Evaluation and Validation of an Ultrasound Modeling Tool

CHRISTOPHER VINCENZO FEBO
Evaluation and Validation of an Ultrasound Modeling Tool

CHRISTOPHER VINCENZO FEBO
Abstract

An ultrasound modeling tool, from the software newFASANT, which uses a ray-tracing technique is analyzed, tested and evaluated for industrial purpose in the automotive industry in collaboration with Volvo Car Corporation. The specific application for which it is evaluated is to simulate the sensor surrounding (traffic environment) for ultrasound sensors on cars. This is used in traffic simulators for testing assisted and automated parking functions in a virtual environment.

A theoretical study is carried out to analyze the method used in the tool. Ray-tracing and its background assumptions are compared to numerical simulation methods by an in-depth study of the main properties and advantages, disadvantages, and a computational complexity investigation. The results of this study deem the method advantageous from a computational speed point of view and complete enough in its results for the specific automotive application.

Tests are carried out on the newFASANT US module by comparing simple cases such as reflection from a disk to known theoretical results to establish the accuracy of the tool. More complex scenarios, with diffraction through narrow slits, are tested to investigate the limits of the software and its method (ray tracing). In these tests the tool is compared to another one, MATLAB toolbox, which makes use of a numerical method. Lastly an evaluation of performance is made with a common application scenario. The results of the tests were close to the theoretical predictions and the assumptions coming from the theoretical study of the methods. The modeling tool is then considered a valid option for simulating ultrasound propagation in car simulation programs.
Sammanfattning

Evaluering och validering av ett ultraljudmodelleringsverktyg


Acknowledgments

I would first like to thank my supervisors Stefan Persson at Volvo Car Corporation and Michael Hanke at the Department of Mathematics at KTH for the valuable help and guidance given to me throughout this project.
I would also like to thank my family, for the support and motivation they have always provided, giving me the opportunity to pursue and complete the study programme which this master thesis concludes.
Finally I would like to thank two friends, Irene Natale and Paolo Enrico Viotti, for making my last two years as a student something special.

C.V. Febo
## Contents

1 Introduction 1
   1.1 Background ................................................. 1
   1.2 Aim ......................................................... 4
       1.2.1 Academic Objective ................................. 4
       1.2.2 Industrial Objective ................................. 4

2 Theory 6
   2.1 Ultrasound Modeling ........................................ 6
       2.1.1 Wave equation ........................................ 6
       2.1.2 Acoustic phenomena ................................... 7
       2.1.3 Geometrical acoustics ................................. 10
   2.2 Simulation methods .......................................... 17
       2.2.1 Ray Tracing ............................................ 17
       2.2.2 Numerical Methods .................................... 18
       2.2.3 Computational complexity investigation ............ 22
   2.3 Tool specific method ........................................ 26
       2.3.1 Ray Tracing in depth .................................. 26
       2.3.2 Optimization of Ray Tracing .......................... 28

3 Tool Evaluation 31
   3.1 newFASANT ................................................ 31
   3.2 Evaluation Method .......................................... 33
       3.2.1 newFASANT US module vs. Theory .................... 33
       3.2.2 newFASANT US module vs. MATLAB/k-Wave ............ 34
       3.2.3 Performance ........................................... 34
   3.3 Results ..................................................... 35
       3.3.1 newFASANT US module vs. Theory - Benchmark tests ............................................................................. 35
3.3.2 newFASANT US module vs. Theory - Reflection from a spherical surface .............................. 38
3.3.3 newFASANT US module vs. MATLAB/k-Wave .............................. 43
3.3.4 Performance ........................................ 47
3.4 Bugs, errors, and fixes ...................................... 54

4 Conclusions .................................................. 57
4.1 Algorithms and theory ...................................... 57
4.2 Software validation ......................................... 58
4.3 Future work ................................................. 58
Chapter 1

Introduction

1.1 Background

Ultrasound and its use in the automotive industry

Ultrasound is an acoustic wave characterized by high frequencies. These frequencies are defined as beyond the range of human hearing, which is considered to be between 20 Hz and 20 kHz. Therefore ultrasonic devices operate at frequencies higher than 20 kHz.[11]

Like any other sound wave, ultrasound travels in air at around 340 m/s and so has a wavelength that is at its largest about 17 mm. The short wavelength (high frequency) makes it useful for many applications since it can be easily directed, and it is inaudible to humans. Some animals, such as bats, are able to perceive ultrasound, and concerns have been raised regarding the disturbance which could be caused to them. The high directivity is a result of the short wavelengths, which ensure that the pressure does not spread out and disperse as quickly as it would with lower frequencies.

Major applications are object detection and distance measuring by means of ultrasonic transmitters, composed of a sensor and a transducer. These transmitters can convert electric signals into ultrasonic waves, thanks to the transducer component, or can do the opposite, with the sensor. By measuring time-span and direction of outgoing and incoming waves it is possible to determine the presence and vicinity of obstacles. Transmitters that handle both emission and reception of waves are usually the ones preferred for industrial purposes.[8]
In the automotive industry these are currently used for assisted parking, an example is visualized in Figure 1.1, to detect obstacles and inform the driver about nearing obstacles. However, with the advent of autonomous vehicles, they are now being used to develop fully automated parking functions and to contribute, together with a range of other sensors, to active safety systems.

Figure 1.1: Assisted parking [6]

For further innovation and development of these systems being able to simulate and virtually test a large range of scenarios is of vital importance. The growing complexity of these systems, which coordinate an increasing number of different kinds of sensors around the vehicle and intake large amounts of data, requires accurate and reliable simulation environments to be built. In our specific case the focus is on the simulation of ultrasound propagation between sensors, vehicles, and other objects, which create a 3D environment of complex geometries.
A schematic representation of the ultrasonic sensors and ultrasound propagating from them can be seen in Figure 1.2 below.

![Image of ultrasonic sensors](image-url)

Figure 1.2: Ultrasonic sensors [5]
1.2 Aim

The aim of this thesis project is to evaluate and validate a commercial software tool, newFASANT, used for simulating ultrasound in a virtual environment. More specifically the tool must be capable of replicating the behaviour of ultrasound sensors on a vehicle, and their interaction with the surrounding environment, with as much accuracy as possible. Once the tool is validated and ready for use there should also be an interface to make it available for integration in vehicle simulation programs.

In detail the project can be seen both from a more academic point of view alongside a more industrial one. This brings us to the definition of two different objectives which can still be considered highly interconnected.

1.2.1 Academic Objective

Method investigation

The simulation tool uses what is called a ray-tracing algorithm for the simulation of ultrasound propagation in space. This algorithm is based on asymptotic techniques such as geometrical optics and the uniform theory of diffraction. The academic objective is therefore to research and evaluate this kind of algorithm, and the techniques used to optimize it. This will be done by opposing Ray-Tracking to other simulation methods, such as numerical ones, which are based on more rigorous techniques that directly solve the wave equation on discretized geometries. The study will be carried out based on a theoretical performance evaluation, in which speed and accuracy of the algorithms will be considered. In addition to this a more ample discussion will be made taking into consideration the use that the simulation tool is being built for since different algorithms can have different adequacy depending on the scenarios.

1.2.2 Industrial Objective

Software Tool Evaluation

Since newFASANT’s Ultrasound Simulation (US) module is still under development the software must also be verified on a more practical
scale. The verification aims at finding and pointing out errors in the tool by comparing simulation results to cases where the outcome is known from theory. The final objective is to have a fully reliable tool which maintains simulation results within an established error tolerance which will be defined.

Integration to vehicle simulation programs

Once the software tool has been verified another industrial goal is to enable the integration of the US module in standard vehicle simulation programs for use in the automotive industry. The possibility of this being done will be investigated by taking advantage of the fact that the module offers the possibility of being called from command line, with arguments such as car position and surrounding environment.
Chapter 2

Theory

2.1 Ultrasound Modeling

2.1.1 Wave equation

Ultrasound, being simply a particular range of acoustic waves, propagates in space and time according to the wave equation. It is a second-order linear partial differential equation that can apply to various physical phenomena. In our specific case we have a source term \( f = f(x, t) \) and an acoustic pressure field \( p = p(x, t) \) both dependent on a space domain, and a time domain [3]:

\[
\Delta p - \frac{1}{c^2} \frac{\partial^2 p}{\partial t^2} = f
\]  

(2.1)

where \( c \) is the speed of sound, at which our acoustic waves propagate, and \( \Delta \) is the Laplace operator, written explicitly as

\[
\Delta p = \sum_{i=1}^{n} \frac{\partial^2 p}{\partial x_i^2}
\]  

(2.2)

The general analytic solution to the equation is given by the superposition of two arbitrary waveforms, \( f \) and \( g \), traveling in opposite directions:

\[
p = f(ct - x) + g(ct + x)
\]  

(2.3)
In our particular case we will be dealing with sinusoidal waves of the form:

\[ p = p_0 \sin(\omega t \pm kx) \]  

(2.4)

where \( \omega \) is the angular frequency and \( k \) is the wave number. This solution is easily obtained by setting either \( f \) or \( g \) to be a sinusoid in the general solution, and the other function to be 0.[3, 18]

### 2.1.2 Acoustic phenomena

To be able to create, use or examine a simulation tool it is of fundamental importance to understand the physical phenomena that the software is set to replicate. The physics of ultrasound, like many other kinds of waves, is governed by a series of properties which depend highly on the medium in which the wave propagates.

**Acoustic impedance**

Specific acoustic impedance is a measure of how much opposition a system generates to the acoustic flow. This transfers to acoustic pressure being applied to the system. When an acoustic wave propagates in a medium, the particles are subject to displacements around an equilibrium. The pressure exerted on a particle, over the velocity of its displacement, defines the specific acoustic impedance \( Z \):

\[ Z = \frac{p}{v} \]  

(2.5)

The SI unit of specific acoustic impedance is the Rayl. [18, 7]

**Acoustic Intensity, Inverse-square Law, and Acoustic Intensity Level**

Acoustic intensity \( I \) is defined as the power transmitted per unit area in the normal direction to that area. The intensity can be related to sound pressure and impedance with the following relation[7]:

\[ I = \frac{p^2}{2Z} \]  

(2.6)

The majority of sound waves we will consider will be of spherical form, in which case it is possible to know the behaviour of intensity in radial direction, by computing it as a function of the distance \( d \) from
the source. Pressure decreases inversely with the square root of the distance, so intensity follows the relationship called inverse-square law\[11\]:

\[
I(d) \propto \frac{1}{d^2} \quad (2.7)
\]

In addition another way of measuring ultrasound waves is the acoustic intensity level, or sound intensity level. It is the measurement of the level of intensity of sound with respect to a reference value, and comes from a logarithmic transformation of an intensity ratio\[11\].

\[
L = 10 \log_{10} \left( \frac{I}{I_0} \right) \text{ dB} \quad (2.8)
\]

In air usually the reference sound intensity is taken to be \(I_0 = 1\mu W/m^2\).

**Reflection and Refraction**

When a wave encounters a boundary layer between two different media reflection and refraction take place. The way they take place is entirely dependent on the acoustic characteristics of the media (impedances, speeds of sound). We will in principle be dealing with smooth surface boundaries, which give specular (mirror-like) reflections, but it is good to note that if surface roughness were in any case comparable to wavelength in order of magnitude, some diffusion in the reflections would occur. Another simplification made is the fact of solely considering longitudinal waves, and not transverse ones, since the latter are non-existent in fluids, such as air\[7\].

According to Snell’s Law, also known as law of refraction, if we take \(\Theta_I\) to be the angle of incidence of the wave, the angle of reflection \(\Theta_R\) would be equal. Whilst the angle of refraction, which tells the direction of transmitted waves, \(\Theta_T\) can be computed from:

\[
\frac{\sin \Theta_I}{c_1} = \frac{\sin \Theta_T}{c_2} \quad (2.9)
\]

where \(c_1, c_2\) are the sound velocities of the two media, which link to impedance and density through \(c = Z/\rho\) \(2.10\).

The intensity ratio of reflected and transmitted waves is given by the coefficients:

\[
R = \left( \frac{Z_1 - Z_2}{Z_1 + Z_2} \right)^2 \quad T = \frac{4Z_1Z_2}{(Z_1 + Z_2)^2} \quad (2.11)
\]
where \( R \) is the reflection coefficient and \( T \) the transmission one. The equation \( R + T = 1 \) (2.12) must also hold.[7]

As for wave shapes and forms, plane waves follow the above rules thoroughly. This is the case for spherical waves in purely reflective scenarios, where the wave shape is preserved. The spherical waves which get reflected behave as if they came from an origin which is the mirror image of the real origin with respect to the interface between the two media.[11]

**Diffraction**

The term diffraction includes a series of phenomena which occur when a wave encounters an obstacle or an aperture. To define diffraction the concept of geometrical shadow of an obstacle must be introduced: this is the region of space in which the wave does not propagate directly from the source. Diffraction however is the bending of waves around obstacles into their geometrical shadow.

The kind of diffraction which occurs is dependent on the ratio between obstacle slit and wavelength.

All this can be modeled according to the Huygens-Fresnel principle which states that each point on a wave-front is a new source of spherical wavelets, and that there exists interference between these wavelets. The envelope wavelets defines the wavefront at any instant.[14, 11] An image representation of this principle can be seen in Figure 2.1.

Figure 2.1: Huygens-Fresnel principle applied to diffraction of a plane wave though a slit
2.1.3 Geometrical acoustics

Geometrical acoustics is a way of modeling sound propagation based on the concept that acoustic energy is transported along lines perpendicular to the wave-front, rays. This branch of acoustics is at the basis of simulation methods such as ray tracing and beam tracing. The theory is based on the assumption of having high frequency sound propagation, making ultrasound a valid application. Due to this the equations involved are nearly identical to those used in geometrical optics, which is the equivalent, more consolidated, study of light propagation as rays.

Some basic properties of geometrical acoustics are summarized below [9]:

- the wave-front is considered locally planar
- the direction of the rays follows the normal to the wave-front
- rays in homogeneous media travel in straight lines, in non-homogeneous media they have non-zero curvature
- power is conserved within a "bundle of rays", called flux tube:

\[
\int \int_{S_1} \overrightarrow{W} \cdot d\overrightarrow{s} = \int \int_{S_2} \overrightarrow{W} \cdot d\overrightarrow{s}
\]  (2.13)

where \( \overrightarrow{W} \) is the power, \( d\overrightarrow{s} \) the infinitesimal surface element, and \( S_1, S_2 \) the areas of wavefront over which the power is calculated. A visualization of this property can be seen in Figure 2.2. As a consequence of this property we obtain the inverse-square law seen in Equation 2.7.

- reflection and refraction follow Snell’s law
- a reflected ray is related linearly to the incoming ray at the reflection point by a reflection coefficient
An equation for rays

The modeling of sound as rays can be justified analytically by the transition from the wave equation to a differential relation which can be applied to rays, called the eikonal equation, thanks to an approximation method. We will derive this equation as is done in [12].

We start by recalling the wave Equation in 2.1 and rewriting it in a homogeneous form with more generic variables as:

$$\nabla^2 \Phi - \frac{n^2}{c_0^2} \frac{\partial^2 \Phi}{\partial t^2} = 0$$

(2.14)

where \( n = \frac{c}{c_0} \) is the refractive index of the medium relative to it, and \( c_0 \) the reference speed of sound. \( \nabla^2 \) is another notation for the Laplace operator described previously in Equation (2.2) Assuming the solution is a plane wave, and that \( n \) is constant, we obtain:

$$\Phi = \Phi_0 e^{i(kr - \omega t)}$$

(2.15)

where \( \omega \) is the angular frequency, \( r \) the distance from the source, and \( k = n \frac{\omega}{c_0} = \frac{2\pi}{\lambda} \) (2.16) the wave number. The wave number can be rewritten as \( k = nk_0 \) (2.17) with \( k_0 \) the wave number in vacuum.

If \( n \) were to vary slightly, enough to consider an inhomogeneous medium but to still have nearly plane waves, using 2.17 we write:

$$\Phi = e^{A(r) + ik_0(L(r) - \omega t)}$$

(2.18)

where \( A \) is the amplitude, and \( L \) the eikonal function. If we substitute the above into Equation 2.14 we obtain

$$ik_0[2\nabla A \cdot \nabla L + \nabla^2 L]\Phi + [\nabla^2 A + (\nabla A)^2 - k_0^2(\nabla L)^2 + n^2k_0^2]\Phi = 0$$ (2.19)
The two terms in brackets must both be equal to 0 for Equation 2.19 to hold, since $A$ and $L$ are both real and measurable. As a result of this we obtain the following system of equations:

\begin{align}
\nabla^2 A + (\nabla A)^2 + k_0 (n^2 - (\nabla L)^2) &= 0 \\
2\nabla A \cdot \nabla L + \nabla^2 L &= 0
\end{align}

(2.20)\hspace{1cm} (2.21)

For the equations 2.20 and 2.20 to hold we must make an approximation assumption, which is at the heart of geometrical optics and geometrical acoustics. We assume that $n$ varies only very slowly as a function of distance, being able to consider it near constant over distances of the order of the wavelength. Another condition required is that the third term in Equation 2.20 goes to 0, and since $k_0 = 4\pi / \lambda_0^2$, we require the following:

$$
(\nabla L)^2 = n^2
$$

(2.22)

The above equation is called the eikonal equation, and can be considered the fundamental equation of geometrical optics and geometrical acoustics. It is a non-linear partial differential equation that describes the constant-phase surfaces of a wave, represented by the level surfaces of function $L$. It provides a link between physical acoustics and geometrical acoustics. From the eikonal equation rays can be defined as lines tangent in every point to $\nabla L$. [12]

Now we go on to derive an equation for rays, as is done in [10]. The generic tangent to the phase surface can be represented by

$$
N = \nabla L
$$

(2.23)

which is the normal to the surface. This is a property of the gradient.

We introduce now two more quantities: the radius vector $\mathbf{r}$ and the elementary displacement $ds$ along the ray. If we take the infinitesimal element of the radius vector, $d\mathbf{r}$, it is logical to assume the following:

$$
\begin{align}
d\mathbf{r} &\to ds, \hspace{1cm} \text{if} \hspace{0.5cm} d\mathbf{r} \to 0 \\
\frac{d\mathbf{r}}{ds} &= s_0
\end{align}
$$

(2.24)

a schematic example can be seen in Figure 2.3 below.
Vector $s_0$ is then a unit vector tangent to a ray. We can now take Equation 2.22 into consideration and write the relations:

$$n \frac{d\mathbf{r}}{ds} = \nabla L \quad \text{or} \quad ns_0 = \nabla L \quad (2.25)$$

By differentiating Equation 2.25 with respect to $s$ and expanding the gradient in cartesian coordinates we obtain

$$\frac{\partial}{\partial s} \left( n \frac{d\mathbf{r}}{ds} \right) = \frac{\partial}{\partial s} \left( i \frac{\partial L}{\partial x} + j \frac{\partial L}{\partial y} + k \frac{\partial L}{\partial z} \right) \quad (2.26)$$

By taking one of the components as example and using Equation 2.25 we can write

$$\frac{\partial}{\partial s} \frac{\partial L}{\partial x} = s_0 \cdot \nabla \frac{\partial L}{\partial x} = \frac{\nabla L}{n} \cdot \nabla \frac{\partial L}{\partial x} \quad (2.27)$$

Now by first factoring out the partial derivative in $x$, then taking into consideration the chain rule derivation backwards, and finally applying Equation 2.22, we obtain the following set of equalities:
\[
\n\nabla L \cdot \nabla \frac{\partial L}{\partial x} = \nabla L \cdot \frac{\partial}{\partial x} \nabla L = \frac{1}{2} \frac{\partial}{\partial x} (\nabla L)^2 = \frac{1}{2} \frac{\partial}{\partial x} n^2 \quad (2.28)
\]

so as to finally obtain
\[
\frac{\partial}{\partial s} \left( \frac{\partial L}{\partial x} \right) = \frac{1}{2} \frac{n^2}{\partial x} = \frac{\partial n}{\partial x} \quad (2.29)
\]

and by extending to the other dimensions
\[
\frac{\partial}{\partial s} \left( \frac{\partial L}{\partial y} \right) = \frac{\partial n}{\partial y} \quad \frac{\partial}{\partial s} \left( \frac{\partial L}{\partial z} \right) = \frac{\partial n}{\partial z} \quad (2.30)
\]

If we take Equation 2.26 and substitute the equalities we have:
\[
\frac{\partial}{\partial s} \nabla L = \nabla n \quad (2.31)
\]

Now, combining the result with Equation 2.25, we have
\[
\frac{\partial}{\partial s} \left( n \frac{\partial r}{\partial s} \right) = \nabla n \quad \text{or} \quad \frac{\partial}{\partial s} (n \mathbf{s}_0) = \nabla n \quad (2.32)
\]

These differential equations come to show that the ray position can be determined simply by the variation of \( n \) without having to pass through the unknown function \( L \).

Finally if we are to consider the specific case of a homogeneous medium with a constant \( n \), relevant to our applications, Equation 2.32 boils down to a second order homogeneous ODE:
\[
\frac{d^2 r}{ds^2} = 0 \quad (2.33)
\]

which has the known solution
\[
\mathbf{r} = a s + b \quad (2.34)
\]

showing that the acoustic ray is in fact a straight line determined by the two constant vectors \( a \) and \( b \), and the initial displacement \( s \).[10]
Fermat’s principle

An important principle in geometrical acoustics is Fermat’s principle, which provides a statement on the path that rays follow and a basis for deriving Snell’s laws of reflection and refraction.

To explain Fermat’s law, following the approach in [10], we introduce the concept of acoustic path length, defined by the integral

\[ \int_{p_0}^{p_1} n \, ds \]  

(2.35)

where \( n \) is the refraction coefficient and \( p_0, p_1 \) are two arbitrary points in space. Now we take two path functions which connect the two points, \( P_0 \) is the ray path and \( P \) a generic path which can be chosen arbitrarily. The relationship that describes the principle is the inequality:

\[ \int_{P_0} n \, ds < \int_P n \, ds \]  

(2.36)

which allows us to state the principle: the acoustic length of a ray is smaller than that of any other line between points \( p_0 \) and \( p_1 \) or, more simply said, a ray follows the shortest path possible (dependent on the refraction coefficient) between the source and the arrival point.

Geometrical Theory of Diffraction

As said in Section 2.1.2 diffraction is a very complex and hard to model phenomenon, which can be treated in different ways according to necessity. Incorporating diffraction into ray theory is the purpose of GTD. The concept at the basis of diffraction in GTD is that when a ray hits a wedge, the wedge becomes a secondary source which generates the new set of diffracted rays. A diffraction coefficient attenuates the intensity of the rays, and as in the general theory of geometrical acoustics, rays follow Fermat’s principle. [17]

A list of basic properties which can be adjoined with the ones at the beginning of Section 2.1.3 summarize some core concepts of GTD [9]:

- diffracted rays coming from a secondary source (wedge), have a radial distribution
- the power in the diffracted field is inversely proportional to the cross sectional area of the flux tube
• a diffracted ray is related linearly to the incoming ray at the diffraction point by a diffraction coefficient.
2.2 Simulation methods

The computational methods for simulating ultrasound propagation can be divided into two main categories: asymptotic techniques, which base themselves on geometrical optics and the geometrical theory of diffraction, and rigorous techniques, which use numerical methods on discretized spatial grids to solve the wave equation directly.[16, 13]

2.2.1 Ray Tracing

Ray tracing is a well known and widely used asymptotic technique which bases itself on the tracing and tracking of single rays which propagate perpendicular to the wave-front. Ray paths are computed by connecting source points to observation points, and then working backwards excluding rays whose paths end up in the geometrical shadows of objects. Reflection and diffraction phenomena are then computed for rays which encounter obstacles, using geometrical acoustics and geometrical theory of diffraction. A simple example of this process can be seen from a 2-dimensional cut of a simple setup in Figure 2.4 below.

![Figure 2.4: A direct ray path occluded by a sphere, and the alternative path with reflection off a wall](image)

The process is then carried out again for newly computed source points generated from reflection and diffraction.[13]
The method can be optimized in many different ways such as subdivision of space or considering rays in bunches. These methods all aim at lowering computation time by being able to exclude or consider more ray paths at a time, saving the cost of checking rays and paths one by one. These methods can also be coupled with parallelization in the computing to obtain very fast algorithms for large and complex scenarios.

The advantages of ray tracing are in fact the speed and the possibility to simulate propagation in environments with large objects and many reflections, since it is not necessary to have finely discretized spatial grids. The disadvantages of this method though come when precision is needed and the propagation medium becomes varied. Ray tracing in principle is not capable of handling cases with refraction and non-homogeneous medium. Moreover when scenarios become smaller and comparable to wavelength size even diffraction is not as easily simulated, because of the limits imposed by geometrical acoustic modeling, on which ray tracing algorithms are based.

2.2.2 Numerical Methods

Numerical methods all share the characteristic of solving the wave equation by numerical calculation. What differs are the ways the PDE, spatial domain, and time domain (in time-dependent simulations) are discretized.

Since these methods are based on discrete approximations of continuous derivatives and functions, there are some extra concepts to be taken into account when setting up simulations. These are numerical stability, diffusion, and dispersion, all influenced by the way the domains are discretized. For these reasons in order to obtain sufficiently accurate solutions, numerical methods impose limits on how coarse or fine grids can become, or how high the order of approximations can be. These limits reflect on computation time and flexibility of the methods.

Below some of the most commonly used numerical methods are listed and explained.

Finite Difference Methods

Finite difference methods are well suited for time-dependent problems and have a relatively simple implementation strategy.
The equation’s derivatives are discretized over a small neighbourhood of grid points. This can be carried out in different ways according to the order of accuracy required and whether the method is needed to be explicit or implicit, which has an effect on how the resulting linear system is solved.

An example of finite difference discretization of the wave equation can be obtained starting from the 1-D version of Equation 2.1

$$\frac{\partial^2 p}{\partial x^2} - \frac{1}{c^2} \frac{\partial^2 p}{\partial t^2} = f$$  \hspace{1cm} (2.37)

A spatial step $h$ and time step $\Delta t$ are used to split the domains evenly. This can be seen in Figure 2.5.

The second derivative in space at a specific time $n$, and space $i$ is approximated as

$$\frac{\partial^2 p_i^n}{\partial x^2} \approx \frac{p_{i+1}^{n} + p_{i-1}^{n} - 2p_i^n}{h^2}$$ \hspace{1cm} (2.38)

the second derivative in time is instead approximated as

$$\frac{\partial^2 p_i^n}{\partial t^2} \approx \frac{p_{i}^{n+1} + p_{i}^{n-1} - 2p_i^n}{\Delta t^2}$$ \hspace{1cm} (2.39)

Figure 2.5: Visualization of the discretized spatial grid and time steps. $n$ is the time index and $i$ the spatial index.
When these discretized derivatives are used to rewrite Equation 2.37 the final result in explicit time step form is:

\[ p_{i}^{n+1} = 2p_{i}^{n} - p_{i}^{n-1} + \left( \frac{c\Delta t}{h} \right)^{2} \left( p_{i+1}^{n} - 2p_{i}^{n} + p_{i-1}^{n} \right) + f_{i}^{n} \]  

\((\frac{c\Delta t}{h})\) is the Courant number, from which the CFL (Courant-Friedrichs-Lewy) condition for stability is derived. The condition states that the Courant number, which represents how fast a solution propagates in one time step compared to one spatial step, must be lower than a certain constant quantity to obtain stability for the method. The assumption is that initial conditions are known allowing for derivation of the first step and work forwards from there on.

Relating to the specific application of simulating ultrasound transducers used for environment detection on vehicles, a finite difference method to simulate the scenarios can be considered appropriate. In particular for small or very simple spatial domains, which require not so many grid points the method is of simple implementation and delivers a good level of accuracy, but this is rarely the case. A major drawback on realistic scenarios is the high computational cost of these methods, which would require large amounts of computing power, efficient parallelization strategies, and a fair amount of time, all factors which are not always preferred in industrial research.

Finite Element Methods

Finite element methods divide the spatial domain into a discrete mesh, but differently from finite differences, the grid is unstructured and adapted to the geometrical detail of the environment, this can in addition be done dynamically, so that the adaptation follows a time domain as well.
The equations are written in a weak formulation before being discretized, and are so solved numerically as integral equations. The advantage of this method is that the solution can be obtained by minimizing an error function which enables the accuracy of the solution to be controlled through mesh refinement. Although this method can be incredibly useful and accurate in presence of complex geometries and steady-state solutions, it is highly inefficient when needing to solve the full time-dependent wave equation on large spatial domains, making it ineffective for our cause.

Pseudospectral Methods

Pseudospectral methods are a valid alternative to finite difference methods, since they use a structured grid over which the governing equations are solved. The difference lies in the fact that instead of using function values to approximate local derivatives the whole field is decomposed into a finite sum of Fourier basis functions. This gives the advantage of being able to use a much coarser mesh, since the basis functions are sinusoidal. Thanks to this the method allows the sampling size to approach the Nyquist limit [1], which corresponds to having two grid points per wavelength. This is a big advantage in contrast to other numerical methods, which usually require 6-10 gridpoints per wavelength in order to obtain stable, accurate solutions. [2]

Another important advantage from a computational perspective is being able to operate in the frequency domain, where differentiation is
replaced with simple multiplication thanks to a property of the Fourier transform. Finally there exist very well optimized algorithms for calculating the Discrete Fourier Transform. However in contrast to all the pros there is an important limit to this method: for the approximation with a sum of Fourier basis to hold it is assumed that the wave field is periodic, or at least close to periodic, an assumption which cannot be made in all cases. Specifically linking the method to our application it would only be applicable in cases where the disturbance in propagation due to objects is minimal, in conclusion not the best option when dealing with many obstacles. In any case, being a numerical method, it shares the same major drawback in computational complexity for large environments with finite differences and the finite element method.

2.2.3 Computational complexity investigation

As pointed out in Section 2.2.1, a ray tracing method can give specific computational advantages when compared to a direct numerical method when simulating ultrasound in large spatial environments, characterized by a homogeneous medium. This can be heuristically shown by comparing average computational complexity of ray tracing algorithms to a numerical algorithm example and analyzing them in the light of our specific application.

An overview of ray-tracing algorithm complexity

In a naive interpretation of a generic ray tracing algorithm, the determination of ray paths can be said to have two steps for each observation point or direction: first the identification of the points that define the path, where reflection, transmission and diffraction occur, and then a shadowing analysis to decide which paths to exclude. Both these steps depend on the number of effects (e.g. deviations, bounces) that a user wishes to simulate. It has to be noted that both steps entail a search or checking algorithm, so they determine whether a facet or an edge determine a significant point for the path or not. If only one effect is computed the first step has complexity \((2n_f + n_e)\), where \(n_f\) is the number of facets (where reflection and transmission can occur, hence the multiplication by 2 of \(n_f\)), and \(n_e\) the number of edges (where diffraction can occur). This same step with an \(m\) number of effects simulated has complexity \((2n_f + n_e)(2n_f + n_e - 1)\ldots(2n_f + n_e - m + 1)\)
since at every step made the number of facets which need to be checked is reduced by 1.

Considering the shadow analysis in the same way it turns out that for one effect the $n_f$ total number of facets needs to be checked once, while for $m$ of them the complexity is $(n_f)^m$.\[13\]

Now by putting these two steps together and considering $m$-effect ray path computation for a total of $p$ observation points, the total comes out as

$$\left[\prod_{i=0}^{n-1} (2n_f + n_e - i) + (n_f)^m\right] \cdot p \quad (2.41)$$

The algorithm can then be optimized in various ways, with faster search algorithms and parallelization of the computing burden with distributed systems. Some of these techniques are analyzed in greater detail in Section 2.3.2, after a more specific version of a ray tracing algorithm has been presented.

Finally a numerical estimation of how many elementary operations it would take in an example case within our realm of applications can be made. The number of facets in a scene with multiple complex geometries, such as cars, poles and other vehicles, can easily be within the order of millions, so we take $10^6$ as reference. The number of effects/bounces which usually need to be considered is a maximum of two, whilst the number of observation points can be within the range of a few hundred, to many thousands, so we take $10^3$. Putting these numbers into Equation 2.41 yields a final result in the order of $10^9$ elementary operations.

**A numerical method complexity estimation**

Taking an example of a setup for a numerical method it is possible to get an idea of exactly how many grid points need to be evaluated by a machine to compute the whole acoustic field for a simulation. We estimate this by considering a 3-dimensional environment plausible for a parking function simulation in the automotive industry. Independent from the number of obstacles the spatial domain needs to be discretized. The spatial discretization can vary depending on the method used and is quantified by how many grid points per wavelength are necessary for an accurate simulation of the sound-waves. For this example we take 8 grid points, a fairly low value but still ac-
ceptable for a finite difference method, the one which requires the most elementary computation when calculating values at gridpoints. Therefore, considering a wavelength of around 6.8 mm at 50 kHz, and a spatial domain with dimensions $8m \times 3m \times 2m$, the number of total gridpoints is in the order of $10^{10}$ with $dx \approx 0.85$ mm.

Another quantity to be found is the number of time-steps, the size of which has an influence on the stability of the numerical method. This stability is ensured by the algorithm meeting the CFL condition:

$$u \frac{dt}{dx} \leq C_{max}$$

which taking a $C_{max}$ of 0.5, to ensure stability, velocity $u$ as 340 m/s, we obtain a stable time discretization with $dt \approx 1.25 \times 10^{-6}$.

If we were to take an estimated propagation time for sound to travel through the whole environment to be $5 \times 10^{-2}$ s, the total number of time-steps would be $4 \times 10^4$.

This would bring the total of grid points to be evaluated within the order of $10^{14}$ to $10^{15}$, which is a fairly large amount of computational power and time to spend when taking into consideration that, depending on the method, each gridpoint may need numerous elementary operations to be evaluated. An even bigger problem could be caused by the amount of memory required to keep track of all the values throughout the simulation.

Numerical algorithms can also be run on parallel architectures, and some work better than others depending on the case, for example a pseudospectral method would only need two gridpoints per wavelength, although requiring more time to evaluate each point.

**Comparison and considerations**

Although after these two reasonings an exact comparison cannot be made and a generically valid conclusion cannot be drawn, a series of considerations can be made alongside the numerical results shown above, based on the necessity of simulating ultrasound in the automotive industry (mostly traffic scenarios).

The ray tracing algorithm has the advantage of being easier to simplify when the output is needed to analyze a simple reflection phenomenon, compute the time propagation of the ultrasound field or measure it’s intensity in few designated points or areas. The ability to cut down simulation time simply by taking more or less observation points, and
more or less iterations of effects such as reflection can be hugely exploited. Another advantage is that ray tracing simulations are time-independent, so however large the environment is, the time it takes will still only be dependent on geometries, effects and observation points. For this reason a loss in accuracy and completeness of the final result can be considered a valid trade-off when having to simulate environments containing large objects.
2.3 Tool specific method

Of the methods mentioned in the above sections the one used in the tool which is evaluated and tested in this thesis is the ray tracing method. This was chosen as the best candidate for the application of simulating ultrasound propagation in large environments, within a reasonable amount of time, for traffic simulators. In the next sections an in depth study of the method will be carried out, and optimization methods for ray tracing algorithms are explained. The two following sections will be principally based on two articles: [13] and [4].

2.3.1 Ray Tracing in depth

A generic approach to a ray tracing algorithm can be divided into three steps, all of which are equally important for the final result:

- Geometric modeling of structures
- Obtainment of the ray paths
- Computation of the acoustic field (from which can be derived in sequence, energy, power, and intensity)

The first two steps of are described in detail in the following two sections.

Geometric modeling of structures

So as to be able to distinguish and model curved surfaces in the right way, geometrical structures are modeled best as parametric surfaces through the use of Non Uniform Rational Basis-Splines, better known as NURBS. The generic form of a NURBS surface can be expressed by the following equation, which is ultimately the tensor product of two NURBS curves [15]

\[
S(x, y) = \sum_{i=1}^{k} \sum_{j=1}^{l} R_{i,j}(x, y)Q_{i,j}
\]  

with rational basis functions:

\[
R_{i,j}(x, y) = \frac{B_{i,n}(x)B_{j,m}(y)\omega_{i,j}}{\sum_{p=1}^{k} \sum_{q=1}^{l} B_{p,n}(x)B_{q,m}(y)\omega_{p,q}}
\]  

(2.44)
the weights being $\omega$ and $Q$. $B$ are the basis functions, which are splines with degrees $n$ and $m$. The indices over which the functions are summed, $i, j, p, q$ represent the control points which together with the weights determine the shape of the curve/surface.

Both the planar surfaces and the curved surfaces modeled by the NURBS curves are subject to a pre-processing step. In this first step of the algorithm all surfaces are meshed in a non-uniform way depending on the amount of curvature, better captured by a finer mesh.[13]

**Obtainment of the ray paths**

Using a naive approach to the problem of obtaining ray paths two major steps are to be taken: first flash points must be determined, these are the points which define the ray path, then a shadowing test to determine occluded rays must be carried out. This process is much more complex, having curved surfaces which affect reflection and diffraction. The more detailed sequence of steps needed to obtain the paths with an $n$ number of reflections can be summarized as follows:

- firstly, taken the mesh composed of flat facets, a first path is computed using the laws described in Section 2.1.2 and Section 2.1.3
- the reflection points determined on the planar facets are taken as references for the real reflection points on the curves, and from these the closest point on the curves are deemed to be the initial points for the optimization process
- starting from the initial points in the previous step a Conjugate Gradient Method is used to minimize the ray paths by moving the reflection points along the curves until the minimal length path is found, this is known to be the real path of the ray thanks to Fermat’s principle (Eq. 2.1.3)
- to compute an $n$-th reflection on a curved surface, reflection $(n - 1)$ on a curved surface is taken as source point, and point of reflection $(n + 1)$ on a plane facet is taken as observation point

the steps are repeated in an iterative manner using the previous $(n - 1)$ reflections for each $n$-th point.
The above algorithm can be used in the same way to compute paths with diffraction phenomena. The only difference is that only edges of facets, and their corresponding curves, are taken into consideration.[13]

### 2.3.2 Optimization of Ray Tracing

The algorithm shown above describes the steps required to compute the paths of rays in a systematic way. When dealing with large environments with complex geometries a series of strategies can be implemented so as to reduce memory consumption and CPU time. These strategies include the subdivision of space in different ways, *a priori* selection of surfaces when determining where rays will encounter obstacles, and parallelization of the strategies for quicker computation.

#### Angular Z-Buffer and Parallelization strategies

The Angular Z-Buffer (AZB) is a form of space partitioning which divides the environment into ”anxels”, dependent on two angles ($\phi, \theta$) at the origin. This type of space partitioning is shown in Figure 2.7 below.

![Figure 2.7: An example of an anxel dependent on $\phi$ and $\theta$](image-url)
The information from this partitioning is then inserted into a matrix with the three dimensions: \( \phi \), \( \theta \), and the identification of the facets contained in the anxel. This is called the AZB matrix. The advantage of having an AZB matrix is being able to know that a ray defined by angles \((\phi_i), \theta_i\) can only hit facets within the anxel it finds itself in, consequently excluding many checks and cutting computation time.

The chosen size of the anxels influences a trade-off between memory and CPU time. The bigger the anxels are, the more CPU time is needed to check the larger amount of facets which will be contained in each one. If the anxels are too small though, one might encounter memory issues when storing very large AZB matrices.

The parallelization strategies discussed in [4] include the use of MPI paradigm, an OpenMP strategy with a common memory, and a fusion of the two, called hybrid parallelization.

The MPI based strategy entails assigning different surfaces to different nodes or processors, which compute each AZB matrix and then broadcast them to all other nodes, so that each one has the full matrix at the end.

The OpenMP implementation, however, makes use of a common part of memory on which all processors write their AZB matrices. This enables dynamic allocation of tasks for computing the matrices. This means that since computation times for AZB matrices are not uniform, processors which finish first and are idle can be handed other partitions of space on which to compute the same.

In [4] the authors determine that the best way of handling the parallelized computation of the AZB matrix is to use a hybrid parallelization, combining the two strategies, in which groups of processors, nodes, use shared memories with OpenMP, whilst the communication between nodes is done using the MPI strategy.

A combination of AZB and Space Volumetric Partitioning

The Angular Z-Buffer can have a drawback in large environments since the further an anxel gets from the source point, the bigger it gets, leading to a possibly too large amount of facets per anxel. To adjust to this possible issue the AZB can be combined with a Space Volumetric Partitioning (SVP), which divides space into "voxels", parallelepipeds of various sizes dependent on necessity. The scene described by the SVP
matrix (similar concept to the AZB matrix) is obtained dividing space recursively until a certain limit is hit in every part of the environment. This limit can for example be the number of facets per voxel, which means the SVP matrix can be very inhomogeneous, due to meshes having the same property. The SVP matrix is computed before the AZB one, so that the AZB matrix can be computed only for facets within the same, or neighbouring voxels, instead of for the whole environment. These subsequent selection processes aim at minimizing computation time by allowing the exclusion of large amounts of environment from the computation of ray paths.

**A* heuristic search method**

A final method of optimization presented in [13] is an A* heuristic search method which enables one to further discard facets from the computation of a ray path. This is done by assigning weights which estimate the contribution that a facet can give to the reflected or diffracted field. A threshold is then decided to determine when a facet’s contribution is not significant, and can be left out of the computation.
Chapter 3

Tool Evaluation

3.1 newFASANT

The newFASANT Ultrasound Module is a powerful simulation tool that computes several parameters of systems that use ultrasound waves by applying Geometrical Theory of Diffraction. The US module, in principle very useful for automotive applications, includes the following possibilities:

- Computation of sound intensity and pressure field at any given point by adding contributions of the paths ultrasound takes from the source to each observation point.

- Computation of time-domain simulation of the delay process at a given point. This allows the visualization of the observed sound pressure at a given point as a result of the propagation in time.

- Computation of the coupling between an ultrasound source and an ultrasound receiver, so the measurement of the energy emitted by the source that is absorbed by the receiver.

- The analysis of the Doppler effect when the ultrasound source, objects, and observation points are moving with respect to each other.

The US module also provides a tool to design ultrasound patterns, generated from a spherical source and defined by angular intensity and frequency. These ultrasound files are saved as .DUS files and can be loaded and used in multiple simulations.
In addition the US module offers the possibility of integration with external tools with a closed-loop simulation support. These simulations are run using the command line with parameters defined from text files.

The relevant aspects of the tool which interest us in our evaluation for automotive simulations, and which we will test, include mostly the computation of sound pressure and intensity, and the accuracy of such predictions related to the simulation of ultrasound sensors. In addition we will briefly look into the possibility of using the closed-loop simulation option for integration with traffic simulators.
3.2 Evaluation Method

The methodology of the tool testing is laid out and divided into three different sets of guidelines each specific for the type of tests carried out. The simulations are first tested and evaluated against cases with theoretically known outcomes, with simple scenarios or more complex ones where simple, trackable variations are made. Then the tool is tested against another simulation software which uses a rigorous numerical method, so as to better understand differences, advantages and disadvantages of its usage. Finally a performance evaluation is carried out, where the tool is tested against its own limits. It is important to research where they lie and what that implies.

3.2.1 newFASANT US module vs. Theory

When comparing the software tool to relatively simple cases where the results can be computed from theory as well as using the software the first limit to establish is an error percentage tolerance. The dimensions of the scenarios which will be simulated by the tool when it is fully functional must be taken into consideration. 0.05 and 3 meters can be considered the lower and upper limit of the distances that a US sensor needs to measure for a parking function on a vehicle. An acceptable limit of simulation error can be established between 0.05 and 3 millimeters, which gives an error percentage tolerance of 0.1%. This can be extended to all quantities observed during the tests, such as intensity, frequency, and phase, so as to have a consistent general rule for validation.

Another evaluation technique taken into consideration is the conservation of symmetries, of constant values, and the correct behaviour of quantities which follow a certain function (e.g. inverse-square law for intensity decay). This comprises the fact of values remaining consistent when for example whole scenarios go through size scaling especially when more complex geometries are involved.
3.2.2 newFASANT US module vs. MATLAB/k-Wave

Within the comparison of the tool with another simulation software based on different methods and developed for slightly different purposes it is crucial to select a fair testing ground. The evaluation is carried out on a scenario which both tools can sustain but where maybe one of the two will perform better. The criteria taken into consideration are accuracy and speed, preferably together so as to be able to discuss trade-offs and advantages of the methods based on different necessities.

3.2.3 Performance

The performance evaluation investigates the limits imposed by different factors: mesh resolution, number of observation points, what kind of simulations are carried out (e.g. if diffracted rays are computed as well), number of CPU cores used. The variation of these factors determines changes in accuracy and speed of the simulations. Obviously limits of non-computability have to be acceptable and real case scenarios should be dealt with by the software over a computational time-scale which is fitting for industrial research purposes.
3.3 Results

3.3.1 newFASANT US module vs. Theory - Benchmark tests

Single ray intensity

To begin the evaluation process two very simple tests are performed, aimed at verifying the simplest cases possible, to give an idea of the methodology and the accuracy in the most obvious scenarios.

For both benchmark tests a constant spherical wave source is used. An intensity level of 1 dB at 1 m distance from the centre, with a reference intensity in air of $I_0 = 1 \text{pW/m}^2$. A frequency of 50 kHz is set.

In the first test intensity of ultrasound rays, which follows the inverse-square law, described in Equation 2.7, is tested by putting observation points along a 1 m line starting from the source and placed in radial direction. In this way it is easy to observe the simulation of the intensity decay. The intensity and it’s decay are computed specifically by solving the following proportion:

$$
\left( \frac{I(W/m^2)}{1.2589 \times 10^{-12}(W/m^2)} \right) = \left( \frac{1(m)}{\text{distance}(m)} \right)^2
$$

1.2589 $\times 10^{-12}(W/m^2)$ is the equivalent to 1 dB using the stated $I_0$ as reference.

A data comparison verified that a high level of accuracy is maintained in simulating the intensity decay along a straight line: the sampled values fit the theoretical curve perfectly and average error is in the order of $10^{-22}\text{pW/m}^2$, so $10^{-14}\%$, negligible. The behaviour can be seen in Figure 3.1 where the only noticeable difference between simulation and theory is the theoretical intensity curve continuing to climb when the distance approaches 0, since it is modeling an ideal point source.
Reflection from a disk

The second benchmark test carried out is still based on an analysis of intensity values. This time though also reflected rays are studied, so attention is given to correct geometrical reflection as well.

A fully reflective disk is placed at 1 m distance from the source, the disk has a radius of 1 m. Observation points are placed on a 2 m by 2 m square grid which lays on the xz-plane, with the source at it’s center. The setup can be seen from Figure 3.2(a), along with the rays traced.

The expected behaviour is for the rays coming from the source point to bounce off the disk and back to the observation points. Since the disk is fully reflective intensity should decay by the inverse-square law, Equation 2.7, dependent on the length of the trajectory of the reflected ray. To compute this intensity theoretically the geometrical paths of the rays are calculated with respect to the observation points using simple geometrical equations such as Pythagoras’ theorem: from the source to each observation point the trajectory of the ray consists of the two equal
sides of an isosceles triangle with its vertex placed on the disk surface. The length of the path depends then also on the angle of reflection off the disk, which is equal to the angle of incidence.

The results from this simulation turned out to be very accurate, as was wished, allowing to create an even more solid standard to base more stressful tests on. The average error upon intensity simulation was, as in the first benchmark test, in the order of $10^{-22}\text{pW/m}^2$, which demonstrates high accuracy. In addition, as can be seen in Figure 3.2(b), geometrical properties were properly simulated since the contours for concentric circles of decreasing intensity. Figure 3.2(a) shows the visualization of the ray tracing.

A simple remark that can come with this simulation is that the quantity and layout (e.g.: rectangular grid or concentric circles) of the observation points has a big effect on how the "intensity distribution" is simulated on the xz-plane. An example in the intensity values along the a diameter of the circle on the xz-plane can be seen in Figure 3.3 where the theoretical prediction is compared to the simulation results.

The result shown proves the accuracy of the simulation.

![Figure 3.2: Different visualization methods for simulation results.](image-url)
3.3.2 newFASANT US module vs. Theory - Reflection from a spherical surface

This test takes into consideration reflections off a more complex geometry. The surface off which the rays bounce is a sphere placed close to a constant source, the sphere has its center at a 2 meter distance from the source, a cone shape at the origin in the figures, and has a radius of 1 m. The observation points are placed on the same plane as the source, so as to capture the rays which bounce back from hitting the fully reflective sphere. The interest, and difficulty of taking a sphere as reflecting object is the curvature of the surface and the impact it has on the scattering of intensity in the reflection due to the curvature. This turned out to be the main challenge when building the theoretical model to compare the simulation to. In Figures 3.4(a) and 3.4(b) there is a visualization of the setup and of the ray tracing in the simulation. The observation points in this example are placed along a line, but tests were also carried out with individual points, or 2-dimensional grids co-planar with the source.
CHAPTER 3. TOOL EVALUATION

(a) Test setup, observation points on dotted line
(b) Ray tracing

Figure 3.4: Test with reflection from sphere

Theoretical considerations

To be able to test the simulation rigorously against a theoretical model, some extra specifications need to be made latching on to Section 2.1.3. The theoretical model is based on the specific case set up for the simulation. Constants such as the distance between sphere and source, and the radius of the sphere, are used so as to simplify the model by reducing the number of variables.

The first characteristic to model for comparison with the software is the reflection angles, since analytically it is possible to compute with an ideal spherical surface for each ray. A 2-dimensional section of the setup is taken as can be seen in Figure 3.5. Relations between the different angles are identified starting from the angle at which a ray leaves the source in $O$. The properties used include the law of reflection, and Snell’s law, both talked about in Section 2.1.2, specifically Equation 2.9. In addition to these properties, to complete the equations the law of sines is used multiple times along with the notion that the internal angles of a triangle add up to a total of $\pi$. The final relation, Equation 3.6, allows to compute the distance from the source at which the ray hits the plane on which the source is laying, which is where the observation points are, all this starting just from the notion of the angle $\theta$, as is seen in Equation 3.1. The set of equations relating all the angles and distances is the following:
\begin{align*}
\lambda &= \pi - \arcsin(2\sin(\theta)) \\
\gamma &= 2(\pi - \lambda) \\
\rho &= \pi - \left(\frac{\pi}{2} - \theta\right) - \gamma \\
\beta &= \pi - \theta - \lambda \\
b &= 1 \cdot \sin(\beta) / \sin(\theta) \\
d &= b \cdot \sin(\gamma) / \sin(\rho)
\end{align*}

The other important characteristic on which the theoretical model has to focus is that of the intensity of the reflected rays, which is affected by the fact that they bounce off a curved surface. To do this we refer to
the property in Section 2.1.3 which states that the power, from which intensity can be derived, is conserved within a flux tube. This is shown in Equation 2.13 and Figure 2.2. The relationship between sound power and intensity is governed by the following equation

\[ I_0 = \frac{W_0}{A_0} \]  

where \( W_0 = 1 \, \text{pW} \), \( A_0 = 1 \, \text{m}^2 \). In absence of obstacles this brings to the inverse square law (Eq. 2.7), which though breaks down when there is reflection from objects which have a scattering effect. An example of the modified behaviour and why the intensity “spreads out” is given in Figure 3.6.

Figure 3.6: Flux tube reflected from spherical surface
Simulation and results

The results from this kind of simulation, where we want to verify the accuracy of reflection angles from a curved surface, are well analyzed by taking a feature of the software which is very useful for automotive applications: distance and propagation time. These two quantities were first computed theoretically using the equations 3.1 to 3.6 and the speed of sound of 340 m/s, and then checked against the values from the simulation. The accuracy of the software turned out to be close to perfection. Some examples of ray path distances compared with theory can be seen in Table 3.1 below. Accuracy in time propagation turns out as a simple consequence of the distance value.

Table 3.1: Propagation distances of rays as a function of observation point distance from source.

<table>
<thead>
<tr>
<th>obs. point position</th>
<th>theoretical distance</th>
<th>simulated distance</th>
<th>relative error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1 m</td>
<td>2.003748177420365</td>
<td>2.00374512917574</td>
<td>3.0482e-06</td>
</tr>
<tr>
<td>0.2 m</td>
<td>2.01490963743175</td>
<td>2.01490779309612</td>
<td>1.8443e-06</td>
</tr>
<tr>
<td>1.850715208989160 m</td>
<td>2.957781121941377</td>
<td>2.95778112194138</td>
<td>2.6645e-15</td>
</tr>
</tbody>
</table>

Below are two figures from the output of a simulation where observation points are set on the plane where the source sits.

(a) Ray tracing result for an observation plane with 961 points  
(b) Intensity heat map of reflected rays plane

Figure 3.7: Visualizing simulation results of reflection from sphere.
3.3.3 newFASANT US module vs. MATLAB/k-Wave

In this section simulations focusing on analyzing diffraction effects are carried out using the newFASANT US module and comparing it to another simulation tool: the k-Wave toolbox in MATLAB.

k-Wave

k-Wave is an open source acoustic toolbox for use in MATLAB and C++. It uses the \( k \)-space pseudospectral method and solves an extended version of the first-order continuum equations. The method is an improvement to generic pseudospectral methods, already introduced in Section 2.2.2. It introduces replacement expressions for the temporal derivative, obtained by comparing with exact solutions of a homogeneous wave equation, in order to have exact numerical solutions for arbitrarily large time-steps. The toolbox enables to compute simulations in 1-D, 2-D, and 3-D, modeling pressure sources and allowing to setup arbitrary detection areas to record acoustic pressure, particle velocity, and acoustic intensity. [18, 2]

More detailed information can be found in the k-Wave user manual.

The specific equations which are discretized and solved with the \( k \)-space method are the following [2]:

\[
\frac{\partial u}{\partial t} = -\frac{1}{\rho_0} \nabla p \tag{3.8}
\]

\[
\frac{\partial p}{\partial t} = -\rho_0 \nabla \cdot u \tag{3.9}
\]

\[
p = c_0^2 \rho \tag{3.10}
\]

which combined can give Equation 2.1, the acoustic wave equation.

Diffraction through slit

The case set up to carry out this test is the one of diffraction through a small slit in a wall-like geometry. This is a standard example for diffraction which can be difficult to model especially when the slit width approaches the size of the wavelength. For the ray-tracing algorithm this is basically aimed at verifying its limits since by definition it is not meant to handle these types of cases, which need more precise and
Two cases in total are set up, one in which the slit is large enough for the ray-tracing software to give reasonable results, the second with a width comparable to the wavelength in which a more tricky to model behaviour of the waves comes into play. For both software tools the result is analyzed on a 2-dimensional plane. In newFASANT this means taking observation points on a line incident to the direction of the waves scattered from the diffraction effect, whilst still having a 3D simulation in all. In k-Wave instead it is simply a 2D simulation, which makes it easier to reduce the computational burden.

The first parameter to consider is the wavelength, which with a 50 kHz frequency wave traveling at 340 m/s is of 6.8 mm. In the first test the slit is 20 cm wide and placed at a distance of 40 cm from the source. Observation points are set at 60 cm from the source along a 40 cm line. In the second test the slit has a width of 5 mm, placed at a distance of 2.5 cm from the source. The observation points are still in a line formation which is 5 cm long and 5 cm away from the source, 2.5 cm away from the slit.

The source, which is a point with an initial pressure output of 20 Pa, was the most challenging aspect to “even out” between the two tools, since the way of defining the source is completely different. This was managed by taking the pressure registered after 1 spatial grid-point, 0.1 mm form the source, in the k-Wave simulation and then computing from that the intensity at 1 m using Equation 2.6. This then was used to define the constant source in the newFASANT simulation tool.

Example images from the 2 simulations can be seen in Figures 3.8 and 3.9 below.
CHAPTER 3. TOOL EVALUATION

Figure 3.8: Result visualization from ray-tracing simulation.

(a) Diffracted rays seen passing through slit from above
(b) Direct and diffracted rays passing through slit in wall

Figure 3.9: Result visualization from numerical method simulation.

(a) Wave
(b)

Results

The results from the simulations are compared and analyzed by plotting the intensity levels measured at the observation points. The challenge in doing this is posed by the fact that the newFASANT simulation is not time-dependent. This means having to take the maximum intensities measured at the observation points throughout the time-dependent numerical method, and comparing them with the intensities coming from the direct+ diffracted rays in the ray-tracing simulation. The end results are plotted in Figure 3.10 below.
A first glance at the figures immediately tells about how differently the simulations are carried out and exposes the fact of diffraction being handled in two completely different ways starting from the underlying equations: GTD for ray tracing (Section 2.1.3) and the wave partial differential equation (Section 2.1.1) for the pseudospectral numerical method.

Taking first a look at Figure 3.10(a) it is noticeable that some sort of consistency is maintained. The fact that ray tracing has to split the effects of direct and diffracted rays to then recombine their contribution, whilst the numerical method simply computes the pressure propagation does lead to suggest that the the plot represented by the line in blue is more accurate. It has to be taken into consideration that even the numerical method will not be completely faithful to reality, because of numerical dispersion, dissipation, and the difficulties of defining fully reflective boundaries for the wall. Overall the result shows that the ray-tracing method in newFASANT can be a valid alternative to a numerical simulation which on the same machine took 16 times as long to solve.

From Figure 3.10(b) a different type of conclusion can be drawn. It is fair to remind that as said in the above paragraphs this test is meant to exceed the limits of a ray tracing algorithm since the slit has a width comparable to the wavelength. For the kind of effect simulated the results from the ray tracing are far too simplified and the intensity level is computed regardless of the wavelength being affected by the sound passing through the aperture. In this case it is fair to say that the more accurate simulation is definitely given by the numerical tool, as clearly expected.

The overall conclusion that can be drawn from these sets of simula-
tions comprising diffraction is that newFASANT US tool behaves well as a high frequency ray simulator, when the wavelength is still negligible, but obviously breaks down with very small details and geometries. However these cases are nearly never needed when simulating traffic and large obstacles in automotive industry.

3.3.4 Performance

Mesh variation

Mesh resolution can be an important factor to consider when building simulations since the amount of memory required to store data points and compute upon them plays a big part in the variation of computation time. Logically the less fine the mesh is, the less points need to be considered, less memory is used, less computations are carried out, and the simulation is faster. But the question is: when does this increase in speed start to effect the accuracy and output of the simulation? The answer is, almost immediately. In fact, depending on the kind of setup and the output you look for, even small changes to the mesh structure can determine differences in output, since ray tracing is sensitive to changes in the inclination of surfaces. For example, if two small sections of a big curved surface were to be “merged”, the angle with respect to the origin would change, meaning that the rays bouncing off that “new version” of the surfaces would reach different observation points, or maybe even none. To show this dependency two tests were set up:

- A simple example where significant mesh reduction of a very simple geometry, specifically a cylinder, is carried out.
- A series of simulations where the mesh of a car is reduced first by a factor of 0.25 and then by 0.0625.

The series of simulations we use to exemplify the effects of mesh reduction have a simple setup: a cylindrical shape, which could be conceptually linked to a lamp post in parking simulations, is placed 15 cm away from a constant source, and 500 observation points over 50 cm are placed aside the cylinder. Three versions of this case are simulated, the first with a cylindrical shape provided by the software geometry maker, the second with a 16-sided prism which reduces by far the number of points present in the mesh representing the cylinder,
and the third is an even more coarse approximation in the form of an octagonal prism. These setups, which are purposefully extreme cases of mesh reduction, arguably a change in shape, can be seen in Figure 3.11.

From the set of figures the reflected rays which are traced from the source to the observation points are observable. The evident difference is the significant reduction of the amount of rays which are actually reflected into the observation points in Figures 3.11(b) and 3.11(c) with respect to Figure 3.11(a).

![Figure 3.11: Reflection off cylindrical mesh and coarse approximations](image)

Another example of the dangers linked to large mesh reduction can be seen in Figure 3.12(b), where intensity level of arriving rays is plotted as a function of the observation points. We can see here that the reduction of curvature linked to simplification of the mesh elements eliminates the scattering effect, so rays bouncing off the geometries with a coarser mesh have an imprecise intensity level measure. This is specifically due to the fact that when a mesh is simplified it is no
longer represented by NURBS surfaces, but by plane facets of larger dimension. This is a problem that can occur with all curved surfaces because the meshing from NURBS curves is conducted inside the tool, as explained in Section 2.3.1. This leads to the fact of needing a .NUR file as input for curved meshes if the user wishes to maintain intensity level measurement accuracy. In the same set of plots Figure 3.12(a) shows the points hit by reflected rays along the observation axis, it is a clear way of seeing how many reflections are missed by reducing the mesh thickness.

Having analyzed the simple example we move to the more significant set of tests, still grouped as 3 identical setups, in this case a car mesh, which is reduced twice. The car is placed perpendicular to the y-axis, with it’s left side facing the origin. At 2 m from the left back wheel, in the origin, there is a constant source, and 200 observation points are put along a line which sits on the x-axis, parallel to the car. The car mesh was reduced in a software environment called MeshLab with a Quadric Edge Collapse Decimation method, which enables to reduce the number of mesh grid points by a certain percentage factor. The analysis on the results is carried out in the same way as for the simplified example above. In Figure 3.13 there are two snapshots, 3.13(a) on the left is the setup, and 3.13(b) on the right shows the ray tracing after the simulation was carried out. Both images are taken from the case where the full mesh was utilized.
The section of the car off which the rays coming from the source are reflected is geometrically rather complex, due to the presence of a wheel rim and the tire. For this reason the data output from the simulations is of more complex form than the one in the cylinder example, so harder to analyze and comment. In Figure 3.14 three plots are shown starting from the same basic data. All three are a plot of the intensity level of the reflected rays, measured at the observation points placed along a 2 m long line.

Firstly in Figure 3.14(a) the data is plotted “raw”, so the intensity levels are visualized point by point. A series of intervals in the plot contain overlapping points, but there remain many differences between simulations: the red points which can be used as reference have a higher numerosity, and evidently represent more details of the geometry off which the rays are reflecting; the blue and green points coming from simulations using simplified meshes are more “clustered”, this gives an idea of the geometry being simplified. As for the values we observe from the y-axis limits that nearly all data stays between $-2 \text{ dB} -15 \text{ dB}$. One small outlying cluster at $-20 \text{ dB}$ is observable in the simulation where the mesh is reduced by a factor of $0.25$. Upon inspection of the ray tracing output the cluster of data is a bunch of rays which get reflected from a part close to the front of the car which has acquired a strange angle due to the re-meshing process.

The plot in Figure 3.14(b) shows a linear interpolation of the data points from the plot in Figure 3.14(a). This helps give a better idea of where the data sets have similar values and where big the differences lie. For example, between 1.4 and 2 meters all sets of data have a series of peaks which come in correspondence to rays reflected off the wheel rim. These peaks though do not overlap due to the fact rays are reflected at different angles since the mesh has a different structure. This can be ignored or it can be a disadvantage, depending on what purpose
the simulation has. A third plot has been drawn up for completeness in Figure 3.14(c): it is a cubic interpolation of the different data sets. It is to be observed differently from the two previous plots because the cubic interpolation can tend to over-fit the actual shapes of the curve, but can also suggest possible insights of undetected details in the geometry due to too large distances between observation points. What can be inferred though is that most of the data sits within a pretty limited interval and it suggest that the expected value of the different data sets are similar. In fact if the expected value of the intensity levels in each simulation is taken we obtain the results shown in Table 3.2, which confirm the similarities, especially between the full mesh and the reduction by a factor of 0.25.

Table 3.2: Average intensity level of reflected rays for each simulation

<table>
<thead>
<tr>
<th>mesh</th>
<th>average intensity level (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>full</td>
<td>-11.5446</td>
</tr>
<tr>
<td>0.25 reduced</td>
<td>-11.3995</td>
</tr>
<tr>
<td>0.0625 reduced</td>
<td>-9.0414</td>
</tr>
</tbody>
</table>
The conclusion that can be inferred from these last plots, together with the cylinder example, is that mesh reduction, although being a powerful form of reduction for computation time, is a fairly dangerous tool to play with when using ray tracing. From a theoretical point of view this can be deduced from the dependence of reflection on the normal of each part of surface. These considerations though are relatable to precise measurements in complex geometries. If the purpose of a simulation were to be to gage for example an average distance on a large body (e.g. the back of a car) then mesh reduction could turn out to be a useful option for obtaining results much faster. The speed-up linked to the re-meshing process can be seen in Table 3.3 below.
Table 3.3: Computation time for each simulation.

<table>
<thead>
<tr>
<th>mesh</th>
<th>computation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>full</td>
<td>5 h 10 m 7 s</td>
</tr>
<tr>
<td>0.25 reduced</td>
<td>10 m 40 s</td>
</tr>
<tr>
<td>0.0625 reduced</td>
<td>1 m 53 s</td>
</tr>
</tbody>
</table>

The measurements in Table 3.3 were obtained by running simulations on a machine with an Intel(R) Core(TM) i7-6820HQ CPU unit and 32 GB of installed RAM memory. This demonstrates that the simulations can be carried out on a portable workstation with ease.
3.4 Bugs, errors, and fixes

During the course of the evaluation, since the US module is still in the final stages of development, coming across anomalies of different nature was a natural part of the process. These were spotted and signalled to the developers, who promptly provided corrected versions of the tool.

The majority of the bugs found were linked either to the ray-tracing algorithm functioning poorly within specific cases, or to geometries and boundaries not interacting with the rays in a correct physical manner. A selection of cases which were dealt with is shown to exemplify.

Reflection bug

The setup of the case comprised a sphere, a plane leveled with the source, and some observation points directly above the source. The expected outcome of the simulation was for the rays to reflect off the sphere. Some of the rays would reach the observation points directly, while others would reflect off the plane as well before reaching the observation points.

As can be seen in Figure 3.15(a) a bunch of rays were apparently bouncing off thin air before being able to reach the plane. This error would not only cause the visualization and ray paths to be wrong, but also simulating the wave traveling a shorter distance and so miscalculating the intensity levels. This anomaly was fixed and the correct behaviour can be observed in Figure 3.15(b)

![Figure 3.15: Reflection off sphere and plane](image)
**Boundary bug**

The case setup had the source positioned inside a box, with a small circular opening in the top. Observation points were placed right above the hole in a circle with larger radius with respect to the hole in the box. This was done so as to obtain diffracted rays in radial symmetry. The outcome was almost correct. As seen in Figure 3.16(a) the diffracted rays had the expected path, but some rays (in green) were reflecting off the inside of the box towards the outside. This signaled a problem linked to the behaviour of rays at edges or boundaries of geometrical shapes. The fixed case is shown in Figure 3.16(b), with only diffracted rays propagating since there is no possible path for direct or reflected rays to reach the observation points from the source.

![Figure 3.16: Diffraction through hole in box](image)

**Diffraction intensity anomaly**

In this case the source was placed inside an open topped cylinder, as seen in figure. Observation points were placed both on the rim of the cylinder and an outer circle, this was done to aid visualization of diffracted rays over the top edge. The expected outcome was to see diffracted rays in radial symmetry, which was observed, and for them to have equal intensity when reaching the observation points. The simulated intensity though did not measure constant, as could have been expected, but had a sinusoidal behaviour, this can be seen in the blue curve in Figure 3.4. The red curve in the figure is the intensity measure after the bug was fixed, which maintains itself constant, because of the radial symmetry, as expected. Figure 3.4 shows the case setup from which it easy to deduce the properties that the rays should have.
Figure 3.17: Intensity levels plotted at observation points placed equidistant from the source

Figure 3.18: Rays diffracting off the edge of an open topped cylinder
Chapter 4

Conclusions

This chapter is aimed at framing the set of conclusions which can be drawn from the analysis of the theory and algorithm study, summarizing the results from the testing and evaluation of the tool in the light of its primary application, and finally suggesting where future work and future research can be carried out both in academia and in the industry.

4.1 Algorithms and theory

The most interesting aspects of the theoretical study carried out originates from the possibility of modeling ultrasound in an ambivalent fashion. The fact that ultrasound can be seen both from the classical wave point of view and also as a series of rays, thanks to its high frequencies, translates directly to the vast array of simulation methods. The advantages and disadvantages of ray tracing compared to direct numerical methods are extremely dependent on the kind of scenario simulated and in what form the results are needed:

- ray-tracing is best for large environments, where the results do not need to be time-dependent and approximations such as homogeneous media and simplified geometries can be made
- numerical methods are best for smaller environments, where results are required to be very specific, dependent on physical properties of medium and obstacles, and the way the sound propagates in time is significant

A practical example of this concept can be seen in Section 3.3.3, where a numerical tool and newFASANT are compared on a very small 2D
space simulation, which shows the limits of ray-tracing. As for the specific applications treated within the thesis ray-tracing turns out to be the best option from an \textit{a priori} theoretical point of view also due to computational complexity. The comparison carried out shows clearly that numerical methods need a higher computational power because for a start they are simply more rigorous and slower algorithms, and secondly the only way of optimizing them and speeding them up is parallelization. This would mean having to always run simulations on a cluster. Ray-tracing on the other hand, as well as parallelization has a number of other techniques with which it can be sped up and optimized, most of these based on space partitioning.

### 4.2 Software validation

The results from the tests set out to evaluate the newFASANT US simulation module turn out to largely respect the theoretical assumptions. For simple scenarios and geometries the tool is extremely quick and accurate, intensity fields are simulated correctly and the effects of reflection, diffraction and transmission are precise in the far-field. The limits of the software are as expected present when dimensions of obstacles reach wavelength dimension, this though is due to the background assumptions of having a high frequency ray breaking down. Relevant to the application of simulating traffic scenarios the advantage of the tool is being able to reduce the time of a single simulation by a careful choice of observation points depending on the data which needs to be retrieved. All in all it is a valid tool for use in a traffic simulator, specifically tailored for sensor evaluation.

### 4.3 Future work

Ray tracing algorithms are still deeply in development, and the study of the various optimization methods, especially how to combine existing ones in the best way possible, is a key topic in the advancement of these methods, so as to be able to exploit even more the computational advantages.

From a more applied and practical perspective after having validated the tool the most interesting topics to investigate in the near future are
how to exploit the potential of the tool at its best, especially within the post processing of data from simulations, and how to best integrate it with existing vehicle simulators and virtual testers.

Technical matters that can be escalated to future work concern firstly the actual implementation of a script for closed-loop simulations, this could be done in C, C++ or MATLAB for a Simulink block. A second specific matter, referring back to Section 3.3.4 concerns imported mesh files representing curved surfaces: the software’s built in meshing step manages curved surfaces only coming from .NUR files containing parameters defining NURBS curves (Section 2.3.1), else-wise it treats the mesh only as a series of flat surfaces, this creates problems when computing intensity fields. A way of creating these files could be implemented by fitting NURBS curves on meshes from existing files, writing a script for such a job could be helpful when large car meshes need to be imported and simulated properly in the tool.

Within this thesis most of the evaluation and analysis of the Ultrasound simulation tool is carried out with a specific kind of application in mind: simulating ultrasound propagation within traffic simulators and vehicle simulation programs so as to test sensors and functions such as assisted and automated parking. The idea is that a program which is simulating a certain traffic scenario, is able to access the US simulation module in closed-loop real time, and retrieve data on the propagation of ultrasound in the virtual environment. A basic example is that of a car using ultrasound sensors to assess the width and depth of a parking slot before performing a manoeuvre, at this point the traffic simulator would access the US module, perform a simulation with the data from the virtual environment, and then use the results to reproduce the car behaviour faithfully.

newFASANT has function that enables it to be called from command line to compute scenarios in closed-loop simulations. This function works from the user (in our case the traffic simulator) providing a .NFIL file containing the details of the scenario which needs to be computed. This file contains information such as simulation parameters, geometry details, ultrasound sources, and observation points. The file can then be retrieved and run from command line in newFASANT, making this process accessible to a program running a closed-loop simulation.
Bibliography


