Spring 4-17-2017

Optimal Experimental Design to Characterize a Wave Source Using Dosimeter Measurements

Renee L. Gooding
University of New Mexico

Renee Gooding

Candidate

Mathematics

Department

This thesis is approved, and it is acceptable in quality and form for publication:

Approved by the Thesis Committee:

Matthew Pennybacker, Chairperson

Daniel Appelo

Olga Lavrova

Optimal Experimental Design to Characterize a Wave Source using Dosimeter Measurements

by

Renee L. Gooding

B.S., Mathematics, University of New Mexico, 2013

THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science
Mathematics

The University of New Mexico

Albuquerque, New Mexico

May, 2017

Dedication

To my husband, who never fails to support and encourage me in everything I do.

Acknowledgments

A special thank you to my thesis adviser, Matthew Pennybacker. Without his patience and wealth of knowledge this thesis would not have been possible. He spent an incredible amount of time and energy guiding me though every step and his constant willingness to meet with me and respond to my, sometimes several, emails does not go unnoticed. I started this thesis with basic Matlab skills and he was always willing to look through my work and modify it or give me hints on ways to improve it. For that, I am forever grateful as I could not have done this thesis without such integral involvement.

I would also like to thank Daniel Appelö for giving me a nudge a year ago to start seriously trying to figure out what I was interested in doing my thesis on. I also sincerely appreciate his willingness to be on my thesis committee and help whenever his expertise was needed. Also, Olga Lavrova for all her guidance and wisdom throughout my internship, and for providing the inspiration for my thesis topic.

I also want to thank my parents for their undying support and encouragement throughout my entire life. Lastly, to my friends for all being geniuses and inspiring me to keep reaching.

Optimal Experimental Design to Characterize a Wave Source using Dosimeter Measurements

by

Renee L. Gooding

B.S., Mathematics, University of New Mexico, 2013
M.S., Mathematics, University of New Mexico, 2017

Abstract

When modeling physical phenomena we want to solve the inverse problem by estimating the parameters that characterize the source model that we are interested in. In this thesis, we focus on the optimal placement of a finite number of individual sensors, called dosimeters, in $\mathbb{R}^2$ and $\mathbb{R}^3$ with a time dependent Gaussian wave source. Using a computational model along with experimental data, we design an iterative process to determine the optimal placement of an additional sensor such that the noise in the measurements has a minimal effect on the parameter estimation. First, we estimate the parameters that characterize the source model using a non-linear least squares optimization method. Then using the estimated parameters along with statistical analysis, we can determine an optimal location to place an additional sensor. Each time we iterate, we use new experimental data to determine a more accurate parameter estimation and optimal sensor placement. Upon reaching the maximum number of iterations, we can determine the optimal location to place an additional sensor such that we can most accurately characterize the wave source.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

When modeling physical phenomena we often try to solve the inverse problem in which there exists some system or function of interest that has a finite number of unknown parameters, whose values characterizes the system. Through the observable measurements of these physical quantites, we try to accurately estimate the actual parameter values [5].

Often to collect experimental measurements we use individual sensors that record data from the environment in which they are placed. A thermoluminescent dosimeter is a specific type of sensor that measures the dosage of ionizing radiation deposited on it from an electromagnetic wave. Interest lies in finding the optimal place to put a dosimeter such that we can characterize how an electromagnetic wave generated from a known source propagates in a spatial domain. Using the data recorded on each sensor, we want to estimate the parameters of the source model.

It is generally too expensive or impracticable to place a sensor at every point in a discretized physical field, and so placement locations have to be chosen carefully [6]. One benefit of using sensors is the ability to optimally design the experiment. When trying to estimate the parameters that describe the source model, the location of each sensor has a direct effect on the accuracy of the model parameter estimation,

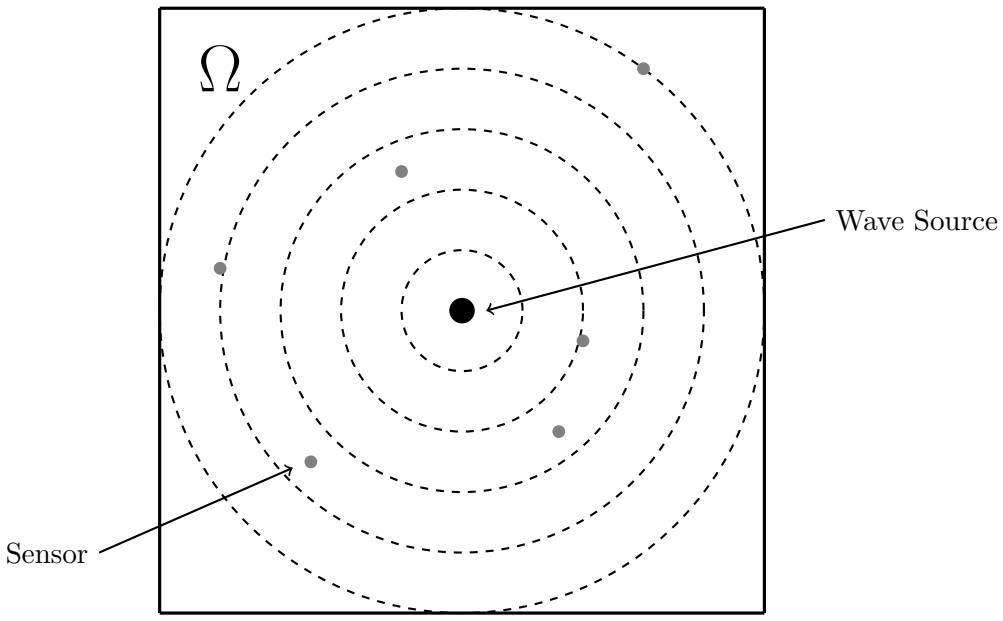


Figure 1.1: Propagating wave generated from a wave source in a bounded domain, Ω, with $N = 6$ sensors.

and so it is crucial that the sensors be placed as best as possible. Through the use of optimization methods and statistical analysis, optimal experimental design strives to choose sensor placements in such a way that the results are reliable and the parameter values can be be accurately estimated. In this thesis, we describe a method to find the optimal placement of a dosimeter such that we can accurately model a wave source.

We consider the scalar wave equation with a Gaussian wave source (1.1), in $\mathbb{R}^2$ that contains N sensors as seen in Figure 1.

$$\begin{aligned} \frac{\partial u}{\partial t} &= b\nabla \cdot \vec{v}, \\ \frac{\partial \vec{v}}{\partial t} &= a\nabla u, \\ f(x,y,t) &= \gamma e^{\frac{-(t-t_0)^2}{\beta^2}} e^{-(\frac{(x-x_0/2)^2}{\alpha^2}+\frac{(y-y_0/2)^2}{\alpha^2})}. \end{aligned}$$

Using measurements, $\vec{c_0}$, from an initial experiment, we estimate the parameters, $\vec{\theta} = (\alpha, \gamma)$, of the wave source. Supposing we want to run the experiment again, but

this time with an additional sensor at $\zeta = (x, y)$, we use a computational model to try to determine where to place the sensor such that we get more accurate estimations for the parameters. This experimental design is an iterative process in which we must estimate the parameters, determine the best location to place the sensor within the domain, and then perform a new experiment using the estimations we acquired.

We start in Chapter 2 with the derivation of a finite difference method to approximate the wave equation in a spatial domain. Additionally, the implementation of Perfectly Matched Layers is introduced. For a computational model, we have to truncate the domain and in doing so, reflections at the boundaries occur. Through the use of Perfectly Matched Layers we can approximate the solution with minimal reflection.

Chapter 3 develops the finite difference method of lines discretization to solve the wave equation in the temporal domain. We discuss the forcing function that is used as the wave source, and the optimization method used to estimate the parameters of the forcing function. In the final chapter, we estimate the covariance matrix of the parameters at each possible additional sensor location. We then define metrics on the covariance matrix for the parameters, and an use a simplified gradient descent algorithm to find where the minimum of each metric occurs. Finally, we describe the iterative method for optimal experimental design such that the parameters estimations are the most accurate.

Chapter 2

The Wave Equation and Perfectly Matched Layers

In this chapter, we derive a staggered grid finite difference method used to approximate partial derivatives in a spatial domain. It also discusses the method of Perfectly Matched Layers to avoid reflection in a bounded domain. We need an accurate and effieicent numerical method to better estimate the parameters of the model and design an improved experiment.

2.1 The 1D Wave Equation

We consider the scalar wave equation, (2.1), in $\mathbb{R}^2$ or $\mathbb{R}^3$ with $u(\vec{x}, t)$ being the wave amplitude and for some medium parameters $a(\vec{x})$ and $b(\vec{x})$.

$$b\nabla \cdot (a\nabla u) = \frac{\partial^2 u}{\partial t^2}. \tag{2.1}$$

In order to solve this on a finite grid, it is helpful to split the second order partial differential equation (PDE) into a system of first order PDEs. To do so, we introduce an auxiliary field, $\vec{v}(\vec{x}, t)$, and let $\tilde{u} = u_t$ and $\vec{v}(\vec{x}, t) = a\nabla u$. Differentiating both

with respect to t, the wave equation can be written as the following system of first order equations, each of which satisfy the scalar wave equation.

$$\begin{aligned} \frac{\partial u}{\partial t} &= b\nabla \cdot \vec{v}, \\ \frac{\partial \vec{v}}{\partial t} &= a\nabla u. \end{aligned} \tag{2.2}$$

To see that $\tilde{u}$ satisfies the wave equation, we simply differentiate in time to find,

$$\begin{aligned} \tilde{u} &= u_t, \\ \tilde{u}_t &= u_{tt} = b\nabla \cdot (a\nabla u) = b\nabla \cdot \vec{v}, \\ \tilde{u}_{tt} &= b(\nabla \cdot \vec{v})_t = b\nabla \cdot (a\nabla \tilde{u}). \end{aligned}$$

The same can be shown for each component of $\vec{v} = a\nabla u$. Splitting the wave equation in this manner is also useful for implementing Perfectly Matched Layers, as will be discussed in detail later on.

In order to solve the system of PDE's (2.2) numerically, each partial derivative is approximated using a finite difference method. Focusing first on solving the one dimensional wave equation on a collocated grid of size N with Dirichlet boundary conditions, Taylor series expansion is used to derive a central difference to approximate each of the partial derivatives. For $i = 1, \cdots, N$ with grid points $x_i = i\Delta x$ we define the grid functions $u_i = u(x_i)$, $v_i = v(x_i)$. For the domain, $0 \leq x \leq N$ with boundary conditions $u_0 = 0$ and $u_N = 0$ we have have the following central difference approximation for the one dimensional wave equation,

$$\begin{aligned} \frac{\partial u_i}{\partial t} = \frac{dv}{dx} = \frac{b(v_{i+1} - v_{i-1})}{2\Delta x} + O(\Delta x^2), \\ \frac{\partial v_i}{\partial t} = \frac{du}{dx} = \frac{b(u_{i+1} - u_{i-1})}{2\Delta x} + O(\Delta x^2). \end{aligned} \tag{2.3}$$

The only boundary conditions necessary for the scalar wave equation are on u; however, because a collocated grid is being implemented, considerations have to be taken

to compute $\frac{\partial v}{\partial t}$ at the boundary as well. Computing $\frac{\partial v_N}{\partial t}$ using the centered difference method gives the following,

$$\frac{\partial v_N}{\partial t} = \frac{\partial u_N}{\partial x} = \frac{u_{N+1} - u_{N-1}}{2\Delta x} + O(\Delta x^2).$$

Because the grid only has N grid points, u_{N+1} does not exist. The same follows for computing $\frac{\partial v_0}{\partial t}$. To deal with this, right and left sided finite differences can be used to approximate $\frac{\partial v}{\partial t}$ at $i = 0$ and $i = N$.

$$\begin{aligned} \frac{\partial v_0}{\partial t} &= \frac{-3u_i + 4u_{i+1} - u_{i+2}}{2\Delta x} + O(\Delta x^2), \\ \frac{\partial v_N}{\partial t} &= \frac{3u_i - 4u_{i+1} + u_{i-2}}{2\Delta x} + O(\Delta x^2). \end{aligned}$$

Alternatively, we can use the PDE and the boundary condition to find that $u_{xx} = 0$ is a compatibility condition from which we may derive an expression for u_{N+1} as follows,

$$\begin{aligned} \frac{u_{N+1} - 2u_N + u_{N-1}}{\Delta x^2} &= 0 \\ \implies u_{N+1} &= 2u_N - u_{N-1}. \end{aligned}$$

To avoid these additional computations, a staggered grid can be used. Again, Taylor series expansion is used to derive the centered difference for the partial derivatives, but now, instead of taking Δx steps, we take $\Delta x/2$ steps. This allows u and v to lie on different grid points within the domain. Equations (2.4) are the finite differences on a staggered grid for the one dimensional wave equation.

$$\frac{\partial u_i}{\partial t} = \frac{b(v_{i+1/2} - v_{i-1/2})}{\Delta x} + O\left((\Delta x/2)^2\right), \text{ for } i = 1, \cdots, N-1 \text{ and } x_i = i\Delta x \tag{2.4a}$$

$$\frac{\partial v_i}{\partial t} = \frac{a(u_{i+1/2} - u_{i-1/2})}{\Delta x} + O\left((\Delta x/2)^2\right), \text{ for } i = 1/2, \cdots, N-1/2 \text{ and} \tag{2.4b}$$

$$x_i = i\Delta x.$$

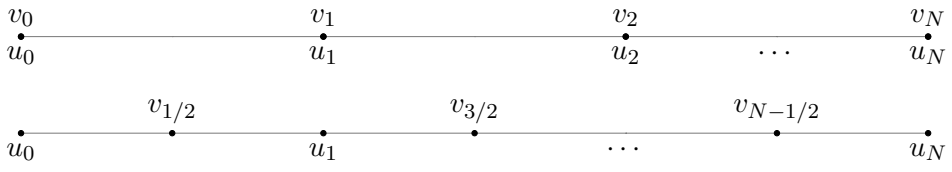


Figure 2.1: *Top:* Collocated grid with grid points $x = i\Delta x$. *Bottom:* Staggered grid with grid points $x = i\Delta x$

As you can see in Figure 2.1, all u lie on integer grid points and all v lie on half grid points in the staggered grid. This eliminates the additional computation for $\frac{\partial v}{\partial t}$ at $i = 0$ and $i = N$. Additionally, while both the collocated and staggered grids compute the derivatives to second order accuracy, on a staggered grid the error term is $O\left((\Delta x/2)^2\right)$, which is one quarter that of a collocated grid.

2.2 The 2D and 3D Wave Equation

Solving the wave equation on a discretized grid in higher dimensions becomes slightly more complicated than the one dimensional problem. Deriving the finite difference scheme that we will use for two dimensions will be done in detail. Once there is a thorough understanding of how the method works in 2D, moving to 3D is intuitive.

For the domain $0 \leq x \leq N$, $0 \leq y \leq N$ and the boundary conditions $u(0, 0, t) = 0$, $u(0, y, t) = 0$, $u(x, N, t) = 0$ and $u(N, y, t) = 0$, we have the following two dimensional wave equation,

$$\frac{\partial u}{\partial t} = b\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y}\right), \tag{2.5a}$$

$$\frac{\partial v_x}{\partial t} = a\frac{\partial u}{\partial x}, \tag{2.5b}$$

$$\frac{\partial v_y}{\partial t} = a\frac{\partial u}{\partial y}. \tag{2.5c}$$

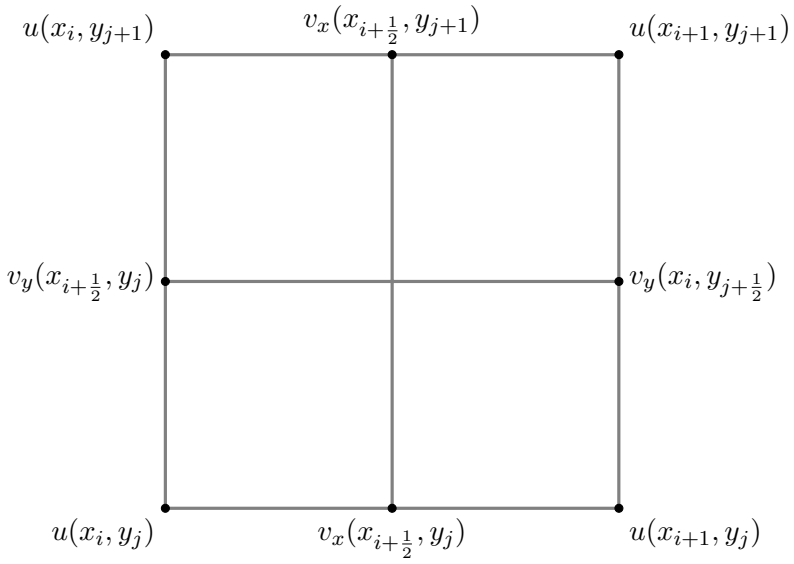


Figure 2.2: Two dimensional finite difference grid for $i, j = 0, 1 \cdots, N-1$.

As in the 1D problem, we will be working on a staggered grid, but now we must discretize in x and y. We want u and $\vec{v} = [v_x, v_y]^T$ on the same plane to keep the number of operations minimal. To do so, we must create a grid for u, a grid for v_x, and a grid for v_y. Again, we want u to lie on integer points and $\vec{v}$ to lie on half points, but we have to be a little more careful because $\vec{v}$ now has an v_x and v_y component in the x and y, respectively. We will denoted the $\vec{v}$ grid with x on half points and y on whole points as v_x, and the $\vec{v}$ grid with x on whole points and y on half points as v_y as in Figure 2.2. We will denote x grid points with i and y grid points with j for $i, j = 0, 1, \cdots, N-1$. Using the Taylor Series expansions for u and $\vec{v}$, we will derive a centered finite difference approximation for each derivative. Starting with (2.5b) we expand u in a Taylor series around $x_{i\pm\frac{1}{2}}$ with grid points $x_i = i\Delta x$ and $y_j = j\Delta y$. For simplicity, we let $a = 1$,

$$u\left(x_{i+\frac{1}{2}} + \frac{\Delta x}{2},\ y_j\right) = u(x_{i+\frac{1}{2}}, y_j) + u'\frac{\Delta x}{2} + \frac{u''(\Delta x/2)^2}{2!} + \frac{u'''(\Delta x/2)^3}{3!} + \cdots,$$
$$u\left(x_{i+\frac{1}{2}} - \frac{\Delta x}{2},\ y_j\right) = u(x_{i+\frac{1}{2}}, y_j) - u'\frac{\Delta x}{2} + \frac{u''(\Delta/2x)^2}{2!} - \frac{u'''(\Delta x/2)^3}{3!} + \cdots.$$

Subtracting the two expressions and solving for the first derivative we get the desired approximation

$$\frac{\partial u}{\partial x}(x_{i+\frac{1}{2}}, y_j) = \frac{u(x_{i+\frac{1}{2}} + \Delta x/2, y_j) - u(x_{i+\frac{1}{2}} - \Delta x/2, y_j)}{\Delta x} + O\left((\Delta x/2)^2\right).$$

This can be written as,

$$\frac{\partial v_x}{\partial t} = \frac{\partial u}{\partial x}(x_{i+\frac{1}{2}}, y_j) = \frac{u_{i+1,j} - u_{i,j}}{\Delta x} + O\left((\Delta x/2)^2\right). \tag{2.6}$$

Following the same process for (2.5c), this time taking steps in y on the grid points $x = i\Delta x$, $y = j\Delta y$, we have the following finite difference approximation

$$\frac{\partial v_y}{\partial t} = \frac{\partial u}{\partial y}(x_i, y_{j+\frac{1}{2}}) = \frac{u_{i,j+1} - u_{i,j}}{\Delta y} + \left(O(\Delta y/2)^2\right). \tag{2.7}$$

We are left with finding the finite difference scheme for $\frac{\partial u}{\partial t} = b\frac{\partial v_x}{\partial x} + b\frac{\partial v_y}{\partial y}$. To do this, we need the finite difference for each derivative on the right hand side. This is done in the same manner as the previous derivatives; however, because there are boundary conditions set on $\partial u/\partial t$, the first value we need to compute is at (x_{i+1}, y_{j+1}). The finite difference for each derivative, respectively, is,

$$\frac{\partial v_x}{\partial x}(x_{i+1}, y_{j+1}) = \frac{v_{i+\frac{3}{2},j+1} - v_{i+\frac{1}{2},j+1}}{\Delta x} + O\left((\Delta x/2)^2\right),$$
$$\frac{\partial v_y}{\partial y}(x_{i+1}, y_{j+1}) = \frac{v_{i+1,j+\frac{3}{2}} - v_{i+1,j+\frac{1}{2}}}{\Delta y} + O\left((\Delta y/2)^2\right).$$

Using Figure 2.2 as a guide, we see that for each derivative we take a half step forward and a half step backward in the direction of the derivative. We now have a finite difference approximation method, with second order accuracy for the two dimensional wave equation.

As mentioned before, solving the 3D wave equation on a discrtized grid is intuitive once the 2D dimensional grid is well understood. In three dimensions $\vec{v} = [v_x, v_y, v_z]^T$

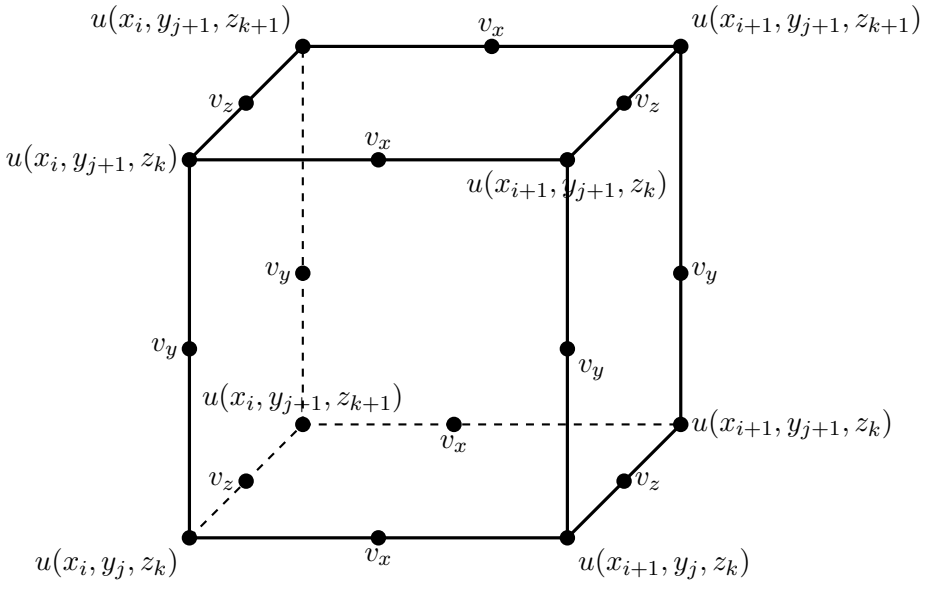


Figure 2.3: 3D Finite Difference grid with $i, j, k = 0, 1, \cdots, N-1$.

and so we we need to introduce a v_z grid in addition to v_x and v_y. The v_z grid has x and y on integer points and z on half points. Note, in the v_x and v_y grids, a z component, which lies on integer points, must be added, see Figure 2.2. We will denote the z coordinate with k. Using the same methodology as in one and two dimensions, we derive the following finite difference approximations for $i, j, k = 0, \cdots, N-1$.

$$
\begin{aligned}
&\frac{\partial v_x}{\partial t}(x_{i+\frac{1}{2}}, y_j, z_k) = \frac{\partial u_{i+\frac{1}{2},j,k}}{\partial x} = \frac{u_{i+1,j,k} - u_{i,j,k}}{\Delta x} + O(\Delta x/2)^2,\\
&\frac{\partial v_y}{\partial t}(x_i, y_{j+\frac{1}{2}}, z_k) = \frac{\partial u_{i,j+\frac{1}{2},k}}{\partial y} = \frac{u_{i,j+1,k} - u_{i,j,k}}{\Delta y} + O(\Delta y/2)^2,\\
&\frac{\partial v_z}{\partial t}(x_i, y_j, z_{k+\frac{1}{2}}) = \frac{\partial u_{i,j,k+\frac{1}{2}}}{\partial z} = \frac{u_{i,j,k+1} - u_{i,j,k}}{\Delta z} + O(\Delta z/2)^2,\\
&\frac{\partial v_x}{\partial x}(x_{i+1}, y_{j+1}, z_{k+1}) = \frac{v_{i+\frac{3}{2},j+1,k+1} - v_{i+\frac{1}{2},j+1,k+1}}{\Delta x} + O(\Delta x/2)^2,\\
&\frac{\partial v_y}{\partial y}(x_{i+1}, y_{i+1}, z_{k+1}) = \frac{v_{i+1,j+\frac{3}{2},k+1} - v_{i+1,j+\frac{1}{2},k+1}}{\Delta y} + O(\Delta y/2)^2,\\
&\frac{\partial v_z}{\partial z}(x_{i+1}, y_{j+1}, z_{k+1}) = \frac{v_{i+1,j+1,k+\frac{3}{2}} - v_{i+1,j+1,k+\frac{1}{2}}}{\Delta z} + O(\Delta z/2)^2,\\
&\text{Where } \frac{\partial u}{\partial t} = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}.
\end{aligned}
\tag{2.8}
$$

2.3 Perfectly Matched Layers

When solving PDE's numerically, an infinite domain must be abridged. In doing so, reflection at the boundaries is introduced. For the wave equation, this is of particular importance; unlike some other problems where solutions decay quickly or are periodic, the oscillating nature of waves along with slow decay with increasing distance, means that using Dirichlet or Neumann boundary conditions will cause inadmissible reflections [4]. A Perfectly Matched Layer (PML) is one way to deal with this for homogeneous medium. The idea behind PML is to create an absorbing layer, sufficiently far from the region of interest, that turns propagating waves into exponentially decaying waves and thus, eliminates reflection at the boundaries. This is done by analytic continuation into complex coordinates, followed by a transformation back to real coordinates, and then truncation of the domain.

2.3.1 Perfectly Matched Layers in 1D

Consider the plane wave in 1D,

$$\vec{w}(x,t) = \sum_{k,\omega} W e^{(ikx-\omega t)} \text{ , } \omega/k \text{ phase velocity,} \tag{2.9}$$

which can be decomposed into expressions of the form

$$W e^{i(kx-\omega t)}, \tag{2.10}$$

for some amplitude W. Equation (2.10) is analytic which allows us to consider it as a function of a complex variable, φ. Instead of evaluating the function on a strictly real domain, a strictly increasing imaginary component is added adjacent to the region of interest. This changes the coordinates from real to complex as shown in Figure 2.4. Let $\varphi = x + if(x)$, $f(x)$ real, be the new complex coordinates. Then the solution in (2.10) becomes,

$$W e^{i(k\varphi-\omega t)} = W e^{i(k(x+if(x))-i\omega t)} = W e^{-kf(x)} e^{i(kx-\omega t)}.$$

For $f(x)$ increasing with x, we now have exponential decay (note that k is assumed greater than zero since we chose x in the positive direction). Thus, within the region of interest the solution has not changed, while outside the region there is now exponential decay, see Figure 2.5. For the negative x direction, $\varphi = -x - if(x)$, k is assumed less than zero, and again the result is exponential decay outside the region of interest as seen in (2.11).

$$We^{i(k(-x-if(x))-\omega t)} = We^{-ikx+kf(x)-i\omega t} = We^{ikx-i\omega t-kf(x)} = We^{-kf(x)}e^{i(kx-\omega t)} \quad (2.11)$$

Using $\varphi = x + if(x)$ as the new coordinate, we can rewrite the system of equations as,

$$\frac{\partial u}{\partial t} = b\frac{\partial v}{\partial \varphi} \quad (2.12a)$$

$$\frac{\partial v}{\partial t} = a\frac{\partial u}{\partial \varphi} \quad (2.12b)$$

We would like to work in the real domain and so we make a coordinate transformation back to real coordinates. If $\varphi = x + if(x)$ is the complex coordinate, then $\frac{\partial \varphi}{\partial x} = 1 + i\frac{\partial f}{\partial x}$, and with some simple rearranging we have,

$$d\varphi = \left(1 + i\frac{\partial f}{\partial x}\right) dx. \quad (2.13)$$

We will denote $\frac{\partial f(x)}{\partial x}$ by $\frac{\sigma_x}{\omega}$, for $\sigma_x > 0$. The solution in Eq(2.11) becomes

$$We^{-k\frac{\sigma_x}{\omega}}e^{i(kx-\omega t)}.$$

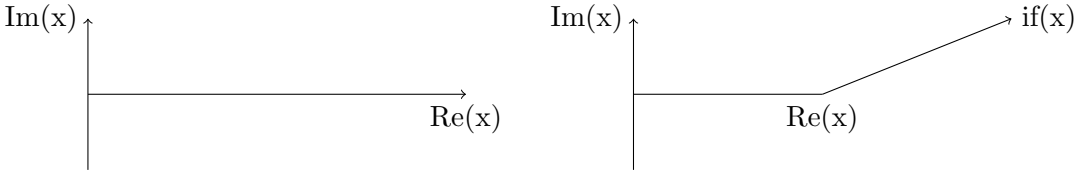


Figure 2.4: *Left*: x evaluated strictly on the real axis after analytic continuation in to complex plane. *Right*: x evaluated along the real axis in the complex plane until a certain point, then x is evaluated on deformed contour with imaginary component, this is the absorbing region.

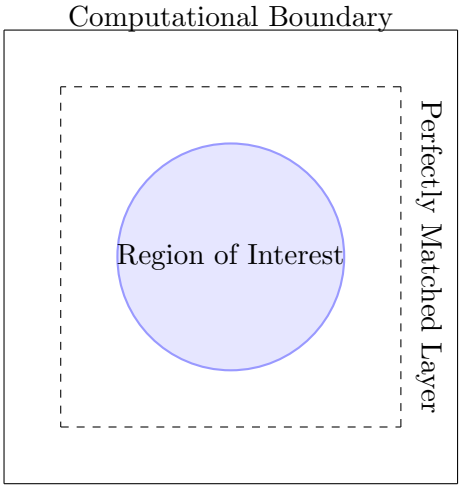


Figure 2.5: Representation of Perfectly Matched Layers implemented in a finite domain. The perfectly matched layer is placed adjacent to the boundary and as the wave $\vec{u}$ propagates from the region of interest, the perfectly matched layer absorbs it with out reflection. The solution in the region of interest remains unchanged.

By introducing ω, we have k/ω in the solution which is the inverse of the phase velocity. In homogeneous medium, this term is constant, meaning it is independent of frequency so all wavelengths will decay at the same rate [4]. Equation (2.13) can now be written as

$$d\varphi = \left(1 + i\frac{\sigma_x}{\omega}\right). \tag{2.14}$$

After the analytic continuation to complex coordinates, anywhere $d\varphi$ is in (2.12), we replace it by the right hand side of (2.14). The PML transformation for one dimension can be summarized by (2.15) [4].

$$\frac{\partial}{\partial x} \rightarrow \frac{1}{1 + i\frac{\sigma_x}{\omega}} \frac{\partial}{\partial x}. \tag{2.15}$$

The transformation (2.15) is frequency dependent, hence to be in the time domain, the $1/\omega$ term from the transformation (2.15) must cancel. From the solution to the wave equation (2.10), and the system of PDE's, (2.2), we have the following

system in 1D:

$$\frac{\partial u}{\partial t} = b\frac{\partial v}{\partial x} = -i\omega e^{ikx-i\omega t} = -i\omega u, \tag{2.16a}$$
$$\frac{\partial v}{\partial t} = a\frac{\partial u}{\partial x} = -i\omega v. \tag{2.16b}$$

Applying the PML transformation (2.15) to (2.16a) and multiplying both sides by $1 + i\frac{\sigma_x}{\omega}$ we get the following,

$$\begin{aligned}
\left(\frac{b}{1+i\frac{\sigma_x}{\omega}}\right)\frac{\partial v}{\partial x} &= -i\omega u, \\
\implies b\frac{\partial v}{\partial x} &= -i\omega u + \sigma_x u, \\
\implies -i\omega u &= b\frac{\partial v}{\partial x} - \sigma_x u, \\
\implies \frac{\partial u}{\partial t} &= b\frac{\partial v}{\partial x} - \sigma_x u.
\end{aligned}$$

Following the same steps for (2.16b), we get a similar equation

$$a\frac{\partial u}{\partial x} = -i\omega v + \sigma_x v.$$

We succeeded in canceling the undesired term, and our equations can now be written without any frequency, putting us in the time domain. Our new system of equations, for the one dimensional wave with PML implemented, for$\sigma_x \geq 0$, is,

$$\begin{aligned}
\frac{\partial u}{\partial t} &= b\frac{\partial v}{\partial x} - \sigma_x u, \\
\frac{\partial v}{\partial t} &= b\frac{\partial u}{\partial x} - \sigma_x v.
\end{aligned} \tag{2.17}$$

. Perhaps now it is easier to see that when $\sigma_x = 0$, the equations are the same and so the solution must also be the same. Therefore, we can solve the wave equation in the area of interest without changing the solution while adding an absorbing layer that eliminates reflection at the boundaries.

Finally, we can truncate the domain at some large x. Once the wave enters the PML, it decays exponentially and only very small values reach the boundary. There,

despite some reflection, they dissipate on their way back and the effect on the solution is minimal [4].

2.3.2 Perfectly Matched Layers in 2D and 3D

For one dimension, the frequency term cancels in the transformation and the equations can be easily converted back to the time domain; for higher dimensions we must introduce auxiliary fields to eliminate the frequency terms. This section will go through the 2D implementation of PML in detail and then briefly discuss the 3D implementation as the derivation for both dimensions are done with the same methods.

The transformation (2.15) remains the same for higher dimensions; however, now we need to apply the two transformations (2.18) with σ_x and σ_y corresponding to each respective axis.

$$\frac{\partial}{\partial x} \rightarrow \frac{1}{1 + i\frac{\sigma_x}{\omega}} \frac{\partial}{\partial x}, \quad \frac{\partial}{\partial y} \rightarrow \frac{1}{1 + i\frac{\sigma_y}{\omega}} \frac{\partial}{\partial y}. \tag{2.18}$$

Recall the 2D wave equation,

$$\begin{aligned}
\frac{\partial u}{\partial t} &= b\frac{\partial v_x}{\partial x} + b\frac{\partial v_y}{\partial y}, && \text{(2.19a)}\\
\frac{\partial v_x}{\partial t} &= a\frac{\partial u}{\partial x}, && \text{(2.19b)}\\
\frac{\partial v_y}{\partial t} &= a\frac{\partial u}{\partial y}, && \text{(2.19c)}\\
u(x,0,t) &= 0, u(x,N,t) = 0, u(0,y,t) = 0, u(N,y,t) = 0.
\end{aligned}$$

Applying the PML transformations (2.18) to (2.19) yields the following,

$$\frac{\partial v_x}{\partial x}\left(\frac{1}{1+i\frac{\sigma_x}{\omega}}\right)+\frac{\partial v_y}{\partial y}\left(\frac{1}{1+i\frac{\sigma_y}{\omega}}\right)=-i\omega u, \tag{2.20a}$$

$$\frac{\partial u}{\partial x}\left(\frac{1}{i+i\frac{\sigma_x}{\omega}}\right)=-i\omega v_x, \tag{2.20b}$$

$$\frac{\partial u}{\partial y}\left(\frac{1}{1+i\frac{\sigma_y}{\omega}}\right)=-i\omega v_y. \tag{2.20c}$$

$$\tag{2.20d}$$

Equations(2.20b) and (2.20c) should look familiar as they are the same as we saw in one dimension for $\partial v_x/\partial t = \partial u/\partial x$. Multiplying through and solving for $i\omega v$ as we did in one dimensions, we get the following equations with PML in x and y on the respective $\vec{v}$ grids,

$$\frac{\partial v_x}{\partial t}=\frac{\partial u}{\partial x}-\sigma_x v_x, \tag{2.21}$$

$$\frac{\partial v_y}{\partial t}=\frac{\partial u}{\partial y}-\sigma_y v_y. \tag{2.22}$$

Next, we multiply (2.20a) by both $1+i\frac{\sigma_x}{\omega}$ and $1+i\frac{\sigma_y}{\omega}$

$$\left(1+i\frac{\sigma_y}{\omega}\right)\frac{\partial v_x}{\partial x}+\left(1+i\frac{\sigma_x}{\omega}\right)\frac{\partial v_y}{\partial y}=-i\omega u\left(1+i\frac{\sigma_y}{\omega}\right)\left(1+i\frac{\sigma_x}{\omega}\right). \tag{2.23}$$

Upon expanding each term in (2.23), we find that the frequency terms ω do not cancel as easily as they did in one dimension. Starting with the left hand side of (2.23), we have the following,

$$\frac{\partial v_x}{\partial x}+\frac{\partial v_y}{\partial y}+i\frac{\sigma_x}{\omega}\frac{\partial v_x}{\partial x}+i\frac{\sigma_y}{\omega}\frac{\partial v_y}{\partial y}. \tag{2.24}$$

We introduce an auxiliary field $\psi(x,y,t)$ for each term with a remaining ω

$$\psi_{yx}=i\frac{\sigma_y}{\omega}\frac{\partial v_x}{\partial x},\quad \psi_{xy}=i\frac{\sigma_x}{\omega}\frac{\partial v_y}{\partial y}. \tag{2.25}$$

Let's consider one of these terms, $\psi_{xy}=i\frac{\sigma_x}{\omega}\frac{\partial v_y}{\partial y}$. This is equivalent to $-i\omega\psi_{xy}=\sigma_x\frac{\partial v_y}{\partial y}$. Using the Fourier Transform for some function $x(t)$, we have the following equivalence,

$$x(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty}\hat{x}(\omega)e^{i\omega t}d\omega,$$

where $\hat{x}$ is the Fourier Transform of $x(t)$. Taking the derivative of $x(t)$ in this form, we introduce an $i\omega$ term.

$$\begin{aligned}\frac{dx(t)}{dt} &= \frac{d}{dt}\left(\frac{1}{2\pi}\int_{-\infty}^{\infty}\hat{x}(\omega)e^{i\omega t}d\omega\right),\\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty}i\omega\hat{x}(\omega)e^{i\omega}d\omega,\\ &= i\omega x(t).\end{aligned}$$

Therefore, $-i\omega\psi_{xy} = \sigma_x\frac{\partial v_y}{\partial y}$ gives us the new partial differential equation

$$\frac{\partial\psi_{xy}}{\partial t} = \sigma_x\frac{\partial v_y}{\partial y}. \tag{2.26}$$

Thus, we have introduced the two new PDE's, (2.28), in to the 2D system (2.19).

$$\frac{\partial\psi_{xy}}{\partial t} = \sigma_x\frac{\partial v_y}{\partial y}, \tag{2.27}$$

$$\frac{\partial\psi_{yx}}{\partial t} = \sigma_y\frac{\partial v_x}{\partial x}. \tag{2.28}$$

Now consider the right hand side of (2.20a),

$$-i\omega u + \sigma_x u + \sigma_y u + i\frac{\sigma_x\sigma_y}{\omega}u. \tag{2.29}$$

Again, we introduce an auxiliary field $\delta(x,y,t) = i\frac{\sigma_x\sigma_y}{\omega}u$ to deal with the remaining ω term. Using the same process as we did with ψ we introduce the last PDE in to the 2D system.

$$-i\omega\delta = \sigma_x\sigma_y u \implies \frac{\partial\delta}{\partial t} = \sigma_x\sigma_y u. \tag{2.30}$$

The frequency terms have been successfully eliminated and we are in the time domain. We now have the system for the 2D wave equation with perfectly matched layers in

x and y for a bounded domain.

$$\frac{\partial u}{\partial t} = b\frac{\partial v_x}{\partial x} + b\frac{\partial v_y}{\partial y} + \psi_{yx} + \psi_{xy} - \sigma_x u - \sigma_y u - \delta, \tag{2.31a}$$

$$\frac{\partial v_x}{\partial t} = a\frac{\partial u}{\partial x} - \sigma_x v_x, \tag{2.31b}$$

$$\frac{\partial v_y}{\partial t} = a\frac{\partial u}{\partial y} - \sigma_y v_y, \tag{2.31c}$$

$$\frac{\partial \psi_{xy}}{\partial t} = \sigma_x \frac{\partial v_y}{\partial y}, \tag{2.31d}$$

$$\frac{\partial \psi_{yx}}{\partial t} = \sigma_y \frac{\partial v_x}{\partial x}, \tag{2.31e}$$

$$\frac{\partial \delta}{\partial t} = \sigma_x \sigma_y u. \tag{2.31f}$$

Notice, (2.31d) and (2.31e) are only nonzero when σ_x and σ_y are non-zero, respectively. Also, (2.31f) is only nonzero when both σ_x and σ_y are nonzero, occurring in the corners of a two dimensional grid. Additionally, when σ_x and σ_y are both zero, we have the original two dimensional wave equation as seen in Figure 2.6.

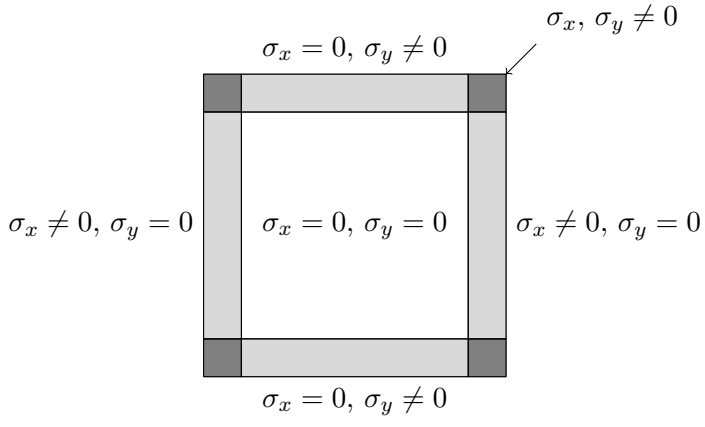


Figure 2.6: Perfectly Matched Layer in 2D with σ representation

Next, recall the system of equations for the 3D wave equation,

$$\begin{aligned}
\frac{\partial u}{\partial t} &= b\frac{\partial v_x}{\partial x} + b\frac{\partial v_y}{\partial y} + b\frac{\partial v_z}{\partial x}, && \text{(2.32a)}\\
\frac{\partial v_x}{\partial t} &= a\frac{\partial u}{\partial x}, && \text{(2.32b)}\\
\frac{\partial v_y}{\partial t} &= a\frac{\partial u}{\partial y}, && \text{(2.32c)}\\
\frac{\partial v_z}{\partial t} &= a\frac{\partial u}{\partial z}. && \text{(2.32d)}
\end{aligned}$$

As we did in two dimensions, we apply the PML transformations for σ_x, σ_y and σ_z. Applying the transformation to (2.32b), (2.32c) and (2.32d) yeilds three new PDE's in the same format as in one and two dimensions.

$$\begin{aligned}
\frac{\partial v_x}{\partial t} &= \frac{\partial u}{\partial x} - \sigma_x v_x,\\
\frac{\partial v_y}{\partial t} &= \frac{\partial u}{\partial y} - \sigma_y v_y,\\
\frac{\partial v_z}{\partial t} &= \frac{\partial u}{\partial y} - \sigma_z v_z.
\end{aligned} \qquad (2.33)$$

Applying the PML transformations to (2.32a), we again need to introduce auxiliary fields to eliminate any unwanted frequency terms. We apply the transformation as usual,

$$\frac{\partial v_x}{\partial x}\left(\frac{1}{1+i\frac{\sigma_x}{\omega}}\right) + \frac{\partial v_y}{\partial y}\left(\frac{1}{1+i\frac{\sigma_y}{\omega}}\right) + \frac{\partial v_z}{\partial z}\left(\frac{1}{1+i\frac{\sigma_z}{\omega}}\right) = -i\omega u. \qquad (2.34)$$

Starting again with the left hand side and multiplying through by each $1+i\frac{\sigma}{\omega}$ term, we now have six terms with a remaining ω and three terms with a remaining ω^2. Again we use the auxiliary field $\psi(x,y,z,t)$ for the six terms with ω, and introduce the auxiliary field $\phi(x,y,z,t)$ for the terms with ω^2.

$$\psi_{zx} = i\frac{\sigma_z}{\omega}\frac{\partial v_x}{\partial x},\ \psi_{xy} = i\frac{\sigma_x}{\omega}\frac{\partial v_y}{\partial y},\ \psi_{zy} = i\frac{\sigma_z}{\omega}\frac{\partial v_y}{\partial y}\ \cdots$$
$$\phi_{yzx} = -\frac{\sigma_y\sigma_z}{\omega^2}\frac{\partial v_x}{\partial x},\ \phi_{xzy} = -\frac{\sigma_x\sigma_z}{\omega^2}\frac{\partial v_y}{\partial y},\ \phi_{yxz} = -\frac{\sigma_y\sigma_x}{\omega^2}\frac{\partial v_z}{\partial z}.$$

Each ϕ can be factored such that it contains ψ. For example, lets consider ϕ_{yzx}. We can factor it as follows,

$$\begin{aligned}\phi_{yzx} = -\frac{\sigma_y\sigma_z}{\omega^2}\frac{\partial v_x}{\partial x} &= i\frac{\sigma_y}{\omega}\left(i\frac{\sigma_z}{\omega}\frac{\partial v_x}{\partial x}\right) \\ &= i\frac{\sigma_y}{\omega}(\psi_{zx})\end{aligned}$$

We can then define a new PDE for each ϕ,

$$\begin{aligned}\frac{\partial \phi_{yzx}}{\partial t} &= \sigma_y\psi_{zx}, \\ \frac{\partial \phi_{xzy}}{\partial t} &= \sigma_z\psi_{xy}, \\ \frac{\partial \phi_{yxz}}{\partial t} &= \sigma_x\psi_{yz}.\end{aligned} \tag{2.35}$$

Next, we expand the right hand side of (2.23).

$$\begin{aligned}-i\omega u\left(1+i\frac{\sigma_y}{\omega}\right)\left(1+i\frac{\sigma_z}{\omega}\right)\left(1+i\frac{\sigma_x}{\omega}\right) = \\ -i\omega u + \sigma_z u + \sigma_x u + \sigma_y u + i\frac{\sigma_x\sigma_z u}{\omega} + i\frac{\sigma_y\sigma_z u}{\omega} - \frac{\sigma_x\sigma_y\sigma_z u}{\omega^2}.\end{aligned}$$

We again need to introduce an auxiliary field, $\rho(x,y,z,t)$ for the two terms including ω, and the final auxiliary field, $\delta(x,y,z,t) = \frac{\sigma_x\sigma_y\sigma_z u}{\omega^2}$. As we did in the left hand side, we can write δ in terms of ρ,

$$\begin{aligned}\rho_{xz} &= i\frac{\sigma_x\sigma_z u}{\omega}, \\ \delta &= \frac{\sigma_x\sigma_y\sigma_z u}{\omega^2}, \\ \Rightarrow \delta &= i\rho_{xz}\frac{\sigma_y}{\omega},\end{aligned}$$

with the respective derivatives,

$$\begin{aligned}\frac{\partial \rho_{xz}}{\partial t} &= \sigma_x\sigma_z u, \ \frac{\partial \rho_{yz}}{\partial t} = \sigma_y\sigma_z u, \\ \frac{\partial \delta}{\partial t} &= \sigma_y\rho.\end{aligned}$$

$\partial u/\partial t$ can now be written with PML as follows,

$$\frac{\partial u}{\partial t} = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} - \sigma_x u - \sigma_y u - \sigma_z u + \frac{\partial v_z}{\partial z} + \psi_{xz} + \cdots + \psi_{xz} + \phi_{yzx} + \phi_{xzy} + \phi_{yxz} - \rho_{xz} - \rho_{xy} - \delta.$$

Of course, there are constraints when implementing PML numerically. Solving PDE's on a discretized grid using the finite difference method only approximates the solutions instead of solving them exactly. The PML still attenuates the waves that enter, but because the solutions are not exact there is some reflection at the PML boundary. Despite this, if the grid is discretized appropriately for the particular PDE, these reflections will be small. Additionally, using a quadratic or cubic PML can reduce the reflection at the PML boundary for a small perfectly matched layer; less than or equal to half a wave length [4].

Chapter 3

Parameter Estimation

This chapter uses the methods discussed in Chapter 1 to develop a finite-difference method of lines discretization in which time stepping is done by a Runge-Kutta method. This is followed by a discussion on optimization methods for parameter estimation.

3.1 Time Domain Simulation

Code to simulate the wave equation is written for one, two and three dimensions; however, the discussion will be done for the two dimensional wave equation with PML (3.1) as the methods easily translate to three dimensions, and the computations are

significantly faster.

$$\begin{aligned}
&\frac{\partial u}{\partial t} = b\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y}\right) + \psi_1 + \psi_2 - \sigma_x u - \sigma_y u - \rho,\\
&\frac{\partial v_x}{\partial t} = a\frac{\partial u}{\partial x} - \sigma_x u,\\
&\frac{\partial v_y}{\partial t} = a\frac{\partial u}{\partial y} - \sigma_y u,\\
&\frac{\partial \psi_1}{\partial t} = \sigma_y \frac{\partial v_x}{\partial x},\\
&\frac{\partial \psi_2}{\partial t} = \sigma_x \frac{\partial v_y}{\partial y},\\
&\frac{\partial \rho}{\partial t} = \sigma_x \sigma_y u.\\
&0 \le x \le \ell_x,\ 0 \le y \le \ell_y.\\
&u(x,0,t) = 0, u(x,\ell_y,t) = 0, u(0,y,t) = 0, u(\ell_x,y,t) = 0.
\end{aligned} \tag{3.1}$$

The initial condition for u is arbitrarily chosen as

$$\exp\left(\frac{-\sqrt{(x-\frac{\ell_x}{2})^2 + (y-\frac{\ell_y}{2})^2}}{4}\right), \tag{3.2}$$

where ℓ_x and ℓ_y are the lengths of the domain in the directions x and y respectively. Initial conditions are also needed on $\vec{v}$, as well as all additional PDE's that are added to the system of equations from the PML; these initial conditions are set to zero. We use the following equations for the PML in x and y,

$$\sigma_x = \begin{cases} x - 95 & \text{if } x \ge 95 \\ -(x-5) & \text{if } x \le 5 \end{cases} \qquad \sigma_y = \begin{cases} y - 95 & \text{if } y \ge 95 \\ -(y-95) & \text{if } y \le 5 \end{cases} \tag{3.3}$$

Using the staggered grid finite difference scheme discussed in Chapter 1, $\frac{\partial u}{\partial x}$, $\frac{\partial u}{\partial y}$, $\frac{\partial v_x}{\partial x}$ and $\frac{\partial v_y}{\partial y}$ are approximated. To verify that the finite differences are approximating the derivative up to second order accuracy, each are written as their own function in MATLAB, and then checked using a function that we can differentiate exactly. For example, suppose we wanted to check the computation for $\partial u/\partial x$. Letting $u = 2x$,

we verify that the code is giving the correct approximation.

Discretization in space makes each derivative on the right hand side of (3.1) an ordinary differential equation (ODE) which we evolve in time using the classic Runge-Kutta method (3.4). Using $dt = 0.1$, Figure 3.1 shows the solution to the wave equation with the prescribed boundary and initial conditions, at three points in time during the simulation.

$$\begin{aligned} t_n &= t_0 \\ k_1 &= f(t_n, y_n) \\ k_2 &= f\left(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2} k_1\right) \\ k_3 &= f\left(t_n + \frac{\Delta t}{2}, y_n + \frac{\Delta t}{2} k_2\right) \\ k_4 &= hf(t_n + \Delta t, y_n + \Delta t k_3) \\ y_{n+1} &= y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ t_{n+1} &= t_n + \Delta t \end{aligned} \tag{3.4}$$

for $n = 1, \cdots, N$

Finally, we focus on increasing the speed of the computations. The easiest way to do this is to eliminate the number of operations that need to be done. We choose to modify the grid by increasing Δx and Δy sufficiently, such that the simulation is still numerically stable. Additionally, the Runge-Kutta algorithm is changed from fourth order to second order, effectively eliminating half the operations needed to solve the differential equations. Although the fourth order Runge-Kutta is more accurate than the second order, at this point we are more concerned with proper implementation than we are with efficiency. Although we did not choose to do so, one can store the additional PDE's derived in the implementation of the PML only where they are not zero; σ_x, σ_y and $\sigma_z \neq 0$.

Remark: The second order Runge-Kutta method is not stable for hyperbolic equations such as the wave equation. Despite this, for the time duration in which the

simulations for this thesis took place, the solution is stable. Instability only occurred after a time period that was much longer than what we consider here. In future work, the classical Runge-Kutta method should be used.

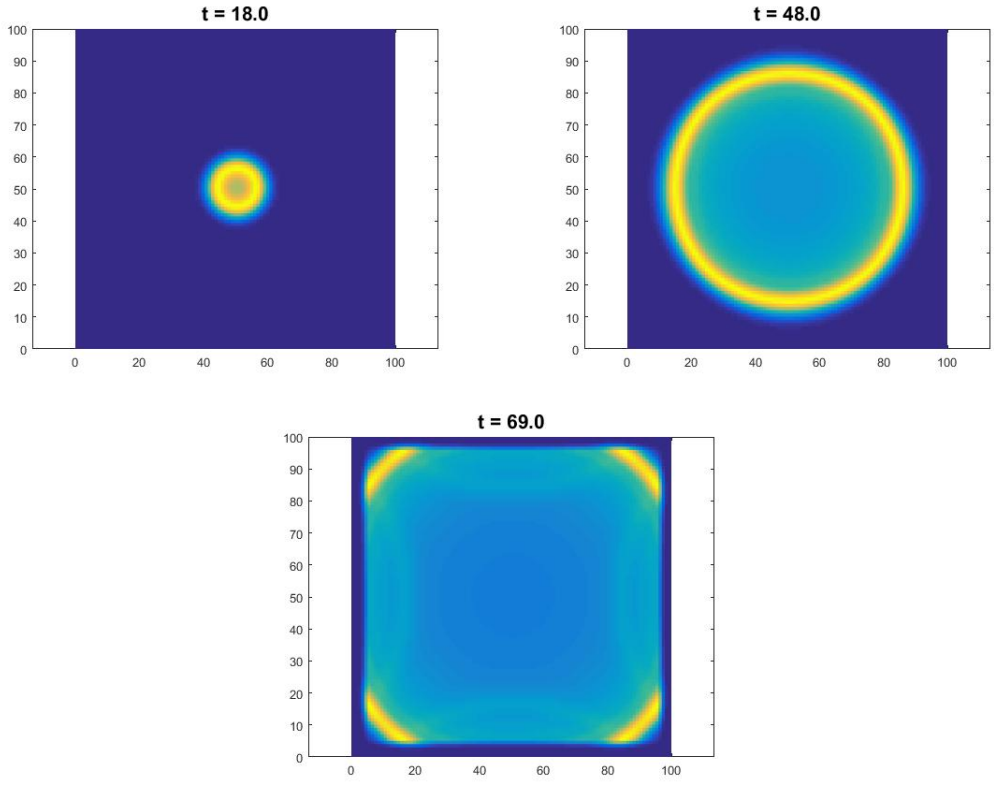


Figure 3.1: Simulation of the wave equation in two dimensions using Finite-Difference Time-Domain with Perfectly Matched Layers on a 100x100 grid with Δx and Δy set to 1/2. Although some reflection is observed in the third frame, it is small compared to the amplitude of the wave, and the effect on the solution is minimal.

3.2 The Forcing Function

$$f(x,y,t) = \gamma e^{\frac{-(t-t_0)^2}{\beta^2}} e^{-(\frac{(x-x_0/2)^2}{\alpha^2}+\frac{(y-y_0/2)^2}{\alpha^2})}. \tag{3.5}$$

We turn the homogeneous wave equation into a non-homogeneous one by adding a

forcing function $f(\vec{x}, \vec{y}, t)$ at some time t_0. Using the time-dependent Gaussian (3.5) as the forcing function f with parameters $\vec{\theta} = (\alpha, \gamma)$ we write the wave equation (2.2) as,

$$\frac{\partial^2 u}{\partial t^2} = b\nabla \cdot (a\nabla u) + f. \tag{3.6}$$

Because we split the wave equation in order to approximate it with finite differences, we need to carefully construct the new system with the forcing function. For some function g, we start with the following system,

$$\frac{\partial u}{\partial t} = b\nabla \cdot \vec{v} + g, \tag{3.7a}$$
$$\frac{\partial \vec{v}}{\partial t} = a\nabla u, \tag{3.7b}$$

and derive the new system,

$$\frac{\partial u}{\partial t} = b\nabla \cdot \vec{v} + g,$$
$$\frac{\partial \vec{v}}{\partial t} = a\nabla u,$$
$$\frac{\partial g}{\partial t} = f(\vec{x}, \vec{y}, t),$$

Recall that it was shown in Chapter 1, that (3.7a) and (3.7b) satisfy the wave equation. This must still hold true; however, now it should satisfy the inhomogenous wave equation (3.6). We take the derivative of (3.7a) with respect to t.

$$\frac{\partial^2 u}{\partial t^2} = b\nabla \cdot \frac{\partial \vec{v}}{\partial t} + \frac{\partial g}{\partial t}, \tag{3.9a}$$
$$= b\nabla \cdot (a\nabla u) + \frac{\partial g}{\partial t}. \tag{3.9b}$$

Equation (3.9b) needs to satisfy (3.6), thus $\frac{\partial g}{\partial t}$ must be f, and so we have added a new PDE, (3.10) to the wave equation.

$$\frac{\partial g}{\partial t} = f. \tag{3.10}$$

Once the initial conditions are changed to zero and the forcing function has been included, we want to estimate the parameters of f using measurements of the solution.

3.3 Cost Function and Optimization

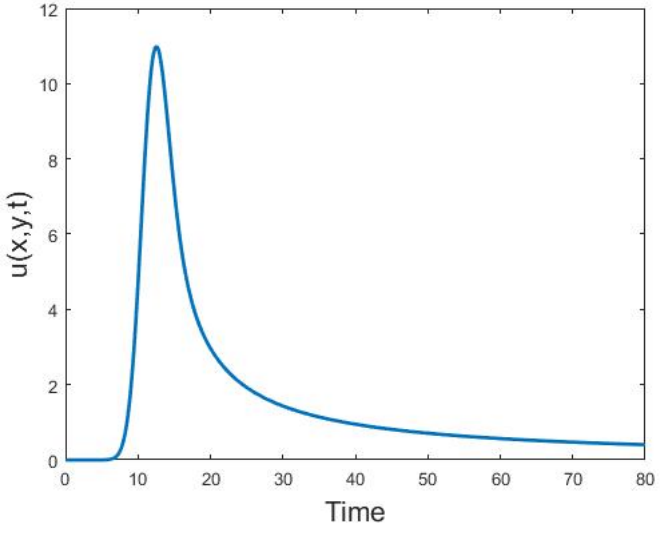
In a real experiment, measurements are taken at a finite number of sensor locations within a given area. In our case we must generate these measurements from simulating the wave equation with chosen parameters, $\vec{\theta}$, in the forcing function. We will denote the vector of synthetic experimental measurements with $\vec{c}_0$. In order to accurately model the result of the experiment, we need to estimate the parameters of the forcing function as best as possible.

The time dependent Gaussian function (3.5) with $\beta = 2$ and $(x_0, y_0) = (50, 50)$ is used as the forcing function f. Although we are only considering the width and amplitude, often the center, (x_0, y_0), of the forcing function can be a beneficial element to add to the parameters of interest. Simulating the wave equation with parameters $\vec{\theta} = (2, 4)^T$, Figure 3.2 shows the solution at $(x, y) = (50, 50)$. In order to create synthetic measurements $\vec{c}_0$, we need to quantify the amount of energy deposited on the sensor in terms of the wave magnitude. We know from basic dimensional analysis, that $(intensity)(time) = (\frac{power}{area})(time) = (\frac{energy}{time})(\frac{time}{area}) = \frac{energy}{area}$, in which the area of the sensor is fixed. We also know that the wave intensity is proportional to the wave magnitude squared, $u^2(x, y, t)$ [1]. Therefore the total energy delivered to the sensor at any given location is proportional to the integral over time of the wave intensity.

$$e(x, y) = \int_0^T u^2(x, y, t)dt. \tag{3.11}$$

Hence, to generate the synthetic data, the integral of the solution squared, with some parameters $\vec{\theta}$, is approximated at each point in the grid and a selection of arbitrary locations are chosen for $\vec{c}_0$.

Once we have generated the vector of measurements, $\vec{c}_0$, we want to estimate the parameters, $\hat{\vec{\theta}}$, that will accurately model the measurements. This is done through



.

Figure 3.2: Solution to the wave equation with Gaussian forcing function at the point (50, 50)

optimization of the cost function,

$$J(u) = ||\vec{c}(u(\vec{\theta})) - \vec{c}_0||_2^2, \tag{3.12}$$

in which the norm of the difference between the observed values $\vec{c}_0$, and the measurements from the measurement functional (3.13) is minimized. We will denote the solution to the wave equation, $u(x, y, t; \vec{\theta})$, with parameters $\vec{\theta}$ as $u(\vec{\theta})$ to avoid cumbersome notation. For the cost function (3.12), we use initial parameters $\vec{\theta}_0$ to compute the measurement functional,

$$\vec{c}_i(u(\vec{\theta})) = \int_0^T u^2(x_i, y_i, t, \vec{\theta})dt. \tag{3.13}$$

We want to find the parameters that gives the values of $\vec{c}(u(\theta))$ as close to $\vec{c}_0$ as possible. To do so we search for the minimum of the cost function. In this case, because we have noiseless data, the minimum of $J(u)$ is zero. Starting with the initial parameter, an optimization algorithm iteratively updates, building a sequence of parameters that converge to a local minimum of the cost function at $\hat{\vec{\theta}}$.

The gradient descent method, Algorithm 1, is a commonly used optimization method to find the minimum of a scalar valued function. It relies on the fact that the negative gradient of a function points in the direction of the fastest descent [3]. As with most optimization algorithms, this means it cannot distinguish between local minima and a global minimum. This algorithm only allows descending travel so for a function with several local minima, it will find the one closest to the initial guess. Due to this, it is highly important that the initial parameters are chosen carefully. The algorithm requires a small step size, $\alpha > 0$, and it stops once a tolerance is met or a maximum number of iterations are completed. Because of its simplicity, we first try to estimate the parameters by implementing the gradient descent algorithm 1, with the cost function $J(u)$ and initial parameters $\vec{\theta}_0$.

Algorithm 1: Gradient Descent

Input : $f : \mathbb{R}^n \to \mathbb{R}$

Output : $\hat{x}$, *approximate local minimum*

Set $\alpha > 0$

Choose $\vec{x}^0$

for $k = 1 : max\ iteration$ **do**

  $df = \nabla f(\vec{x}^{k-1})$

  $\vec{x}^k = \vec{x}^{k-1} - \alpha * df$

  if $||\vec{x}^k - \vec{x}^{k-1}|| < TOL$ **then**

    Stop

  end if

end for

Simulating the wave equation with (3.5) as the forcing function and $\vec{\theta} = (2, 4)^T$, measurements are arbitrarily selected at $x, y = [33, 55, 77]$ to be $\vec{c}_0$. We consider how to determine a suitable step size, α, to be used in the gradient descent algorithm. Using $\vec{\theta}_0 = (2.1, 4.1)^T$ as the initial parameter, and setting the tolerance to 10^{-6}, four different values for α are used to compare convergence time. Figure 3.3 shows

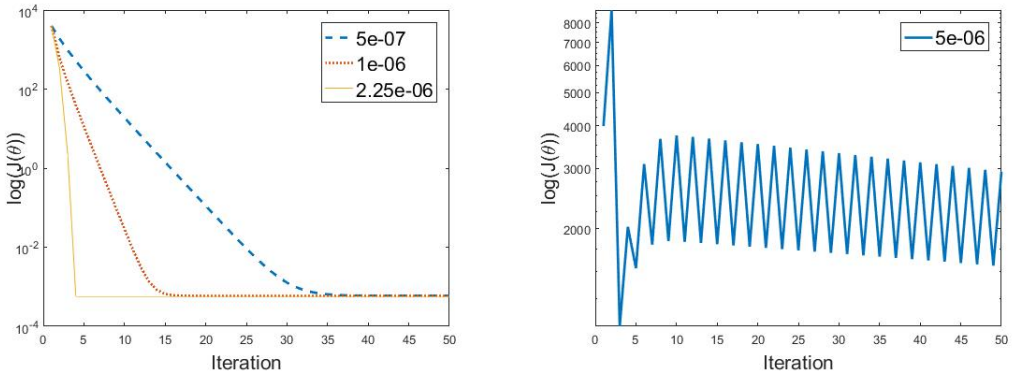


Figure 3.3: The convergence rate of the gradient descent algorithm for different step sizes α. *Left:* Shows convergence at different iterations for corresponding α. *Right:* Lack of convergence for larger α.

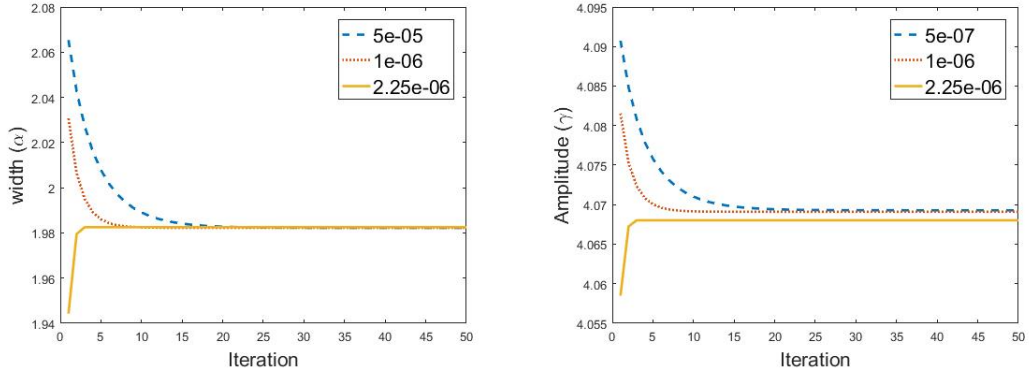


Figure 3.4: Parameter Estimations for the gradient descent algorithm for different step sizes α. *Left:* Width estimation for corresponding α. *Right:* Amplitude estimation for corresponding α.

the convergence rate for $\alpha = 5 \cdot 10^{-7}, 10^{-6}, 2.25 \cdot 10^{-6},$ and $5 \cdot 10^{-6}$ and Figure 3.4 shows the associated estimated parameters.

Each α in Figure 3.3(a) converges, and in particular, the largest of the three values converges the fastest. Figure 3.3(b) shows that for a slightly larger α there is no convergence. The algorithm overshoots the minimum and bounces back and forth, never converging. The smaller α is, the more likely it is to converge. Despite this,

the smaller α is, the longer it will take to converge. Ideally, one wants to choose the step size that allows the fastest convergence, meaning a slightly larger α, but because of the sensitivity of the algorithm with respect to α, this can cause divergence.

The estimated parameter values associated with each step size, α, along with the value of the cost function at convergence, are shown in Table 3.1. Even though the tolerance was set at 10^{-6}, the estimations, while still reasonable, are not as accurate as we would expect and the cost function is not zero. To compute the gradient for the algorithm, we use finite differences, so some error is introduced in the estimation. Additionally, if the minimum of the function is particularly flat, then the gradient descent takes very small steps and it stops iterating before actually meeting the tolerance.

Method	α	$\hat{\vec{\theta}}$	$J(u(\vec{\theta}))$	Iterations
	$5 \cdot 10^{-7}$	$(1.9824, 4.0693)^T$	0.00059546	39
Gradient	10^{-6}	$(1.9823, 4.0691)^T$	0.00058928	18
Descent	$2.25 \cdot 10^{-6}$	$(1.9826, 4.0680)^T$	0.00057096	4
`lsqnonlin`		$(2.0000, 3.9999)^T$	$3.3731 \cdot 10^{-23}$	4

Table 3.1: Comparison of Parameter Estimation

Matlab has several built in functions for optimization that are faster and more accurate. Table 3.1 shows the estimated parameters, $\hat{\vec{\theta}}$, and iterations for the gradient descent algorithm versus the Matlab function `lsqnonlin`. Using $\vec{\theta}_0 = (2.1, 4.1)^T$ and $\alpha = 2.25 \cdot 10^{-6}$ in gradient descent, convergence occurred in the same number of iterations as `lsqnonlin`; however, `lsqnonlin` is more accurate. Setting $\vec{\theta} = (2.5, 3.8)^T$, slightly further from the actual parameters and using $\alpha = 2.25 \cdot 10^{-6}$ for gradient descent, the two algorithms are again compared. The gradient descent method converges to $\hat{\vec{\theta}} = (2.1062, 3.6210)^T$ in 9 iterations while `lsqnonlin` converges to $\hat{\vec{\theta}} = (1.9999, 4.0000)^T$ in 5 iterations. Due to the speed and sophistication of `lsqnonlin`, we choose to use it over the gradient descent optimization.

Lastly, we have the ability to choose the number of measurements recorded in

an experiment and so we are interested in whether this number affects the convergence rate for the parameter estimation method. The algorithm, `lsqnonlin`, is used with 3, 20, 45 and 77 random measurement points. For each set of measurements, `lsqnonlin` converged in four iterations. Thus, we can determine that the number of measurement points does not change the convergence rate.

3.4 Noisy Data

All experimental data is affected by noise. Noise is anything that causes unwanted disturbances or indistinguishable changes to the data. In parameter estimation we ultimately want to be able to model the data without noise. Understanding how the noise affects the estimation of the parameters can be crucial in accurate modeling.

In the above discussion concerning optimization methods, `lsqnonlin` converged to $\vec{\theta}$ with high accuracy every time. This is because when we simulate an experiment with a computer, the results are ideal in that no noise, beyond numerical error, affects the data. To account for this, we need to add noise to the synthetic measurements, $\vec{c}_0$. In an experiment, one would have some knowledge about the types of noise expected; however, for the purposes of this thesis we will simply add varying percentages of the measurements to $\vec{c}_0$.

Again, using $(x, y) = [33, 55, 77]$ for the measurement locations of $\vec{c}_0$ and $\vec{\theta} = (2, 4)^T$, we add $5\%, 10\%, 15\%$, and 20% of each measurement to $\vec{c}_0$. The resulting parameter estimations are shown in Table 3.2. The addition of noise in the form of a percentage overestimates the amplitude parameter, while accurately estimating the width. Also, notice that with each addition of 5% noise, the error in the estimated amplitude increases by approximately 2.5%, or half that of the percentage of the noise added. In most cases, we find that with the addition of noise in the measurements $\vec{c}_0$, the estimations are still valid; however, the minimum of the cost function (3.12) is no longer zero. In this case, because we have just added a percentage of each

measurement to the measurements themselves, we only increase the amplitude and so the algorithm converges to the actual parameters that describe the data. Because of this, the value of the cost function at convergence is still very close to zero.

%	$\hat{\theta}$	$J(u(\vec{\theta}))$	Approx. Error in γ
5%	$(1.9999, 4.0988)^T$	$7.5932 \cdot 10^{-26}$	2.5%
10%	$(2.0000, 4.1952)^T$	$5.6596 \cdot 10^{-26}$	5%
15%	$(2.0000, 4.2895)^T$	$1.1763 \cdot 10^{-26}$	7.5%
20%	$(2.00004.3818)^T$	$4.757 \cdot 10^{-25}$	10%

Table 3.2: Parameter estimation with the addition of noise as a percent of the measurement value.

Chapter 4

Optimal Experimental Design

This chapter discusses the methods involved in determining the optimal placement of an additional sensor in an experiment. It then combines the parameter estimation done in Chapter 3 along with the sensor placement covered here, to design an optimal experiment in which the results are most reliable.

4.1 Optimal Sensor Placement

For optimal sensor placement we try to determine the best location to place a new sensor in an experiment such that there is the least amount of variation in the parameter estimation for the model. To do so, we fix some measurement points and quantify the uncertainty associated with placing a new sensor at (x, y). Let $\vec{\theta}$ be the parameters and $\vec{c}(u(\vec{\theta}))$ be the measurement functional from Chapter 3, (3.13), where $u(\vec{\theta})$ is the solution to the wave equation with parameters $\vec{\theta}$.

4.1.1 The Covariance Matrix

To measure the uncertainty of the parameter estimation at the new location (x, y), we need to compute the covariance matrix, C_θ, for the parameters $\vec{\theta}$. For n random variables, the covariance matrix is an $n \times n$ matrix that contains the variance of each variable along the diagonal entires, $i = j$, and the covariance of the variables on the off-diagonal entries, $i \neq j$. It assumes a normal probability distribution for the variables. For two random variables x_1 and x_2, the covariance of x_1 with x_2 is the same as the convariance of x_2 with x_1, thus the matrix is symmetric. The covariance matrix for $\vec{x} = [x_1, x_2, \cdots, x_n]^T$ is shown below.

$$C_{\mathbf{x}} = \begin{bmatrix} \sigma_{x_1} & \sigma_{x_1 x_2} & \dots & \sigma_{x_1 x_n} \\ \sigma_{x_2 x_1} & \sigma_{x_2} & \dots & \sigma_{x_2 x_n} \\ \vdots & & \ddots & \\ \sigma_{x_n x_1} & \dots & \dots & \sigma_{x_n x_n} \end{bmatrix} \tag{4.1}$$

From the covariance matrix we gain knowledge about the spread of the n dimensional data. The covariance describes how the variables vary together; if x_1 is positively covaried with x_2 then they both increase and decrease together and the slope of the data is positive. Likewise if they are negatively covaried, one increases while the other decreases and the slope is negative. The variance elements tell us how spread out the data is in each variable.

Starting with the the measurements $\vec{c}(u(\vec{\theta}))$, we will derive the covariance matrix for the associated parameters. We assume that the uncertainty in measurements are independent and identically distributed (IID), meaning they have the same probability distribution, and are independent of one another. When this is the case, the variables have a covariance of 0 so the covariance matrix is zero for all entries $i \neq j$. The assumption that they have the same probability distribution means that the variance, σ^2, for each variable is the same. The covariance matrix for the observed

measurements can then be written as in equation (4.2) where I is the identity matrix.

$$C_m = \begin{bmatrix} \sigma^2 & 0 & \dots & 0 \\ 0 & \sigma^2 & \dots & 0 \\ \vdots & & \ddots & \\ 0 & \dots & \dots & \sigma^2 \end{bmatrix} = \sigma^2 \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & & \ddots & \\ 0 & \dots & \dots & 1 \end{bmatrix} = \sigma^2 I \tag{4.2}$$

Next, for $f : \mathbb{R}^n \to \mathbb{R}^m$, let X be the $n \times m$ Jacobian matrix $X_{ij} = \frac{\partial f_i}{\partial x_j}$ with,

$$X = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \dots & \dots & \dots \\ \vdots & & & \vdots \\ \frac{\partial f_m}{\partial x_1} & \dots & \dots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \tag{4.3}$$

The Jacobian tells us how the function changes with respect to each variable. According to the Error Propagation Law [2] one can compute the covariance matrix for the measurements, C_m, using the Jacobian along with the covariance matrix for the parameters, C_θ as follows,

$$C_m = X C_\theta X^T \tag{4.4}$$

We want to estimate the variance in the parameters from the variation in the measurements at different locations so we want to compute the Jacobian for $\vec{c}(u(\vec{\theta}))$, $X_{ij} = \frac{\partial c_i}{\partial \vec{\theta_j}}(u(\vec{\theta}))$. To estimate X, we perturb each parameter by small ϵ and simulate the wave equation for each modified $\vec{\theta}$. Using the measurement functional (3.13), we can then estimate the Jacobian as in (4.5) using Algorithm 2.

$$X_{ij} = \frac{\partial c_i}{\partial \theta_j} u(\theta) \approx \frac{c_i(u(\theta + \epsilon e_j)) - c_i(u(\theta))}{\epsilon}. \tag{4.5}$$

We can then derive C_θ from (4.4) as follows:

$$\begin{aligned} & C_m = X C_\theta X^T \\ & \implies C_\theta = (X^T X)^{-1} X^T C_m X (X^T X)^{-1} \\ & \text{Recall, } C_m = \sigma^2 I \\ & \implies C_\theta = \sigma^2 (X^T X)^{-1} \end{aligned} \tag{4.6}$$

Algorithm 2: Jacobian

Approximation of the Jacobian of a function f at $\vec{x}$

Input : $f_x = f(\vec{x})$

Output : J, *approximated Jacobian*

Set $\epsilon > 0$

for $k = 1 : length(x)$ **do**

$y = x$

$y^k = y^k + \epsilon$

$f_y = f(y)$

$J(:, k) = \frac{(f_y - f_x)}{\epsilon}$

end for

In C_θ, each of the diagonal elements is an estimated variance for the associated parameter. For $\theta = (2, 4)^T$, we arbitrarily fix the point $(x, y) = (33, 25)$ and consider the covariance matrix of the parameters when an additional sensor is added at $(58, 50)$, a distance of 36.05 units from the fixed sensor point.

$$C_\theta = \begin{bmatrix} 2.3537 & -9.0884 \\ -9.0884 & 35.0931 \end{bmatrix} \tag{4.7}$$

The covariance matrix (4.7), shows a negative covariance between the two parameters. If the amplitude increases, the width that gives the same sensor reading, decreases and vice versa; this seems reasonable as it is intuitively what one would expect. The value in the upper left corner is the estimated variance for the width, and the value in the bottom corner is the estimated variance for the amplitude. With the addition of this point, there is more variance in the amplitude parameter than the width. We want to find the best location for an additional sensor to be placed, with potentially more than one fixed location, such that the variation in both parameters is the smallest. To do so, we need to find a way to quantify the covaraince matrix C_θ.

Suppose we have a covariance matrix such that the covariances are zero, then the eigenvalues are exactly the variances of the variables. Recall that for a linear transformation of $\vec{x}$ by a matrix A, we can write,

$$A\vec{x} = \lambda\vec{x},$$

where the eigenvalue λ is the magnitude of the eigenvector $\vec{x}$. If the covariance matrix consists of only the variances, then the largest eigenvalue describes the variable with the largest spread. If the covariances are not zero, then we need to consider what the eigenvalues along with the associated eigenvectors are telling us about the spread.

Suppose we have the following covariance matrix,

$$C = \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix} \tag{4.8}$$

with eigenvalues $\lambda_1 = 5$, $\lambda_2 = 1$ and eigenvectors $\vec{v}_1 = (1,1)^T$ and $\vec{v}_2 = (-1,1)^T$. The eigenvectors tell us the direction of the spread of the variables with magnitude λ. Figure 4.2 shows the spread for the given covariance matrix (4.8).

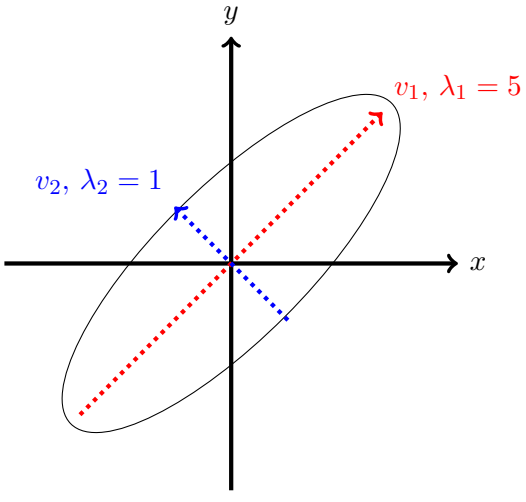


Figure 4.1: Spread of data for the covariance matrix (4.8) determined from eigenvalues and associated eigenvectors.

It becomes clear that the eigenvalues are a measure of the variance of the data. Therefore, if we want to find the location for the next sensor placement, such that

the variance in the parameters is the least, we need to consider some metric on the eigenvalues of the covariance matrix. To do so, we introduce the metrics,

$$\Phi_1 = det(C_\theta) = \prod_{i=1}^{n} \lambda_i, \tag{4.9a}$$

$$\Phi_2 = Tr(C_\theta) = \sum_{i=1}^{n} \lambda_i, \tag{4.9b}$$

$$\Phi_3 = \rho(C_\theta) = \max |\lambda_i|, \tag{4.9c}$$

where each is a functional of the covariance matrix C_θ of the model parameters $\vec{\theta}$. Each of the metrics gives a quantification of the overall variation in the parameters at given locations. Because we want to find (x, y) such that the overall variance is the least, we seek to minimize each Φ.

4.1.2 Design Parameter Optimization

To estimate the covariance matrix C_θ we need to first approximte the Jacobian, X, of $\vec{c}(u(\vec{\theta}))$ with parameters θ, using Algorithm 2. Once the Jacobian has been estimated we can approximate the covariance matrix using (4.6). Although we do not know the variance, σ^2, of the parameters, because we have assumed them IID, it is just a coefficient for the covariance matrix C_θ and so we set it to 1.

$$C_\theta = (X^T X)^{-1} \tag{4.10}$$

For $\vec{\zeta} = (x, y) \in \Pi$ where Π is the space of design parameters, we wish to minimize $\Phi(C_\theta(\zeta))$. Fixing the same three points we have been, $x, y = [33, 55, 77]$, we create a grid and treat each point on the grid as a potential fourth location for a sensor to be placed. To get some intuition as to what the metrics (4.9) look like and where minimums occur, we plot each Φ as a function of the fourth location (x, y), as seen in Figure 4.2. Each Φ has a global minimum at the center; however, we would like to apply the methods discussed in this thesis to an electromagnetic wave in which

the intensity at the center is very high. Therefore, it is not a practical place to put a sensor, thus we need to look for local minima away from the center.

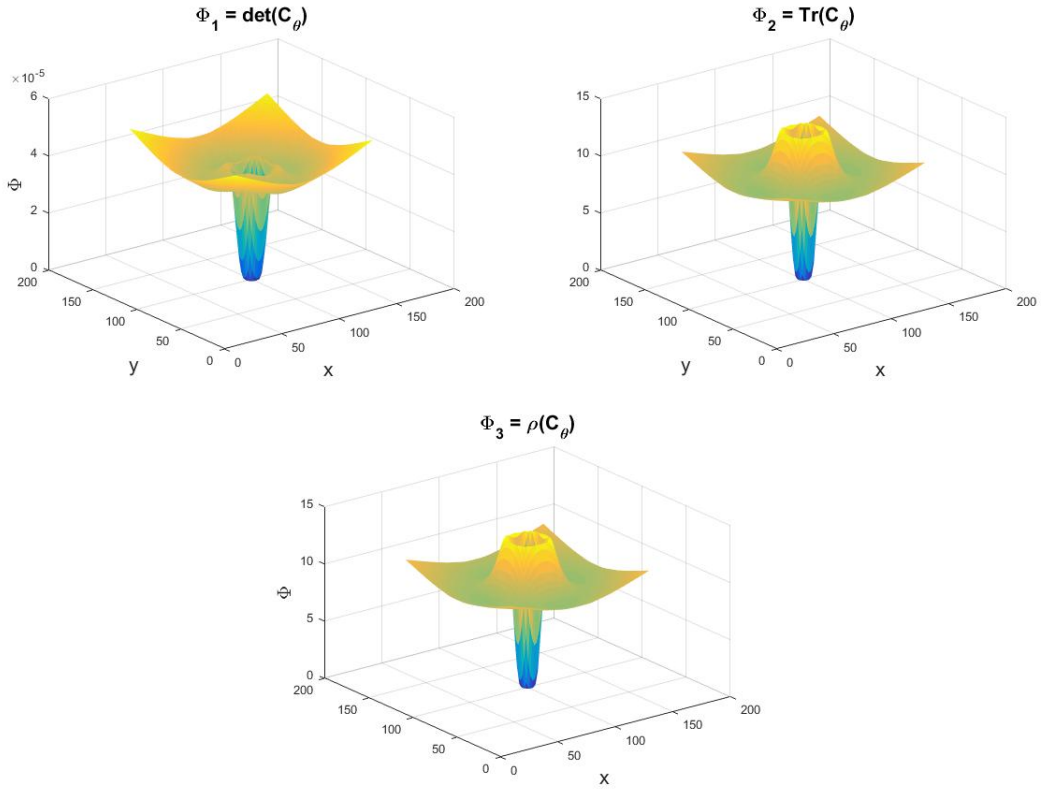


Figure 4.2: Φ_1, Φ_2, Φ_3 plotted as a function of grid points (x, y). *Top left:* $\Phi_1 = det(C_\theta)$. *Top right:* $\Phi_2 = Tr(C_\theta)$. *Bottom:* $\Phi_3 = \rho(C_\theta)$

Since we have data for the Φ functions on the computational grid, to find the location of a local minimum for each, we implement a simplified version of the gradient descent algorithm. Again, we start with an initial guess for the design parameter $\vec{\zeta}_0$ and try to find the location $\vec{\zeta}$ such that we are at a minimum of Φ. Starting with $\vec{\zeta}_0$, Algorithm 3 evaluates the four surrounding locations and moves to the location where Φ is smaller. Doing this iteratively, the algorithm stops once it reaches a $\vec{\zeta}$ such that no surrounding locations have a smaller value of Φ. Like the gradient descent, it can only move downhill and so we must use Figures 4.2 to carefully choose $\vec{\zeta}_0$ such that the algorithm does not find the global minimum at the center.

Algorithm 3: Simplified Gradient Descent

Input : $f = \Phi(C_\theta(z))$

Output : z *approximated minimum location*

Choose $z = (x, y)$

for $k = 1 : max\ iteration$ **do**

  $z_1 = z + (1, 0)$

  $z_2 = z + (0, 1)$

  $z_3 = z + (-1, 0)$

  $z_4 = z + (0, -1)$

  for $i = 1 : 4$ **do**

    if $f(z_i) < f(z)$ **then**

      Go to $f(z_i)$

    else

      $z = z$

    end if

  end for

end for

4.2 Optimal Experimental Design

Now that we have an understanding of how to estimate model parameters and how to find an optimal sensor placement, we combine the two ideas to design an optimal experiment such that the results can be most accurately modeled.

In a real experiment, we have to deal with noisy data that prevents us from finding the exact parameters of interest, as discussed in Chapter 3. In optimal experimental design we strive to design an experiment such that the location of the sensors minimize the effect of noise on the parameter estimation. In doing so, we can more accurately model the phenomena. Using data collected from conducting a real experiment, we estimate the parameters $\vec{\hat{\theta}}$. We then use these parameters along with the computational model to find the optimal location to put a new sensor. Because we have to deal with noisy data, we determine the experimental design iteratively as shown in Figure 4.2.

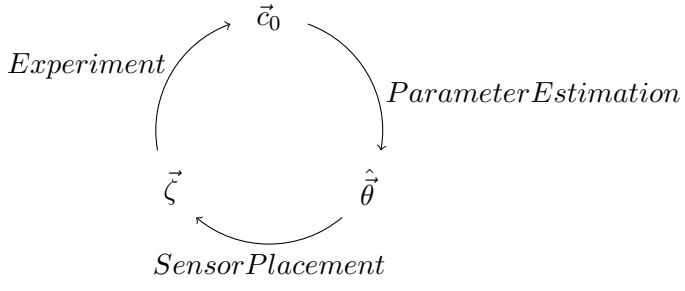


Figure 4.3: Iterative optimal experimental design diagram.

Starting with experimental measurements $\vec{c}_0$, we solve the parameter estimation problem by simulating the wave equation with an initial guess, $\vec{\theta}_0$, and minimizing the cost function (3.12) to find the estimated parameters, $\vec{\hat{\theta}}$. Then we use $\vec{\hat{\theta}}$ and an initial $\vec{\zeta_0}$, to solve the design parameter problem by minimizing the metrics (4.9) on the covaraince matrix, C_θ, to determine the location, $\vec{\zeta}$, in which there is the least variation in the model parameters. We then perform the experiment again, this time

placing the 4^{th} sensor at the location, $\zeta = (x, y)$, that has just been determined. Going through the steps again, we estimate the parameters using $\vec{\theta_0} = \vec{\hat{\theta}}$, and $\zeta_0 = \zeta$, we determine where to move the 4^{th} sensor so that when we perform the experiment again, we get a slightly better parameter estimation. This process is iterated until a maximum number of iterations has been executed. Once we reach the maximum, we have determined the optimal location for the 4^{th} sensor so that performing the finale experiment with the senor in place gives us the most accurate parameter estimation.

Chapter 5

Conclusion

Through the work in this thesis, we have developed an iterative method to determine the optimal location for an additional sensor to be added such that we can reliably estimate the model parameters of a Gaussian wave source. Starting with experimental data, we minimize the cost function to estimate the model parameters, $\vec{\theta} = (\alpha, \gamma)$. Using the estimated parameters, $\hat{\vec{\theta}}$, we can determine where to optimally place an additional sensor. Keeping in mind that all experimental data has noise, we design an iterative process to determine where to place a sensor such that the effect of noise on the estimated parameters in minimal.

We only considered adding one additional sensor, but we are not limited to this. The optimization methods and iterative design discussed in this thesis apply to adding any number of sensors. No matter how many we choose to add or have the resources for, we ultimately want the variation in the parameter estimation to be the least, so we seek to minimize the metrics defined on the covariance matrix, C_θ.

Using the optimal experimental design method to characterize a wave source has many applications. In particular, we can expand the finite difference method of lines discretization to the Finite-Difference Time-Domain method for solving Maxwell's Equations. Then using the optimization methods discussed, we can use measure-

ments from thermoluminescent dosimeters to optimally design an experiment so that we can accurately characterize the propagation of an electromagnetic wave in a physical domain.

There remain lingering questions that merit consideration. For example, how well can we characterize a more general wave source. In this thesis, we considered the case where the forcing function was explicitly know, if this is not the case, how well can we expect to characterize the model? Additionally, there are many metrics that can be defined on the covariance matrix beyond the three that we considered here. Is there a way to determine which is the optimal one to use for a particular problem? Lastly, we assume that the noise in the measurements is a Gaussian distribution, but what if that is not the case? How can we determine the distribution of the noise, and if it is not Gaussian, do the methods to find the optimal sensor placement still apply?

References

[1] Thomas Arias. Notes on interference and diffraction. Cornell Univeristy, Department of Physics, September 2015.

[2] Kai Oliver Arras. An intoduction to error propogation: Derivation, meaning and examples of the equation $c_y = f_x c_x f_x^t$. Technical report, Robitic Systems, Swiss Ferderal Institute of Technology Lausanne, September 1998.

[3] Florent Brunet. *Contributions to Parametric Image Registration and 3D Surface Reconstruction*. PhD thesis, Univeristy d'Auvergne, November 2010.

[4] Steve G. Johnson. Notes on perfectly matched layers (pml). Technical report, MIT, March 2010.

[5] Albert Tarantola. *Inverse Problem Theory and the Methods for Model Parameter Estimation*. SIAM, Philadelphia, 2005.

[6] Juri Ranieri. Amina Chebira. Martin Vetterli. Near-optimal sensor placement for linear inverse problems. January 2014.