TECHNISCHE UNIVERSITÄT ILMENAU

Fraunhofer IZFP

Testbed for Surface Detection with Ultrasound

Master Thesis in Communications and Signal Processing

submitted by

Sudhanshu Apte

born on 1st March, 1995 in Panvel

in the
Electronic Measurements and Signal Processing Laboratory

Department of Electrical Engineering and Information Technology
Technische Universität Ilmenau

Responsible Professor:	**Univ.-Prof. Dr.-Ing. Giovanni Del Galdo**
Supervisor:	**Dr.-Ing. Florian Römer, Fraunhofer IZFP**
Supervisor:	**M. Sc. Eduardo José Pérez Mejía, Fraunhofer IZFP**
Submission Date:	**10th Sept, 2023**

Acknowledgments

First of all, I would like to thank my family for their continuous support. Without that support I can't imagine completing this thesis.

Second, I would like to thank Prof. Del Galdo and my supervisors Eduardo and Florian for their guidance and support throughout my tenure as a research assistant and master's thesis student. Especially I would like to express my deepest gratitude to Eduardo for helping with the most trivial doubts in my thesis. I would also like to thank Jan for giving me the first opportunity to work as a research assistant in Ultrasound group. I consider myself extremely fortunate for getting the opportunity to work in the lab with everyone else.

Finally, I'm grateful to my friends here in Ilmenau and in India for making my life easier and happier inspite of raging corona crisis. I will definitely miss the summer evenings spent with my friends at Mensa lawn drinking beer and listening to music.

Abstract

Ultrasound Non-destructive testing involves the detection and localization of possible defects. The measurement setup involves a positioner controlling the position and orientation of a transducer and a specimen under test. If the positioner is not aware of its initial orientation aiming over the specimen, then calibration is required which involves finding an optimum orientation of a transducer. In order to gain any information about the interior of the specimen, an ultrasound must travel through the specimen at an optimum angle such that the reflected signal has maximum energy as the reflectivity is angle-dependent. In order to tackle the time-consuming nature of hardware-dependent calibration, this thesis explores techniques to emulate physical hardware setup with a ray tracer and speed-up measurement operation for faster prototyping of calibration methods.

The presented ray tracer model consists of a mathematical model to simulate measurements and geometric elements emulating a physical measurement setup. The model works under a discrete setting, replacing, the ultrasound wave fields with a discrete set of rays assuming the frequency of operation is high and the medium is homogeneous. The amplitude and time of flight of each received ray over the sensor area are used to modulate pulse shape, which further modulates a carrier, and the total field is calculated by summing all the scaled, time-shifted copies of the signal. Ray tracer allows transmitting pulse shape and opening angle to be programmable, which is not often the case in real-world settings.

The virtual measurement setup in place allows analysis of calibration algorithms by tuning different calibration parameters in less time, also, giving an indication of the combination of parameter values that might work in a real-world setting. In this work, stochastic approximation algorithms are investigated by setting up different test scenarios to evaluate performance metrics of convergence speed, accuracy, and robustness. Calibration algorithms automate calibration and reduce significant measurement time as compared to manual calibration, which involves moving the transducer to each angle and waiting for the measurement. The calibration algorithms are totally independent of the data source and work under both simulated and physical measurement environments.

Zusammenfassung

Bei der zerstörungsfreien Prüfung mit Ultraschall geht es um die Erkennung und Lokalisierung von Fehlern. Der Messaufbau umfasst einen Positionierer, der die Position und Ausrichtung eines Ultraschallkopfes zu einer zu prüfenden Probe steuert. Wenn der Positionierer seine Anfangsausrichtung zur Probe nicht kennt, ist eine Kalibrierung erforderlich, bei der eine optimale Ausrichtung des Schallkopfs gefunden werden muss. Um Informationen über das Innere der Probe zu erhalten, muss der Schall in die Probe in einem optimalen Winkel eindringen, um die Energie des reflektierte Signal zu maximieren. Um die Entwicklung von neuen Kalibriermethoden zu unterstützen, wird in dieser Arbeit der Aufbau der physikalischen Hardware mit einem Raytracer nachgebildet, um so in Simulationen eine schnellere Entwicklung von Kalibriermethoden zu ermöglichen.

Das vorgestellte Raytracer-Modell ermöglicht die Simulation von Messungen und geometrischen Elementen, die den physikalischen Messaufbau nachbilden. Die Ultraschallwellenfelder werden durch einen diskreten Satz von Strahlen ersetzt, unter der Annahme, dass die Betriebsfrequenz hoch und das Medium homogen ist. Die Amplitude und die Laufzeit jedes empfangenen Strahls über den Sensorbereich werden zur Modulation der Impulsform verwendet, die wiederum einen Träger moduliert, und das Gesamtfeld wird durch Summierung aller skalierten, zeitverschobenen Kopien des Signals berechnet. Auf diese Art lassen sich auch die Form des Sendeimpulses und der Öffnungswinkel programmieren, was eine Vorteil im Vergleich zu echten Messungen darstellt.

Der virtuelle Messaufbau ermöglicht die Analyse von Kalibrierungsalgorithmen und ermöglicht Aussagen über Kombinationen von Parameterwerten, die in einer realen Umgebung funktionieren könnten. In dieser Arbeit werden stochastische Approximationsalgorithmen untersucht, indem verschiedene Testszenarien eingerichtet werden, um die Leistungsmetriken Konvergenzgeschwindigkeit, Genauigkeit und Robustheit zu der Algorithmen zu bewerten. Die Kalibrierungsalgorithmen automatisieren die Kalibrierung und verkürzen die Messzeit im Vergleich zur manuellen Kalibrierung, bei der der Prüfkopf in jeden Winkel bewegt und die Messung abgewartet werden muss. Die Kalibrierungsalgorithmen sind völlig unabhängig von der Datenquelle und funktionieren sowohl in simulierten als auch in realen Messumgebungen.

Contents

List of Figures

1 Introduction

In Ultrasonic Non-destructive Testing (UNDT), a transducer transmits ultrasound pulses over a specimen under test and captures the reflected signal to identify possible defects and locations of those defects[2]. The calibration process involves finding the optimum angle at which the transducer should aim over the specimen to receive the maximum amplitude of the reflected signal for accurate identification and localization of possible defects. Calibration of a transducer is always performed before starting the measurement campaign. The optimum angle is the combination of azimuth and elevation (ϕ, θ) in 3D space.

One of the simplest methods to find the optimum pair of angles is to collect measurements at each possible pair and see which one yields the best result. The approach is simple, but taking measurements at the finest resolution will consume a lot of time and resources. Another problem with such a direct approach is noise. If the measurements are corrupted by noise, then direct computation of gradient is not possible[3]. Hence we need some other technique that would estimate the optimum angle with less number of measurements and also tackle noise.

The goal is to estimate optimum pair of angles with better accuracy and minimum number of measurements which are corrupted by noise. In such scenarios, where only a limited set of noisy measurements are available, the most popular approach is stochastic approximation[4]. Stochastic approximation provides several advantages in solving the given problem. First, it automates the calibration process and reduces human intervention. It requires very few measurements to find the optimum[4][3]. Another advantage is that it can cope with noise. Various techniques exist under stochastic approximation, and most of these techniques have some common fundamental operation, but each has its own advantage. All these techniques try to solve the problem in an iterative manner where in each iteration the gradient of an objective function is approximated and the parameter to be optimized is updated.

In order to carry out the required performance analyses of each calibration method, a measurement setup is required. The measurement setup needs to have a transducer and a few objects. Also, the transducer should be attached to a robotic positioning system that enables us to control the position and angle of the transducer. The hardware dependency involved in measurement operations is a hindrance to faster testing and optimization of calibration algorithm. Also, the flexibility to manipulate certain transducer parameters is often not available in physical measurement setup.

In this thesis, a ray tracer model is developed which could emulate real-world ultrasound setup and reduce hardware dependency allowing analysis of stochastic approximation techniques based on performance metrics such as computational cost per iteration, number of iterations required to converge, and accuracy of the estimated optimum[3]. Another important part of ultrasonic measurement is the choice of transmit pulse. Several types of transmit pulses are implemented and tested. Currently, few of them can be generated only in simulations but not in physical hardware. The calibration framework presented in this thesis combines ray tracer model and calibration algorithms to provide a virtual calibration platform for faster testing and analysis of calibration methods.

1.1 Measurement Setup : Ray tracer

In the measurement setup displayed in Fig. 1, $g(x)$ is the transducer shooting pulse $p(t)$ and $a_2 p(t-\tau)$ is the time shifted pulse echo from the defect surface. In order to emulate such measurement setup, a ray tracer model is developed with an ultrasound transducer and controlling hardware. The motivation behind the development of this model is to remove hardware dependency and reduce measurement time. The ray tracer model has the facility to change the position and orientation of a transducer as in the real world case[5]. Orientation is controlled by two angles, the azimuth and elevation in 3D space, and the position of the transducer can be changed along the x-y plane. In order to calibrate ultrasound transducer specimen objects are required in the measurement setup. Hence, in this model, a specimen object is also included[6, pp. 169–177].

The main advantage of such a model while changing position and orientation, no hardware movement is involved, so the process of measurement collection is much faster and calibration can be performed without any hardware dependency. Usage of such a model allows us to understand the behavior of different calibration algorithms at a lesser time and cost. The model provides an opportunity to test performance of algorithms in a simulated environment which gives an indication of whether to invest time and other resources in testing a particular method with a physical hardware setup. Furthermore, the model provides a way to change physical parameters of transducer such as pulse shape and directivity[7]. In the real world setup to perform calibration with different directivity patterns is not easy as it would

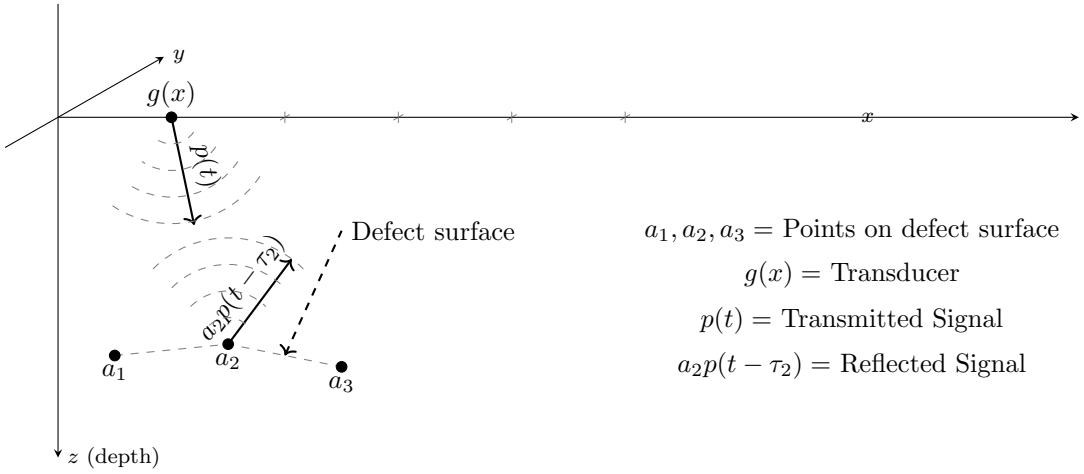


Figure 1: Ultrasound Measurement Setup

require actual transducers with different opening angles, which basically leads to higher investment costs.

The ray tracer model has some obvious limitations. Firstly, only convex surfaces are considered while performing calibration, which poses an important limitation to real-world testing. While developing this model, few simplistic assumptions were made. The first assumption is that reflecting surfaces follow Lambertian reflectance. Lambert's law states that the reflected energy from a reflector is directly proportional to the cosine of the angle between the direction of the transmit signal and the surface normal[8, pp. 13–15]. The mentioned property does not always work with every reflecting surface. The second assumption is that the medium is homogeneous and that the frequency of operation is relatively high. Hence the ultrasound wave propagation is characterized without solving ultrasound field equations which are computationally expensive[9]. The wavefields are treated as a discrete set of rays traveling along straight lines.

1.2 Stochastic Approximation

The stochastic approximation is generally used to maximize or minimize an objective function where the function itself is unknown and only noisy measurements are available[3]. The motivation behind using stochastic approximation methods as they require fewer measurements, which in turn need less time to maximize an objective function and estimate unknown parameters[3]. Also, fewer measurements translate into less transducer hardware movement in real world setup.

Stochastic approximation methods use iterative updates in order to find the optimum value of the parameter. In each iteration, the parameter is updated and steered towards the optimum. The direction of parameter updates is decided by the gradient. The gradient is estimated with finite difference approximation as direct gradient computation is not possible[3]. Gradient basically decides in which direction the parameter would move forward. Another factor involved is step size, which decides the rate of convergence[10]. In the main update equation, step size scales the gradient in order to control the rate of convergence. There is no straightforward way to select step size; hence it must be selected on an empirical basis. The value of step size is selected in a way that avoids slow rate of convergence or divergence[10]. There must also be a stopping criterion for the algorithm to save time and memory resources. The stopping criterion could be based on the difference between the updated parameter value and the parameter value before the update, or it could simply be a fixed number of iterations.

In order to determine the most suitable one for the current applications, different methods of stochastic approximation are implemented and analyzed in terms of computational cost per iteration, convergence behavior and accuracy[3]. The fundamental operation in each method remains the same, which is the computation of the gradient and updating of the parameter vector. One of the differences lies in the selection of step sizes. Few methods use adaptive step size in the update equation where in each iteration step size is reevaluated whereas some methods use fixed step size throughout the execution[4]. Another major difference lies in the computational cost per iteration to compute the gradient.

1.3 Calibration Framework

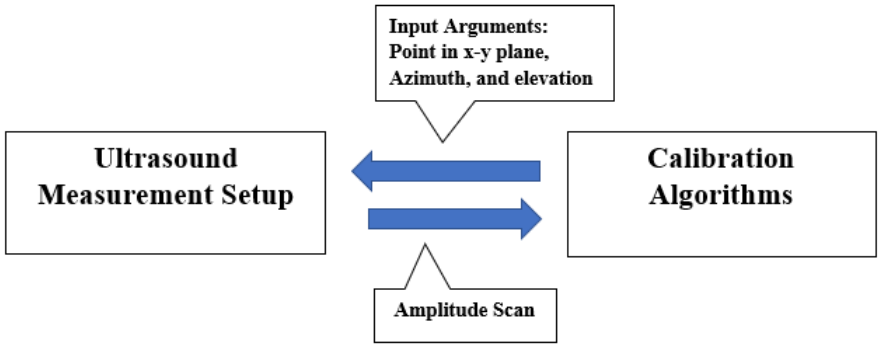


Figure 2: Calibration Framework

In order to conduct calibration operation using a virtual measurement platform, it is important to combine section 1.1 and 1.2 in one framework. Calibration framework displayed in Fig. 2 presents a way to combine and connect different interfaces given in section 1.1 and 1.2. The framework allows rapid testing of calibration algorithms under different parameter settings. Also, it provides an opportunity to adjust parameters and optimize the performance of algorithms prior to real-world testing.

The first section of this framework creates a virtual hardware setup and configures it with the required parameters. The second section uses the configured hardware setup from the first section to generate Amplitude scan (A-scan). It consists of a mathematical model to simulate the operation of a transducer and generate data that is similar to the data generated by a real-world transducer[11]. In transducer operation, the waves are treated as a set of rays that have a certain amplitude and time of flight. The mathematical model uses this information to manipulate the parameters of a pulse shape and generate scaled time-shifted pulses. The choice of the pulse shape is kept programmable and can be selected from the given list. The generated carrier is modulated with scaled, time-shifted copies of pulses and summed up to generate the final signal or an A-scan[11].

The second section deals with the actual calibration part. As mentioned earlier calibration of the transducer is performed by capturing the reflected signal and maximizing the energy of the same. Maximization of the signal energy is performed by a set of stochastic approximation algorithms as mentioned in section 1.2. The virtual measurement platform explained in section 1.1 provides an interface for calibration algorithms to collect measurements by giving the required set of input arguments. Input arguments mainly consist of a point in the x-y plane and a pair of angle (θ, ϕ) in 3D space. An algorithm controls the orientation of a transducer and tries to find an optimum pair of angle (θ, ϕ). At the start of an operation, the parameter to be optimized (θ, ϕ) is initialized with a random guess, and algorithm-specific execution is followed. When the necessary convergence criterion is met, the execution of the algorithm is stopped and optimized parameter (θ, ϕ) is returned. The calibration algorithms are totally independent of the data source and work under both simulated environments as well as with a physical measurement setup.

Part I
Ray Tracer

2 Ray Tracer : Geometry

The ray tracer is a simplified way to simulate ultrasound measurement operation. Ray tracer allows the characterization of ultrasound wave propagation without actually solving ultrasound field equations which are computationally expensive[9][5]. Ultrasound measurement operation mainly involves transducer, specimen objects and controlling hardware. In this section creation of transducer, specimen objects and geometrical setup required to control position and orientation of transducer is explained. In ultrasound measurement operation transducers radiate ultrasound waves towards an object and the reflected wave fields impinging on the sensor area of a transducer generate a total field. It is calculated by integrating all the fields over the entire sensor area (1)[12] of a transducer. In the case of a ray tracer, the continuous field is replaced by a discrete set of rays. Transducer shoots rays into 3D space and intersection of these rays with objects is checked. Rays that intersect with given objects reflect back with certain amplitude and time of flight.

$$s'(t) \propto \iint_{\Omega} f_{\omega}(\underline{r})\, \underline{dr} \tag{1}$$

$$\underline{r} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2}$$

$$\underline{r} \in \Omega \tag{3}$$

The total field calculated in a continuous setting is $s'(t)$ which changes over time, $f_{\omega}(\underline{r})$ is the total field as a function of position vector $\underline{r} \in \mathbb{R}^3$, ω is the frequency of operation and $\Omega \subset \mathbb{R}^3$ is the sensor area of transducer. In discrete setting the integral is replaced by summation and all the amplitudes of received rays are summed up to get the total field.

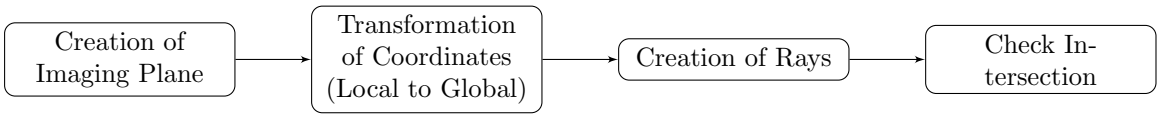


Figure 3: Components of Ray Tracer Model

The main processing blocks involved in ray tracer are given in Fig. 3. Each block implements specific functionality and it is implemented as illustrated in Fig. 3. In the upcoming sections, each block is explained from theoretical and implementation perspective.

2.1 Transducer

A transducer can be designed by a combination of geometric elements. It can be treated as a camera that has a similar behavior as the transducer itself. It has a focus and an imaging plane. An imaging plane can be called an area of the sensor where wave fields are impinging[6, pp. 84–91][12]. Here, instead of dealing directly with wave fields, it is assumed that the medium is homogeneous and frequency is relatively high; hence the wave fields can be treated as rays[9]. The imaging plane is discretized into a grid of pixels and for each pixel, there is a ray associated to it. The amplitude of each ray indicates the intensity of the field at that pixel.

2.1.1 Ray

A ray has a source $\underline{s} \in \mathbb{R}^3$ and a direction $\underline{d} \in \mathbb{R}^3$ where $||\underline{d}|| = 1$ [13][6, pp. 169–177]. A ray can be mathematically represented as follows,

$$\underline{r} = \underline{s} + t' \cdot \underline{d} \tag{4}$$

where t' indicates how far away a ray has travelled in the given direction.

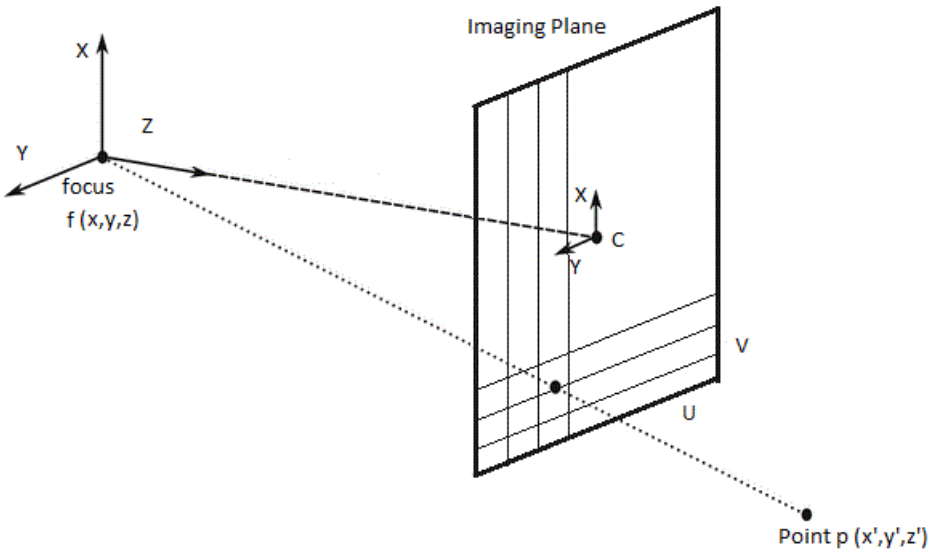


Figure 4: Imaging Plane [1, pp. 610–613]

2.1.2 Imaging Plane

The creation of transducer is based on pinhole camera model which consists of an imaging plane depicting the transducer sensor area[14]. Hence an imaging plane is created of certain height and width illustrated in Fig. 4. As the model works under discrete setting the plane is discretized into a grid of horizontal and vertical pixels. Each pixel has a ray with a certain amplitude associated with it. An orientation of a plane can be controlled by azimuth and co-elevation angles. The focus is kept behind at a distance from the center of a plane. Now each ray is shot from focus through each pixel into space. The direction of each ray is given by the pixel coordinates and focus.

$$\underline{f} = \underline{c} - l \cdot \underline{n} \tag{5}$$

$\underline{f} \in \mathbb{R}^3$ is the focus, l is the focal length, $\underline{c} \in \mathbb{R}^3$ is the center and $\underline{n} \in \mathbb{R}^3$ is the normal vector of an imaging plane. The local pixel coordinates are transformed into global pixel coordinates as the imaging plane lives in 3D space and if the orientation of the plane is changed using azimuth and elevation the global coordinates will not remain the same as the local coordinates[6, pp. 84–91][6, pp. 7–10]. Transformation is performed by using a transformation matrix which consists of orthonormal basis vectors to describe a plane in 3D space[6, pp. 84–91]. The basis vectors to describe a plane in 3D space which is passing through the origin are given as,

$$\underline{x} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \underline{y} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{6}$$

$$\underline{z} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{7}$$

However, if a plane is rotated along azimuth or elevation the coordinates of basis vectors change. In order to accommodate this rotation, basis vectors must be redefined as a function of azimuth and elevation.

$$\underline{n} = \begin{bmatrix} \sin(\theta) \cdot \cos(\phi) \\ \sin(\theta) \cdot \sin(\phi) \\ \cos(\theta) \end{bmatrix} \tag{8}$$

$$\underline{y'} = \begin{bmatrix} -\sin(\phi) \\ \cos(\phi) \\ 0 \end{bmatrix} \tag{9}$$

The basis vector $\underline{x'}$ is found by performing a cross product between $\underline{n}$ and $\underline{y'}$ where $\underline{n}$ is the normal vector.

$$\underline{x'} = \underline{y'} \times \underline{n} \tag{10}$$

The above defined basis vectors as a function of azimuth and elevation can be put into a transformation matrix. This transformation matrix transforms local coordinates into global coordinates[6, pp. 7–10].

$$T = \begin{bmatrix} \underline{x'} & \underline{y'} \end{bmatrix} \tag{11}$$

$$G_{ijk} = T_{ih} \cdot L_{hjk} \tag{12}$$

The equation (12) given above is written in einstein summation format [15]. $G \in \mathbb{R}^{3 \times n_v \times n_h}$ contains the global pixel coordinates where n_v and n_h are the number of vertical and horizontal pixels, $T \in \mathbb{R}^{3 \times 2}$ is the transformation matrix and $L \in \mathbb{R}^{2 \times n_v \times n_h}$ contains the local pixel coordinates.

2.2 Specimen Objects

In an earlier section, components required to depict transducer operation are already been defined. In the real-world ultrasound setup, the transducer shoots ultrasound pulses and these pulses are reflected back from an object under test. In ray tracer pulses are represented by a set of rays shot from a source into 3D space. These rays intersect with objects and are reflected back towards the source[9].

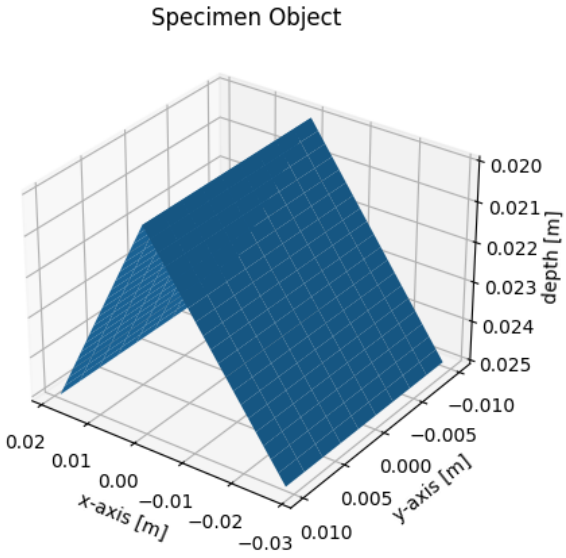


Figure 5: Specimen Object

Specimen objects mainly involve the creation of planes with different orientations. However, while creating objects the plane is not discretized but rather left continuous[6, pp. 169–177]. A plane has a

center, height, width, azimuth, and elevation and is kept at a distance from an imaging plane. If rays shot from a source intersect with an object, they reflect back towards an imaging plane with a certain amplitude. The amplitude of each reflected ray defines the intensity of the reflection. Specimen object created and used in this work is given in Fig. 5 however the geometry and orientation can't be changed in the simulator rather its fixed.

2.2.1 Check Ray Intersection

One of the simplistic assumptions made earlier is that the reflecting surfaces follow Lambertian scattering. Hence the amplitude of a reflected ray is calculated by taking the inner product of the ray direction vector and surface normal. Another parameter of interest is the time of flight of the reflected ray which can be calculated by finding the distance between the intersection point of ray and the source.

If a ray hits the plane then the point at which it intersects must be on the plane as well as on the ray. The equation of the plane is given as follows,

$$\underline{n}^T \cdot (\underline{p'} - \underline{p'_0}) = 0 \tag{13}$$

$\underline{n} \in \mathbb{R}^3$ is the normal, $\underline{p'_0} \in \mathbb{R}^3$ is any point on the plane but in this work it is assumed to be the center of plane. If the point $\underline{p'} \in \mathbb{R}^3$ lies on the ray then it can be written as follows,

$$\underline{p'} = \underline{s} + t' \cdot \underline{d} \tag{14}$$

$\underline{s} \in \mathbb{R}^3$ is the source which is the coordinate of center of the pixel, $\underline{d} \in \mathbb{R}^3$ is the direction and t' defines at what distance does the ray intersect with a plane. Now the above equation (14) of ray can be substituted in the equation of a plane. This substitution allows us to find $t' = \frac{\underline{n}^T \cdot (\underline{p'_0} - \underline{s})}{\underline{n}^T \cdot \underline{d}}$ which can be further used to find the time of flight[13]. If the ray direction is orthogonal to the normal of a plane or the ray is on the plane then $\underline{n}^T \cdot \underline{d} = 0$ which indicates that there is no intersection. Furthermore, if $\underline{n}^T \cdot \underline{d} \neq 0$ but the value of t is negative then in that case plane is behind the ray source hence there is no intersection.

The above procedure finds an intersection point on a plane assuming that the plane has infinite height and width. But in our case the plane is finite so it is necessary to check if the intersection point is inside the given plane dimensions. It can be tested by transforming global coordinates of point $\underline{p'}$ to corresponding local coordinates[6, pp. 84–91][6, pp. 7–10]. Once the local coordinates are obtained then it can be checked easily if the point is inside the plane. Global coordinates of point $\underline{p'} \in \mathbb{R}^3$ are obtained from the equation (14) and the transformation matrix is already defined in (11). The columns of transformation matrix T are orthonormal hence $T^T \cdot T = I$. So $L \in \mathbb{R}^{2 \times n_v \times n_h}$ can be calculated as below,

$$G_{ijk} = T_{ij} \cdot L_{hjk} \tag{15}$$

$$T_{iw} \cdot G_{ijk} = T_{iw} \cdot T_{ij} \cdot L_{hjk} \tag{16}$$

$$L_{wjk} = T_{iw} \cdot G_{ijk} \tag{17}$$

The local coordinates are available in $L \in \mathbb{R}^{2 \times n_v \times n_h}$. Now each local coordinate x'_l and y'_l is checked against the given width and height of the plane. If the plane has a certain height h and width w then for intersection following condition must be satisfied,

$$\underline{p'} = \begin{bmatrix} a' \\ b' \\ c' \end{bmatrix} \tag{18}$$

$$-w/2 + a' \leqslant x'_l \leqslant w/2 + a' \tag{19}$$

$$-h/2 + b' \leqslant y'_l \leqslant h/2 + b' \tag{20}$$

If both conditions are satisfied then the intersection point $\underline{p'}$ is inside the plane dimensions.

2.2.2 Implementation of Ray Intersection

First condition that needs to be tested is whether ray direction is orthogonal to the normal of object plane. Let $\underline{d}$ and $\underline{n}$ be the ray direction and normal vector respectively. Parameters $t' = \infty$, $intersection = 0$ and $\underline{p'} = \begin{bmatrix}\infty\\ \infty\\ \infty\end{bmatrix}$ are initialized.

if $|\underline{n}^T \cdot \underline{d}| > 0$ **then**
  Check further conditions
end if

return $intersection, \underline{p'}, t', \underline{n}$

In the next step at what distance does the ray intersect with object plane is calculated.

$$t' = \frac{\underline{n}^T \cdot (\underline{p'_0} - \underline{s})}{\underline{n}^T \cdot \underline{d}} \tag{21}$$

If $t' < 0$ then the object is behind the ray source hence intersection can not occur.

if $|\underline{n}^T \cdot \underline{d}| > 0$ **then**
  $t' = \frac{\underline{n}^T \cdot (\underline{s} + \underline{d})}{\underline{n}^T \cdot \underline{d}}$
  if $t' < 0$ *or* $\underline{n}^T \cdot \underline{d} \geq 0$ **then**
    $t' = \infty$
  else
    Check further conditions
  end if
end if

return $intersection, \underline{p'}, t', \underline{n}$

Next condition to be checked is whether intersection occurs within the plane dimensions.

$$\underline{p'} = \underline{s} + t' \cdot \underline{d} \tag{22}$$

$$\underline{m} = T^T \cdot (\underline{p'} - \underline{p'_0}) \tag{23}$$

Here $\underline{m} \in \mathbb{R}^2$ contains local coordinates, $\underline{p'} \in \mathbb{R}^3$ is the intersection point,$\underline{p'_0} \in \mathbb{R}^3$ is the center of plane and $T \in \mathbb{R}^{3\times 2}$ is the transformation matrix.

if $|\underline{n}^T \cdot \underline{d}| > 0$ **then**
  $t' = \frac{\underline{n}^T \cdot (\underline{p'_0} - \underline{s})}{\underline{n}^T \cdot \underline{d}}$
  if $t' < 0$ *or* $\underline{n}^T \cdot \underline{d} \geq 0$ **then**
    $t' = \infty$
  else
    $\underline{p'} = \underline{s} + t' \cdot \underline{d}$
    $\underline{m} = T^T \cdot (\underline{p'} - \underline{p'_0})$
    if $|m[0]| > w/2$ *or* $|m[1]| > h/2$ **then**
      $t' = \infty$
      $\underline{p'} = \begin{bmatrix}\infty\\ \infty\\ \infty\end{bmatrix}$
    else
      $intersection =$ **true**
    end if
  end if
end if

return $intersection, \underline{p'}, t', \underline{n}$

After the conditions related to the intersection are checked the required parameters for further calculation are returned. The distance at which the intersection occurs is given by t'. Hence the time of flight can be calculated by dividing the parameter t' by the velocity of the wave in the medium.

$$\tau = \frac{2t'}{c_0} \tag{24}$$

The amplitude can be calculated by taking dot product of ray direction and normal of the object plane.

$\beta = -\underline{n}^T \cdot \underline{d}$

3 Mathematical Model and Implementation

In this section, a mathematical model to obtain ultrasound measurements using a ray tracer is explained. A mathematical model presents a way to calculate the total field over a sensor area of the transducer[12]. The total field is defined as the total contribution of all rays received over an imaging plane depicting the sensor area of transducer[12]. The total field generated over a transducer in continuous setting is calculated by an equation given in (1). In discrete settings, each received ray carries a certain amplitude and time of flight. The amplitude states the intensity of the field at that pixel. Carrier signal generated at higher frequency is modulated with a pulse shape. The parameters of pulse shape are changed according to the obtained amplitude and time of flight of the received rays. The resultant signal is the sum of time-shifted copies of a carrier-modulated pulses[11].

3.1 Mathematical Model

An imaging plane is discretized into a grid of horizontal and vertical pixels. Each pixel is associated with a ray; hence the total number of rays shot is equivalent to the number of horizontal and vertical pixels. The total field is calculated based on the number of rays received and their associated amplitudes. If a ray does not intersect with an object, then the time of flight of that ray is set to ∞. In such a scenario contribution of that ray to the total field is considered zero because it does not intersect with any object. The rays which intersect with any of the objects are modulated with a pulse shape and considered in the calculation of the total field. A transducer also has an opening angle which defines the directivity pattern of a transducer. Directivity is the measure of how focused is the radiated energy by a transducer in a particular direction[7]. It scales the amplitude of a reflected ray based on its angle of incidence on the sensor area. The other amplitude associated with each ray is based on the angle of departure, which is calculated by taking the inner product between the ray direction and normal $\left|\underline{n}^T \cdot \underline{d}\right|$.

While calculating total field, the number of horizontal pixels n_h, number of vertical pixels n_v, total number of rays shot $n_v \cdot n_h$ and a pulse shape $p(t)$ is chosen. Each received ray has an amplitude β based on the angle of departure; amplitude a based on the angle of incidence and time of flight τ. If the number of rays shot is equal to the number of rays received, then the total contribution of all rays after pulse shape modulation is given as the sum of scaled shifted copies of pulse shapes[11][12].

$$p(t) = \exp\left(-\alpha \cdot t^2\right) \tag{25}$$

$$s(t) = \sum_{n=0}^{n_v \cdot n_h} a_n \cdot \beta_n \cdot p(t - \tau_n) \tag{26}$$

$p(t)$ is often chosen to be a Gaussian envelope used for modulation where α is the bandwidth factor. α defines the bandwidth occupied by a signal in the time domain which is the same as the reciporocal of standard deviation σ. $s(t)$ is the generated base-band signal and $p(t - \tau_n)$ is the time-shifted copy of pulse shape. In the next step carrier signal is generated and modulated with a pulse shape.

$$c(t) = \cos\left(2\pi f_c \cdot (t - \tau) + \phi\right) \tag{27}$$

$c(t)$ is the carrier signal where f_c is the carrier frequency, τ is the time of flight, ϕ is the phase factor and t is the time axis. The carrier signal $c(t)$ is modulated with the base-band signal $s(t)$.

$$x(t) = \sum_{n=0}^{n_v \cdot n_h} a_n \cdot \beta_n \cdot p(t - \tau_n) \cdot \cos\left(2\pi f_c \cdot (t - \tau_n) + \phi\right) \tag{28}$$

$x(t)$ is the sum of scaled shifted copies of a carrier-modulated pulse. It is the total field generated by a transducer. As a post-processing step signal $x(t)$ is normalized by a total number of rays so that the maximum amplitude of the signal remains equal to 1.

3.2 Pulse Selection

In ultrasonic non-destructive testing applications changing pulse shape and its parameters is not possible easily. However in simulations it is possible to study the impact of pulse shape on the performance of calibration. In this work several different types of pulse shapes are implemented and tested. The pulse

shape is selected based on the role it plays in calibration performance and its implementation feasibility in physical hardware. An autocorrelation function and power spectral density of a pulse shape affect the performance of calibration algorithms. Both autocorrelation and power spectral density of each pulse shape were studied.

3.2.1 Sinc Pulse

A sinc filter is also called an ideal low pass filter as it blocks all the frequencies below its cutoff frequency. The impulse response of the Sinc filter is a Sinc function and the frequency response is a rectangular function. If we observe in frequency domain it blocks all the frequencies except in the given pass-band. In practical applications, the impulse response is truncated and windowed to a finite length[16, pp. 80–82]. Sinc filter in the frequency domain is written as,

$$P(f') = A' \cdot \text{rect}\left(\frac{f'}{2 \cdot B}\right) \tag{29}$$

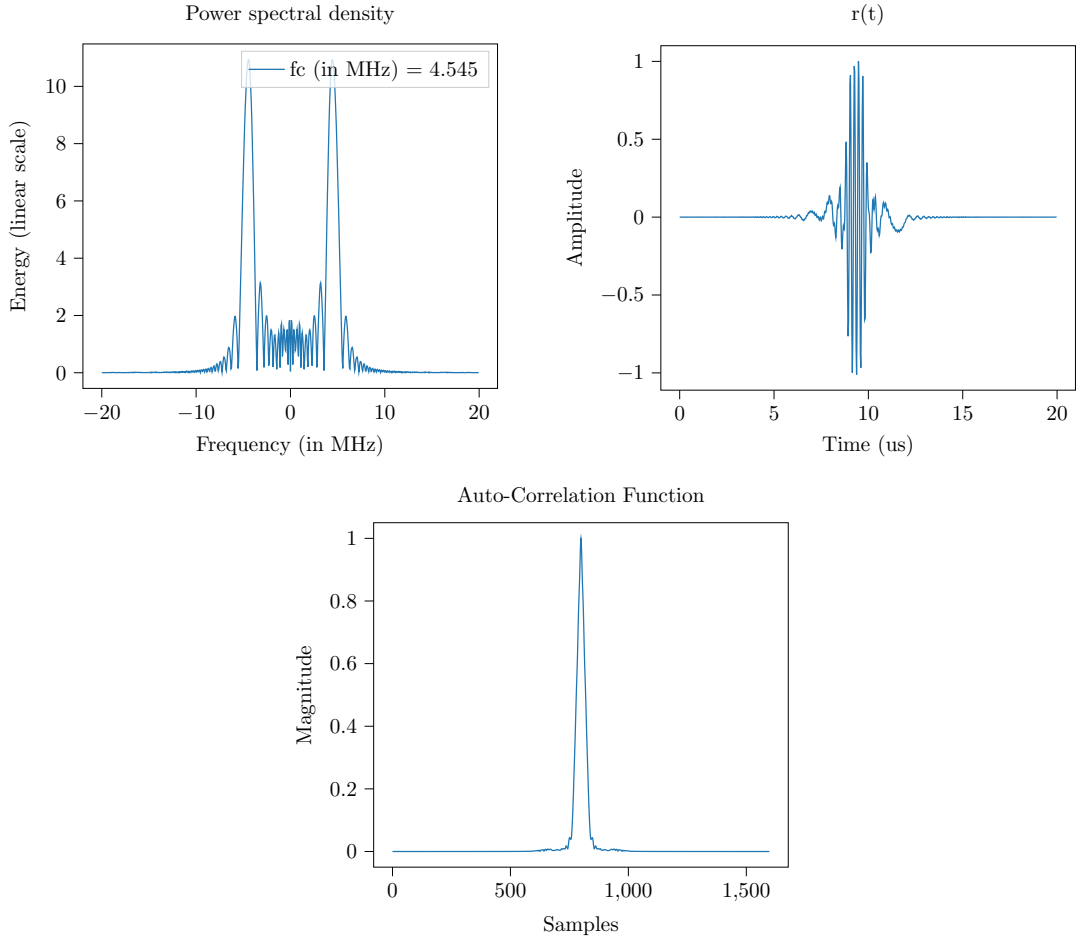
A' is amplitude, f' is frequency and B is Bandwidth. The inverse Fourier transform of above frequency response will be the impulse response in time domain.

$$p(t) = A' \cdot 2B \cdot \text{sinc}\left(\pi \cdot 2B \cdot t\right) \tag{30}$$

$p(t)$ is the impulse response of sinc filter in the time domain. In the next step, the impulse response is truncated to a finite length. The truncated impulse response is used as a pulse shape and multiplied with a carrier frequency which can be transmitted into a medium. Auto-correlation and power spectral density of the carrier modulated signal are calculated to analyze its effect on calibration performance.

$$r(t) = p(t) \cdot \cos\left(2\pi f_c t\right) \tag{31}$$

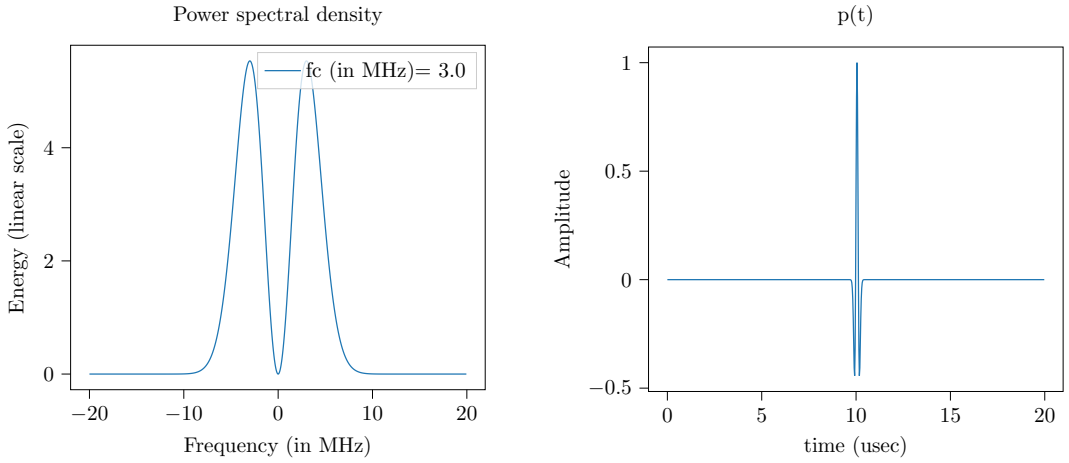
$r(t)$ is the carrier modulated signal where f_c is the center frequency.



3.2.2 Ricker Wavelet

Ricker wavelet is mostly used in seismic data analysis to simulate seismic wave propagation. It is basically the second derivative of the Gaussian function. Ricker wavelet is expressed only as a function of center frequency in time domain [17].

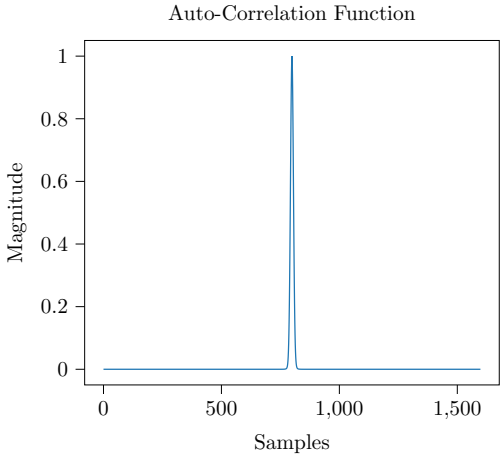
$$p(t) = (1 - \frac{1}{2} \cdot (2\pi f_c)^2 t^2) \cdot \exp\left(-\frac{1}{4} \cdot (2\pi f_c)^2 t^2\right) \tag{32}$$



Fourier transform of above expression $p(t)$ would give frequency response $P(f')$. In the expression of $p(t)$ any change in parameter t does not affect power spectral density $S_{xx}(f')$. Power spectral density $S_{xx}(f')$ and auto-correlation function $r_{xx}(\tau')$ are also calculated.

$$P(f') = \frac{(2\pi f_c)^2}{\sqrt{\pi}(2\pi f_c)^3} \exp\left\{\frac{-(2\pi f_c)^2}{(2\pi f_c)^3}\right\} \tag{33}$$

$$S_{xx}(f') = |P(f')| \tag{34}$$



3.2.3 Gabor function

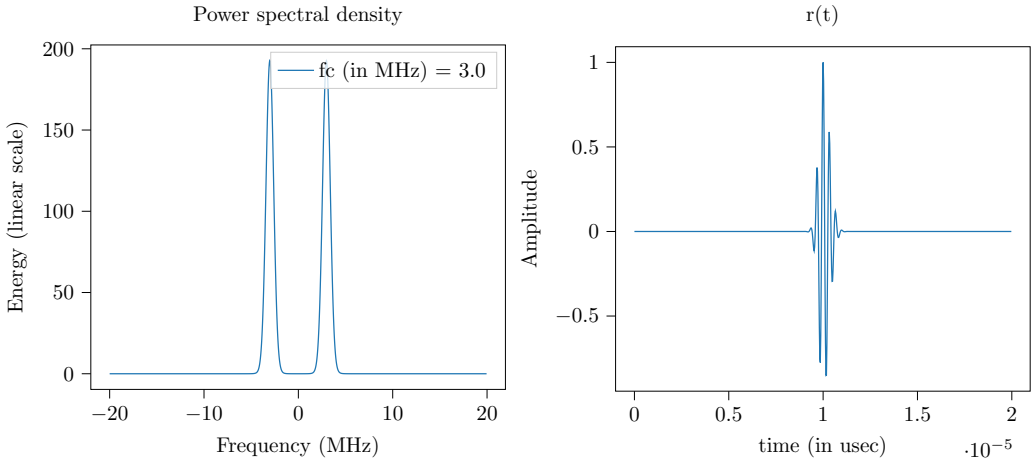
Gabor function is expressed mathematically as a modulated Gaussian function. It is used to represent a received signal $r(t)$ which is assumed to be a sum of scaled, time-shifted copies of a transmitted signal. Gabor function is same as Gaussian function but gabor function is often represented in complex form[18].

$$p(t) = \exp\left(-\alpha \cdot t^2\right) \tag{35}$$

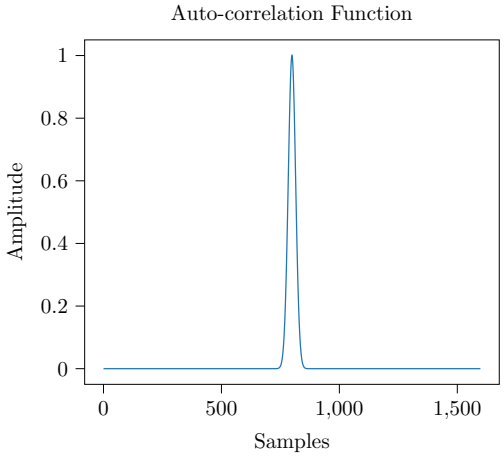
$$c(t) = \cos\left(2\pi f_c t + \phi\right) \tag{36}$$

$$p'(t) = \exp\left(-\alpha \cdot t^2\right) \cdot \cos\left(2\pi f_c t + \phi\right) \tag{37}$$

Here $p(t)$ is the pulse shape, $c(t)$ is the carrier and $p'(t)$ is the modulated gaussian function. In the above equations (35) and (36), α is the bandwidth factor which defines the time duration of pulse echo in time domain, f_c is the center frequency, ϕ is the phase factor and t is the sampling interval.



Power spectral density $S_{xx}(f)$ and auto-correlation function $r_{xx}(\tau)$ of $p(t)$ are calculated.



4 Summary and Interpretation of Part I

In a ray tracer, a virtual ultrasound measurement setup is created in order to emulate ultrasound measurement operation. The virtual measurement setup consists of a transducer, a specimen, and controlling hardware. Basic geometric elements are used to create a transducer and specimen objects. In the ray tracer model, it is assumed that the medium is homogeneous and that the frequency of operation is relatively high. Hence, ultrasound wave fields radiated by a transducer are treated as a discrete set of rays traveling in straight lines. It allows characterization of ultrasound wave propagation without actually solving ultrasound field equations which are computationally expensive. The operation of a transducer is emulated by treating it as a camera that has a source and an imaging plane. An imaging plane is assumed to be the sensor area of the transducer. Specimen objects are also created to test, ultrasound calibration.

Rays shot by a transducer are checked for intersection with a specimen object. Amplitude and time of flight of rays that intersect with objects are calculated. A mathematical model is used to calculate the total field generated by these reflected rays over the sensor area of a transducer. According to the model, each ray is modulated with a pulse shape with its associated amplitude and time of flight. Each of these scaled time-shifted copies is modulated by a carrier frequency and summed up in order to obtain the total field. This total field generated is considered the A-scan generated by a transducer.

The Ray tracer framework provides an interface to simulate ultrasound measurement operation. It allows us to simulate measurements without any physical hardware setup. It completely removes hardware dependency involved in calibration testing; hence the time required to simulate any measurement scenario is very minimal. The main advantage is that certain parameters are not easily programmable in a real-world hardware setup, such as opening angle or Signal to Noise Ratio(SNR). These parameters are made programmable in this framework, allowing quick simulation of different measurement scenarios. In the actual hardware setup, the transmit pulse shape is not programmable along with its parameters, but here in simulations, it can be made programmable.

The limitations of the ray tracer are the simplistic assumptions being made to reduce computational complexity and simpler characterization of wave propagation. In this model, reflecting surfaces are assumed to follow Lambertian scattering and specimen objects are only considered to be convex. If the medium is not homogeneous and the frequency of operation is not high enough, then the waves cannot be treated as rays. In spite of the mentioned limitations, the ray tracer provides a simple model to simulate ultrasound measurement operation without any hardware dependency. The framework provides a complete virtual hardware setup consisting of a transducer, specimen, and positioning system. The position and orientation of a transducer can be controlled by passing necessary input arguments to an interface, and measurements can be obtained at the desired position and orientation.

Part II
Stochastic Approximation

5 Stochastic Approximation : Introduction

Stochastic Approximation methods play a major role in the automation of calibration task using fewer measurements. The main goal of the calibration task is to estimate the calibration angle at which the reflected energy received by a transducer from the specimen under test is maximized. The given maximization task can be solved by forming a regular grid for the calibration angle (θ, ϕ) and taking measurements at each pair of values, and deciding which one gives the better result. The major problem with the mentioned approach is the time required to take measurements and accuracy due to constraints over grid resolution. Stochastic approximation techniques overcome this problem as they require fewer measurements and also they are accurate in estimating parameters. They start with an initial guess of the parameter and try to approximate the gradient as the function that needs to be maximized is not directly available. The approximated gradient is used to update the parameter and steer it towards the optimum. All stochastic approximation-based techniques are iterative hence in each iteration gradient is approximated and using that gradient parameter is updated.

In the real world setting a transducer is attached to positioning hardware and moved over the specimen to collect measurements at different locations. The collected measurements are used by calibration algorithms to estimate unknown parameters. In order to remove hardware dependency, a ray tracer model is implemented which can simulate ultrasound measurement operation and collect measurements at the desired location. In this work calibration algorithms use a ray tracer model to collect measurement data however the algorithms are totally independent of the data generation platform. Calibration algorithms provide an independent framework that works with simulated data as well as with hardware measurement data. In this part, each algorithm is explained with a general introduction and mathematical details. Also, the performance of the algorithm is evaluated by assuming a certain test scenario and observing the cost function plots.

6 Stochastic Gradient Ascent

Stochastic gradient ascent is one of the fundamental algorithm used to solve optimization problems. The stochastic gradient ascent has a simple implementation, but the computational cost increases with a number of parameters[3]. An algorithm follows an iterative procedure where in each iteration it calculates the gradient of an objective function. The gradient provides the direction in which maximum change is possible. At the end of each iteration, the calculated gradient is used to update the parameters[19, pp. 246–254]. It can be seen as the parameters are steered towards their optimum value using the estimated direction. The procedure is repeated until an optimum parameter is found or the criterion for a maximum number of iterations is reached.

6.1 Algorithm Overview

In our problem there is an objective function $f(\underline{\theta})$ that needs to be optimized but it is not directly available; instead a set of noisy measurements are available. In general case an algorithm works for parameters in $\mathbb{R}^n$ but in our application the parameter that needs to be optimized is in $\mathbb{R}^2$ since the goal is to solve calibration problem. In the next step, measurements are obtained over a parameter vector $\underline{\theta}$ in the presence of noise where $\underline{\theta} \in \mathbb{R}^2$ consists of azimuth and elevation angles. In such cases it is not possible to calculate gradient directly, so it is estimated using finite difference approximation[3]. The approximated gradient at $\mathbf{k}^{th}$ iteration can be given as,

$$\hat{g}_k(\theta(1)_k) = \frac{f(\theta_k^{(1)} + \delta, \theta_k^{(2)}) - f(\theta_k^{(1)} - \delta, \theta_k^{(2)})}{2 \cdot \delta} \tag{38}$$

$$\hat{g}_k(\theta(2)_k) = \frac{f(\theta_k^{(1)}, \theta_k^{(2)} + \delta) - f(\theta_k^{(1)}, \theta_k^{(2)} - \delta)}{2 \cdot \delta} \tag{39}$$

In the equation (38) δ is a small step in positive or negative direction, $f(\theta_k^{(1)} \pm \delta, \theta_k^{(2)})$ is a function evaluation and $\theta_k^{(1)}$ is the first element of parameter vector at which the function is evaluated[3].

$$\theta_{k+1}^{(1)} = \theta_k^{(1)} + \mu \cdot \hat{g}_k(\theta_k^{(1)}) \tag{40}$$

$$\theta_{k+1}^{(2)} = \theta_k^{(2)} + \mu \cdot \hat{g}_k(\theta_k^{(2)}) \tag{41}$$

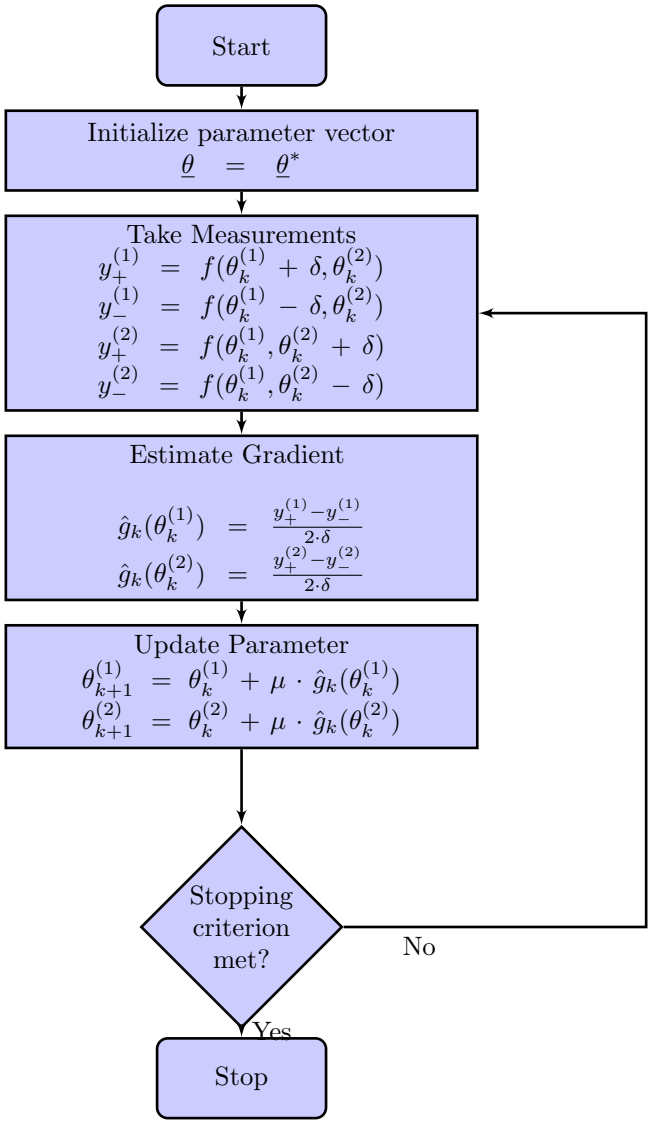
In the above equations (40) and (41) μ is the step size , $\hat{g}_k(\theta_k^{(1)})$ and $\hat{g}_k(\theta_k^{(2)})$ is the partial derivative approximated with respect to first and second element of parameter vector. Also, $\theta_{k+1}^{(1)}$ and $\theta_{k+1}^{(2)}$ are the updated elements of parameter vector. The value of μ must be decided empirically as there is no specific criterion to decide that value[19, pp. 246–254]. The major drawback is it becomes computationally expensive as the number of parameters increases[4]. In our application, there are two parameters to be optimized so it takes 4 function evaluations in each iteration.

$$y_+^{(1)} = f(\theta_k^{(1)} + \delta, \theta_k^{(2)}) \tag{42}$$
$$y_-^{(1)} = f(\theta_k^{(1)} - \delta, \theta_k^{(2)}) \tag{43}$$
$$y_+^{(2)} = f(\theta_k^{(1)}, \theta_k^{(2)} + \delta) \tag{44}$$
$$y_-^{(2)} = f(\theta_k^{(1)}, \theta_k^{(2)} - \delta) \tag{45}$$

Using above function evaluations gradient with respect to each parameter is calculated. Hence if number of parameters are $\mathbf{N}'$ then $2\mathbf{N}'$ function evaluations are required in each iteration.

6.2 Flowchart

In the below flowchart, $\theta^{(1)}$ and $\theta^{(2)}$ are the elements of parameter vector $\underline{\theta} \in \mathbb{R}^2$. Also $\hat{g}(\theta^{(1)})$ and $\hat{g}(\theta^{(2)})$ are the elements of gradient vector $\underline{\hat{g}}(\underline{\theta})$.



7 Variants of Stochastic Gradient Ascent

In an earlier section, stochastic gradient ascent in its original form has been explained from its theoretical and implementation perspective. The stochastic gradient ascent has certain variants which are derived from the original algorithm, but they do have some modifications in their implementation. It is important to study the intuition behind these modifications and their impact on the performance of the algorithm.

7.1 Stochastic Gradient Ascent with Momentum

7.1.1 Algorithm Overview

In the original gradient ascent after gradient approximation, the parameters are updated using only the current gradient information. In the updated version of the algorithm, once the gradient is approximated, the parameters are updated using the current gradient information and also a fraction of the previous parameter update[20]. The purpose behind this additional step is to accelerate gradient ascent convergence.

The fraction μ of the previous parameter update is called the momentum. Intuitively, the momentum term increases the speed of the algorithm if successive gradients are pointing in the same direction and reduce it if it's pointing in the opposite direction[20]. Because of the momentum term, the oscillations produced by gradient ascent in the region of convergence are reduced and the rate of convergence is also increased. Here η is used to represent momentum factor. Typically the value of η is kept in the range of 0.8 to 0.9. If the value of η is set to 0 then basically no previous update is added to the current update term and the algorithm is the same as the original one. The update equation for the modified algorithm is as follows [9],

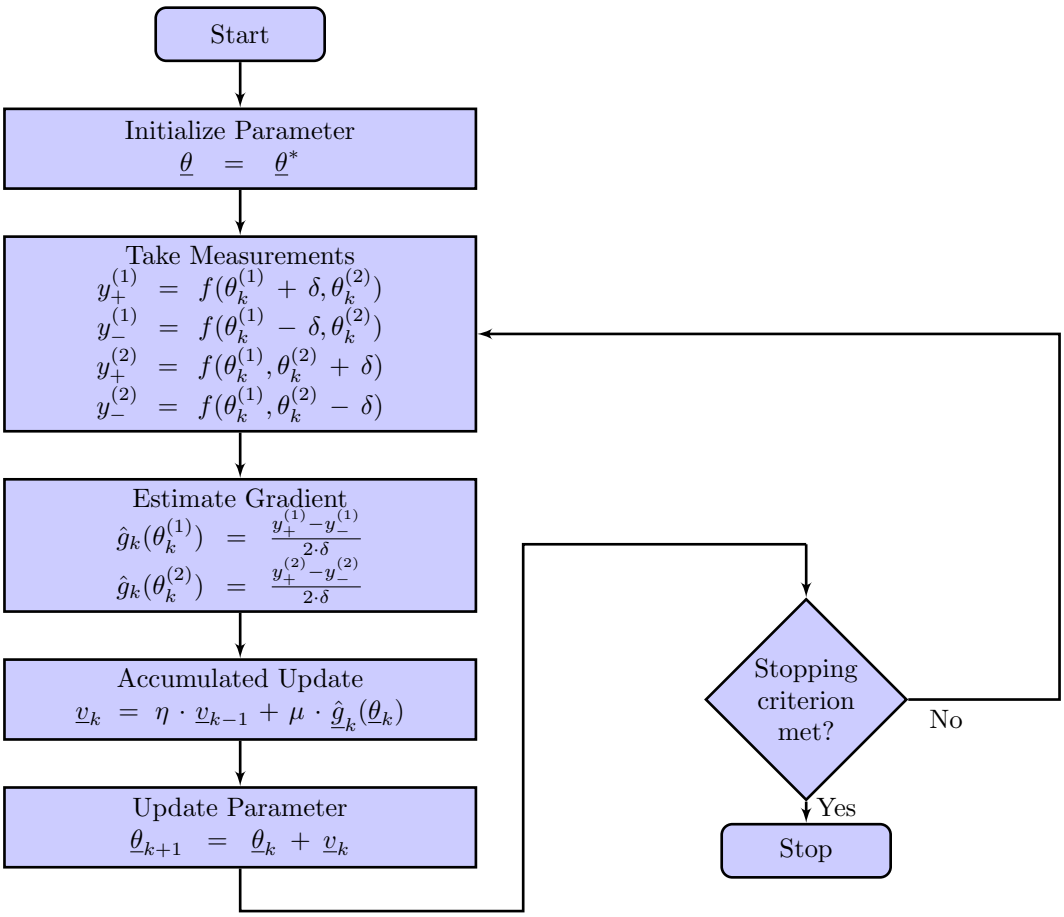
$$\underline{v}_k = \eta \cdot \underline{v}_{k-1} + \mu \cdot \hat{\underline{g}}(\underline{\theta}_k) \tag{46}$$

$$\underline{\theta}_{k+1} = \underline{\theta}_k + \underline{v}_k \tag{47}$$

In the above update equations (46) and (47) $\underline{\theta} \in \mathbb{R}^2$ is the parameter vector being updated, η is the momentum term, μ is the step size, $\hat{\underline{g}}(\underline{\theta}_k)$ is the approximated gradient vector and $\underline{v}_{k-1}$ is the previous update and $\underline{v}_k$ is the accumulated current update.

7.1.2 Flowchart

In the below flowchart, $\theta^{(1)}$ and $\theta^{(2)}$ are the elements of parameter vector $\underline{\theta} \in \mathbb{R}^2$. Also $\hat{g}(\theta^{(1)})$ and $\hat{g}(\theta^{(2)})$ are the elements of gradient vector $\hat{\underline{g}}(\underline{\theta})$.



7.2 Nesterov Accelerated Gradient

7.2.1 Algorithm Overview

The purpose of momentum term in the parameter update equation given in (46) is basically to accelerate convergence of algorithm but there are few disadvantages which come along. Momentum adds a fraction of previous update along with the current gradient without anticipating the future position of parameters. So there lies the problem of overshooting the maximum where the algorithm is very close to maximum. To resolve this problem a modification is required in the update equation which is given by Nesterov's Accelerated Gradient[21][22].

In Accelerated Gradient at approximate future position of parameters the gradient is estimated. This term is the so called correction factor. Now in the update equation first term is the previous update and second term is the correction factor. So accelerated gradient takes a step in the direction of previous update vector and then calculates gradient at that position[22]. This calculation allows us to to control the speed of algorithm and make sure that the overshooting of maximum is avoided.

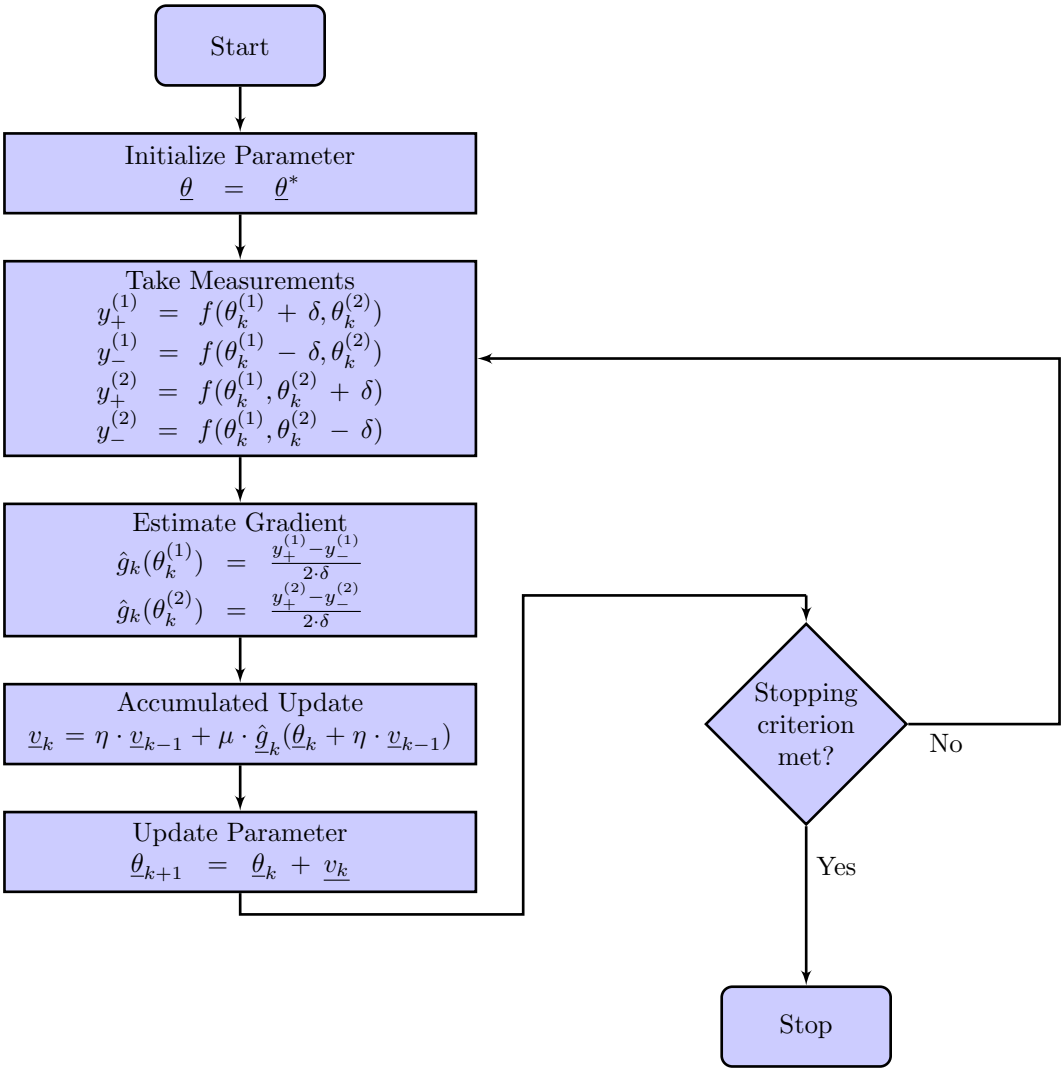
$$\underline{v}_k = \eta \cdot \underline{v}_{k-1} + \mu \cdot \underline{\hat{g}}(\underline{\theta}_k + \eta \cdot \underline{v}_{k-1}) \tag{48}$$

$$\underline{\theta}_{k+1} = \underline{\theta}_k + \underline{v}_k \tag{49}$$

In the above update equations (48) and (49)$\underline{\theta} \in \mathbb{R}^2$ is the parameter vector, $\underline{v}_k$ is the update vector, $\underline{g}(\underline{\theta}_k + \mu \cdot \underline{v}_{k-1})$ is the gradient estimated at the approximate future position of parameters[9].

7.2.2 Flowchart

In the below flowchart, $\theta^{(1)}$ and $\theta^{(2)}$ are the elements of parameter vector $\underline{\theta} \in \mathbb{R}^2$. Also $\hat{g}(\theta^{(1)})$ and $\hat{g}(\theta^{(2)})$ are the elements of gradient vector $\underline{\hat{g}}(\underline{\theta})$.


Start
Initialize Parameter
$\underline{\theta} = \underline{\theta}^*$
Take Measurements
$y_+^{(1)} = f(\theta_k^{(1)} + \delta, \theta_k^{(2)})$
$y_-^{(1)} = f(\theta_k^{(1)} - \delta, \theta_k^{(2)})$
$y_+^{(2)} = f(\theta_k^{(1)}, \theta_k^{(2)} + \delta)$
$y_-^{(2)} = f(\theta_k^{(1)}, \theta_k^{(2)} - \delta)$
Estimate Gradient
$\hat{g}_k(\theta_k^{(1)}) = \frac{y_+^{(1)} - y_-^{(1)}}{2 \cdot \delta}$
$\hat{g}_k(\theta_k^{(2)}) = \frac{y_+^{(2)} - y_-^{(2)}}{2 \cdot \delta}$
Accumulated Update
$\underline{v}_k = \eta \cdot \underline{v}_{k-1} + \mu \cdot \underline{\hat{g}}_k(\underline{\theta}_k + \eta \cdot \underline{v}_{k-1})$
Update Parameter
$\underline{\theta}_{k+1} = \underline{\theta}_k + \underline{v}_k$
Stopping criterion met?
No
Yes
Stop

8 SPSA

An important property of Simultaneous Perturbation Stochastic Approximation (SPSA) is it requires only two function evaluations in each iteration irrespective of the dimension of parameter vector[4]. If number of parameters $\mathbf{N}'$ then for finite difference approximation based method such as gradient ascent explained in section 6, $2\mathbf{N}'$ function evaluations are required per iteration. However SPSA requires only 2 function evaluations irrespective of the number of parameters being optimized[3].

8.1 Algorithm Overview

SPSA while evaluating the function generates a simultaneous perturbation vector of dimension equal to the dimension of the parameter vector. It is chosen from an asymmetric Bernoulli distribution with probability of 1/2 for each ± 1 outcome[4].

$$y_+ = f(\theta_k^{(1)} + \Delta_k^{(1)} \cdot c'_k, \theta_k^{(2)} + \Delta_k^{(2)} \cdot c'_k) \tag{50}$$

$$y_- = f(\theta_k^{(1)} - \Delta_k^{(1)} \cdot c'_k, \theta_k^{(2)} - \Delta_k^{(2)} \cdot c'_k) \tag{51}$$

$$c'_k = \frac{c'}{(k+1)^\gamma} \tag{52}$$

In the above expressions $\underline{\Delta}_k \in \mathbb{R}^2$ is the perturbation vector where as $\Delta_k^{(1)}$ and $\Delta_k^{(2)}$ are the elements of perturbation vector, c'_k is the gain sequence and $\underline{\theta} \in \mathbb{R}^2$ is the parameter vector where as $\theta^{(1)}$ and $\theta^{(2)}$ are the elements of parameter vector. Using above function evaluations (50) gradient can be approximated as[4],

$$\hat{g}_k^{(1)}(\underline{\theta}_k) = \frac{y_+ - y_-}{2 \cdot c'_k \cdot \Delta_k^{(1)}} \tag{53}$$

$$\hat{g}_k^{(2)}(\underline{\theta}_k) = \frac{y_+ - y_-}{2 \cdot c'_k \cdot \Delta_k^{(2)}} \tag{54}$$

In the above equations (53) and (54) $\hat{g}_k^{(1)}(\underline{\theta}_k)$ is the first element and $\hat{g}_k^{(2)}(\underline{\theta}_k)$ is the second element of gradient vector $\underline{\hat{g}}(\underline{\theta})$. After gradient computation the final step is parameter update equation which is written as,

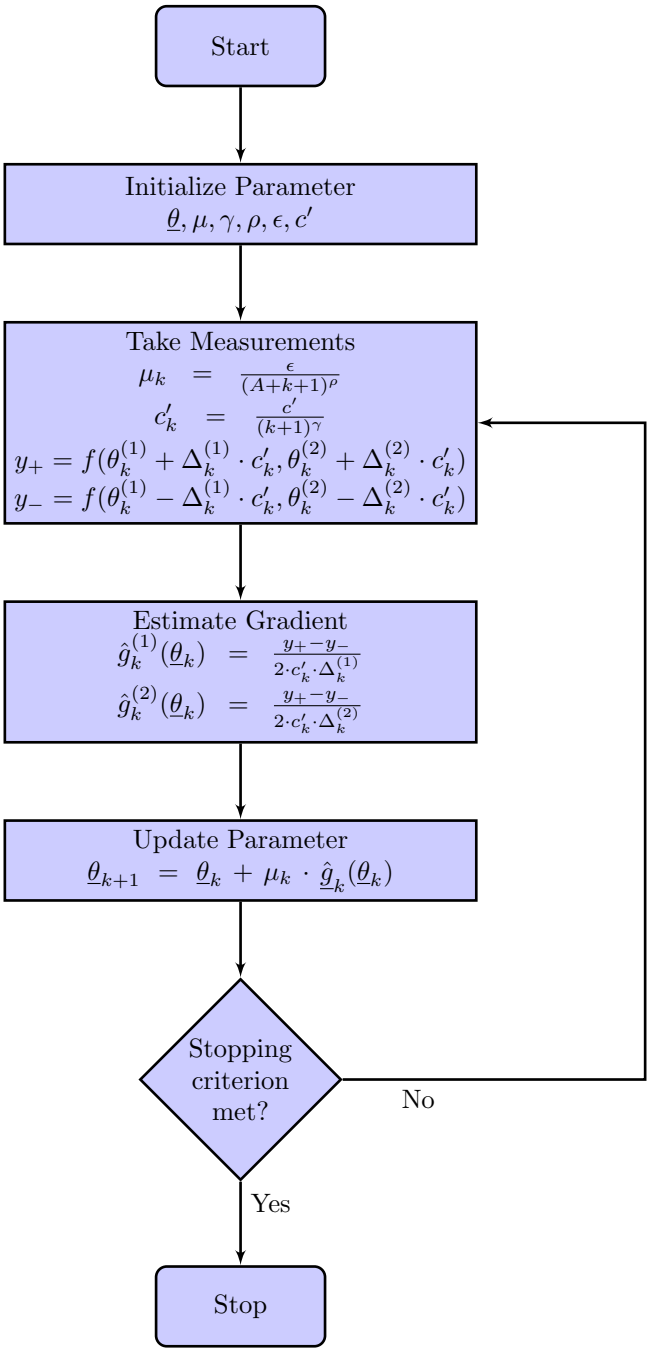
$$\mu_k = \frac{\epsilon}{(A+k+1)^\rho} \tag{55}$$

$$\theta_{k+1}^{(1)} = \theta_k^{(1)} + \mu_k \cdot \hat{g}_k^{(1)}(\underline{\theta_k}) \tag{56}$$

$$\theta_{k+1}^{(2)} = \theta_k^{(2)} + \mu_k \cdot \hat{g}_k^{(2)}(\underline{\theta_k}) \tag{57}$$

In the equation (55) μ_k is an adaptive step size and θ_{k+1} in equation (56) is the updated parameter[4]. There are certain guidelines to choose the values of ϵ, A, c', ρ and γ in literature. When the measurements are highly corrupted by noise, then the value of ϵ should be smaller and c' should be larger[4]. It is also recommended in the literature to set the value of c' close to the standard deviation of function measurements. Typical values of ρ and γ are chosen as 0.602 and 0.101, respectively[4]. The value of A is chosen as a small percentage of a maximum number of iterations.

8.2 Flowchart



9 Newton's Method

Newton's method is one of the second-order optimization methods. It tries to maximize an objective function by computing the second-order gradient. The convergence of Newton's method is much faster than the gradient ascent. The major difference between gradient ascent and Newton's method is that it incorporates the curvature of the objective function rather than just the steepness of function[23]. Also, the number of function evaluations per iteration performed by Newton's method for current application are 9 which is more than gradient ascent.

9.1 Algorithm Overview

The curvature of an objective function is given by the second derivative of it, and in multidimensional cases, it is called the Hessian matrix. The Hessian matrix consists of second-order partial derivatives with respect to each parameter. If d parameters are being optimized then dimension of Hessian matrix is $H \in \mathbb{R}^{d\times d}$. In a few simple cases, convergence can occur even in a single step. The major limitation is that it needs an initial guess of parameters closer to the true optimum. Convergence of newton's method largely depends on its initial guess [24]. Here in our case, the direct objective function is not available, hence gradient can not be computed directly. Finite difference approximation is used to calculate both gradient and hessian matrix[3]. In our application $\underline{\theta} \in \mathbb{R}^2$ is the parameter vector hence the dimension of Hessian matrix is $H \in \mathbb{R}^{2\times 2}$.

In the current setup, we have two parameters that need to be optimized i.e. azimuth and elevation angle. So in hessian matrix calculation, the objective function is $f(\underline{\theta})$ and $\underline{\theta} \in \mathbb{R}^2$ is the parameter vector. As already mentioned objective function is not available directly so it is evaluated at a certain parameter vector $\underline{\theta}^*$ in order to calculate gradients using finite difference approximation. Here θ^1 and θ^2 are some initial parameter values in the parameter vector $\underline{\theta}$.

$$y_+^{(1)} = f(\theta^{(1)} + \delta, \theta^{(2)}) \tag{58}$$

$$y_-^{(1)} = f(\theta^{(1)} - \delta, \theta^{(2)}) \tag{59}$$

$$y_+^{(2)} = f(\theta^{(1)}, \theta^{(2)} + \delta) \tag{60}$$

$$y_-^{(2)} = f(\theta^{(1)}, \theta^{(2)} - \delta) \tag{61}$$

$$\hat{g}(\theta^{(1)}) = \frac{y_+^{(1)} - y_-^{(1)}}{2 \cdot \delta} \tag{62}$$

$$\hat{g}(\theta^{(2)}) = \frac{y_+^{(2)} - y_-^{(2)}}{2 \cdot \delta} \tag{63}$$

From the above expression, the first order gradient is completed where $\hat{g}(\theta^{(1)})$ and $\hat{g}(\theta^{(2)})$ are the elements of gradient vector $\underline{\hat{g}}(\underline{\theta})$. Now in similar way second order partial derivatives with respect to each parameter is calculated whereas θ^1 and $\theta^{(2)}$ are kept same as above in equation (58) and (59).

$$y^{(0)} = f(\theta^{(1)}, \theta^{(2)}) \tag{64}$$

$$y^{(11)} = f(\theta^{(1)} + \delta, \theta^{(2)} + \delta) \tag{65}$$

$$y^{(12)} = f(\theta^{(1)} + \delta, \theta^{(2)} - \delta) \tag{66}$$

$$y^{(21)} = f(\theta^{(1)} - \delta, \theta^{(2)} + \delta) \tag{67}$$

$$y^{(22)} = f(\theta^{(1)} - \delta, \theta^{(2)} - \delta) \tag{68}$$

Using above function evaluations each element in hessian matrix $H \in \mathbb{R}^{2\times 2}$ is calculated.

$$H = \begin{bmatrix} h^{(11)} & h^{(12)} \\ h^{(21)} & h^{(22)} \end{bmatrix} \tag{69}$$

$$h^{(11)} = \frac{y_+^{(1)} - 2 \cdot y^{(0)} + y_-^{(1)}}{\delta^{(2)}} \tag{70}$$

$$h^{(12)} = \frac{y^{(11)} - y^{(12)} - y^{(21)} + y^{(22)}}{4 \cdot \delta^{(2)}} \tag{71}$$

$$h^{(21)} = \frac{y^{(11)} - y^{(21)} - y^{(12)} + y^{(22)}}{4 \cdot \delta^{(2)}} \tag{72}$$

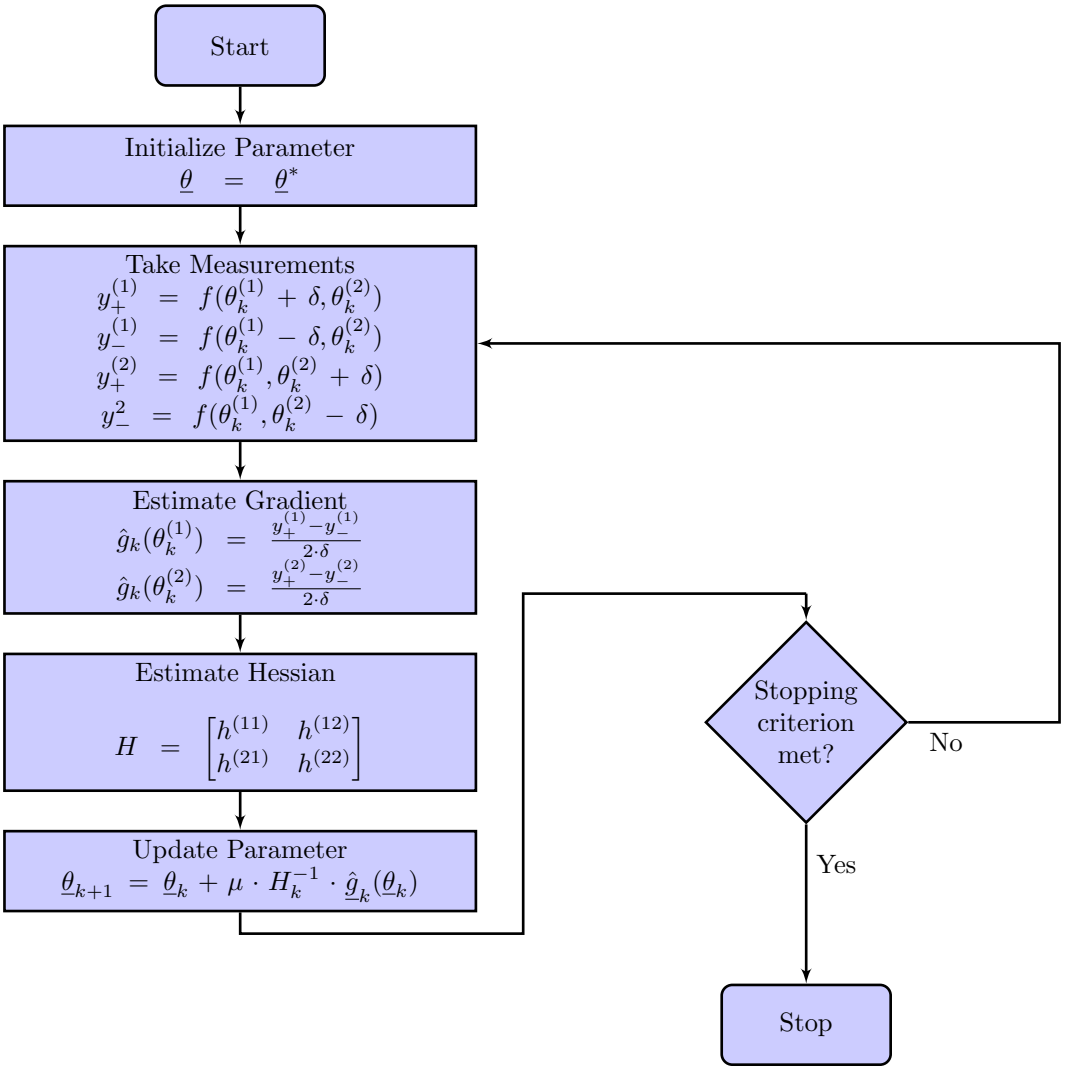
$$h^{(22)} = \frac{y_+^{(2)} - 2 \cdot y^{(0)} - y_-^{(2)}}{\delta^{(2)}} \tag{73}$$

After calculating the Hessian matrix, the inverse of the same is computed which can be inserted into the parameter update equation. If the Hessian matrix is not positive definite, then minimization or maximization of an objective function can't be achieved. So the necessary condition for the Newton's method is hessian matrix must be invertible and positive definite. The update equation can be written as,

$$\underline{\theta}_{k+1} = \underline{\theta}_k + \mu \cdot H_k^{-1} \cdot \underline{\hat{g}_k}(\underline{\theta}_k) \tag{74}$$

In the above update equation (74) $H^{-1} \in \mathbb{R}^{2\times 2}$ is the inverse of hessian matrix, $\underline{\hat{g}}(\underline{\theta}) \in \mathbb{R}^2$ is the estimated gradient vector and $\underline{\theta} \in \mathbb{R}^2$ is the parameter vector.

9.2 Flowchart



10 Quasi Newton Method

The quasi newtons method is another method used to minimize or maximize an objective function. Fundamentally, quasi-newton's method is the same as newton's method except for a few modifications. In newton's method hessian matrix is computed at each iteration, which becomes computationally expensive in the multidimensional case where it requires computing second-order partial derivatives with respect to each parameter[23]. In the case of Quasi Newton's method hessian matrix is not computed rather, it is updated in every iteration[23]. The algorithm starts with some initial guess of the hessian matrix which should be positive-definite, and in each iteration the hessian is updated. Quasi Newton's method require same number of function evaluations as gradient ascent.

10.1 Algorithm Overview

The same procedure is followed as Newton's method except in the calculation of Hessian matrix. At the start of the algorithm hessian is initialized with a scaled identity matrix $H = \lambda \cdot I$. In the next step, the gradient is approximated using finite difference approximation and the hessian matrix is updated using the difference between successive gradients[23]. The update equation for the hessian matrix is as follows,

$$\underline{\theta}_{k+1} = \underline{\theta}_k + \underline{\Delta}\theta \tag{75}$$

$$H_{k+1} = H_k + \frac{\underline{b}_k \cdot \underline{b}_k^T}{\underline{b}_k^T \cdot \underline{\Delta}\theta} - \frac{H_k \cdot \underline{\Delta}\theta \cdot (H_k \cdot \underline{\Delta}\theta)^T}{\underline{\Delta}\theta^T \cdot H_k \cdot \underline{\Delta}\theta} \tag{76}$$

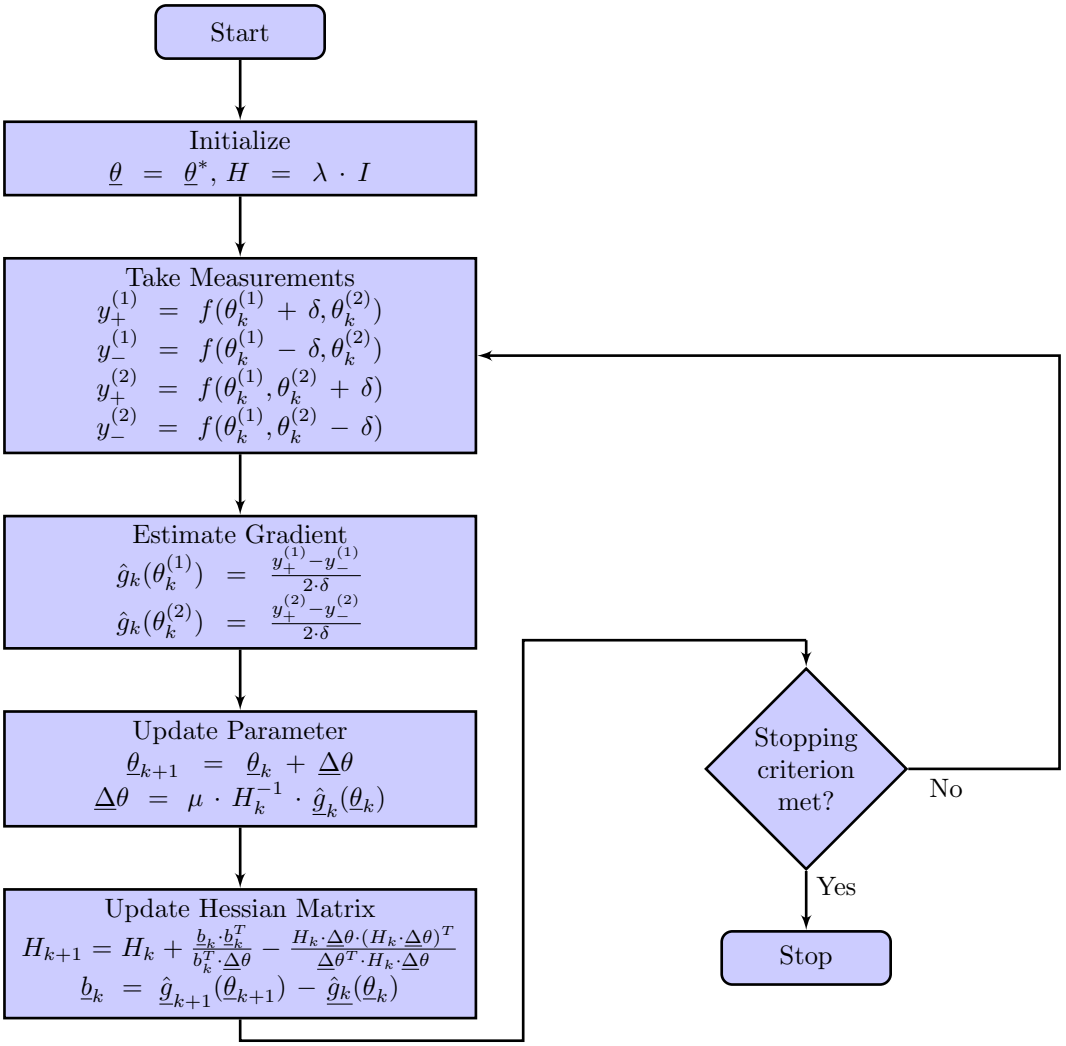
$$\underline{b}_k = \underline{\hat{g}}_{k+1}(\underline{\theta}_{k+1}) - \underline{\hat{g}}_k(\underline{\theta}_k) \tag{77}$$

$$\underline{\Delta}\theta = \mu \cdot H_k^{-1} \cdot \underline{\hat{g}}_k(\underline{\theta}_k) \tag{78}$$

In the above expressions $\underline{\theta} \in \mathbb{R}^2$ is the parameter vector, H_{k+1} is the updated hessian matrix, $\underline{\hat{g}}(\underline{\theta})$ is the approximated gradient; $\underline{b}_k$ is the difference between successive gradients, $\underline{\Delta}\theta$ is the change to be updated in the parameter, and μ is the step size. In our application, the initial guess of the Hessian matrix is taken as a scaled identity matrix. The scaling factor is taken as a positive real number to make sure that the hessian matrix is positive definite. The above update equation for hessian matrix (76) try to minimize $\|H_{k+1} - H_k\|_F$. Also the algorithm allows to update H^{-1} without computing second order derivatives in each iteration which in turn reduces computational complexity.

10.2 Flowchart

In the below flowchart, $\theta^{(1)}$ and $\theta^{(2)}$ are the elements of parameter vector $\underline{\theta} \in \mathbb{R}^2$. Also $\hat{g}(\theta^{(1)})$ and $\hat{g}(\theta^{(2)})$ are the elements of gradient vector $\underline{\hat{g}}(\underline{\theta})$.



11 Summary and Interpretation of Part II

In the section on stochastic approximation, different stochastic approximation-based algorithms are explained from a theoretical and implementation perspective. At the start of the stochastic approximation section, a short introduction about stochastic approximation and the motivation for studying and using these methods is provided. Each algorithm is explained along with the mathematics and parameters involved in it. Fundamentally, in the execution of all algorithms some common steps are being followed. The intuition behind these steps is conveyed in the introductory section of each algorithm. Various parameters are involved in the mathematics of the respective algorithm. These parameters play a certain role in the overall performance of the algorithm. The meaning of each parameter and the range in which it operates is tried to be conveyed in each section. In the description of each algorithm, the advantages and disadvantages it possesses are highlighted. From an implementation perspective, at the end of each section of the respective algorithm a flowchart is provided depicting the implementation steps and flow of execution.

Part III
Calibration Framework

12 Calibration Framework

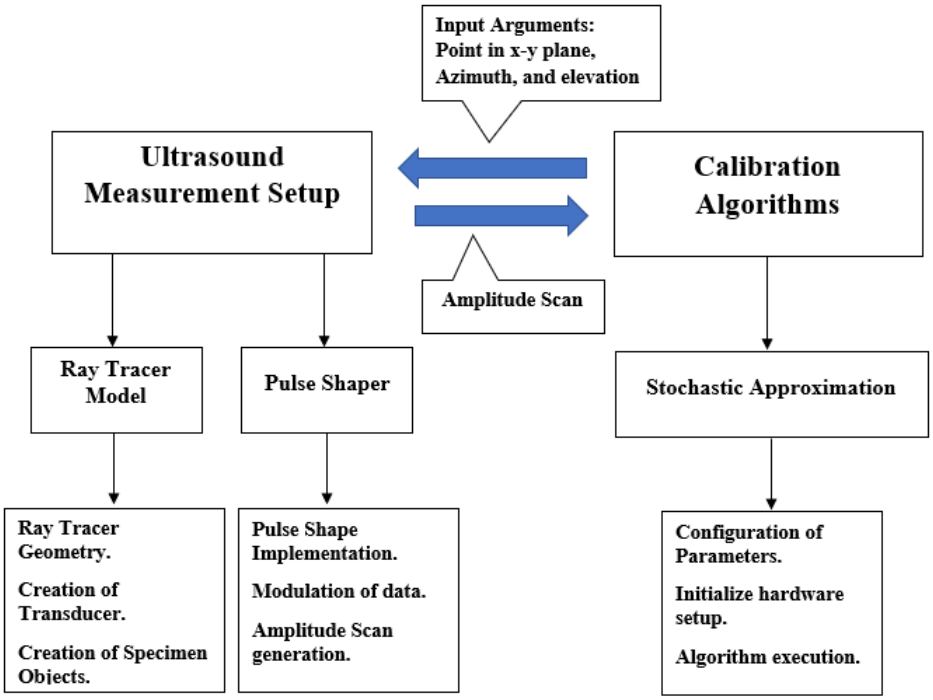


Figure 6: Calibration Framework

12.1 Ultrasound Measurement Setup

The presented virtual ultrasound measurement setup in Fig. 6 consists of two sections working in combination to simulate ultrasound measurement operation. The first section deals with the creation of ray tracer elements, and the second section deals with the simulation of measurements using a mathematical model given in subsection 3.1. Raytracer model involves emulating the geometric aspects of physical measurement setup and creation of an equivalent pinhole camera model emulating transducer operation as explained in section 2.1.2. The second section focuses more on transducer-specific parameters and signal processing aspects for data generation. While starting an operation first, the ray tracer model is initialized which creates the necessary components for the measurement setup. A transducer is initialized with the necessary arguments and starts shooting rays over the specimen. In order to process reflected rays, a mathematical model is invoked which generates a carrier and pulse shape. The parameters of the rays modulate pulse shape parameters which further modulate the carrier. In the last step, all the scaled, time-shifted copies are summed up to get one A-scan vector.

12.1.1 Test Scenarios

In this section the virtual ultrasound measurement setup illustrated in Fig. 6 is tested under different measurement scenarios for the calculation of total field. The measurement scenario is based on the specimen object given in Fig. 7. Transducer is moved over the specimen object at specific point and measurement is obtained by shooting rays over the object. According to the equation (28) the received rays are modulated and summed up to get total field.

12.1.1.1 Example Scenario 1

In this scenario transducer is moved over the specimen at point $(-10\,\text{mm}, 0, 22\,\text{mm})$ and measurement is obtained. Further, the transducer is moved to another point $(0, 0, 20\,\text{mm})$ as shown in Fig. 7 and again

measurement is obtained.

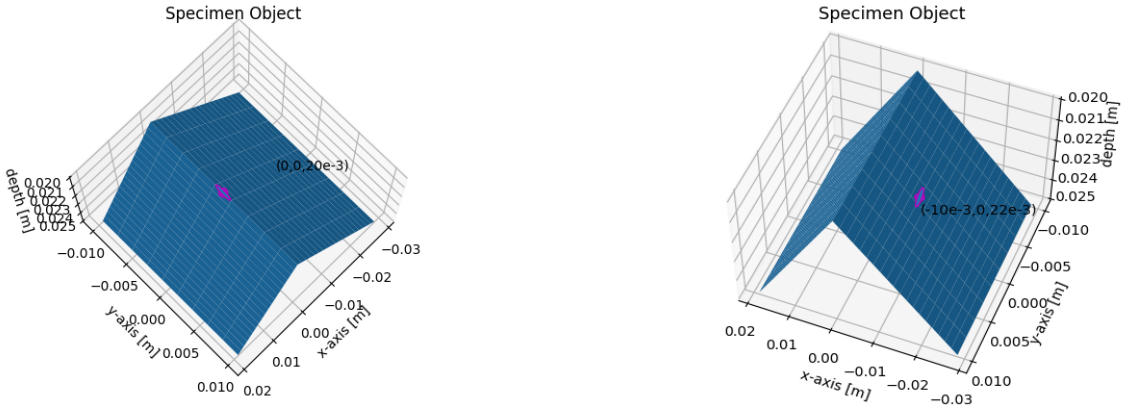


Figure 7: Specimen object from different viewing angles

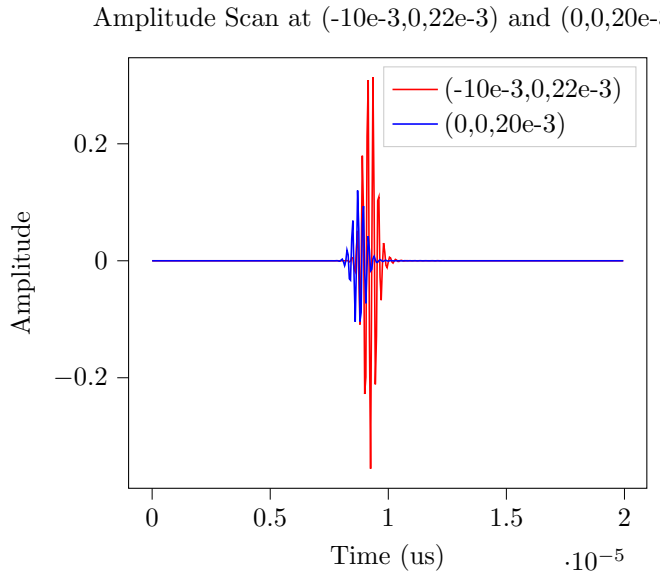


Figure 8: Amplitude Scans obtained at different measurement points for scenario 1

The total field generated over transducer sensor area after taking measurements at different points over the specimen is compared in Fig. 8.

12.1.1.2 Example Scenario 2

In this scenario transducer is moved over the specimen at point $(10\,\text{mm}, 0, 22\,\text{mm})$ and Further, the transducer is moved to another point $(0, 0, 20\,\text{mm})$ as shown in Fig. 9 and again measurement is obtained.

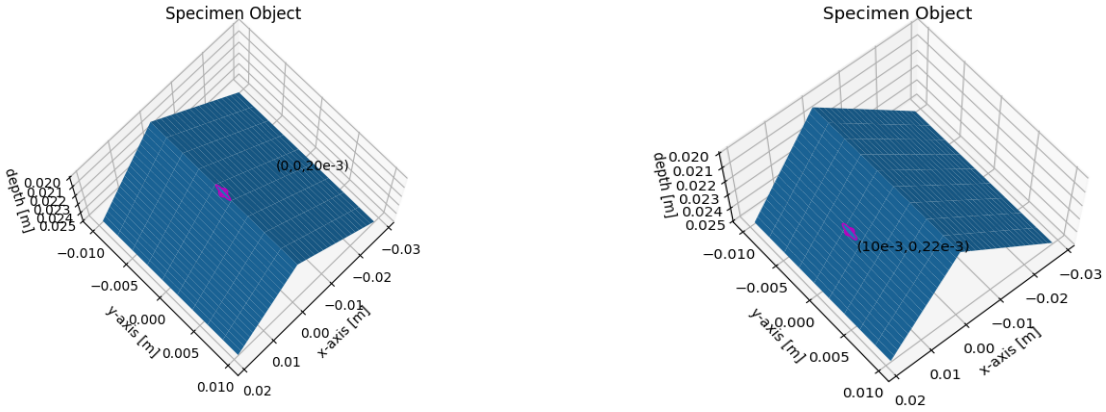


Figure 9: Specimen object from different viewing angles

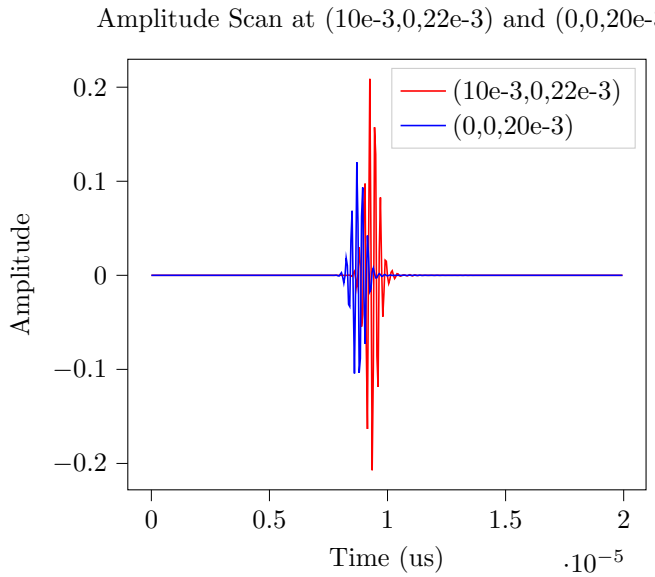


Figure 10: Amplitude Scans obtained at different measurement points for scenario 2

The total field generated over transducer sensor area after taking measurements at different points over the specimen is compared in Fig. 10.

12.2 Calibration Algorithms

Calibration Algorithms is an independent section given in Fig. 6 that can process data collected from the virtual platform or physical hardware setup. It consists of a set of calibration methods that allow faster and automatic calibration of transducer orientations. The goal of these methods is to make the direction of the ultrasound wave vector parallel to the surface normal as discussed in earlier sections. In this framework, calibration algorithms use the interface provided by the ray tracer and collect measurements at the desired position. The collected measurements are processed by calibration methods and results are returned.

12.2.1 Test Scenarios

In this section, different stochastic approximation based calibrations algorithms explained in section 5 are tested under different parameter settings. These algorithms are collecting data from the virtual measurement setup explained in section 12.1, though each of these algorithms are independent of data

source. In each configured scenario one of the concerned parameters is varied and others are kept constant. This allows us to study and analyze performance of each method individually under certain parameter settings. Also, inter algorithm performance analysis is also performed by considering a certain scenario which allows us to compare performance of algorithms being run under the same parameter settings.

12.2.1.1 Stochastic Gradient Ascent

The rate of convergence of stochastic gradient Ascent depends largely on the step size being used in the algorithm. Gradient Ascent uses fixed step sizes and it is interesting to investigate the effect of step size on performance and rate of convergence. Hence, a test scenario is considered where different values of step sizes are used to study and understand the effect on cost function plots. The Pseudo-Energy vs. Number of iterations explains the effect of step size on rate of convergence. The other plot shows the cost function and path traced by the parameters that need to be optimized. The Pseudo Energy can be mathematically expressed as follows,

$$E = \|\underline{x}\|^2 \tag{79}$$

E is the pseudo-energy and $\underline{x}$ is the A-scan. In this work, while calculating pseudo-energy instead of considering all the samples in A-scan vector, a window of samples is considered based on time of flight. Matched filtering is performed between the received time-shifted copy and known template of signal and the center of the window is placed where the peak is observed in the filter output. The calculated pseudo-energy is maximum when the transducer points normal to the specimen surface under test with the assumption of lambertian scattering model[8, pp. 13–15].

The results obtained are displayed in Fig. 11 and Fig. 12. The plot shown in Fig. 11 shows the effect of step size on rate of convergence and heat map displayed in Fig. 12 shows the path traced by two parameters over the cost function. From the plot Fig. 11 it is observed that as the step size is increased, the rate of convergence increases. Also, the number of iterations required for the largest step size is very small as compared to the other curves. As from the cost function plot given in Fig. 12, the estimated parameter reaches the true solution in the given number of iterations for the largest step size.

Table 1: Test Scenario

Sr.No	Parameters	Value
1	Initial guess Azimuth	0
2	Initial guess elevation	0
3	SNR	30 dB
4	Iterations	30

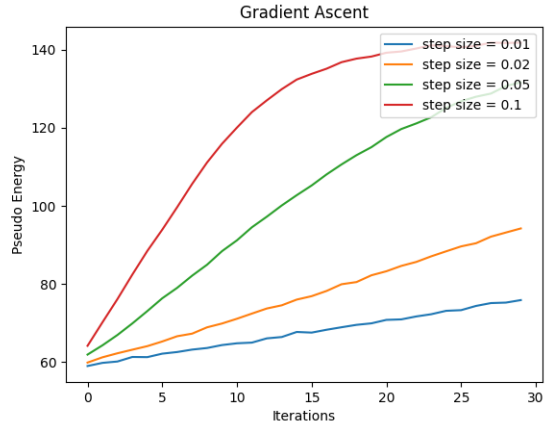


Figure 11: Pseudo Energy vs Number of Iterations required for convergence

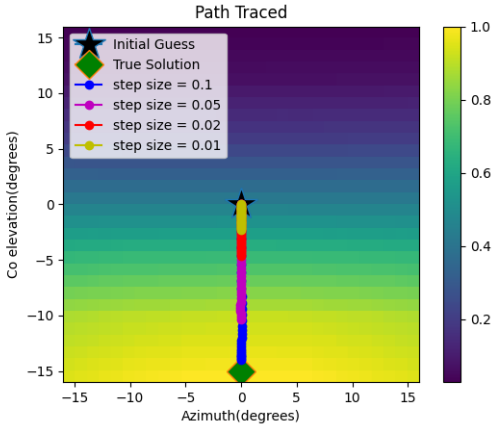


Figure 12: Path traced by estimated parameters over the cost function

12.2.1.2 Variants of Stochastic Gradient Ascent

Stochastic Gradient Ascent with Momentum : The modified stochastic gradient ascent explained in section 7.1 introduces a new parameter called "Momentum". It is important to understand the impact of Momentum on the rate of convergence and stability of an algorithm. Therefore, different values of momentum are used and the number of iterations required to reach a true solution is observed. Also, if the momentum value is large then its effect on the stability of the algorithm is studied. While conducting this test, other parameters such as signal-to-noise ratio and step size are kept constant.

The results obtained are displayed in Fig. 13 and Fig. 14. It is observed that the rate of convergence increases as the momentum factor increases. However, from the plots shown in Fig. 13 and Fig. 14, it is also observed that for higher momentum values oscillations are produced when the algorithm is about to converge. Hence, it is necessary to maintain a balance while choosing momentum factor such that it does not produce any oscillations also better rate of convergence than original stochastic gradient ascent is obtained.

Table 2: Test Scenario

Sr.No	Parameters	Value
1	Step Size	0.05
2	Initial guess Azimuth	0
3	Initial guess elevation	0
4	SNR	30 dB
5	Iterations	30

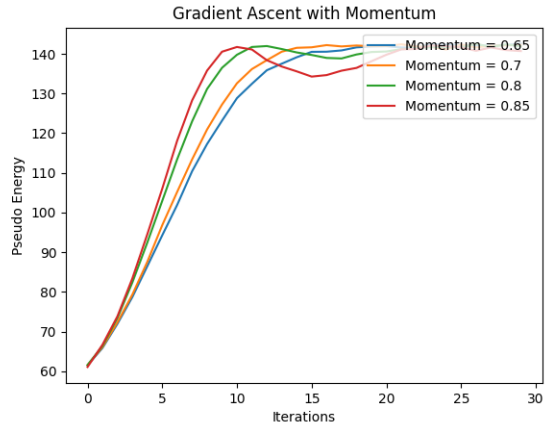


Figure 13: Pseudo Energy vs Number of Iterations required for convergence

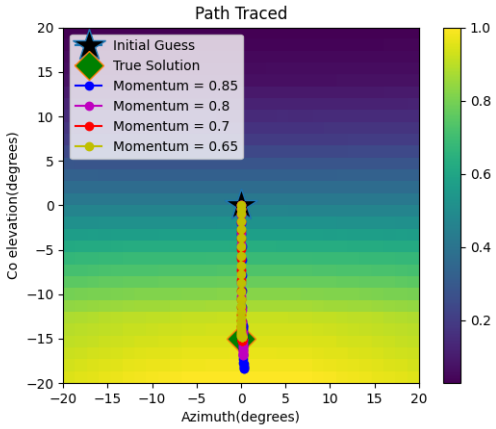


Figure 14: Path traced by estimated parameters over the cost function

Nesterov's Accelerated Gradient : In order to test this algorithm same test scenario being configured in 2 is used. Different momentum values are used and cost function plots are studied for the change in rate of convergence with momentum. In the results shown in Fig. 15 and Fig. 16 it is observed that for higher momentum values the oscillations produced are less as compared to gradient ascent with momentum. The correction factor involved controls the speed of algorithm and avoids oscillations in the region of convergence.

Table 3: Test Scenario

Sr.No	Parameters	Value
1	Step Size	0.05
2	Initial guess Azimuth	0
3	Initial guess elevation	0
4	SNR	30 dB
5	Iterations	30

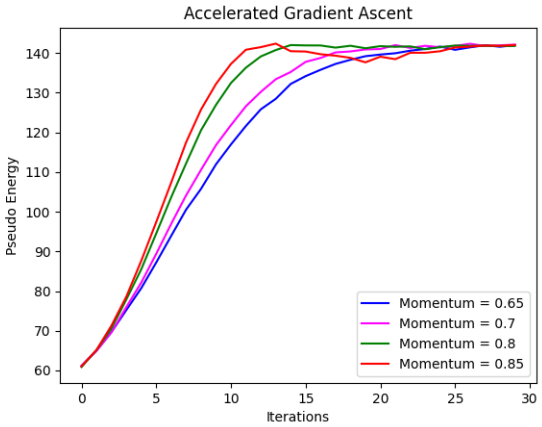


Figure 15: Pseudo Energy vs Number of Iterations required for convergence

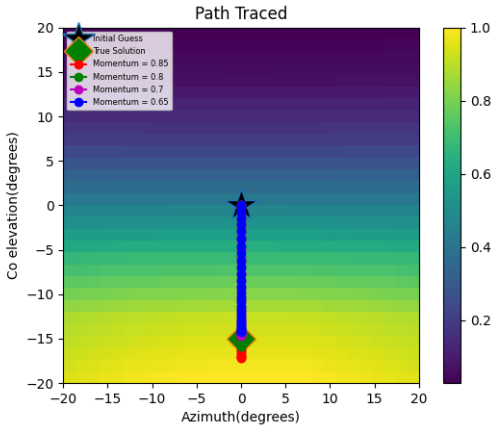


Figure 16: Path traced by estimated parameters over the cost function

12.2.1.3 SPSA

SPSA use adaptive step size where in each iteration step size is updated. Hence, the step size parameter is not fully in our control to understand and analyse the performance of algorithm. In order to analyse performance different signal to noise ratios are used, allowing us to test rate of convergence, stability and accuracy of algorithm. Random noise is generated of different power and added to the data generated by ray tracer model. In the generated cost function plot given in Fig.18 at lower values of signal to noise ratio's (SNR) whether algorithm remains stable and estimates parameter closer to the true optimum is observed. Also the variance in Energy vs Number of iterations plot displayed in Fig. 17 is observed for different signal to noise levels.

Table 4: Test Scenario

Sr.No	Parameters	Value
1	Initial guess Azimuth	0
2	Initial guess elevation	0
3	ρ	0.602
4	γ	0.101
5	c'	1
6	A	5
7	ϵ	0.1
8	Iterations	30

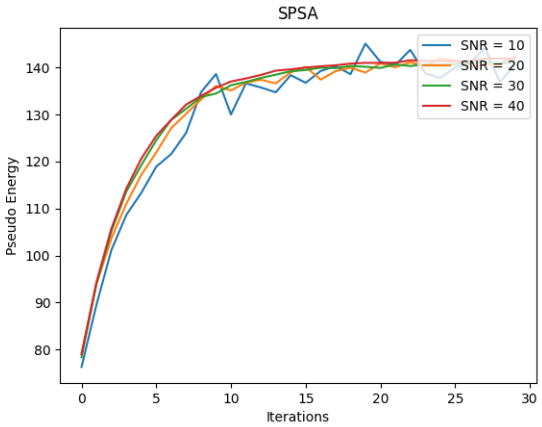


Figure 17: Pseudo Energy vs Number of Iterations required for convergence

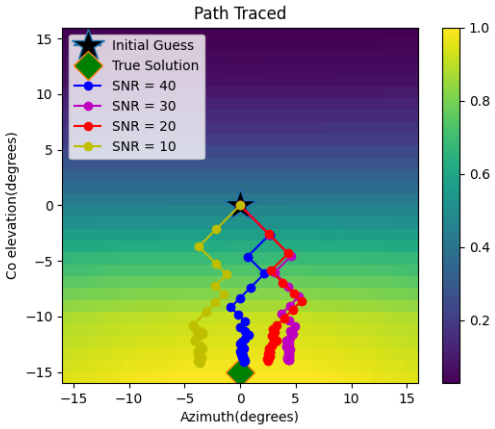


Figure 18: Path traced by estimated parameters over the cost function

12.2.1.4 Newton's Method

The convergence of Newton's method often requires an initial guess of a parameter closer to the true optimum. Hence, a test is conducted by running the algorithm with different initial guess of parameter (θ, ϕ). In the results obtained, how far away an initial guess from the true optimum could be set, and still the algorithm would converge is studied.

The results obtained after conducting the test are displayed in Fig. 19 and Fig. 20. When the initial guess of a parameter is far away from the true optimum, the algorithm diverges as shown in Fig. 20. In order to have a better initial guess, any other algorithm which does not have stringent requirements for initial guess is run for a few iterations and then Newton's method is invoked. However while conducting this test, no other algorithm is run rather the initial guesses are set manually.

Table 5: Test Scenario

Sr.No	Parameters	Value
1	Step Size	0.05
2	SNR	30 dB
3	Iterations	30

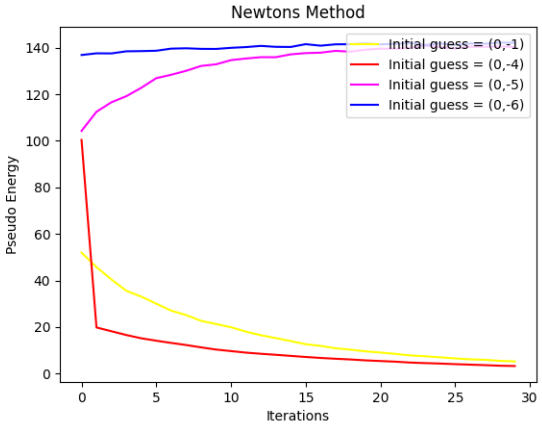


Figure 19: Pseudo Energy vs Number of Iterations required for convergence

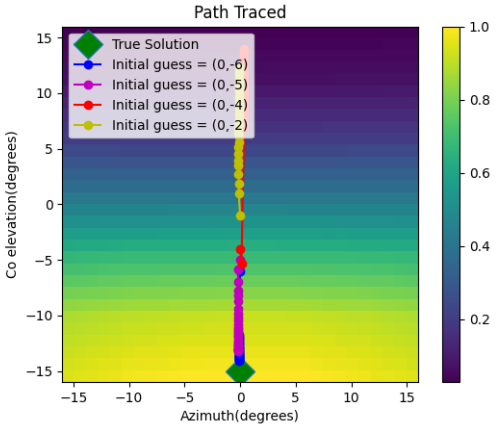


Figure 20: Path traced by estimated parameters over the cost function

12.2.1.5 Quasi Newton's Method

In Quasi Newton's method, the hessian matrix is not computed instead, it is initialized with a scaled identity matrix and updated in each iteration using equation(76). In the example test scenario, the effect of scaling factor λ, which scales an identity matrix used to initialize hessian matrix, is studied. Different scaling factors are used while running an algorithm and its effect on the rate of convergence is observed from the generated cost-function plot.

From the results shown in Fig. 21 and Fig. 22 it can be said that for smaller scaling factors the rate convergence is much higher as compared to large scaling factors.

Table 6: Test Scenario

Sr.No	Parameters	Value
1	Initial guess Azimuth	0
2	Initial guess elevation	0
3	Step Size	0.05
4	SNR	30 dB
5	Iterations	30

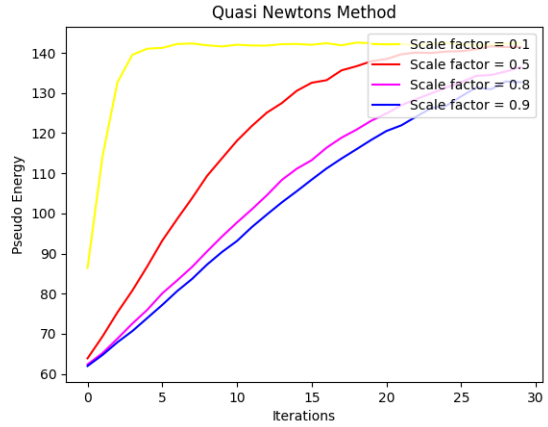


Figure 21: Pseudo Energy vs Number of Iterations required for convergence

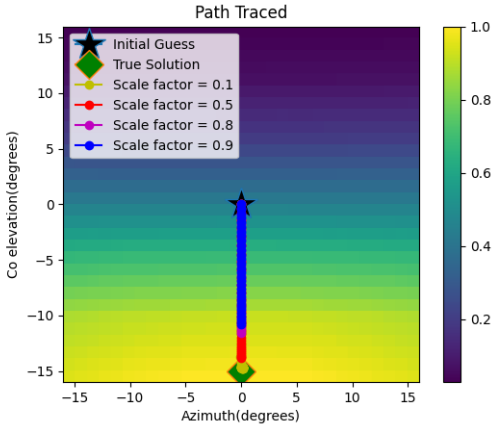


Figure 22: Path traced by estimated parameters over the cost function

12.2.1.6 Algorithm Performance Comparison

In the previous subsections, performance analysis of individual algorithms is carried out by varying certain parameters. It is also interesting to investigate and compare the performance of algorithms with each other under certain fixed test scenarios. As mentioned in section 1.2 the basic performance metric of the number of iterations required for convergence is taken into consideration for comparison. The number of function evaluations performed by each algorithm in each iteration decides the computational cost per iteration. Hence, if an algorithm is computationally expensive, then concurrently the time required for each algorithm to complete its execution increases. Also, the term function evaluation in a real-world setting means changing transducer orientation using a hardware positioning system and taking a measurement which itself is time-consuming in nature. Therefore, the algorithm which takes a minimum number of iterations, in turn, a less number of function evaluations can be considered to be the optimum one.

The parameters mentioned in the table 7 are not applicable to all algorithms. Some of the parameters such as Scaling factor is only used by Quasi Newton's Method; also the Step Size is not used by SPSA but used by all other algorithms. The results obtained are displayed in Fig. 23 and Fig. 24. In the configured test scenario, all algorithms are using the same initial guess; hence, if the pseudo-energy curve for Newton's method is observed then it can be confirmed that the method needs a better initial guess. In practice, the better initial guess can only be obtained by running a few iterations of some other algorithm which is not sensitive to the initial guess. Also, Newton's method requires 9 function evaluations per

Table 7: Test Scenario

Sr.No	Parameters	Value
1	Initial guess Azimuth	0
2	Initial guess elevation	0
3	Step Size	0.05
4	SNR	30 dB
5	Iterations	30
6	Momentum	0.7
7	Scaling factor	0.9

iteration and computation of the hessian matrix, which itself is computationally expensive. Therefore, Newton's method can't be used unless the approximate true solution is already known so that the initial guess can be set closer to it. In the case of Quasi Newton's method, the hessian matrix computation is not performed in each iteration; rather it is updated hence it is computationally less expensive. From the result plot shown in Fig. 23 it has similar rate of convergence as gradient ascent; also, the number of function evaluations is same as gradient ascent.

In the case of gradient ascent, the number of function evaluations required is twice the number of parameters to be optimized. It becomes computationally expensive when the number of parameters are increased, and hence, not suitable for real time applications. However, its speed can be increased by increasing the step size. Variants of gradient ascent introduce a momentum factor in their update equation. Because of momentum there is a significant increase in the rate of convergence; however, while individual testing of algorithm done in subsection 2 it is observed that if the value is not controlled then oscillations are produced in the region of convergence. In the case of accelerated gradient the performance is similar to gradient ascent with momentum but for higher momentum values, the oscillations are controlled by accelerated gradient compared to gradient ascent with momentum. Both algorithms need the same number of function evaluations as the original gradient ascent.

SPSA takes 2 function evaluations per iteration irrespective of the number of parameters being optimized, which makes it computationally efficient. It uses adaptive step size where in each iteration it is updated. Also, it requires less number of iterations for convergence as can be observed in Fig. 23. Because of less computational complexity and faster rate of convergence, SPSA can be used in real world applications.

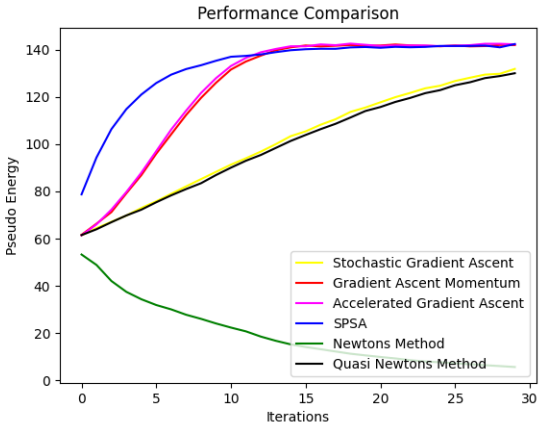


Figure 23: Pseudo Energy vs Number of Iterations required for convergence

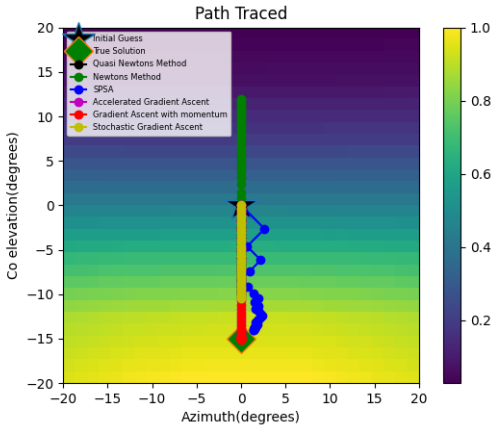


Figure 24: Path traced by estimated parameters over the cost function

Conclusion

In this thesis, a testbed for the calibration of ultrasound transducers is developed. The main goal in developing this framework is to be able to perform calibration without any hardware dependency and with less computational complexity. The available calibration techniques involve hardware dependency, which makes the calibration process time-consuming and costly. The proposed framework contains a virtual hardware setup that can simulate ultrasound measurements, and the algorithm framework consists of several calibration algorithms.

In the first part of this thesis, a ray tracer model is introduced. The purpose of the ray tracer is to create a virtual ultrasound measurement setup to simulate measurement operation. Ray tracer has a transducer, specimen, and a positioning system to change transducer position and orientation. In ray tracer wave propagation is simplified by treating ultrasound wavefields as discrete rays while assuming that the medium is homogeneous and the frequency of operation is high[9][5]. The operation of the transducer is treated similarly to that of a camera as the waves are assumed to be a set of rays traveling in straight lines. Basic geometric elements are used to create transducer and specimen object[6]. A mathematical model is used which modulates each reflected ray with a pulse shape and its associated amplitude and time of flight. Pulse modulated signal is multiplied with a carrier frequency and all the scaled time-shifted copies are summed up to get the total field[11]. Ray tracer approximates ultrasound wave propagation and allows hardware-independent simulation of measurement operation.

The second part deals with the actual calibration algorithm framework. The importance of stochastic approximation in the automation of the calibration process is discussed[4]. A set of stochastic approximation-based algorithms are studied in order to automate the calibration process. Each algorithm is discussed in detail with a general introduction and mathematics involved in the flow of execution. The parameters involved in the mathematical description of each algorithm are explained along with the operating range of their values.

The final part of this thesis explains sections of the calibration framework and its interfaces. The first part of this calibration framework is the creation of measurement setup and the second is the algorithm framework. Usage of measurement setup to simulate measurement operation is introduced by evaluating different test scenarios to compare total field generated. An interface provided by the measurement setup is used by the algorithm framework to collect measurements. Components in the algorithm framework are introduced and discussed in detail. These algorithms collect measurements at the desired position and process them to find the optimum angle. Various parameters which influence the performance of the algorithm are studied and analysed[3]. These parameters are manipulated individually in a certain range to analyze their impact on the cost function. Cost functions of different algorithms are compared in a certain parameter setting[3]. Tests were conducted under varying signal-to-noise ratios on specific algorithms and cost functions were compared for stability. Also, the sensitivity of an algorithm to its initial guess is tested. The main purpose of these tests is to get an indication of the performance of each algorithm in a simulated scenario before going forward in a real-world setting.

The calibration framework proposed in this work greatly reduces the time and computational complexity involved in ultrasound calibration. Also, it provides an opportunity to manipulate transducer and algorithm-specific input parameters for further analysis which in physical hardware is not possible. The algorithm part is completely independent of the measurement setup. It provides necessary arguments to measurement setup about the position and orientation of transducer and collects measurement. Several simplistic assumptions were made in the ray tracer model used for the creation of the measurement setup. These assumptions allow a simple simulation of wave propagation in ultrasound, reducing time and computational complexity[9]. The performance of any calibration algorithm in such a simulated environment provides an indication of the performance of the same in a real-world setting.

References

[1] Peter Sturm. *Pinhole Camera Model*. Springer US, Boston, MA, 2014.

[2] M.G. Duncan. Real-time analysis signal processor for ultrasonic nondestructive testing. *IEEE Transactions on Instrumentation and Measurement*, 39(6):1024–1029, 1990.

[3] D. C. Chin. Comparative study of stochastic algorithms for system optimization based on gradient approximations. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 27(2):244–249, 1997.

[4] J. C. Spall. Implementation of the simultaneous perturbation algorithm for stochastic optimization. *IEEE Transactions on Aerospace and Electronic Systems*, 34(3):817–823, 1998.

[5] Zhengqing Yun and Magdy F. Iskander. Ray tracing for radio propagation modeling: Principles and applications. *IEEE Access*, 3:1089–1100, 2015.

[6] David Eberly. *3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics*. 01 2001.

[7] S. C. Wooh and Yijun Shi. Three-dimensional beam directivity of phase-steered ultrasound. *Journal of the Acoustical Society of America*, 105:3275–3282, 1999.

[8] Frank L. Pedrotti, Leno M. Pedrotti, and Leno S. Pedrotti. *Introduction to Optics*. Cambridge University Press, 3 edition, 2017.

[9] Sandy Sefi. Ray tracing tools for high frequency electromagnetics simulations. 01 2003.

[10] Keiichi Ito and Tom Dhaene. Adaptive initial step size selection for simultaneous perturbation stochastic approximation. *SpringerPlus*, 5, 02 2016.

[11] Ramazan Demirli and Jafar Saniie. Model-Based Estimation of Ultrasonic Echoes. Part I: Analysis and Algorithms. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 48(3):787–802, 2001.

[12] James Wiskin, David T. Borup, S. A. Johnson, and Michael J. Berggren. Non-linear inverse scattering: high resolution quantitative breast tissue tomography. *The Journal of the Acoustical Society of America*, 131 5:3802–13, 2012.

[13] Chang-Fa Yang, Boau-Cheng Wu, and Chuen-Jyi Ko. A ray-tracing method for modeling indoor wave propagation and penetration. *IEEE Transactions on Antennas and Propagation*, 46(6):907–919, 1998.

[14] Carlo Tomasi. A simple camera model. 2015.

[15] K. Åhlander. Einstein summation for multidimensional arrays. *Computers Mathematics with Applications*, 44(8):1007–1017, 2002.

[16] Mark Owen. *Practical Signal Processing*. Cambridge University Press, USA, 2007.

[17] Yanghua Wang. The Ricker wavelet and the Lambert W function. *Geophysical Journal International*, 200(1):111–115, 11 2014.

[18] Florian Boßmann, Gerlind Plonka, Thomas Peter, Oliver Nemitz, and Till Schmitte. Sparse deconvolution methods for ultrasonic ndt. *Journal of Nondestructive Evaluation*, 31, 09 2012.

[19] Simon Haykin. Adaptive filter theory. 1986.

[20] Sebastian Ruder. An overview of gradient descent optimization algorithms. 09 2016.

[21] Hiroshi Ninomiya. Neural network training based on quasi-newton method using nesterov's accelerated gradient. In *2016 IEEE Region 10 Conference (TENCON)*, pages 51–54, 2016.

[22] Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. Advances in optimizing recurrent networks, 2012.

[23] Aryan Mokhtari and Alejandro Ribeiro. Stochastic quasi-newton methods. *Proceedings of the IEEE*, 108(11):1906–1922, 2020.

[24] Jingchen Liang. Gradient descent and newton's method with backtracking line search in linear regression. In *2021 2nd International Conference on Computing and Data Science (CDS)*, pages 394–397, 2021.

Notation and Symbols

t	time
ω	frequency of operation
f_s	sampling frequency
f_c	center frequency
T_0	sampling period
c_0	speed of sound in a medium
$s'(t)$	total field generated in continuous setting
$\underline{\theta}$	parameter vector
y_+	function evaluation along positive step
y_-	function evaluation along negative step
λ	scaling factor for identity matrix in hessian matrix initialization
$\underline{\Delta}$	perturbation vector
β	amplitude based on angle of departure
a	amplitude based on angle of incidence
μ	step size
E	pseudo energy
$\underline{\hat{g}}(\underline{\theta})$	gradient vector
$p(t)$	pulse shape
T	transformation matrix
G	global coordinates tensor
L	local coordinates tensor
n_v	number of vertical pixels
n_h	number of horizontal pixels
t'	distance at which ray intersection occur
α	bandwidth factor
$\underline{d}$	direction vector of a ray
$\underline{c}$	center of plane
$\underline{f}$	focus of imaging plane
l	focal length
$\underline{n}$	normal vector
τ	time of flight
$\underline{s}$	source coordinates of a ray
w	width of plane
h	height of plane
k	iteration index
δ	step size while taking function evaluation
H	hessian matrix

Abbreviations

A-scan Amplitude scan.

SNR Signal to Noise Ratio.

SPSA Simultaneous Perturbation Stochastic Approximation.

UNDT Ultrasonic Non-destructive Testing.

Declaration

I declare that the work is entirely my own and was produced with no assistance from third parties.
I certify that the work has not been submitted in the same or any similar form for assessment to any other examining body and all references, direct and indirect, are indicated as such and have been cited accordingly.

(Sudhanshu Apte)
Ilmenau, 10^{th} Sept, 2023