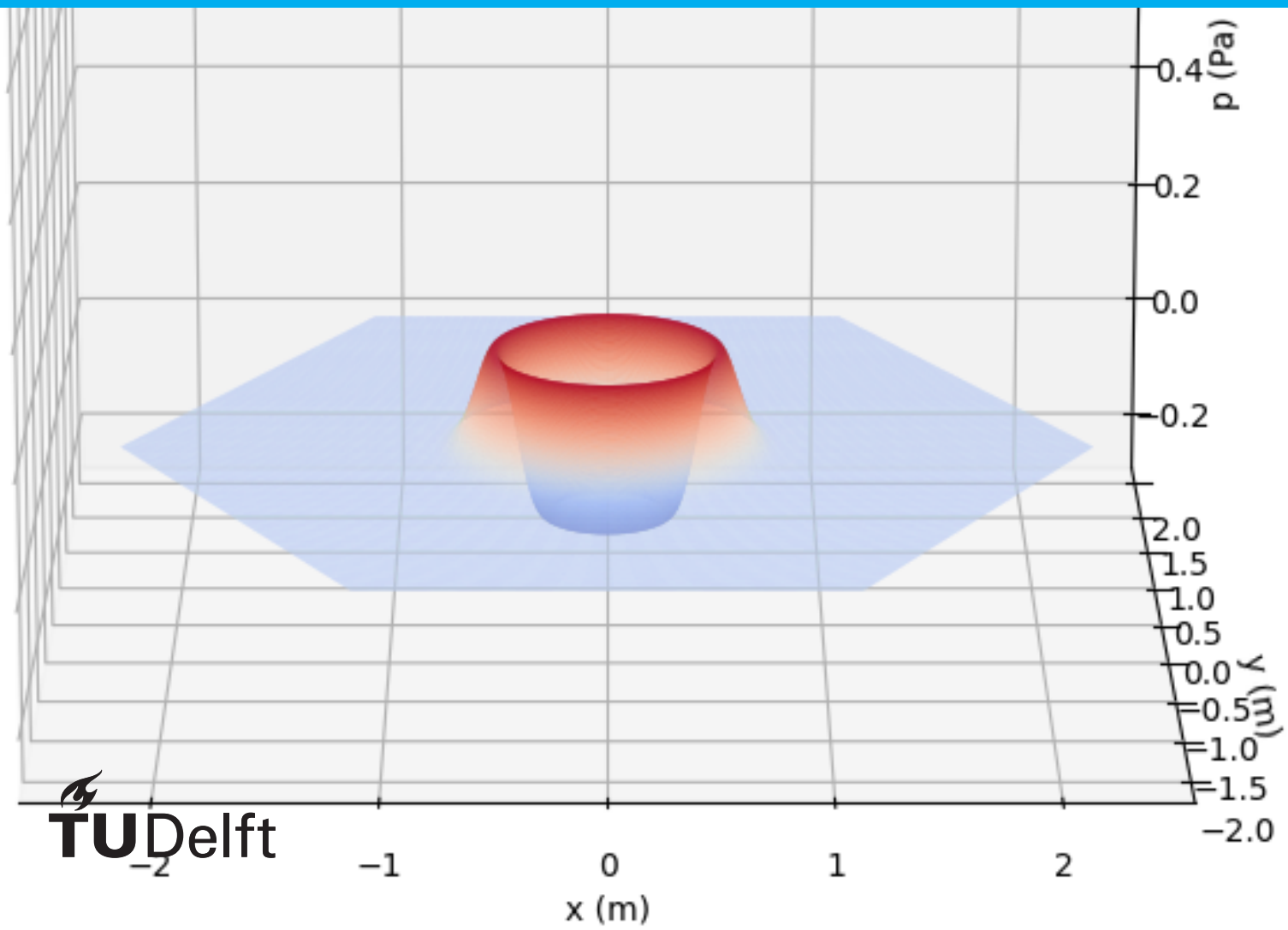
Finite Element Method for two dimensional Westervelt Equation

Leo Hendriks - 4605594

Technische Universiteit Delft
September 21, 2021
Under the supervision of
D.J.P. Lahaye and M. Verweij



TU Delft

Contents

1 Abstract 1

2 Introduction 2

3 Linear Wave Behaviour 3
- 3.1 1D Wave solution . . . 3
- 3.2 2D Wave Solution . . . 5
  - 3.2.1 Separation of Variables. . . . 5
  - 3.2.2 Green's Function . . . 7

4 Finite Element Method for linear Wave 12
- 4.1 Weak Form . . . 12
- 4.2 Space Discretization . . . 12
- 4.3 Time Differentiation . . . 13
- 4.4 Verlet Integration . . . 13
  - 4.4.1 Verlet for linear wave equation . . . 14

5 Finite Element Method for Non-linear Diffusion 15
- 5.1 Weak Form . . . 15
- 5.2 Space Discretization . . . 15
- 5.3 Iteration. . . . 16

6 Finite Element Method for Westervelt 17
- 6.1 Weak Form . . . 17
- 6.2 Space Discretization . . . 17
- 6.3 Verlet Integration . . . 18
- 6.4 Iteration. . . . 18

7 Implementation 20
- 7.1 Mesh Generation . . . 20
- 7.2 Base Functions . . . 20
- 7.3 Element-wise Computation . . . 21
- 7.4 Vectorized Implementation . . . 22
- 7.5 Boundary Conditions . . . 24

8 Results 25
- 8.1 One Dimensional Wave Example . . . 25
  - 8.1.1 Linear Solution. . . . 25
  - 8.1.2 Westervelt Equation . . . 27
- 8.2 Circular Symmetric Two Dimensional Wave Example . . . 28
  - 8.2.1 Linear Solution. . . . 29
  - 8.2.2 Westervelt Equation . . . 30
- 8.3 Non-linear Diffusion . . . 31
- 8.4 Mesh Size vs Error. . . . 33
- 8.5 Time Step vs Error. . . . 36

9 Conclusion 38

10 Discussion 39

Bibliography 41

A Appendix 42

1

Abstract

This essay shows a two dimensional implementation of the finite element method for the Westervelt equation. To do this the finite element method is first applied to the linear wave equation, then to non-linear diffusion and finally to the Westervelt equation. Both an element by element and a faster vectorized implementation are given for the finite element method. To verify the numerical solution two analytical solutions are used. The first is a one dimensional wave and the second a circularly symmetric wave.

We found that the two-dimensional implementation was successful in computing the Westervelt equation. The error of the solution has this relation with the mesh element size $\epsilon \propto lc^{1.7}$. It was also found that the time step used to compute the solution needs to be small enough for the implementation to converge.

2

Introduction

For most applications waves can be described by the general linear wave equation. Which can be solved analytically on simple domains or numerically approximated by using the finite difference method, finite element method or the integral equation method, when dealing with difficult domains. However, when dealing with high pressure or high frequency waves these approaches are no longer sufficient. This is due to the non-linear effects caused by the large and fast changes in pressure.

The non-linear effects are caused by both the distortion of the wave profile due to convection and the difference in compression which results in different propagations speeds [1]. From these two sources the second one is by far the most influential. There are various different equations that are used to describe non-linear wave equations. In this thesis we will be focusing on the Westervelt equation [2].

$$\nabla^2 u - \frac{1}{c_0^2}\frac{\partial^2 u}{\partial t^2} = -\frac{\beta}{\rho_0 c_0^4}\frac{\partial^2 u^2}{\partial t^2} \tag{2.1}$$

With u the pressure field, c_0 the small signal sound speed, ρ_0 the ambient density of mass and β the coefficient of non-linearity. The coefficient of non-linearity β is a dimensionless quantity that is inherent to the material that the wave is propagating through. This value is discussed in depth in both [1] and [2] though it is most often found empirically.

This equation is equal to the linear wave equation when using $\beta = 0$. It can be seen that the non-linear term depends on $\frac{\partial^2 u^2}{\partial t^2}$ which becomes more influential when the amplitude of u increases or the frequency of change in u increases. Thus we see that the expected behaviour of this equation agrees with the theory of non-linear waves.

There are no analytical solutions for the Westervelt equation, thus for solving non-linear wave problems one must rely on using numerical methods. The finite element method is an incredibly powerful tool to solve any partial differential equation on difficult domains. The finite element method can be used for a lot of different applications like calculating the heat diffusion in a chip or the air flow around a building. In the work of [3] and [4] it has been shown that the finite element method can be extended to solve the Westervelt equation in one dimension. The goal of this thesis is to further expand on their work to solve the Westervelt equation in two dimensions.

In Chapter 3 we will discuss two different analytical solutions to the linear wave equation that we will use to verify the FEM implementation. Then in the next chapter we will first apply the regular finite element method to solve the linear wave equation. To get familiar with non-linearities we will then use the finite element method to solve the non-linear diffusion equation in chapter 5. After this we will solve the Westervelt equation by using a finite element solution. The next chapter will discuss the implementation of the theory discussed in the previous chapter. We will be using both an element by element implementation and a faster vectorized implementation. Using these implementations we will show and discuss the results in chapter 8. Here we will also look at the relation between the error and the time step and element size. In chapter 9 the conclusion to this thesis can be found. Finally we will discuss the results in Chapter 10.

3

Linear Wave Behaviour

In this chapter we will discuss several analytical solutions to the wave equation that we can later use to verify our FEM implementation. In classical mechanics when trying to describe a simple wave a very powerful formula is the wave equation:

$$\frac{\partial^2 u}{\partial t^2} = c_0^2 \nabla^2 u \tag{3.1}$$

This equation can be used to describe all kind of waves, like light, water and sound waves. the c_0 in this formula is the speed of the wave and u is a function of time and space that shows the wave behaviour. This can for example be a pressure field or the perturbation of a string in a direction.

To verify our two dimensional FEM implementation from section 4 we will find two analytical solutions to the linear wave equation. The first is a one dimensional analytical solution with the same boundary and initial conditions as [4]. We can compare this solution to the FEM solution in a two dimensional channel with Neumann zero boundaries. Secondly, we will also compute a circular symmetric two dimensional solution for a Gaussian initial condition on the infinite domain.

3.1. 1D Wave solution

In 1D we will look at the semi-infinite domain with a time dependent boundary condition at zero as was done in [4]. Our objective is to solve the following problem:

$$\begin{cases} & \dfrac{\partial^2 u}{\partial t^2} = c_0^2 \dfrac{\partial^2 u}{\partial x^2} & & \text{(3.2a)} \\ u(x,0) = 0 & \dfrac{\partial u(x,0)}{\partial t} = 0 & 0 \le x < \infty & \text{(3.2b)} \\ & u(0,t) = h(t) & & \text{(3.2c)} \end{cases}$$

Here h(t) is any time dependent function. To solve this problem we will extend it to the infinite domain and use appropriate initial conditions to simulate the boundary at $x = 0$. The solution to our problem is then equal to this solution for all values $0 \le x < \infty$. This is the definition of the new problem.

$$\begin{cases} & \dfrac{\partial^2 u}{\partial t^2} = c_0^2 \dfrac{\partial^2 u}{\partial x^2} & & \text{(3.3a)} \\ u(x,0) = f(x) & \dfrac{\partial u(x,0)}{\partial t} = g(x) & -\infty < x < \infty & \text{(3.3b)} \end{cases}$$

The functions $f(x)$ and $g(x)$ are the initial conditions. The solution to this problem can be found by using characteristic coordinates. We now rewrite u as the sum of two new function, one that moves right with speed c_0 and one that moves left with the same speed.

$$u(x,t) = F(x - c_0 t) + G(x + c_0 t) = F(\zeta) + G(\eta) \tag{3.4}$$

This sum can be entered into our initial conditions.

$$\begin{cases} u(x,0) = F(x) + G(x) = f(x) & \text{(3.5a)} \\ \dfrac{\partial u(x,0)}{\partial t} = \dfrac{\partial}{\partial t}\Big[F(x-c_0 t) + G(x+c_0 t)\Big]\Big|_{t=0} = g(x) & \text{(3.5b)} \end{cases}$$

Using the chain rule we can rewrite 3.5b to the following. After applying the chain rule we evaluate the derivatives at $t = 0$.

$$\Big[\frac{\partial(x-c_0 t)}{\partial t}\frac{\partial F(x-c_0 t)}{\partial(x-c_0 t)} + \frac{\partial(x+c_0 t)}{\partial t}\frac{\partial G(x+c_0 t)}{\partial(x+c_0 t)}\Big]\Big|_{t=0} = g(x) \tag{3.6a}$$

$$-c_0\frac{\mathrm{d}F(x)}{\mathrm{d}x} + c_0\frac{\mathrm{d}G(x)}{\mathrm{d}x} = g(x) \tag{3.6b}$$

Now by rewriting 3.5a and combining that with 3.6b we can find an expression for $F(x)$ and $G(x)$.

$$g(x) = -c_0\frac{\mathrm{d}f}{\mathrm{d}x} + c_0\frac{\mathrm{d}G}{\mathrm{d}x} + c_0\frac{\mathrm{d}G}{\mathrm{d}x} \tag{3.7a}$$

$$2c_0\frac{\mathrm{d}G}{\mathrm{d}x} = c_0\frac{\mathrm{d}f}{\mathrm{d}x} + g \tag{3.7b}$$

$$G(x) = \frac{1}{2}f(x) + \frac{1}{2c_0}\int_0^x g(\tilde{x})\,d\tilde{x} + const \tag{3.7c}$$

$$F(x) = \frac{1}{2}f(x) - \frac{1}{2c_0}\int_0^x g(\tilde{x})\,d\tilde{x} - const \tag{3.7d}$$

By filling in this result in our initial proposition 3.4 we find our solution on an infinite domain, the constants can be neglected by adding 3.7c and 3.7d with proper shifts. This is the d'Alembert's Solution to the one dimensional wave problem.

$$u(x,t) = \frac{f(x-c_0 t) + f(x+c_0 t)}{2} + \frac{1}{2c_0}\int_{x-c_0 t}^{x+c_0 t} g(\tilde{x})\,d\tilde{x} \tag{3.8}$$

We will now return to the original problem 3.2. Our goal is extend the original problem to the infinite domain. Our previous derivation still holds for $0 < x < \infty$. We can thus apply 3.7c and 3.7d with the initial conditions 3.2b, in other words $f(x) = 0$ and $g(x) = 0$.

$$F(x) = 0 \qquad x > 0 \tag{3.9a}$$

$$G(x) = 0 \qquad x > 0 \tag{3.9b}$$

We will now solve our problem for two different cases $x > ct$ and $x < ct$. For the first case, when $x > ct > 0$ it follows that $x - ct > 0$ and $x + ct > 0$. We can use this in equation 3.4 to find the solution for this case.

$$u(x,t) = F(x-c_0 t) + G(x+c_0 t) = 0 \qquad \textit{for} \quad x > c_0 t \tag{3.10}$$

This result follows directly from 3.9a and 3.9b. We now move on to the second case where $x < ct$. We will now use the boundary condition at $x = 0$.

$$u(0,t) = F(-c_0 t) + G(c_0 t) = h(t) \qquad \textit{for} \quad t > 0 \tag{3.11}$$

We can substitute $z \equiv -c_0 t$ into this equation and rewrite it to $F(z)$. Note that $z < 0$, since $t > 0$, so this does not disagree with 3.9a.

$$F(z) = h\left(\frac{-z}{c_0}\right) - G(-z) \qquad \textit{for} \quad z < 0 \tag{3.12}$$

We now again use equation 3.4 to find our solution.

$$u(x,t) = F(x-c_0 t) + G(x+c_0 t) = h\left(t - \frac{x}{c_0}\right) - G(c_0 t - x) + G(x+c_0 t) \qquad \textit{for} \quad 0 \le x < c_0 t \tag{3.13}$$

We restrict our solution here to only $x \geq 0$, since that is the only interval we are interested in. From $0 \leq x < c_0 t$ it follows that $c_0 t - x > 0$ and $x + c_0 t > 0$. Using this we can simplify our result and write down our final solution to problem 3.2.

$$u(x,t) = \begin{cases} h\left(t - \frac{x}{c_0}\right) & 0 \leq x < c_0 t \\ 0 & x > c_0 t \end{cases} \tag{3.14}$$

This solution is the boundary condition moving to the right at speed c_0, which is exactly what one would expect for this problem.

3.2. 2D Wave Solution

In this section we will find an analytical solution to the 2D wave equation for a circularly symmetric case. We can use this solution to verify the FEM solution we will find in section 4. We wish to solve the following problem on the infinite domain:

$$\begin{cases} \dfrac{\partial^2 u}{\partial t^2} = c_0^2 \nabla^2 u & \text{(3.15a)} \\ u(r,\theta,0) = u(r,0) = Ae^{\frac{-r^2}{2\sigma^2}} \qquad \dfrac{\partial u(r,\theta,0)}{\partial t} = 0 & \text{(3.15b)} \end{cases}$$

This is the general linear wave equation with an initial displacement of a Gaussian peak.

3.2.1. Separation of Variables

To solve this problem we will use separation of variables. Also note that this problem is circularly symmetric, so the solution does not depend on θ.

$$u(r,\theta,t) = u(r,t) = \phi(r)h(t) \tag{3.16}$$

Filling this in into the partial differential equation and dividing by $c_0^2 \phi(r) h(t)$ gives us the separable solution. As a separation constant we will use $-\lambda$.

$$\frac{1}{c_0^2 h}\frac{\mathrm{d}^2 h}{\mathrm{d}t^2} = -\lambda = \frac{1}{\phi}\nabla^2 \phi \tag{3.17}$$

We can split this up into two ordinary differential equations. We can also fill in the Laplacian in polar coordinates. Since ϕ does not depend on θ, we will also immediately drop the second term of the Laplacian.

$$\frac{\mathrm{d}^2 h}{\mathrm{d}t^2} + \lambda c_0^2 h = 0 \tag{3.18a}$$

$$\frac{1}{r}\frac{\mathrm{d}}{\mathrm{d}r}\left(r\frac{\mathrm{d}\phi}{\mathrm{d}r}\right) + \lambda\phi = 0 \tag{3.18b}$$

The solution to the first ode is often seen in separation of variables and is equal to:

$$h(t) = Acos(c\sqrt{\lambda}t) + Bsin(c\sqrt{\lambda}t) = Acos(c_0\sqrt{\lambda}t) \tag{3.19}$$

The fact that $B = 0$ follows from the initial condition $\frac{\partial u(r,\theta,0)}{\partial t} = 0$.

To solve the second ordinary differential equation we need two boundary conditions on $\phi(r)$. Firstly, we note that the solution should be finite, thus $|\phi(0)| < \infty$. Secondly we will choose a sufficiently large R for which we will set $\phi(R) = 0$. This means that our solution will be valid up to a maximum time t_{max} where the initial displacement reaches the boundary. We will now rewrite the second ordinary differential equation slightly and substitute in $z = \sqrt{\lambda}r$.

$$r^2\frac{\mathrm{d}^2\phi}{\mathrm{d}r^2} + r\frac{\mathrm{d}\phi}{\mathrm{d}r} + \lambda r^2\phi = 0 \Rightarrow \tag{3.20a}$$

$$z^2\frac{\mathrm{d}^2\phi}{\mathrm{d}z^2} + z\frac{\mathrm{d}\phi}{\mathrm{d}z} + z^2\phi = 0 \tag{3.20b}$$

The solution to this ordinary differential equation are the Bessel functions of order 0.

$$\phi(r) = c_1 J_0(z) + c_2 Y_0(z) = c_1 J_0(\sqrt{\lambda} r) + c_2 Y_0(\sqrt{\lambda} r) \tag{3.21}$$

Since $Y_0(\sqrt{\lambda} r)$ is infinite for $r = 0$, it follows from the first boundary condition that $c_2 = 0$. From the second boundary condition we can find the eigenvalues of this problem.

$$J_0(\sqrt{\lambda} R) = 0 \qquad \Rightarrow \tag{3.22a}$$

$$\lambda_n = \left(\frac{z_n}{R}\right)^2 \tag{3.22b}$$

where z_n is the n$^{\text{th}}$ zero of J_0. We have now found an infinite set of solutions. We can now use the principle of superposition and fill in our two solutions to the ordinary differential equations to find the general solution to our problem (we combine the two arbitrary constants into one constant c_n).

$$u(r,t) = \sum_{n=1}^{\infty} c_n J_0(\sqrt{\lambda_n} r) cos(c_0 \sqrt{\lambda_n} r) \tag{3.23}$$

The last thing that is left is to compute the constants c_n. To do this we need to use the initial solution 3.15b. From now on it is more convenient to use z_n instead of λ_n.

$$Ae^{\frac{-r^2}{2\sigma^2}} = \sum_{n=1}^{\infty} c_n J_0\left(\frac{z_n}{R} r\right) \tag{3.24}$$

We can isolate one specific c_n by using the orthogonality of the Bessel functions with weight r. To do this, we multiply both sides by $r J_0\left(\frac{z_n}{R} r\right)$ and integrate from 0 to R.

$$c_n = \frac{\int_0^R Ae^{\frac{-r^2}{2\sigma^2}} J_0\left(\frac{z_n}{R} r\right) r \, dr}{\int_0^R J_0\left(\frac{z_n}{R} r\right)^2 r \, dr} \tag{3.25}$$

The denominator of this fraction can be calculated analytically. In [5] the following result is given (note that the Bessel functions are indeed orthogonal).

$$\int_0^1 J_i(\mu_n x) J_i(\mu_m x) x \, dx = \frac{\delta_{nm}}{2} J_{i+1}(\mu_n)^2 \tag{3.26}$$

where μ_n is equal to the n$^{\text{th}}$ zero of J_i. We can now find our integral by entering $i = 0$, substituting in $r = Rx$ and using that $\mu_n = z_n$ when $i = 0$.

$$\int_0^R J_0\left(z_n \frac{r}{R}\right)^2 \frac{r}{R} \frac{dr}{R} = \frac{1}{2} J_1(z_n)^2 \Rightarrow \tag{3.27a}$$

$$\int_0^R J_0\left(\frac{z_n}{R} r\right)^2 r \, dr = \frac{R^2}{2} J_1(z_n)^2 \tag{3.27b}$$

Using this result we can write down our final solution for our problem.

$$u(r,t) = \sum_{n=1}^{\infty} \frac{2}{R^2 J_1(z_n)^2} \int_0^R \left[Ae^{\frac{-r^2}{2\sigma^2}} J_0\left(\frac{z_n}{R} r\right) r \right] dr \, J_0\left(\frac{z_n}{R} r\right) cos\left(c_0 \frac{z_n}{R} r\right) \tag{3.28}$$

We will use a numerical approximation to compute this integral and choose a large enough N to approximate the sum.

3.2.2. Green's Function

As an alternative to separation of variables we can use the Green's function to solve the partial differential equation. The pro of this solution is that it is valid on the infinite domain.For a 2D wave the solution using Green's function is given by:

$$\begin{aligned} u(\vec{x},t) = &\int_0^t \iint G(\vec{x},t;\vec{x}_0,t_0)Q(\vec{x}_0,t_0)\,d^2\vec{x}_0\,dt_0 \\ &+ \iint \left[\frac{\partial u}{\partial t_0}(\vec{x}_0,0)G(\vec{x},t;\vec{x}_0,0) - u(\vec{x}_0,0)\frac{\partial G}{\partial t_0}(\vec{x},t;\vec{x}_0,0)\right] d^2\vec{x}_0 \\ &- c_0^2\int_0^t \left[\oint \left(u(\vec{x}_0,t_0)\nabla_{\vec{x}_0}G(\vec{x},t;\vec{x}_0,t_0) - G(\vec{x},t;\vec{x}_0,t_0)\nabla_{\vec{x}_0}u(\vec{x}_0,t_0)\right)\cdot\hat{n}\,dl\right] dt_0 \end{aligned} \tag{3.29}$$

This equation can be simplified for our problem. Using $Q(\vec{x},t)=0$, $\frac{\partial u(r,\theta,0)}{\partial t}=0$ and the fact that we are using the infinite domain we can reduce the equation to just:

$$u(\vec{x},t) = -\iint u(\vec{x}_0,0)\frac{\partial G}{\partial t_0}(\vec{x},t;\vec{x}_0,0)\,d^2x_0 \tag{3.30}$$

From [6] we know that the Green's function for a 2D wave is given by (with $r=|x-x_0|$):

$$G(\vec{x},t;\vec{x}_0,t_0) = G(r,t;t_0) = \begin{cases} 0 & r > c_0(t-t_0) \\ \dfrac{1}{2\pi c_0}\dfrac{1}{\sqrt{c_0^2(t-t_0)^2-r^2}} & r < c_0(t-t_0) \end{cases} \tag{3.31}$$

Using properties of the Green's function we can show that $\frac{\partial G}{\partial t_0} = -\frac{\partial G}{\partial t}$ from which we find:

$$u(\vec{x},t) = \frac{1}{2\pi c_0}\iint_{|\vec{x}-\vec{x}_0|<c_0t} u(\vec{x}_0,0)\frac{\partial}{\partial t}\frac{1}{\sqrt{c_0^2t^2-|\vec{x}-\vec{x}_0|^2}}\,d^2\vec{x}_0 \tag{3.32}$$

Now using $\vec{x}_1 \equiv \vec{x}_0 - \vec{x}$ we can rewrite this into an integral over $\vec{x}_1$:

$$u(\vec{x},t) = \frac{1}{2\pi c_0}\iint_{|\vec{x}_1|<c_0t} u(\vec{x}_1+\vec{x},0)\frac{\partial}{\partial t}\frac{1}{\sqrt{c_0^2t^2-|\vec{x}_1|^2}}\,d^2\vec{x}_1 \tag{3.33}$$

We can now simplify this equation by integrating using polar coordinates.

$$u(r,\theta,t) = \frac{1}{2\pi c_0}\int_0^{2\pi}\int_0^{c_0t} u(\sqrt{r_1^2+r^2+2r_1rcos(\theta_1-\theta)},0)\frac{\partial}{\partial t}\left(\frac{1}{\sqrt{c_0^2t^2-r_1^2}}\right) r_1\,dr_1\,d\theta_1 \tag{3.34}$$

Now lets see if we can simplify the time derivative:

$$G(r_1,t) = \frac{1}{2\pi c_0}\frac{1}{\sqrt{c_0^2t^2-r_1^2}} = \frac{1}{2\pi c_0 r_1}Q(\xi) \tag{3.35}$$

where $\xi = \dfrac{c_0t}{r_1}$ and where:

$$Q(\xi) = \frac{1}{\sqrt{\xi^2-1}} \tag{3.36}$$

Using Q we find the following result using the chain rule:

$$\frac{\partial Q}{\partial r_1} = \frac{\mathrm{d}Q}{\mathrm{d}\xi}\frac{\partial\xi}{\partial r_1} = -\frac{c_0t}{r_1^2}\frac{\mathrm{d}Q}{\mathrm{d}\xi} \Rightarrow \frac{\mathrm{d}Q}{\mathrm{d}\xi} = -\frac{r_1^2}{c_0t}\frac{\partial Q}{\partial r_1} \tag{3.37}$$

From this we can rewrite the integral from a time derivative to a derivative in r_1.

$$\frac{\partial G}{\partial t} = \frac{1}{2\pi c_0 r_1}\frac{\partial Q}{\partial t} = \frac{1}{2\pi c_0 r_1}\frac{\mathrm{d}Q}{\mathrm{d}\xi}\frac{\partial\xi}{\partial t} = \frac{1}{2\pi r_1^2}\frac{\mathrm{d}Q}{\mathrm{d}\xi} = -\frac{1}{2\pi c_0 t}\frac{\partial Q}{\partial r_1} \tag{3.38}$$

We can now enter this result in our integral to find the following equation.

$$u(r,\theta,t) = -\frac{1}{2\pi c_0 t}\int_0^{2\pi}\int_0^{c_0 t} u(\sqrt{r_1^2+r^2+2r_1 r cos(\theta_1-\theta)},0)r_1\frac{\partial}{\partial r_1}\frac{1}{\sqrt{\left(\frac{c_0 t}{r_1}\right)^2-1}}\,dr_1\,d\theta_1 \tag{3.39}$$

Integration by parts gives us:

$$u(r,\theta,t) = \frac{1}{2\pi c_0 t}\int_0^{2\pi}\int_0^{c_0 t}\frac{u(\sqrt{r_1^2+r^2+2r_1 r cos(\theta_1-\theta)},0)+r_1\frac{\partial u(\sqrt{r_1^2+r^2+2r_1 r cos(\theta_1-\theta)},0)}{\partial r_1}}{\sqrt{\left(\frac{c_0 t}{r_1}\right)^2-1}}\,dr_1\,d\theta_1 \tag{3.40}$$

Now filling in the original equation gives:

$$u(r,\theta,t) = \frac{1}{2\pi c_0 t}\int_0^{2\pi}\int_0^{c_0 t}\frac{\left(1-\frac{r_1^2}{\sigma^2}-\frac{r_1 r cos(\theta_1-\theta)}{\sigma^2}\right)e^{\frac{-r_1^2-r^2-2r_1 r cos(\theta_1-\theta)}{2\sigma^2}}}{\sqrt{\left(\frac{c_0 t}{r_1}\right)^2-1}}\,dr_1\,d\theta_1 \tag{3.41}$$

We can rewrite this into:

$$u(r,\theta,t) = \frac{1}{2\pi c_0 t}\int_0^{c_0 t}\left[\int_0^{2\pi}\frac{\left(1-\frac{r_1^2}{\sigma^2}\right)e^{-\frac{r_1 r}{\sigma^2}cos(\theta_1-\theta)}-\frac{r_1 r}{\sigma^2}cos(\theta_1-\theta)e^{-\frac{r_1 r}{\sigma^2}cos(\theta_1-\theta)}}{e^{\frac{r_1^2+r^2}{2\sigma^2}}\sqrt{\left(\frac{c_0 t}{r_1}\right)^2-1}}\,d\theta_1\right]dr_1 \tag{3.42}$$

From [5] we know that the following holds for the modified Bessel function:

$$I_\alpha(x) = \frac{1}{\pi}\int_0^\pi e^{xcos(\theta)}cos(\alpha\theta)\,d\theta - \frac{sin(\alpha\pi)}{\pi}\int_0^\infty e^{-xcosh(t-\alpha t)}\,dt \tag{3.43}$$

Using this we can find the following two results:

$$I_0(x) = \frac{1}{\pi}\int_0^\pi e^{xcos(\theta)}\,d\theta \tag{3.44a}$$

$$I_1(x) = \frac{1}{\pi}\int_0^\pi cos(\theta)e^{xcos(\theta)}\,d\theta \tag{3.44b}$$

By substituting in $\theta = \theta' + \pi$ and using $cos(\theta+\pi) = -cos(\theta)$ we can find the following result. We also use that $I_0(x)$ is an even function and $I_1(x)$ is odd.

$$\begin{aligned}\int_0^{2\pi} e^{xcos(\theta)}\,d\theta &= \int_0^\pi e^{xcos(\theta)}\,d\theta + \int_\pi^{2\pi} e^{xcos(\theta)}\,d\theta \\ &= \pi I_0(x) + \int_0^\pi e^{-xcos(\theta')}\,d\theta' = \pi I_0(x) + \pi I_0(-x) = 2\pi I_0(x)\end{aligned} \tag{3.45a}$$

$$\begin{aligned}\int_0^{2\pi} cos(\theta)e^{xcos(\theta)}\,d\theta &= \int_0^\pi cos(\theta)e^{xcos(\theta)}\,d\theta + \int_\pi^{2\pi} cos(\theta)e^{xcos(\theta)}\,d\theta \\ &= \pi I_1(x) + \int_0^\pi -cos(\theta')e^{-xcos(\theta')}\,d\theta' = \pi I_1(x) - \pi I_1(-x) = 2\pi I_1(x)\end{aligned} \tag{3.45b}$$

Now we substitute in $\theta' = \theta_1 - \theta$ and use the fact that $0 \leq \theta < 2\pi$.

$$\begin{aligned} 2\pi I_0(x) = \int_0^{2\pi} e^{xcos(\theta')}\, d\theta' &= \int_\theta^{2\pi+\theta} e^{xcos(\theta_1-\theta)}\, d\theta_1 \\ &= \int_{2\pi}^{2\pi+\theta} e^{xcos(\theta_1-\theta)}\, d\theta_1 + \int_\theta^{2\pi} e^{xcos(\theta_1-\theta)}\, d\theta_1 \\ &= \int_0^{\theta} e^{xcos(\theta_1-\theta)}\, d\theta_1 + \int_\theta^{2\pi} e^{xcos(\theta_1-\theta)}\, d\theta_1 \\ &= \int_0^{2\pi} e^{xcos(\theta_1-\theta)}\, d\theta_1 \end{aligned} \tag{3.46a}$$

$$\begin{aligned} 2\pi I_1(x) = \int_0^{2\pi} cos(\theta') e^{xcos(\theta')}\, d\theta' &= \int_\theta^{2\pi+\theta} cos(\theta_1-\theta) e^{xcos(\theta_1-\theta)}\, d\theta_1 \\ &= \int_{2\pi}^{2\pi+\theta} cos(\theta_1-\theta) e^{xcos(\theta_1-\theta)}\, d\theta_1 + \int_\theta^{2\pi} cos(\theta_1-\theta) e^{xcos(\theta_1-\theta)}\, d\theta_1 \\ &= \int_0^{\theta} cos(\theta_1-\theta) e^{xcos(\theta_1-\theta)}\, d\theta_1 + \int_\theta^{2\pi} cos(\theta_1-\theta) e^{xcos(\theta_1-\theta)}\, d\theta_1 \\ &= \int_0^{2\pi} cos(\theta_1-\theta) e^{xcos(\theta_1-\theta)}\, d\theta_1 \end{aligned} \tag{3.46b}$$

We can fill in the result of equation [3.46a] and [3.46b] into equation [3.42].

$$u(r,\theta,t) = \frac{1}{c_0 t} \int_0^{c_0 t} \frac{\left(1 - \frac{r_1^2}{\sigma^2}\right) I_0(\frac{r_1 r}{\sigma^2}) + \frac{r_1 r}{\sigma^2} I_1(\frac{r_1 r}{\sigma^2})}{e^{\frac{r_1^2 + r^2}{2\sigma^2}} \sqrt{\left(\frac{c_0 t}{r_1}\right)^2 - 1}}\, dr_1 \tag{3.47}$$

This can be rewritten to:

$$u(r,\theta,t) = \frac{1}{c_0 t} \int_0^{c_0 t} \frac{\partial}{\partial r_1} \left(\frac{r_1 I_0\left(\frac{r_1 r}{\sigma^2}\right)}{e^{\frac{r_1^2 + r^2}{2\sigma^2}}} \right) \frac{1}{\sqrt{\left(\frac{c_0 t}{r_1}\right)^2 - 1}}\, dr_1 \tag{3.48}$$

The equation 3.47 can be approximated numerically similar to the solution found by separation of variables. However there are two downsides to using this solution. Firstly the integral goes to infinity on the two integration boundaries, which makes it harder to approximate. Secondly, to use this solution you need to solve the integral for each value of r and t. For the separation of variables solution however you just need to numerically approximate the integral once to calculate the constants. After which you can compute the solution for any r or t values by computing a simple sum. This saves a lot of computation time. The pro of working on the infinite domain does not outweigh these downsides and that is why we will use the separation of variables solution in the rest of this thesis.

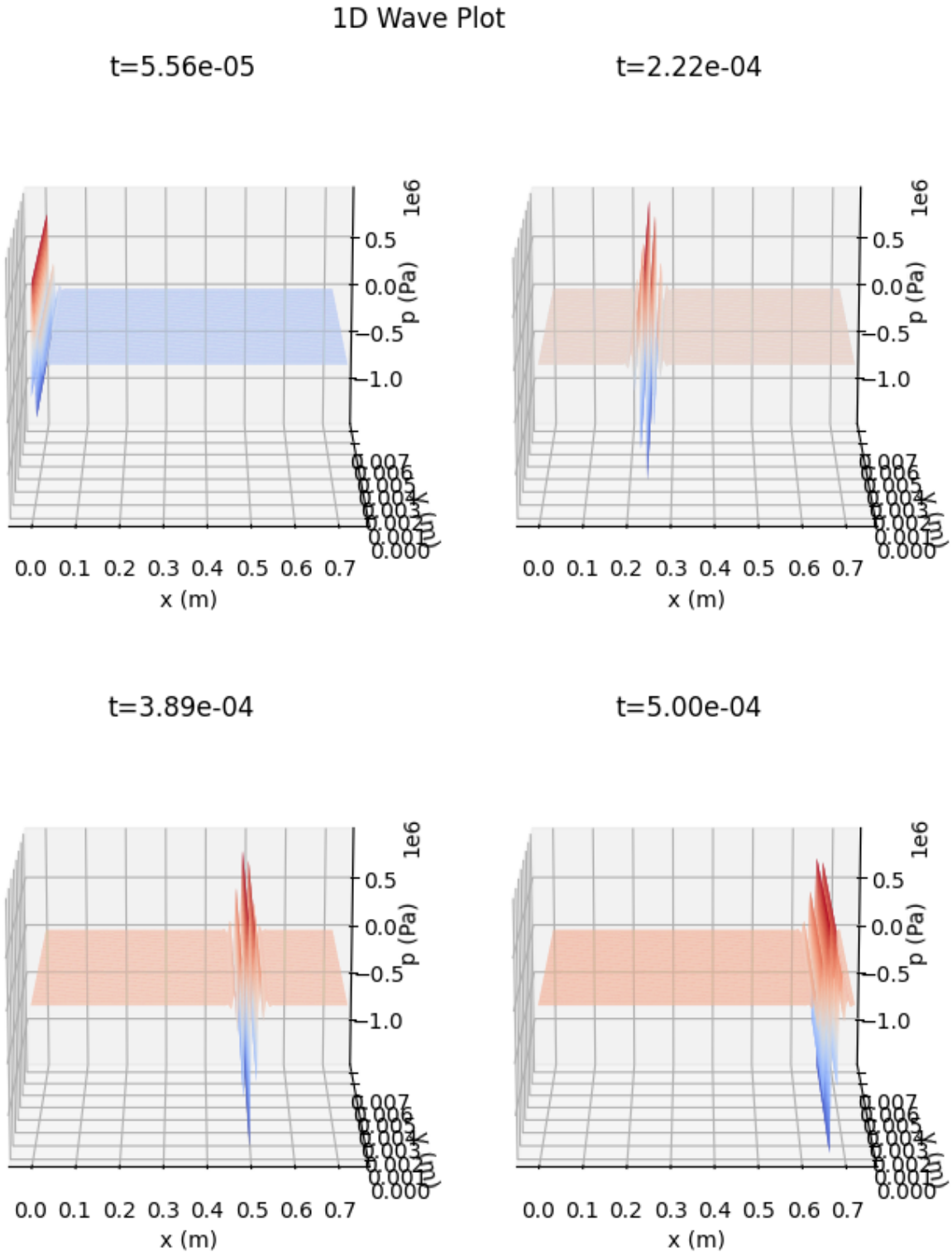


Figure 3.1: 2D plot of the finite element solution to the 1-dimensional problem. For more information on the parameters of the plot see table 8.1.

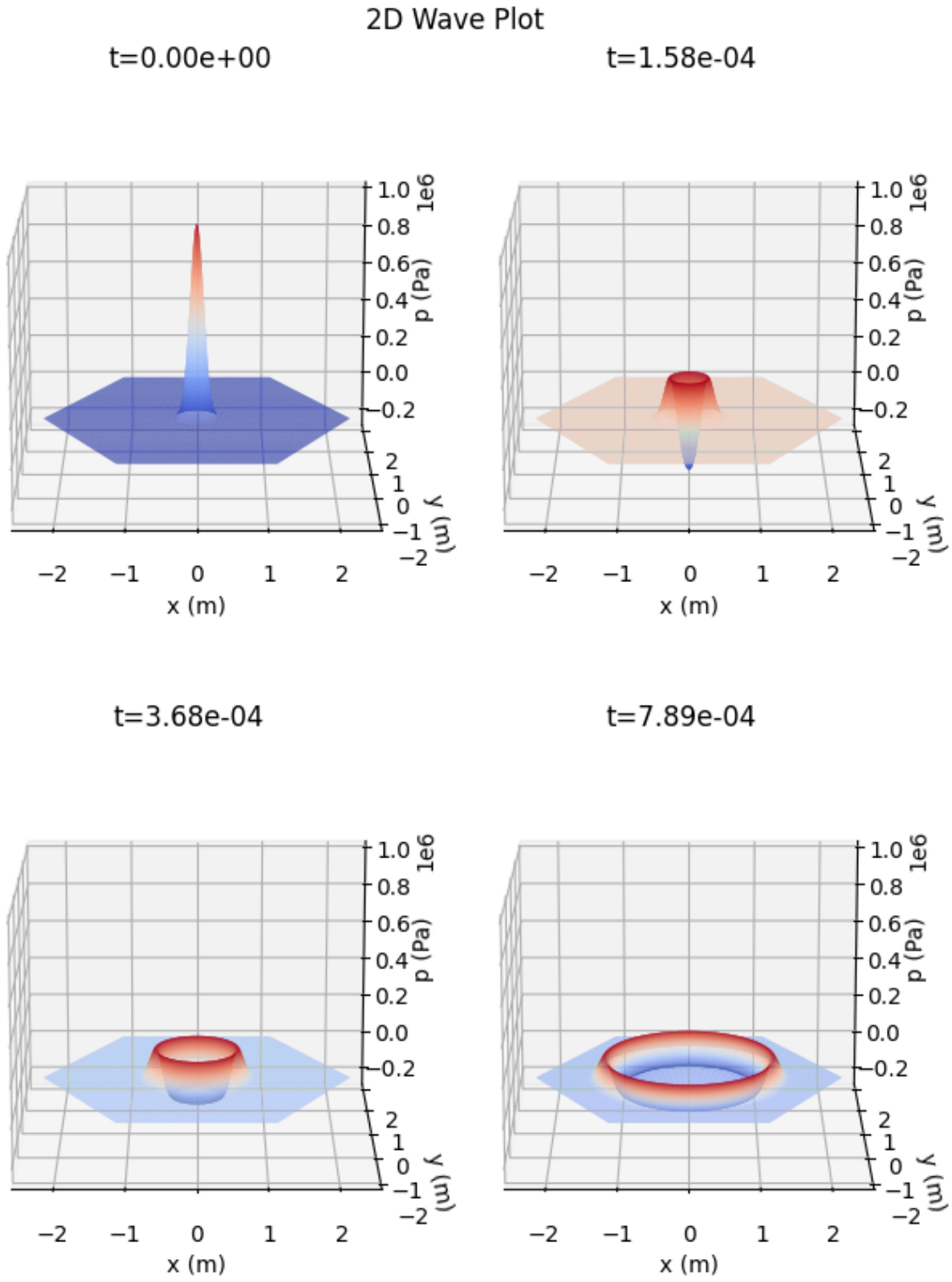


Figure 3.2: 2D plot of the finite element solution to the 2-dimensional circularly symmetric problem. For more information on the parameters of the plot see table 8.2.

4

Finite Element Method for linear Wave

In this chapter we will use the Galerkin's finite element method to solve the linear wave equation with the following boundary and initial conditions in 2 dimensions.

$$\begin{cases} \nabla^2 u - \dfrac{1}{c_0^2}\dfrac{\partial^2 u}{\partial t^2} = f \text{ on } \Omega & \text{(4.1a)} \\ u(x,y,0) = g(x,y) \qquad \dfrac{\partial u(x,y,0)}{\partial t} = h(x,y) & \text{(4.1b)} \\ u(x,y,t) = 0 \,\forall\, t \text{ on } \partial\Omega_D \qquad \dfrac{\partial u(x,y,t)}{\partial n} = \nabla u(x,y,t)\cdot\vec{n} = 0 \,\forall\, t \text{ on } \partial\Omega_N & \text{(4.1c)} \end{cases}$$

Furthermore it is given that $\partial\Omega_D \cup \partial\Omega_N = \partial\Omega$ and $\partial\Omega_D \cap \partial\Omega_N = \emptyset$. Furthermore f is an arbitrary source function that can depend on x, y and t.

4.1. Weak Form

Before we can use the finite element method on the wave equation we first need to convert it to its weak form. To do this we first multiply 4.1a by a test function $v(x,y)$ and then integrate over the domain.

$$\iint_\Omega v\nabla^2 u\,dx\,dy - \frac{1}{c_0^2}\iint_\Omega v\frac{\partial^2 u}{\partial t^2}\,dx\,dy = \iint_\Omega vf\,dx\,dy \tag{4.2}$$

For 4.2 and 4.1a to be equivalent we need to have 2 conditions on our test functions v, namely that the test functions need to be smooth and that $v(x,y,t) = 0$ on $\partial\Omega_D$.

To simplify the equation 4.2 we will make use of Green's first identity, which is based on the divergence theorem and $\nabla(v\nabla u) = v\nabla^2 u + \nabla v\nabla u$.

$$\iint_\Omega \left(v\nabla^2 u + \nabla v\nabla u\right)dx\,dy = \int_{\partial\Omega} v\frac{\partial u}{\partial n}\,ds \tag{4.3}$$

Because we know that $\frac{\partial u}{\partial n} = 0$ on $\partial\Omega_N$ and $v(x,y,z,t) = 0$ on $\partial\Omega_D$, we find that the right hand side is equal to zero.

Using equation 4.3 we can simplify the leftmost term of equation 4.2.

$$-\iint_\Omega \nabla v\nabla u\,dx\,dy - \frac{1}{c_0^2}\iint_\Omega v\frac{\partial^2 u}{\partial t^2}\,dx\,dy = \iint_\Omega vf\,dx\,dy \tag{4.4}$$

This is the weak form of the linear wave equation.

4.2. Space Discretization

The next step in our solution is to create a mesh of our domain consisting of a set of nodes $\vec{x_i} = (x_i, y_i)$ and a set of elements e_k. Let η denote the number of nodes. For each of the nodes we create a basis function $\phi_i(x,y)$ with $\phi_i(\vec{x} = \vec{x_i}) = 1$ and $\phi_i(\vec{x} = \vec{x_j}) = 0$ for $\forall\, j \neq i$.

Using these base functions we can approximate our solution u by $u^h(x,y,t) = \sum_{i=1}^{\eta} c_i(t)\phi_i(x,y)$ and let our test functions equal the basis functions. By entering $u \approx u^h$ and $v = \phi_j$ into equation 4.4 we find the following equation:

$$-\sum_{i=1}^{\eta}\left[\iint_\Omega \nabla\phi_i \nabla\phi_j \, dx\, dy\right] c_i - \frac{1}{c_0^2}\sum_{i=1}^{\eta}\left[\iint_\Omega \phi_i\phi_j \, dx\, dy\right]\frac{\partial^2 c_i}{\partial t^2} = \iint_\Omega \phi_j f \, dx\, dy \qquad \forall j \tag{4.5}$$

We can rewrite this system of ordinary differential equations using matrices.

$$-S\vec{c}(t) - \frac{1}{c_0^2} M \frac{\partial^2}{\partial t^2}\vec{c}(t) = \vec{f}(t) \tag{4.6}$$

This equation uses the following matrices and vectors:

$$S_{i,j} = \iint_\Omega \nabla\phi_i \nabla\phi_j \, dx\, dy \qquad \forall i,j \tag{4.7a}$$

$$M_{i,j} = \iint_\Omega \phi_i\phi_j \, dx\, dy \qquad \forall i,j \tag{4.7b}$$

$$\vec{c}_j(t) = c_j(t) \qquad \forall j \tag{4.7c}$$

$$\vec{f}_j(t) = \iint_\Omega \phi_j(x,y) f(x,y,t) \, dx\, dy \qquad \forall j \tag{4.7d}$$

$\vec{c}$ and $\vec{f}$ are $\eta \times 1$ vectors, since j goes from 1 to η. And S and M are $\eta \times \eta$ matrices, since i also ranges from 1 to η.

4.3. Time Differentiation

To solve this second order system of ordinary differential equations we first rewrite our problem to have the differential equations on the left side.

$$\frac{\partial^2}{\partial t^2}\vec{c}(t) = M^{-1}\left(-c_0^2 S\vec{c}(t) - c_0^2\vec{f}(t)\right) \tag{4.8}$$

This equation can be solved using a multitude of solvers for second order system of ordinary differential equations. It is also possible to solve this system by using the following system of first order equations, created using $\frac{\partial}{\partial t}\vec{c} = \sigma$:

$$M\sigma = M\frac{\partial \vec{c}}{\partial t} \tag{4.9a}$$

$$\frac{\partial \sigma}{\partial t} = M^{-1}\left(-c_0^2 S\vec{c} - c_0^2\vec{f}\right) \tag{4.9b}$$

In this project we will directly solve the second order system by using Verlet integration.

4.4. Verlet Integration

The Verlet method can be used to solve second order systems of ordinary differential equations. This method is typically used in physics for solving Newtonian problems. To use Verlet integration on a second order system of ordinary differential equation like 4.10a one needs two initial conditions.

$$\frac{\partial^2 \vec{x}}{\partial t^2} = \vec{p}(\vec{x}(t), t) \tag{4.10a}$$

$$\vec{x}(t_0) = \vec{x}_0 \tag{4.10b}$$

$$\frac{\partial \vec{x}}{\partial t}(t_0) = \vec{v}_0 \tag{4.10c}$$

One can now integrate over time by choosing a step size Δt. The smaller the step size chosen, the smaller the error and the longer the computation time. We will use the following notation in the future:

$$t^\ell = t_0 + \ell * \Delta t \tag{4.11a}$$
$$\vec{x}^\ell = \vec{x}(t^\ell) \tag{4.11b}$$

Using this notation we can write down the formula for Verlet integration.

$$\vec{x}^0 = \vec{x}_0 \tag{4.12a}$$
$$\vec{x}^1 = \vec{x}_0 + \Delta t \vec{v}_0 + \frac{1}{2}\Delta t^2 \vec{p}(\vec{x}_0, t_0) \tag{4.12b}$$
$$\vec{x}^\ell = 2\vec{x}^{\ell-1} - \vec{x}^{\ell-2} + \Delta t^2 \vec{p}(\vec{x}^{\ell-1}, t^{\ell-1}) \tag{4.12c}$$

Using Verlet integration we can calculate the value of $\vec{x}(t)$ for any t after t_0.

4.4.1. Verlet for linear wave equation

For the linear wave equation we have rewritten our problem into a second order system of ordinary differential equations that we can solve using Verlet integration, since the two necessary initial conditions follow from 4.1b.

$$\vec{c}_i(0) = u(x_i, y_i, 0) = g(x_i, y_i) \qquad \forall i \tag{4.13a}$$
$$\frac{\partial}{\partial t}\vec{c}_i(0) = \frac{\partial}{\partial t}u(x_i, y_i, 0) = h(x_i, y_i) \qquad \forall i \tag{4.13b}$$

We now find the finite element solution to the linear wave equation (using $\vec{c}^\ell = \vec{c}(t^\ell)$) by using these initial conditions and using equation 4.8.

$$\vec{c}^0 = \vec{c}(0) \tag{4.14a}$$
$$\vec{c}^1 = \vec{c}(0) + \Delta t \frac{\partial}{\partial t}\vec{c}(0) + \frac{1}{2}\Delta t^2 M^{-1}\left(-c_0^2 S\vec{c}(0) - c_0^2 \vec{f}(t_0)\right) \tag{4.14b}$$
$$\vec{c}^\ell = 2\vec{c}^{\ell-1} - \vec{c}^{\ell-2} + \Delta t^2 M^{-1}\left(-c_0^2 S\vec{c}^{\ell-1} - c_0^2 \vec{f}(t^{\ell-1})\right) \tag{4.14c}$$

5

Finite Element Method for Non-linear Diffusion

Before solving the Westervelt equation we will solve the Non-linear Diffusion equation as an intermediate step. In this chapter we will use an iteration procedure combined with the finite element method to solve the non-linear diffusion equation in 2 dimensions:

$$\begin{cases} \nabla^2 u + \alpha u^2 = f \text{ on } \Omega & \text{(5.1a)} \\ u(x,y) = 0 \text{ on } \partial\Omega_D & \text{(5.1b)} \\ \dfrac{\partial u(x,y)}{\partial n} = \nabla u(x,y) \cdot \vec{n} = 0 \text{ on } \partial\Omega_N & \text{(5.1c)} \end{cases}$$

Here α is the constant of non-linearity. It is given that $\partial\Omega_D \cup \partial\Omega_N = \partial\Omega$ and $\partial\Omega_D \cap \partial\Omega_N = \emptyset$. Furthermore f is an arbitrary source function that can depend on both x and y.

5.1. Weak Form

As in the previous section we first need to convert this equation to the weak form. For this we multiply by the by a test function $v(x,y)$ and then integrate over the domain.

$$\iint_\Omega v\nabla^2 u\,dx\,dy + \iint_\Omega v\alpha u^2\,dx\,dy = \iint_\Omega vf\,dx\,dy \tag{5.2}$$

For the weak form to be equivalent to equation 5.1a we need the same conditions on the test functions as the previous section. To find the final weak form we again use the Green's first identity 4.3.

$$-\iint_\Omega \nabla v\nabla u\,dx\,dy + \iint_\Omega v\alpha u^2\,dx\,dy = \iint_\Omega vf\,dx\,dy \tag{5.3}$$

5.2. Space Discretization

For the space discretization our solution we will create a mesh with basis functions as in section 4.2. Using these base functions and approximating our solution with $u^h(x,y) = \sum_{i=1}^{\eta} c_i\phi_i(x,y)$ we rewrite the weak form into the following non-linear problem:

$$-\sum_{i=1}^{\eta}\left[\iint_\Omega \nabla\phi_i\nabla\phi_j\,dx\,dy\right]c_i + \alpha\iint_\Omega \phi_j[u^h(x,y)]^2\,dx\,dy = \iint_\Omega \phi_j f\,dx\,dy \qquad \forall j \tag{5.4}$$

This problem can be rewritten using vectors and matrices.

$$-S\vec{c} + \alpha\vec{b}(\vec{c}) = \vec{f} \tag{5.5}$$

The vectors $\vec{c}$ and $\vec{f}$ and matrix S are identical to equation 4.7. We define the $\eta \times 1$ vector $\vec{b}$ as (η denotes the number of nodes):

$$\vec{b}(\vec{c})_j = \iint_\Omega \phi_j[u^h(x,y)]^2\,dx\,dy \qquad \forall j \tag{5.6}$$

5.3. Iteration

We now want to solve this non-linear problem by iterating to a better and better solution. We will use the index n to denote the current iteration. Filling in $\vec{c}^n$ and $u^n = \sum_{i=1}^{\eta} \vec{c}_i^n \phi_i(x,y)$ into equation 5.5 gives us this:

$$-S\vec{c}^n + \alpha\vec{b}(\vec{c}^n) = \vec{f} \tag{5.7}$$

with

$$\vec{b}(\vec{c}^n)_j = \iint_\Omega \phi_j [u^n]^2 \, dx\, dy \tag{5.8}$$

By using the approximation $[u^n]^2 \approx u^n u^{n-1}$ we can rewrite the vector $\vec{b}(\vec{c}^n)$ to the following matrix vector product:

$$\vec{b}(\vec{c}^n) \approx N(\vec{c}^{n-1})\vec{c}^n \tag{5.9}$$

with

$$N(\vec{c}^n)_{i,j} = \iint_\Omega \phi_i \phi_j u^n \, dx\, dy \tag{5.10}$$

With N of size $\eta \times \eta$. We can do this since

$$\vec{b}(\vec{c}^n)_j \approx \iint_\Omega \phi_j u^n u^{n-1} \, dx\, dy = \iint_\Omega \phi_j \sum_{i=1}^{\eta} \vec{c}_i^n \phi_i u^{n-1} \, dx\, dy = \sum_{i=1}^{\eta} \left[\iint_\Omega \phi_i \phi_j u^{n-1} \, dx\, dy \right] \vec{c}_i^n \qquad \forall j \tag{5.11}$$

In section 7.3 we will find an implementation to compute this N matrix.

Finally, we write down our complete iteration process. For our initial $\vec{c}^0$ we will use the solution to the linear problem ($\alpha = 0$). To make sure that our solution converges we multiply by a damping constant ω.

$$\vec{c}^0 = -S^{-1} f \tag{5.12a}$$

$$\vec{c}^n_{temp} = (-S + \alpha N(\vec{c}^{n-1}))^{-1} f \tag{5.12b}$$

$$\vec{c}^n = \vec{c}^{n-1} + \omega(\vec{c}^n_{temp} - \vec{c}^{n-1}) \tag{5.12c}$$

The iteration will continue until either a maximum number of iterations is reached or the difference between $\vec{c}^n$ and $\vec{c}^{n-1}$ is below a certain minimum difference.

6

Finite Element Method for Westervelt

In this chapter we will use the finite element method, the iteration from the previous section and Verlet integration to solve the Westervelt equation in 2 dimensions.

$$\begin{cases} \nabla^2 u - \dfrac{1}{c_0^2}\dfrac{\partial^2 u}{\partial t^2} + \dfrac{\beta}{\rho_0 c_0^4}\dfrac{\partial^2 u^2}{\partial t^2} = f \text{ on } \Omega & \text{(6.1a)} \\ u(x,y,t) = 0 \text{ on } \partial\Omega_D,\ \forall t & \text{(6.1b)} \\ \dfrac{\partial u(x,y,t)}{\partial n} = \nabla u(x,y,t)\cdot\vec{n} = 0 \text{ on } \partial\Omega_N,\ \forall t & \text{(6.1c)} \\ u(x,y,0) = g(x,y) \text{ on } \Omega & \text{(6.1d)} \\ \dfrac{\partial u(x,y,0)}{\partial t} = h(x,y) \text{ on } \Omega & \text{(6.1e)} \end{cases}$$

Here β is the constant of non-linearity and ρ_0 is the material ambient density. It is again given that $\partial\Omega_D \cup \partial\Omega_N = \partial\Omega$ and $\partial\Omega_D \cap \partial\Omega_N = 0$. Furthermore f is again an arbitrary source function that can depend on x, y and t.

6.1. Weak Form

As in the previous two sections, before we can use the finite element method we first need to convert the equation to its weak form. The first step in this is to multiply by a test function $v(x,y)$ and then integrate over the domain.

$$\iint_\Omega v\nabla^2 u\,dx\,dy - \iint_\Omega v\frac{1}{c_0^2}\frac{\partial^2 u}{\partial t^2}\,dx\,dy + \iint_\Omega v\frac{\beta}{\rho_0 c_0^4}\frac{\partial^2 u^2}{\partial t^2}\,dx\,dy = \iint_\Omega vf\,dx\,dy \tag{6.2}$$

Again this is only equivalent to equation 5.1a when the test functions $v(x,y)$ are sufficiently smooth and $v(x,y) = 0$ on $\partial\Omega_D$. Now we can use Green's Identity and our boundary conditions on u and v to write down the weak form.

$$-\iint_\Omega \nabla v\nabla u\,dx\,dy + \iint_\Omega v\frac{\partial^2}{\partial t^2}\left[\frac{\beta}{\rho_0 c_0^4}u^2 - \frac{1}{c_0^2}u\right]dx\,dy = \iint_\Omega vf\,dx\,dy \tag{6.3}$$

6.2. Space Discretization

By now using the same mesh and test functions as in section 4.2 and approximation u by $u^h(x,y,t) = \sum_{i=1}^{\eta} c_i(t)\phi_i(x,y)$, we rewrite the weak form into this equation.

$$-\sum_{i=1}^{\eta}\left[\iint_\Omega \nabla\phi_i\nabla\phi_j\,dx\,dy\right]c_i(t) + \frac{\partial^2}{\partial t^2}\left[\iint_\Omega \phi_j\left(\frac{\beta}{\rho_0 c_0^4}u^2 - \frac{1}{c_0^2}u\right)dx\,dy\right] = \iint_\Omega \phi_j f\,dx\,dy \qquad \forall j \tag{6.4}$$

We can rewrite this again into an equation with matrices and vectors. Matrix S and vectors $\vec{c}$ and $\vec{f}$ are identical to equation 4.7.

$$-S\vec{c}(t) + \frac{\partial^2}{\partial t^2}\vec{q}(\vec{c}(t))\big|_t = \vec{f}(t) \tag{6.5}$$

The vector $\vec{q}$ can be calculated using this formula and is of size $\eta \times 1$.

$$\vec{q}(\vec{c}(t))_j = \iint_\Omega \phi_j \left(\frac{\beta}{\rho_0 c_0^4} [u^h(x,y,t)]^2 - \frac{1}{c_0^2} u^h(x,y,t) \right) dx\,dy \qquad \forall j \tag{6.6}$$

6.3. Verlet Integration

To solve this problem we will first numerically approximate the time integration by using the Verlet method. As in section 4.4 we will use ℓ as an index to denote the time step. Thus t^ℓ again follows 4.11a for a chosen time step Δt. From now on we will use $\vec{c}^\ell = \vec{c}(t^\ell)$ and $\vec{q}^\ell = \vec{q}(\vec{c}^\ell)$. We now write down an expression for $\vec{q}^\ell$ with this new notation.

$$\vec{q}_j^\ell = \iint_\Omega \phi_j \left(\frac{\beta}{\rho_0 c_0^4} [u^h(x,y,t^\ell)]^2 - \frac{1}{c_0^2} u^h(x,y,t^\ell) \right) dx\,dy \qquad \forall j \tag{6.7}$$

$$= \iint_\Omega \phi_j \left(\frac{\beta}{\rho_0 c_0^4} \left[\sum_{i=1}^{\eta} c_i^\ell \phi_i(x,y) \right]^2 - \frac{1}{c_0^2} \sum_{i=1}^{\eta} c_i^\ell \phi_i(x,y) \right) dx\,dy \qquad \forall j \tag{6.8}$$

This $\vec{q}^\ell$ can be computed more easily from $\vec{c}^\ell$ using the following matrix vector product. The matrix M is identical to equation 4.7b and matrix N equal to 5.10.

$$q^\ell = \frac{\beta}{\rho_0 c_0^4} N(\vec{c}^\ell)\vec{c}^\ell - \frac{1}{c_0^2} M\vec{c}^\ell \tag{6.9}$$

Using equation 6.5 we can write down the Verlet integration step by filling in equation 4.12c.

$$\vec{q}^\ell = 2\vec{q}^{\ell-1} - \vec{q}^{\ell-2} + \Delta t^2 S\vec{c}^{\ell-1} + \Delta t^2 \vec{f}(t^{\ell-1}) \tag{6.10}$$

We now wish to rewrite this into an equation for $\vec{c}^\ell$, since those are the function values we are looking for. To be able to do this we need to approximate $\vec{q}^\ell$. We will do this using a very similar approximation to section 5.3, but instead of the solution of the previous iteration step we will use the solution of the previous time step.

$$[u^h(x,y,t^\ell)]^2 \approx u^h(x,y,t^{\ell-1}) u^h(x,y,t^\ell) \qquad \Rightarrow \tag{6.11}$$

$$\vec{q}^\ell \approx \frac{\beta}{\rho_0 c_0^4} N(\vec{c}^{\ell-1})\vec{c}^\ell - \frac{1}{c_0^2} M\vec{c}^\ell \tag{6.12}$$

Using this result we can write down our Verlet integration by filling in our equations for $\vec{q}$ into equation 6.10 and rewriting to $\vec{c}^\ell$. For the initial $\vec{c}^0$ and $\vec{c}^1$ we will use the same equations 4.14a and 4.14b from the linear wave equation.

$$\vec{c}^\ell = \left(\frac{\beta}{\rho_0 c_0^4} N(\vec{c}^{\ell-1}) - \frac{1}{c_0^2} M \right)^{-1} \left[\Delta t^2 \left(S\vec{c}^{\ell-1} + \vec{f}(t^{\ell-1}) \right) + \frac{2\beta}{\rho_0 c_0^4} N(\vec{c}^{\ell-1})\vec{c}^{\ell-1} - \frac{2}{c_0^2} M\vec{c}^{\ell-1} - \frac{\beta}{\rho_0 c_0^4} N(\vec{c}^{\ell-2})\vec{c}^{\ell-2} + \frac{1}{c_0^2} M\vec{c}^{\ell-2} \right] \tag{6.13}$$

6.4. Iteration

To reduce the influence of the approximation 6.11 we will use an iteration similar to section 5.3. The second index will be used to denote this iteration, so $\vec{c}^{\ell,n}$ denotes the nth iteration of the solution $\vec{c}$ at time t^ℓ. Using this notation our new approximation for $\vec{q}^\ell$ becomes:

$$\vec{q}^{\ell,n} \approx \frac{\beta}{\rho_0 c_0^4} N(\vec{c}^{\ell,n-1})\vec{c}^{\ell,n} - \frac{1}{c_0^2} M\vec{c}^{\ell,n} \tag{6.14}$$

Using this result we can again solve equation 6.10 to $\vec{c}^{\ell,n}$. The initial $\vec{c}^{\ell,0}$ is calculated using 6.13. As in section 5.3 we again apply a damping constant ω.

$$\vec{c}^{\ell,0} = \left(\frac{\beta}{\rho_0 c_0^4} N(\vec{c}^{\ell-1}) - \frac{1}{c_0^2} M\right)^{-1} \left[\Delta t^2 \left(S\vec{c}^{\ell-1} + \vec{f}(t^{\ell-1})\right) + \frac{2\beta}{\rho_0 c_0^4} N(\vec{c}^{\ell-1})\vec{c}^{\ell-1} - \frac{2}{c_0^2} M\vec{c}^{\ell-1} - \frac{\beta}{\rho_0 c_0^4} N(\vec{c}^{\ell-2})\vec{c}^{\ell-2} + \frac{1}{c_0^2} M\vec{c}^{\ell-2}\right] \tag{6.15a}$$

$$\vec{c}^{\ell,n}_{temp} = \left(\frac{\beta}{\rho_0 c_0^4} N(\vec{c}^{\ell,n-1}) - \frac{1}{c_0^2} M\right)^{-1} \left[\Delta t^2 \left(S\vec{c}^{\ell-1} + \vec{f}(t^{\ell-1})\right) + \frac{2\beta}{\rho_0 c_0^4} N(\vec{c}^{\ell-1})\vec{c}^{\ell-1} - \frac{2}{c_0^2} M\vec{c}^{\ell-1} - \frac{\beta}{\rho_0 c_0^4} N(\vec{c}^{\ell-2})\vec{c}^{\ell-2} + \frac{1}{c_0^2} M\vec{c}^{\ell-2}\right] \tag{6.15b}$$

$$\vec{c}^{\ell,n} = \vec{c}^{\ell,n-1} + \omega(\vec{c}^{\ell,n}_{temp} - \vec{c}^{\ell,n-1}) \tag{6.15c}$$

This iteration will continue until either a maximum number of iterations is reached or the difference between $\vec{c}^{\ell,n}$ and $\vec{c}^{\ell,n-1}$ is below a certain minimum difference. When iteration is stopped we set $\vec{c}^{\ell} = \vec{c}^{\ell,n}$ and can move to the next time step.

Thus our time Differentiation to the Westervelt equation consists of using equation 6.13 to compute the next time step. Then iterating using equation 6.15 to refine the solution for this time step. Then we move to the next time step and again iterate to refine the solution. This process is repeated until we have reached our end time.

7

Implementation

This chapter will cover all information regarding the implementation of the previously discussed theory into Python. We will first discuss how the mesh and its base functions were calculated and then go over and element-wise approach to compute all the necessary matrices and vectors. This is then further improved into a vectorized implementation. In the last section the implementation of the different boundary conditions is discussed.

7.1. Mesh Generation

To create a two dimensional mesh we will be relying on *GMSH*. *GMSH* has an API package available that allows you to generate a mesh directly in python. To generate the mesh the following input parameters are used in my implementation.

- A list of ordered vertices that are defined by their x and y coordinate.
- A boolean list of equal size to the number of vertices. A "1" in the list on position i indicates that there is a Neumann boundary condition between vertex i and $i+1$. A "0" in the list indicates a Dirichlet boundary condition.
- A float lc value that defines the target element size for the mesh.

The mesh is created by drawing a line from vertex through vertex in the order of the first list. Finally the path is closed by going from the last vertex in the list to the first vertex. The space inside this line path is then used to create the two dimensional mesh. The mesh is defined by these output variables.

- A *coor* matrix of size $n\_nodes \times 2$ with on each row the coordinates of the nodes.
- An *elems* matrix of size $n\_elems \times 3$ with on each row the indexes of the three corner nodes of the element.
- An *i_bnd_dir* list filled with the indices of the nodes on the Dirichlet boundary.
- An *i_bnd_neu* list filled with the indices of the nodes on the Neumann boundary.

By using these output variables we can compute all the necessary matrices and vectors for the FEM implementation.

7.2. Base Functions

To compute the matrices and vectors we need to compute the base functions on our mesh. In this project we will use only first order base functions. For each node we define a two dimensional base function following these two conditions:

- The base functions should be linear on each element.
- Each base functions should equal 1 in its corresponding node and should equal 0 in all other nodes.

Since we want our base functions to be linear on each element we can describe our base function j on a specific element as:

$$\phi_j = a_j x + b_j y + c_j \tag{7.1}$$

Say we have an element e_i, with its three corresponding nodes $\vec{x_1} = (x_1, y_1)$, $\vec{x_2} = (x_2, y_2)$ and $\vec{x_3} = (x_3, y_3)$. We

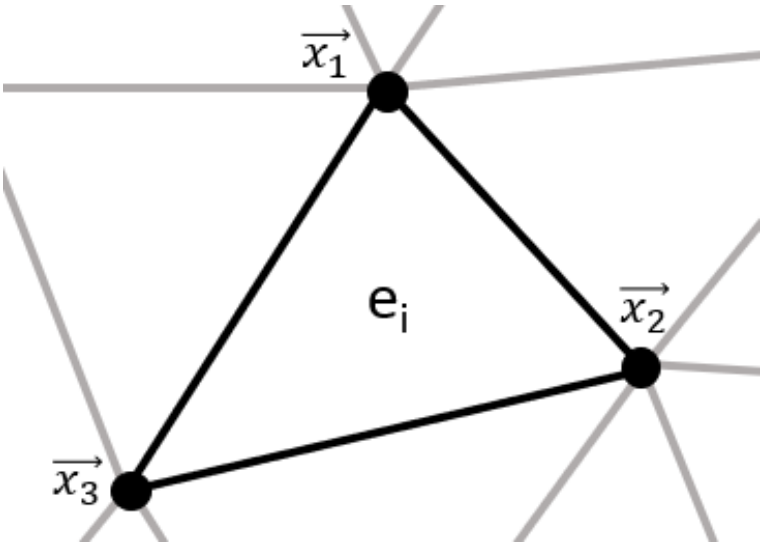


Figure 7.1: The element e_i in the mesh.

can then find the value of the three non-zero base functions on this element by solving this matrix product.

$$\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.2}$$

From this we find that the base functions on the element e_i equal:

$$\phi_1 = \frac{y_2 - y_3}{\Delta} x + \frac{x_3 - x_2}{\Delta} y + \frac{x_2 y_3 - x_3 y_2}{\Delta} \tag{7.3a}$$

$$\phi_2 = \frac{y_3 - y_1}{\Delta} x + \frac{x_1 - x_3}{\Delta} y + \frac{x_3 y_1 - x_1 y_3}{\Delta} \tag{7.3b}$$

$$\phi_3 = \frac{y_1 - y_2}{\Delta} x + \frac{x_2 - x_1}{\Delta} y + \frac{x_1 y_2 - x_2 y_1}{\Delta} \tag{7.3c}$$

$$\text{with } \Delta = x_1 y_2 + x_2 y_3 + x_3 y_1 - x_1 y_3 - x_2 y_1 - x_3 y_2 \tag{7.3d}$$

ϕ_1, ϕ_2 and ϕ_3 are the base functions corresponding to the nodes $\vec{x_1}$, $\vec{x_2}$ and $\vec{x_3}$ respectively. All other base functions are equal to zero on this element.

7.3. Element-wise Computation

The easiest way to compute the matrices S, M and N and the vector f is to use an element by element approach using the base function calculated in the previous section. By calculating them element by element we can make use of the fact that there are only three non-zero base functions on each element. Thus the element matrix is only a 3×3 matrix and the element is only 3×1. By looping over the element matrices for each element and adding them at the correct indices we can construct the full S, M and N matrices. The same can be done for the element vectors to construct the f vector. Since all of the matrices are sparse, we will be using a sparse implementation for all the matrices in our computation.

To compute the element stiffness matrix S_{e_i} we can make use of the fact that the base functions are linear on each element. From this it follows that the gradient is constant on the element, thus we can compute the integral by simply multiplying the constant gradients by the area of the element. The area of a triangular

element is equal to $\frac{\Delta}{2}$. The values of the gradient can trivially be found from equations 7.3.

$$S_{e_i} = \begin{bmatrix} \iint_{e_i} \nabla\phi_1\nabla\phi_1\,dx\,dy & \iint_{e_i} \nabla\phi_1\nabla\phi_2\,dx\,dy & \iint_{e_i} \nabla\phi_1\nabla\phi_3\,dx\,dy \\ \iint_{e_i} \nabla\phi_2\nabla\phi_1\,dx\,dy & \iint_{e_i} \nabla\phi_2\nabla\phi_2\,dx\,dy & \iint_{e_i} \nabla\phi_2\nabla\phi_3\,dx\,dy \\ \iint_{e_i} \nabla\phi_3\nabla\phi_1\,dx\,dy & \iint_{e_i} \nabla\phi_3\nabla\phi_2\,dx\,dy & \iint_{e_i} \nabla\phi_3\nabla\phi_3\,dx\,dy \end{bmatrix}$$
$$= \frac{1}{2\Delta}\begin{bmatrix} (y_2-y_3)^2+(x_3-x_2)^2 & (y_2-y_3)(y_3-y_1)+(x_3-x_2)(x_1-x_3) & (y_2-y_3)(y_1-y_2)+(x_3-x_2)(x_2-x_1) \\ (y_2-y_3)(y_3-y_1)+(x_3-x_2)(x_1-x_3) & (y_3-y_1)^2+(x_1-x_3)^2 & (y_3-y_1)(y_1-y_2)+(x_1-x_3)(x_2-x_1) \\ (y_2-y_3)(y_1-y_2)+(x_3-x_2)(x_2-x_1) & (y_3-y_1)(y_1-y_2)+(x_1-x_3)(x_2-x_1) & (y_1-y_2)^2+(x_2-x_1)^2 \end{bmatrix}$$

To calculate the element mass matrix and element N matrix we can use this approximation from [7] (page 16, formula (39)).

$$\iint_{e_i} \phi_i^n \phi_j^m \phi_k^p\,dx\,dy \approx |\Delta| \frac{n!m!p!}{(n+m+p+2)!} \tag{7.4}$$

Where ϕ_i, ϕ_j and ϕ_k are non-zero first order base functions on the element e_i. Using equation 7.4 we can directly compute the element mass matrix by setting $p = 0$ and using the correct values for n and m.

$$M_{e_i} = \begin{bmatrix} \iint_{e_i} \phi_1\phi_1\,dx\,dy & \iint_{e_i} \phi_1\phi_2\,dx\,dy & \iint_{e_i} \phi_1\phi_3\,dx\,dy \\ \iint_{e_i} \phi_2\phi_1\,dx\,dy & \iint_{e_i} \phi_2\phi_2\,dx\,dy & \iint_{e_i} \phi_2\phi_3\,dx\,dy \\ \iint_{e_i} \phi_3\phi_1\,dx\,dy & \iint_{e_i} \phi_3\phi_2\,dx\,dy & \iint_{e_i} \phi_3\phi_3\,dx\,dy \end{bmatrix} = \frac{|\Delta|}{24}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix} \tag{7.5}$$

To compute the element N matrix we need to use equation 5.10. This equation gives us this element matrix N_{e_i}.

$$N_{e_i}(\vec{c}^n) = \begin{bmatrix} \iint_{e_i} \phi_1\phi_1 u^n\,dx\,dy & \iint_{e_i} \phi_1\phi_2 u^n\,dx\,dy & \iint_{e_i} \phi_1\phi_3 u^n\,dx\,dy \\ \iint_{e_i} \phi_2\phi_1 u^n\,dx\,dy & \iint_{e_i} \phi_2\phi_2 u^n\,dx\,dy & \iint_{e_i} \phi_2\phi_3 u^n\,dx\,dy \\ \iint_{e_i} \phi_3\phi_1 u^n\,dx\,dy & \iint_{e_i} \phi_3\phi_2 u^n\,dx\,dy & \iint_{e_i} \phi_3\phi_3 u^n\,dx\,dy \end{bmatrix} \tag{7.6}$$

To compute this element matrix we need to simplify u^n. Since on element e_i all base functions equal zero except for the three base functions corresponding to the three nodes of that element, we can reduce u^n to $c_1^n\,\phi_1 + c_2^n\,\phi_2 + c_3^n\,\phi_3$. Here c_1^n, c_2^n and c_3^n are equal to the values of $\vec{c}^n$ corresponding to the nodes $\vec{x}_1$, $\vec{x}_2$ and $\vec{x}_3$ respectively. Using this we can compute the element matrix N_{e_i} by summing over the following matrices.

$$N_{e_i}(\vec{c}^n) = c_1^n N_1 + c_2^n N_2 + c_3^n N_3 \tag{7.7}$$

These three matrices can be computed using the quadrature approximation given in equation 7.4.

$$N_1 = \begin{bmatrix} \iint_{e_i} \phi_1\phi_1\phi_1\,dx\,dy & \iint_{e_i} \phi_1\phi_1\phi_2\,dx\,dy & \iint_{e_i} \phi_1\phi_1\phi_3\,dx\,dy \\ \iint_{e_i} \phi_1\phi_2\phi_1\,dx\,dy & \iint_{e_i} \phi_1\phi_2\phi_2\,dx\,dy & \iint_{e_i} \phi_1\phi_2\phi_3\,dx\,dy \\ \iint_{e_i} \phi_1\phi_3\phi_1\,dx\,dy & \iint_{e_i} \phi_1\phi_3\phi_2\,dx\,dy & \iint_{e_i} \phi_1\phi_3\phi_3\,dx\,dy \end{bmatrix} = \frac{|\Delta|}{120}\begin{bmatrix} 6 & 2 & 2 \\ 2 & 2 & 1 \\ 2 & 1 & 2 \end{bmatrix} \tag{7.8a}$$

$$N_2 = \begin{bmatrix} \iint_{e_i} \phi_2\phi_1\phi_1\,dx\,dy & \iint_{e_i} \phi_2\phi_1\phi_2\,dx\,dy & \iint_{e_i} \phi_2\phi_1\phi_3\,dx\,dy \\ \iint_{e_i} \phi_2\phi_2\phi_1\,dx\,dy & \iint_{e_i} \phi_2\phi_2\phi_2\,dx\,dy & \iint_{e_i} \phi_2\phi_2\phi_3\,dx\,dy \\ \iint_{e_i} \phi_2\phi_3\phi_1\,dx\,dy & \iint_{e_i} \phi_2\phi_3\phi_2\,dx\,dy & \iint_{e_i} \phi_2\phi_3\phi_3\,dx\,dy \end{bmatrix} = \frac{|\Delta|}{120}\begin{bmatrix} 2 & 2 & 1 \\ 2 & 6 & 2 \\ 1 & 2 & 2 \end{bmatrix} \tag{7.8b}$$

$$N_3 = \begin{bmatrix} \iint_{e_i} \phi_3\phi_1\phi_1\,dx\,dy & \iint_{e_i} \phi_3\phi_1\phi_2\,dx\,dy & \iint_{e_i} \phi_3\phi_1\phi_3\,dx\,dy \\ \iint_{e_i} \phi_3\phi_2\phi_1\,dx\,dy & \iint_{e_i} \phi_3\phi_2\phi_2\,dx\,dy & \iint_{e_i} \phi_3\phi_2\phi_3\,dx\,dy \\ \iint_{e_i} \phi_3\phi_3\phi_1\,dx\,dy & \iint_{e_i} \phi_3\phi_3\phi_2\,dx\,dy & \iint_{e_i} \phi_3\phi_3\phi_3\,dx\,dy \end{bmatrix} = \frac{|\Delta|}{120}\begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 6 \end{bmatrix} \tag{7.8c}$$

Finally, we compute the element vector $\vec{f}$ by first applying a Newton-Cotes rule and then approximating the integral again using equation 7.4.

$$f_{e_i}(t) = \begin{bmatrix} \iint_{e_i} \phi_1 f(\vec{x}_1,t)\,dx\,dy \\ \iint_{e_i} \phi_2 f(\vec{x}_2,t)\,dx\,dy \\ \iint_{e_i} \phi_3 f(\vec{x}_3,t)\,dx\,dy \end{bmatrix} \approx \begin{bmatrix} f(\vec{x}_1,t)\iint_{e_i} \phi_1\,dx\,dy \\ f(\vec{x}_2,t)\iint_{e_i} \phi_2\,dx\,dy \\ f(\vec{x}_3,t)\iint_{e_i} \phi_3\,dx\,dy \end{bmatrix} \approx \frac{|\Delta|}{6}\begin{bmatrix} f(\vec{x}_1,t) \\ f(\vec{x}_2,t) \\ f(\vec{x}_3,t) \end{bmatrix} \tag{7.9}$$

7.4. Vectorized Implementation

Though solving the finite element method through an element-wise implementation is the easiest way to do it, it is also computationally quite slow. There are two main reasons for this. Firstly, looping over all elements takes a long time especially as the element size decreases and thus the number of elements increases.

Secondly, cumulatively building up a sparse matrix by going element matrix by element matrix is a computationally intensive operation.

In this section we will instead attempt to compute all the matrices and vector through a vectorized implementation. In the vectorized implementation we can use the fast linear algebra in the python package *Numpy*.This approach is based on [8]. The goal is to compute a list of data $data$ and two list of indices i and j in such a way that $S_{i[k],j[k]} = data[k]$ for all k. Duplicate combinations of i and j are allowed and will be added together in the python sparse matrix implementation. This allows us to create the full sparse matrices on initialization.

```
S = sparse.csc_matrix((data, (i, j)), shape=(n_nodes, n_nodes))
```

Listing 7.1: Example of how the stiffness matrix is computed from the lists i, j and $data$.

In [8] it is shown how to compute the lists i, j and $data$ to create the stiffness matrix. The list $data$ contains contains the values of the first element stiffness matrix from top left to bottom right followed by the values of the second element stiffness matrix etc. This list is created in an efficient vectorized manner. The lists i and j contain the correct indices corresponding to going over the element matrices one by one in this manner. In these computations we will also use the list $area$ that contains the value of Δ for each element. The list $area$ is computed by taking a cross product.

```
ind_i = []
ind_j = []
for ind1 in range(3):
        for ind2 in range(3):
                ind_i.append(ind1)
                ind_j.append(ind2)

i = elems[:, ind_i].ravel("F")
j = elems[:, ind_j].ravel("F")
```

Listing 7.2: Computation of the lists i and j in a similar manner to [8].

```
c1 = coor[elems[:, 0], :]
d21 = coor[elems[:, 1], :] - c1
d31 = coor[elems[:, 2], :] - c1
area = np.cross(d21, d31)
```

Listing 7.3: Computation of the list $area$.

```
a = np.sum((coor[elems[:, 1]] - coor[elems[:, 2]]) ** 2, axis=1) / area / 2.
b = np.sum((coor[elems[:, 2]] - coor[elems[:, 0]]) ** 2, axis=1) / area / 2.
c = np.sum((coor[elems[:, 1]] - coor[elems[:, 2]]) * (coor[elems[:, 2]] - coor[elems
    ↪ [:, 0]]), axis=1) / area / 2.

data = numpy.concatenate((a, c, - a - c, c, b, - b - c, - a - c, - b - c, a + b + 2
    ↪ * c), axis=None)
```

Listing 7.4: Computation of the list $data$ for the stiffness matrix as was done in [8].

To efficiently compute the mass matrix we will use the same list of indices i and j. To get the list with the data we will take the Kronecker product of the flattened indices of M_{e_i} and the $area$ list and divide the result by 24. This directly gives us the correct $data$ list to compute the mass matrix.

```
temp = np.abs(area) / 24.
m_ek = np.array([[2.0, 1.0, 1.0], [1.0, 2.0, 1.0], [1.0, 1.0, 2.0]]).ravel()
data = np.kron(m_ek, temp)
```

Listing 7.5: Computation of the list $data$ for the mass matrix.

The largest bottleneck in the computation time is the computation of the sparse N matrix, since this needs to be done for each iteration in each time step. Therefore the goal was to make this as efficient as possible. The goal is to calculate the index lists i and j beforehand together with a matrix N_{data} that contains all constant values of N_1, N_2 and N_3 for each element. The final data list could then be computed by simply multiplying $\vec{c}^n$ and N_{data}.

```
data = c[elems] * N_data
```

Listing 7.6: Computation of the list $data$ for the N matrix from the vector $\vec{c}$ and N_{data}.

For this to work the matrix N_{data} needs to be 3 dimensional. The first dimension denotes the position in N_k from top left to bottom right. The second dimension denotes which element N_k is calculated. The last dimension denotes which of the three N_k from 7.8 is used. In total N_{data} is of size $9 \times n_{elems} \times 3$. This matrix is computed by taking the matrix product of the indices of the three N_k and the $area$ list.

```
temp = np.abs(area) / 120.
n_ek = np.array(
        [[[6., 2., 2.],
        [2., 2., 1.],
        [2., 1., 2.]],
        [[2., 2., 1.],
        [2., 6., 2.],
        [1., 2., 2.]],
        [[2., 1., 2.],
        [1., 2., 2.],
        [2., 2., 6.]]]).reshape((9, 3))
N_data = temp[:, np.newaxis] @ n_ek[:, np.newaxis, :]
```

Listing 7.7: Computation of the N_{data}. The *np.newaxis* makes sure that the dimensions match up with what was described.

The index lists i and j for the N matrix can be computed by concatenating the i and j that were computed in Listing 7.2 three times.

For each of these vectorized calculations it has been verified that the result is equal to the matrix found by using the element-wise approach. By implementing these vectorized ways of computing the matrices the computation time was greatly decreased. Previously the calculation time of the sparse matrices was the main bottleneck of the finite element method. Now this only takes up around three percent of the total computation time. The current bottleneck is the solving of the sparse linear system that needs to be done at every iteration. This takes up around 95 percent of the computation time for a fine mesh.

7.5. Boundary Conditions

The boundary conditions are implemented in the same way for all three FEM implementations. By default the finite element method applies Neumann zero boundary conditions on all boundaries. For the purposes of this project we will only be using zero boundary conditions, so there is no implementation for non-zero Neumann boundaries. To enforce Dirichlet boundaries we change all values for nodes on the Dirichlet boundary to the prescribed values at that time step. This is done after each time and iteration step.

8

Results

In this chapter we will discuss the results of the finite element method compared to the one dimensional analytical wave and the two dimensional circularly symmetric wave. It will then go over the results of the non-linear diffusion finite element solution and also compare these to an analytical result. Finally we will analyse the relation between the mesh size vs error and time step vs error.

8.1. One Dimensional Wave Example

To verify our two dimensional implementation we will first compare it to the the one dimensional result from [4] and [3]. We will also use the analytical result found in 3.1 to verify our Finite Element result. The analytical solution is for a semi-infinite domain, in our numerical implementation we will approximate this by choosing a sufficiently large L. By doing this our solution will be valid until the displacement at $x = 0$ reaches the boundary at L. Thus our solution will be valid until around $T_{valid} = \frac{L}{c_0}$. Our boundary condition on the boundary at $x = L$ does not influence the result for $t < T_{valid}$. We will use a Neumann zero boundary, since that does not require additional effort in the FEM implementation.

To simulate a one dimensional result in our two dimensional finite element method we will use a rectangular domain. In the x-direction this domain will simulate the one dimensional result and in the y-direction the solution should have the same value for all y. In the y-direction we will use Neumann zero boundaries on both ends. To reduce the computation time we will not use a very wide rectangle in the y-direction. This does not affect our results, since we expect all y-values to be identical. The mesh that was used can be seen in figure 8.1.

As in [3] we will use the following time dependant boundary condition at $x = 0$.

$$h(t) = P_0 \sin\left[2\pi f_0 (t - T_d)\right] \exp\left[-\left(\frac{t - T_d}{T_w/2}\right)^2\right] \tag{8.1}$$

We will use the constants in table 8.1 in our calculation. They are similar to what was used in [3].
Using these values we can make a plot of the time dependent boundary condition. This function is a sinus that oscillates with a frequency of f_0. A Gaussian envelop is placed over the sinus to only have a pulse of about six periods. This boundary condition is chosen because of the rapid oscillation from low pressure to high pressure. The pulse that this creates is also very similar to ones used in practical ultrasound imaging for medical applications.

8.1.1. Linear Solution

For the linear wave equation we expect this boundary condition to induce a wave with the same shape that is moving to the right with speed c_0. Running the linear finite element method implementation with the given values we get the following result. For visibility we just plot the x-dimension versus the pressure for all values. The y-dimension is removed from the plot by plotting the x-location and pressure value of each node and then drawing a line through all those points.

The finite element solution does what we expect. The boundary condition travels to the right with speed c_0 and does not change in amplitude. The solution does not vary in the y-direction. If this was the case we

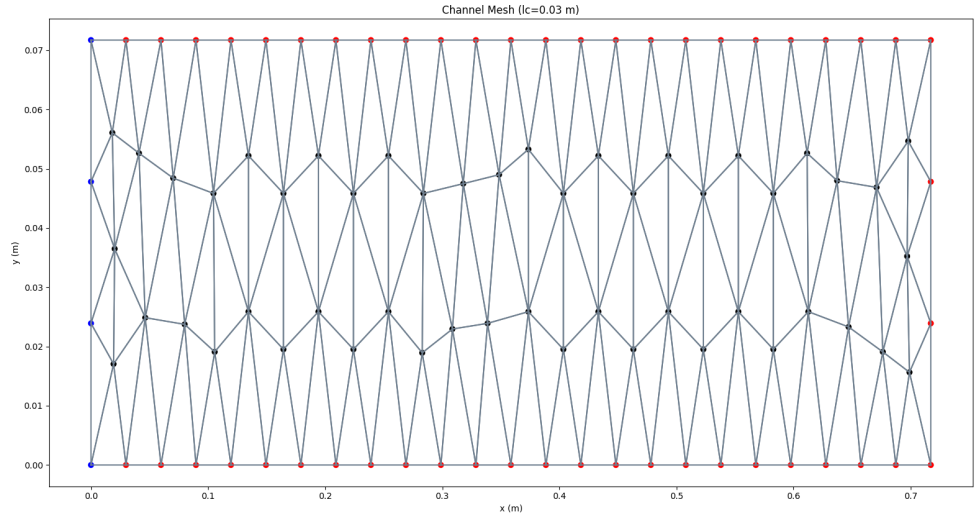


Figure 8.1: Plot of the mesh with a significantly larger element size ($lc = 0.03m$) and larger $L_y = \frac{L_x}{10}$ so single elements are visible. Interior nodes are denoted by a black dot. The nodes on the Neumann boundary are red and the nodes on the Dirichlet boundary are blue.

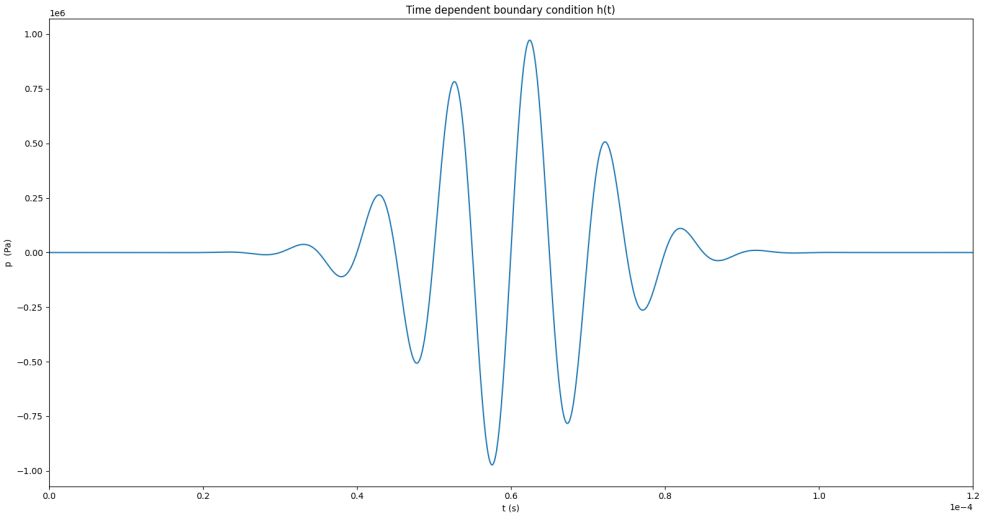


Figure 8.2: Plot of the Time dependant boundary condition using the parameters given in table 8.1.

would be seeing multiple different pressure values for the same x-value in figure 8.3. To better compare the finite element solution to the analytical solution we zoom in on the wave and plot the analytical solution over it.

Figure 8.4 shows that the finite element solution is almost identical to the analytical solution. The finite element solution seems to have a slight shift to the right. To take a better look at this error we can look at the following plot.

Figure 8.5 shows how the error changes over time. We can see here that the error increases with the time as was also seen in [3]. The increase in error can also partially be caused by reflections due to the wave getting closer to the right boundary.

Symbol	Value	Unit	Name
ρ_0	1000	$\frac{kg}{m^3}$	material ambient density
c_0	1500	$\frac{m}{s}$	wave speed
β	10		coefficient of non-linearity
P_0	10^6	Pa	coefficient of nonlinearity
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	$x_{sh} + T_{end} \cdot c_0$	m	Domain length
L_y	$\frac{L_x}{100}$	m	Width in y-direction
lc	$\frac{c_0}{36 f_0}$	m	Mesh element size
dt	$\frac{lc}{4c_0}$	s	Time step
ϵ	$10^{-9} \cdot P_0$	Pa	Tolerance for iteration
m_{iter}	50		Maximum number of iterations
ω	0.7		Damping in iteration

Table 8.1: Values for parameters used in finite element method for the one dimensional wave equation.

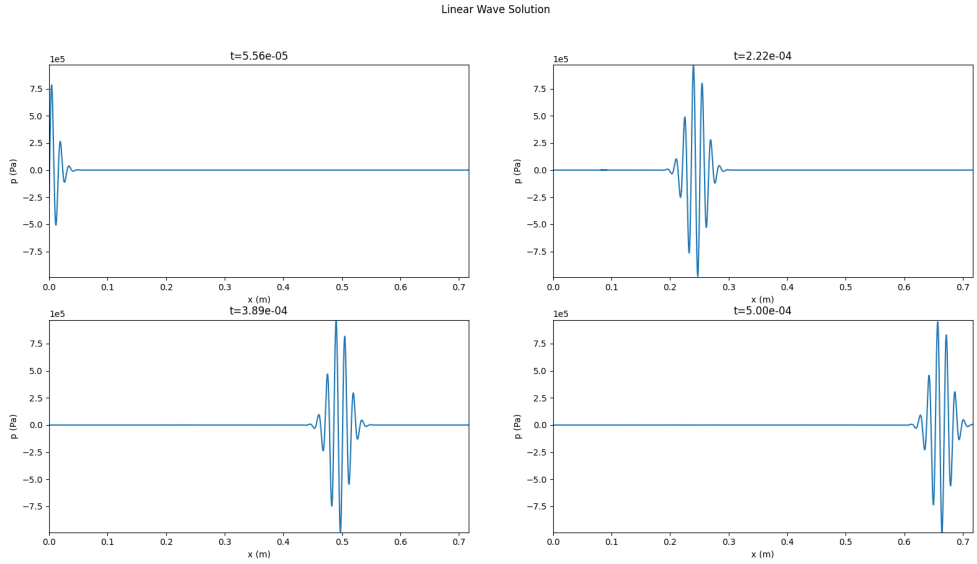


Figure 8.3: Plot of the finite element solution for four different time values.

8.1.2. Westervelt Equation

Due to the non-linear term in the Westervelt equation the wave speed for high pressure values is higher than the wave speed for low pressure values. Thus for the non-linear solution we expect the slope of high to low pressure to become steeper and the slope of low to high pressure to become gentler. When the faster high pressure wave overtakes the slower low pressure part of the wave it creates a shock wave. We first plot the finite element solution for different time values to see how it changes over time.

In figure 8.6 we see that the solution is initially similar to the linear solution. Then for larger time values we see that the changes from high to low pressure indeed become steeper. This also results in an increase of the amplitude for the low pressure and a decrease for the higher pressure. In the last time frame one can see the creation of shock waves that distort the original shape of the wave. To better be able to see these differences we will plot the solution side by side with the analytical solution in figure 8.7.

In figure 8.8 we see that the difference between the linear and Westervelt solution increases over time. Also unlike in figure 8.5 the difference is no longer similar in shape to the original wave.

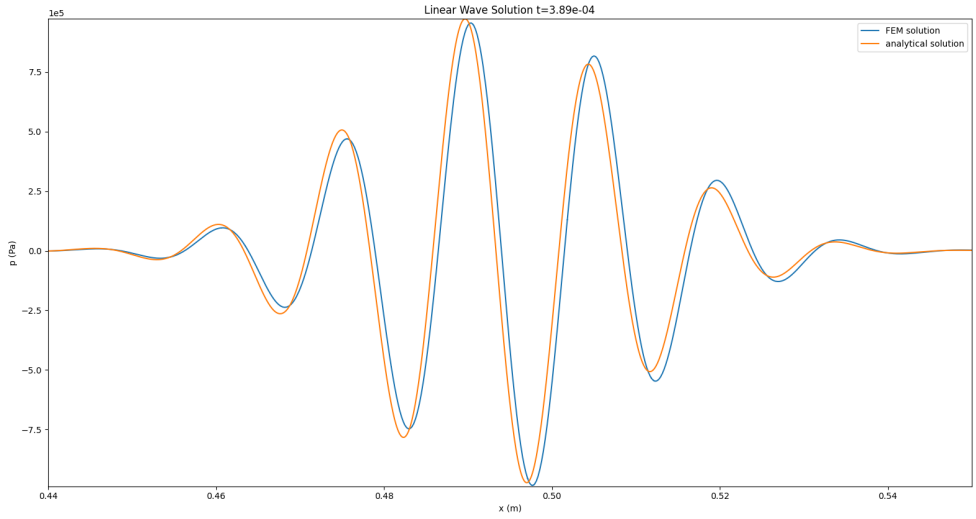


Figure 8.4: Plot of the finite element solution and the analytical solution zoomed in on the wave.

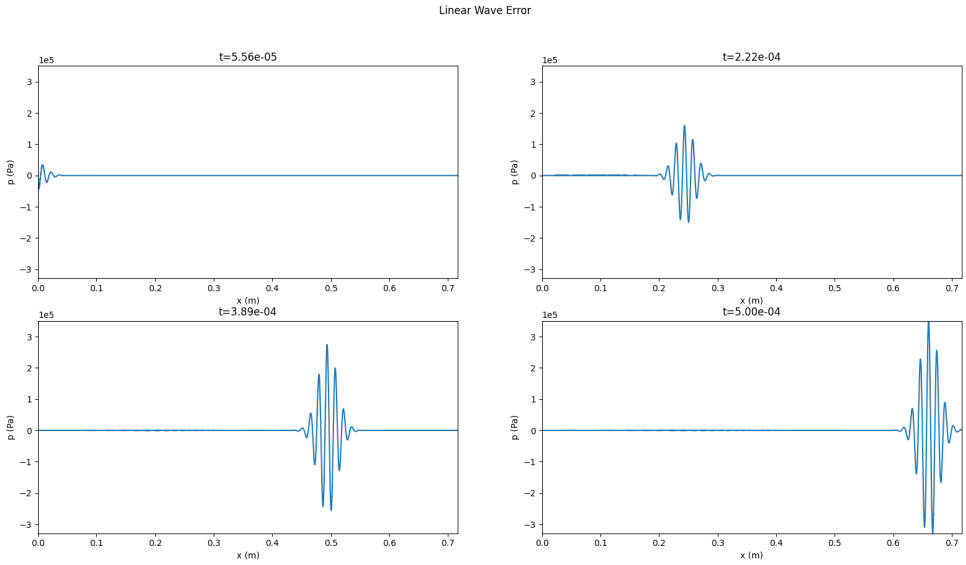


Figure 8.5: Plot of the error between the fem solution and analytical solution for four different time values.

8.2. Circular Symmetric Two Dimensional Wave Example

We will now verify our two dimensional finite element implementation by comparing it to the analytical solution for a circular symmetric wave that was found in section 3.2. To numerically simulate the infinite domain we will use a sufficiently large circular domain. In our finite element solution we can construct a mesh by its vertices. Thus we choose to use a hexagonal mesh for this problem. As in the previous section our solution is only valid up until a maximum time of around $T_{valid} = \frac{\sqrt{3}R}{2c_0}$, which is the distance from the centre to the closest side of the hexagon. On all sides of the hexagon we will use Neumann zero boundary conditions.

As in section 3.2 we will use a two dimensional Gaussian function as our initial condition.

$$u(r,0) = P_0 \exp\left[\frac{-r^2}{2\sigma^2}\right] \tag{8.2}$$

For our parameters we will use values similar to what we used in the last section. All parameters that were

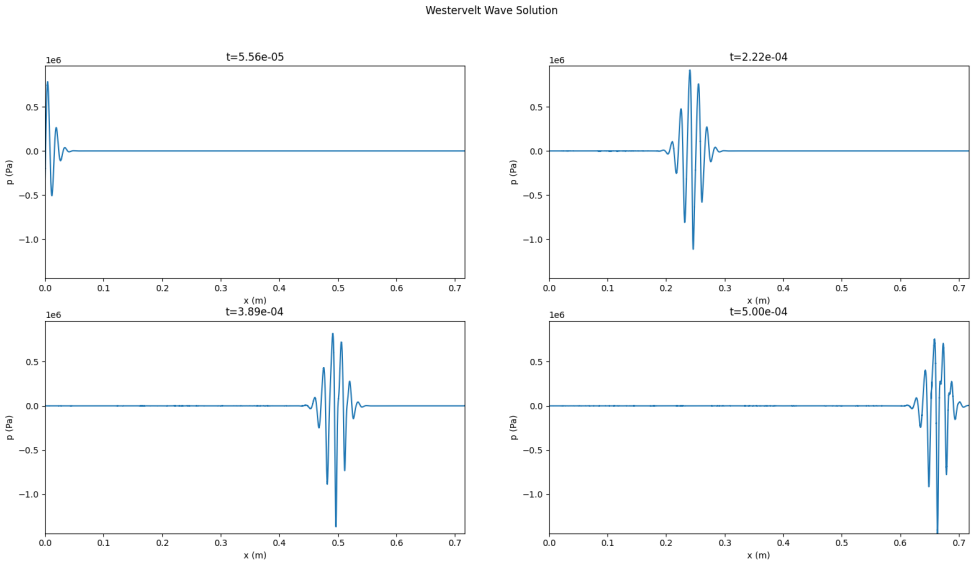


Figure 8.6: Plot of the Westervelt finite element solution for four different time values.

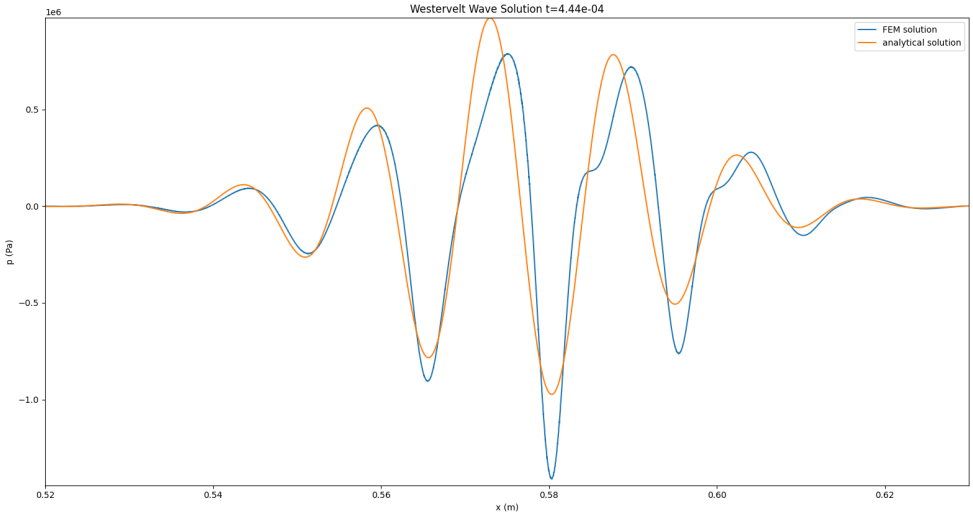


Figure 8.7: Plot of the Westervelt finite element solution and the analytical solution zoomed in on the wave.

used can be found in table 8.2.
In figure 8.10 the initial condition is shown in two dimensions.

8.2.1. Linear Solution

For the linear wave equation we expect the result to be similar to a raindrop falling on a water surface. The initial condition will cause a ripple that decreases in amplitude due to the wave having a larger area. As in the previous section we will plot our solution in one dimension. Because of the circular symmetry of the problem we will plot the radius versus the pressure.

In figure 8.11 the difference between the analytical solution and the finite element solution is so small that the analytical solution is not visible below the finite element solution. The behaviour is as we expected. The initial condition causes a circularly symmetric ripple that decreases in intensity as the time increases.

Figure 8.12 shows the difference with the analytical solution. As in the previous section the error does

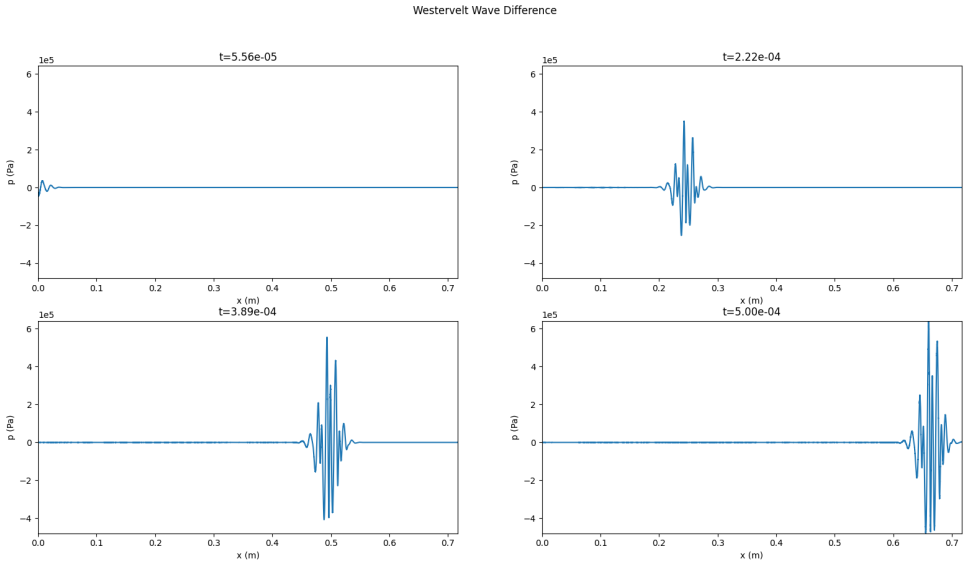


Figure 8.8: Plot of the difference between the fem solution and analytical solution for four different time values.

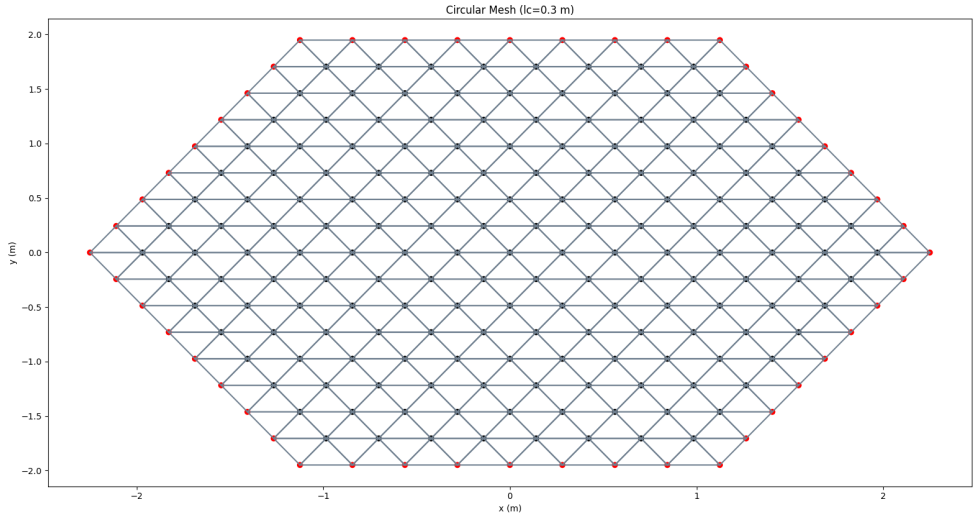


Figure 8.9: Plot of the mesh with a significantly larger element size ($lc = 0.03m$)so single elements are visible. Interior nodes are denoted by a black dot. The nodes on the Neumann boundary are red.

seem to slightly increase with time. The error here is significantly smaller than in the one dimensional example. In the plot we see that the error increases when the rate of change of the solution increases.

8.2.2. Westervelt Equation

In this section the difference due to the non-linear term in the Westervelt equation should be significantly smaller due to the difference between the low and high pressure decreasing as time increases. The number of oscillations is also lower for this example, which should also lower the difference between the Westervelt and Linear solution.

The difference between the analytical and Westervelt solution is not discernable in these plots. Thus we conclude that there is barely any influence from the non-linear term in this problem. In figure 8.14 we see that the difference between the analytical and Westervelt solution is very similar to 8.12. By increasing the

Symbol	Value	Unit	Name
ρ_0	1000	$\frac{kg}{m^3}$	material ambient density
c_0	1500	$\frac{m}{s}$	wave speed
β	10		coefficient of non-linearity
P_0	10^6	Pa	coefficient of nonlinearity
t_{end}	10^{-3}	s	End time for FEM
σ	0.1	m	Source width
R	$\frac{3}{2}c_0 t_{end}$	m	Domain radius
lc	$\frac{\sigma}{10}$	m	Mesh element size
dt	$\frac{lc}{4c_0}$	s	Time step
ϵ	$10^{-9} \cdot P_0$	Pa	Tolerance for iteration
m_{iter}	50		Maximum number of iterations
ω	0.7		Damping in iteration

Table 8.2: Values for parameters used in finite element method for the circular symmetric wave equation.

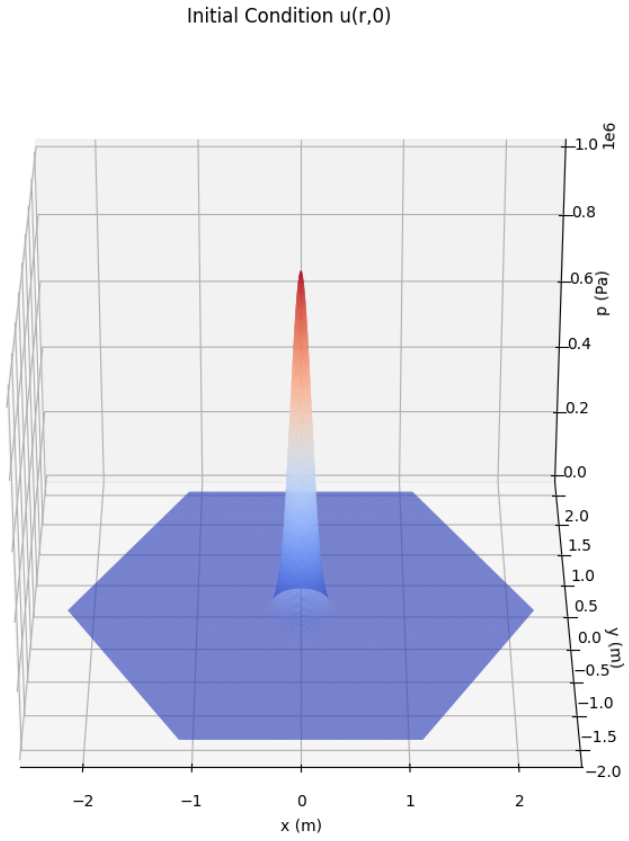


Figure 8.10: 2D Plot of the initial condition using the parameters given in table 8.2.

value of β the non-linear effects can be made more visible.

8.3. Non-linear Diffusion

To verify the solution found in chapter 5 we will compare it to a one dimensional reference solution from [9]. In one dimension equation 8.3 is a solution to the non-linear diffusion equation with $\alpha = -1$ and $f = 0$.

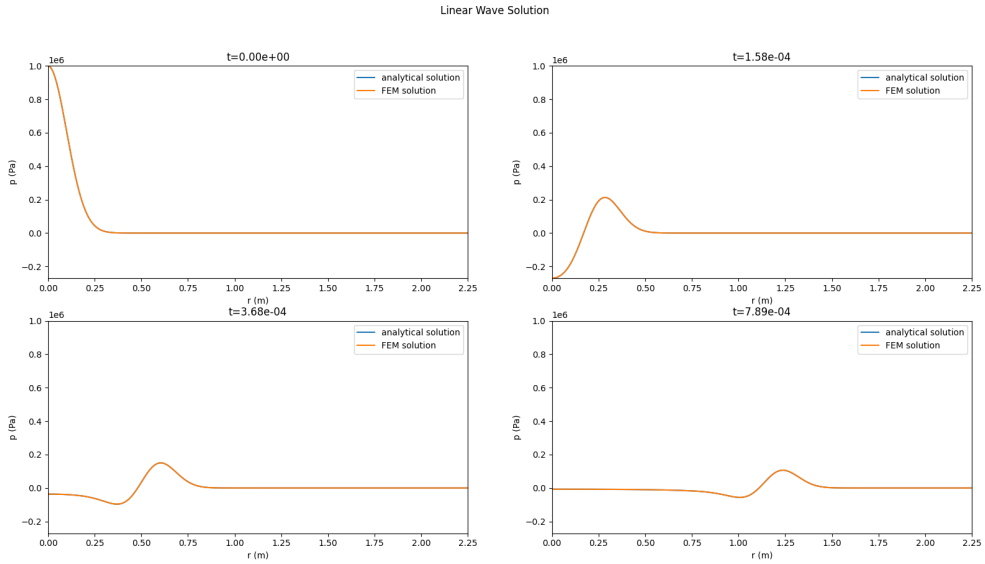


Figure 8.11: Plot of the finite element solution for four different time values.

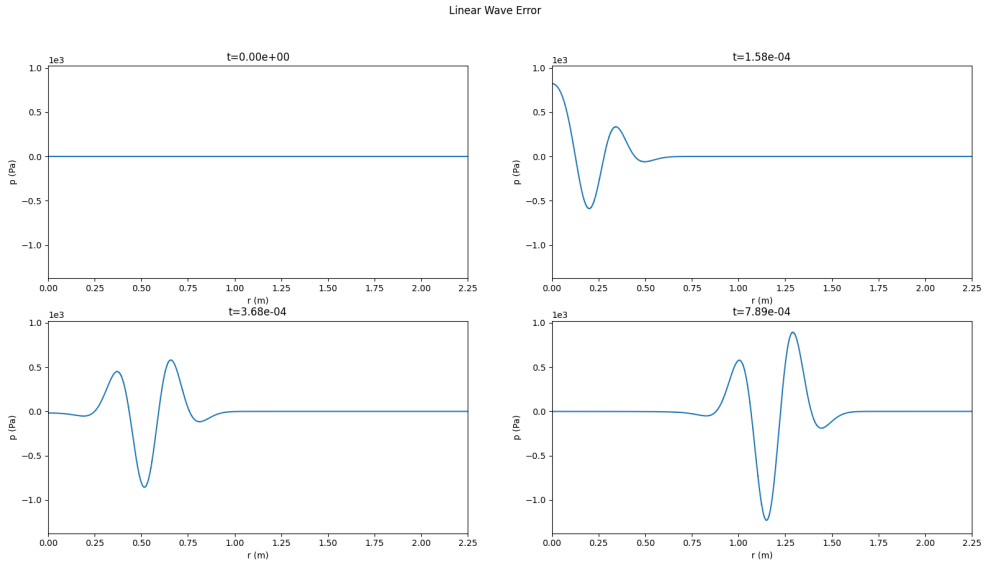


Figure 8.12: Plot of the error between the fem solution and analytical solution for four different time values.

$$\begin{cases} u(x) = \dfrac{6}{(1+x)^2} & \text{(8.3a)} \\ u(0) = 6 \qquad u(1) = \dfrac{3}{2} & \text{(8.3b)} \end{cases}$$

As in section 8.1 we will use a rectangular domain to simulate a one dimensional wave. The mesh that was used is almost identical to the mesh plotted in figure 8.1. The only difference is that both the left and right side of the rectangle have Dirichlet boundary conditions. The parameters that were used for the Non-linear Diffusion solution can be found in table .

In figure 8.15 the result of the finite element method is shown. The initial iteration step is a straight line between the two boundary conditions. This is to be expected, since it is the trivial solution to $\frac{\partial^2 u}{\partial x^2} = 0$. For each iteration after the initial condition the solution is slowly moved to the correct analytical solution. In the

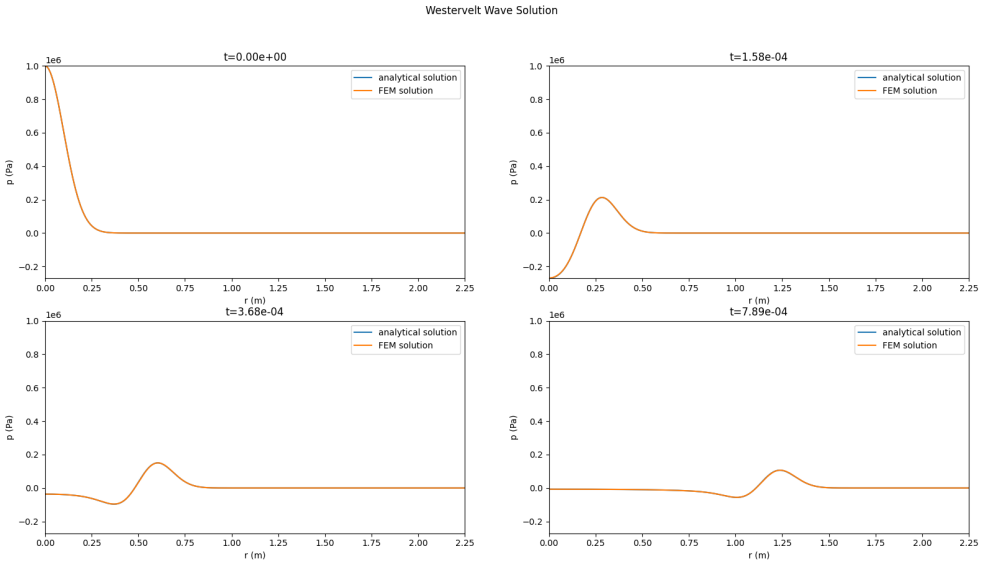


Figure 8.13: Plot of the Westervelt finite element solution and the linear analytical solution for four different time values.

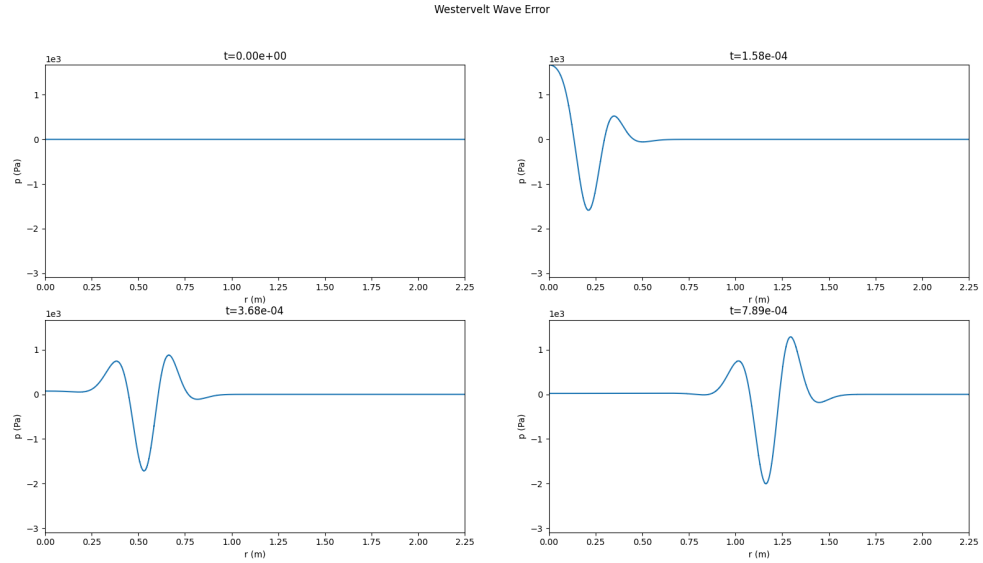


Figure 8.14: Plot of the difference between the fem solution and analytical solution for four different time values.

difference plot in figure 8.16 the decrease in the difference over the iterations can be seen.

8.4. Mesh Size vs Error

To evaluate the influence of the mesh size on the error we will compare the results of the finite element implementation to both our analytical solutions. As an error norm we will be using the ℓ^1-norm on the difference between the two solutions.

$$\epsilon = \max_i(|\vec{c}_i - u_{ana}(\vec{x}_i, t)|) \tag{8.4}$$

Where u_{ana} is the analytical solution evaluated in the node corresponding to $\vec{c}_i$. To compute the error we will be using the $\vec{c}$ solution from the last time step of the computation.

We will first look at the influence of the mesh size on the error for the two time dependant reference

Symbol	Value	Name
α	-1	coefficient of non-linearity
L_x	1.0	Domain length
L_y	$\frac{L_x}{10} = 0.10$	Width in y-direction
lc	0.002	Mesh element size
ϵ	10^{-9}	Tolerance for iteration
m_{iter}	50	Maximum number of iterations
ω	0.3	Damping in iteration

Table 8.3: Values for parameters used in finite element method for the Non-linear Diffusion equation.

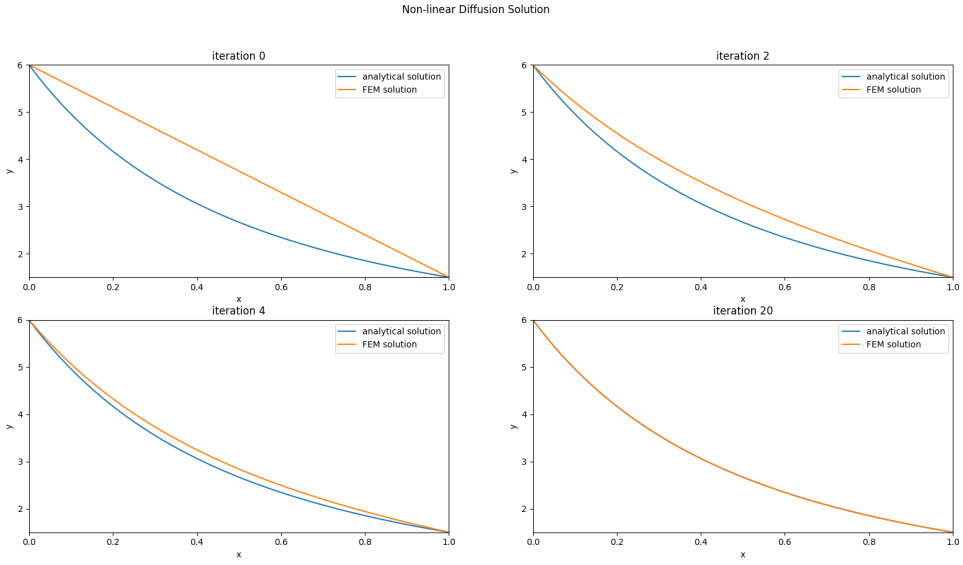


Figure 8.15: Plot of the finite element solution and the analytical solution to the Non-linear diffusion equation. Iteration 0 denotes the initial solution $\bar{c}^0$.

solutions. For the calculations we will use the same parameters as were used in tables 8.1 and 8.2. We will only change the value of lc for both problems. We will solve the problem with the values of lc in the tables multiplied by a scaling of between 100 and 1. These scaling values are spaced on a logarithmic scale. After solving the problem we compute the error and plot the lc versus error ϵ on a logarithmic scale to find the relation between them. For the finite element method we expect this to be a relation of the form $\epsilon \propto lc^2$.

As can be seen in the figure there were a lot of points that show weird behaviour for the resulting error of larger mesh sizes, especially for the one dimensional problem. In the 1-dimensional case the solutions would become inaccurate for larger mesh sizes due to the oscillations in the wave happening faster then the distance between nodes in the mesh. This causes some strange behaviour in the solution that leads to a kind of averaging of the wave over the interval. Due to this behaviour the error decreases even though the solution is not accurate. For this reason we choose to eliminate a large number of data points for the one dimensional problem. For the two dimensional problem this behaviour takes longer to take shape since the Gaussian curve is easier to approximate using linear elements. For larger values of lc this effect again occurs, but it is less present than for the one-dimensional case.

From these two fits we found a relation of $\epsilon \propto lc^{1.56}$ and $\epsilon \propto lc^{1.66}$ from the 1-dimensional and 2-dimensional cases respectively. These two relations are similar, which is what we would expect. However both of them are lower than the quadratic relation that we expected. This is especially the case for the 1-dimensional channel problem. It would be good to better estimate the one dimensional error fit by having more data points for lower lc values. Unfortunately this was not possible, due to an error in the *GMSH* API.

My best guess for the cause of this error is that in the one dimensional case the mesh is very long and thin, which causes the triangular elements to become very stretched out, especially as the element size decreases. When lowering the lc value too much the elements become so stretched out that *GMSH* can no

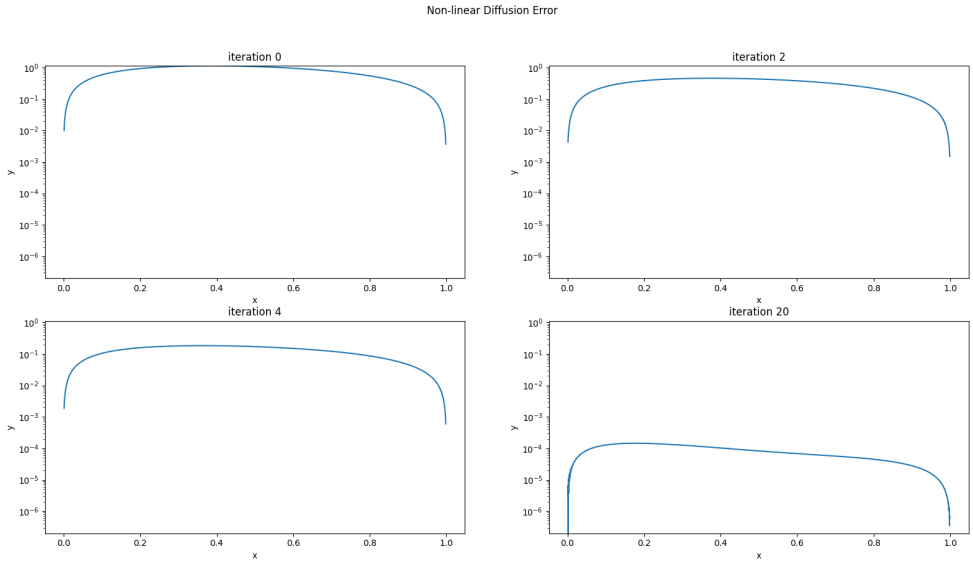


Figure 8.16: Plot of the difference between the finite element solution and the analytical solution to the Non-linear diffusion equation with a log scale on the y-axis. Iteration 0 again denotes the initial solution $\vec{c}^0$. The difference on the Dirichlet boundary nodes is not included in the plots since the difference is always zero.

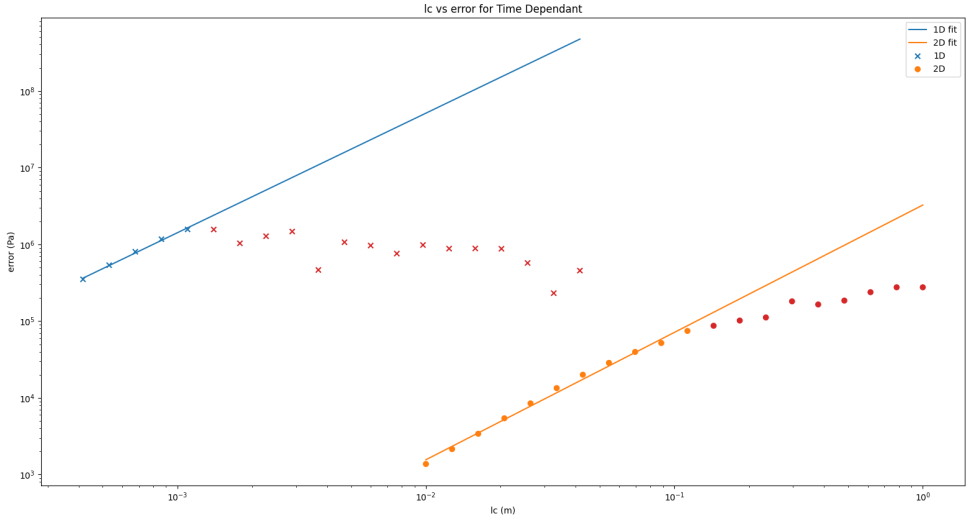


Figure 8.17: Plot of the lc versus the error on a logarithmic scale. The red markers were used to designate points that were not included in fitting the line. The plot includes two linear fits for the relation between the error and the mesh size.

longer differentiate between the two elements and an error occurs in *GMSH*.

The same procedure was used to analyse the relationship between the error and the element size for the non-linear diffusion. The parameters were kept equal to the ones given in table 8.3. The lc is again scaled by values between 100 and 1 on a logarithmic scale. This resulting logarithmic plot of the error can be found in figure 8.18.

For this problem all the data points seem to follow the linear fit, although there is some more inaccuracy for the higher element sizes. This is probably because the non-linear diffusion problem does not have an initial or boundary condition that is hard to approximate with linear elements. From the linear fit we find a relation of $\epsilon \propto lc^{1.98}$, which agrees much better with our expectation of $\epsilon \propto lc^2$. This leads me to believe that the different relations that we found for the time dependant problems are related to the Verlet Time

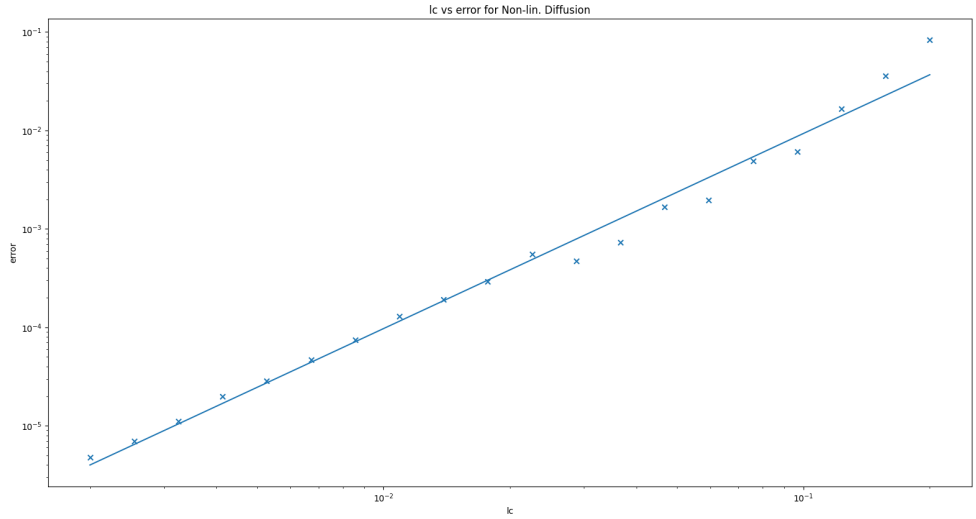


Figure 8.18: Plot of the lc versus the error on a logarithmic scale. The plot includes a linear fit for the relation between the error and the mesh size.

Integration.

8.5. Time Step vs Error

To investigate the influence of the time step size on the error we will be using a similar approach to the one we used in the last section. We will be using the same error norm as equation 8.4. As previously we will be using the parameters of tables 8.1 and 8.2. The only parameter that we will change is the size of the time step. We will be solving both time dependant problems for twenty different time step values. The value of the time step in the tables will be multiplied by logarithmically scaled values between 10 and 0.1. After calculating the solution for each of these time values we will calculate the error on the last time step.

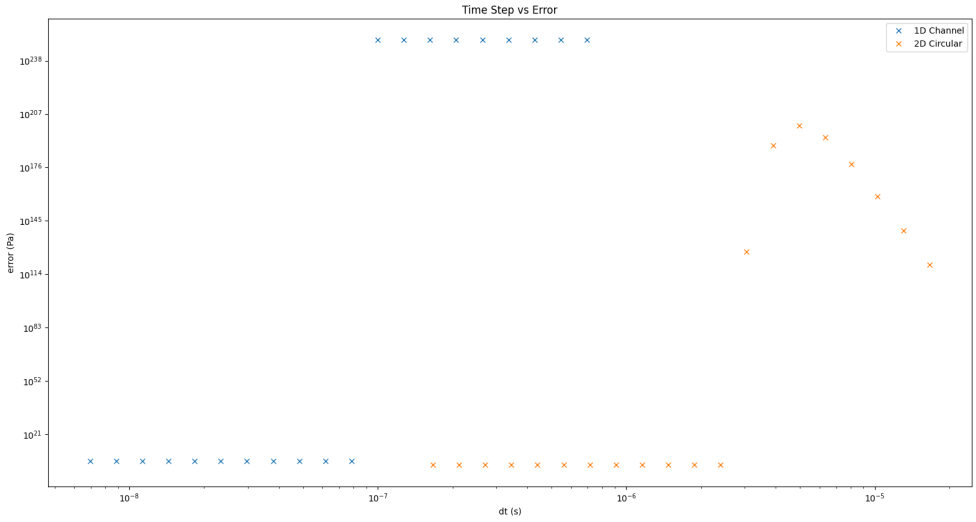


Figure 8.19: Plot of the time step versus the error on a logarithmic scale. For the solutions where the solution would become NaN, the error has been plotted as the value 10^{250}.

In figure 8.19 we see that the error explodes when the time step is too large. For the two dimensional problem the implementation gives a solution which is incredibly large and thus the error is very large. For the one dimensional channel the solution consists of NaN values. This is because the solution blows up so much that the values pass the maximum float limit of python of $1.7976931348623157 \times 10^{308}$. When this happens python returns "Not a Number"(NaN). To still include these data points in the error plot they have been plotted as the value 10^{250}. From this figure we can conclude that the convergence of the finite element method depends on the time step used.

We will now plot only the values for which the solution converges and zoom in on the two error values for the two different solutions.

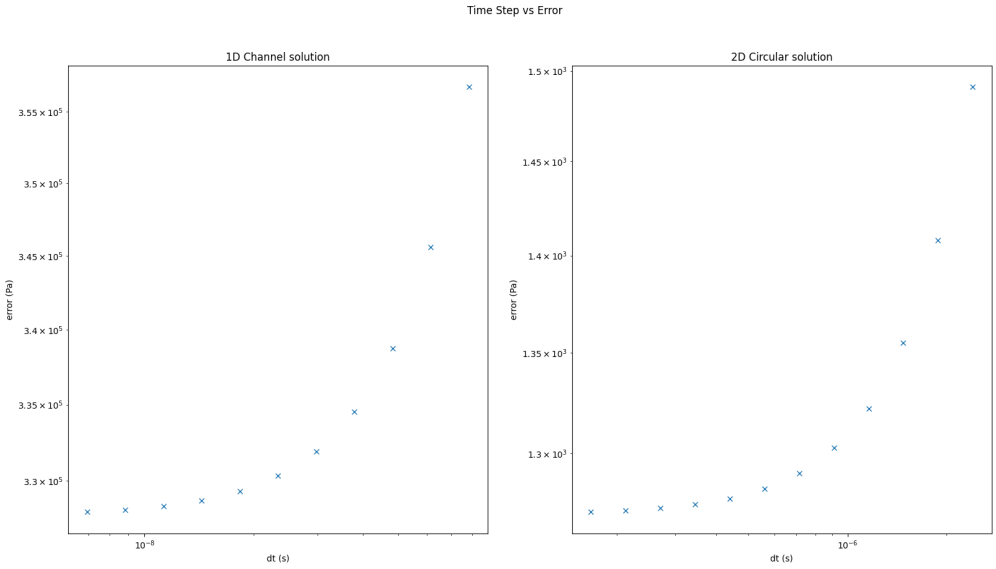


Figure 8.20: Plot of the time step versus the error on a logarithmic scale. This plot includes only the values for which the solution converges and has been split in two subplots for each solution.

The figure 8.20 shows that both solutions show the same relation between the time step and the error. Unlike for the mesh size the logarithmic plot does not show a straight line. From this we can conclude that the time step does not follow a relation of the form $\epsilon \propto dt^k$ to the error. The error decreases with the time step until it plateaus, the error can then not decrease further unless decreasing the mesh size. This is because the error caused by approximating the domain with a mesh can not be decreased by decreasing the time step.

9

Conclusion

In this thesis we found a way to compute the two dimensional Westervelt equation with the finite element method. To do this we first had to solve the linear wave equation and the non-linear diffusion equation using the finite element method. We found a way to iterate on each time step that allows us to approximate the non-linearity in the Westervelt equation.

We were able to verify this finite element implementation by comparing it to the one dimensional work of [3]. After which we also compared it to a two dimensional linear circularly symmetric solution. From this we conclude that the finite element method can be used to solve the Westervelt equation in two dimensions. The non-linear diffusion finite element solution was also verified by comparing it to the work done in [9]. We have also shown that a faster vectorized implementation can be used instead of the element-by-element one.

The relation between the error and the mesh size was also analysed. For the time-dependant finite element implementation the error was found to scale with around $\epsilon \propto lc^{1.7}$. This value is below our expected relation of $\epsilon \propto lc^2$. For the non-linear diffusion equation the relation between the mesh size and error agrees with our estimate, because it equals $\epsilon \propto lc^{1.98}$.

When investigating the relation between the time step and the error we found that the convergence of the finite element method depends on the time step used. Unlike the mesh size, the time step does not follow a relation to the error of the form $\epsilon \propto dt^k$. This is because the error stops decreasing when the time step gets very small.

10

Discussion

In this research we focused on implementing the finite element method for two dimensional domains. All the theory that was discussed in this thesis can be extended to three dimensional domains. This would make the results more applicable to real life situations. The main work that would need to be done would be in chapter 7, where most of the calculations of the integrals would have to be changed.

The finite element implementation could be further improved by adding adaptive mesh refinement for areas of large change in the solution values. This could greatly improve both the accuracy of the program as well as the computation time. For example in the current implementation a large amount of computation time is spend on the area of the domain the wave has not reached yet that is equal to zero. Another feature that could be added is to add the option of non-zero Neumann boundary conditions to the code implementation.

For two dimensional finite element method it was verified that the results of the vectorized implementation gave equal results to the element-wise implementation. To quantify the improvement that this change in implementation gave it would be good to do a more quantitative research of these changes on the computation time for different mesh sizes and time steps.

When looking at figure 8.4 we can see that the difference between the analytical and the numerical FEM solution seem to be a slight shift to the right. This might be caused by one of the approximations we did when applying the finite element method. It is also possible that the root of this shift is an error somewhere in our theory or implementation that can be fixed. Further investigation would be required to figure this out. The shift might be related to the way that the time dependent Dirichlet boundary is implemented, since it does not seem to appear in figure 8.11.

In section 8.2 the differences between the analytical linear and finite element Westervelt solution are very small. The value for the non-linearity constant that was used in this section $\beta = 10$ was the same one as used in the section before it. In that section it did give very noticeable results. The reason that the non-linear phenomena in this section were less noticeable is firstly because the wave oscillates less compared to the one dimensional wave. The second reason is that the amplitude of the wave quickly decreases. It would be good to increase the non-linearity constant in this section to better show the effects of the Westervelt equation on the circularly symmetric problem.

When verifying the finite element implementation we mostly made use of linear reference solutions especially when calculating the errors. It would be good to further verify the non-linear finite element solutions by comparing them to a non-linear reference solution. Unfortunately there are no analytical solutions to the Westervelt Equation. Thus we would have to rely on a previously verified numerical solution from another source to verify the implementation for non-linear problems. This would help to further proof that the theory in this paper is correct.

The analysis of the mesh size vs error could be best improved by getting more data points for the 1-dimensional problem. To do this the problem would need to be solved with smaller values of lc, which currently result in errors. By fixing the error one would be able to better approximate the relation by making a better fit. The error could also be fixed by increasing the width of the channel, though this would also greatly increase computation time. Another way to improve this section would be by getting more different reference solutions to better confirm the relation between the error and mesh size.

Further research should be done into how the convergence of the finite element method relates to the time step used. For this we would have to investigate the time step for which convergence stops for different

mesh sizes and problems.

In this thesis we found that the relation between the mesh size and the error is below the expected quadratic relation for the time dependant problems. For the non-linear diffusion equation we found the correct relation between the two. In section 8.5 we found that the error stops decreasing when decreasing the time step at a certain point. I would hypothesize that the error for the finite element method consists of a summation of an error caused by the Verlet Integration and an error caused by the approximation of the domain by the mesh. Thus when we plot the relation between the mesh size and the error with a sufficiently small time step used to make the error of the Verlet Integration negligible. I expect that the relation for the time dependant problems will also be closer to the expected $\epsilon \propto lc^2$. Verifying this would take a lot of extra computation time and was not possible to include in this thesis. For future research it would be interesting to look further into this error relation.

Bibliography

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

[2] 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.

[3] Zijta, M., 2017. *Solving The Westervelt Equation With Losses Using First And Second Order Finite Element Method*, Delft University of Technology, Faculty of Applied Sciences & Faculty of Electrical Engineering, Mathematics and Computer Science.

[4] 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.

[5] Churchill, R.V., 1972. *Operational Mathematics*, 3rd edition, McGraw-Hill Book Company.

[6] Haberman, R., 2014. *Applied Partial Differential Equations with Fourier Series and Boundary Value Problems*, 5th edition, Pearson Education Limited.

[7] Segal, A., 2017. *Finite element methods for the incompressible Navier-Stokes equations*, Delft University of Technology, Faculty of Applied Sciences & Faculty of Electrical Engineering, Mathematics and Computer Science.

[8] Funken, S., Praetorius, D. and Wissgott, P., 2011. *Efficient Implementation of Adaptive P1-FEM in Matlab* in *Computational Methods in Applied Mathematics*, Vol. 11., No. 4, pp. 460-490.

[9] Plaquevent-Jourdain, B., 2019. *Méthodes d'analyse numérique pour la résolution d'équations différentielles et aux dérivées partielles non-linéaires*, Delft University of Technology, Faculty of Applied Sciences & Faculty of Electrical Engineering, Mathematics and Computer Science.

A

Appendix

All code related to this project can be found in the GitLab repository at `https://gitlab.com/leo.hendriks/bep-fem/`.