Full Waveform Inversion

ALTEN Netherlands

This version is annotated by S MELISSEN, in red before 2018-10-10

This version is annotated by S MELISSEN, in orange after 2018-10-11

I tried to ~~ocreret~~ **correct** and point out some typos

I used the `courier font` for `explicit code` and `file names`

I moved the heavier, superfluous math to appendices

And in blue I ask questions that may be relevant before 2018-10-10.

And in purple I ask questions that may be relevant after 2018-10-11.

OK for Bernhard and ***NOT OK for Bernhard***s of last cycle included

October 12, 2018

Contents

1 Preface

This document describes a method to predict the size and shape of unexplored oil reserervoirs based on acoustic testing that ALTEN developed based on Peter Haffinger's Ph.D. thesis (TU Delft 2013). The current version is in progress and written by S. Melissen, under the supervision of André Prins and Bernhard Righolt. It follows a pre-version by Nikita Mahto. The development currently takes place on the "serial"-branch of the git repository, where the parallelization and non-C++ code is being completely split out to have a comprehensible, single CPU version that works on a variety of machines in a more or less "plug-and-play" manner. The Master branch functions relatively well (see "Results"). This branch will be split out into a development branch, where the parallelization introduced in the Master branch is cleaned up completely and clearly split out as one option upon several computational methods. The Docs branch - where this document will be posted - is supposed to clear up documentation issues, the code being in a functioning, albeit very confusing state.

2 Introduction

The Full Wave inversion is **an advanced** ***OK for Bernhard*** imaging technique which can be achieved by ~~illuminating~~**irradiating** the interior of ~~the~~**an** object with e.g. acoustic or electromagnetic waves, while receivers placed around the object measure the response. From these measurements an image of the properties of the object's interior can be derived.
It has various applications including,

1. seismology
2. medical: ultrasonic applications

Figure 1 displays the perspective views of Axial Volcanos internal structure, constructed using elastic full waveform inversion (FWI) results along seismic lines across caldera and along the secondary magma reservoir. (a): P-wave velocity structure. (b): Total gradient magnitude of the P-wave velocity structure. (c) Reflectivity structure. The red mesh marks the extent of the main magma reservoir on (a) and (b).

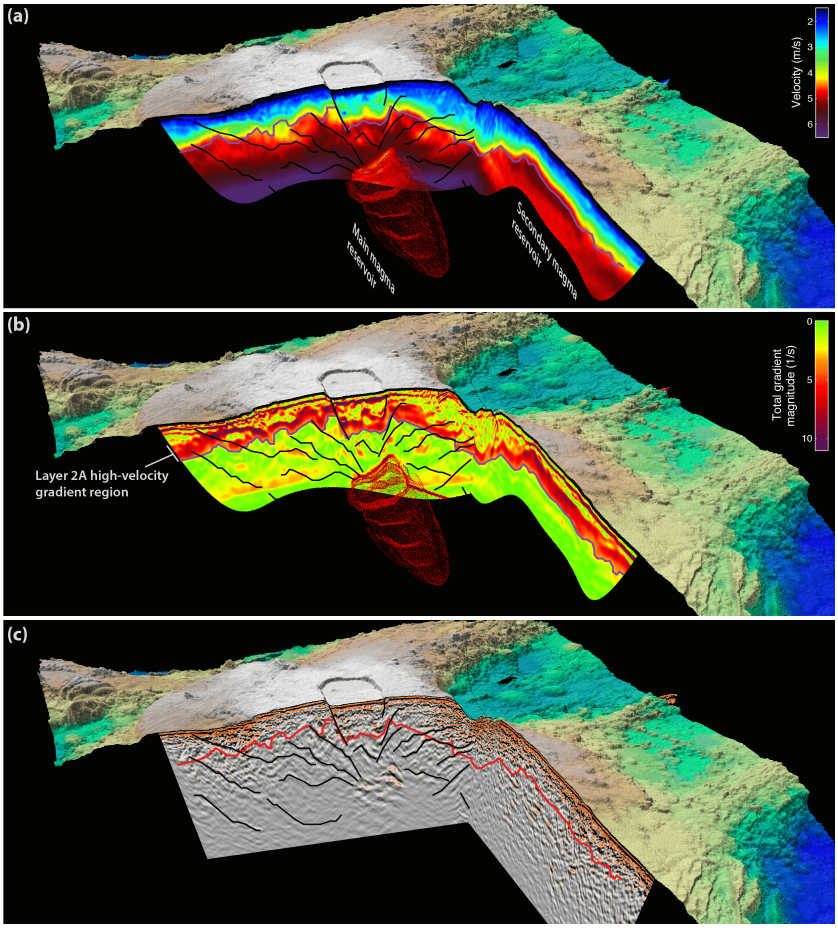

Figure 1: Seismic Full Waveform Inversion

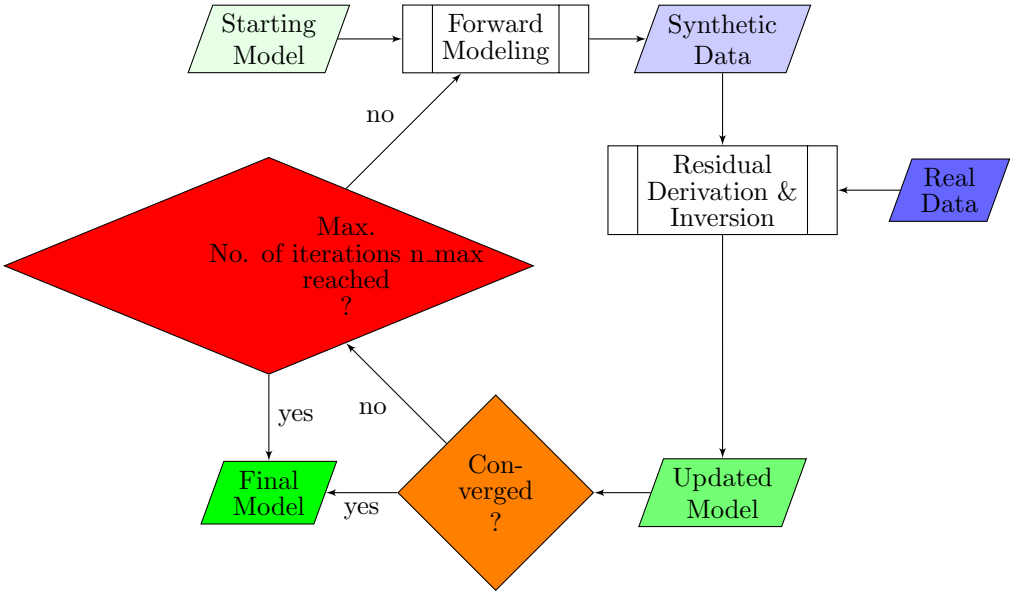

Figure 2: Full Waveform Inversion Methodology. **Note that this is now in LaTeX, rendering it more easily manipulable. STM: In the "Results" section I see five entries. Five is `n_max`, the maximum number of iterations. It seems to be a nicely converged temple. Perhaps the current tolerance is too tight? Or maybe something is wrong in evaluating the tolerance?**

In FWI, the aim is to find the media properties on a dense subsurface grid, instead of image amplitudes. Synthetic seismic data are forward modeled and compared with the measured dataset. The differences are then used to update the full set of media properties and the process is repeated until a satisfactory match between modeled and measured dataset is obtained. In Figure 2 **a** flow chart for the process is shown. It can be seen that, in contrast to imaging techniques, a closed loop exists, meaning that feedback from the obtained media properties is used to verify any inconsistency between the measured data and the inversion result. In the flow chart we can see that we begin with forward modelling and estimate the synthetic data. Further, we iterate the data in order to minimise the residuals as compared to the real data and obtain the final data once the residuals are below the specified tolerance levels. **STM: Show this requirement in the picture *OK for Bernhard* STM: implemented!**.
This document ~~explains~~**provides** an overview of the equations employed, the computational model and the code algorithm. **STM: I may be wrong, but I don't think that is what it does. It provides an overview of "the model" - the mathematical equations used - the parallellization (here referred to as "the model") and finalizes with a description of each individual module in the code (I would not call this "the code algorithm" at all myself.) BR: Agree, this is implemenation STM: Thanks Bernhard, I reorganized this document. Also, the following sentence is added to the Introduction.**
The approach is based on the Ph.D. thesis by Peter R. HAFFINGER (TUDelft 2013) "Seismic Broadband Full Waveform Inversion by shot/receiver refocusing"

3 The Model

~~2. Mathematical Model~~

3.1 Physical model

BR: For this section, BR prefers to start with eq 5; because G follows from 2, which follows from 5 STM: Pushed to next iteration of improvements. The mathematical model outlines the equations used in the code for the FWI. **STM: weird sentence** We begin with the Green's function and apply it to the ~~the~~ simplest acoustic equation - the Helmholtz equation. For the acoustic case with constant density, the wave equation can be described by **two equations *OK for Bernhard***, being the data equation and the object equation. The data equation describes the seismic dataset, in terms of a total field p_{tot} at each grid point in the subsurface, the contrast function χ and the Green's function $\mathcal{G}$ **STM: corrected** in a background medium: **STM: It is slightly confusing that the contrast function is defined later unannounced and Green's is provided immediately**

$$p_{\text{data}}(\mathbf{x}_{\text{r}}, \mathbf{x}_{\text{s}}, \omega) = \int \mathcal{G}(\mathbf{x}_{\text{r}}, \mathbf{x}, \omega) p_{\text{tot}}(\mathbf{x}, \mathbf{x}_{\text{s}}, \omega) \chi(\mathbf{x}) \, \mathrm{d}\mathbf{x} \tag{1}$$

The indices s and r are for "source" and "receiver" *OK for Bernhard*. Reading from right to left, equation 1 can be understood as follows: A source transmits a wavefield that propagates to every point in the subsurface. Note that this wavefield p_{tot} is generally quite complex because it takes the interaction of all scatterers in the subsurface already into account. This wavefield is creating secondary sources in all points where the contrast function χ is non-zero. The secondary sources transmit energy through a smooth background medium to the receivers, represented by the Green's function $\mathcal{G}$ in equation 1.
The measured seismic data at every receiver are then a summation of all secondary sources. It should be mentioned that direct waves, including ground roll and surface waves are supposed to be removed from the measured data to obtain p_{data}. **BR: Not sure if I really understand this. STM: From what I understand, this means that within this formalism, what we understand as the "received data" is simply all the sound that has bounced off the walls once, so "all secondary sources". Primary sources, the sound maker, is removed from the data and I guess tertiary and up is simply not accounted for? Do you agree? Comment treated?**

We use the 2-D case for the Helmholtz equation. Further, the Green's function is calculated for this equation. The contrast function has to be determined. Further the conjugate scheme is established to determine the error functional.

Measured seismic data always contains some form of noise, so the inversion process is required to be regularised. Therefore, we extend the conjugate gradient scheme ~~in a way that it contains~~ **so as to include** a multiplicative regularisation factor.

3.1.1 The Green's function

~~2.1. The Green's function~~
A Green's function is the impulse response of an inhomogeneous linear differential equation

defined on a domain, with specified initial conditions or boundary conditions. The Green's function of this equation is defined as the solution $\mathcal{G}(\mathbf{x}, \mathbf{y})$, of the equation

$$[\nabla^2 \mathcal{G}(\mathbf{x}, \mathbf{y}) + k_0^2 \mathcal{G}(\mathbf{x}, \mathbf{y}) = -\delta(\mathbf{x} - \mathbf{y}).] \tag{2}$$

The solution to (2) is then given by

$$[u_{\text{ind}}(\mathbf{x}) = \int_{\mathbf{y} \in \mathbb{R}^n} \mathcal{G}(\mathbf{x}, \mathbf{y}) f_{\text{ind}}(\mathbf{y}) \, \mathrm{d}\mathbf{y}. \tag{3}$$

BR: What is "ind" **STM: to be determined** The interesting cases are the 2D and 3D case. We will first **STM: only?** focus on the 2D case. The Green's function is then given by

$$[\mathcal{G}(\mathbf{x}, \mathbf{y}) = \frac{\imath}{4} H_0^{(1)}(k_0 \|\mathbf{x} - \mathbf{y}\|) = -\frac{1}{4} Y_0(k_0 \|\mathbf{x} - \mathbf{y}\|) + \frac{\imath}{4} J_0(k_0 \|\mathbf{x} - \mathbf{y}\|).] \tag{4}$$

In the above equation, the H_0 , J_0 and Y_0 are defined as the Henkel's functions and can be further researched at : `http://amcm.pcz.pl/get.php?article=2012_1/art_06.pdf`

3.1.2 Helmholtz equation

~~2.2 Helmholtz equation~~

The simplest model we can use is the acoustic Helmholtz equation. We use a scalar pressure field $u(\mathbf{x})$ and the domain is modeled using $\chi(\mathbf{x})$. and is called the contrast and can be directly related to the wave speed at that point in the domain. Mathematically the equation has the form:

$$\nabla^2 u(\mathbf{x}) + k(\mathbf{x})^2 u(\mathbf{x}) = -f_{\text{ext}}(\mathbf{x}). \tag{5}$$

The wave number k is given by $k = \frac{\omega}{c}$ where ω is the angular frequency **STM: in Hz?** and c ~~is~~ the acoustic wave velocity **in m/s** ***OK for Bernhard*** of the true medium. We split the pressure field into $u(\mathbf{x}) = u_0(\mathbf{x}) + u_{\text{ind}}(\mathbf{x})$, where $u_0(\mathbf{x})$ is defined as the field given by the background velocity and external sources,

$$\nabla^2 u_0(\mathbf{x}) + k_0^2 u_0(\mathbf{x}) = -f_{\text{ext}}(\mathbf{x}). \tag{6}$$

Substituting in (6) results in

$$\nabla^2 u_{\text{ind}}(\mathbf{x}) + k_0^2 u_{\text{ind}}(\mathbf{x}) = -f_{\text{ind}}(\mathbf{x}), \tag{7}$$

with $f_{\text{ind}}(\mathbf{x}) = k_0^2 \chi(\mathbf{x}) u(\mathbf{x})$ the induced source term.

3.1.3 The Contrast function

~~2.3. The Contrast function~~

The contrast function is defined as:

$$\chi(\mathbf{x}) = 1 - \left(\frac{c_0(\vec{x})}{c(\vec{x})} \right)^2. \tag{8}$$

It depends on the difference between a known background medium $c_0(\vec{x})$ and the true, but unknown, subsurface model $c(\vec{x})$. The total field on the right-hand side in equation 1 is dependent on the contrast, because it contains the interaction between all subsurface scatterers. Then it

follows that there is a nonlinear relationship between the subsurface properties and the measured seismic data.
Notice that the contrast is generally unknown, and will be approximated using an iterative scheme. During each step the induced source is considered constant and known so we basically solve the same equation as (3) each step.

3.2 Solver and Noise

3.2.1 The Conjugate Gradient (CG) Scheme

The Conjugate Gradient Scheme
BR: Should this be the start of a new chapter? STM: Proposal following BR's question to refer to this and the following chapters as a subsection of NEW 3.2 "Solver and noise" and to refer to the previous sections as subsections under a 2.1 "Physical Model" with current chapter 3 referred to in the future as "Solver and Noise" and the main chapter 2 as "Model". In the inversion scheme, we try to minimise the difference between the measured data p_{data} and the modelled data that is obtained using the currently best estimate of the contrast function and the fixed total field.
The residual between the measured and modelled data is obtained as:

$$r(\mathbf{x}_{\text{r}}, \mathbf{x}_{\text{s}}, \omega) = p_{\text{data}}(\mathbf{x}_{\text{r}}, \mathbf{x}_{\text{s}}, \omega) - [\mathcal{K}_\chi]\,(\mathbf{x}_{\text{r}}, \mathbf{x}_{\text{s}}, \omega), \tag{9}$$

BR: I always found the K symbol confusing STM: I have verified that this is the same notation that Haffinger uses. So I will add a specific reference to that thesis here, where you can find more background on that particular part.
with

$$[\mathcal{K}_\chi]\,(\mathbf{x}_{\text{r}}, \mathbf{x}_{\text{s}}, \omega) = \int \mathcal{G}(\mathbf{x}_{\text{r}}, \mathbf{x}, \omega) p_{\text{tot}}(\mathbf{x}, \mathbf{x}_{\text{s}}, \omega) \chi(\mathbf{x})\, \mathrm{d}\mathbf{x} \tag{10}$$

a linear operator in χ, and where we adopt the specific notation by Haffinger (Haffinger, Ph.D. thesis 2013, TU Delft) for consistency. The error functional is equal to

$$F(\chi) = \eta \int |r(\mathbf{x}_{\text{r}}, \mathbf{x}_{\text{s}}, \omega)|^2\, \mathrm{d}\mathbf{x}_{\text{s}}\, \mathrm{d}\mathbf{x}_{\text{r}}\, \mathrm{d}\omega, \tag{11}$$

with

$$\eta^{-1} = \int |p_{\text{data}}(\mathbf{x}_{\text{r}}, \mathbf{x}_{\text{s}}, \omega)|^2\, \mathrm{d}\mathbf{x}_{\text{s}}\, \mathrm{d}\mathbf{x}_{\text{r}}\, \mathrm{d}\omega, \tag{12}$$

so that for $\chi = 0$ we have $F = 1$. Notice the implicit dependency on χ.
We want to find a sequence of contrast functions, $\chi^{(n)}(\vec{x}), n = 1, 2, ...$, in which error functional decreases with increasing iterations. Therefore, after each iteration the contrast function is updated as:

$$\chi^{(n)}(\vec{x}) = \chi^{(n-1)}(\vec{x}) + \alpha_n \zeta_n(\vec{x}) \tag{13}$$

Here, the step size of the update is determined by the parameter α_n while the update directions are conjugate gradient directions given by

$$\zeta_1(\vec{x}) = g_1(\vec{x}), \zeta_n(\vec{x}) = g_n(\vec{x}) + \gamma_n \zeta_{n-1}(\vec{x}) \tag{14}$$

The functional derivative w.r.t. χ in direction $\mathbf{d}$ of the error functional is equal to

$$\frac{\partial F(\chi, \mathbf{d})}{\partial \chi} = \lim_{\epsilon \to 0} \frac{F(\chi + \epsilon \mathbf{d}) - F(\chi)}{\epsilon} \tag{15a}$$

$$= -2\eta \int \mathrm{Re}\left\{ \left[\mathcal{K}_{\mathbf{d}}\right](\mathbf{x}_\mathrm{r}, \mathbf{x}_\mathrm{s}, \omega)^\dagger r(\mathbf{x}_\mathrm{r}, \mathbf{x}_\mathrm{s}, \omega) \right\} \, \mathrm{d}\mathbf{x}_\mathrm{s} \, \mathrm{d}\mathbf{x}_\mathrm{r} \, \mathrm{d}\omega \tag{15b}$$

The derivation of this equation is provided in section 7.1. ***OK for Bernhard***
For the discrete $\mathcal{K}$ operator the integrand will have the form

$$g_n(\vec{x}) = \eta \mathrm{Re}\left\{ [\mathcal{K}^\star \mathbf{r}_{n-1}](\vec{x}) \right\} \tag{16}$$

where the $\star$ denote element wise multiplication per row and Re denotes that only the real part will be used. The adjoint operator $[\mathcal{K}^\star \mathbf{r}_{n-1}]$ can be seen as a backprojection operator that maps the residual between measured data and modelled data from the surface domain to its associated location in the scattering domain.
In our conjugate gradient scheme we make use of the Polak-Ribière direction,

$$\gamma_n = \frac{\int g_n(\vec{x})[g_n(\vec{x}) - g_{n-1}(\vec{x})] d(\vec{x})}{\int g_{n-1}(\vec{x}) g_{n-1}(\vec{x}) d(\vec{x})} \tag{17}$$

The optimal step size is found from the minimisation of cost functional equation, by setting the derivative equal to zero. Consequently the optimal step size becomes:

$$\alpha_n = \frac{\mathrm{Re}\left\{ \int\int\int r^\star_{n-1}(\mathbf{x}_\mathrm{r}, \mathbf{x}_\mathrm{s}, \omega)[\mathcal{K}\zeta_n](\mathbf{x}_\mathrm{r}, \mathbf{x}_\mathrm{s}, \omega) d\vec{x_s} d\vec{x_r} d\omega \right\}}{\int\int\int \mid [\mathcal{K}\zeta_n](\mathbf{x}_\mathrm{r}, \mathbf{x}_\mathrm{s}, \omega) \mid^2 d\vec{x_s} d\vec{x_r} d\omega} \tag{18}$$

To initialise the conjugate gradient scheme, we assume, $\chi^0(\vec{x}) = 0$, leading to $r_0(\mathbf{x}_\mathrm{r}, \mathbf{x}_\mathrm{s}, \omega) = p_{data}(\mathbf{x}_\mathrm{r}, \mathbf{x}_\mathrm{s}, \omega)$. **STM: correction of superscript** Therefore, we find the gradient as :

$$g_1(\vec{x}) = \eta \mathrm{Re}\left\{ [\mathcal{K}^\star p_\mathrm{data}](\vec{x}) \right\} \tag{19}$$

and we get the first update parameter as :

$$\alpha_1 = \frac{\mathrm{Re}\left\{ \int\int\int p^\star_\mathrm{data}(\mathbf{x}_\mathrm{r}, \mathbf{x}_\mathrm{s}, \omega)[\mathcal{K} g_1](\mathbf{x}_\mathrm{r}, \mathbf{x}_\mathrm{s}, \omega) d\vec{x_s} d\vec{x_r} d\omega \right\}}{\int\int\int \mid [\mathcal{K} g_1](\mathbf{x}_\mathrm{r}, \mathbf{x}_\mathrm{s}, \omega) \mid^2 d\vec{x_s} d\vec{x_r} d\omega} \tag{20}$$

3.2.2 Multiplicative Regularisation

~~2.5. Multiplicative Regularisation~~
Since all seismic data contains some form of noise, the inversion process needs to be stabilised. Therefore, the CG scheme is extended in a way that it contains a multiplicative regularisation factor.
The error functional $\mathcal{F}^{tot}$ becomes a product of the original error functional and a newly introduced regularisation factor $\mathcal{F}^{reg}$:

$$\mathcal{F}^{tot}_n = \mathcal{F}^{data}_n \mathcal{F}^{reg}_n \tag{21}$$

The following weighting function is used:

$$b^2_n = (\int_{\vec{x}} d\vec{x})^{-1} (\mid \nabla \chi^{n-1} \mid^2 + \delta^2_{n-1})^{-1} \tag{22}$$

In the above equation, the δ^2_n is the steering factor and can be defined as:

$$\delta_n^2 = (\int_{\vec{x}} d\vec{x})^{-1} \int_{\vec{x}} \mid \nabla \chi^{n-1} \mid^2 d\vec{x} \tag{23}$$

The new total cost functional then becomes a ~~sum~~ **product *OK for Bernhard*** of two second order polynomials:

$$\mathcal{F}_n^{tot} = (\mathcal{A}_2\alpha_n^2 + \mathcal{A}_1\alpha_n + \mathcal{A}_0)(\mathcal{B}_2\alpha_n^2 + \mathcal{B}_1\alpha_n + \mathcal{B}_0) \tag{24}$$

~~in which the constants are given by:~~ **with the constants A_0, ..., A_2, B_2 in appendix 7.2 *OK for Bernhard***

3.2.3 Nonlinear field update based on the domain equation

~~2.6. Nonlinear field update based on the domain equation~~

Initially, the subsurface properties are unknown and the very first inversion is based on the assumption that wavefield propagation occurs in smooth non-scattering background medium only. Using an approximate smooth property model of the subsurface immediately tells us that the wavefield propagation in such an approximate medium cannot be ~~exact~~ **accurate *OK for Bernhard*** and that for ~~exact~~ **accurate *OK for Bernhard*** results we should make the wavefields consistent with the currently best estimate of the true medium. For this reason we assume $p_{tot} \approx p_0$.

Since the output of linear full waveform inversion is a property model, we do not only obtain structural information but also a "first order" **BR: open interpretation of "first order" but I am okay with it STM: Added quotation marks** approximation of the subsurface properties at every grid point in the inversion domain. We can now use this property model to update the total field by solving the domain equation with in principle any suitable numerical method. Therefore, we iteratively build up the total fields as a sum of the background field and a number of basis functions:

$$\phi_n(\mathbf{x}_r, \mathbf{x}_s, \omega) = \int_{\vec{x} \in D} \mathcal{G}(\mathbf{x}_r, \mathbf{x}_s, \omega) \partial \mathcal{W}_n d\vec{x}' \tag{25}$$

And the incremental contrast sources $\partial \mathcal{W}_n$ are given by:

$$\partial \mathcal{W}_1 = \chi^{(1)} p_{tot}^0, \mathcal{W}_n = \chi^{(n)} p_{tot}^{n-1} - \chi^{(n-1)} p_{tot}^{n-2}, n > 1. \tag{26}$$

The weighting factors are then determined and p_{tot} is updated according to the equation:

$$p_{tot}^{(N)}(\mathbf{x}_r, \mathbf{x}_s, \omega) = p_0(\mathbf{x}_r, \mathbf{x}_s, \omega) + \sum_{N=1}^{N} \alpha_n^{(N)}(\mathbf{x}_s, \omega) \phi_n(\mathbf{x}_r, \mathbf{x}_s, \omega) \tag{27}$$

4 Implementation

~~4 Code Algorithm~~

STM: This section sounded more like the manual for each individual module of the code.

BR: Agree, chapter 3 is a subsection of 4?

STM: Yes it is, will be implemented as such. New introductory text:

The current code implementation for ALTEN can be found on `http://amcm.pcz.pl/get.php?article=2012_1/art_06.pdf`, with the host at `https://git.alten.nl/`

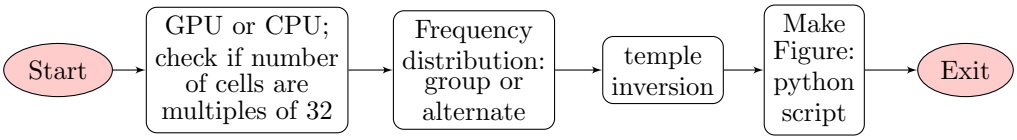

Figure 3: High Level activity diagram for main.cpp

parallelized-fwi.git. The Code Algorithm is complicated and can be divided into the mathematical functions and the computational functions.
For the mathematical functions, please refer to the `FWI.vpp`. It explains the `main.cpp` and the classes defined for this project. The main function basically creates the objects required for the inversion and then called the inversion `cpu.h` function which performs the main tasks. A high level representation of the `main.cpp` can be seen in Figure 3.

4.1 Input parameters

STM: I always use the "courier" / "texttt" font to highlight real code stuff among text. I find it extremely pleasant to look at and see other people do it too. It distinguishes code from text and I did it here.
BR: Agree, and you created the "code" macro in Latex I hope?
STM: Why would I want to do that?
The input parameters can be defined by running the `GUI.py` file. It generates a GUI where input parameters can be set. The inputs for the program **with the default values between parentheses** are as follows. **STM: WARNING: nFreq appears to be number of frequencies per processor. Sigi will tackle this, it's not good that a variable specifically introduced for the parallelization has this kind of name, should be nFreq_per_proc or sth..**:

1. `int nxt (64)`: No. of cells in x (needs to be a multiple of 32 for GPU)
2. `int nzt (32)`: No. of cells in y (needs to be a multiple of 32 for GPU)
3. `int nSrct (17)`: No. of sources and receivers
4. `int nFreq_Total (15)`: No. of Frequencies
5. `int calc_alpha (0)`: 1 to calculate alpha in CG scheme, 0 otherwise
6. `int n_max (5)`: Max outer loop iterations
7. `int n_iter1 (50)`: Max iterations for conjugate gradient scheme
8. `int n_iter2 (100)`: Max p_total iterations
9. `int do_reg (1)`: Perform the multiplicative Regularization **if?**
10. `double F_min1 (10)`: Minimum Frequency
11. `double F_max1 (40)`: Maximum Frequency
12. `int freq_dist_group (1)`: 1 for group distribution of frequency
0 for alternate distribution
13. `int nFreq_input ({1})`: number of frequencies per processor
14. `int gpu (0)`: 1 for GPU, otherwise 0 for CPU

4.2 Subroutines

4.2.1 `main.cpp`

A high level representation of the `main.cpp` can be viewed in the `FWI.vpp` as a UML diagram.

The `main.cpp` initialises the MPI functions responsible for parallel computations. Frequency distribution group and alternate are determined here and depending on the user input, the objects are created. Further, the main function creates the objects for the temple inversion and calls the appropriate functions as seen in FWI_temple.

4.2.2 `temple_inversion`

The main part of the code can be seen in the `temple_inversion`. Over here, the Green's functions and created and the mathematical model detailed in section 2 is executed. A brief explanation of the functions can be seen as follows:

4.2.2.1 `createGreens()`

This function creates an array of Green's functions of the Helmholtz equations. It is called from the Inversion class and refers to the equation 7

4.2.2.2 `CreateP0()`

p_0 is the field calculated with the known contrast ~~and done to the~~ **and follows the** explanation given in section **3.2.3** ~~2.6~~.

4.2.2.3 `createTotalField()`

After the initial approximation, first iterative p_{tot} is calculated here. This is done by calling the function calcfield(). In this function, the equations defined in section ~~2.5~~ 3.2.2 are calculated. At first the incremental contrast sources are defined according to equation 26 and then the ϕ_n is constructed. Based on this 27 is constructed. This step concludes the forward modelling step.

Based on the input parameters, ~~alpha~~α is used if ~~alpha~~α is 1 and is not used if it is set to 0.

4.2.2.4 `calculateData()`

Once the total field p_{tot} is obtained, the p_{data} is calculated according to 1.

4.2.2.5 `Reconstruct()`

The reconstruct function might be the most important function of the code. The code runs in an iterative loop to minimise the residuals. According to tolerance values determined by the user and the max number of iterations, the residuals are minimised in order to obtain p_{data}. Once the residuals are minimised below the tolerance level, the final model is obtained.

The Conjugate Gradient scheme detailed in section ~~2.4~~**3.2.1** is coded here.

4.2.2.6 `MakeFigure()`

Finally, the `MakeFigure()` function is called which calls a python script to generate the figures.

4.3 Parallelization: from a single CPU to Multiple and GPU

STM: Feels like this section, and the way we refer to it, is "Parallellization of the Computations: from a single CPU to multiple." BR: OK, it is also the first time the code is referred. Does this always remain relevant? If so, incldue a reference to the the source (redmine repository + commit host maybe)

The message passing interface (MPI) is a standardized means of exchanging messages between multiple computers running a parallel program across distributed memory.
In parallel computing, multiple computers - or even multiple processor cores within the same computer - are called nodes. Each node in the parallel arrangement typically works on a portion of the overall computing problem. The challenge then is to synchronize the actions of each parallel node, exchange data between nodes and provide command and control over the entire parallel cluster. The message passing interface defines a standard suite of functions for these tasks.
For the FWI, the user sets in the input parameters, if GPU or CPU should be used. This is already checked in the beginning of the code and the appropriate functions corresponding to each are used. Further, the user also sets whether the frequency distribution should be group or alternate. This means that instead of splitting calculations over frequency chunks, split them alternatively over frequency for a better load balancing.

5 Donwloading, Installing and Running the Code

4.//How/to/use/the/FWI

This section details a step by step process to use the FWI.
The development tools that need to be installed are:

1. `sudo apt-get install git` \\For checking out the repositories
2. `sudo apt-get install qt5-default` \\For QtCreator and qmake reuired for building
3. `sudo apt-get install g++` \\Required for compiling
4. `sudo apt-get install mpich` \\Needed during Qmake

For the OpenCL:

1. `sudo apt-get install opencl-headers`
2. `sudo apt-get install ocl-icd-libopencl1`
3. `sudo apt-get install ocl-icd-opencl-dev`

For the Eigen libraries:

1. `sudo apt-get install libeigen3-dev`

For the MakeFigure function and GUI:

1. `sudo apt-get install python2.7-dev`
2. `sudo apt-get install python2.7`
3. `sudo apt-get install python-tk`

4. `sudo apt-get install python-numpy`

5. `sudo apt-get install python-matplotlib`

Next, the repository needs to be checked out:

```
git clone -o redmine https://git.alten.nl/parallelized-fwi.git
git checkout docs-libraires-getting-started
```

Further, to build the project, first create a folder titled build-try_qt-Desktop-Release and then:

```
mkdir Build
cd Build
qmake ../
make -j4
try_qt.pro
```

6 Results and Analysis - example: Temple Reservoir

The example chosen for the FWI code is the logo of the Delphi ~~reseral~~**research** consortium. The logo includes a Greek temple consisting of four major pillars under a triangularly shaped roof, as shown in Figure 4. Additionally we added a layer below the temple to represent the basement. The whole structure is embedded in a homogeneous background ~~with a~~ **which carries waves at a** ~~constant~~ **uniform** ***OK for Bernhard*** acoustic velocity of $c_0 = 2000m/s$. **The temple material's carriage speed is** $c = 2218m/s$. **In the original data in `temple.txt`, you can see a list of zeros and `1.869...e-01`. The zeros are the background's contrast "with itself" and the 1.869...e-01 the result of** $1 - (\frac{2000}{2218})^2$. The Results obtained from running the FWI code can be seen in the following figures: **STM: I would personally explain well which specific settings yield to the particular results presented here. BR: There are not that many settings right? These are iterations of the algorithm STM: That is not what I meant: this is a "run" for which a number of choices are made: you will want to be sure to document these choices carefully. I do not see here the number of sources and receivers for example, although they are clearly parameters to this run.**

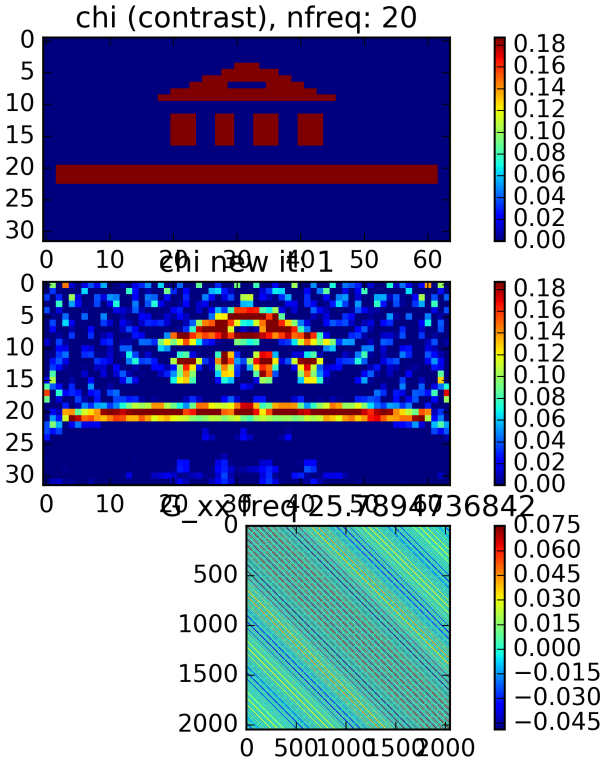

Figure 4: Chi estimation Result 1

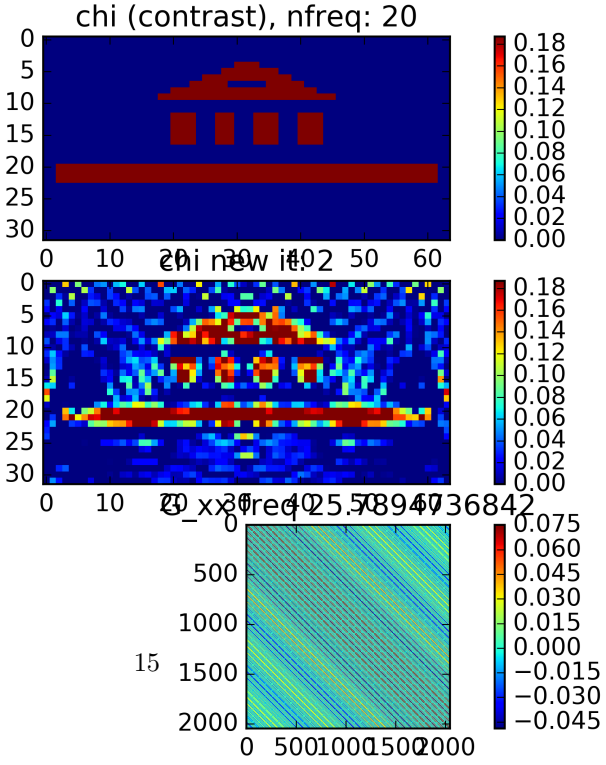

Figure 5: Chi estimation Result 2

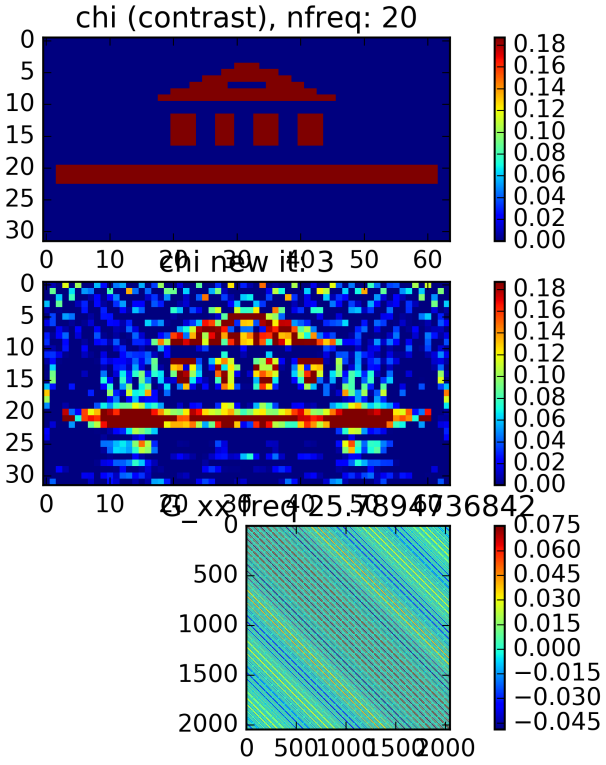

Figure 6: Chi estimation Result 3

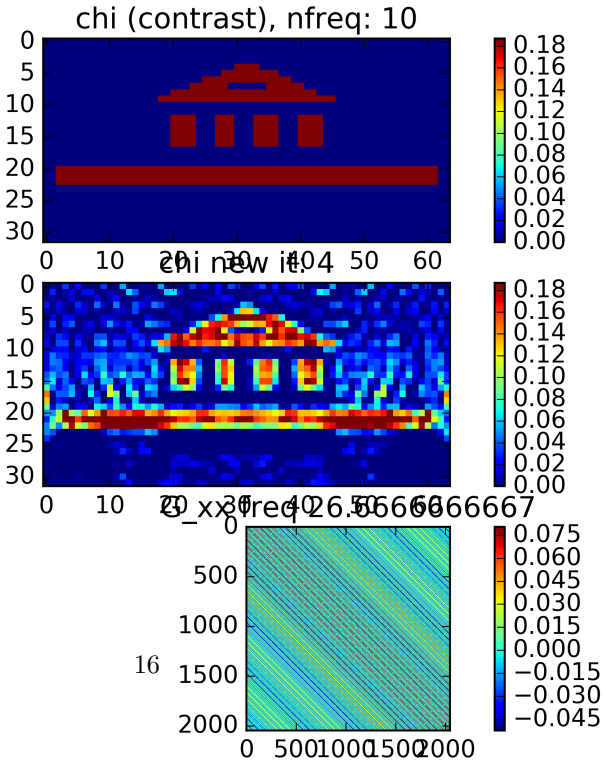

Figure 7: Chi estimation Result 4

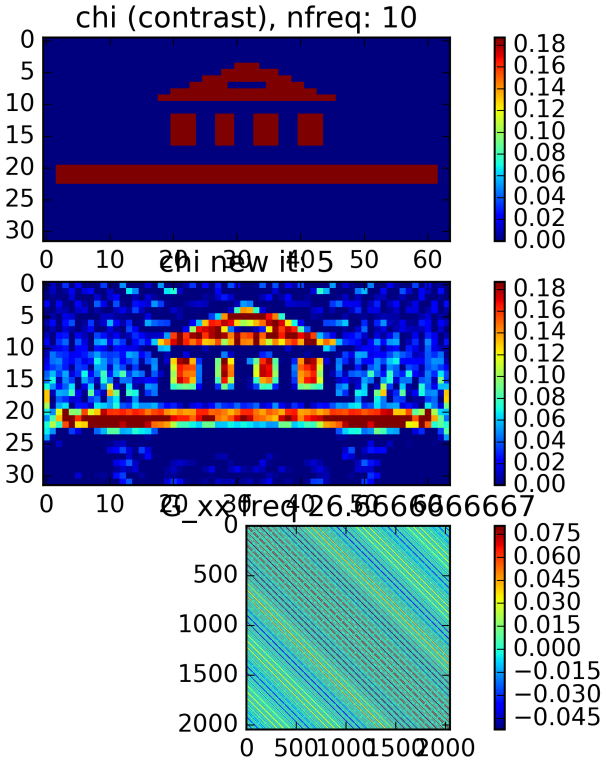

Figure 8: Chi estimation Result 5

7 Appendices

7.1 Derivation of the functional derivative of the error functional

$$
\begin{aligned}
\frac{\partial F(\chi,\mathbf{d})}{\partial \chi} &= \lim_{\epsilon\to 0}\frac{F(\chi+\epsilon\mathbf{d})-F(\chi)}{\epsilon}\\
&= \lim_{\epsilon\to 0}\frac{\eta}{\epsilon}\int \left(p_{\text{data}}-[\mathcal{K}_\chi]-\epsilon\,[\mathcal{K}_\mathbf{d}]\right)\left(p_{\text{data}}-[\mathcal{K}_\chi]-\epsilon\,[\mathcal{K}_\mathbf{d}]\right)^\dagger\\
&- \left(p_{\text{data}}-[\mathcal{K}_\chi]\right)\left(p_{\text{data}}-[\mathcal{K}_\chi]\right)^\dagger\,\mathrm{d}\mathbf{x}_\mathrm{s}\,\mathrm{d}\mathbf{x}_\mathrm{r}\,\mathrm{d}\omega\\
&= -\lim_{\epsilon\to 0}\frac{\eta}{\epsilon}\int \epsilon\left[[\mathcal{K}_\mathbf{d}]\left(p_{\text{data}}-[\mathcal{K}_\chi]\right)^\dagger+[\mathcal{K}_\mathbf{d}]^\dagger\left(p_{\text{data}}-[\mathcal{K}_\chi]\right)\right]\\
&- \epsilon^2\,[\mathcal{K}_\mathbf{d}]\,[\mathcal{K}_\mathbf{d}]^\dagger\,\mathrm{d}\mathbf{x}_\mathrm{s}\,\mathrm{d}\mathbf{x}_\mathrm{r}\,\mathrm{d}\omega\\
&= -2\eta\int \mathrm{Re}\left\{[\mathcal{K}_\mathbf{d}]\left(p_{\text{data}}-[\mathcal{K}_\chi]\right)^\dagger\right\}\,\mathrm{d}\mathbf{x}_\mathrm{s}\,\mathrm{d}\mathbf{x}_\mathrm{r}\,\mathrm{d}\omega\\
&= -2\eta\int \mathrm{Re}\left\{[\mathcal{K}_\mathbf{d}]\,(\mathbf{x}_\mathrm{r},\mathbf{x}_\mathrm{s},\omega)^\dagger r(\mathbf{x}_\mathrm{r},\mathbf{x}_\mathrm{s},\omega)\right\}\,\mathrm{d}\mathbf{x}_\mathrm{s}\,\mathrm{d}\mathbf{x}_\mathrm{r}\,\mathrm{d}\omega
\end{aligned}
$$

7.2 Prefactors of the Total Cost Equation

$$\mathcal{A}_2 = \eta\int\int\int \mid \mathcal{K}\zeta_n \mid^2 d\vec{x_s}d\vec{x_r}d\omega \tag{28}$$

$$\mathcal{A}_1 = -2\eta\mathrm{Re}\left\{\int\int\int r^\star_{n-1} \mid \mathcal{K}\zeta_n \mid d\vec{x_s}d\vec{x_r}d\omega\right\},\hat{E} \tag{29}$$

$$\mathcal{A}_0 = \eta\int\int\int \mid r_{n-1} \mid^2 d\vec{x_s}d\vec{x_r}d\omega = \mathcal{F}^{data}_{n-1},\hat{E} \tag{30}$$

$$\mathcal{B}_2 = \mid\mid b_n\nabla\zeta_n \mid\mid^2_D,\hat{E} \tag{31}$$

$$\mathcal{B}_1 = 2 < b_n\nabla\chi^{n-1}, b_n\nabla\zeta_n >_D,\hat{E} \tag{32}$$

$$\mathcal{B}_0 = \mid\mid b_n\nabla\chi^{n-1} \mid\mid^2_D + \delta^2_{n-1} \mid\mid b_n \mid\mid^2_D \tag{33}$$