the homogeneous Dirichlet boundary condition at $x = L$ and therefore $u_n(t) = 0$. Those terms can just be eliminated from the set of equations.

Recognizing that the set of equations described by (3.41) can be written in matrix vector notation, we get

$$\mathbf{K}\mathbf{u} + \mathbf{M}\ddot{\mathbf{u}} = \mathbf{g}, \tag{3.43}$$

where the elements of the mass matrix $\mathbf{M}$ and the stiffness matrix $\mathbf{K}$ are

$$\begin{aligned} \mathbf{K}(2i,2j) &= \textstyle\int_0^L \phi_i' \phi_j' dx \qquad \text{and} \\ \mathbf{M}(2i,2j) &= \frac{1}{c_0^2}\textstyle\int_0^L \phi_i \phi_j dx \quad \forall i,j \in \{\frac{1}{2}, 1, ..., n-1, n-\frac{1}{2}\} \end{aligned} \tag{3.44}$$

respectively.

Note again that $\mathbf{M}(2i,2j)$ and $\mathbf{K}(2i,2j)$ equal zero if $|j-i| > 1$ and thus $\mathbf{M}$ and $\mathbf{K}$ are very sparse $(2n-1) \times (2n-1)$ matrices. The elements of $\mathbf{g}$ are given by

$$\mathbf{g}(2j) = \begin{cases} -u_0(t)\int_0^L \phi_0' \phi_{\frac{1}{2}}' dx - \frac{1}{c_0^2}\ddot{u}_0(t)\int_0^L \phi_0 \phi_{\frac{1}{2}} dx, & \text{if } j = \frac{1}{2} \\ -u_0(t)\int_0^L \phi_0' \phi_1' dx - \frac{1}{c_0^2}\ddot{u}_0(t)\int_0^L \phi_0 \phi_1 dx, & \text{if } j = 1 \\ 0, & else \end{cases} \tag{3.45}$$

The right-hand-side vector $\mathbf{g}$ is a column vector of length $2n-1$ and only the first two values are not equal to zero because of the time-dependent Dirichlet boundary condition.

The discretization in time is the same for the second order as the first order FEM.

3.3.2 Discretization of the nonlinear term

We start by expanding the time derivative in the right-hand-side term of (3.4) in the same way as we did in 3.2.2. Then the following equation is derived

$$\frac{2\beta}{\rho_0 c_0^4}\int_0^L \Big(\sum_{i=0}^n \dot{u}_i\phi_i\Big)\Big(\sum_{m=0}^n \dot{u}_m\phi_m\Big)\phi_j dx + \frac{2\beta}{\rho_0 c_0^4}\int_0^L \Big(\sum_{i=0}^n \ddot{u}_i\phi_i\Big)\Big(\sum_{m=0}^n u_m\phi_m\Big)\phi_j dx, \tag{3.46}$$

Now if we recognize that both integrals vanish if $m \neq j-1, j-\frac{1}{2}, j, j+\frac{1}{2}, j+1$ (because of the choice of the basis functions), we can eliminate the summation over m to arrive at

$$\begin{aligned} &\textstyle\sum_{i=0,\frac{1}{2},1,...}^n \dot{u}_i \Bigg(\dot{u}_{j-1}\frac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_{j-1}\phi_i\phi_j dx + \dot{u}_{j-\frac{1}{2}}\frac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_{j-\frac{1}{2}}\phi_i\phi_j dx + \\ &\dot{u}_j\frac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_j\phi_i\phi_j dx + \dot{u}_{j+\frac{1}{2}}\frac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_{j+\frac{1}{2}}\phi_i\phi_j dx + \dot{u}_{j+1}\frac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_{j+1}\phi_i\phi_j dx \Bigg) \quad + \\ &\textstyle\sum_{i=0,\frac{1}{2},1,...}^n \ddot{u}_i \Bigg(u_{j-1}\frac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_{j-1}\phi_i\phi_j dx + u_{j-\frac{1}{2}}\frac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_{j-\frac{1}{2}}\phi_i\phi_j dx + \\ &u_j\frac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_j\phi_i\phi_j dx + u_{j+\frac{1}{2}}\frac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_{j+\frac{1}{2}}\phi_i\phi_j dx + u_{j+1}\frac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_{j+1}\phi_i\phi_j dx \Bigg), \end{aligned} \tag{3.47}$$

for the j^{th} equation, where again $j \in \{\frac{1}{2}, ..., n-\frac{1}{2}\}$. (3.47) is the term that should be added to the right-hand-side of (3.41).

As one can see above we know have 5 terms instead of the 3 terms we had using first order FEM. So now we have to identify five new matrices $\mathbf{C_1}$, $\mathbf{C_2}$, $\mathbf{C_3}$,$\mathbf{C_4}$ and $\mathbf{C_5}$ with elements

$$
\begin{aligned}
\mathbf{C_1}(2i,2j) &= \tfrac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_{j-1}\phi_i\phi_j dx, \\
\mathbf{C_2}(2i,2j) &= \tfrac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_{j-\frac{1}{2}}\phi_i\phi_j dx \\
\mathbf{C_3}(2i,2j) &= \tfrac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_{j}\phi_i\phi_j dx \\
\mathbf{C_4}(2i,2j) &= \tfrac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_{j+\frac{1}{2}}\phi_i\phi_j dx \\
\mathbf{C_5}(2i,2j) &= \tfrac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_{j+1}\phi_i\phi_j dx \quad \forall i,j \in \{\tfrac{1}{2},...,n-\tfrac{1}{2}\}
\end{aligned}
\tag{3.48}
$$

respectively, we can express (3.47) more compactly as

$$
\begin{aligned}
&\textstyle\sum_{i=0,\frac{1}{2},1,...}^{n} \dot{u}_i \Big(\dot{u}_{j-1}\mathbf{C_1}(2i,2j) + \dot{u}_{j-\frac{1}{2}}\mathbf{C_2}(2i,2j) + \dot{u}_j\mathbf{C_3}(2i,2j) + \dot{u}_{j+\frac{1}{2}}\mathbf{C_4}(2i,2j) + \dot{u}_{j+1}\mathbf{C_5}(2i,2j) \Big) \quad + \\
&\textstyle\sum_{i=0,\frac{1}{2},1,...}^{n} \ddot{u}_i \Big(u_{j-1}\mathbf{C_1}(2i,2j) + u_{j-\frac{1}{2}}\mathbf{C_2}(2i,2j) + u_j\mathbf{C_3}(2i,2j) + u_{j+\frac{1}{2}}\mathbf{C_4}(2i,2j) + u_{j+1}\mathbf{C_5}(2i,2j) \Big).
\end{aligned}
\tag{3.49}
$$

In each of the equations $j = \frac{1}{2}, ..., n-\frac{1}{2}$ where the term between brackets does not vanish for $i = 0$ and $i = n$ (this is only the case for $j = 1$, $j = \frac{1}{2}$, $j = n-1$ and $j = n-\frac{1}{2}$ because of the choice of basis functions), those contributions to the summation, which account for the Dirichlet boundary conditions, are separated and absorbed into the $\mathbf{g}$ vector as mentioned in Section 3.2.1.

Now if we look carefully at the first term of (3.23) we see that for a fixed j^{th} equation, all elements $\mathbf{C_1}(2i,2j) \quad \forall i \in \{\frac{1}{2},...,n-\frac{1}{2}\}$ are multiplied by $\dot{u}_{j-1}$. All elements $\mathbf{C_2}(2i,2j) \quad \forall i \in \{\frac{1}{2},...,n-\frac{1}{2}\}$ are multiplied by $\dot{u}_{j-\frac{1}{2}}$. All elements $\mathbf{C_3}(2i,2j) \quad \forall i \in \{\frac{1}{2},...,n-\frac{1}{2}\}$ are multiplied by $\dot{u}_j$. All elements $\mathbf{C_4}(2i,2j) \quad \forall i \in \{\frac{1}{2},...,n-\frac{1}{2}\}$ are multiplied by $\dot{u}_{j+\frac{1}{2}}$ and all elements $\mathbf{C_5}(2i,2j) \quad \forall i \in \{\frac{1}{2},...,n-\frac{1}{2}\}$ are multiplied by $\dot{u}_{j+1}$. The second term of (3.23) is similar except each row of a $\mathbf{C}$ matrix is multiplied by u_{j-1}, $u_{j-\frac{1}{2}}$, u_j, $u_{j+\frac{1}{2}}$ and u_{j+1} respectively. This applies for all $j = \frac{1}{2}, ..., n-12$. Therefore (3.49) can be written in full matrix form as

$$
\begin{aligned}
&\left(\begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & \dot{u}_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \dot{u}_{n-1\frac{1}{2}} \end{bmatrix} \mathbf{C_1} + \begin{bmatrix} \dot{u}_0 & 0 & \cdots & 0 \\ 0 & \dot{u}_{\frac{1}{2}} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \dot{u}_{n-1} \end{bmatrix} \mathbf{C_2} + \begin{bmatrix} \dot{u}_{\frac{1}{2}} & 0 & \cdots & 0 \\ 0 & \dot{u}_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \dot{u}_{n-\frac{1}{2}} \end{bmatrix} \mathbf{C_3} + \right. \\
&\left. \begin{bmatrix} \dot{u}_1 & 0 & \cdots & 0 \\ 0 & \dot{u}_{1\frac{1}{2}} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \dot{u}_n \end{bmatrix} \mathbf{C_4} + \begin{bmatrix} \dot{u}_{1\frac{1}{2}} & 0 & \cdots & 0 \\ 0 & \dot{u}_3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 0 \end{bmatrix} \mathbf{C_5} \right) \dot{\mathbf{u}} + \\
&\left(\begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & u_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & u_{n-1\frac{1}{2}} \end{bmatrix} \mathbf{C_1} + \begin{bmatrix} u_0 & 0 & \cdots & 0 \\ 0 & u_{\frac{1}{2}} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & u_{n-1} \end{bmatrix} \mathbf{C_2} + \begin{bmatrix} u_{\frac{1}{2}} & 0 & \cdots & 0 \\ 0 & u_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & u_{n-\frac{1}{2}} \end{bmatrix} \mathbf{C_3} + \right. \\
&\left. \begin{bmatrix} u_1 & 0 & \cdots & 0 \\ 0 & u_{1\frac{1}{2}} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & u_n \end{bmatrix} \mathbf{C_4} + \begin{bmatrix} u_{1\frac{1}{2}} & 0 & \cdots & 0 \\ 0 & u_{1\frac{1}{2}} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 0 \end{bmatrix} \mathbf{C_5} \right) \ddot{\mathbf{u}}.
\end{aligned}
\tag{3.50}
$$

Note that all explicitly given matrices in (3.50) are diagonal matrices, such that when matrix multiplication is used, the first row of $\mathbf{C_x}$ is multiplied by the first term on the diagonal and so on. Also

note that we know u_0 and u_n from the given Dirichlet boundary conditions and there derivatives can therefore be calculated analytically or numerically.

Now name the matrices in the first row of (3.50) from left to right $\mathbf{H_{-1}}(\dot{\mathbf{u}})$, $\mathbf{H_{-\frac{1}{2}}}(\dot{\mathbf{u}})$ $\mathbf{H_0}(\dot{\mathbf{u}})$, $\mathbf{H_{\frac{1}{2}}}(\dot{\mathbf{u}})$, $\mathbf{H_1}(\dot{\mathbf{u}})$ and in the second row $\mathbf{I_{-1}}(\mathbf{u})$,$\mathbf{I_{-\frac{1}{2}}}(\mathbf{u})$, $\mathbf{I_0}(\mathbf{u})$, $\mathbf{I_{\frac{1}{2}}}(\mathbf{u})$,$\mathbf{I_1}(\mathbf{u})$, where the bracket notation means that $\mathbf{H}$ is a function of $\dot{\mathbf{u}}$ and $\mathbf{I}$ is a function of $\mathbf{u}$.

Defining

$$\begin{aligned}
\mathbf{N_1}(\dot{\mathbf{u}}) &= \mathbf{H_{-1}}(\dot{\mathbf{u}})\mathbf{C_1} + \mathbf{H_{-\frac{1}{2}}}(\dot{\mathbf{u}})\mathbf{C_2} + \mathbf{H_0}(\dot{\mathbf{u}})\mathbf{C_3} + \mathbf{H_{\frac{1}{2}}}(\dot{\mathbf{u}})\mathbf{C_4} + \mathbf{H_1}(\dot{\mathbf{u}})\mathbf{C_5} \quad , \\
\mathbf{N_2}(\mathbf{u}) &= \mathbf{I_{-1}}(\mathbf{u})\mathbf{C_1} + \mathbf{I_{-\frac{1}{2}}}(\mathbf{u})\mathbf{C_2} + \mathbf{I_0}(\mathbf{u})\mathbf{C_3} + \mathbf{I_{\frac{1}{2}}}(\mathbf{u})\mathbf{C_4} + \mathbf{I_1}(\mathbf{u})\mathbf{C_5}
\end{aligned} \tag{3.51}$$

we can finally state the semi-discrete matrix form of the finite element formulation of the one-dimensional Westervelt equation:

$$\mathbf{Ku} + \mathbf{M}\ddot{\mathbf{u}} = \mathbf{N_1}(\dot{\mathbf{u}})\dot{\mathbf{u}} + \mathbf{N_2}(\mathbf{u})\ddot{\mathbf{u}} + \mathbf{g}, \tag{3.52}$$

where $\mathbf{g}$ now also includes the Dirichlet boundary condition contributions of the nonlinear term:

$$\mathbf{g}(2j) = \begin{cases}
-u_0(t)\int_0^L \phi_0'\phi_{\frac{1}{2}}'\,dx - \frac{1}{c_0^2}\ddot{u}_0(t)\int_0^L \phi_0\phi_{\frac{1}{2}}\,dx \\
+\dot{u}_0\left(\dot{u}_0\frac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_{\frac{1}{2}}\phi_0\phi_0 dx + \dot{u}_{\frac{1}{2}}\frac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_{\frac{1}{2}}\phi_0\phi_{\frac{1}{2}}dx\right) \\
+\ddot{u}_0\left(u_0\frac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_0\phi_0\phi_{\frac{1}{2}}dx + u_{\frac{1}{2}}\frac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_{\frac{1}{2}}\phi_0\phi_{\frac{1}{2}}dx\right), & \text{if } j = \frac{1}{2} \\
-u_0(t)\int_0^L \phi_0'\phi_1'dx - \frac{1}{c_0^2}\ddot{u}_0(t)\int_0^L \phi_0\phi_1 dx \\
+\dot{u}_0\left(\dot{u}_0\frac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_0\phi_0\phi_1 dx + \dot{u}_{\frac{1}{2}}\frac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_{\frac{1}{2}}\phi_0\phi_1 dx + \dot{u}_1\frac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_1\phi_0\phi_1 dx\right) \\
+\ddot{u}_0\left(u_0\frac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_0\phi_0\phi_1 dx + u_{\frac{1}{2}}\frac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_{\frac{1}{2}}\phi_0\phi_1 dx + u_1\frac{2\beta}{\rho_0 c_0^4}\int_0^L \phi_1\phi_0\phi_1 dx\right), & \text{if } j = 1 \\
0, & else
\end{cases} \tag{3.53}$$

The discretization of time still is the same as with the first order FEM.

3.3.3 Discretization of the attenuation term

As we did before, we want to bring that term of attenuation into our numeric scheme as well. To make it easier we will forget about the nonlinear term for now.
For the derivation one should look in the same section for first order FEM because the second order FEM does not change the derivation, only the $\mathbf{K}$, $\mathbf{M}$ and $\mathbf{g}$ are different. So we get the same semi-discretized system

$$\mathbf{Ku} + \mathbf{M}\Big(\frac{1}{c_0^2}\ddot{\mathbf{u}} - \frac{\delta}{c_0^4}\dddot{\mathbf{u}}\Big) = \mathbf{g}, \tag{3.54}$$

where the elements of the mass matrix $\mathbf{M}$ and the stiffness matrix $\mathbf{K}$ are

$$\begin{aligned}
\mathbf{K}(2i,2j) &= \int_0^L \phi_i'\phi_j'dx \quad \text{and} \\
\mathbf{M}(2i,2j) &= \int_0^L \phi_i\phi_j dx \quad \forall i,j \in \{\tfrac{1}{2},1,...,n-1,n-\tfrac{1}{2}\}
\end{aligned} \tag{3.55}$$

respectively.
Note again that $\mathbf{M}(2i,2j)$ and $\mathbf{K}(2i,2j)$ equal zero if $|j-i| > 1$ and thus $\mathbf{M}$ and $\mathbf{K}$ are very sparse $(2n-1)\times(2n-1)$ matrices. The elements of $\mathbf{g}$ are given by

$$\mathbf{g}(2j) = \begin{cases} -u_0(t)\int_0^L \phi_0'\phi_{\frac{1}{2}}'\,dx - \frac{1}{c_0^2}\ddot{u}_0(t)\int_0^L \phi_0\phi_{\frac{1}{2}}\,dx + \frac{\delta}{c_0^4}\ddot{u}_0(t)\int_0^L \phi_0\phi_{\frac{1}{2}}\,dx, & \text{if } j = \frac{1}{2} \\ -u_0(t)\int_0^L \phi_0'\phi_1'\,dx - \frac{1}{c_0^2}\ddot{u}_0(t)\int_0^L \phi_0\phi_1\,dx + \frac{\delta}{c_0^4}\ddot{u}_0(t)\int_0^L \phi_0\phi_1\,dx, & \text{if } j = 1 \\ 0, & else \end{cases} \tag{3.56}$$

The right-hand-side vector $\mathbf{g}$ is a column vector of length $2n-1$ and only the first two values are not equal to zero because of the time-dependent Dirichlet boundary condition.
To get the complete scheme of the Westervelt equation we have to add the derivations from the previous section to what we did in this section. So the complete scheme of the Westervelt equation is

$$\mathbf{K}\mathbf{u} + \mathbf{M}\Big(\frac{1}{c_0^2}\ddot{\mathbf{u}} - \frac{\delta}{c_0^4}\dddot{\mathbf{u}}\Big) = \mathbf{N_1}(\dot{\mathbf{u}})\dot{\mathbf{u}} + \mathbf{N_2}(\mathbf{u})\ddot{\mathbf{u}} + \mathbf{g}, \tag{3.57}$$

with $\mathbf{M}$ and $\mathbf{K}$ from (3.55), $\mathbf{N1}$ and $\mathbf{N1}$ from (3.51) and $\mathbf{g}$ is as given above with the nonlinear terms from (3.53).
The discretization in time is the same for the second order as for the first order FEM.

4 Results and Discussion

In this section we show and discuss the results obtained from the implementations that are discussed in Chapter 3. We have made implementations for solving the linear wave equation, the Westervelt equation with the nonlinear term and the Westervelt equation complete with nonlinear term and attenuation term. These implementations are made for both the first order finite element method and the second order finite element method. All implementations were made in Matlab. First we have plotted the solutions to the equations at the shock wave formation distance. These figures show us the effect of the nonlinear term and the attenuation term on the solutions. We will look at the form of the wave as it changes because of nonlinearity and attenuation. To make the difference between first and second order FEM visible we have plotted the difference between the two at various times. We have also plotted the solutions at two times the shock wave formation distance, which shows us the efficiency of finite element method implemented in Matlab. To get a better insight into the effect of nonlinearity and attenuation we look at the frequency spectrum of the Westervelt equation with and without attenuation term. In the frequency spectrum energy is shifted to higher harmonics, because of nonlinearity. So we expect to see a lot of peaks in the frequency spectrum of the Westervelt equation. Finally, we choose a domain that has a sudden change in small signal sound speed in the middle of the domain. This causes there to be reflection and transmission waves. The amplitude of these waves are calculated and the waves are plotted for different changes in speed.

4.1 Source function and Parameters

So when we started in Chapter 3 we assumed that we had a source function working on the Westervelt equation. The source function we use for our results is given by

$$p(0,t) = P_0 \sin[2\pi f_0(t - T_d)] \exp\Big[-\Big(\frac{t - T_d}{T_w/2}\Big)^2\Big]\Big], \tag{4.1}$$

where P_0 is the pressure amplitude, f_0 is the frequency of the source, T_d is a time delay, T_w is a pulse width and T_{end} is the end of the pulse. We set, as is customary in ultrasound imaging pulses, $T_d = \frac{6}{f_0}$, $T_w = \frac{3}{f_0}$ and $T_{end} = 2T_d$ [10]. The sinus component is the single frequency source function. The exponent makes sure that only a small part of the sinus is taken by putting a Gaussian envelope over the function. This means that the shock wave formation distance that was calculated in Section 2.2 is still valid for this source function. In Figure 4.1 the source function is shown.

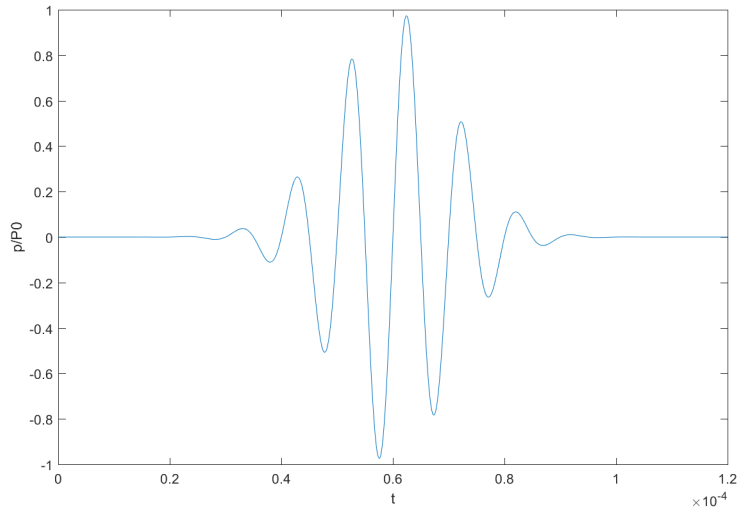

Figure 4.1: *A single frequency source function with a Gaussian envelope as described by* (4.1)*. The delay and width parameters are chosen so that a pulse of about six periods is generated. In this case* $T_d = \frac{6}{f_0}$ *and* $T_w = \frac{3}{f_0}$.

.

In the simulations that are made from the Matlab implementations the following parameters are used. These parameters are found in the thesis of Huijssen[10]. For the surroundings we have chosen the physical properties of water. The only thing that we have chosen differently is the value of β. For water the value of β is typically 3.5, but we have chosen to set $\beta = 10$. Because the shock wave distance depends inversely on β, the computation time is shortened.
Some of the parameters depend on other parameters. For these parameters we will discuss why we chose to give them these values if they have not been discussed before. The domain length of the wave is $L = x_{sh} + T_{end} \cdot c_0$, which assures us that the wave at least passes the shock wave distance. This domain is divided into an equidistant mesh. We cannot take infinitely small element size, because the time step needs to be decreased as well for the propagation to be solved correctly. This will lead to an infinite computation time. So from [14] we know that for quartic elements a well chosen mesh size is $\frac{c_0}{18f_0}$. In this thesis we use first and second order elements. For quartic elements there are 3 auxiliary points between every two element points. So when keeping equal refinement the mesh size for first order elements has to be 4 times finer than the mesh size of quartic element so $dx = \frac{c_0}{72f_0}$. To compare the first order elements to the second order elements dx is the same for both methods. This means that for second order element there is an auxiliary point between every two element points. For the time step it is important that $\frac{c_0 dt}{dx} < 1$, because of the CourantFriedrichsLewy (CFL) condition.[15] This is a necessary condition for convergence of the solution when solving partial differential equations numerically with difference schemes. This means that $dt < \frac{dx}{c_0}$. We choose $dt = \frac{dx}{4c_0}$ which agrees with the condition and still gives us an acceptable computation time.

Table 1: *A summary of the parameters used for simulations of the one-dimensional Westervelt equation in all Matlab simulations presented in this section. These parameters are used, unless specified differently.*

ρ_0	1000	$\frac{\text{kg}}{\text{m}^3}$	Material ambient density
c_0	1500	$\frac{\text{m}}{\text{s}}$	Small signal sound speed
β	10		Coefficient of nonlinearity
P_0	10^6	Pa	Maximum pressure amplitude
f_0	10^5	Hz	Single source frequency
T_d	$\frac{6}{f_0}$	s	Source envelope delay
T_w	$\frac{3}{f_0}$	s	Source envelope width
T_{end}	$2 \cdot T_d = \frac{12}{f_0}$	s	Source end time
x_{sh}	$\frac{\rho_0 c_0^3}{\beta P_0 2\pi f_0}$	m	Shock formation distance
L	$x_{sh} + T_{end} \cdot c_0$	m	Domain length
dx	$\frac{c_0}{72f_0}$	m	Mesh element size
dt	$\frac{dx}{4c_0}$	s	Maximum time step
ϵ	$10^{-9} \cdot P_0$	Pa	Absolute tolerance for successive approximation
δ	0.007	m^2/s	Attenuation coefficient.
	1&2		Element order
	equidistant		Mesh type

4.2 Solutions to multiple wave equations using FEM

Figure 4.2 shows a plot of the result obtained with the first and second order FEM for the linear wave equation. In Figure 4.3 the difference between the result of the first and the second order element method is shown. The solution is plotted at the shock wave distance. In the figure the analytic solution is plotted as well. This is the well known solution to the linear wave equation.

$$p(x,t) = f(t - \frac{x}{c_0}) = P_0 \sin[2\pi f_0(t - \frac{x}{c_0} - T_d)] \exp\Big[- \Big(\frac{t - \frac{x}{c_0} - T_d}{T_w/2} \Big)^2 \Big] \Big] \tag{4.2}$$

The solution of the Westervelt equation without term of attenuation is plotted in Figure 4.4. In Figure 4.5 the difference between the result of the first and the second order element method is shown. In this implementation we used the method of successive approximation to solve the nonlinear system of equations for each time step. Note that we choose to set the absolute tolerance $\epsilon = 10^{-9} \cdot P_0 = 10^{-3}$ Pa. We argue that the number of iterations this would take during each time step was going to be small, because the time step taken is very small and the nonlinear term is small compared to the linear terms in the Westervelt equation. To verify this, we plotted the number of iterations needed to solve the nonlinear system of equations for each time step versus the time step. The result is shown in Figure 4.6. From this figure we can see that the number of iterations needed is at most 4. This justifies the implementation of successive approximation.
In Figure 4.7 we have plotted the results of the complete Westervelt equation for the first and second order FEM. In Figure 4.8 the difference between the result of the first and the second order element method is shown.

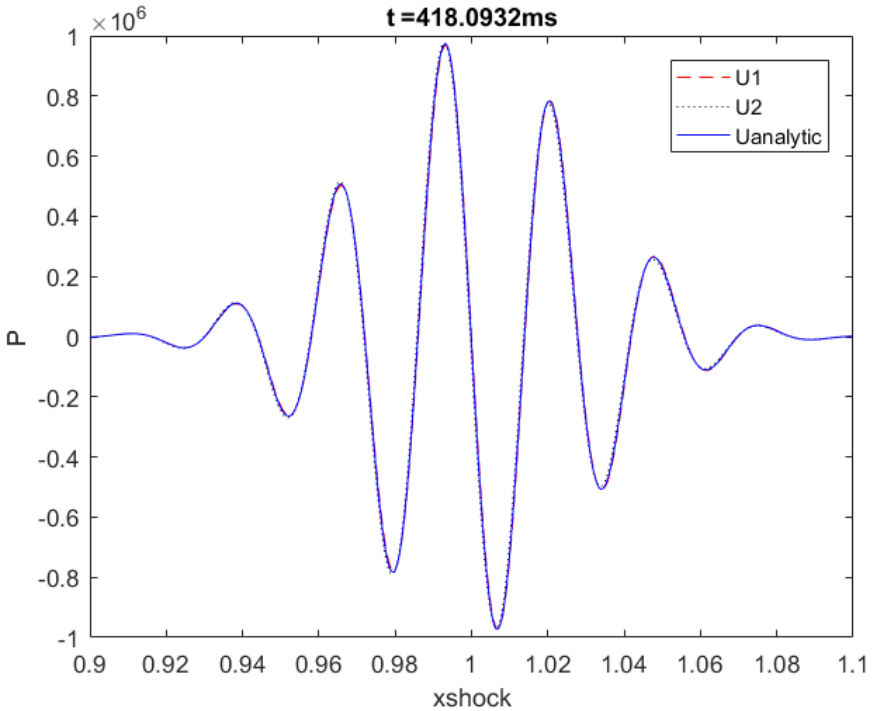

Figure 4.2: *Solution of the linear wave equation calculated for the first and second order FEM The analytic solution to the linear wave equation is also added to the figure. Parameters were set as in Table 1.*

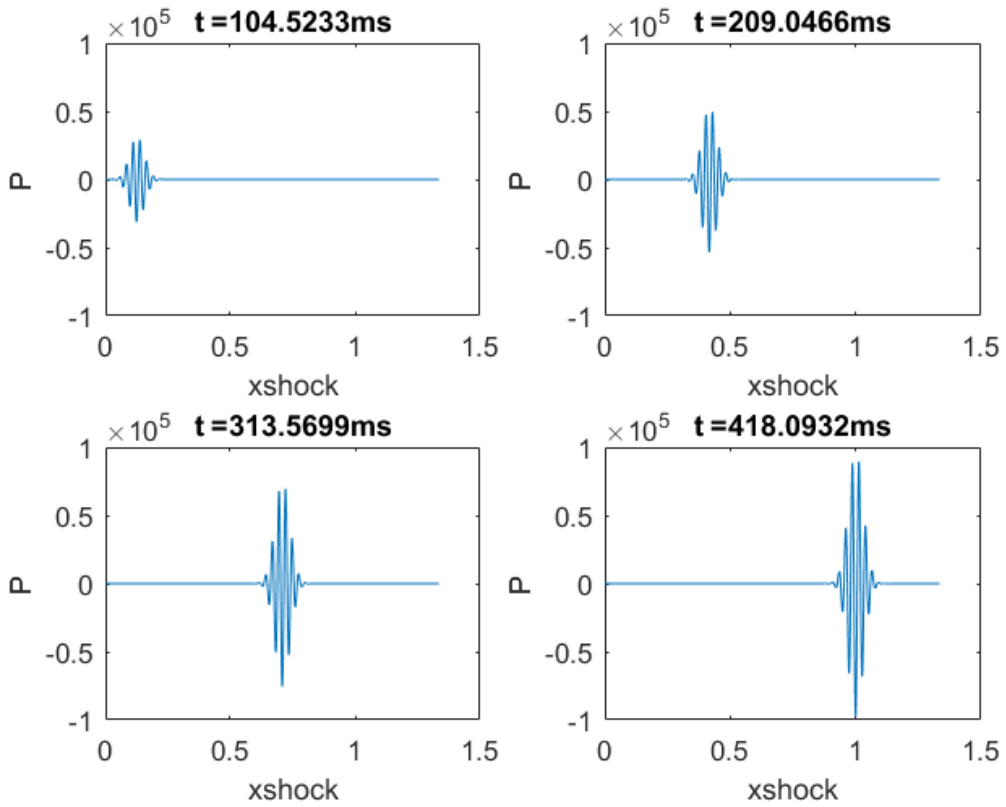

Figure 4.3: *The difference of the solution between the linear wave equation between first and second order FEM*

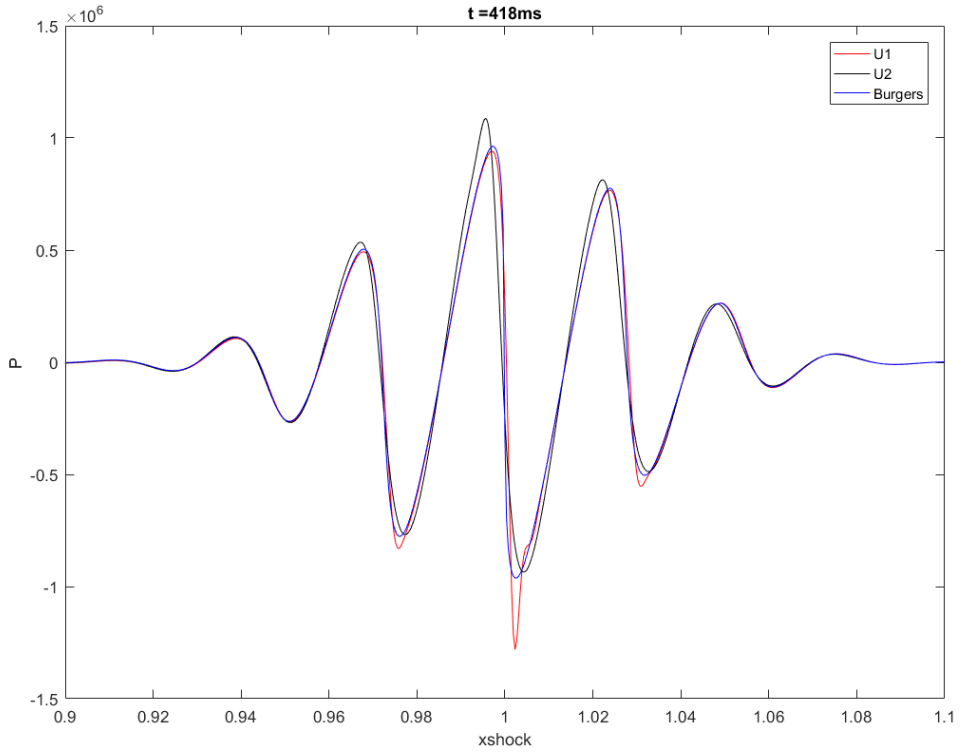

Figure 4.4: *Solution of the nonlinear wave equation without attenuation term calculated with our own implementation for the first and second order FEM. The implicit Burgers solution is also added to the figure. Parameters were set as in Table 1.*

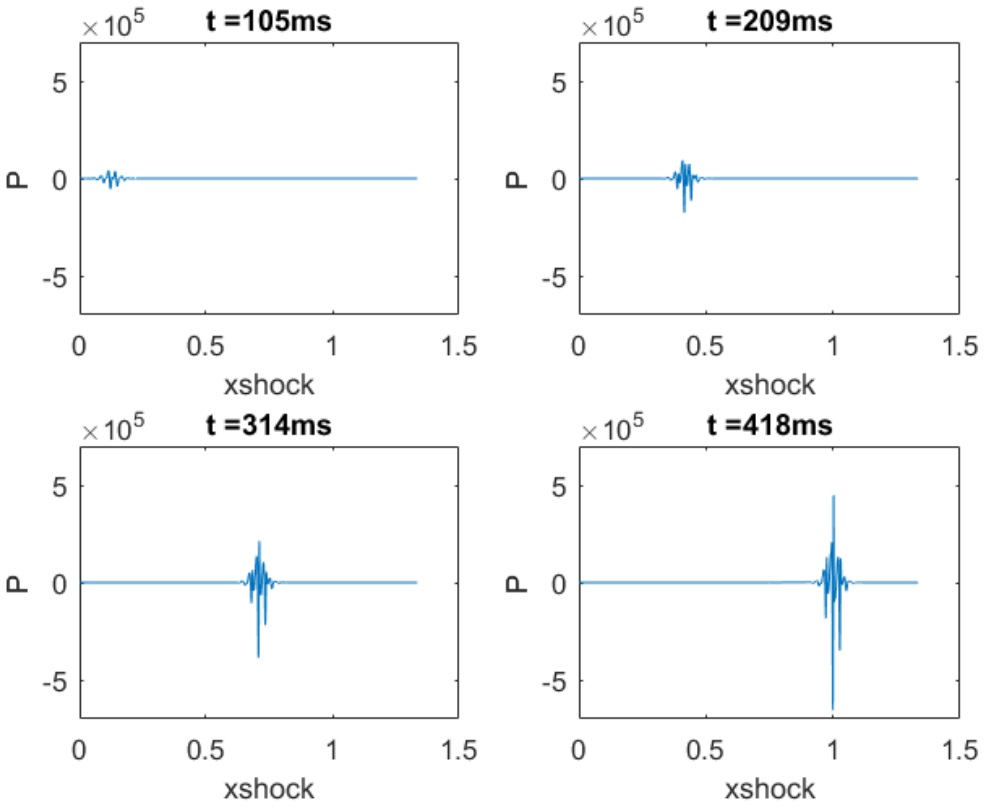

Figure 4.5: *The difference of the solutions of the nonlinear wave equation without attenuation term between first and second order FEM at different moments in time*

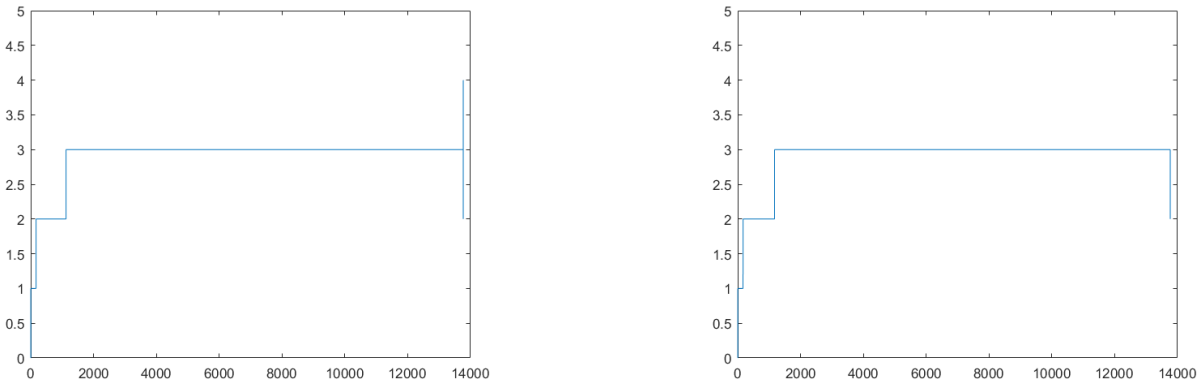

Figure 4.6: *Number of iterations that the successive approximation algorithm took to solve the nonlinear wave equation up to a desired tolerance per time step versus the number of the time steps. The figure on the left plots the iterations of the first order FEM, the figure on the right plots the iterations of the second order FEM. No more than four iterations were needed.*

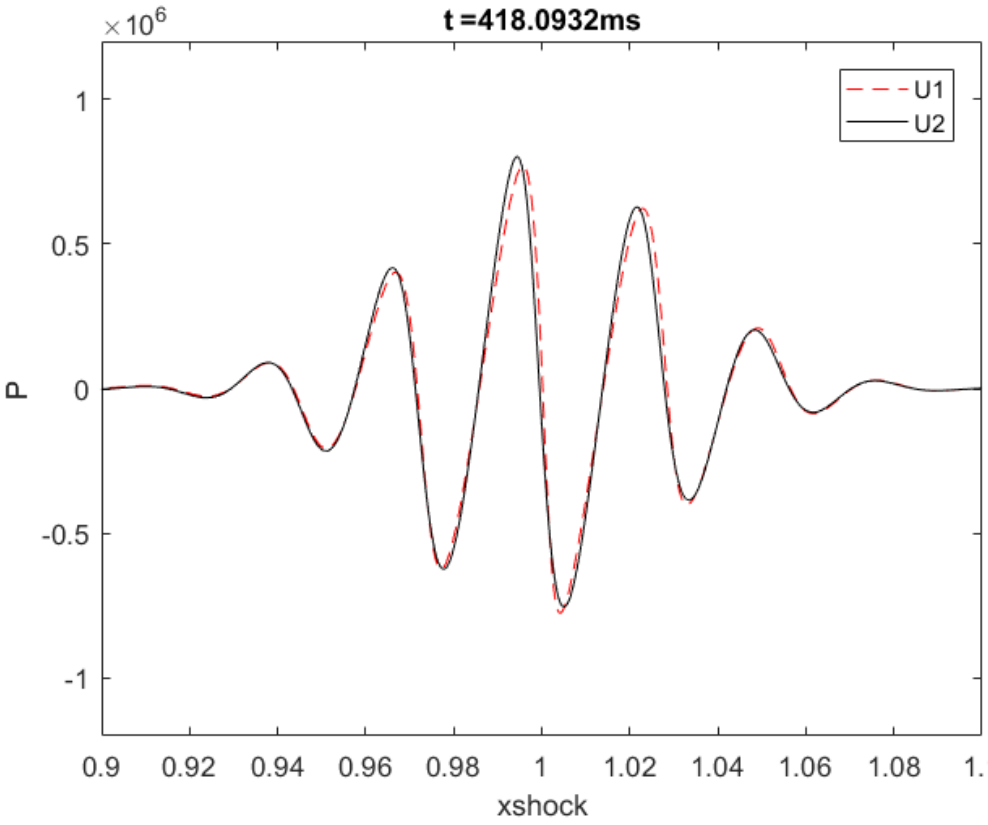

Figure 4.7: *Solution of the complete Westervelt equation calculated with our own implementation for the first and second order FEM. Parameters were set as in Table 1.*

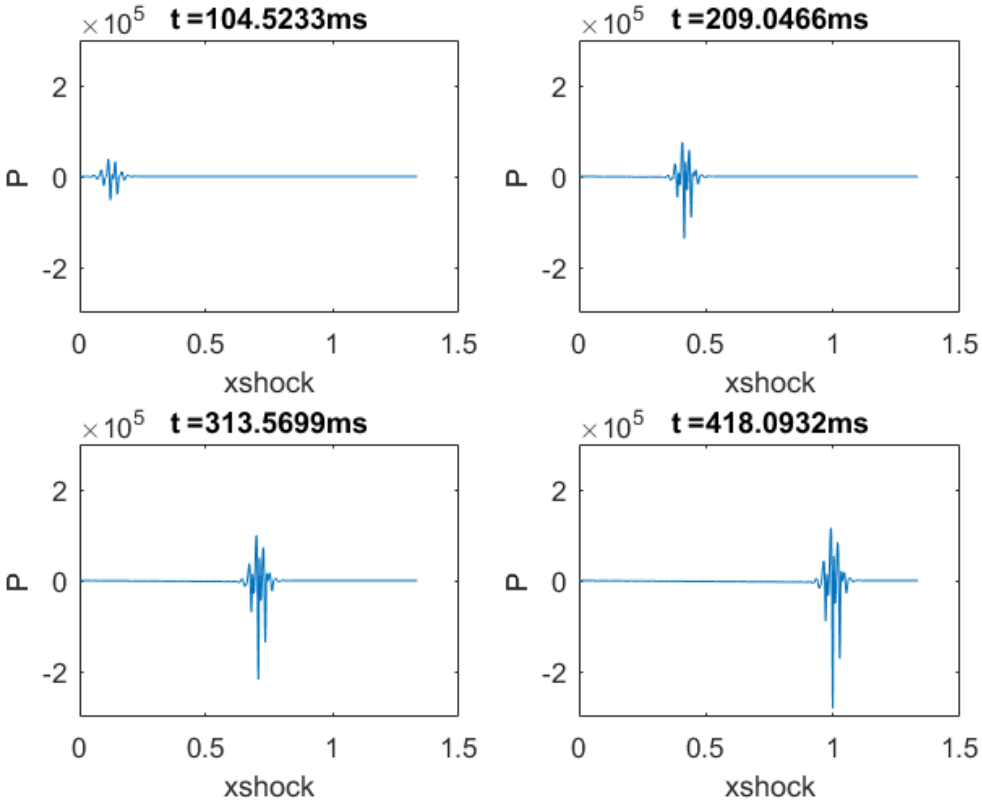

Figure 4.8: *The difference in the solutions of the complete Westervelt equation between first and second order FEM at different moments in time*

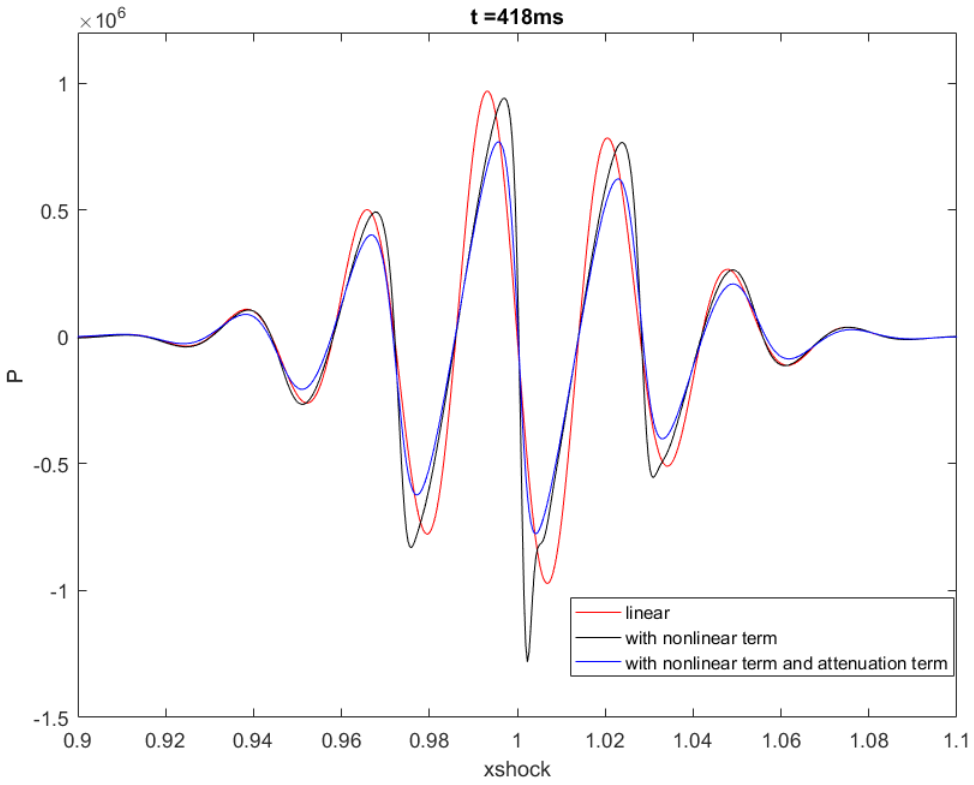

Figure 4.9: *The solutions of the linear wave equation, the Westervelt equation without attenuation term and the complete Westervelt equation plotted using the first order fem*

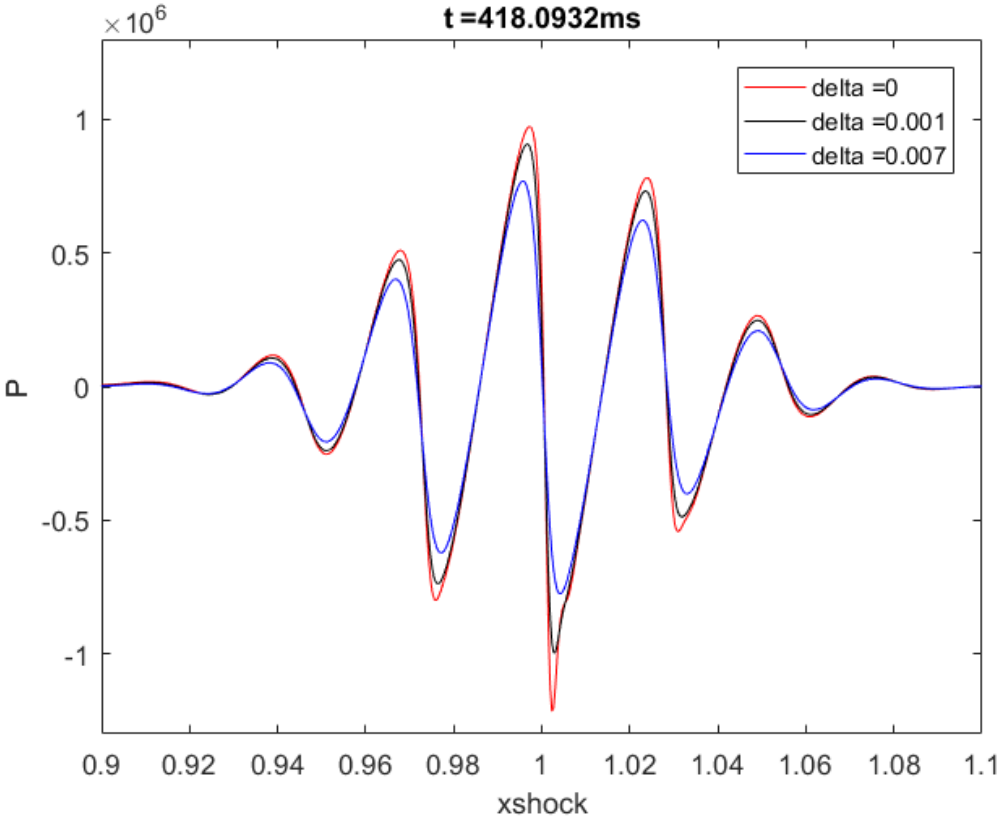

Figure 4.10: *The solutions of the Westervelt equation with attenuation term with different coefficients of attenuation using first order FEM*

In Figure 4.2 one cannot see the difference between the first order FEM, second order FEM and the analytical solution to the linear wave equation. In Figure 4.3 the difference between the solution for second order FEM and first order FEM is plotted. You can clearly see that the difference is a linear wave in itself, which is good because there are no nonlinear terms working on the solution. According to the scale of the y-axis, the greatest difference between the two solutions is about 10 percent of the solution. This is large error, but can easily be explained. Because of the different order of FEM of the solutions and the not infinitely small mesh size, the solution of second order FEM has a small phase shift compared to the solution of the first order FEM. The slope of the solution, however, is really massive in the points of the x-axis. So even though the phase shift is really small, the difference between the solution is enormous in the points on the x-axis, so the error seems to be a really large percentage of the solution.

In Figure 4.4 one can clearly see the difference between the first order FEM, second order FEM and the Burgers solution to the Westervelt wave equation without attenuation term. In Figure 4.5 the difference between the solution for second order FEM and first order FEM is plotted. It is evident that a nonlinear term works on the wave because the difference is not a linear wave anymore. According to the scale seen in the plot, the greatest difference between the two solutions is now a lot more that 10 percent of the solution. This is visible in Figure 4.4. Especially at the bottom, the first order FEM differentiates a lot from the second order FEM and the Burgers solution. At the top the solution of the second order FEM differentiates a bit from the Burgers solution. This can be explained through the orders of FEM. The second order FEM uses second order basis functions, which results in a better solution on a small grid part, because it uses more information of the neighboring points. But using second order FEM is still a numerical approximation, so the solution is better than the first order FEM but not the exact solution of the Westervelt equation. While we now use the Burgers solution as a solution of the Westervelt equation to compare the solutions of FEM to, it is in itself an approximation, because of interpolation done in Matlab and neglecting higher-order terms of the Taylor expansion from Section 2.1. A part of the difference between the solutions of the second and first order FEM still is an result of the phase shift between the two solutions.
Also a distinct feature in Figure 4.4 is the shallower slope of the numerical solution compared to the solution of the linear wave equation when going from a minimum to a maximum in the wave, and a steeper slope when going from a maximum to a minimum. The explanation for this phenomenon is given in Section 2.2. For a better comparison one should look at Figure 4.9, where the solutions of the linear wave equation, the Westervelt equation without attenuation term and the complete Westervelt equation are plotted with the same parameters for first order FEM at the same time.

In Figure 4.7 the difference between the first order FEM and the second order FEM is larger than in Figure 4.2, but smaller than in Figure 4.4. This can also be seen in Figure 4.8. Here the difference is still a nonlinear wave, but it is a lot smaller than the difference wave plotted in Figure 4.5. The difference can again be explained by the difference in order of the FEM and the phase shift of the different solution. Because of the damping from the attenuation term, the amplitude of the wave is not only smaller at the shock wave distance but also the effect of the nonlinear term on the slope is less. The effect is better visible in Figure 4.9. This figure shows that the solution of the Westervelt equation with attenuation and nonlinear term has a smaller amplitude than both the linear equation and the Westervelt equation without attenuation term. Also while the slope going from a maximum to a minimum is steeper than the slope of the solution to the linear wave equation at that point, it is a shallower slope than the slope of the solution to the Westervelt equation without attenuation term at that point. The explanation for this effect caused by the attenuation term is given in Section 2.2.

It is clear that the attenuation term has a substantial influence on the solution, both on its amplitude and its shape. In Figure 4.10 the effect of choosing a different coefficient of attenuation is shown. When a higher term of attenuation is chosen, the amplitude of the wave is smaller and the slope of the wave is less steep.

4.3 Solutions at two times the shock wave distance

Because we want to see some more effect of the nonlinearity and attenuation on the source wave we have plotted the wave at two times the shock wave distance. The figures are shown below. We have done this for the linear wave equation, the Westervelt equation without attenuation term and the Westervelt equation with attenuation term. For the linear wave equation the difference between first order FEM and second order FEM are bit more clear than in Figure 4.2, but overall it is still a small error. Now Figure 4.12 shows a massive change. While the maxima and minima still align, except for a small phase shift, they are at a completely different heights and the waves shows a lot of nonlinearity at distinct places. This can be explained by the nonlinear term. As explained in Section 2.2 it causes the high pressure points to travel faster than the low pressure points. At the shock wave formation distance the high pressure point is directly above the low pressure point. So at this point the derivative $\frac{dp}{dx}$ goes to $-\infty$. Because we are using a numerical method to solve the Westervelt equation this ruins the computation of the solution. So as one can see the solution using FEM is not usable after the shock wave distance. In Figure 4.13 the complete Westervelt equation is shown at two times the shock wave distance. As mentioned in Section 2.2 the attenuation damps the effect of the nonlinear term and makes sure the high pressure point will not overtake the low pressure point. Because of this the derivative $\frac{dp}{dx}$ never goes to $-\infty$ so the computation using the numerical method still works. The solutions in this plot are a lot more aligned than the solutions in Figure 4.12. There is still a difference between the first order FEM and second order FEM but this is caused by the difference in order and a small phase shift and not by enormous computational mistakes. Also when looking at the y-axis it is clear that the wave has been greatly damped along the way.

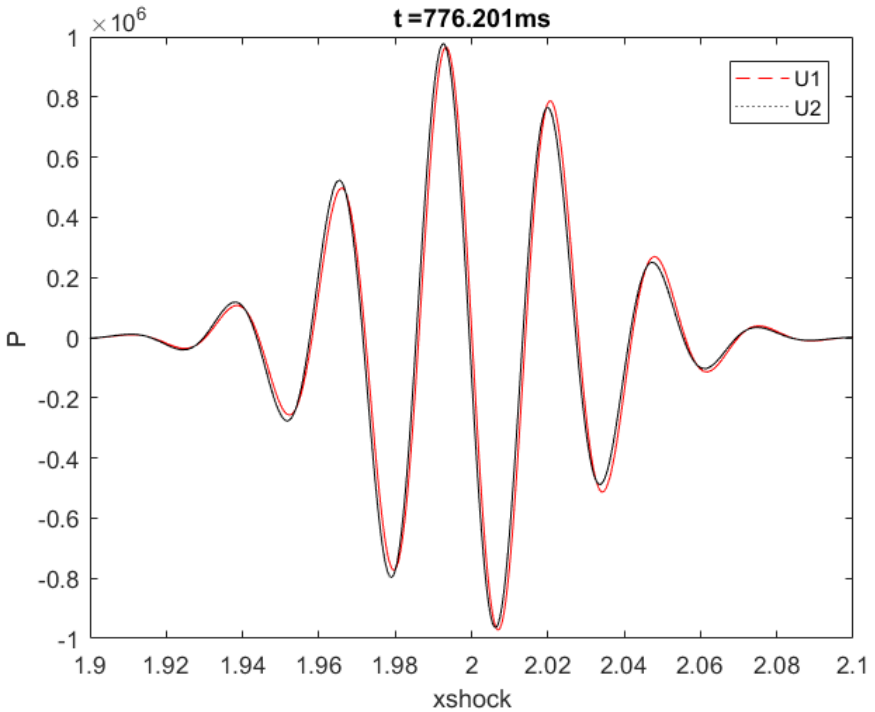

Figure 4.11: *The solution of the linear wave equation at two times the shock wave distance. Both the solutions obtained by using first order FEM and second order FEM are plotted.*

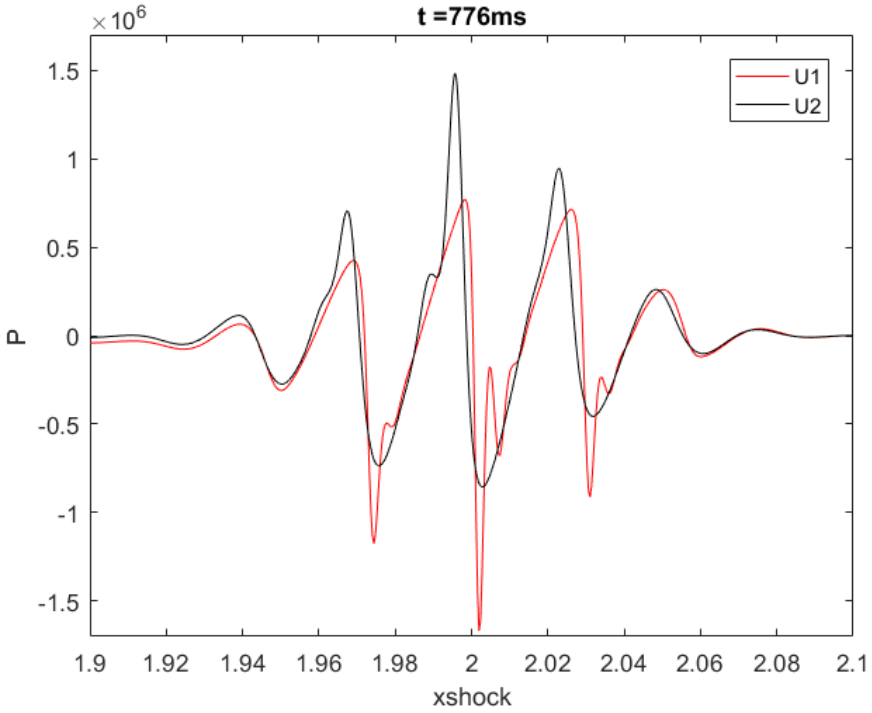

Figure 4.12: *The solution of the Westervelt equation without attenuation term at two times the shock wave distance. Both the solutions obtained by using first order FEM and second order FEM are plotted.*

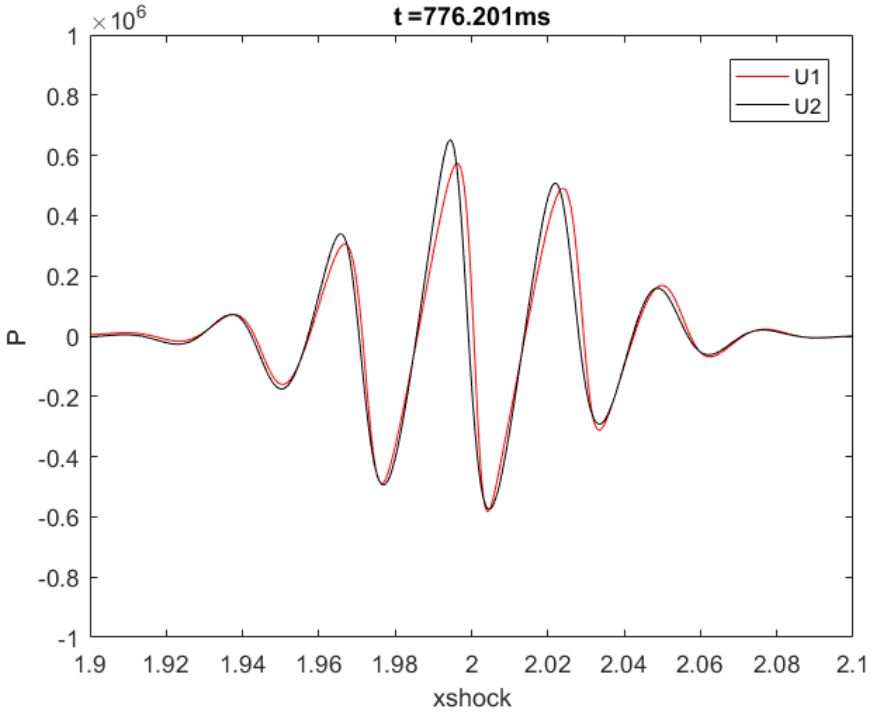

Figure 4.13: *The solution of the Westervelt equation with attenuation term at two times the shock wave distance. Both the solutions obtained by using first order FEM and second order FEM are plotted.*

4.4 Fast Fourier Transform

To get a better insight into our solution we have plotted the absolute value of the fast Fourier transform of the solutions. To find this value we used the Matlab command fft. We have plotted the absolute value of the fast Fourier transform at multiple times for all the different solutions we have shown earlier in this report. Also to see the effect of the pressure amplitude of the source wave on the solution we have plotted the Fourier spectrum for the Westervelt equation with and without the attenuation term with $P_0 = 2 \cdot 10^6\ Pa$. As one can see in Figure 4.14 for the linear wave the absolute value of the Fourier transform has only one peak. This agrees with the frequency spectrum of a wave with linear propagation that is given in the thesis of Huijssen[10]. You can only see one peak, because there is no energy transferred to higher harmonics as there is no nonlinearity.
Nonlinear behavior in the time domain causes an energy shift towards higher-order harmonics in the frequency domain. So for the frequency spectrum of the solutions to the Westervelt equation with and without attenuation term one would expect there to be more peaks. As we saw before in Figure 4.9 the attenuation term damps the effect of the non-linear term. So we would expect a lot of peaks in the spectrum of the solution of the Westervelt equation without attenuation term and damped and less peaks in the spectrum of the complete Westervelt equation. And if we look at Figures 4.15 and 4.17 the theory is right. In the sixth frame of Figure 4.15 six peaks are clearly visible. But in the sixth frame of Figure 4.17 only four peaks are clearly visible. The first peak is also a lot smaller than the first peak in the spectrum of the Westervelt equation without attenuation term, because of damping. Another thing that stands out in both figures is the fact that the higher-order peaks are smaller for the second order FEM solution than the first order FEM. This means that the solution is more nonlinear for the first order FEM than the second order FEM.
In the figures that are discussed above all the parameters have the values given in Table 1. In Figures 4.16 and 4.18 we have changed the pressure amplitude from $1 \cdot 10^6 Pa$ to $2 \cdot 10^6 Pa$. While all the aspects described above remain the same, there are clearly more peaks visible in both plots. The first peak of the spectrum is also a lot larger in both figures than the first peaks of Figures 4.15 and 4.17. So this means that if we would plot the wave at the shock wave distance for these solutions we would get a wave that is more nonlinear. The shock formation distance for

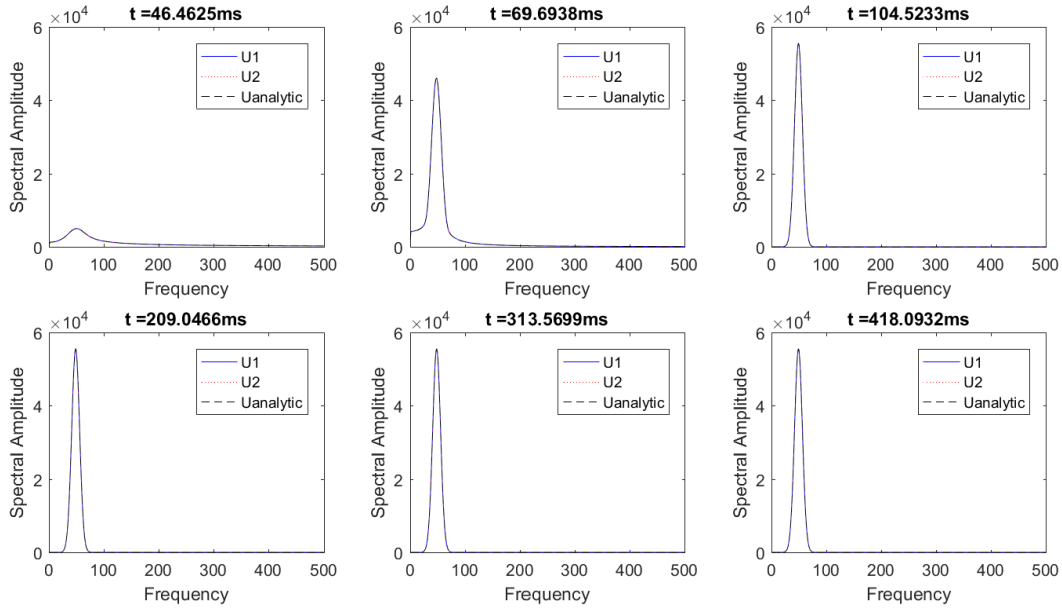

Figure 4.14: *The absolute values of the Fourier Transform of the linear wave equation plotted at different times*

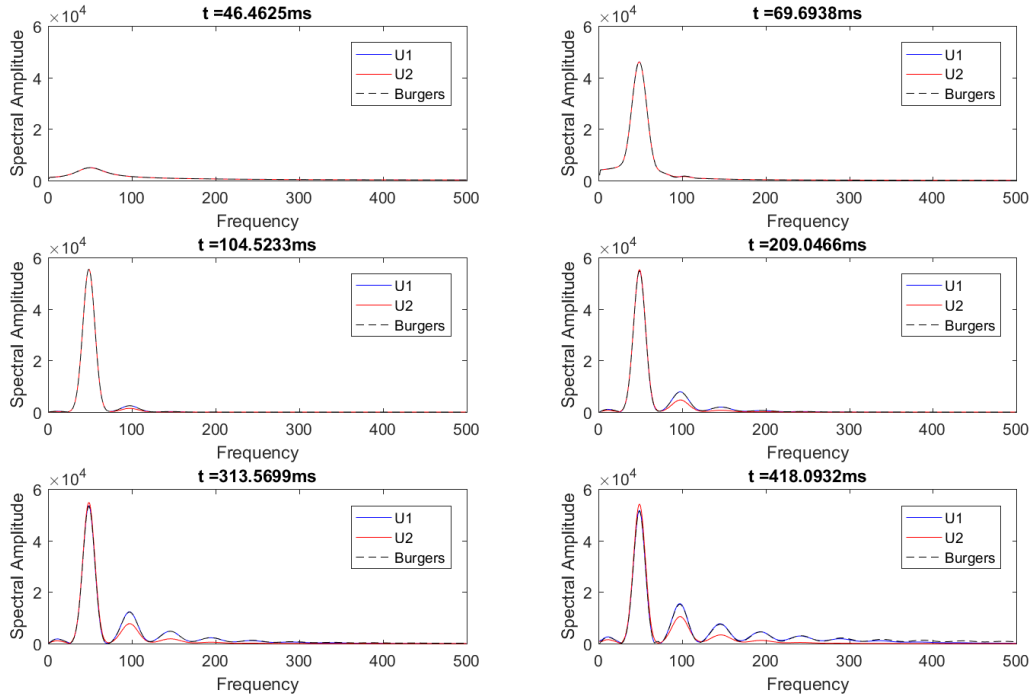

Figure 4.15: *The absolute values of the Fourier Transform of the Westervelt equation without attenuation term plotted at different times. The solutions for first and second order FEM are plotted together with the Burgers solution. Here* $P0 = 10^6 Pa$.

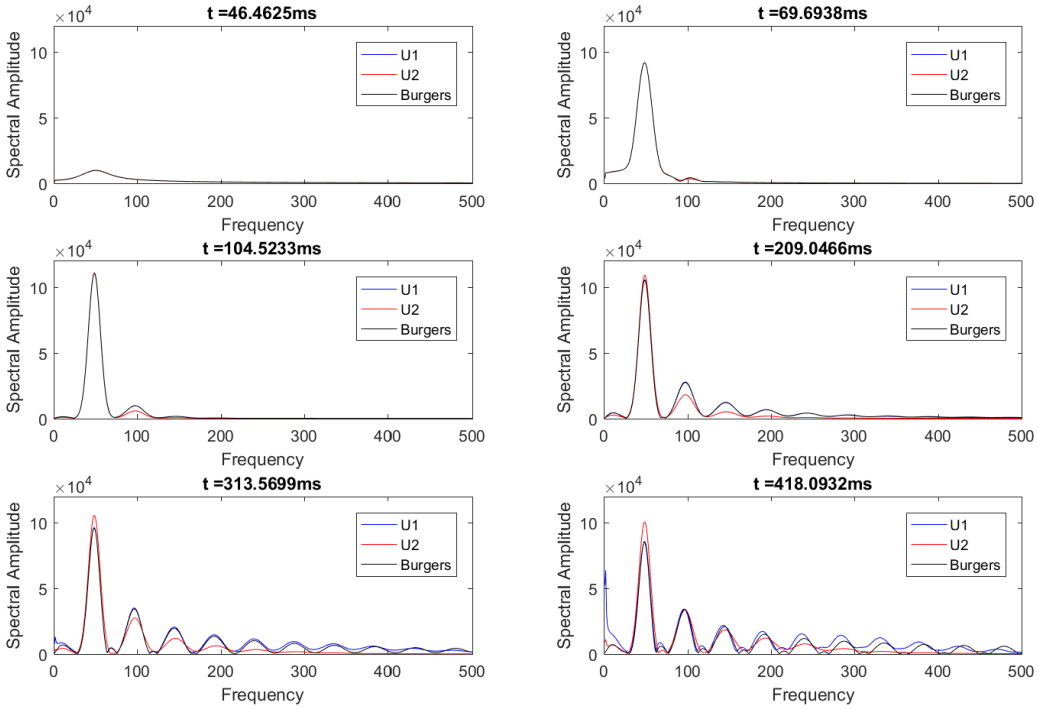

Figure 4.16: *The absolute values of the Fourier Transform of the Westervelt equation without attenuation term plotted at different times. The solutions for first and second order FEM are plotted together with the Burgers solution. Here* $P0 = 2 \cdot 10^6 Pa$.

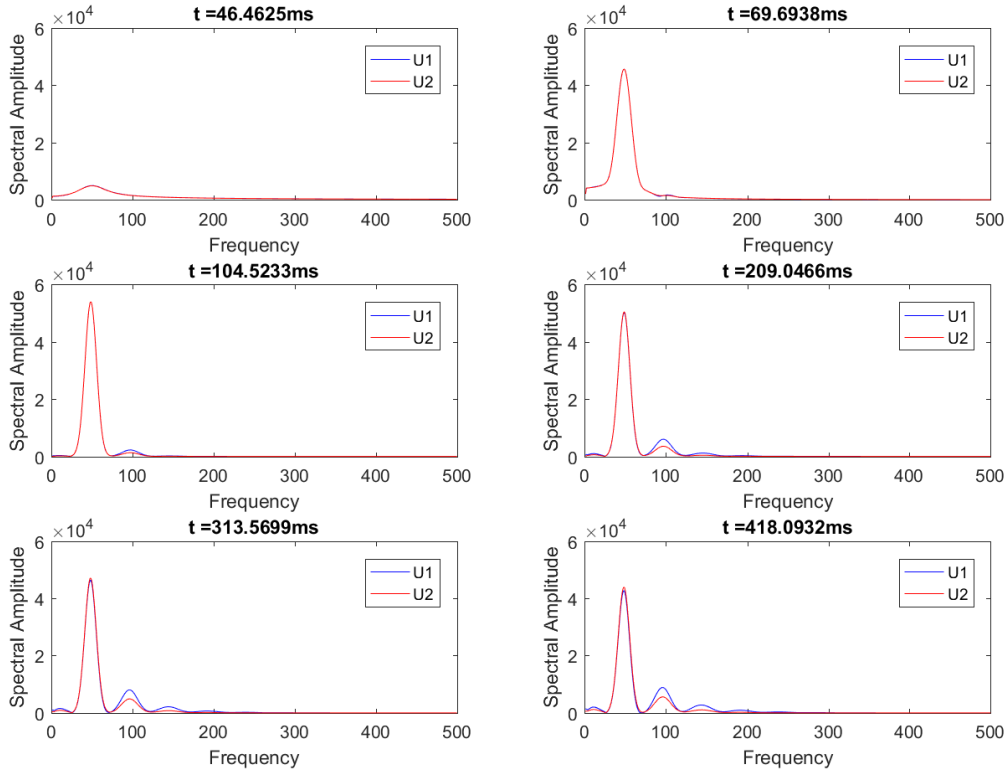

Figure 4.17: *The absolute values of the Fourier Transform of the Westervelt equation with attenuation term plotted at different times. The solutions for first and second order FEM are plotted together. Here* $P0 = 10^6 Pa$.

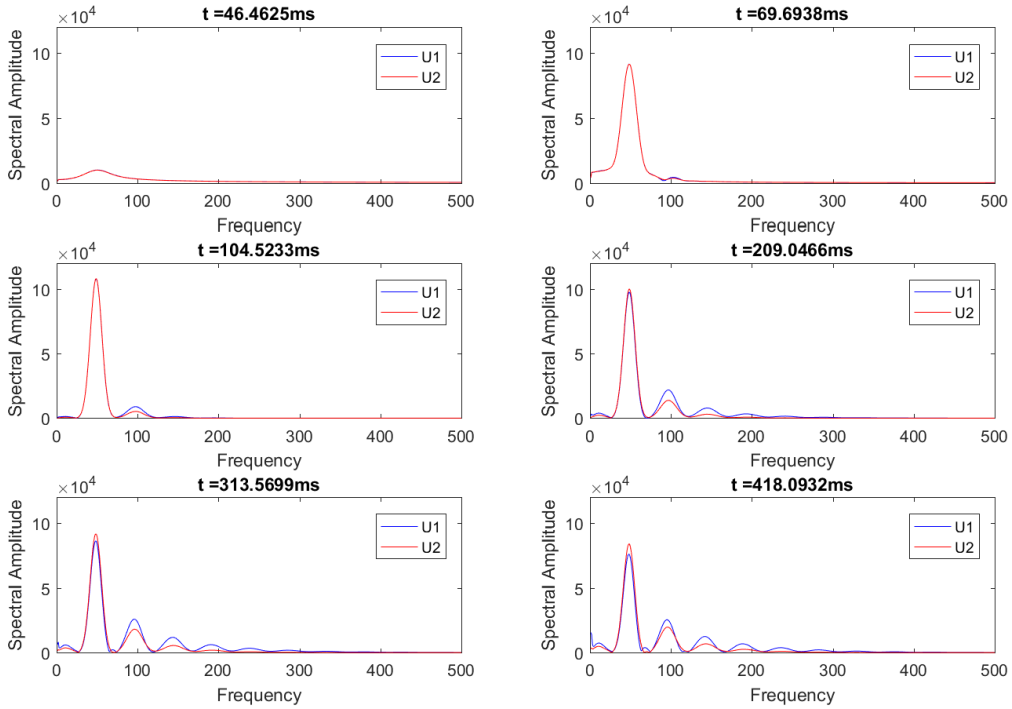

Figure 4.18: *The absolute values of the Fourier Transform of the Westervelt equation with attenuation term plotted at different times. The solutions for first and second order FEM are plotted together. Here* $P0 = 2 \cdot 10^6 Pa$.

4.5 Change in small signal sound speed

In all the figures above this point the parameters of the surroundings were constant during the propagation. Now we want to show the influence of a change in parameter during progation. So we assume that at a certain point in the mesh the material of the surroundings changes. We have chosen this point to be at the middle of the travel length. We change only the small signal sound speed of all the parameters. We appoint a larger small signal sound speed on the second half of the domain length than on the first half. So before the interface the wave has a velocity c_1 and after this interface it has a velocity c_2, with $c_1 < c_2$. Because of this velocity change there will be reflection and transmission of the wave. In Figure 4.19 we have plotted the first order FEM solution of the linear wave equation with three different small signal sound speeds c_2. In this figure we have chosen the three different velocities to be: 1800 m/s, 2000 m/s and 2500 m/s. The small signal sound speed before the interface is still 1500 m/s and the ambient density is still 1000 kg/m^3 on both sides of the domain. Now using Section 2.4 we can derive the reflection and transmission coefficients for these different solutions.

Table 2: *The reflection and transmission coefficients calculated for three solutions with different velocities after the interface. The reflection and transmission coefficients are derived using Equation 2.11, 2.12 and 2.13 from Section 2.4.*

	$R = \frac{Z_2 - Z_1}{Z_1 + Z_2}$	$T = \frac{2Z_2}{Z_1 + Z_2}$
$c_2 = 1600m/s$	0.09	1.09
$c_2 = 2000m/s$	0.14	1.14
$c_2 = 2500m/s$	0.25	1.25

So in Figure 4.19 one should see that for all the solutions the amplitude of the transmitted wave is larger than the incident wave and the amplitude of the reflected wave is a lot smaller than the amplitude of the incident wave. Also the solution of the wave with the highest c_2 has the largest amplitude for the reflected and transmitted wave. Now if one looks at Figure 4.19 the three solutions in the first frame are traveling at the exact same speed so at that moment there is no difference. However in the third and fourth frame it is clear that the amplitude of the transmitted wave is larger than the amplitude of the incident wave for all solutions and the solutions with the highest c_2 has the largest amplitude. This is the same for the reflected wave so our theory is right. In the figure one can see as well that the reflected waves of all solutions move at the same speed, because they move with the same c_1, but the transmitted waves move with different speeds, because of the difference in c_2.
For the Westervelt equation without and with attenuation term we have plotted a solution for a sudden change in velocity as well. Those are plotted in Figures 4.21 and 4.22. We have made the figures with parameter $c_2 = 2000$ m/s, so we have the same reflection and transmission coefficient for these solutions as the second line of Table 2. You see in these plots that the nonlinearty has influence on both the reflected as the transmitted wave. Also because of damping both the reflected and the transmitted wave are a lot smaller in Figure 4.22.

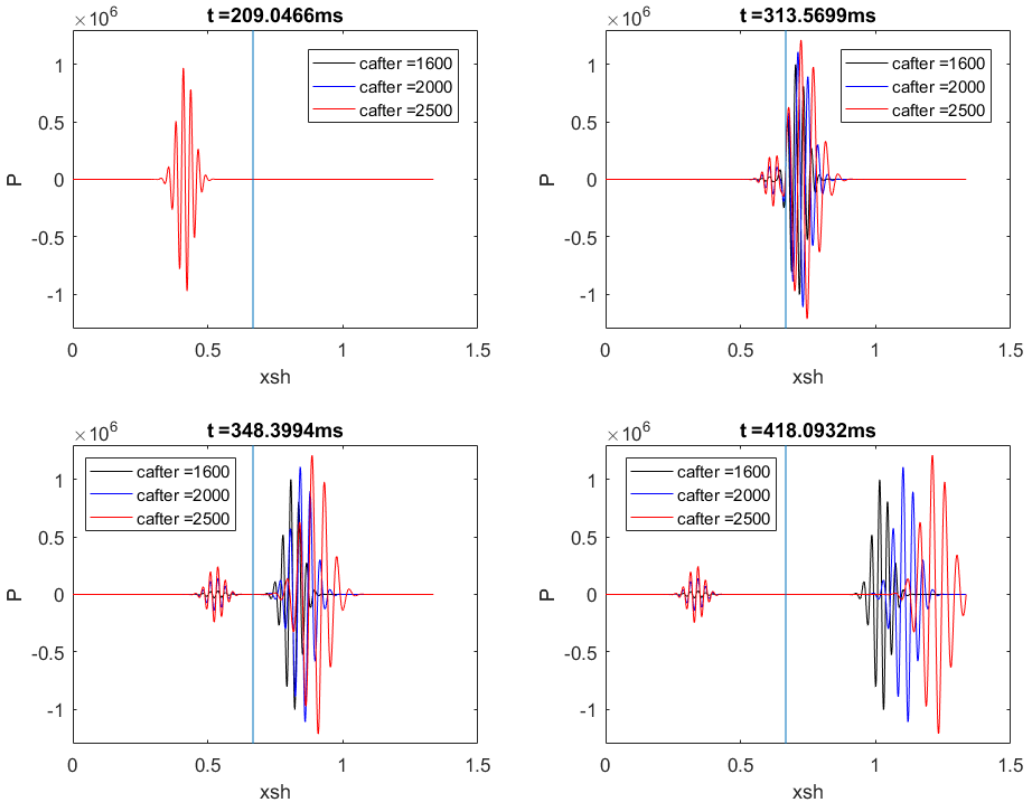

Figure 4.19: *The effect of a sudden change in velocity for three solutions. This is plotted for the first order FEM solutions of the linear wave equation at different times. The different velocities that the waves experience are given in the legend.*

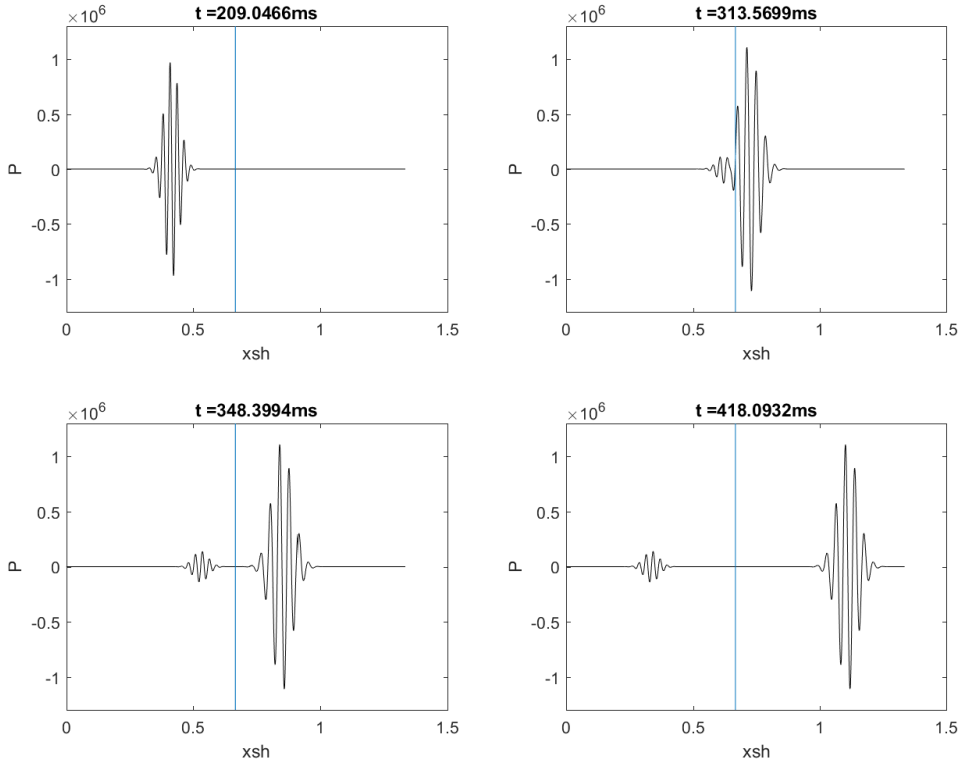

Figure 4.20: *The effect of a sudden change in velocity for the linear wave equation at different times. Here* $c_1 = 1500\ m/s$ *and* $c_2 = 2000\ m/s$

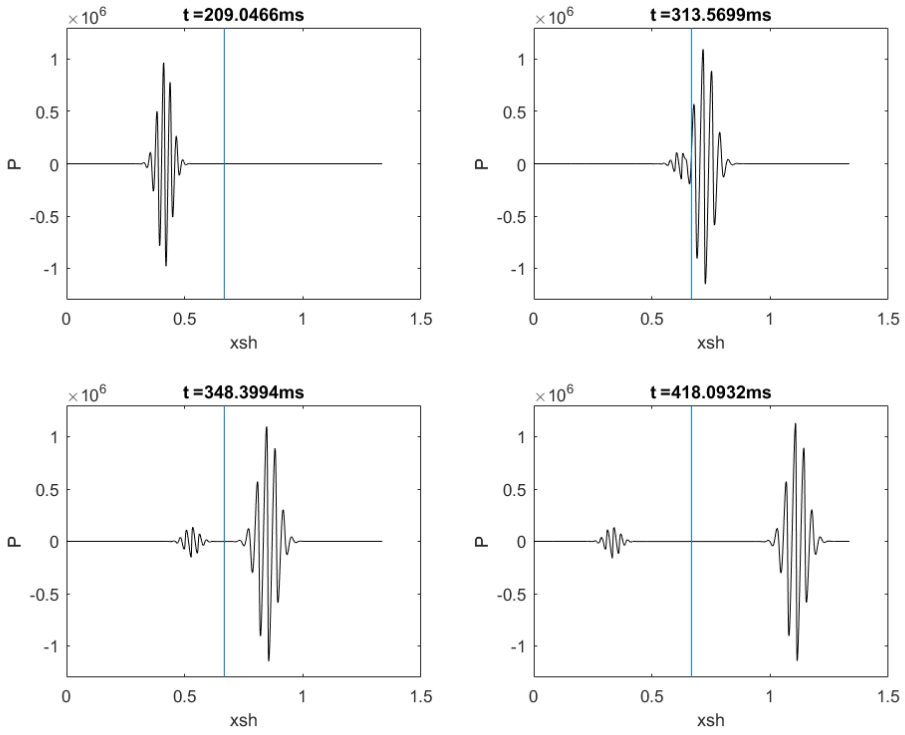

Figure 4.21: *The effect of a sudden change in velocity for the Westervelt equation without attenuation term at different times. Here* $c_1 = 1500\ m/s$ *and* $c_2 = 2000\ m/s$

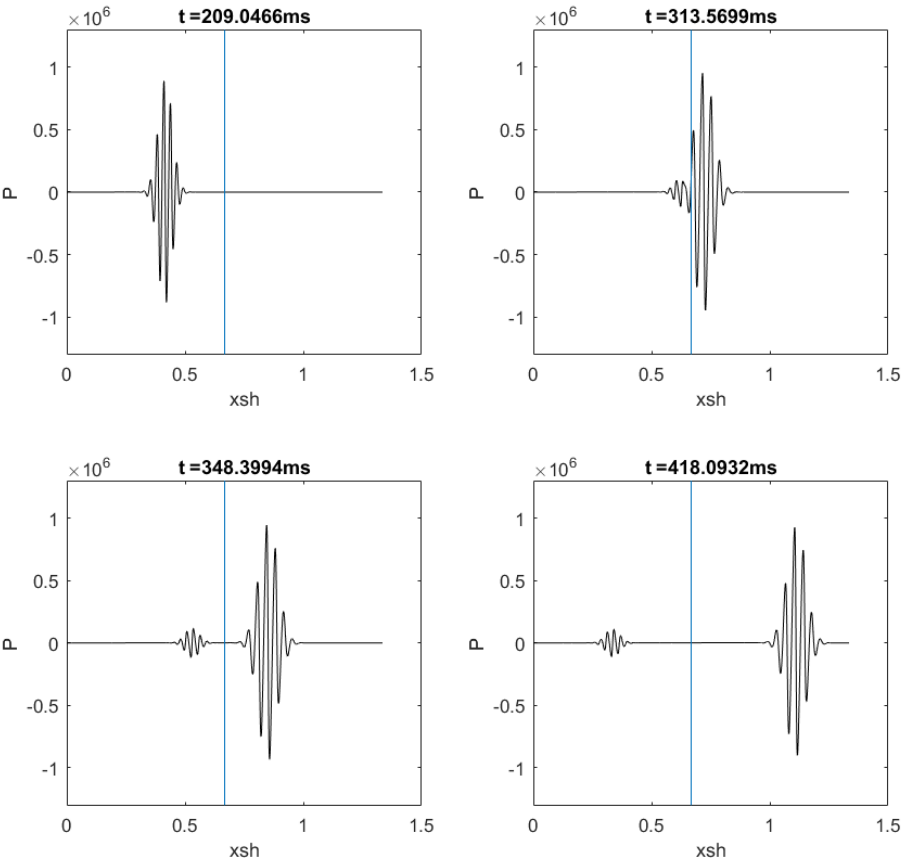

Figure 4.22: *The effect of a sudden change in velocity for the Westervelt equation with attenuation term at different times. Here* $c_1 = 1500\ m/s$ *and* $c_2 = 2000\ m/s$

5 Conclusion

In Chapter 1 we formulated the Westervelt equation and explained its relation to medical acoustic imaging. In this thesis we discuss using first order and second order FEM to solve the Westervelt equation and the observations we obtain from our different results using FEM.
We conclude from our simulations in Matlab that the finite element method is an adequate method to solve the Westervelt equation, because the analytic solution of the linear wave equation is the same as both solutions of FEM except for a small phase shift. For the solutions of the Westervelt equation without attenuation term we cannot say the solution completely agrees with the Burgers solution, but that is logical because they are all approximations. However, we can conclude that for approximations they are all valid. The finite element method cannot not be completely accurate in space because it depends on nonzero mesh size. Another thing that accounts for inaccuracy is our choice in basis functions. Higher-order basis functions depend on more points in the mesh. Taking more points into account gives us a better solution. Because of a finite amount of computation time, we cannot use an infinitely high-order basis functions. There is also an error in time, because we use backwards differential schemes to solve the equation iteratively in time. To improve accuracy we should put research into an adaptive mesh, where we choose a mesh that has a small mesh size around the peaks.
The difference between the first and second order finite element method was for the linear wave equation only caused by the small phase shift of the solutions. This phase shift was caused by the mesh size used in the FEM. For the Westervelt equation with and without attenuation term the difference was larger, because of the effect of the nonlinear term on the Westervelt equation. The nonlinearity caused the slope of the solution to the Westervelt equation to steepen when going from a maximum to a minimum. The attenuation term inhibited the effect of the nonlinearity on the form of the wave during propagation and thus made the slope of the wave less steep. When we looked at the solutions at two times the shock wave formation distance, it was clear that the solutions for the Westervelt equation without attenuation term were not usable. The enormous derivatives of the solution caused computation problems when using the finite element method. The solutions of the complete Westervelt equation at two times the shock wave formation distance were not deformed. From this we conclude that no shock waves were formed and thus the attenuation damped the effect of the nonlinear term.
In the frequency spectra of the solutions we saw the transfer of energy to higher harmonics for the solution for the equations with a nonlinear term. However, in the frequency spectra of the complete Westervelt equation there were a lot less peaks than in the spectra of the Westervelt equation without attenuation term. As we know the attenuation term damps the effect of the nonlinear term, which causes the transfer of energy to higher harmonics. So, the attenuation stopped the arising of more peaks
The finite element is a favorable method to solve a wave equation on an inhomogeneous domain. We chose an inhomogeneous domain with a sudden change in small signal sound speed of the wave in the middle of the domain. Before the interface the small signal sound speed was smaller than after the interface. The rest of the parameters were the same before and after this interface. Using theory about reflection and transmission we calculated the amplitude of the reflected and transmitted wave for different changes in small signal sound speed. From these amplitudes we concluded that for the largest change in small signal sound speed, the amplitudes of the reflected and transmitted waves are the largest. We simulated these waves with our implementation in Matlab and concluded that the theory about reflection and transmission agrees with our simulations.

Appendices

A Implementation of the Burgers solution

```
%Burgers solution
clc; clear all; close all;

%parameters
c0=1500;
rho=1000;
beta=10;
P0 = 1e6;
f0 = 1e5;

xsh = rho*c0^3/(beta*P0*2*pi*f0);
L = xsh +12*c0/f0;
n=round(72*f0*L/c0);
Mesh=linspace(0,L,n+1)';
dx=L/(length(Mesh));

dt=L/(n*c0*4);
dtau=L/(n*c0*4);
Td = 6/f0;
Tw = 3/f0;
T=L/c0;
t=[0:dt:T];
TauEnd = L/c0;
tau = [0:dtau:TauEnd];

%defining the source signal
source = P0.*sin(2*pi*f0*(tau-Td)).*exp(-((tau-Td)./(Tw/2)).^2); %Source signal
burgers(1,:) = source; % burgers(x=0,tau)

%interpolation
for j=2:length(Mesh)
    phi = tau + beta/(rho*c0^3) * dx * burgers(j-1,:); %calculating the shifted times
    burgers(j,:) = interp1(tau,burgers(j-1,:),phi); %interpolating in time to find ...
        the values
end

%Solution plotted at different times
time=round(xsh*T/L*1/dt + 6/f0/dt);
Time=[round(time/9) round(time/6) round(time/4) round(time/2) round(3*time/4) ...
    round(time)];
for i =1:6
    for k=1:length(Mesh)
        tijd(k)=round(Time(i)-1/dt*Mesh(k)/c0);
        if tijd(k)>0
            pburg(k,i)=burgers(k,tijd(k));
        else
            pburg(k,i)=0;
        end
    end
    plot(Mesh/xsh,pburg);
end
```

B Derivation of the backward differential formulas

In this section we briefly derive the backward differential equations used in the time stepping sequence. In the time stepping sequence we need the first, second and third differential equations. In this

appendix we drop vector notation, such as bold font, but the derivation applies to a single equation as well as to systems of (time depending) equations.

Suppose we want to estimate the first, second and third derivative at time t_k (denoted by a superscript)

$$\dot{u}^k \approx \frac{au^k + bu^{k-1} + cu^{k-2}}{\Delta t}, \tag{B.1a}$$

$$\ddot{u}^k \approx \frac{au^k + bu^{k-1} + cu^{k-2} + du^{k-3}}{(\Delta t)^2}, \tag{B.1b}$$

$$\dddot{u}^k \approx \frac{au^k + bu^{k-1} + cu^{k-2} + du^{k-3} + eu^{k-4}}{(\Delta t)^2} \tag{B.1c}$$

for unknown coefficients a, b, c, d, e (the coefficients with the same letter do not have the same for all the equations). Using Taylor expansion on each u around t_k we get

$$\begin{array}{lllllllllll}
u^k & = u^k & & & & & & & & & \\
u^{k-1} & = u^k & - & \Delta t\dot{u}^k & + & \frac{(\Delta t)^2}{2}\ddot{u}^k & - & \frac{(\Delta t)^3}{6}\dddot{u}^k & + & \frac{(\Delta t)^4}{24}\ddddot{u}^k & + \mathcal{O}\big((\Delta t)^5\big) \\
u^{k-2} & = u^k & - & 2\Delta t\dot{u}^k & + & 2\Delta t\ddot{u}^k & - & \frac{8(\Delta t)^3}{6}\dddot{u}^k & + & \frac{16(\Delta t)^4}{24}\ddddot{u}^k & + \mathcal{O}\big((\Delta t)^5\big) \\
u^{k-3} & = u^k & - & 3\Delta t\dot{u}^k & + & \frac{9(\Delta t)^2}{2}\ddot{u}^k & - & \frac{27(\Delta t)^3}{6}\dddot{u}^k & + & \frac{27(\Delta t)^4}{8}\ddddot{u}^k & + \mathcal{O}\big((\Delta t)^5\big) \\
u^{k-4} & = u^k & - & 4\Delta t\dot{u}^k & + & 8(\Delta t)^2\ddot{u}^k & - & \frac{32(\Delta t)^3}{3}\dddot{u}^k & + & \frac{32(\Delta t)^4}{3}\ddddot{u}^k & + \mathcal{O}\big((\Delta t)^5\big)
\end{array} \tag{B.2}$$

Now we just have to plug these equations into (B.1a), (B.1b) and (B.1c). So first we will try to find the coefficients $\{a, b, c\}$ of (B.1a) such that all higher and lower (including the zeroth) orders of derivatives vanish except for the first derivative. To do so, we solve the system of equations (given in matrix form), where we use the Taylor expansions of (B.2) up to order $\mathcal{O}\big((\Delta t)^3\big)$.

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & -1 & -2 \\ 0 & 1/2 & 2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \tag{B.3}$$

which has solution $\begin{bmatrix} 3/2 & -2 & 1/2 \end{bmatrix}^T$. Note that the smallest local truncation error in u $\mathcal{O}\big((\Delta t)^3\big)$ is, which means that the local truncation error in $\dot{u}^k$ is $\frac{\mathcal{O}\left((\Delta t)^3\right)}{(\Delta t)^1} = \mathcal{O}\big((\Delta t)^2\big)$.
The coefficients of the second and third order derivative can be found using systems of equations in matrix form as well. For the second order derivative the following of equations has to be solved.

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -1 & -2 & -3 \\ 0 & 1/2 & 2 & 9/2 \\ 0 & -1/6 & -8/6 & -27/6 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \tag{B.4}$$

which has the solution $\begin{bmatrix} 2 & -5 & 4 & -1 \end{bmatrix}^T$. Since the smallest local truncation error is $\mathcal{O}\big((\Delta t)^4\big)$ in u, the local truncation error in $\ddot{u}^k$ is $\frac{\mathcal{O}\left((\Delta t)^4\right)}{(\Delta t)^2} = \mathcal{O}\big((\Delta t)^2\big)$.

Now only the coefficients of the third order derivative have to be found.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 0 & -1 & -2 & -3 & -4 \\ 0 & 1/2 & 2 & 9/2 & 8 \\ 0 & -1/6 & -8/6 & -27/6 & -32/3 \\ 0 & 1/24 & 16/24 & 27/8 & 32/3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \tag{B.5}$$

which has the solution $\begin{bmatrix} 2.5 & -9 & 12 & -7 & 1.5 \end{bmatrix}^T$. Since the smallest local truncation error is $\mathcal{O}\big((\Delta t)^5\big)$ in u, the local truncation error in $\dddot{u}^k$ is $\frac{\mathcal{O}\left((\Delta t)^5\right)}{(\Delta t)^3} = \mathcal{O}\big((\Delta t)^2\big)$.

For the first, second and third derivative the diffential equations of order $\mathcal{O}\big((\Delta t)\big)$ have also been used in the time stepping sequence. The derivation of those differential equations are analogous to the derivations above.
So in conclusion we have

$$\begin{aligned}
\dot{u}^k &= \frac{3u^k-4u^{k-1}+u^{k-2}}{2\Delta t} &+ \mathcal{O}\big((\Delta t)^2\big)\\
\ddot{u}^k &= \frac{2u^k-5u^{k-1}+4u^{k-2}-u^{k-3}}{(\Delta t)^2} &+ \mathcal{O}\big((\Delta t)^2\big)\\
\dddot{u}^k &= \frac{2.5u^k-9u^{k-1}+12u^{k-2}-7u^{k-3}+1.5u^{k-4}}{(\Delta t)^3} &+ \mathcal{O}\big((\Delta t)^2\big)\\
\dot{u}^k &= \frac{u^k-u^{k-1}}{\Delta t} &+ \mathcal{O}\big(\Delta t\big)\\
\ddot{u}^k &= \frac{u^k-2u^{k-1}+u^{k-2}}{(\Delta t)^2} &+ \mathcal{O}\big(\Delta t\big)\\
\dddot{u}^k &= \frac{u^k-3u^{k-1}+3u^{k-2}-u^{k-3}}{(\Delta t)^3} &+ \mathcal{O}\big(\Delta t\big)
\end{aligned} \tag{B.6}$$

C Matlab code for the implementations of various finite element schemes

C.1 linear wave equation for both orders

```
clear all; close all;
tic
c = 1500;
rho = 1000;
f0 = 1e5;
P0 = 1e6;
beta = 10;

%xshock
xsh=rho*c^3/(beta*P0*2*pi*f0);

L=xsh+12*c/f0;
n=round(72*f0*L/c);
N=2*n;

Mesh=linspace(0,L,n+1)';

dt=L/(n*c*4);
T=L/c;
k=round(T/dt)+1;
t=0:dt:(k-1)*dt;

%first order FEM
K=Kmatrix(Mesh,n);
M=Mmatrix(Mesh,n,c);
[gM gK]=vectorg(Mesh,n,c);

u0t = zeros(size(t));
u0t=Pulse(t,P0,f0);

g1=sparse(n-1,length(t));
g1(1,:)=-u0t*gK-u0t*gM;

u1(:,1)=zeros(n-1,1);
u1(:,2)=u1(:,1)+dt*zeros(n-1,1);

A=dt^2*K+M;
B=dt^2*K+2*M;
u1(:,3)=A\(g1(:,3)*dt^2+2*M*u1(:,2)-M*u1(:,1));

for i=4:round(T/dt)+1
    u1(:,i)=B\(g1(:,i)*dt^2+5*M*u1(:,i-1)-4*M*u1(:,i-2)+M*u1(:,i-3));
end
U1=[u0t; u1; zeros(1,i)];

%second order FEM
Mesh2=linspace(0,L,N+1)';
elmat=GenerateTopologyother(n);
K=sparse(N+1,N+1);

for i=1:2:length(elmat)
    Ke(:,:)=GenerateElementK(Mesh2(i),Mesh2(i+1),Mesh2(i+2));
    for j=1:3
        for k=1:3
            K(elmat(i,j),elmat(i,k))=K(elmat(i,j),elmat(i,k))+Ke(j,k);
        end
    end
end
```

```
k=GenerateElementK(Mesh2(1),Mesh2(2),Mesh2(3));
gK1=k(1,2);
gK2=k(1,3);
K=K(2:N,2:N);

M=sparse(N+1,N+1);
for i=1:2:length(elmat)
    Me(:,:)=GenerateElementM(Mesh2(i),Mesh2(i+1),Mesh2(i+2));
    for j=1:3
        for k=1:3
            M(elmat(i,j),elmat(i,k))=M(elmat(i,j),elmat(i,k))+Me(j,k);
        end
    end
end
M=M*1/c^2;
m=GenerateElementM(Mesh2(1),Mesh2(2),Mesh2(3));
gM1=m(1,2)*1/c^2;
gM2=m(1,3)*1/c^2;
M=M(2:N,2:N);

g2=sparse(zeros(N-1,length(t)));
g2(1,:)=-u0t*gK1-( [0 u0t(1:round(T/dt))] -2*u0t + [u0t(2:round(T/dt)+1) 0] ...
    )*gM1/dt^2;
g2(2,:)=-u0t*gK2-( [0 u0t(1:round(T/dt))] -2*u0t + [u0t(2:round(T/dt)+1) 0] ...
    )*gM2/dt^2;

u2(:,1)=zeros(N-1,1);
u2(:,2)=u2(N-1,1)+dt*zeros(N-1,1);

u2(:,3)=(dt^2*K+M)\(g2(:,3)*dt^2+2*M*u2(:,2)-M*u2(:,1));
for i=4:round(T/dt)+1
    u2(:,i)=(dt^2*K+2*M)\(g2(:,i)*dt^2+5*M*u2(:,i-1)-4*M*u2(:,i-2)+M*u2(:,i-3));
end
U2=[u0t; u2; zeros(1,i)];

Uanalytic = linearpropagation(f0,P0,c,Mesh,t); %Linear Analytic solution
figure(1) %plot U1,U2,Uanalytic
time=round(xsh*T/L*1/dt + 6/f0/dt);
plot(Mesh/xsh,U1(:,time),'r')
axis([0.9 1.1 -1.3*10^6 1.3*10^6])
hold on
plot(Mesh2/xsh,U2(:,time),'k')
plot(Mesh/xsh,Uanalytic(:,time),'b')
xlabel('xshock')
ylabel('P')
strtime=num2str(time*dt*10^6);
title(strcat('t = ', strtime, 'ms'));
legend('U1','U2','Uanalytic')

figure(2) %plot difference U1 and U2
for l=1:length(Mesh)-1
    difU(l,:)=(U2(2*l,:)+U2(2*l-1,:))/2-U1(l,:);
end
subplot(2,2,1)
plot(Mesh/xsh,[0; difU(:,round(time/4))])
hold on
plot(Mesh/xsh,[0; diflin1(:,round(time/4))])
plot(Mesh/xsh,[0; diflin2(:,round(time/4))])
axis([0 1.5 -1*10^5 1*10^5])
xlabel('xshock')
ylabel('P')
strtime=num2str(time/4*dt*10^6);
title(strcat('t = ', strtime, 'ms'));
subplot(2,2,2)
plot(Mesh/xsh,[0; difU(:,round(2*time/4))])
hold on
```

```
plot(Mesh/xsh,[0; diflin1(:,round(2*time/4))])
plot(Mesh/xsh,[0; diflin2(:,round(2*time/4))])
axis([0 1.5 -1*10^5 1*10^5])
strtime=num2str(time/2*dt*10^6);
title(strcat('t = ', strtime, 'ms'));
xlabel('xshock')
ylabel('P')
subplot(2,2,3)
plot(Mesh/xsh,[0; difU(:,round(3*time/4))])
hold on
plot(Mesh/xsh,[0; diflin1(:,round(3*time/4))])
plot(Mesh/xsh,[0; diflin2(:,round(3*time/4))])
axis([0 1.5 -1*10^5 1*10^5])
strtime=num2str(time*3/4*dt*10^6);
title(strcat('t = ', strtime, 'ms'));
xlabel('xshock')
ylabel('P')
subplot(2,2,4)
plot(Mesh/xsh,[0; difU(:,round(4*time/4))])
hold on
plot(Mesh/xsh,[0; diflin1(:,round(4*time/4))])
plot(Mesh/xsh,[0; diflin2(:,round(4*time/4))])
axis([0 1.5 -1*10^5 1*10^5])
strtime=num2str(time*dt*10^6);
title(strcat('t = ', strtime, 'ms'));
xlabel('xshock')
ylabel('P')
```

C.2 the Westervelt equation without attenuation term for both orders

```
clear all;close all;
tic
c = 1500;
rho = 1000;
f0 = 1e5;
P0 = 1e6;
beta = 10;
a=beta/(rho*c^4);

xsh=rho*c^3/(beta*P0*2*pi*f0); %xshock
L=xsh+12*c/f0;
n=round(72*f0*L/c);
Mesh=linspace(0,L,n+1)';

dt=L/(n*c*4);
T=L/c;
k=round(T/dt)+1;
t=0:dt:(k-1)*dt;

%1st order
%define matrices K,M and the C's
M=Mmatrix(Mesh,n,c);
K=Kmatrix(Mesh,n);
[gM gK]= vectorg(Mesh,n,c);

[C1 C2 C3 gC1 gC2 gC3]=Cmatrix(Mesh, beta ,rho ,c,n);

%BC
u0t = Pulse(t, P0, f0);
u0t_tt=Pulsett(t,P0,f0);
u0t_t=Pulset(t,P0,f0);
g1 = sparse(ones(1,k),1:k,-u0t_tt*gM - ...
    u0t*gK+2*a*u0t_t.*(u0t_t*gC1+u0t_t*gC2)+2*a*u0t_tt.*(u0t*gC1+u0t*gC2),n-1,k,k);

%IC
u1=zeros(n-1,1);
ut1=zeros(n-1,1);

u2=u1+dt.*ut1;

% time solver, linear
W1=zeros(n-1,k);
W1(:,1)=u1;
W1(:,2)=u2;

W1(:,3)=(dt^2*K+M)\(g1(:,3)*dt^2+2*M*W1(:,2)-M*W1(:,1));
for j=4:k
    W1(:,j)=(dt^2*K+2*M)\(g1(:,j)*dt^2+5*M*W1(:,j-1)-4*M*W1(:,j-2)+M*W1(:,j-3));

end

%time solver, non-linear
U1=zeros(n-1,k);
U1(:,1)=u1;
U1(:,2)=u2;

iteratie1=zeros(1,k);

% now we solve for i=3
for i=3
    fguess=(dt^2*K+M)\(g1(:,3)*dt^2+2*M*U1(:,2)-M*U1(:,1));
    ft=3*P0*ones(size(fguess));
```

```
62
63      while (max(abs(ft-fguess))>10^-9*P0);
64          ft=fguess;
65          iteratie1(i)=iteratie1(i)+1;
66
67          HA=sparse(1:n-1,1:n-1,([u0t(i); ft(1:n-2)]-[u0t(i-1); U1(1:n-2,i-1)])/dt , ...
                n-1,n-1,n-1);
68          HB=sparse(1:n-1,1:n-1,(ft-U1(:,i-1))/dt  , n-1,n-1,n-1);
69          HC=sparse(1:n-1,1:n-1,([ft(2:n-1);0]-[U1(2:n-1,i-1);0])/dt  , n-1,n-1,n-1);
70          N1=HA*C1+HB*C2+HC*C3; clear HA HB HC;
71
72          LA=sparse(1:n-1,1:n-1,[u0t(i); ft(1:n-2)]  , n-1,n-1,n-1);
73          LB=sparse(1:n-1,1:n-1,ft , n-1,n-1,n-1);
74          LC=sparse(1:n-1,1:n-1,[ft(2:n-1); 0]  , n-1,n-1,n-1);
75          N2=LA*C1+LB*C2+LC*C3; clear LA LB LC;
76
77          fguess=(dt^2*K+M-N1*dt-N2)\(g1(:,i)*dt^2+2*M*U1(:,i-1)-M*U1(:,i-2)...
78              -N1*U1(:,i-1)*dt+N2*(-2*U1(:,i-1)+U1(:,i-2)));
79
80      end
81      U1(:,i)=fguess;
82  end
83
84  for i=4:k
85      fguess=(dt^2*K+2*M)\(g1(:,j)*dt^2+5*M*U1(:,j-1)-4*M*U1(:,j-2)+M*U1(:,j-3));
86      ft=3*P0*ones(size(fguess));
87
88      while (max(abs(ft-fguess))>10^-6*P0);
89          ft=fguess;
90           iteratie1(i)=iteratie1(i)+1;
91
92
93          HA=sparse(1:n-1,1:n-1,(3*[u0t(i); ft(1:n-2)]-4*[u0t(i-1); U1(1:n-2,i-1)]...
94              +[u0t(i-2); U1(1:n-2,i-2)])/(2*dt) , n-1,n-1,n-1);
95          HB=sparse(1:n-1,1:n-1,(3*ft-4*U1(:,i-1)+U1(:,i-2))/(2*dt)  , n-1,n-1,n-1);
96          HC=sparse(1:n-1,1:n-1,(3*[ft(2:n-1);0]-4*[U1(2:n-1,i-1);0]+...
97              [U1(2:n-1,i-2);0])/(2*dt)  , n-1,n-1,n-1);
98          N1=HA*C1+HB*C2+HC*C3; clear HA HB HC;
99
100         LA=sparse(1:n-1,1:n-1,[u0t(i); ft(1:n-2)]  , n-1,n-1,n-1);
101         LB=sparse(1:n-1,1:n-1,ft , n-1,n-1,n-1);
102         LC=sparse(1:n-1,1:n-1,[ft(2:n-1); 0]  , n-1,n-1,n-1);
103         N2=LA*C1+LB*C2+LC*C3; clear LA LB LC;
104
105         fguess=(dt^2*K+2*M-3/2*N1*dt-2*N2)\(g1(:,i)*dt^2+...
106             (5*M+N1*-4*dt/2-5*N2)*U1(:,i-1)+(-4*M+1/2*dt*N1+4*N2)*U1(:,i-2)+...
107             (M-N2)*U1(:,i-3));
108
109     end
110     U1(:,i)=fguess;
111 end
112
113 U1=[u0t;U1;zeros(1,k)];
114 w1=[u0t; W1;zeros(1,k)];
115
116 Uanalytic = linearpropagation(f0,P0,c,Mesh,t); %Linear Analytic solution
117
118 %2nd order
119 N=2*n;
120 Mesh2=linspace(0,L,N+1)';
121 B=round(T/dt)+1;
122
123 %define matrices K,M and the C's
124 elmat=GenerateTopologyother(n);
125 K=sparse(N+1,N+1);
126
```

```
for i=1:2:length(elmat)
    Ke(:,:)=GenerateElementK(Mesh2(i),Mesh2(i+1),Mesh2(i+2));
    for j=1:3
        for k=1:3
            K(elmat(i,j),elmat(i,k))=K(elmat(i,j),elmat(i,k))+Ke(j,k);
        end
    end
end
k=GenerateElementK(Mesh2(1),Mesh2(2),Mesh2(3));
gK1=k(1,2);
gK2=k(1,3);
K=K(2:N,2:N);

M=sparse(N+1,N+1);
for i=1:2:length(elmat)
    Me(:,:)=GenerateElementM(Mesh2(i),Mesh2(i+1),Mesh2(i+2));
    for j=1:3
        for k=1:3
            M(elmat(i,j),elmat(i,k))=M(elmat(i,j),elmat(i,k))+Me(j,k);
        end

    end
end
M=M*1/c^2;
m=GenerateElementM(Mesh2(1),Mesh2(2),Mesh2(3));
gM1=m(1,2)*1/c^2;
gM2=m(1,3)*1/c^2;
M=M(2:N,2:N);

C1=sparse(N+1,N+1);
C2=sparse(N+1,N+1);
C3=sparse(N+1,N+1);
C4=sparse(N+1,N+1);
C5=sparse(N+1,N+1);
for i=1:2:length(elmat)-2

    [Ce1(:,:), Ce2(:,:), Ce3(:,:), Ce4(:,:), Ce5(:,:) ...
        ]=GenerateElementCtry(Mesh2(i),Mesh2(i+1),Mesh2(i+2), Mesh2(i+3), ...
        Mesh2(i+4) , beta, rho, c);

    for j=1:3
        for k=1:3
            C1(elmat(i,j),elmat(i,k))=C1(elmat(i,j),elmat(i,k))+Ce1(j,k);
            C2(elmat(i,j),elmat(i,k))=C2(elmat(i,j),elmat(i,k))+Ce2(j,k);
            C3(elmat(i,j),elmat(i,k))=C3(elmat(i,j),elmat(i,k))+Ce3(j,k);
            C4(elmat(i,j),elmat(i,k))=C4(elmat(i,j),elmat(i,k))+Ce4(j,k);
            C5(elmat(i,j),elmat(i,k))=C5(elmat(i,j),elmat(i,k))+Ce5(j,k);
        end
    end
end
for i=length(elmat)
    [Ce1(:,:), Ce2(:,:), Ce3(:,:), Ce4(:,:), Ce5(:,:) ...
        ]=GenerateElementC(Mesh2(i),Mesh2(i+1),Mesh2(i+2), beta, rho, c);

    for j=1:3
        for k=1:3
            C1(elmat(i,j),elmat(i,k))=C1(elmat(i,j),elmat(i,k))+Ce1(j,k);
            C2(elmat(i,j),elmat(i,k))=C2(elmat(i,j),elmat(i,k))+Ce2(j,k);
            C3(elmat(i,j),elmat(i,k))=C3(elmat(i,j),elmat(i,k))+Ce3(j,k);
            C4(elmat(i,j),elmat(i,k))=C4(elmat(i,j),elmat(i,k))+Ce4(j,k);
            C5(elmat(i,j),elmat(i,k))=C5(elmat(i,j),elmat(i,k))+Ce5(j,k);
        end
    end
end

gC12=C1(1,3); gC22=C2(1,3);
```

```
190
191 C1=C1(2:N,2:N);
192 C2=C2(2:N,2:N);
193 C3=C3(2:N,2:N);
194 C4=C4(2:N,2:N);
195 C5=C5(2:N,2:N);
196
197 %BC
198 u0t = Pulse(t, P0, f0);
199 u0t_tt=Pulsett(t,P0,f0);
200 u0t_t=Pulset(t,P0,f0);
201
202 g2=sparse(zeros(N-1,B));
203 g2(1,:)=(-u0t*gK1-u0t_tt*gM1+a*u0t_t.*(u0t_t*gC22));
204 g2(2,:)=(-u0t*gK2-u0t_tt*gM2+a*u0t_tt.*(u0t*gC12));
205
206 gzC2=sparse(zeros(N-1,B));
207 gzC2(1,:)=(-u0t'*gK1-u0t_tt'*gM1)';
208 gzC2(2,:)=(-u0t'*gK2-u0t_tt'*gM2)';
209
210 %IC
211 u1=zeros(N-1,1);
212 ut1=zeros(N-1,1);
213
214 u2=u1+dt.*ut1;
215
216 % time solver, linear
217 W2=zeros(N-1,B);
218 W2(:,1)=u1;
219 W2(:,2)=u2;
220
221 W2(:,3)=(dt^2*K+M)\(gzC2(:,3)*dt^2+2*M*W2(:,2)-M*W2(:,1));
222
223 for j=4:B
224     W2(:,j)=(dt^2*K+2*M)\(gzC2(:,j)*dt^2+5*M*W2(:,j-1)-4*M*W2(:,j-2)+M*W2(:,j-3));
225
226 end
227
228 %time solver, non-linear
229 G=zeros(N-1,B);
230 G(:,1)=u1;
231 G(:,2)=u2;
232
233 iteratie2=zeros(1,B);
234 %
235 % now we solve for i=3
236 for i=3
237     fguess=(dt^2*K+M)\(g2(:,3)*dt^2+2*M*G(:,2)-M*G(:,1));
238     ft=3*P0*ones(size(fguess));
239
240     while (max(abs(ft-fguess))>10^-9*P0);
241         ft=fguess;
242         iteratie2(i)=iteratie2(i)+1;
243
244         HA=sparse(1:N-1,1:N-1,([0; u0t(i); ft(1:N-3)]-[0;u0t(i-1); ...
                G(1:N-3,i-1)])/dt ,N-1,N-1,N-1);
245         HB=sparse(1:N-1,1:N-1,([u0t(i); ft(1:N-2)]-[u0t(i-1); G(1:N-2,i-1)])/dt ...
                ,N-1,N-1,N-1);
246         HC=sparse(1:N-1,1:N-1,(ft-G(:,i-1))/dt  , N-1,N-1,N-1);
247         HD=sparse(1:N-1,1:N-1,([ft(2:N-1);0]-[G(2:N-1,i-1);0])/dt  ,N-1,N-1,N-1);
248         HE=sparse(1:N-1,1:N-1,([ft(3:N-1);0;0]-[G(3:N-1,i-1);0;0])/dt  ,N-1,N-1,N-1);
249         G1=HA*C1+HB*C2+HC*C3+HD*C4+HE*C5; clear HA HB HC HD HE;
250
251         LA=sparse(1:N-1,1:N-1,[0;u0t(i); ft(1:N-3)]  , N-1,N-1,N-1);
252         LB=sparse(1:N-1,1:N-1,[u0t(i); ft(1:N-2)]  , N-1,N-1,N-1);
253         LC=sparse(1:N-1,1:N-1,ft , N-1,N-1,N-1);
```

```
254         LD=sparse(1:N-1,1:N-1,[ft(2:N-1);0]  , N-1,N-1,N-1);
255         LE=sparse(1:N-1,1:N-1,[ft(3:N-1);0; 0]  , N-1,N-1,N-1);
256         G2=LA*C1+LB*C2+LC*C3+LD*C4+LE*C5; clear LA LB LC LD LE;
257
258         fguess=(dt^2*K+M-G1*dt-G2)\(g2(:,i)*dt^2+2*M*G(:,i-1)-...
259             M*G(:,i-2)-G1*G(:,i-1)*dt+G2*(-2*G(:,i-1)+G(:,i-2)));
260
261     end
262     G(:,i)=fguess;
263 end
264
265 for i=4:B
266     fguess=(dt^2*K+2*M)\(g2(:,j)*dt^2+5*M*G(:,j-1)-4*M*G(:,j-2)+M*G(:,j-3));
267     ft=3*P0*ones(size(fguess));
268
269     while (max(abs(ft-fguess))>10^-6*P0);
270         ft=fguess;
271         iteratie2(i)=iteratie2(i)+1;
272
273         HA=sparse(1:N-1,1:N-1,([u0t(i); u0t(i); ft(1:N-3)]-[u0t(i);u0t(i-1); ...
                G(1:N-3,i-1)])/dt ,N-1,N-1,N-1);
274         HB=sparse(1:N-1,1:N-1,([u0t(i); ft(1:N-2)]-[u0t(i-1); G(1:N-2,i-1)])/dt ...
                ,N-1,N-1,N-1);
275         HC=sparse(1:N-1,1:N-1,(ft-G(:,i-1))/dt  , N-1,N-1,N-1);
276         HD=sparse(1:N-1,1:N-1,([ft(2:N-1);0]-[G(2:N-1,i-1);0])/dt  ,N-1,N-1,N-1);
277         HE=sparse(1:N-1,1:N-1,([ft(3:N-1);0;0]-[G(3:N-1,i-1);0;0])/dt  ,N-1,N-1,N-1);
278         G1=HA*C1+HB*C2+HC*C3+HD*C4+HE*C5; clear HA HB HC HD HE;
279
280         LA=sparse(1:N-1,1:N-1,[u0t(i);u0t(i); ft(1:N-3)]  , N-1,N-1,N-1);
281         LB=sparse(1:N-1,1:N-1,[u0t(i); ft(1:N-2)]  , N-1,N-1,N-1);
282         LC=sparse(1:N-1,1:N-1,ft , N-1,N-1,N-1);
283         LD=sparse(1:N-1,1:N-1,[ft(2:N-1);0]  , N-1,N-1,N-1);
284         LE=sparse(1:N-1,1:N-1,[ft(3:N-1);0; 0]  , N-1,N-1,N-1);
285         G2=LA*C1+LB*C2+LC*C3+LD*C4+LE*C5; clear LA LB LC LD LE;
286
287         fguess=(dt^2*K+2*M-3/2*G1*dt-2*G2)\(g2(:,i)*dt^2+...
288             (5*M+G1*-4*dt/2-5*G2)*G(:,i-1)+(-4*M+1/2*dt*G1+4*G2)*G(:,i-2)+...
289             (M-G2)*G(:,i-3));
290
291     end
292     G(:,i)=fguess;
293 end
294
295 U2=[u0t;G;zeros(1,B)];
296 w2=[u0t; W2;zeros(1,B)];
297
298 figure(1) %plot U1,U2,Burgers
299 time=round(xsh*T/L*1/dt + 6/f0/dt);
300 plot(Mesh/xsh,U1(:,time),'r')
301 axis([0.9 1.1 -1.5*10^6 1.5*10^6])
302 hold on
303 plot(Mesh2/xsh,U2(:,time),'k')
304 plot(Mesh/xsh,pburg,'b')
305 xlabel('xshock')
306 ylabel('P')
307 strtime=int2str(time*dt*10^6);
308 title(strcat('t = ', strtime, 'ms'));
309 legend('U1','U2','Burgers')
310
311 figure(2) %plot difference U1 and U2
312 for l=1:length(Mesh)-1
313     difU(l,:)=(U2(2*l,:)+U2(2*l-1,:))/2-U1(l,:);
314 end
315 subplot(2,2,1)
316 plot(Mesh/xsh,[0; difU(:,round(time/4))])
317 axis([0 1.5 -7*10^5 7*10^5])
```

```
318 xlabel('xshock')
319 ylabel('P')
320 strtime=int2str(time/4*dt*10^6);
321 title(strcat('t = ', strtime, 'ms'));
322 subplot(2,2,2)
323 plot(Mesh/xsh,[0; difU(:,round(2*time/4))])
324 axis([0 1.5 -7*10^5 7*10^5])
325 strtime=int2str(time/2*dt*10^6);
326 title(strcat('t = ', strtime, 'ms'));
327 xlabel('xshock')
328 ylabel('P')
329 subplot(2,2,3)
330 plot(Mesh/xsh,[0; difU(:,round(3*time/4))])
331 axis([0 1.5 -7*10^5 7*10^5])
332 strtime=int2str(time*3/4*dt*10^6);
333 title(strcat('t = ', strtime, 'ms'));
334 xlabel('xshock')
335 ylabel('P')
336 subplot(2,2,4)
337 plot(Mesh/xsh,[0; difU(:,round(4*time/4))])
338 axis([0 1.5 -7*10^5 7*10^5])
339 strtime=int2str(time*dt*10^6);
340 title(strcat('t = ', strtime, 'ms'));
341 xlabel('xshock')
342 ylabel('P')
343
344 figure(3)
345 plot(1:B,iteratie1)
346 axis([0 14000 0 5])
347
348
349 figure(4)
350 plot(1:B,iteratie2)
351 axis([0 14000 0 5])
```

C.3 The complete Westervelt equation for both orders

```
clear all;close all;
tic
c = 1500;
rho = 1000;
f0 = 1e5;
P0 = 1e6;
beta = 10;
delta=0.007;

xsh=rho*c^3/(beta*P0*2*pi*f0);
L=xsh+12*c/f0;
n=round(72*f0*L/c);
Mesh=linspace(0,L,n+1)';

dt=L/(n*c*4);
T=L/c;
k=round(T/dt)+1;
t=0:dt:(k-1)*dt;

%first order FEM
%define matrices K,M and the C's
K=Kmatrix(Mesh,n);
M=Mmatrixd(Mesh,n);
diffmesh=diff(Mesh);
gM=1/6*diffmesh(1);
gK=-1/diffmesh(1);

[C1 C2 C3 gC1 gC2 gC3]=Cmatrix(Mesh, beta ,rho ,c,n);

%BC
u0t = Pulse(t, P0, f0);
u0t_tt=Pulsett(t,P0,f0);

g1 = sparse(ones(1,k),1:k,-u0t_tt*gM/c^2 - u0t*gK+delta/c^4*gM*( -1/2*[0 0 ...
    u0t(1:round(T/dt)-1)]+[0 u0t(1:round(T/dt))]  -[u0t(2:round(T/dt)+1) 0] ...
    +1/2*[u0t(3:round(T/dt)+1) 0 0] )*1/(dt^3),n-1,k,k);

%IC
u1=zeros(n-1,1);
ut1=zeros(n-1,1);

u2=u1+dt.*ut1;

% time solver, linear
W1=zeros(n-1,k);
W1(:,1)=u1;
W1(:,2)=u2;

W1(:,3)=(dt^3*K+dt/c^2*M-delta/c^4*M)\(g1(:,3)*dt^3+2*dt/c^2*M*W1(:,2)-...
    dt/c^2*M*W1(:,1)+3*delta/c^4*M*W1(:,1)-3*delta/c^4*M*W1(:,2));
W1(:,4)=(dt^3*K+dt*2/c^2*M-delta/c^4*M)\(g1(:,4)*dt^3+dt/c^2*5*M*W1(:,3)-...
    dt/c^2*4*M*W1(:,2)+dt/c^2*M*W1(:,1)-delta/c^4*M*W1(:,1)+...
    3*delta/c^4*M*W1(:,2)-3*delta/c^4*M*W1(:,3));
for i=5:round(T/dt)
    W1(:,i)=(dt^3*K+2*dt/c^2*M-2.5*delta/c^4*M)\(g1(:,i)*dt^3+...
        dt/c^2*5*M*W1(:,i-1)-dt/c^2*4*M*W1(:,i-2)+dt/c^2*M*W1(:,i-3)-...
        9*delta/c^4*M*W1(:,i-1)+12*delta/c^4*M*W1(:,i-2)-...
        7*delta/c^4*M*W1(:,i-3)+1.5*delta/c^4*M*W1(:,i-4));
end

%time solver, non-linear
U1=zeros(n-1,k);
```

```
U1(:,1)=u1;
U1(:,2)=u2;

%solve for i=3
i=3;
fguess=W1(:,i);
ft=3*P0*ones(size(fguess));

while (max(abs(ft-fguess))>10^-9*P0);
    ft=fguess;

    HA=sparse(1:n-1,1:n-1,([u0t(i); ft(1:n-2)]-[u0t(i-1); U1(1:n-2,i-1)])/dt , ...
        n-1,n-1,n-1);
    HB=sparse(1:n-1,1:n-1,(ft-U1(:,i-1))/dt  , n-1,n-1,n-1);
    HC=sparse(1:n-1,1:n-1,([ft(2:n-1);0]-[U1(2:n-1,i-1);0])/dt  , n-1,n-1,n-1);

    N1=HA*C1+HB*C2+HC*C3; clear HA HB HC;

    LA=sparse(1:n-1,1:n-1,[u0t(i); ft(1:n-2)]  , n-1,n-1,n-1);
    LB=sparse(1:n-1,1:n-1,ft , n-1,n-1,n-1);
    LC=sparse(1:n-1,1:n-1,[ft(2:n-1); 0]  , n-1,n-1,n-1);
    N2=LA*C1+LB*C2+LC*C3; clear LA LB LC;

    fguess=(dt^3*K+dt/c^2*M-delta/c^4*M-N1*dt^2-N2*dt)\(g1(:,i)*dt^3+...
        2*dt/c^2*M*U1(:,i-1)-dt/c^2*M*U1(:,i-2)-N1*U1(:,i-1)*dt^2+...
        dt*N2*(-2*U1(:,i-1)+U1(:,i-2))-3*delta/c^4*M*U1(:,i-1)+...
        3*delta/c^4*M*U1(:,i-2));

end
U1(:,i)=fguess;

%solve for i=4
for i=4
    fguess=W1(:,i);
    ft=3*P0*ones(size(fguess));

    while (max(abs(ft-fguess))>10^-9*P0);
        ft=fguess;

        HA=sparse(1:n-1,1:n-1,([u0t(i); ft(1:n-2)]-[u0t(i-1); U1(1:n-2,i-1)])/dt , ...
            n-1,n-1,n-1);
        HB=sparse(1:n-1,1:n-1,(ft-U1(:,i-1))/dt  , n-1,n-1,n-1);
        HC=sparse(1:n-1,1:n-1,([ft(2:n-1);0]-[U1(2:n-1,i-1);0])/dt  , n-1,n-1,n-1);
        N1=HA*C1+HB*C2+HC*C3; clear HA HB HC;

        LA=sparse(1:n-1,1:n-1,[u0t(i); ft(1:n-2)]  , n-1,n-1,n-1);
        LB=sparse(1:n-1,1:n-1,ft , n-1,n-1,n-1);
        LC=sparse(1:n-1,1:n-1,[ft(2:n-1); 0]  , n-1,n-1,n-1);
        N2=LA*C1+LB*C2+LC*C3; clear LA LB LC;

        fguess=(dt^3*K+dt*2/c^2*M-delta/c^4*M-N1*dt^2-N2*dt)\(g1(:,i)*dt^3+...
            dt/c^2*5*M*U1(:,i-1)-dt/c^2*4*M*U1(:,i-2)+dt/c^2*M*U1(:,i-3)-...
            delta/c^4*M*U1(:,i-3)+3*delta/c^4*M*U1(:,i-2)-...
            3*delta/c^4*M*U1(:,i-1)-N1*U1(:,i-1)*dt^2+dt*N2*(-2*U1(:,i-1)+U1(:,i-2)));

    end
    U1(:,i)=fguess;

end
for i=5:k
    fguess=W1(:,i);
    ft=3*P0*ones(size(fguess));

    while (max(abs(ft-fguess))>10^-9*P0);
        ft=fguess;

```

```
125         HA=sparse(1:n-1,1:n-1,(3*[u0t(i); ft(1:n-2)]-4*[u0t(i-1); ...
                U1(1:n-2,i-1)]+[u0t(i-2); U1(1:n-2,i-2)])/(2*dt) , n-1,n-1,n-1);
126         HB=sparse(1:n-1,1:n-1,(3*ft-4*U1(:,i-1)+U1(:,i-2))/(2*dt)  , n-1,n-1,n-1);
127         HC=sparse(1:n-1,1:n-1,(3*[ft(2:n-1);0]-4*[U1(2:n-1,i-1);0]+...
128             [U1(2:n-1,i-2);0])/(2*dt)  , n-1,n-1,n-1);
129         N1=HA*C1+HB*C2+HC*C3; clear HA HB HC;
130
131         LA=sparse(1:n-1,1:n-1,[u0t(i); ft(1:n-2)]  , n-1,n-1,n-1);
132         LB=sparse(1:n-1,1:n-1,ft , n-1,n-1,n-1);
133         LC=sparse(1:n-1,1:n-1,[ft(2:n-1); 0]  , n-1,n-1,n-1);
134         N2=LA*C1+LB*C2+LC*C3; clear LA LB LC;
135
136         fguess=(dt^3*K+dt*2/c^2*M-2.5*delta/c^4*M-N1*dt^2-N2*dt)\(g1(:,i)*dt^3+...
137             dt/c^2*5*M*U1(:,i-1)-dt/c^2*4*M*U1(:,i-2)+dt/c^2*M*U1(:,i-3)-...
138             9*delta/c^4*M*U1(:,i-1)+12*delta/c^4*M*U1(:,i-2)-...
139             7*delta/c^4*M*U1(:,i-3)+1.5*delta/c^4*M*U1(:,i-4)-...
140             N1*U1(:,i-1)*dt^2+dt*N2*(-2*U1(:,i-1)+U1(:,i-2)));
141
142     end
143     U1(:,i)=fguess;
144 end
145
146 U1=[u0t;U1;zeros(1,k)];
147 w1=[u0t; W1;zeros(1,k)];
148
149 %2ND ORDER
150 N=2*n;
151 Mesh2=linspace(0,L,N+1)';
152 B=round(T/dt)+1;
153
154 %define matrices K,M and the C's
155 elmat=GenerateTopologyother(n);
156 K=sparse(N+1,N+1);
157
158 for i=1:2:length(elmat)
159     Ke(:,:)=GenerateElementK(Mesh2(i),Mesh2(i+1),Mesh2(i+2));
160     for j=1:3
161         for k=1:3
162             K(elmat(i,j),elmat(i,k))=K(elmat(i,j),elmat(i,k))+Ke(j,k);
163         end
164     end
165 end
166 k=GenerateElementK(Mesh2(1),Mesh2(2),Mesh2(3));
167 gK1=k(1,2);
168 gK2=k(1,3);
169 K=K(2:N,2:N);
170
171 M=sparse(N+1,N+1);
172 for i=1:2:length(elmat)
173     Me(:,:)=GenerateElementM(Mesh2(i),Mesh2(i+1),Mesh2(i+2));
174     for j=1:3
175         for k=1:3
176             M(elmat(i,j),elmat(i,k))=M(elmat(i,j),elmat(i,k))+Me(j,k);
177         end
178
179     end
180 end
181
182 m=GenerateElementM(Mesh2(1),Mesh2(2),Mesh2(3));
183 gM1=m(1,2);
184 gM2=m(1,3);
185 M=M(2:N,2:N);
186
187 C1=sparse(N+1,N+1);
188 C2=sparse(N+1,N+1);
189 C3=sparse(N+1,N+1);
```

```
C4=sparse(N+1,N+1);
C5=sparse(N+1,N+1);
for i=1:2:length(elmat)-2
    [Ce1(:,:), Ce2(:,:), Ce3(:,:), Ce4(:,:), Ce5(:,:) ...
        ]=GenerateElementCtry(Mesh2(i),Mesh2(i+1),Mesh2(i+2), Mesh2(i+3), ...
        Mesh2(i+4) , beta, rho, c);
    for j=1:3
        for k=1:3
            C1(elmat(i,j),elmat(i,k))=C1(elmat(i,j),elmat(i,k))+Ce1(j,k);
            C2(elmat(i,j),elmat(i,k))=C2(elmat(i,j),elmat(i,k))+Ce2(j,k);
            C3(elmat(i,j),elmat(i,k))=C3(elmat(i,j),elmat(i,k))+Ce3(j,k);
            C4(elmat(i,j),elmat(i,k))=C4(elmat(i,j),elmat(i,k))+Ce4(j,k);
            C5(elmat(i,j),elmat(i,k))=C5(elmat(i,j),elmat(i,k))+Ce5(j,k);
        end
    end
end
for i=length(elmat)
    [Ce1(:,:), Ce2(:,:), Ce3(:,:), Ce4(:,:), Ce5(:,:) ...
        ]=GenerateElementC(Mesh2(i),Mesh2(i+1),Mesh2(i+2), beta, rho, c);
    for j=1:3
        for k=1:3
            C1(elmat(i,j),elmat(i,k))=C1(elmat(i,j),elmat(i,k))+Ce1(j,k);
            C2(elmat(i,j),elmat(i,k))=C2(elmat(i,j),elmat(i,k))+Ce2(j,k);
            C3(elmat(i,j),elmat(i,k))=C3(elmat(i,j),elmat(i,k))+Ce3(j,k);
            C4(elmat(i,j),elmat(i,k))=C4(elmat(i,j),elmat(i,k))+Ce4(j,k);
            C5(elmat(i,j),elmat(i,k))=C5(elmat(i,j),elmat(i,k))+Ce5(j,k);
        end
    end
end

gC12=C1(1,3); gC22=C2(1,3);

C1=C1(2:N,2:N);
C2=C2(2:N,2:N);
C3=C3(2:N,2:N);
C4=C4(2:N,2:N);
C5=C5(2:N,2:N);

%BC
u0t = Pulse(t, P0, f0);
u0t_tt=Pulsett(t,P0,f0);
u0t_t=Pulset(t,P0,f0);
u0t_ttt=[diff(u0t_tt), 0];

g2=sparse(N-1,B);
g2(1,:)=(-u0t*gK1-u0t_tt*gM1/c^2+u0t_t.*(u0t_t*gC22))+u0t_ttt*gM1*delta/c^4;
g2(2,:)=(-u0t*gK2-u0t_tt*gM2/c^2+u0t_tt.*(u0t*gC12))+u0t_ttt*gM2*delta/c^4;

gzC2=sparse(N-1,B);
gzC2(1,:)=(-u0t'*gK1-u0t_tt'*gM1)';
gzC2(2,:)=(-u0t'*gK2-u0t_tt'*gM2)';

%IC
u1=zeros(N-1,1);
ut1=zeros(N-1,1);
u2=u1+dt.*ut1;

% time solver, linear
W2=zeros(N-1,B);
W2(:,1)=u1;
W2(:,2)=u2;

W2(:,3)=(dt^3*K+dt/c^2*M-delta/c^4*M)\(gzC2(:,3)*dt^3+2*dt/c^2*M*W2(:,2)-...
    dt/c^2*M*W2(:,1)+3*delta/c^4*W2(:,1)-3*delta/c^4*W2(:,2));
W2(:,4)=(dt^3*K+dt*2/c^2*M-delta/c^4*M)\(gzC2(:,4)*dt^3+dt/c^2*5*M*W2(:,3)-...
    dt/c^2*4*M*W2(:,2)+dt/c^2*M*W2(:,1)-delta/c^4*W2(:,1)+3*delta/c^4*W2(:,2)-...
```

```
    3*delta/c^4*W2(:,3));
for i=5:round(T/dt)
    W2(:,i)=(dt^3*K+2*dt/c^2*M-2.5*delta/c^4*M)\(gzC2(:,i)*dt^3+...
        dt/c^2*5*M*W2(:,i-1)-dt/c^2*4*M*W2(:,i-2)+dt/c^2*M*W2(:,i-3)-...
        9*delta/c^4*M*W2(:,i-1)+12*delta/c^4*M*W2(:,i-2)-...
        7*delta/c^4*M*W2(:,i-3)+1.5*delta/c^4*M*W2(:,i-4));
end

%time solver, non-linear
G=zeros(N-1,B);
G(:,1)=u1;
G(:,2)=u2;
%
% now we solve for i=3
for i=3
    fguess=W2(:,i);
    ft=3*P0*ones(size(fguess));

    while (max(abs(ft-fguess))>10^-9*P0);
        ft=fguess;

        HA=sparse(1:N-1,1:N-1,([0; u0t(i); ft(1:N-3)]-[0;u0t(i-1); ...
            G(1:N-3,i-1)])/dt ,N-1,N-1,N-1);
        HB=sparse(1:N-1,1:N-1,([u0t(i); ft(1:N-2)]-[u0t(i-1); G(1:N-2,i-1)])/dt ...
            ,N-1,N-1,N-1);
        HC=sparse(1:N-1,1:N-1,(ft-G(:,i-1))/dt  , N-1,N-1,N-1);
        HD=sparse(1:N-1,1:N-1,([ft(2:N-1);0]-[G(2:N-1,i-1);0])/dt  ,N-1,N-1,N-1);
        HE=sparse(1:N-1,1:N-1,([ft(3:N-1);0;0]-[G(3:N-1,i-1);0;0])/dt  ,N-1,N-1,N-1);
        G1=HA*C1+HB*C2+HC*C3+HD*C4+HE*C5; clear HA HB HC HD HE;

        LA=sparse(1:N-1,1:N-1,[0;u0t(i); ft(1:N-3)]  , N-1,N-1,N-1);
        LB=sparse(1:N-1,1:N-1,[u0t(i); ft(1:N-2)]  , N-1,N-1,N-1);
        LC=sparse(1:N-1,1:N-1,ft , N-1,N-1,N-1);
        LD=sparse(1:N-1,1:N-1,[ft(2:N-1);0]  , N-1,N-1,N-1);
        LE=sparse(1:N-1,1:N-1,[ft(3:N-1);0; 0]  , N-1,N-1,N-1);
        G2=LA*C1+LB*C2+LC*C3+LD*C4+LE*C5; clear LA LB LC LD LE;

        fguess=(dt^3*K+dt/c^3*M-delta/c^4*M +G1*dt^2-G2*dt)\(g2(:,i)*dt^3+...
            2*dt/c^2*M*G(:,i-1)-dt/c^2*M*G(:,i-2)-G1*G(:,i-1)*dt^2+...
            G2*dt*(-2*G(:,i-1)+G(:,i-2))+3*delta/c^4*G(:,i-2)-3*delta/c^4*G(:,i-1));

    end
    G(:,i)=fguess;
end

for i=4
    fguess=W2(:,i);
    ft=3*P0*ones(size(fguess));

    while (max(abs(ft-fguess))>10^-9*P0);
        ft=fguess;

        HA=sparse(1:N-1,1:N-1,([0; u0t(i); ft(1:N-3)]-[0;u0t(i-1); ...
            G(1:N-3,i-1)])/dt ,N-1,N-1,N-1);
        HB=sparse(1:N-1,1:N-1,([u0t(i); ft(1:N-2)]-[u0t(i-1); G(1:N-2,i-1)])/dt ...
            ,N-1,N-1,N-1);
        HC=sparse(1:N-1,1:N-1,(ft-G(:,i-1))/dt  , N-1,N-1,N-1);
        HD=sparse(1:N-1,1:N-1,([ft(2:N-1);0]-[G(2:N-1,i-1);0])/dt  ,N-1,N-1,N-1);
        HE=sparse(1:N-1,1:N-1,([ft(3:N-1);0;0]-[G(3:N-1,i-1);0;0])/dt  ,N-1,N-1,N-1);
        G1=HA*C1+HB*C2+HC*C3+HD*C4+HE*C5; clear HA HB HC HD HE;

        LA=sparse(1:N-1,1:N-1,[0;u0t(i); ft(1:N-3)]  , N-1,N-1,N-1);
        LB=sparse(1:N-1,1:N-1,[u0t(i); ft(1:N-2)]  , N-1,N-1,N-1);
        LC=sparse(1:N-1,1:N-1,ft , N-1,N-1,N-1);
        LD=sparse(1:N-1,1:N-1,[ft(2:N-1);0]  , N-1,N-1,N-1);
        LE=sparse(1:N-1,1:N-1,[ft(3:N-1);0; 0]  , N-1,N-1,N-1);
```

```
315         G2=LA*C1+LB*C2+LC*C3+LD*C4+LE*C5; clear LA LB LC LD LE;
316
317
318         fguess=(dt^3*K+2*dt/c^2*M-delta/c^4*M-G1*dt^2-G2*dt)\(g2(:,i)*dt^3+...
319             5*dt/c^2*M*G(:,i-1)-dt/c^2*4*M*G(:,i-2)+dt/c^2*M*G(:,i-3)-...
320             G1*G(:,i-1)*dt^2+G2*dt*(-2*G(:,i-1)+G(:,i-2))-delta/c^4*G(:,i-3)+...
321             3*delta/c^4*G(:,i-2)-3*delta/c^4*G(:,i-1));
322
323     end
324     G(:,i)=fguess;
325 end
326
327 for i=5:B
328     fguess=W2(:,i);
329     ft=3*P0*ones(size(fguess));
330
331     while (max(abs(ft-fguess))>10^-6*P0);
332         ft=fguess;
333
334         HA=sparse(1:N-1,1:N-1,([0; u0t(i); ft(1:N-3)]-[0;u0t(i-1); ...
                G(1:N-3,i-1)])/dt ,N-1,N-1,N-1);
335         HB=sparse(1:N-1,1:N-1,([u0t(i); ft(1:N-2)]-[u0t(i-1); G(1:N-2,i-1)])/dt ...
                ,N-1,N-1,N-1);
336         HC=sparse(1:N-1,1:N-1,(ft-G(:,i-1))/dt  , N-1,N-1,N-1);
337         HD=sparse(1:N-1,1:N-1,([ft(2:N-1);0]-[G(2:N-1,i-1);0])/dt  ,N-1,N-1,N-1);
338         HE=sparse(1:N-1,1:N-1,([ft(3:N-1);0;0]-[G(3:N-1,i-1);0;0])/dt  ,N-1,N-1,N-1);
339         G1=HA*C1+HB*C2+HC*C3+HD*C4+HE*C5; clear HA HB HC HD HE;
340
341         LA=sparse(1:N-1,1:N-1,[0;u0t(i); ft(1:N-3)]  , N-1,N-1,N-1);
342         LB=sparse(1:N-1,1:N-1,[u0t(i); ft(1:N-2)]  , N-1,N-1,N-1);
343         LC=sparse(1:N-1,1:N-1,ft , N-1,N-1,N-1);
344         LD=sparse(1:N-1,1:N-1,[ft(2:N-1);0]  , N-1,N-1,N-1);
345         LE=sparse(1:N-1,1:N-1,[ft(3:N-1);0; 0]  , N-1,N-1,N-1);
346         G2=LA*C1+LB*C2+LC*C3+LD*C4+LE*C5; clear LA LB LC LD LE;
347
348
349         fguess=(dt^3*K+2*dt/c^2*M-3/2*G1*dt^2-2*dt*G2-...
350             2.5*delta/c^4*M)\(g2(:,i)*dt^3+(5*dt/c^2*M+G1*-4*dt^2/2-5*G2*dt-...
351             9*delta/c^4*M)*G(:,i-1)+(12*delta/c^4*M-4*dt/c^2*M+1/2*dt^2*G1+...
352             4*G2*dt)*G(:,i-2)+(-7*delta/c^4*M+dt/c^2*M-G2*dt)*G(:,i-3)+...
353             1.5*delta/c^4*M*G(:,i-4));
354
355     end
356     G(:,i)=fguess;
357 end
358
359 U2=[u0t;G;zeros(1,B)];
360 w2=[u0t; W2;zeros(1,B)];
361
362 figure(1)
363 time=round(xsh*T/L*1/dt + 6/f0/dt);
364 plot(Mesh/xsh,U1(:,time),'r--')
365 axis([0.9 1.1 -1.2*10^6 1.2*10^6])
366 hold on
367 plot(Mesh2/xsh,U2(:,time),'k')
368 xlabel('xshock')
369 ylabel('P')
370 strtime=num2str(time*dt*10^6);
371 title(strcat('t = ', strtime, 'ms'));
372 legend('U1','U2')
373
374 figure(2)
375 for l=1:length(Mesh)-1
376     difU(l,:)=(U2(2*l,:)+U2(2*l-1,:))/2-U1(l,:);
377 end
378 subplot(2,2,1)
```

```
plot(Mesh/xsh,[0; difU(:,round(time/4))])
axis([0 1.5 -3*10^5 3*10^5])
xlabel('xshock')
ylabel('P')
strtime=num2str(time/4*dt*10^6);
title(strcat('t = ', strtime, 'ms'));
subplot(2,2,2)
plot(Mesh/xsh,[0; difU(:,round(2*time/4))])
axis([0 1.5 -3*10^5 3*10^5])
strtime=num2str(time/2*dt*10^6);
title(strcat('t = ', strtime, 'ms'));
xlabel('xshock')
ylabel('P')
subplot(2,2,3)
plot(Mesh/xsh,[0; difU(:,round(3*time/4))])
axis([0 1.5 -3*10^5 3*10^5])
strtime=num2str(time*3/4*dt*10^6);
title(strcat('t = ', strtime, 'ms'));
xlabel('xshock')
ylabel('P')
subplot(2,2,4)
plot(Mesh/xsh,[0; difU(:,round(4*time/4))])
axis([0 1.5 -3*10^5 3*10^5])
strtime=num2str(time*dt*10^6);
title(strcat('t = ', strtime, 'ms'));
xlabel('xshock')
ylabel('P')
```

C.4 Custom function definitions called in the implementations

```
function [ M ] = Mmatrix( Mesh,n,c )
%assembles M matrix for 1st order FEM
dMesh=diff(Mesh);
M = spdiags([dMesh(2:n)  2*(dMesh(1:n-1)+dMesh(2:n))  ...
    dMesh(1:n-1)]/(6*c^2),-1:1,n-1,n-1);
end
```

```
function [ K ] = Kmatrix( Mesh,n )
%assembles K matrix for 1st order FEM
diffmesh=diff(Mesh);
K=sparse(n-1,n-1);
K=spdiags((1./diffmesh(2:n)+1./diffmesh(1:n-1)),0,K);
K=spdiags(-1./diffmesh(2:(n-1)),-1,K);
K=spdiags(-1./diffmesh(1:(n-1)),1,K);
end
```

```
function [ gM gK ] = vectorg( Mesh,n,c )
%elements of g; they multiply with the known BC at x=0
diffmesh=diff(Mesh);
gM=1/(6*c^2)*diffmesh(1);
gK=-1/diffmesh(1);
end
```

```
function [ elmat ] = GenerateTopologyother( n )
%generates topology
elmat=zeros(n,3);
for i=1:2:2*n-1
    elmat(i,1)=i;
    elmat(i,2)=i+1;
    elmat(i,3)=i+2;
end
```

```
function [ K ] = GenerateElementK( x1,x2, x3 )
%generates elements of the K matrix for 2nd order FEM
d3=1/6*(1/(x3-x2)+1/(x3-x1))^3*(x3-x2)*(x3-x1)-1/6*(x1-x2)^3/((x3-x2)^2*(x3-x1)^2);
d1=-1/3*(x1-x3)*(4*x1^2-2*x1*(3*x2+x3)+3*x2^2+x3^2)/((x1-x3)*(x1-x2))^2;
d2=1/3*(x3-x1)^3/((x2-x1)^2*(x2-x3)^2);

a=-1/3*(x1-x3)*(x1^2+x1*(x3-3*x2)+3*x2^2-3*x2*x3+x3^2)/...
    ((x1-x2)*(x1-x3)*(x3-x2)*(x3-x1));
b=-1/3*(x1-x3)^3/((x1-x2)*(x1-x3)*(x2-x1)*(x2-x3));
K=[d1 b a;b d2 b; a b d3];
end
```

```
function [ M ] = GenerateElementM( x1,x2, x3 )
%generates elements of the M matrix for 2nd order FEM
d1=-(x1-x3)*(6*x1^2+3*x1*(x3-5*x2)+10*x2^2-5*x2*x3+x3^2)/(30*(x1-x2)^2);
d2=-(x1-x3)^5/(30*(x1-x2)^2*(x2-x3)^2);
d3=-(x1-x3)*(6*x3^2+3*x3*(x1-5*x2)+10*x2^2-5*x2*x1+x1^2)/(30*(x2-x3)^2);

a=(x1-x3)*(3*x1^2-10*x1*x2+4*x1*x3+10*x2^2-10*x2*x3+3*x3^2)/(60*(x1-x2)*(x2-x3));
b1=-(x1-x3)^3*(3*x1-5*x2+2*x3)/(60*(x1-x2)^2*(x2-x3));
b2=(x1-x3)^3*(3*x3-5*x2+2*x1)/(60*(x1-x2)*(x2-x3)^2);
M=[d1 b1 a;b1 d2 b2; a b2 d3];
```

```
end
```

```
function [ C1,C2,C3,gC1,gC2,gC3 ] = Cmatrix( Mesh, beta, rho, c, n )
%assembles C matrices for 1st order FEM
C1=sparse(n+1,n+1);
C2=sparse(n+1,n+1);
C3=sparse(n+1,n+1);

factor=2*beta/(rho*c^4);

for i=2:n
    C1(i,i)=factor*1/12*(Mesh(i)-Mesh(i-1));
    C2(i,i)=factor*1/4*(Mesh(i+1)-Mesh(i))+factor*1/4*(Mesh(i)-Mesh(i-1));
    C3(i,i)=factor*1/12*(Mesh(i+1)-Mesh(i));
end
for i=1:n
    C1(i,i+1)=factor*1/12*(Mesh(i+1)-Mesh(i));
    C2(i,i+1)=factor*1/12*(Mesh(i+1)-Mesh(i));
    C3(i,i+1)=0;

    C1(i+1,i)=0;
    C2(i+1,i)=factor*1/12*(Mesh(i+1)-Mesh(i));
    C3(i+1,i)=factor*1/12*(Mesh(i+1)-Mesh(i));
end

gC1=C1(1,2);
gC2=C2(1,2);
gC3=C3(1,2);

C1=C1(2:n,2:n);
C2=C2(2:n, 2:n);
C3=C3(2:n, 2:n);
end
```

```
function [ M ] = Mmatrixd( Mesh,n)
%assembles M matrix for 1st order FEM without 1/c^2
dMesh=diff(Mesh);
M=sparse(n-1,n-1);
M = spdiags([dMesh(2:n)  2*(dMesh(1:n-1)+dMesh(2:n))  dMesh(1:n-1)]/(6),-1:1,n-1,n-1);
end
```

```
function [ C1,C2,C3,C4,C5 ] = GenerateElementCtry( a,b,c, d, e,beta, rho, vel )
%Generates elements of C matrices for second order FEM
factor=2*beta/(rho*vel^4);
%C1
A1=-1/(420*(a-b)*(b-c)^2)*(a-c)*(4*a^3+a^2*(9*c-21*b)+6*a*(7*b^2-7*b*c+2*c^2)-...
    35*b^3+63*b^2*c-42*b*c^2+10*c^3);
B1=0;
R1=1/(420*(c-d)^2*(d-e))*(c-e)*(10*c^3-6*c^2*(-2*e+7*d)+c*(63*d^2-42*e*d+9*e^2)-...
    35*d^3+42*d^2*e-21*d*e^2+4*e^3);
D1=0;
E1=0;
F1=(c-e)^3*(2*c^2-7*c*d+3*c*e+7*d^2-7*d*e+2*e^2)/(210*(c-d)^2*(d-e)^2);
G1=0;
H1=0;
I1=0;
C1=factor*[A1 B1 R1; D1 E1 F1; G1 H1 I1];

%C2
A2=-(a-c)^3*(3*a^2-14*a*b+8*a*c+21*b^2-28*b*c+10*c^2)/(420*(a-b)*(b-c)^3);
B2=-(c-e)^3*(10*c^2-28*c*d+8*c*e+21*d^2-14*d*e+3*e^2)/(420*(c-d)^3*(d-e));
```

```
R2=(c-e)^3*(2*c^2-7*c*d+3*c*e+7*d^2-7*d*e+2*e^2)/(210*(c-d)^2*(d-e)^2);
D2=0;
E2=(c-e)^5*(4*c-7*d+3*e)/(420*(c-d)^3*(d-e)^2);
F2=(c-e)^5*(3*c-7*d+4*e)/(420*(c-d)^2*(d-e)^3);
G2=0;
H2=(c-e)^3*(2*c^2-7*c*d+3*c*e+7*d^2-7*d*e+2*e^2)/(210*(c-d)^2*(d-e)^2);
I2=0;
C2=factor*[A2 B2 R2; D2 E2 F2; G2 H2 I2];

%C3
A3=1/(140*(b-c)^3)*(a-c)*(a^3+a^2*(4*c-7*b)+a*(21*b^2-28*b*c+10*c^2)-...
    35*b^3+84*b^2*c-70*b*c^2+20*c^3)+1/(140*(c-d)^3)*(c-e)*(e^3+10*c^2*(e-7*d)+...
    4*c*(21*d^2-7*d*e+e^2)-35*d^3+21*d^2*e-7*e^2*d+20*c^3);
B3=E2;
R3=-1/(420*(c-d)*(d-e)^2)*(c-e)*(4*c^3+c^2*(9*e-21*d)+6*c*(7*d^2-7*d*e+2*e^2)-...
    35*d^3+63*d^2*e-42*d*e^2+10*e^3);
D3=-(c-e)^3*(10*c^2-28*c*d+8*c*e+21*d^2-14*d*e+3*e^2)/(430*(c-d)^3*(d-e));
E3=(c-e)^7/(140*(d-c)^3*(d-e)^3);
F3=-(c-e)^3*(3*c^2-14*c*d+8*c*e+21*d^2-28*d*e+10*e^2)/(430*(c-d)*(d-e)^3);
G3=R1;
H3=F2;
I3=0;
C3=factor*[A3 B3 R3; D3 E3 F3; G3 H3 I3];

%C4
A4=B2; B4=R2; R4=0; D4=E2; E4=F2; F4=0; G4=R2; H4=F3; I4=0;

C4=factor*[A4 B4 R4; D4 E4 F4; G4 H4 I4];

%C5
A5=G3; B5=0; R5=0; D5=R2; E5=0; F5=0; G5=R3; H5=0; I5=0;

C5=factor*[A5 B5 R5; D5 E5 F5; G5 H5 I5];
end
```

```
function [ s ] = Pulse( t, P0, f0 )
% Function that evaluates the source function.
tw = 3/f0;
td = 6/f0;
s = P0.*exp(-((t-td)./(tw/2)).^2).*sin(2.*pi.*f0.*(t - td));

% The step function, setting all values greater than Tend identically to
% zero.
i=find(t>12/f0,1,'first');
s(i:end) = 0;
end
```

```
function [ s ] = Pulset( t, P0, f0 )
% Function that evaluates the second time derivative of the source
% function.
tw = 3./f0;
td = 6./f0;
s = 2.*pi.*P0.*exp(-4.*(t-td).^2./tw.^2).*cos(2.*pi.*f0.*(t-td))-8.*P0.*...
    (t-td).*exp(-4.*(t-td).^2./tw.^2).*sin(2.*pi.*f0.*(t-td))./tw^2

i=find(t>12./f0,1,'first');
s(i:end) = 0;
end
```

```
function [ s ] = Pulsett( t, P0, f0 )
% Function that evaluates the second time derivative of the source
```

```
% function.
tw = 3./f0;
td = 6./f0;
s = ...
    -8.*P0.*exp(-4.*(t-td).^2./tw.^2).*sin(2.*pi.*f0.*(t-td))./tw.^2 + ...
    64.*P0.*(t-td).^2.*exp(-4.*(t-td).^2./tw.^2).*sin(2.*pi.*f0.*(t-td))./tw.^4 - ...
    32.*P0.*(t-td).*exp(-4.*(t-td).^2./tw.^2).*...
        cos(2.*pi.*f0.*(t-td)).*pi.*f0./tw.^2 - ...
    4.*P0.*exp(-4.*(t-td).^2./tw.^2).*sin(2.*pi.*f0.*(t-td)).*pi.^2.*f0.^2;

i=find(t>12./f0,1,'first');
s(i:end) = 0;
end
```

```
function s = linearpropagation(f0,P0,c,x,t)
    % Function that solve the linear wave equations analytically.
    [T,X] = meshgrid(t,x);

    tw = 3/f0;
    td = 6/f0;
    s = P0.*exp(-(((T-X/c)-td)./(tw/2)).^2).*sin(2.*pi.*f0.*((T-X/c) - td));
end
```

References

[1] Woo, J. S. K., 2001. *A short History of the development of 3-D Ultrasound in Obstetrics and Gynecology*, http://www.ob-ultrasound.net/history-3D.html, A short History of the developments of Ultrasound in Obstetrics and Gynecology

[2] Paefgen, V., Doleschel, D. and Kiessling, F., 2015. *Evolution of contrast agents for ultrasound imaging and ultrasound-mediated drug delivery*, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4584939/, PMC US National Library of Medicine

[3] Hamilton, M. F. and Morfey, C.L., 1998. *Model Equations* in *Nonlinear Acoustics* edited by Hamilton, M. F. and Blackstock, D. T., Chap. 3, p. 41-63, Academic Press.

[4] Blackstock, D. T., Hamilton, M. F. and Pierce, A.D., 1998. *Progressive Waves in Lossless and Lossy fluids* in *Nonlinear Acoustics* edited by Hamilton, M. F. and Blackstock, D. T., Chap. 4, p. 66-147, Academic Press.

[5] Lauterborn, W., Kurz, T. and Akhatov, I., 2007. *Nonlinear Acoustics in Fluids* in *Handbook of Acoustics*, Chap. 8, p. 257-297, Springer.

[6] V.W. Sparrow and R. Raspet, 1991. *A numerical method for general finite amplitude wave propagation in two dimensions and its application to spark pulses*, J. Acoust. Soc. Am. 90 , no. 5, 26832691.

[7] Solovchuk, M. and Sheu, T. W. H., 2013. *Simulation of nonlinear Westervelt equation for the investigation of acoustic streaming and nonlinear propagation effects* The Journal of the Acoustical Society of America, Vol. 134, No. 5, p. 3931-3942

[8] Jing, Y., Tao, M. and Clement, G. T., 2011. *Evaluation of a wave-vector-frequency-domain method for nonlinear wave propagation*, The Journal of the Acoustical Society of America

[9] Dirkse, B., 2014. *Finite Element Method Appliet to the One-dimensionel Westervelt Equation*, Delft University of Technology, Faculty of Applied Sciences & Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology

[10] Huijssen, J., 2008. *Modeling of Nonlinear Medical Diagnostic Ultrasound*, PrintPartners Ipskamp B.V., Enschede

[11] Blackstock, D. T., 1966. *Connection between the Fay and Fubini Solutions for Plane Sound Waves of Finite Amplitude*, The journal of the Acoustical Society of America, Vol. 39, No. 6, p. 1019-1026

[12] Riley, L. A., 2006. *Reflection, Refraction, and Transmission*, http://webpages.ursinus.edu/lriley/courses/p212/lectures/node19.html, PHYS212 : Classical and Quantum Mechanical Waves

[13] Lahaye, D. and Vermolen, F., 2011. *Building Virtual Models in Engineering: An Introduction to Finite Elements*, Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft Institute for Applied Mathematics

[14] Unknown Author, 2013. *Nonlinear Acoustics: Modeling of the 1D Westervelt Equation*, COMSOL Multiphysics Model guide, Model ID: 12783

[15] Unknown Author, 2012. *Courant-Friedrichs-Lewy condition*, https://www.cfd-online.com/Wiki/Courant-Friedrichs-Lewy-condition