Imperial College London

Department of Earth Sciences and Engineering

TIMELAPSE FULL-WAVEFORM INVERSION ON THE FULLWAVE3D PROGRAMME WITH A ROBUST QUALITY-CONTROLLED WORKFLOW

Master thesis in Applied Computational Sciences and Engineering (ACSE-9)
28^{th} *August*, 2019

Deborah Pelacani Cruz

https://github.com/dekape
dp4018@ic.ac.uk

Supervisors:

Pr. Mike Warner
Imperial College London
South Kensington, London
SW7 2AZ

Dr. Christophe Ramananjaona
Geoscience Research Centre
Total E&P UK Ltd, Aberdeen
AB12 3LG

This technical report has been submitted in partial fulfilment of the requirements for the degree of Master of Science in Applied Computational Sciences and Engineering, and has been typeset to match the style of a *Geophysics* paper.

Timelapse Full-Waveform Inversion on the *Fullwave3D* Programme with a Robust Quality-Controlled Workflow

Deborah Pelacani Cruz *

ABSTRACT

In the oil and gas sector, timelapse techniques have been established as an effective tool to characterise changes in pore fluid properties as hydrocarbons are extracted from the subsurface. To quantify these changes, these techniques are commonly combined with conventional full-waveform inversion (FWI) to integrate two or more surveys carried out over the same geological region. The high-level objective of this project is to review and implement pre-established timelapse techniques into Fullwave3D, an FWI programme that is not primarily designed for timelapse applications, such that the recovered velocity models show accurate and stable differences between a baseline and monitor survey, before and after production. We thoroughly assess the most adequate approach for implementing three well-known timelapse techniques within Fullwave3D, named parallel, sequential and double-difference, with no code modification. We conclude that these techniques produce stable results only when the residuals from the baseline and monitor inversion are equivalent, and devise an improved version of sequential method that increases the likelihood of this happening. We show that initial inversion constraints are favourable to guarantee this condition and that the double-difference method provides a theoretical way of ensuring it. So as to adapt to the intricate UI of Fullwave3D, we develop a tailored diagnostic *Python* package, FullwaveQC, and utilise it to identify and remove the causes of spurious inversion features throughout the models. We finalise by proposing an integrated workflow of Fullwave3D and FullwaveQC that ensures accurate inversions and stable timelapse results.

Keywords: FWI, Timelapse, Quality-Control, Fullwave3D, FullwaveQC

* Imperial College London, Department of Earth Sciences and Engineering, South Kensington campus, London, U.K. E-mail: dp4018@ic.ac.uk

INTRODUCTION

Reflection seismology is commonly used as the main method to image the subsurface and explore hidden geological structures for onshore and offshore settings. The method consists of propagating an acoustic or elastic signal through the subsurface and recording the reflected signals sampled over spatial and temporal domains, generating subsets of seismic data. The inversion of seismic data enables the recovery of specific subsurface properties that are useful in the identification and characterisation of geological features, such as hydrocarbon reservoirs or aquifers.

Full-waveform inversion (FWI) is a local tomography inversion technique capable of recovering high-resolution and high-fidelity subsurface models from seismic data. For the oil and gas sector it has be shown to be a successful method in reservoir characterisation through the recovery of pressure and shear wave velocities in the subsurface (Sirgue et al., 2009; Ratcliffe et al., 2011; Guasch et al., 2012). During production optimisation, however, it is critical that not only the subsurface is characterised, but also the changes in fluid contents as the hydrocarbons are removed. For this reason, the concept of timelapse FWI was introduced and has become an extensive area of research (Yang et al., 2015; Maharramov and Biondi, 2014, 2015; Maharramov et al., 2016). Overall, timelapse techniques aim to integrate two or more surveys carried out over the same geological region in order to understand and quantify its evolution.

As an iterative optimisation method, FWI starts from an initial model estimate and then seeks to match its predicted seismic waveforms, trace by trace, with an observed field dataset (Warner et al., 2013b), aiming to improve the fit accuracy over each iteration. The theory of FWI has been well-established since the 1980's (Tarantola, 1984), but the rising need of 3D and 4D modelling have lead to a slow acceptance of the method in terms of efficiency, robustness and applicability.

The high-level objective of this project is to review and implement three pre-established timelapse techniques into an acoustic FWI setting, such that the recovered P-wave velocity models show accurate and stable differences between a baseline

and a monitor survey. These timelapse techniques are herein implemented in the Fullwave3D programme (Warner et al., 2013b) in order to have their impacts measured during FWI. Although these techniques are tested on 2D datasets, the underlying inversion and timelapse principles are consistent and can straight-forwardly be extended to 3D datasets.

This project has been carried out at Total E&P UK Geoscience Research Centre in Aberdeen. A synthetic timelapse dataset has been utilised to test timelapse techniques that are commonly used under the company's current practices. The novelty of the project lies in implementing and assessing these techniques in the newly acquired Fullwave3D programme, which is unfamiliar to the company, and which, at first glance, seems unsuitable for timelapse implementation. To address the high-level objective, the following low-level objectives are set:

1. To help Total become familiar with a new intricate FWI programme;

2. To apply Fullwave3D to a synthetic timelapse dataset with no code modification, and assess its performance in a set of elementary timelapse configurations;

3. To develop a stand-alone diagnostic *Python* package targeted to Fullwave3D as to help new users of the code with implementation and quality-control;

4. To utilise this package to identify and remove, where possible, the causes of spurious features observed in the inversion models;

5. To recommend a robust quality-controlled inversion workflow to be applied co-jointly to Fullwave3D and the diagnostic package; and

6. To evaluate the produced timelapse results and recommend how Fullwave3D might be productively improved to increase its accuracy in timelapse applications.

The first section of this report provides a literature revision of FWI and timelapse theories. The second section introduces the synthetic model used throughout this project and how Total's internal tools were used to generate the synthetic seismic shots used as the observed dataset. The third and fourth sections explore the relevant features of Fullwave3D and of the developed quality-check package FullwaveQC, along with the development methodology of the latter. The fifth section introduces the inversion strategy and inversion parameters chosen for the particular dataset. The sixth section presents the main results and findings, discussing the challenges of performing accurate inversions with the Fullwave3D programme, and how FullwaveQC was used to identify anomalous inversion features. It then proposes a sustainable workflow for the implementation of a quality-controlled timelapse inversion using Fullwave3D and FullwaveQC. The seventh section discusses the limitations and further work for this project, in performing accurate timelapse inversions with Fullwave3D, and in expanding and improving the functionalities of the FullwaveQC package.

THEORY

Full Waveform Inversion

FWI as a Local Inversion Problem

FWI consists of a sequence of locally linearised inversions and iterative updates of a starting model based on the minimisation of an objective function. It starts by considering the following forward problem:

$$\mathbf{G}(\mathbf{m}) = \mathbf{p} \quad , \tag{1}$$

where $\mathbf{m}$ is a column vector containing the physical parameters of a subsurface model, and $\mathbf{p}$, also a column vector, contains the predicted outcome of propagating an acoustic or elastic source signal through all points of the model. The operator $\mathbf{G}$ is a representation of the physics acting on that subsurface. In the FWI case, $\mathbf{G}$ is a spatial and temporal projection of the wave equation that will propagate and disturb a source wavelet through the model and produce a seismic wave response.

Ideally, the true subsurface model $\mathbf{m}'$ could be obtained from an observed seismic wavefield $\mathbf{d}$ by solving the following inverse problem:

$$\mathbf{m}' = \mathbf{G}^{-1}(\mathbf{d}) \tag{2}$$

There are several problems associated with this inverse formulation: 1) the observed data $\mathbf{d}$ is usually inexact or incomplete, 2) $\mathbf{d}$ is unlikely to be noise free, 3) if $\mathbf{G}$ is a linear operator, its inverse is only well defined if the problem is equidimensional, otherwise the solution is non-exact and/or non-unique, and 4) the problem is likely to be ill-posed such that small variations in the data may cause spurious variations in the model. The main problem associated with the inverse formulation is that $\mathbf{G}$ is a non-linear operator with an inverse that is not analytically defined (Warner et al., 2013b). This means that Equation (2) cannot be directly solved and iterative methods are required instead. The following steps of mathematical formulations are aimed to show how iterative methods can transform the above non-linear problem into a series of linear approximations that converge to an optimal solution.

Starting from an initial model $\mathbf{m_0}$, new models are generated by adding to it a perturbation $\delta\mathbf{m}$ such that the new models are progressively closer to the true model. We assume this model update takes the linear form:

$$\mathbf{m} = \mathbf{m_0} + \delta\mathbf{m} \tag{3}$$

In FWI, we seek to find the model $\mathbf{m}$ that produces a predicted field $\mathbf{p}$ as close as possible to the observed data $\mathbf{d}$. The term *'as close as possible'* can be mathematically represented in several ways, but it should be described by a scalar objective function that evaluates to zero when the predicted and observed data are identical. FWI can then make use of optimisation techniques in order to minimise this objective function and solve the inverse problem.

In traditional FWI, the chosen objective function is composed of the L2-norm of the residual data, i.e. the L2-norm of the difference between the predicted and observed data. It is usually combined with some method of regularisation, which serves the purpose of improving the convergence, stability and roughness of the recovered models, or providing it with extra

physical constraints. Although sophisticated FWI algorithms make use of elaborate objective functions, for the sake of simplicity this introductory formulation is carried out with an L2-norm objective function with no regularisation:

$$\begin{aligned} f(\mathbf{m}) &= \frac{1}{2}||\mathbf{G}(\mathbf{m}) - \mathbf{d}||_2^2 \equiv \frac{1}{2}||\mathbf{p} - \mathbf{d}||_2^2 \\ &\equiv \frac{1}{2}\sum_{s=1}^{n_s}\sum_{r=1}^{n_r}\sum_{t=1}^{n_t}||p_{s,r,t} - d_{s,r,t}||_2^2 \quad , \end{aligned} \tag{4}$$

where n_s, n_r, and n_t are the number of shots, receivers and time samples, respectively, for a given survey.

The objective function f is minimised when its gradient with respect to the model is zero:

$$\nabla_{\mathbf{m}} f(\mathbf{m}) \equiv \frac{\partial f}{\partial \mathbf{m}} \equiv \begin{bmatrix} \frac{\partial f}{\partial m_0} \\ \frac{\partial f}{\partial m_1} \\ \vdots \\ \frac{\partial f}{\partial m_n} \end{bmatrix} = \mathbf{0} \tag{5}$$

If $f(\mathbf{m})$ is expanded as a multi-variable Taylor series and minimised with respect to the model, then it becomes trivial to calculate the perturbation $\delta\mathbf{m}$ that would take us from the initial to the optimal model. For this derivation it must be noted that 1) as inferred from Equation (3), differentiating with respect to $\mathbf{m}$ is the same as differentiating with respect to $\delta\mathbf{m}$ as $\mathbf{m_0}$ is a constant, and that 2) f, $\partial f/\partial \mathbf{m}$, and $\partial^2 f/\partial \mathbf{m}^2$, evaluated at $\mathbf{m} = \mathbf{m_0}$ do not depend upon $\delta\mathbf{m}$. The expansions then takes the following form:

$$\begin{aligned} f(\mathbf{m}) = f(\mathbf{m_0} + \delta\mathbf{m}) &= f(\mathbf{m_0}) + \delta\mathbf{m}^T \frac{\partial f}{\partial \mathbf{m}}\bigg|_{\mathbf{m}=\mathbf{m_0}} \\ &+ \frac{1}{2}\delta\mathbf{m}^T \frac{\partial^2 f}{\partial \mathbf{m}^2}\bigg|_{\mathbf{m}=\mathbf{m_0}} \delta\mathbf{m} + \mathcal{O}(\delta\mathbf{m}^3) \quad , \end{aligned} \tag{6}$$

which differentiating with respect to $\mathbf{m}$ gives

$$\begin{aligned} \frac{\partial f}{\partial \mathbf{m}}\bigg|_{\mathbf{m}=\mathbf{m_0}+\delta\mathbf{m}} &= \frac{\partial f}{\partial \mathbf{m}}\bigg|_{\mathbf{m}=\mathbf{m_0}} + \left(\frac{1}{2}\delta\mathbf{m}^T \frac{\partial^2 f}{\partial \mathbf{m}^2}\bigg|_{\mathbf{m}=\mathbf{m_0}}\right)^T \\ &+ \left(\frac{1}{2}\frac{\partial^2 f}{\partial \mathbf{m}^2}\bigg|_{\mathbf{m}=\mathbf{m_0}} \delta\mathbf{m}\right) + \frac{\partial}{\partial\, \delta\mathbf{m}}\mathcal{O}(\delta\mathbf{m}^3) \end{aligned} \tag{7}$$

Setting Equation (7) equal to zero and combining the middle terms results in:

$$\frac{\partial f}{\partial \mathbf{m}}\bigg|_{\mathbf{m}=\mathbf{m_0}} + \frac{\partial^2 f}{\partial \mathbf{m}^2}\bigg|_{\mathbf{m}=\mathbf{m_0}} \delta\mathbf{m} + \mathcal{O}(\delta\mathbf{m}^2) = 0 \quad , \tag{8}$$

which in turn can be rearranged for a first order approximation of the model perturbation when the second order terms are neglected:

$$\delta\mathbf{m} \approx -\left(\frac{\partial^2 f}{\partial \mathbf{m}^2}\right)^{-1} \frac{\partial f}{\partial \mathbf{m}} \equiv -\mathbf{H}^{-1}\nabla_{\mathbf{m}} f(\mathbf{m}) \tag{9}$$

Here $\mathbf{H}$ and $\nabla_{\mathbf{m}} f$ are the Hessian and gradient of the function f, respectively, both evaluated at $\mathbf{m_0}$. The Hessian, Equation (10), is a square symmetric matrix containing all the second-order differentials of the objective function f with respect to all combinations of the model parameters. It provides a measure of the local curvature of the n-dimensional error surface of the model $\mathbf{m_0}$ at which it is evaluated (Warner et al., 2012).

$$\mathbf{H} := \frac{\partial^2 f(\mathbf{m})}{\partial \mathbf{m}^2} \equiv \begin{bmatrix} \frac{\partial^2 f}{\partial m_1^2} & \frac{\partial^2 f}{\partial m_1 \partial m_2} & \cdots & \frac{\partial^2 f}{\partial m_1 \partial m_n} \\ \frac{\partial^2 f}{\partial m_2 \partial m_1} & \frac{\partial^2 f}{\partial m_2^2} & \cdots & \frac{\partial^2 f}{\partial m_2 \partial m_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial m_n \partial m_1} & \frac{\partial^2 f}{\partial m_n \partial m_2} & \cdots & \frac{\partial^2 f}{\partial m_n^2} \end{bmatrix} \tag{10}$$

The gradient of f on the other hand represents the direction to which perturb the model in the direction of the minimum. Rough approximations will likely direct the minimisation of the objective function to converge towards local minima. For a robust local inversion, this gradient needs to be calculated as precisely as possible.

Ideally, if the model perturbation could be directly calculated from Equation (9), the updated model from Equation (3) would be the closest-to-true model that could be obtained with a first order approximation. In practice, however, the Hessian is a very large matrix and computing its inverse is usually not within computational reach, or the matrix is not sufficiently well-posed such that its inverse exists and is stable. For this reason, the common approach is to approximate the Hessian with a matrix or scalar whose inverse is well defined and cheaper to compute.

Likewise, for significantly large models, computing the gradient of f by perturbing each model parameter individually becomes computationally unfeasible. The adjoint formulation discussed by Tarantola (1984) computes this gradient in a much cheaper procedure without trading it off for accuracy provided the formulation is numerically stable. This formulation defines the method of FWI and is discussed in sections below.

The combination of Equations (3) and (9) demonstrates how we have linearised a non-linear inversion problem with a Born approximation (Born, 1926). The model perturbation needed to correct the initial model is approximated as linear product of the objective function and linear inversion methods can be applied to calculate it. If we iteratively repeat the steps of Equations (3) and (9), and succeed at finding the direction to the global minimum, each step should then progressively provide a closer-to-true model, which for an L2-norm minimisation also represents the maximum likelihood model estimation. This series of iterative inversion steps form the core of local inversion methods for non-linear problems and are summarised in the diagram of Figure 1.

Approximating the Hessian and the Step Length

In practice, an accurate approximation of the Hessian will result in an accurate computation of the model perturbation. However, since the local inversion method is based on an it-

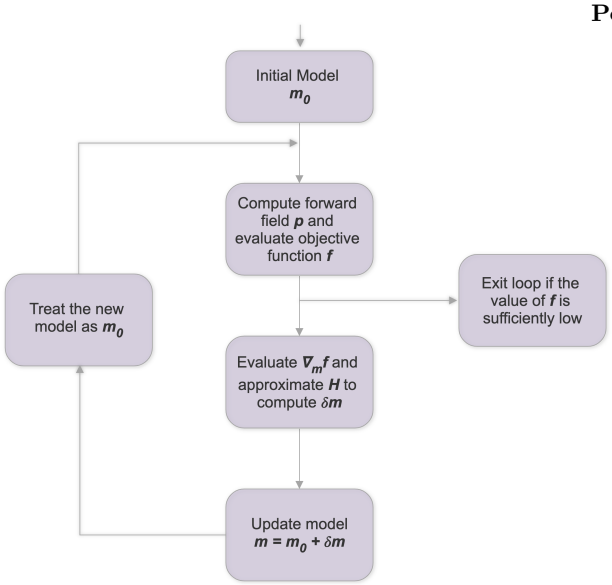


Figure 1: Diagram of a local inversion workflow for the implementation of FWI. The forward wavefield $\mathbf{p}$ is computed on the model using a source wavelet and a computational implementation of Equation (1).

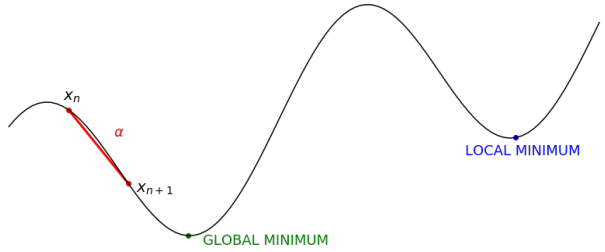


Figure 2: Schematisation of the role of the step length during optimisation. While the gradient of the function f evaluated at the model space x_n defines the direction of descent, the step length α defines by how much, such that $f(x_{n+1}) = f(x_n) - \alpha_n \frac{\partial f}{\partial x_n}$.

erative workflow, an accurate estimation of the Hessian is not required and is likely to exhaust computational resources. Approximating the Hessian therefore can be as trivial as approximating it by a constant. This method is commonly referred as the method of steepest descent, to which Equation (9) takes the following form:

$$\delta \mathbf{m} \approx -\alpha \nabla_{\mathbf{m}} f(\mathbf{m}) \tag{11}$$

Another way of approximating the Hessian is by purely preserving its diagonal elements, which then results in a diagonal matrix that is trivial to invert (Warner et al., 2013b):

$$\delta m_i \approx -\alpha \frac{1}{H_{ii}} \frac{\partial f}{\partial m_i} \tag{12}$$

Other common methods of approximating the Hessian to better incorporate its effects include L-BFGS and Conjugate Gradient. These are not described in this report but the underlying approach for their implementation remains the same.

In Equations (11) and (12), α is a global scaling factor, commonly referred to as the step-length. It controls by how much the model descents in the direction established by the first derivative (Figure 2). It is then obvious that an optimal step-length can not only improve the convergence rate of but also prevent the optimisation from converging towards local minima. See Figure 3 for an applied example.

The optimal step length can be estimated by assuming a linear relationship between the model perturbation $\delta \mathbf{m}$ and the data residual $\delta \mathbf{d} \equiv \mathbf{p} - \mathbf{d}$. This leads to a quadratic relationship assumption between the objective functional and the model perturbation, as inferred from Equation (4). This assumption then evolves into a quadratic relationship between the objective function and the step length, which varies linearly with the model perturbation (Equations (11) and (12)).

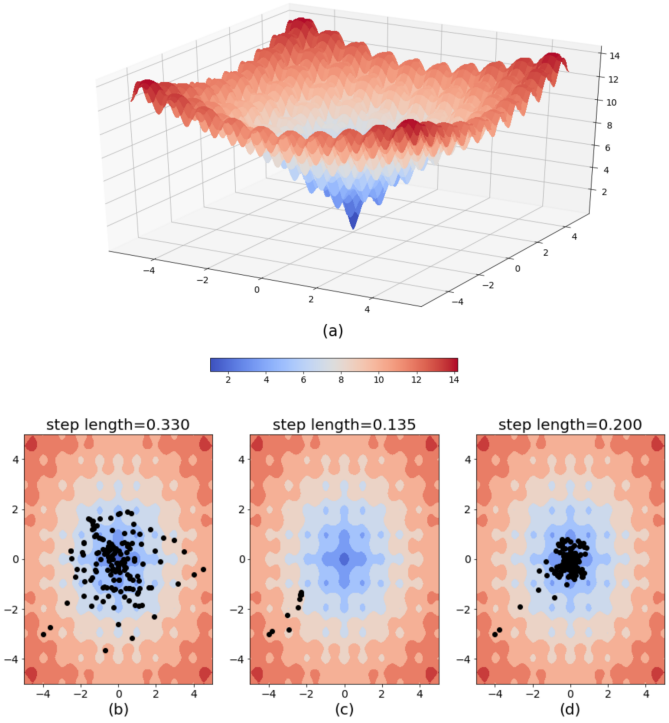


Figure 3: Different step lengths applied during optimisation of the Ackley function (Ackley, 1987), highlighting the importance of choosing an appropriate step length. The Ackley function (a) is a function containing several local minima and a global minimum at (0, 0). Surface plots (b-d) are a bird-eye view of (a). The starting point for the optimisations in (b-d) is at coordinate (-4, -3) and these were performed for 150 iterations using the method of steepest descent. We see from the figures that a large optimisation step length (b) resulting in a slower convergence, a small step length(c) resulting in getting trapped in a local minimum, and the ideal optimal step length (d) reaching the global minimum at a good convergence rate.

By perturbing the model with three distinct α, for a constant initial model and a constant gradient, it is then possible to apply a numerical line search method to estimate the value of α that will minimise the objective function by minimising the quadratic interpolation (Figure 4).

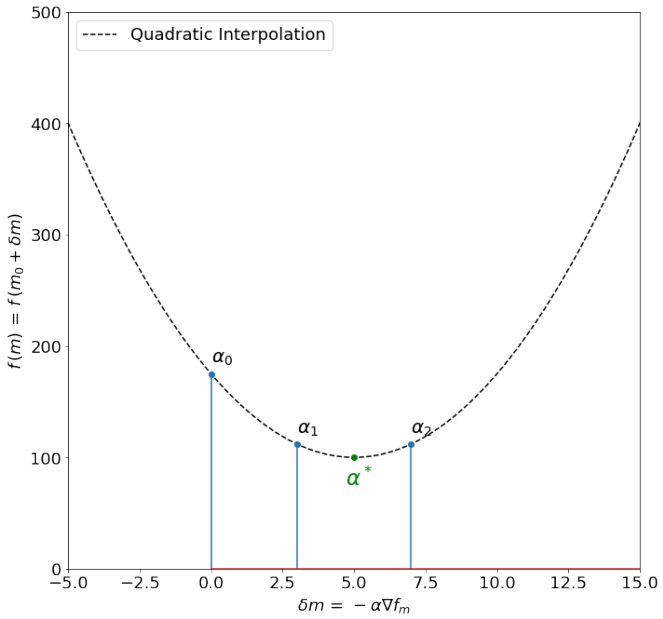


Figure 4: Numerical line search of the optimal step length α^* based on the quadratic relationship assumption between the f and α. For visual purposes, the graph has been simplified to represent a scalar model m.

Other methods to analytically compute the optimal step length exist, as per example the exact line search method, but these are dependent on the stability of the adjoint computation of the gradient.

The Adjoint Method for Computing the Gradient

The adjoint method for computing the gradient $\nabla_{\mathbf{m}} f$ of the objective functional, as described by Tarantola (1984), consists of the following steps:

1. Forward propagation of a source wavefield through the starting model to produce a predicted dataset $\mathbf{p}$,
2. Computation of the residual by subtracting the predicted and observed data for each time sample, receiver and shot,
3. Back propagation of the residual wavefield from the receivers,
4. Cross-correlation of the forward and back propagated wavefields in time to produce gradients at each point within the model's interior,
5. Stacking of the individual gradients to produce one final global gradient.

The following derivation shows why the above series of steps result in the computation of the functional gradient. We start by considering the simplest form of the wave equation propagating through an acoustic, isotropic, inhomogeneous, constant-density fluid medium with no attenuation or dispersion:

$$\frac{1}{c(x)^2}\frac{\partial^2 u(x,t)}{\partial t^2} - \nabla^2 u(x,t) = s(x,t) \quad , \tag{13}$$

where u is the forward propagating wavefield measured as acoustic pressure, s is the driving source that produces that wavefield and c is the spatial projection of pressure wave velocities that defines the model space we wish to recover. It is worth noting that both u and s are functions of space and time, whereas c is a function of space only.

Although Equation (13) provides an oversimplification of the wave theory, it is relatively straight forward to add variables such as density, shear, attenuation, anisotropy and other physical effects to it. It will increase the prediction accuracy with regards to the acquired data, along with the numerical complexity for resolving this partial differential equation, but it will not affect the general approach by which it is solved.

It is then possible to write Equation (13) using a discretisation matrix, and highlight the linear relationship between the wavefield u and source s:

$$\mathbf{A}\mathbf{u} = \mathbf{s} \quad , \tag{14}$$

where $\mathbf{A}$ is a spatial and temporal discretisation of the operator

$$\frac{1}{c^2}\frac{\partial^2}{\partial t^2} - \nabla^2$$

Although $\mathbf{u}$ and $\mathbf{s}$ are linearly related, $\mathbf{A}$ is an operator that contains the projection of velocities in space, i.e. it contains the model. Differentiating Equation (14) with respect to the model then gives:

$$\frac{\partial \mathbf{A}}{\partial \mathbf{m}}\mathbf{u} + \mathbf{A}\frac{\partial \mathbf{u}}{\partial \mathbf{m}} = \frac{\partial \mathbf{s}}{\partial \mathbf{m}} = \mathbf{0} \tag{15}$$

If the propagating wavefield $\mathbf{u}$ is defined for all spaces in the model, then the predicted data can be constructed by extracting the wavefield at the receiver locations, defined by the acquisition geometry of the observed data, using a *restriction matrix* $\mathbf{R}$:

$$\mathbf{p} = \mathbf{R}\mathbf{u} \tag{16}$$

Multiplying Equation (15) by the restriction matrix $\mathbf{R}$, and using the relationship of Equation (16) to replace $\mathbf{u}$ by $\mathbf{p}$, we obtain the following expression:

$$\frac{\partial \mathbf{p}}{\partial \mathbf{m}} = \mathbf{R}\frac{\partial \mathbf{u}}{\partial \mathbf{m}} = -\mathbf{R}\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial \mathbf{m}}\mathbf{u} \quad , \tag{17}$$

which represents the variation of the data with respect to the model. Another formulation for this variation can be obtained within the expansion of Equation (5), such that:

$$\nabla_{\mathbf{m}} f = \frac{\partial f}{\partial \mathbf{m}} = \frac{\partial}{\partial \mathbf{m}}\left(\frac{1}{2}||\mathbf{p}-\mathbf{d}||^2\right) = \frac{\partial}{\partial \mathbf{m}}\left(\frac{1}{2}(\mathbf{p}-\mathbf{d})^T\delta\mathbf{d}\right)$$

$$= \frac{\partial(\mathbf{p}-\mathbf{d})^T}{\partial \mathbf{m}}\delta\mathbf{d} = \left(\frac{\partial \mathbf{p}}{\partial \mathbf{m}}\right)^T\delta\mathbf{d} \tag{18}$$

The final expression for the gradient is then obtained by combining Equations (17) and (18):

$$\nabla_{\mathbf{m}} f = -\mathbf{u}^T \left(\frac{\partial \mathbf{A}}{\partial \mathbf{m}}\right)^T \mathbf{A}^{-T}\mathbf{R}^T \delta \mathbf{d} \tag{19}$$

It is easy and convenient to design the discretisation matrix $\mathbf{A}$ such that it is symmetrical in space, where $\mathbf{A}$ represents the field being propagated forward in time and $\mathbf{A^T}$ represents the field propagating backwards in time. The term $\left(\mathbf{A}^{-1}\right)^T \mathbf{R}^T \delta \mathbf{d}$ can then be understood as a variation of Equation (14) where the residual at the receivers are used as virtual sources:

$$\begin{aligned} \mathbf{A}^T \delta \mathbf{u} &= \mathbf{R}^T \delta \mathbf{d} \\ \delta \mathbf{u} &= \mathbf{A}^{-T} \mathbf{R}^T \delta \mathbf{d} \end{aligned} \tag{20}$$

Therefore, the expression above represents the back propagation of the wavefield generated by the residual of the data and Equation (19) can be further simplified as:

$$\nabla_{\mathbf{m}} f = -\mathbf{u}^T \left(\frac{\partial \mathbf{A}}{\partial \mathbf{m}}\right)^T \delta \mathbf{u} \tag{21}$$

Equation (21) tells us that the gradient of the objective function is given by the zero-lag temporal cross-correlation of the forward propagating wavefield, originated from a particular source, with the back propagating wavefield from the data residuals at each receiver, and weighted by the derivative of $\mathbf{A}$ with respect to the model $\mathbf{m}$ transposed. Even though the adjoint formulation might represent an unintuitive way of computing the gradient, it provides a way of doing so with only two forward models, as opposed to one per model parameter: one for computing the forward wavefield, and another for compute the back-propagated wavefield. It means there is high demand for storage, for storing both wavefields, but it is compensated by a significant decrease in computational time for very large models.

Limitations of FWI

FWI makes use of expensive localised iterative methods to solve inversion problems, and its level of complexity increases with the number of parameters in the model. Only recently the computational resources have become sufficient to perform these iterations on seismic data, specially for producing high-resolution and high-fidelity 3D models that require elaborate objective functions and accurate forms of the wave equation.

The biggest downfall of localised inversion methods for non-convex optimisation is that they risk converging to a local rather than a global minimum, as shown in Figure 3. Ideally, stochastic methods would prevent this from happening, but for large datasets the computational expenses required are still unattainable.

Cycle-skipping has been promoted as the main cause for local minima convergence (Shah et al., 2013) and shown to be relevant to FWI because of the oscillatory nature of seismic data (Beydoun and Tarantola, 2005). As the objective function in FWI is computed through a trace-by-trace comparison for each time sample, a phase shift of over half a period between the observed and predicted traces will immediately deviate the objective function from the correct descent direction (Beydoun and Tarantola, 2005; Pratt, 1999; Warner et al., 2013a)(Figure 5). In this case, the objective function will be minimised but towards a local minimum instead.

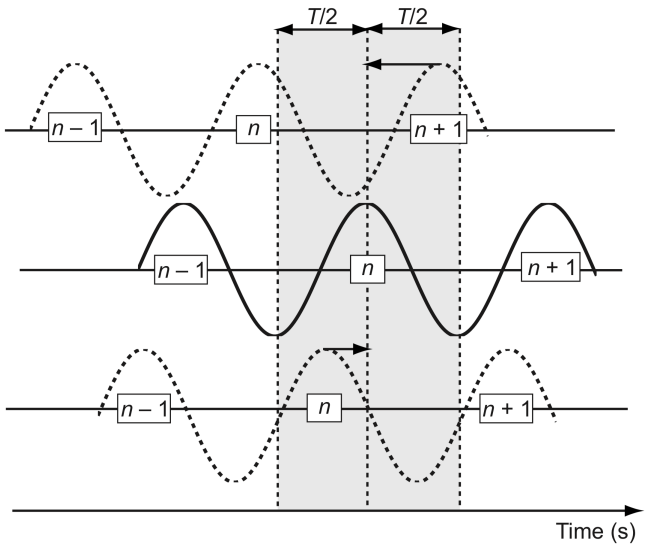


Figure 5: A schematic cycle-skipping diagram from Virieux and Operto (2009) where the dashed lines represent the predicted data and the solid line the observed data. The upper predicted trace is out of phase with the observed trace by more than half a cycle. In this case, the objective function will attempt to match its $n + 1^{th}$ cycle with the observed n^{th} cycle, producing cycle-skipping artefacts. The lower predicted trace is within half a cycle of the observed data and will not result in cycle skipping.

In attempt to mitigate this issue, FWI relies on a starting model that is as smooth and as close-to-true as possible so that the gradient descent converges to the correct minimum. This is often a hard task to achieve when little information is known about the subsurface. A robust method for ensuring adequacy for an FWI starting model and subsequent iterations has been proposed and tested by Shah et al. (2013). By iteratively computing the phase difference between the observed and predicted dataset during FWI it becomes trivial to identify and prevent cycle-skipped artefacts generated by the starting and the consecutively generated models.

From the diagram in Figure 5 it is evident that lower frequencies are less prone to be influenced by cycle-skipping as half a cycle for these frequencies implies a larger time offset. Hence, a common technique to prevent cycle-skipping artefacts is to firstly invert the lower frequencies of the data, and then progress to invert higher frequencies (Shah et al., 2013; Virieux and Operto, 2009; Bunks et al., 1995). The final model of each frequency block is used as the starting model for the next. This technique is commonly referred as frequency continuation and requires careful filtering of both the observed data and the source wavelet. Provided frequency continuation is adequately used as part of the inversion strategy, the starting model is only required to not be cycle skipped at the first usable frequency, then the progressive models are also guaranteed to not be cycle-skipped.

Overview of Timelapse Schemes

The three FWI timelapse techniques implemented in this project are based on the theories of Yang et al. (2015), hereby referred to as parallel, sequential, and double difference waveform inversion (DDWI) techniques. These techniques aim to predict the differences between a baseline and a monitor survey by quantifying the variation $\delta \boldsymbol{V_p} = \boldsymbol{V_{p_m}} - \boldsymbol{V_{p_b}}$ of P-wave velocities at each grid point of the recovered subsurface models.

The parallel and sequential techniques can be straightforwardly implemented with conventional FWI. These are schematised in Figures 6 and 7, respectively. The parallel technique consists of inverting the two survey datasets independently starting from the same initial model. The sequential, consists of firstly inverting the baseline dataset and using its output model as the starting model for the monitor dataset inversion.

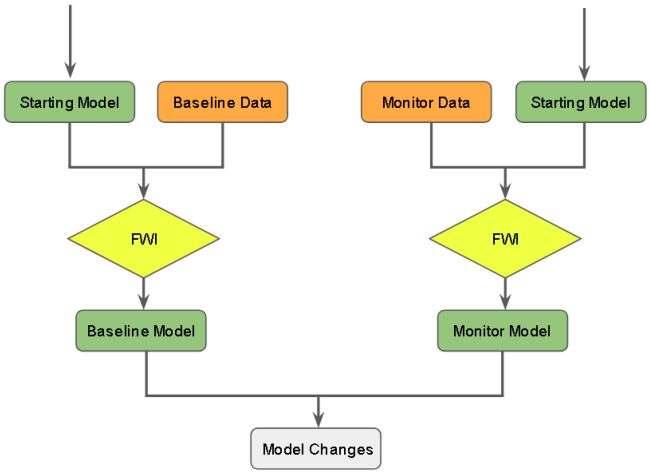


Figure 6: Schematic workflow of parallel timelapse. Adapted from Yang et. al (2015) (Yang et al., 2015).

The DDWI technique also uses the baseline model as a starting model for the monitor inversion, but consists of minimising the following objective function:

$$f_{\mathrm{DDWI}}(\mathbf{m_m}) = \frac{1}{2} \left\| (\mathbf{p}_\mathrm{m} - \mathbf{p}_\mathrm{b}) - (\mathbf{d}_\mathrm{m} - \mathbf{d}_\mathrm{b}) \right\|_2^2 \quad , \qquad (22)$$

where $\mathbf{p}$ and $\mathbf{d}$ are the predicted and observed datasets, respectively, for the baseline and monitor models. Equation (22) can then be rearranged to:

$$f_{\mathrm{DDWI}}(\mathbf{m_m}) = \frac{1}{2} \left\| \mathbf{p}_\mathrm{m} - (\mathbf{d}_\mathrm{m} - \mathbf{d}_\mathrm{b} + \mathbf{p}_\mathrm{b}) \right\|_2^2 \qquad (23)$$

The right term of Equation (23) in parenthesis can be understood as the dataset produced by the a predicted baseline model, added to the observed dataset difference, which, for a perfect baseline model, should recreate the observed monitor dataset. As the baseline model is unlikely to be a perfect match to the true model, minimising Equation (23) introduces a predicted baseline constraint on the monitor inversion such that only the changes between the datasets are accommodated throughout the iterations. Equation (23) can then be rewritten as:

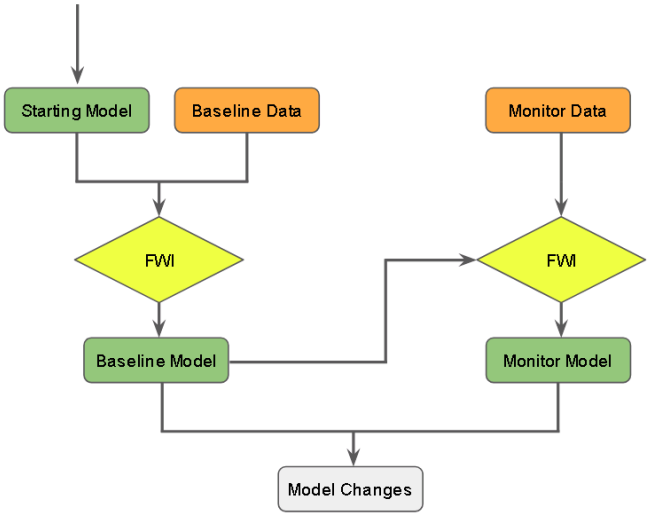


Figure 7: Schematic workflow of sequential timelapse. Adapted from Yang et. al (2015) (Yang et al., 2015).

$$f_{\mathrm{DDWI}}(\delta \mathbf{m}) = \frac{1}{2} \left\| \mathbf{p}_\mathrm{m} \left(\mathbf{m}_\mathrm{b} + \delta \mathbf{m} \right) \right\} - \mathbf{d}_{\mathrm{DDWI}} \|_2^2 \quad , \qquad (24)$$

where $\mathbf{p}_\mathrm{m}$ is the predicted monitor dataset and is a function of the baseline model $\mathbf{m_b}$ added to some model update, and $\mathbf{d}_{\mathrm{DDWI}}$ is the predicted DDWI monitor dataset behaving as the observed monitor dataset. The formulation of Equation (24) represents a standard L2-norm FWI objective function, which can be implemented with regular FWI solvers. However, instead of recovering the monitor model $\mathbf{m_m}$, the formulation recovers the model difference $\delta \mathbf{m_m}$ in the baseline model $\mathbf{m_b}$.

The DDWI scheme has been demonstrated to be successful on perfectly repeated surveys (Yang et al., 2015; Zheng et al., 2011; Zhang and Huang, 2013; Denli and Huang, 2009), and requires minimal manipulation of the input data to be implemented with conventional FWI. Its workflow is presented in Figure 8.

SEAM SYNTHETIC DATASET

The observed baseline and monitor datasets used in this project were generated from a 2D slice of the SEG Advanced Modelling Program (SEAM) 3D timelapse velocity models (Figure 9). The models simulate a realistic layered-strata marine environment with an anticline structure, bounded by 65^o dipping normal faults, serving as a structural trap to a hydrocarbon reservoir. The reservoir is characterised by the presence of both gas and oil, correspondent to the lowest velocities in the model, located to the left and at the top of the anticline, respectively. The baseline and monitor subsurface models, henceforth referred as true models, represent the geological environment before and after production.

The datasets were generated on a 15m resampled grid of the true models using a 1.52 ms modelling time step. It was generated by forward modelling of the wave equation with Total's internal FWI software in a finite-difference stencil of 8^{th} order

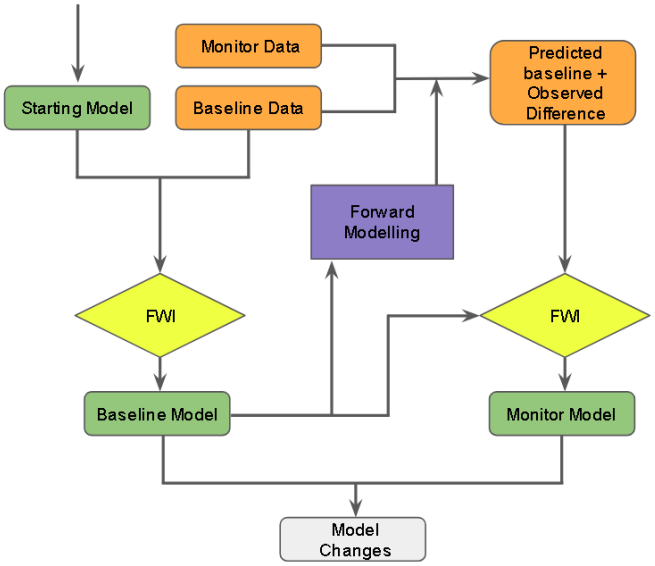


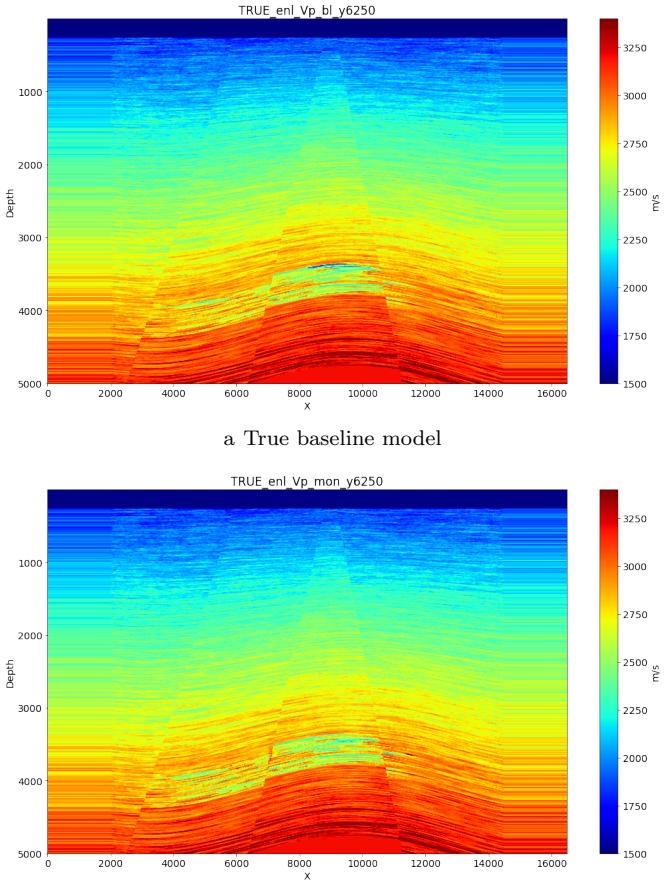
Figure 8: Schematic workflow of Double Difference Waveform Inversion (DDWI) timelapse. Adapted from Yang (2015) (Yang et al., 2015).

in space and 2^{nd} order in time, using an acoustic, isotropic, and constant-density kernel. Noise and distortion factors were deliberately eliminated during the simulation for the simplicity of methodology testing. The data was simulated without free surface multiples or ghosts, no attenuation, internal or external noise, or survey geometry distortion from sea currents. It is not within the scope of this work to test how these field-related artefacts would affect the implementation of timelapse FWI on Fullwave3D.

The acquisition geometry was chosen as to replicate a shallow-water single in-line towed-streamer survey of 10 km in length over the synthetic region. The first shot was recorded at 1 km and the last shot at 14 km on the spatial projection of Figure 9. Overall, 521 shots were recorded throughout the model space, each containing 801 receivers distributed along the streamer, on a 12.5 m regular spacing and positioned 10 m below the water surface. The shots were simulated with an acoustic source positioned 5m below the water surface, at a 25 m separation, with the first receiver positioned at a lateral offset of 190 m (Figure 10). The survey was acquired for 1501 time samples, with a 4 ms sampling rate, using a 15 Hz maximal frequency, 6 Hz peak Ricker wavelet of order 2. The data was filtered using an Ormsby band-pass bandwidth of corner values 1.5 Hz, 2.5 Hz, 15 Hz and 17 Hz.

FULLWAVE3D

Fullwave3D is a programme developed at the Imperial College London that is able to perform robust FWI on acoustic and elastic problems to recover subsurface velocity models. It has the ability to incorporate or invert for anisotropy and can be performed in either time or frequency domains. It is parallelised with MPI in an master-slave distributed memory model, where each seismic shot is individually processed by a



a True baseline model

b True monitor model

Figure 9: The true models used to generate the observed data. The models extend 16km laterally and 5km in depth, modelled at a resolution of 2m x 2m grid cells. The reservoir is characterised by the low velocity event at the 9km lateral distance and 3.5km depth, which correspond to the presence of oil and gas.

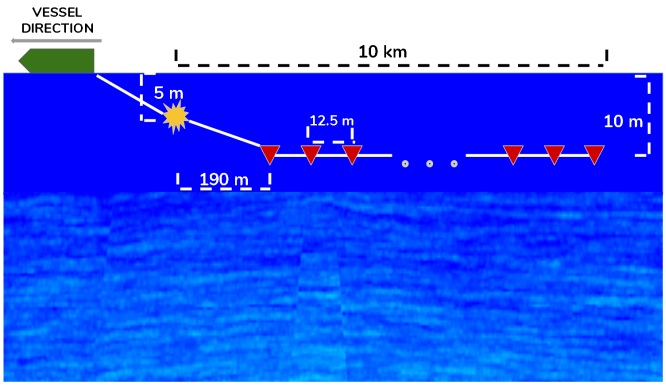


Figure 10: Schematisation of the data acquisition geometry over the synthetic model from Figure 9. The yellow explosion symbol represents the hypothetical positioning of the source and the red triangles the positioning of the hydrophones. The dataset acquired from this synthetic survey over the true models is used throughout this project as the observed dataset. Figure not to scale.

slaver node, and each node is parallelised in a shared memory OpenMP environment.

Fullwave3D was used throughout this study as the main tool for performing FWI for the recovery of P-wave velocity models and testing of timelapse schemes. This section serves to briefly introduce the functionalities of this programme. Although Fullwave3D offers an API with over 100 adjustable parameters, only the ones relevant to this study are discussed. We refer the reader to the Fullwave3D documentation for the programme's full list of functionalities.

Fullwave3D is able to perform FWI using one of the four objective functions, all without regularisation: 1) the standard L2-norm from Equation (30), 2) the L1-norm, 3) the Huber norm, and 4) the Student T norm. These can be modified at each iteration block. It is advised in the documentation that, for close-to-perfect data, the L2-norm will produce the best inversion results. For the consistency of the project, the L2-norm has been chosen as the only objective function throughout the workflow. The approximation of the Hessian in the programme is done through either the method of steepest descent or through conjugate gradient, and the optimal step length is computed as a global factor using the numerical line search method for each inversion iteration.

The Fullwave3D architecture allows for straightforward runs of forward and inversion operations, called synthetic and tomography modes, respectively. The synthetic mode allows for the production of synthetic seismic data given a velocity model, used to compute the residual of the model with respect to an observed dataset during inversion. The tomography mode is the one that performs FWI optimisation and produces the desired model updates. The synthetic mode is also useful for quality checking the initial and finalised models.

The programme requires three main input files for its simplest run: 1) a set of observed data, used to compute the residuals at each iteration, 2) a source wavelet, to be propagated through the model and generate a predicted wavefield, and 3) a velocity model, used as the true model for a synthetic run or as a starting model for a tomography run.

A crucial detail that ensures the success of an FWI run is that the virtual sources and receivers, used to generate the predicted data in the forward runs, must be correctly aligned in the model space. Aligning the model geometry with the data in Fullwave3D is done through a conjoint programme named SegyPrep. SegyPrep is able to obtain the correct geometry setup by analysing headers of SEG-Y files, but also accepts manual inputs in case the headers are incorrectly assembled. SegyPrep generates input files to be read as inputs by the Fullwave3D programme, performing any resampling and source or data delays that are required as pre-processing. Neither programmes require the data to be pre-stacked.

Fullwave3D solves the wave equation using a finite-difference scheme on a rectilinear grid, with order that varies depending on the kernel selected. For this project, the low-order kernel was used with correspondent finite-difference scheme of 4^{th} order in time and 6^{th} order in space. For the numerical stability of this scheme, Fullwave3D requires that the time step is sufficiently short such that a wave travelling at the highest velocity only travels half of the grid spacing. That is equivalent to a CFL condition with a maximum Courant number of 2. For a chosen model grid size of Δx the time sampling Δt required for the CFL condition is:

$$\Delta t \leq \frac{\Delta x}{2V_{max}} \quad , \tag{25}$$

where V_{max} is the maximum velocity of present in the model.

The low kernel also requires a that the shortest wavelength in the model is covered by at least 4 grid cells in order to prevent numerical dispersion and anisotropy. This criterion sets the following upper bound on the allowed model frequencies:

$$f \leq \frac{V_{min}}{4\Delta x} \tag{26}$$

Fullwave3D also has the ability to perform frequency continuation using iteration blocks. Each iteration block performs FWI for a user-defined number of iterations at a specific frequency, to which both the data and the source wavelet are automatically filtered accordingly. The final inverted model of an iteration block is automatically used as the starting model for the next.

SegyPrep has an internal check that guarantees the chosen model spacing and time sampling meet the stability and dispersion criteria. When performing an inversion with Fullwave3D, however, one must ensure that the frequencies in the frequency continuation set up fall below the dispersion criterion from Equation (26).

Fullwave3D deals with boundary conditions by adding extra cells to the outer model space and applying to them an absorbing property to prevent unwanted boundary reflections. It is advised that enough absorbing boundary cells are placed outside the model to cover at least 2 of the longest wavelengths during propagation of the source, and that these boundaries are at least one Fresnel radius away from the source location. The minimum number of boundary cells required can be computed by using the following velocity, wavelength and frequency relationship for non-dispersive media:

$$n_a = 2\,\lambda_{max}\,\frac{1}{\Delta x} \equiv 2\left(\frac{v_{max}}{f_{min}}\right)\frac{1}{\Delta x} \quad , \tag{27}$$

where v_{max} is the maximum propagation speed in the model, f_{min} is the minimum frequency in the source, and Δx is the model resolution in grid cell size.

The distance in grid cells required to guarantee one Fresnel radius can be calculated by:

$$n_f = \frac{1}{2\Delta x}\sqrt{\lambda_{max}\,d} \equiv \frac{1}{2\Delta x}\sqrt{\frac{v_{max}}{f_{min}}\,d} \quad , \tag{28}$$

where d is the lateral spacing between the source and the first receiver in the survey geometry.

FULLWAVEQC

Although Fullwave3D has been extensively used and proven to be a robust tool to perform FWI, the programme still requires an advanced level of expertise to elect the most adequate inversion parameters for a given problem and only provided minimum input checking concerning common issues of cycle skipping, numerical stability and numerical dispersion. These

issues are left for the user to mitigate and this is rarely an obvious task to achieve.

FullwaveQC is a standalone cross-platform *Python* package that was developed throughout this project as quality-check tool to assist inversion runs performed with Fullwave3D. The main motivation behind this package is to provide tools such that an inexperienced user of Fullwave3D can directly identify common modelling problems before and after the inversion workflow. The package is useful to anyone utilising the Rev689 version of Fullwave3D to perform FWI on SEG-Y formatted seismic data. It is open under the MIT license for users to utilise and expand its functionalities.

Software Development Life Cycle

Architectural Design

The capabilties of FullwaveQC as a software were demonstrated and tested throughout the evolution of this project. In this section of the report we present a summary of the architecture of these modules and their particular functionalities. To learn about the full extent and usage of FullwaveQC, the reader is encouraged to explore its documentation in the *GitHub* repository along with the *Jupyter Notebooks* included in the examples folder.

FullwaveQC uses functional programming as its main paradigm and is organised into five modules: *tools*, *visual*, *geom*, *siganalysis* and *inversion*.

The *fullwaveqc.tools* module has two class definitions for loading a SEG-Y file: *SegyData*, corresponding to a seismic data file, and *Model* to a velocity model file. Objects of these two classes should be constructed using the *load* function, to which the user must specify the type of file, *SegyData* or *Model*, to be loaded. It is important to make that distinction between data and model as the *load* function searches for different information depending on the mode given as input. This distinction can be observed in the code snippet below

```
class SegyData:
    def __init__(self, filepath, name=None):
        if name is not None:
            self.name = name
        else:
            self.name =
                filepath.split("/")[-1].split('.')[0]
        self._type = "data"
        self.filepath = filepath
        self.src_pos = -1    # all source positions
        self.rec_pos = -1    # all receiver positions
        self.nsrc = -1       # no of sources
        self.nrec = -1       # no of receivers
        self.src_z = -1      # all source depths
        self.rec_z = -1      # all receiver depths
        self.samples = -1    # no of recorded samples
        self.dt = -1         # sampling rate
        self.dsrc = -1       # source spacing
        self.drec = -1       # receiver spacing
        self.data = None     # trace amplitude data

    def attr(self, attr_name=None):
        v = vars(self)
        if attr_name is not None:
            try:
                v = v[attr_name]
            except KeyError:
                v = None
                warnings.warn("Attribute \"%s\" not
                    found. Returning None" %
                    attr_name)
        return v


class Model:
    def __init__(self, filepath, name=None):
        if name is not None:
            self.name = name
        else:
            self.name =
                filepath.split("/")[-1].split('.')[0]
        self._type = "model"
        self.filepath = filepath
        self.dx = -1          # model resolution
        self.nx = -1          # no of cells inline
        self.nz = -1          # no of cells in depth
        self.minvp = -1       # min model Vp
        self.maxvp = -1       # max model Vp
        self.data = None      # Vp model (nx, nz)

        return

    def attr(self, attr_name=None):
        v = vars(self)
        if attr_name is not None:
            try:
                v = v[attr_name]
            except KeyError:
                v = None
                warnings.warn("Attribute \"%s\" not
                    found. Returning None" %
                    attr_name)
        return v
```

The *load* function reads and structures the data in an arrangement that is required for other functions of FullwaveQC to perform. Some functionalities are designed to work purely with a *Model* object and others purely with a *SegyData* object.

In addition to the *load* function, *fullwaveqc.tools* also provides useful manipulation tools that are used by other modules of the package, as well as other tools that can serve numerous purposes to the user. Features in the module include model smoothing, amplitude normalisation, band-pass filtering, removal of empty traces and automated creation of a double-difference monitor dataset.

The *fullwaveqc.geom* module has two simple functionalities concerning the geometry alignment by SegyPrep. The function *boundarycalc* provides an estimation of the number of absorbing boundary cells needed to avoid undesirable reflections in the forward modelling steps, as well as the number of cells required from a source to the first absorbing boundary in order

to guarantee one Fresnel radius from the source to the first receiver. The *surveygeom* functionality reads and plots the geometry files produced by SegyPrep so that the user can check the interpreted acquisition geometry is within expectations for the particular survey.

The *fullwaveqc.visual* module provides tools to visualise both *Model* and *SegyData* objects. The *amplitude*, *wiggle* and *vpmodel* functions provide standard visualisation of these SEG-Y files and are built as *contourf* plots from the external *matplotlib.pyplot* module. An enhanced utility from the *fullwaveqc.visual* module is to be able to simultaneous visualise two *SegyData* objects through interleaving amplitude plots and interleaving trace wiggles. These utilities are provided in the *interamp* and *interwiggle* functions, respectively. This straightforward visual comparison of two datasets is a simple but beneficial mechanism to identify cycle-skipped traces and mismatched reflections. It is recommended that these functions are used to compare an observed dataset to a predicted dataset produced by Fullwave3D synthetic runs as a visual method of validating a dataset.

In addition, the *fullwaveqc.visual* module also provides functions to produce a V_p log profile of a model and to create an animation of the iterations of an inversion run. The former can be used to visualise the variation of models with depth and to calculate the RMS error between a true and predicted model. The latter can be used to identify spurious trends to the inversion, or anomalous structures that arise from particular iterations and/or frequency blocks.

The *fullwaveqc.inversion* module provides two functionalities to retrieve and plot the global functional and global step length values for every inversion iteration of a tomography project. These are particularly useful to identify increasing functional values within a frequency block and particularly small step length values, which can, in a few occasions, be inferred as a model divergence throughout the inversion workflow.

The *fullwaveqc.siganalysis* module contains functions to assess the quality of a predicted dataset using elaborate signal processing functionalities. The functions *wavespec* and *dataspec*, for example, are able to perform a Fast Fourier Transform (FFT) of a wavelet or data signal and compute its frequency spectrum. In the case of *dataspec*, this calculation is done for an entire seismic shot. The user can pass an *fftsmooth* parameter to these functions, which pads the signal with zeros to increase the number of FFT samples and thus the accuracy and smoothness of the frequency spectrum. For computational efficiency purposes, the spectrum is always calculated with an amount of sampling points that is the closest power of 2 to the number of sampling points specified by the user. This provides a significant decrease in computational time as the FFT is theoretically designed to work faster in this condition.

Another useful feature of the *fullwaveqc.siganalysis* module is the computation of phase differences between two datasets. The *phasediff* function computes this difference for each trace of each shot and returns the results in a source-receiver spatial continuity. As shown by Shah et al. (2013), phase jumps of 360^o in such projection are an indication of cycle skipping. The phase difference calculation can therefore be particularly useful for the identification of cycle skipping of a starting model at the lowest iteration frequency to ensure no cycle-skipping artefacts are propagated to the subsequent iterations.

Lastly, the *fullwaveqc.siganalysis* module provides the *xcorr* function that computes the zero-lag cross-correlation between the predicted and observed datasets in the same continuity as *phasdiff*. This can be useful to quantify the amplitude fit between an observed and a predicted dataset.

Development Methodology

The development of FullwaveQC was not initially part of this project proposal and emerged out of the need of efficient visualisation and quality checking tools after the first attempts of performing FWI on the synthetic data. For this reason, FullwaveQC used the Agile development methodology and its functionalities were created, enhanced and adapted for this particular problem when needed.

Within the Agile methodology, the *fullwaveqc.visual* module was the first module to be developed for the package. The internal visualisation tools provided by Total failed to read and display the outputs of Fullwave3D and lacked any sort of interleaving display feature. The *fullwaveqc.visual* module was therefore necessary to provided more personalised visual capabilities to Fullwave3D users. The *fullwaveqc.inversion* and *fullwaveqc.siganalysis* were developed as an attempt to single out possible cycle-skipping artefacts and local minima convergence, as this is one of the most common problems associated with FWI. The *fullwaveqc.geom* module was created after the identification of incorrect geometry set ups in the initial FWI runs in attempt to easily identify such incorrect set ups before inversion and prevent them from occurring in successive projects. The last module to be developed was *fullwaveqc.tools*, in particular the data structure associated with the *load* function, as a way of organising and connecting the functions available in the other modules.

The usage and development of the packaged occurred simultaneously. As the package was being used, it was also being tested and debugged. Unit tests and continuous integration with TravisCI were also implemented as a way of ensuring the overall sustainability of the code development.

Code Metadata

FullwaveQC was built in a Linux environment as a standalone cross-platform *Python* package using *PyCharm* as IDE. The tools included in the package utilise the *Segyio* 1.8.6 library (Kvalsvik, 2019) to read and write appropriately formatted SEG-Y files. For data displaying and manipulation, it makes use of the standard open sourced libraries *numpy*, *scipy* and *matplotlib*. Other library requirements for full functionalities include *pip*, *ipython*, and *pytest*.

FullwaveQC requires no compilation and can easily be installed by cloning the *GitHub* repository in a desired computer directory and running the *setup.py* file. It is recommended that FullwaveQC is installed using the Conda package manager for the software dependencies using the *Python 3.7* version. FullwaveQC is best used inside *.py* scripts or Jupyter Notebooks and does not support command-line interface. The code snippet below shows how to clone the repository, install and import FullwaveQC into a *.py* script:

```
# In the command-line
# clone the repository with git
git clone https://github.com/msc-acse/acse-9-inde
    pendent-research-project-dekape
cd acse-9-independent-research-project-dekape

"""
This will give you the following folder structure:


|-- /acse-9-independent-research-project-dekape
|------ /fullwaveqc
|---------- /docs
|---------- /examples
|---------- /fullwaveqc
|---------- /tests
|---------- LICENSE
|---------- README.md
|---------- requirements.txt
|---------- setup.py
|------
|-- ...

"""

# install requirements in python environment
pip install -r requirements.txt

# install fullwaveqc in python environment; dir.txt
    has the location of where the package has been
    installed
python setup.py install --record dir.txt
```

```
# In a python script or Jupyter Notebook import
    FullwaveQC as
from fullwaveqc import tools, geom, visual,
    siganalysis, inversion

# Your python code here...
```

The source code of FullwaveQC is located in the *fullwaveqc* folder, and unit tests of each module in the *tests* folder. The *docs* folder contains the *html* documentation files created using the automatic documentation generator *Sphinx v2.1.2*. The *examples* folder contains a summary of all models produced throughout the project, along with their Fullwave3D runfiles, and Jupyter Notebook examples that demonstrates the full functionalities of FullwaveQC.

FullwaveQC performs by storing the seismic datasets in memory for analysis. For this reason the usage of FullwaveQC requires a RAM capacity at least as large as the dataset in hands.

INVERSION STRATEGY

Inversions with Fullwave3D were performed in the time domain without anisotropy, as to agree with the acquisition parameters of the synthetic data. The default method for approximating the Hessian was steepest descent and the computation of the global step length was done with numerical line search. In order to increase the speed of computation, only 1 out of 5 shots were used per iteration, and these were rotated to ensure all shots were accounted for throughout the workflow. All runs were performed using the low-order kernel. The inversions presented were performed in Total's Pangea HPC cluster in Pau, France, using 12 CPUs, each with 16 active threads. These parameters were not modified or tested throughout the execution of the project in light of consistency and methodology testing.

In field data, correct extraction of a source wavelet from the acquired dataset is important for FWI convergence, and failing to do so will most likely introduce spurious effects within the models and push the inversion to a local minimum. See for example the discussion on the source wavefield by Warner et al. (2013b). For this project, the exact survey wavelet was used for forward modelling and inversion during FWI. It is also not within the scope of this work to test or provide a method to extract the correct source wavelet from the data.

Signal analysis of the observed data, Figure 11, shows relevant frequency contents up to 15 Hz. This upper limit set the maximum frequency we wish to invert for. Using Equations (25) and (26), with minimum and maximum model velocities of 1500 m/s and 3700 m/s, respectively, the minimum model resolution required is of 25 m grid cell-size and a maximum of 3.4 ms time sampling. The 25 m grid cell-size limit was used throughout this project as a a trade-off between resolution and computational time.

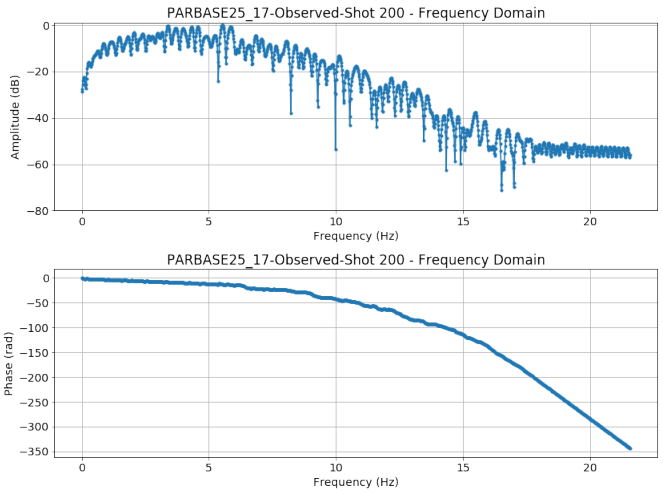


Figure 11: Frequency density and unwrapped phase of the observed baseline data, shot number 200 (at 6 km lateral offset).

The frequency continuation scheme was set as 13 blocks of frequencies 3.0, 3.4, 3.9, 4.5, 5.2, 6.0, 6.9, 7.9, 9.0, 10.2, 11.5, 12.9 and 14.4 Hz. A slightly lower frequency than the maximum allowed of 15 Hz eventually prevents the models from producing numerical dispersion artefacts during the last inversion iterations. Each block ran for 10 iterations, totalling 130 iterations per run, defining the stopping criteria for the inversions. For the given parallelisable resources and a total of 104 interleaving shots per iteration to process, Fullwave3D

performed in average 1 iteration every 2 minutes. The parameters described above were chosen arbitrarily. Further and more elaborate discussions on how to optimise the frequency continuation blocks for performance can be found in the inversion strategy chapter of Warner et al. (2013b).

In compliance with the minimum absorbing boundary requirements established by Equations (27) and (28), 100 extra absorbing cells were added to the left, right and bottom of the model, and 150 cells were added to the top of the model, of which 100 were absorbing.

RESULTS AND DISCUSSION

A series of 63 evolving inversion models were produced throughout the time frame of this project, from which inversion parameters and incorrect set ups were progressively identified, tested and corrected. Tests were performed on the baseline inversion starting from a smoothed linear-gradient starting model with restricted a priori information (Figure 12a). Once a stable inversion was found, the same settings and parameters were used to invert the monitor dataset in the parallel, sequential and double difference techniques. The adjustments made for each model can be found in the *examples* folder of the *GitHub* repository. Here we introduce the most significant baseline inversion and corresponding timelapse results acquired prior the usage of FullwaveQC, and then continue to show the most useful applications of the package and the final quality-checked results.

Baseline Inversion

Figure 12b shows the first substantial result obtained from the baseline inversion. The PARBASE25_7 model, whose SegyPrep and Fullwave runfiles are shown in Appendix A, is the inversion outcome starting from the model in Figure 12a. The depth and velocity of the water column in the starting model are exact with respect to the true model, and any model updates in this region were disregarded by setting a minimum velocity cutoff for updates.

The structure of the anticline, normal faults and location of low-velocity events are well recovered in PARBBASE25_7, although the amplitude of these events are underestimated by approximately 56% (Figure). This can be explained by the fact that the 90m-thick hydrocarbon layer is smaller than the largest seismic resolution of 100 m. As a consequence, the recovery of the amplitude is limited by the seismic coverage of the survey and frequency bandwidth of both the data and the FWI inversion. The model layering is also well resolved considering the coarser grid. The footprints that extend throughout the left and right sides of the model, from seabed to basement, are features caused by the lack of survey illumination.

Anomalous high velocities of 3100 m/s above the reservoirs and at the seabed are observed in PARBASE25_7, in particular at the location of the last source at 14 km. Some faint reflection-like structures are seen throughout the model and a significant amount of noise in both vertical and horizontal directions is also resolved.

Taking into account the many spurious artefacts compared to the true baseline model, it is a straightforward claim that the current inversion result of PARBASE25_7 is rather mediocre.

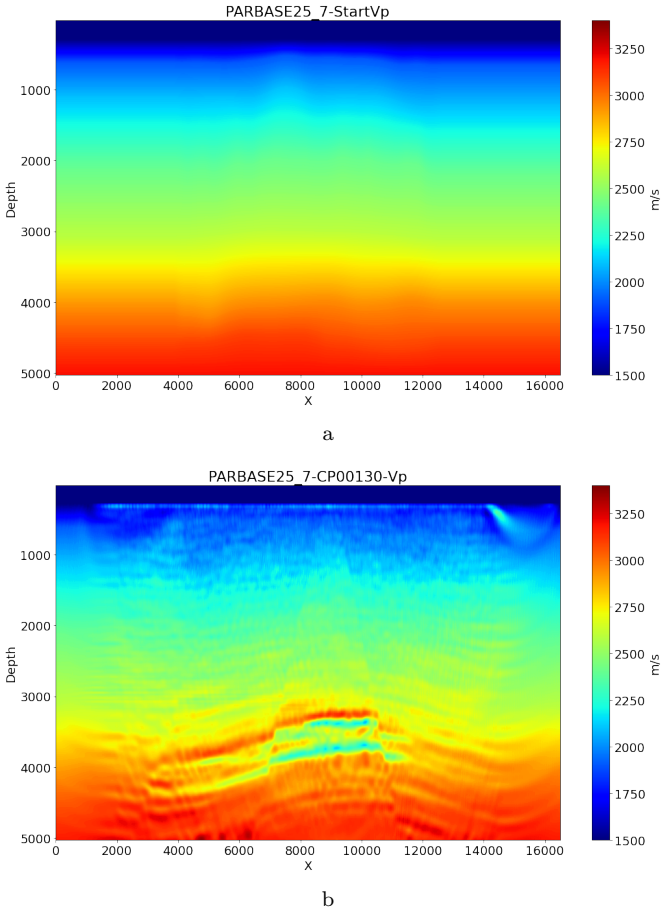


Figure 12: (a) The starting model for PARBASE25_7 inversion and (b) the final model produced after 130 iterations. Axes given in meters.

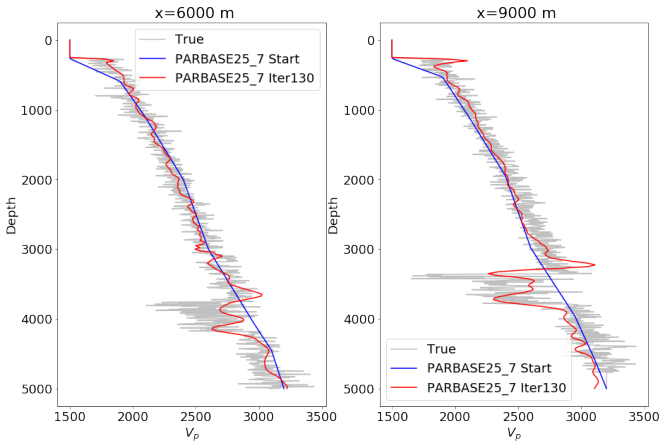


Figure 13: P-wave velocity depth profiles of the starting and final models of PARBASE25_7 at lateral offsets of 6 and 9 km. Depth is shown in meters and V_p in m/s.

These results are improved by conducting a series of quality assessments using FullwaveQC.

A forward model over PARBASE25_7 with the same modelling parameters is performed as to produce synthetic shots for a direct comparison to the observed data. This comparison is done using the interleaving features of FullwaveQC and are shown in Figure 14. Figure 14 highlights the agreement of the predicted and observed datasets of PARBASE25_7, such that the predicted dataset shows negligible phase differences in the main reflection features. However, careful observation shows that the predicted dataset contains higher frequencies compared to the observed data. Closely comparing the density spectrum of the wavelet, observed and predicted datasets using FullwaveQC's *siganalysis* module, it is possible to investigate the cause of these spurious high-frequencies. In Figure 15, it is clear that not only the peak frequency of the predicted dataset does not match the peak frequency of the wavelet, but that its peak frequency is at 0 Hz. A non-zero phase wavelet input is likely to be the cause of low frequencies being suppressed. This phenomenon indicates that the wavelet is wrongfully pre-processed somewhere between the inversion set up and the forward modelling.

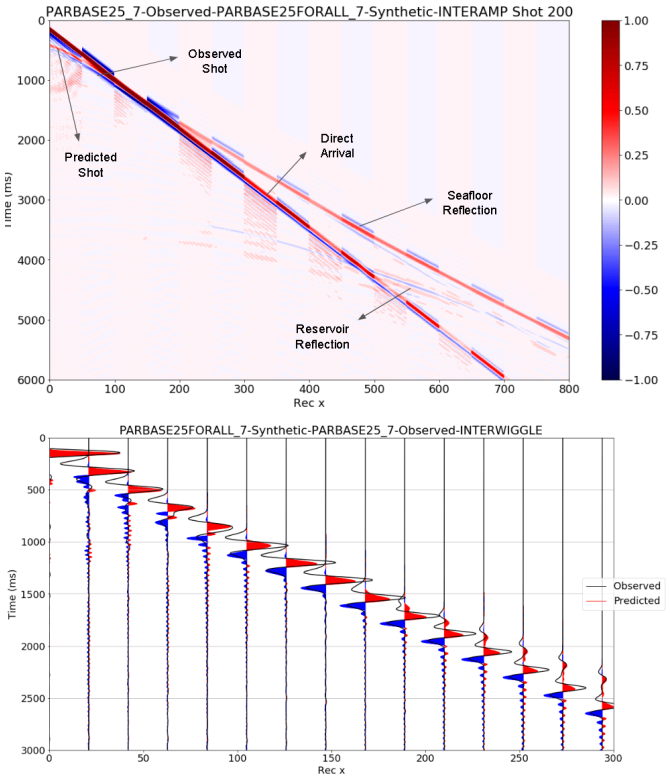


Figure 14: Interleaving amplitude and wiggle plots of the predicted and observed baseline data of the final PARBASE25_7 model. Shot number 200 (6 km offset). Receiver axis is given in offset index. The observed signal is represented as black wiggles and predcited with red and blue fillings. Main features observed are the direct arrival from the propagation of the wavelet in water, the seabed reflection and the reservoir reflection.

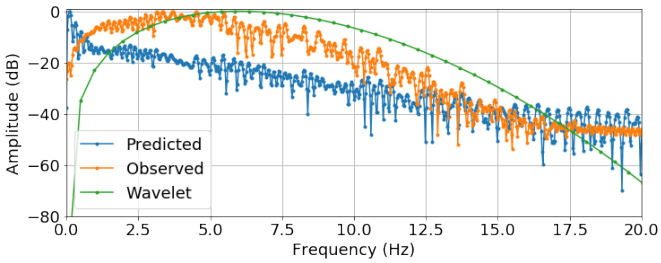


Figure 15: Frequency density observed from the wavelet and predicted and observed datasets at shot 200 (6km lateral offset). Predicted shot corresponding to the final model of PARBASE25_7.

Extensive investigation of the Fullwave3D workflow showed the the cause of this unexpected pre-processing was the SOURCE DELAY parameter, which is used to align the arrivals of the predicted data to that of the observed. In the PARBASE25_7 inversion, the source delay applied was of -200 ms such that the peak of the wavelet would be at 0 s in time (Figure 16). In doing so, however, any signal in the negative time domain is neglected by Fullwave3D and is responsible for introducing zero and high-frequencies to the wavelet (Figure 17). The frequencies introduced by sharply cutting the wavelet fall beyond the limit established by the kernel.

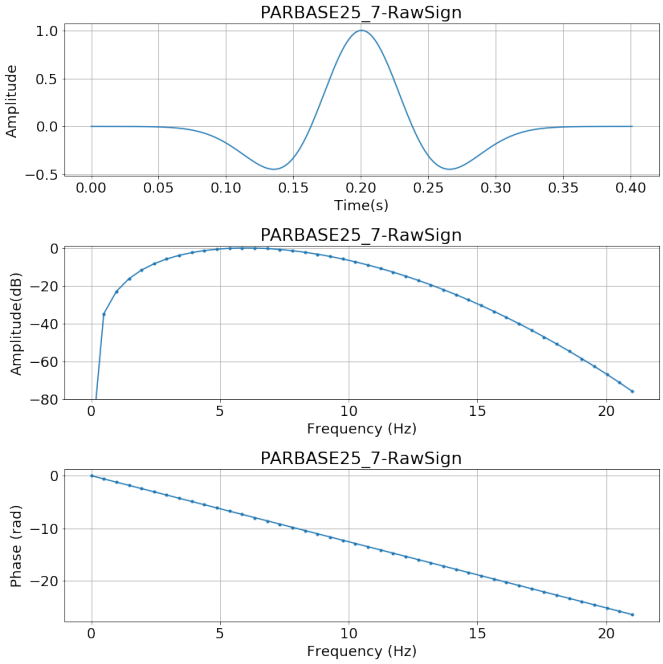


Figure 16: Exact survey wavelet used as input for the inversions in Fullwave3D before source delay of -200 ms. Phases are unwrapped in the frequency domain.

Further issues with the boundary set up were identified, where the distance between the source and the absorbing boundaries was zero, violating the requirements of one Fresnel zone radius. With thorough inspection, it was also discovered that

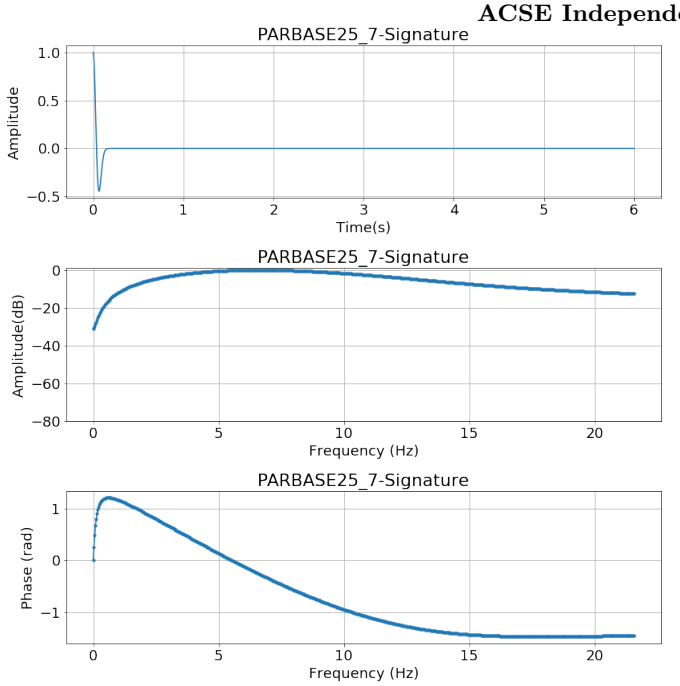


Figure 17: Exact wavelet with a source delay of -200ms applied internally by Fullwave3D and used for propagation during forward modelling. Phases are unwrapped in the frequency domain.

Fullwave3D by default applies Gardner's law (Gardner et al., 1974) to the model as to account for the change in model density with depth. For the next substantial baseline run, the Gardner effect was removed to produce synthetics corresponding to a constant density model, the source delay problem was fixed by adding a positive data delay of the same amount instead, the number of extra boundaries was increased at the top of the model to agree with the conditions of Equation (28), and finally, a small amount of Gaussian smoothing was added to the gradient at each iteration. By smoothing the gradient we aim to accomplish two things: 1) remove the high frequency horizontal events that do not belong to the model, and 2) slow down the convergence and decrease the chances of converging to a local minimum. The gradient smoothing was applied using a built-in Fullwave3D parameter, proportionally to the wavelength of each cell at the particular frequency of inversion. The adjusted model PARBASE25_12 is show in Figure 18 and corresponding runfiles in Appendix B.

PARBASE25_12 recovers the velocities at the reservoir and basement of the model better than PARBASE25_7, as seen in Figure 19. The reflection-like features of PARBASE25_7 vanish and the high velocity anomaly at the last source position disappear. These improvements are attributed to the correction of the boundaries as per agreement with Equation (28). The model recovered in PARBASE25_12 is overall smoother and more accurate. A similar investigation of the predicted and observed dataset frequencies shows the peak and high frequencies in better agreement (Figure 20).

It is immediately visible, however, that the seabed velocities

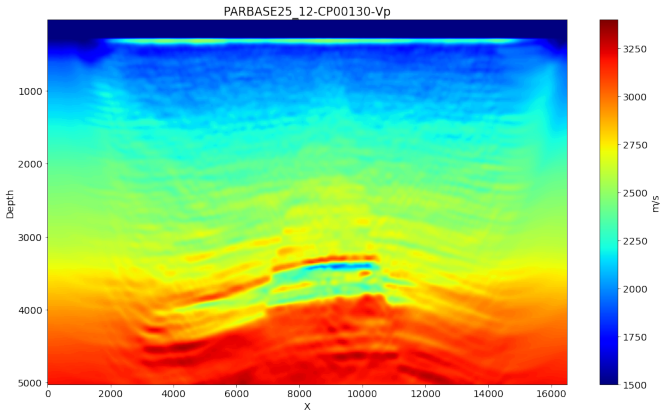


Figure 18: The final inversion model of PARBASE25_12 after 130 iterations.

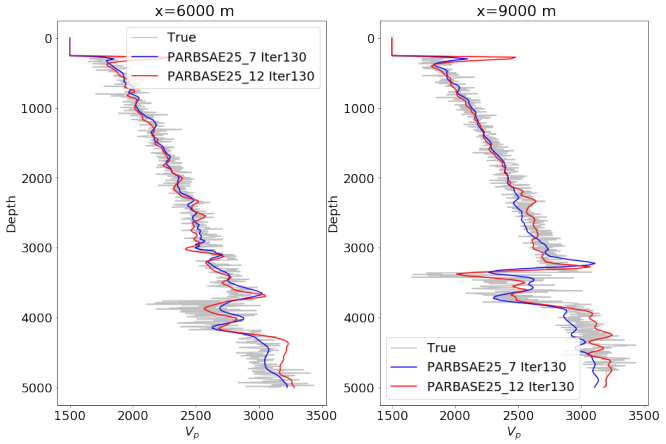


Figure 19: P-wave velocity depth profiles of PARBASE25_7 and PARBASE25_12 models compared to the true baseline model. Lateral offsets at 6 and 9km in the model space, and low velocity events correspond to the oil and gas reservoirs, respectively. Depth is shown in meters and V_p in m/s.

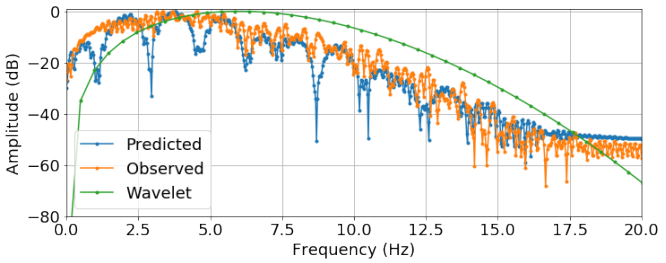
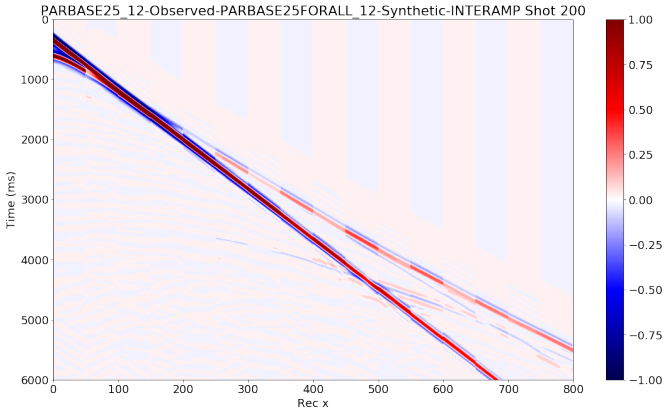
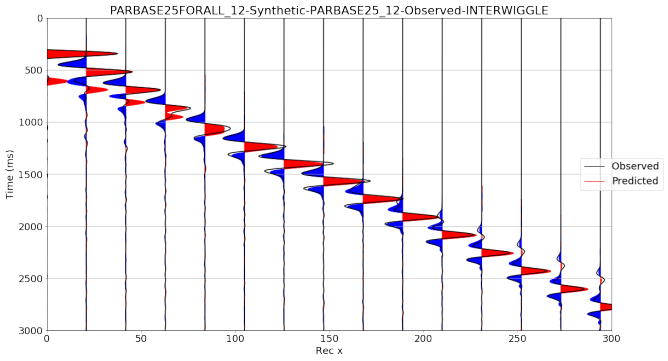


Figure 20: Frequency density observed from the wavelet and predicted and observed datasets at shot 200 (6 km lateral offset). Predicted shot corresponding to the final model of PARBASE25_12.

in the PARBASE25_12 model are incorrectly resolved, showing velocities up to 25% higher than expected. Indeed, as we analyse the synthetic shots produced by forward modelling PARBASE25_12 interleaved with the observed shots (Figure 21), we observed the discrepancy in reflection strength from the two seafloor signals. At near offsets the strength of the seafloor reflection in the predicted dataset is nearly 5 times as large as that of the observed.



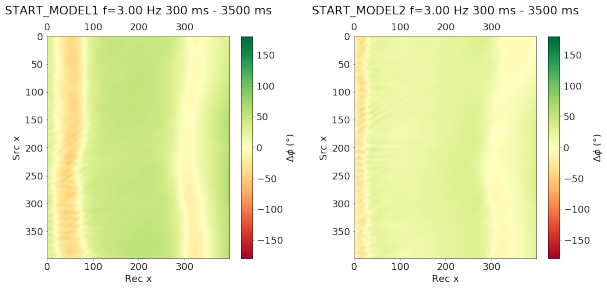
a



b

Figure 21: Comparison of shot 200 (at 6 km lateral offset) of the observed and predicted dataset of PARBASE25_12 through interleaving amplitudes and wiggle traces.

One possible explanation for this phenomenon is that the optimisation of the model is stuck in a local minimum. We are left to investigate the adequacy of the starting model and use Fullwave3D to produce synthetic shots and FullwaveQC to produce interleaving and phase differences plots at near-offsets. Despite the phase difference plots (Figure 22a) for the starting model not displaying sharp jumps of 360^o, the interleaving plot (Figure 23a) clearly shows the seabed reflection with over half a cycle skip. An improved starting model (Figure 24a) is therefore designed by increasing the velocity at the seabed such that the transition between the water column and the model is faster but also smoother (Figure 24b). This assumption introduces more a priori knowledge to the starting model that may or may not be available from field surveys, but can sometimes be obtained with near-surface modelling using first break picking. Similar analysis over the new starting model shows a skip of less than half a cycle (Figure 23b) and a lower-amplitude, smoother transition in the phase differences (Figure 22b).



a b

Figure 22: Phase difference between observed and predicted datasets from the (a) old starting model (b) new starting model. The phase differences are calculated for a frequency of 3Hz, for 400 shots and 400 receivers at a window of 300-3500ms. Axes are given in source and receiver indices.

The resulting inversion starting from the improved starting model is PARBASE25_18, shown in Figure 25. With the appropriate corrections, PARBASE25_18 provides a more accurate and robust inversion than the other runs and is considered to be our best baseline solution. The improvement in results can be further confirmed by computing the zero-lag cross correlation of the synthetic shots at the reservoir window in Figure 26.

Timelapse Results

Computing timelapse differences is a simple task of subtracting a baseline model from a monitor model. For each relevant PARBASE25 inversion, a corresponding monitor inversion was produced using the same parameters. Starting from the same starting model, the PARMON25 models were produced, and starting from the last iteration output of the PARBASE25 inversions, the SEQMON25 models were produced. These represent the monitor inversions for the parallel and sequential schemes, respectively.

To generate the double-difference monitor dataset for the DDWI inversion, it is firstly required that the amplitude of the PARBASE25 synthetics with respect to the observed dataset is normalised. This is required because the amplitude balancing performed by Fullwave3D during inversion interferes with the amplitude range that is returned from its synthetic shots, and as the true difference is added to the baseline synthetics, not only their location must be precise but their relative amplitude should also be. The normalisation performed for this project consists of matching the maximum amplitude values of the first trace of each shot and scaling the remaining traces accordingly. Provided the datasets are noise-free, which here they are, this assumption is justifiable. The DDWI monitor inversion is then the result of using this normalised baseline-constrained dataset as the observed data starting from the last iteration output of the corresponding PARBASE25 inversions. Measuring the

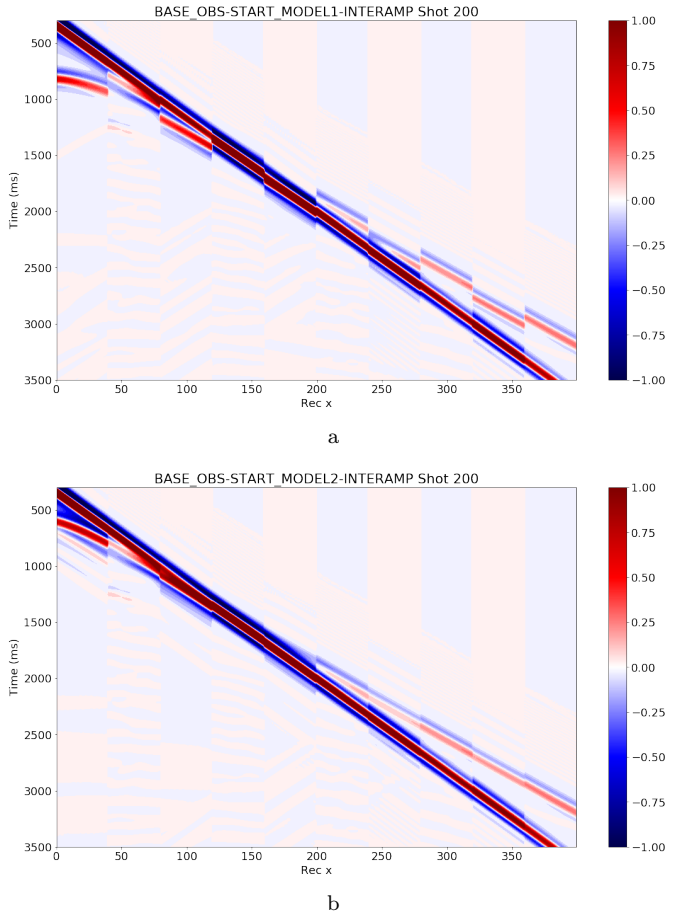


Figure 23: Interleaving amplitudes of observed and predicted datasets for the (a) old starting model and (b) new starting model. The first block refers to the predicted dataset and the second to the observed. Cycle skipping observed in the old starting model as shown by the approximately 1 cycle skip in the seabed reflection.

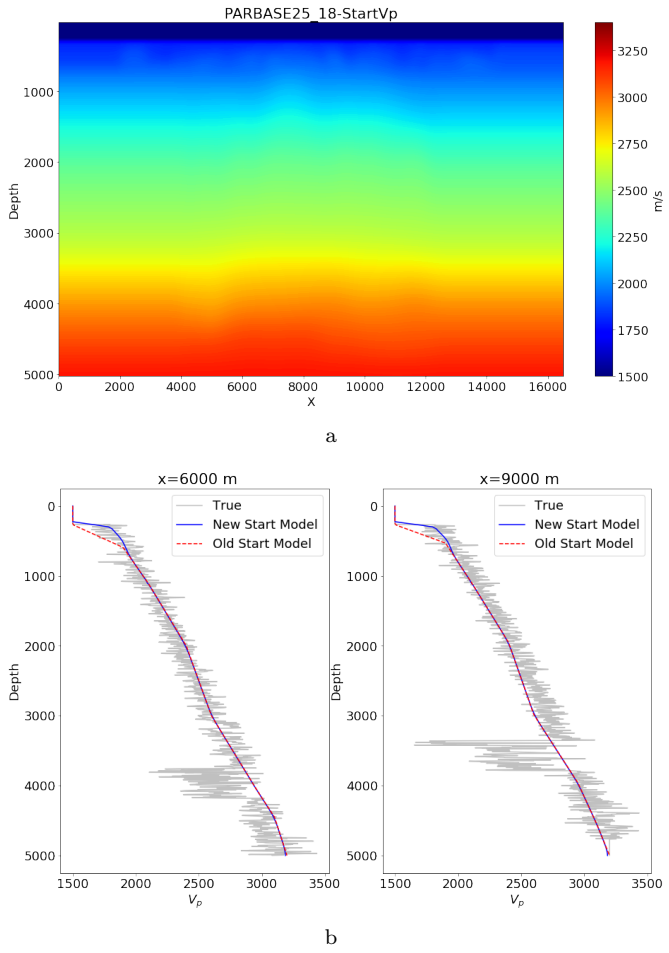


Figure 24: (a) Improved starting model (b) Comparison of old and new starting models with a V_p depth profile. The only modification of the new starting model is a better and smoother agreement at the seabed.

impacts of normalisation techniques on the generation of the double-difference monitor dataset could become a mini-project of itself and is not explored here.

The true timelapse difference is obtained by subtracting the true baseline model from the true monitor model and is shown in Figure 27a. There are two main features from the true difference we wish to retrieve from the timelapse schemes: 1) the removal of the hydrocarbon reservoir, characterised by the increase in P-wave velocity between the sedimentary layers above the anticline and adjacent to the western faults, and 2) the geomechanical effects above and below the reservoir location as a results pressure depletion after production (Hatchell and Bourne, 2005).

The results from applying the timelapse schemes to the PARBASE25_7 and PARBASE25_18 are shown in Figure 27. Overall, for both inversion settings, the differences are better resolved at regions above the reservoir. This can be explained by the energy decrease of the wavelet as it is strongly reflected by the reservoir. The corrections made from PARBASE25_7

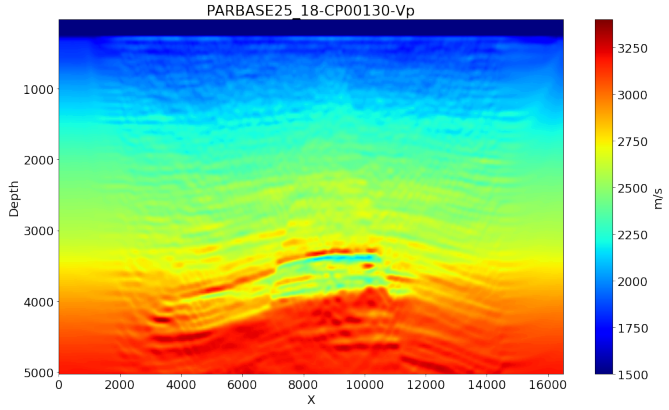


Figure 25: The final inversion model of PARBASE25_18 after 130 iterations.

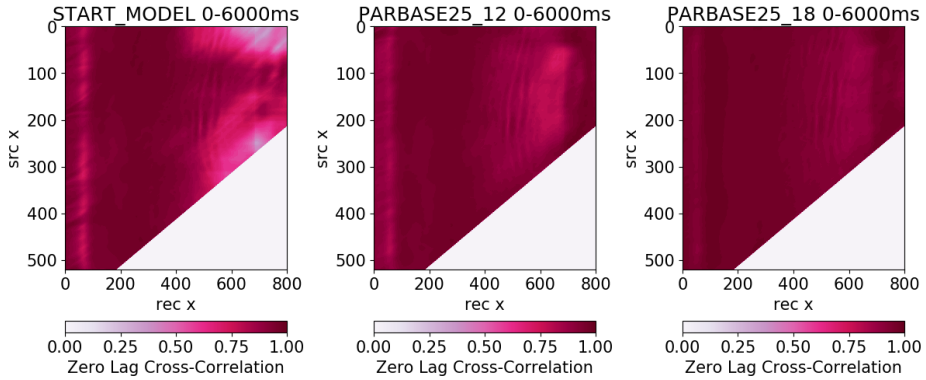


Figure 26: Zero-lag full window cross-correlation between observed dataset and predicted datasets for (a) new starting model, (b) PARBASE25_12 and (c) PARBASE25_18. Cross-correlation calculated for all shots and all receivers. The bottom-right empty triangle represents the lack of seismic coverage due to the survey leaving the model space. The vertical events at receiver 70 represents the interference of the seabed reflection with the direct arrival. The events between receivers 500-800 are mostly corresponding to the reservoir reflections. Axes given in source and receiver indices.

to PARBASE25_18 are successful in isolating the changes at the reservoir and recovering the geomechanical effects associated with pressure depletion. Random vertical and horizontal artefacts are removed and strength of footprints lessened. All three methods recover the magnitude of variation in P-wave velocity at the gas in a range of 41-48% of the true difference, with the sequential method showing the highest fit. The parallel method, however, shows the best estimation of velocity magnitude at the oil reservoir on the left of the model with a 90% fit, while the sequential and DDWI methods recover only 50%. The parallel method fails at recovering the correct polarity on the geomechanical effects. The sequential method recovers these polarities above of the reservoir but overshoots below it. The DDWI method overshoots the magnitude of these effects both above and below the reservoir.

In the parallel method, the baseline and monitor datasets are prone to be inverted at different convergence levels. This means that the baseline residual $\delta\mathbf{d}_b \equiv \mathbf{p}_b - \mathbf{d}_b$ differs from the monitor residual $\delta\mathbf{d}_m \equiv \mathbf{p}_m - \mathbf{d}_m$. When computing the difference between these models, the discrepancy between the two residuals will be resolved as noise. A similar difficulty is observed for the sequential method. As we use the final baseline model as the starting model for the monitor, the baseline residual $\delta\mathbf{d}_b$ gets injected into the monitor predicted dataset, to which the monitor inversion tries to accommodate for. This consequently leads to a higher difference in convergence between the baseline and the monitor models (Figure 29a), and further increases the difference between $\delta\mathbf{d}_b$ and $\delta\mathbf{d}_m$. Another issue with the sequential method is its attempt to further invert the background structures inherited from the baseline model, in a way that this background will be essentially inverted for twice the amount of iterations in the monitor model with respect to the baseline. The result is a timelapse model that contains a substantial amount of resolved background structures. This is clearly seen in Figure 27c, which represents the originally proposed sequential scheme applied with the inversion parameters of PARBASE25_7.

For an improved implementation of the sequential method, the above issues are mitigated by re-inverting the baseline as well as inverting the monitor datasets using the last iteration of the first baseline inversion as the starting model. This improved version of the sequential method, schematised in Figure 28, is essentially equivalent to the parallel method, but where both inversions are constrained by a more detailed starting model. With these adjustments, the improved sequential method aims to push both inversions towards the same convergence such that $\delta\mathbf{d}_b \approx \delta\mathbf{d}_m$. Spurious artefacts are further prevented by smoothing the starting model and the results are seen in the sequential inversion of PARBASE25_18 in Figure 27f. The improved method significantly isolates the reservoir timelapse changes, removing coherent background structures. In the convergence analysis (Figure 29b) it is also visible that the method succeeds at pushing the baseline and monitor models towards the same convergence.

The theory behind DDWI is designed such that the difference between the residuals is mathematically minimised. Taking the derivative of Equation (22) with respect to the monitor predicted dataset $\mathbf{p}_\mathrm{m}$, gives:

$$\frac{\partial f_{\mathrm{DDWI}}(\mathbf{m})}{\partial \mathbf{p}_\mathrm{m}} = (\mathbf{p}_\mathrm{m} - \mathbf{d}_\mathrm{m}) - (\mathbf{p}_\mathrm{b} - \mathbf{d}_\mathrm{b}) = \delta\mathbf{d}_\mathrm{m} - \delta\mathbf{d}_\mathrm{b} \quad , \quad (29)$$

which shows that as the objective function $f_{\mathrm{DDWI}}(\mathbf{m})$ is minimised, so is the difference between the baseline and monitor residuals.

As the minimisation of the residuals is guaranteed by the DDWI method, one would expect this method to be more robust and provide more accurate changes the than the improved sequential. It is observed from Figure 27 that both methods perform well, but that the improved sequential shows more accurate amplitude estimations. The poorer performance from the DDWI could be a consequence of the normalisation of the datasets. It is expected that more elaborate means of normalising would lead the DDWI to over-perform the improved sequential method. However, this would not be expected within a noise-free dataset and further investigation is required.

It is clear from this study that in order for timelapsing meth-

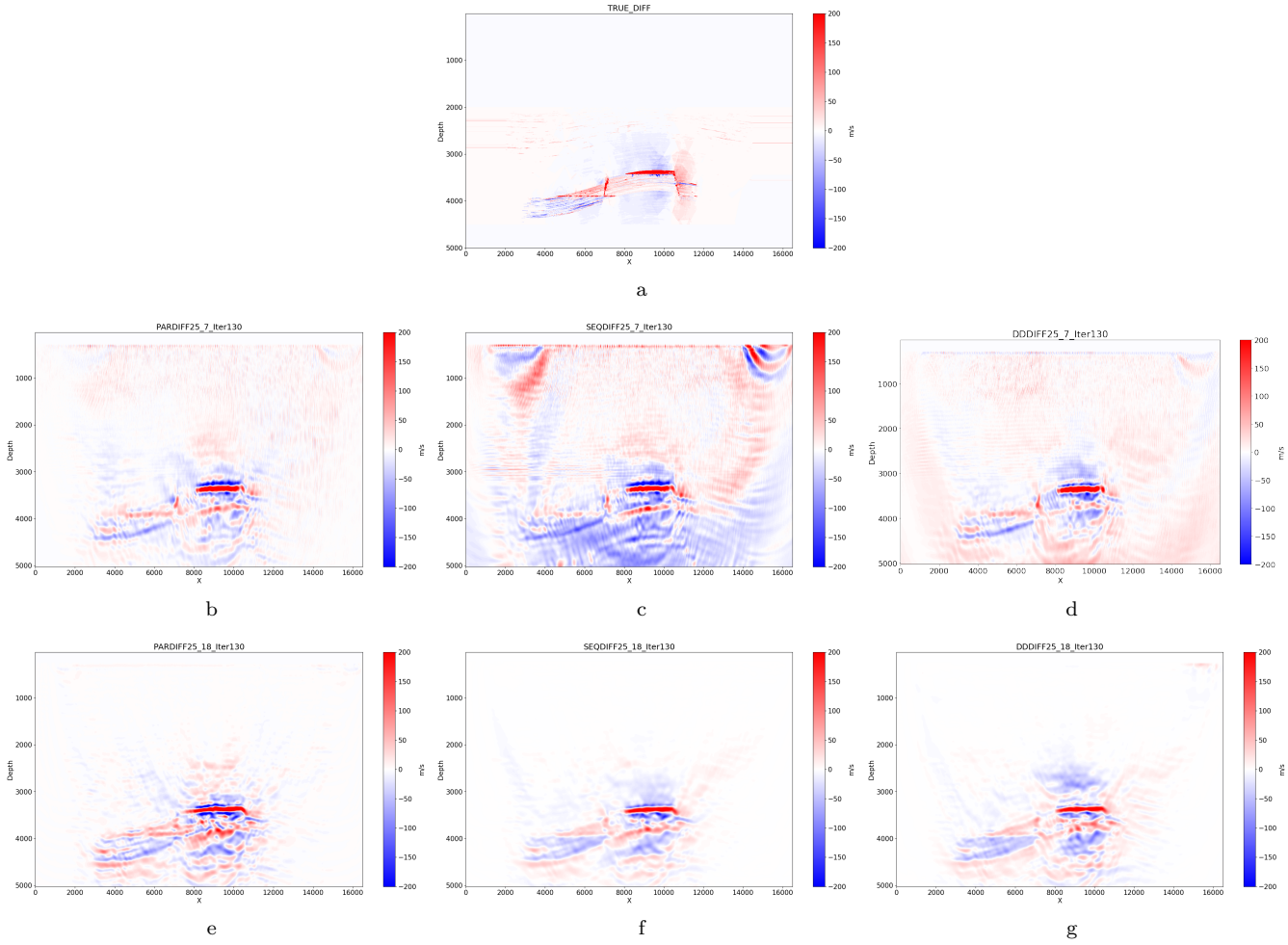


Figure 27: (a) True difference. (b-d) Recovered differences from applying the parallel, sequential and DDWI timelapse methods with the parameters of PARBASE25_7. (e-f) Recovered differences from applying the parallel, sequential and DDWI timelapse methods with the parameters of PARBASE25_18.

ods to provide useful and stable results, the convergences of the baseline and monitor inversions must be as close to each other as possible. Ideally this would be achieved if both inversions reached their respective global minima, but this is often not guaranteed. By running the inversions at higher resolutions with better starting models, and also with an adequate amount of gradient smoothing as to slow down convergence it is possible to increase the likelihood of this happening. These, however, are not always feasible due to the lack of prior model knowledge and and by limitations in computational time. A more realistic requirement is that both inversions result in equivalent local minima. With the parallel method there are not any clever ways of guaranteeing that. The standard sequential method further exacerbates this problem. The improved sequential and DDWI, however, help mitigate it and generally will perform better at removing coherent background noise and isolating the changes of the reservoir.

Quality Controlled Workflow

Given the extensive experiments performed with FWI in Fullwave3D throughout this project, the following workflow is proposed on the usage of SegyPrep, Fullwave3D and FullwaveQC programmes to produce stable and accurate timelapse results:

1. With the *fullwaveqc.siganalysis.dataspec* function, inspection of the frequency density of the observed datasets in order to identify the maximum frequency to invert for - this can be lower but fewer details will be recovered in the model;

2. For the maximum frequency and extreme velocities expected within the model, application of Equations (26) and (25), or similar variations, to choose resolution and sampling rate for kernel stability;

3. Designing of an appropriate inversion strategy for the particular problem, including frequency continuation blocks and Gardner's density models;

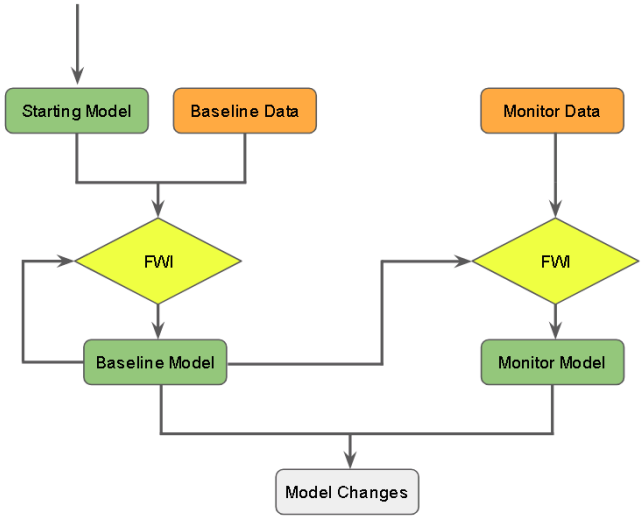


Figure 28: Improved sequential method where the baseline is re-inverted to achieve the same number of iterations as the monitor inversion.

4. Evaluation of the minimum number of absorbing boundaries with *fullwaveqc.geom.boundarycalc*
5. Verification of the wavelet with *fullwaveqc.siganalysis.wavespec* - must be within maximum frequency and zero-phased;
6. Formulation of a starting model for the baseline inversion;
7. Validation of acquisition geometry using *fullwaveqc.geom.surveygeom* after running SegyPrep on the starting model;
8. Production of synthetics over starting model with Fullwave3D;
9. Generation of interleaving amplitudes, interleaving wiggles and low-frequency phase difference plots at near-offsets. This can be done conjointly with the *fullwaveqc.siganalysis* and *fullwaveqc.visual* modules;
10. Identification of cycle-skipping phenomena, and repetition of items 3-6 until starting model is adequate;
11. Verification of the frequency content of the predicted data, which should be matched to the frequency content of both the wavelet and the observed data. This can be done with *fullwaveqc.siganalysis.datasepc* and *fullwaveqc.siganalysis.wavespec*. If not matched, some pre-processing is behaving unexpectedly;
12. Performing of inversion with Fullwave3D;
13. Production of synthetics over final inversion model with Fullwave3D;
14. Generation of interleaving amplitudes, interleaving wiggles and zero-lag cross correlations at windows of interest - checking for better match to observed dataset than starting model synthetics;
15. Analysis on the progression of the objective functional and step length values with the *fullwaveqc.inversion* module. Plateauing of functional and/or negative step lengths might indicate the model has encountered a minimum;
16. Animation of the iterations with *fullwaveqc.visual. animate* to identify unusual artefacts resulting from specific frequency blocks;
17. Repetition of items 12-16 for the monitor dataset, although the signal analysis is not strictly necessary;
18. Side-by-side comparison of the progression of the objective functional for both baseline and monitor datasets. If convergences are disparate, inversion parameters and starting model might need adjustments;
19. For the sequential method: repetition of items 12-18 for both the baseline and model datasets using the smoothed final baseline model as starting model;
20. For the DDWI method: appropriate normalisation of baseline synthetics and creation of the double-difference monitor dataset with *fullwaveqc.tools.ddwi*. Repetition of items 12-17 for the monitor inversion using the final baseline model as starting model.

LIMITATIONS AND FURTHER WORK

Fullwave3D and Timelapse Inversions

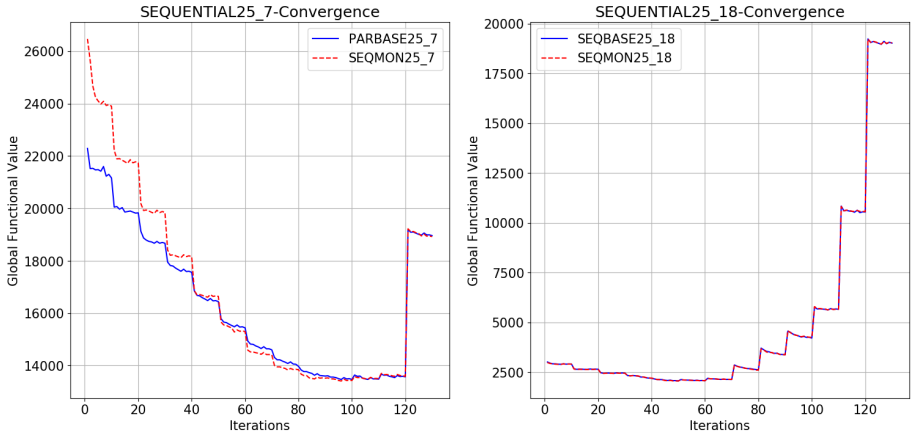
As previously discussed, for timelapse methods to be stable, it is required that the convergence of the baseline inversion is somewhat equivalent to that of the monitor, such that the residuals from both inversions are as close to each other as possible. As it stands, Fullwave3D provides no internal means to relate the baseline to the monitor inversion. Within the parallel and sequential methods, one straightforward way of leading both inversions towards similar a convergence is to implement a weighted total variation (TV) regularisation to the L2-norm objective function in the monitor inversion. TV regularisation has been shown to be successful in reducing high oscillatory artefacts in the recovered model differences (Maharramov and Biondi, 2015). In general terms, the TV regularisation for the parallel and sequential methods would consists of minimising the following objective function for the monitor dataset:

$$f(\mathbf{m_m}) = \frac{1}{2}||\mathbf{G}(\mathbf{m_m}) - \mathbf{d_m}||_2^2 + \lambda||\mathbf{W}\Delta\mathbf{m}||_{TV} \quad , \qquad (30)$$

where $\Delta\mathbf{m} = \mathbf{m}_\mathrm{m} - \mathbf{m}_\mathrm{b}$, $\mathbf{W}$ is a weighting mask that penalises the objective function for updates in desired regions, and λ is a Lagrange multiplier. The regularisation term, therefore, minimises the difference between the models at specific regions.

The main difficulty in the implementation of TV in Fullwave3D is that the TV norm often comes in the form of a first order Tikhonov regularisation under an L1-norm. Minimising an L1-norm can be a challenging task due to its singular nature (Bach et al., 2012). With adequate mathematical manipulations, however, this regularisation is a straightforward addition to Fullwave3D.

As for the DDWI method, not only it could benefit from a TV regularisation feature in Fullwave3D, but also from an option to adjust the amplitude balancing in the synthetic mode.



a b

Figure 29: Convergence analysis of the (a) sequential method with PARBASE25_7 inversion parameters and (b) improved sequential method with PARBASE25_18 inversion parameters.

This is already performed behind the scenes by Fullwave3D in the tomography mode, and provides a much more consistent normalisation than the one used throughout this project. The normalisation used in this project, also the one embedded in the *fullwaveqc.tools.ddwi*, only makes sense under a zero-noise condition in the datasets, and even so does not guarantee the most accurate recovered model difference. We predict that improving the accuracy of this dataset normalisation would significantly increase the accuracy of the recovered differences from the DDWI method, especially in the case of field-acquired datasets.

Although the results presented in this report show quantifiable and reliable differences between the implemented timelapse methods, these findings and conclusions are limited to the specific acquisition parameters of the observed datasets. The further work necessary to validate the generalisation of these findings consists of adding to the datasets anisotropy, attenuation, and non-constant density distortion factors as well as random noise, multiples and ghosting artefacts. Ideally, these factors would be automatically incorporated in field-generated datasets as opposed to synthetic ones. In the field, the data generated would also be an elastic response to the wave propagation, and this factor would also be useful to test for. Luckily, Fullwave3D is able to accommodate for most of these field-related traits into the inversion workflow, and performing these tests would be an uncomplicated continuation of this work. It would be valuable to repeat the presented tests using the conjugate gradient method of approximating the Hessian and the higher-order kernel of Fullwave3D to ideally achieve faster and more accurate convergences.

In addition, the datasets were acquired at perfectly repeatable conditions for the baseline and monitor surveys. This is often impractical and has been shown to destabilise the timelapse computation when the discrepancy in survey conditions is large (Yang et al., 2015). This is a problem that Fullwave3D does not account for as its design does not include any sort of timelapse support.

FullwaveQC

With efficient error handling throughout its development, FullwaveQC is able to identify and deal with unexpected inputs. For example, the *fullwaveqc.siganalys.phasediff* function performs a trace-by-trace comparison between two datasets for it to be well defined. In the case of missing or undefined traces, it uses *Try-Except* statements to set the phase differences corresponding to these traces to zero. The same procedure is applied to the *fullwaveqc.siganalysis.xcorr* function, and the application of this error-handling feature is clearly shown in the bottom-right zero-valued triangle observed in Figure 26. In Figure 26 the far-offset traces of the last shots are undefined because geographically they lie outside the model space. The package, however, has not been tested on datasets with imperfect or missing data and is likely to require further features for different project and dataset types. With that in mind, FullwaveQC has been sustainably built and is well documented such that its functionalities are easy to develop on.

FullwaveQC also lacks functionalities which, for this project, were not particularly necessary but that for other types of project could provide valuable quality-check results. These include visualisation of the gradient computed by Fullwave3D at each iteration step, a feature to join and split SEG-Y data per shot, and a function to retrieve a minimal-phase wavelet from the dataset using a matched Wiener filter.

Apart from familiarising with Fullwave3D's workflow and using FullwaveQC to investigate and produce better inversion models, a large portion of this project was dedicated to debugging and improving the *fullwaveqc.siganalys.phasediff* function. The first difficulty encountered as the function was developed, was how to properly deal with phase unwrapping. To compute phase differences between two datasets, an FFT is performed to each paired trace individually and difference between the two phases at the chosen frequency calculated. As shown in Figure 30a, there is no guarantee that when performing the FFT at these traces separately that both phases would fall in the same quadrant, which in turn induces deceiv-

ing values in phase differences. To mitigate this, the phases are individually unwrapped in the frequency domain using the *numpy.unwrap* function, which changes absolute jumps in phase greater than 2π by adjusting the phases to their 2π compliments. This alleviates the problem, but is not able to deal with the phases wrapped in the receiver domain (Figure 30b). This is likely a product of the instability of phase unwrapping inside the frequency domain as shown in Figure 30b. As a matter of fact, instability of phase unwrapping is a highly discussed topic (Navarro et al., 2012) and a vast range of techniques have been discussed and implemented aiming at improving the accuracy of phase unwrapping whilst retaining computational performance. This is an issue that this project does not explore and we highlight that as a limitation for the *phasediff* function.

For some stable conditions, however, the computation of phase differences displays accurate results when unwrapping in both frequency and receiver domains (Figure 30c). In the FullwaveQC package, we have not devised a method of ensuring these stable conditions. It is left to the user to identify what signals are appropriate for the phase difference computation. In this work, however, we have observed instabilities in highly oscillatory, low-amplitude and/or noisy signals. As unwrapping is performed by removing 2π compliments from adjacent phases, from the beginning to the end of an array, unwrapping phase differences in space is effectively dependent on the phase calculated at the first point of the array. In the source-receiver continuity, this means that if the phase computation of the first trace of a shot is incorrect, the unwrapping in the receiver domain will propagate that inaccuracy to the other traces of the same shot. This effect is observed in Figure 30d. For this reason, the *phasediff* function will behave properly at low frequencies where signals are well defined. The option of windowing the phase calculation within the source-receiver continuity provides a way to ensure this stability, but this must be checked outside and prior the usage of the function. The phase difference analysis performed in this project had their windows carefully selected by visualising the datasets with the *fullwaveqc.visual* module.

The main purpose of providing a phase difference function was to identify possible cycle skipping artefacts from the starting model. These are characterised by high jumps in phase differences in the source-receiver spatial continuity, but as we observed from Figures 23a and 22a, these jumps were not identified clearly in the starting model. We suspect that this is due to a signal shadowing from the direct arrival. Provided the geometry is setup correctly, the direct arrival corresponding to the wavelet propagation through water from the source to the receivers are guaranteed to be in phase with the observed data. We see this effect in Figure 23a and it is likely that the phase difference computation has been strongly influence by it. A further work proposed for FullwaveQC is to provide a *muting* function that will remove the direct arrival from a trace by using the velocity of water to calculate the exact times at which the wavelet reaches each offset. This method of removing the direct arrival becomes less trivial as distortion factors are incorporated in the datasets.

For its trace-by-trace transforms and unwrapping operations, the *phasediff* function is the most time-consuming function in the FullwaveQC package. The function consists of looping through every trace of each shot, and a computational analysis shows an order $O(n^1)$ between performance time and problem-size. However, we succeeded in optimising the function and reduce its computational time by a factor of 10 by performing the FFT with a number of samples that is a power of 2 (Figure 31). This ensures an FFT performance of $O(nlog(n))$ instead of $O(n^2)$ and, despite working with more samples, it significantly improves performance.

Another technical impracticality with FullwaveQC is its dependency to the Segyio library and lack of adaptability for differently formatted SEG-Y formats. The *load* function was designed such that it reads a standard SEG-Y output from Fullwave3D, but it was observed that SEG-Y files generated by Total's internal tools do not have their headers in the same order. The objects outputted from the *load* function therefore provide an *attr()* method to easily access and manually correct the objects' attributes. As some other functionalities of FullwaveQC might throw errors for incorrectly loaded SEG-Y files, it is required that for all loaded objects these attributes are inspected by the user. Some warnings are raised inside the *load* function when it identifies model or time spacings that are less or equal to zero.

CONCLUSION

Overall, we successfully implemented the parallel, sequential and double-difference timelapse techniques within Fullwave3D without interfering with its main architecture. We have developed a diagnostic package, FullwaveQC, which provides useful features to determine and remove undesired behaviours during the inversion workflow. Amongst these behaviours, we have found that Fullwave3D performs a different pre-processing on the wavelet compared to Total's internal FWI software, which can introduce spurious high frequencies to the predicted datasets. We have successfully used FullwaveQC to identify cycle-skipping phenomena in the original starting model, and have shown that smoothing the gradient per iteration can bring positive results to the recovered timelapse models. From the learning curve of this project, we then went on to suggest how Fullwave3D and FullwaveQC can be best used conjointly as to produce robust timelapse models by performing a series of quality-check steps before and after implementation of the inversion workflow.

For the timelapse schemes, we have seen how correcting and improving the inversion parameters are successful in isolating the changes at the reservoir and suppressing background coherent noise. We have demonstrated that stable timelapse results are only produced when the difference between the baseline and monitor residuals are minimised. The parallel method is only guaranteed to meet this criterion when both baseline and monitor inversions reach their respective global minima or equivalent local minima. We have shown that the sequential method, as traditionally established, further aggravates this problem by injecting the baseline residual into the monitor inversion, forcing it to also accommodate for this numerical noise. We have devised, however, an improved version of the sequential method, consisting or rerunning the baseline and running the monitor model from a smoothed version of the first baseline inversion, such that the method effectively becomes a parallel

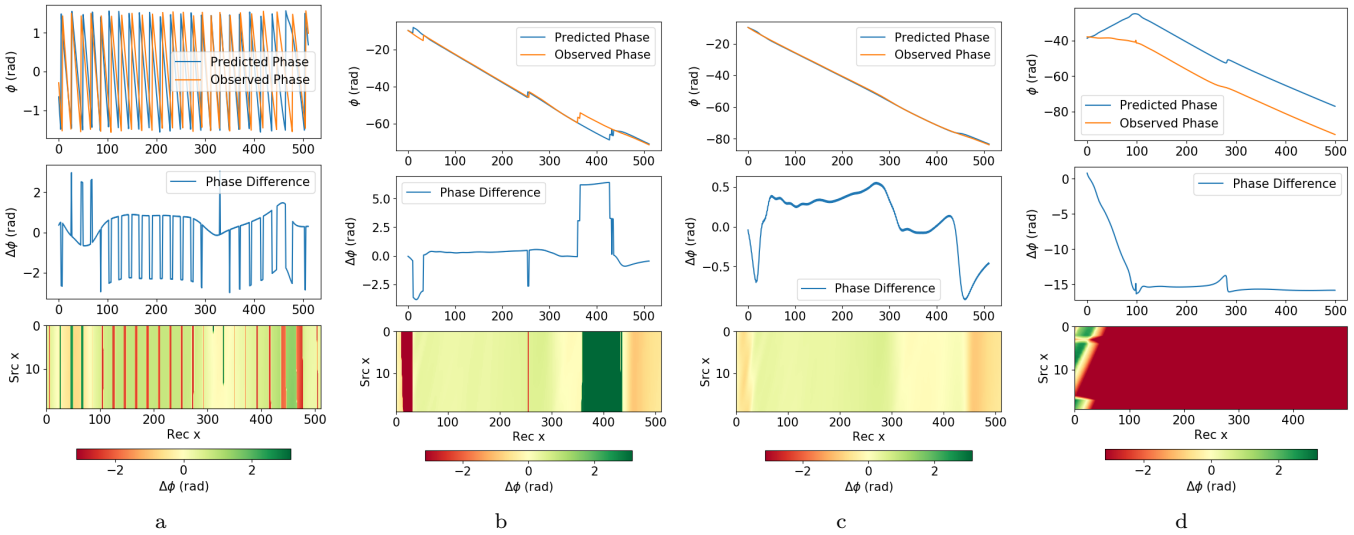


Figure 30: Phase differences calculated between the observed and predicted dataset over the improved starting model for shot 10 and frequency of 3Hz. The phases and phase differences are (a) wrapped in both frequency and receiver domains, (b) unwrapped in the frequency domain, and (d-e) unwrapped in both frequency and receiver domains. (a-c) are phase differences calculated of a 1500-200ms window, where the signal is dominated by the direct arrival and seabed reflections. (d) are phase differences calculated over a 2500-300ms window where the signal is dominated by weaker internal reflections.

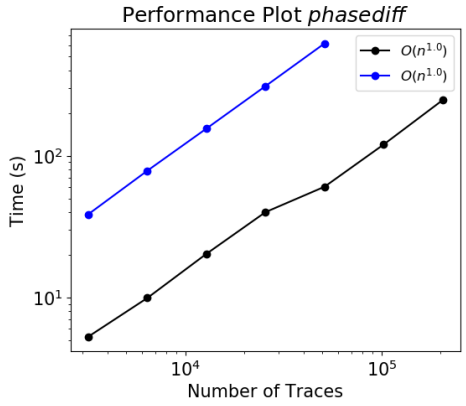


Figure 31: Convergence of the *phasediff* function of FullwaveQC. Blue (upper) line is the function operating the FFT without a number of samples that is a power of 2. Black (lower) line is the function operating the FFT with a number of samples that is a power of 2. Time is correspondent to a phase difference computation between the observed and improved model datasets, for 200 sources and full-window signal.

run, but a with a closer-to-true starting model constraint. We have seen that for the datasets provided, this is sufficient to force the baseline and monitor to equivalent minima, although it does not guarantee convergence to the global minimum.

For the DDWI method, we have shown how inverting for model differences mathematically implies that the difference between baseline and monitor residuals are minimised. However, when generating the appropriate DDWI dataset, its accuracy is likely to be limited by the simplistic dataset normalisation hereby applied. It was not within the scope of this project to devise the best way of normalising the dataset, especially because this normalisation problem could become a project of itself. We do realise that Fullwave3D performs amplitude balancing inside the tomography mode, and we suggest that it is adapted such that this option is also available for the synthetic mode. We expect that this more advanced normalisation method would allow Fullwave3D to integrate better with FullwaveQC and produce more accurate DDWI timelapse models, especially to non-synthetic datasets. On Fullwave3D, we also suggest that a weighted TV regularisation option is implemented, as it has been shown to remove high-oscillatory effects from recovered difference models in timelapse implementation.

As far as the development of FullwaveQC, we have shown how it can be useful to diagnose the inputs and outputs of Fullwave3D, but that it is limited to the Rec689 version of the code. We demonstrated how instabilities caused by phase unwrapping can deviate the *phasediff* function from producing robust and useful quality-check results, but have also shown that carefully selecting a low frequency for analysis and window that contains well-defined signal responses is able to stabilise this computation.

ACKNOWLEDGEMENTS

I would like to thank Melissa Grey and Tatiana Kalinicheva for taking the time to introduce me to the Fullwave3D programme. I would also like to thank Total and the GRC Team for sponsoring my MSc degree as well as for providing the datasets and internal software tools to perform this work. Finally, I would like to thank Pr. Mike Warner, Dr. Christophe Ramananjaona, and Dr. James Beckwith for the sharing their expertise and for providing the guidance for the completion of this project.

REFERENCES

Ackley, D. H., 1987, A connectionist machine for genetic hill-climbing: Kluwer Academy Publishers.

Bach, F., R. Jenatton, J. Mairal, and G. Obozinski, 2012, Structured sparsity through convex optimization: Statistical Science, **27**, 450–468.

Beydoun, W. B., and A. Tarantola, 2005, First Born and Rytov approximations: Modeling and inversion conditions in a canonical example: The Journal of the Acoustical Society of America, **83**, 1045–1055.

Born, M., 1926, "Zur Quantenmechanik der Stoßvorgänge On the quantum mechanics of collision processes. [Preliminary announcement (1)]: Technical report.

Bunks, C., F. M. Saleck, S. Zaleski, and G. Chavent, 1995, Multiscale seismic waveform inversion: Geophysics, **60**, 1457–1473.

Denli, H., and L. Huang, 2009, Doubledifference elastic waveform tomography in the time domain: , Society of Exploration Geophysicists, 2302–2306.

Gardner, G. H., L. W. Gardner, and A. R. Gregory, 1974, Formation Velocity and Density - The Diagnostic Basics for Stratigraphic Traps: Geophysics, **39**, 770–780.

Guasch, L., M. Warner, T. Nangoo, J. Morgan, A. Umpleby, I. Stekl, and N. Shah, 2012, Elastic 3D full-waveform inversion, *in* SEG Technical Program Expanded Abstracts: Society of Exploration Geophysics, 1–5.

Hatchell, P., and S. Bourne, 2005, Rocks under strain: Strain-induced time-lapse time shifts are observed for depleting reservoirs: The Leading Edge, **24**, 1222–1225.

Kvalsvik, J., 2019, Before Machine Learning: handling seismic data with Python and segyio: Presented at the 81st EAGE Conference and Exhibition 2019 Workshop Programme.

Maharramov, M., and B. Biondi, 2014, Joint full-waveform inversion of time-lapse seismic data sets: 954–959.

———, 2015, Robust Simultaneous Time-lapse Full-waveform Inversion with Total-variation Regularization of Model Difference: 77th EAGE Conference and Exhibition 2015, 2–7.

Maharramov, M., B. Biondi, and M. Meadows, 2016, Time-lapse inverse theory with applications: Geophysics, **81**.

Navarro, M. A., J. C. Estrada, M. Servin, J. A. Quiroga, and J. Vargas, 2012, Fast two-dimensional simultaneous phase unwrapping and low-pass filtering: Optics Express, **20**, 2556–2561.

Pratt, R. G., 1999, Seismic waveform inversion in the frequency domain, Part 1: Theory and verification in a physical scale model: Geophysics, **64**, 888–901.

Ratcliffe, A., C. Win, V. Vinje, G. Conroy, M. Warner, A. Umpleby, I. Stekl, T. Nangoo, and A. Bertrand, 2011, Full waveform inversion: A North Sea OBC Case Study, *in* SEG Technical Program Expanded Abstracts 2011: Society of Exploration Geophysicists, 2384–2388.

Shah, N., M. Warner, T. Nangoo, A. Umpleby, I. Stekl, J. Morgan, and L. Guasch, 2013, Quality assured full-waveform inversion: Ensuring starting model adequacy: 1–5.

Sirgue, L., O. I. Barkved, J. P. Van Gestel, O. J. Askim, and J. H. Kommedal, 2009, 3D Waveform Inversion on Valhall Wide-azimuth OBC: Presented at the 71st EAGE Conference and Exhibition incorporating SPE EUROPEC 2009.

Tarantola, A., 1984, Inversion of seismic reflection data in the acoustic approximation: Geophysics, **49**, 1259–1266.

Virieux, J., and S. Operto, 2009, An overview of full-waveform inversion in exploration geophysics: Geophysics, **74**, WCC1–WCC26.

Warner, M., J. Morgan, A. Umpleby, I. Stekl, and L. Guasch, 2012, Which Physics for Full-wavefield Seismic Inversion?: 74th EAGE Conference and Exhibition incorporating EUROPEC 2012.

Warner, M., T. Nangoo, N. Shah, A. Umpleby, and J. Morgan, 2013a, Full-waveform inversion of cycle-skipped seismic data by frequency down-shifting: 903–907.

Warner, M., A. Ratcliffe, T. Nangoo, J. Morgan, A. Umpleby, N. Shah, V. Vinje, I. Štekl, L. Guasch, C. Win, G. Conroy, and A. Bertrand, 2013b, Anisotropic 3D full-waveform inversion: Geophysics, **78**.

Yang, D., M. Meadows, P. Inderwiesen, J. Landa, A. Malcolm, and M. Fehler, 2015, Double-difference waveform inversion: Feasibility and robustness study with pressure data: Geophysics, **80**.

Zhang, Z., and L. Huang, 2013, Double-difference elastic-waveform inversion with prior information for time-lapse monitoring: Geophysics, **78**, R259R273.

Zheng, Y., P. Barton, and S. Singh, 2011, Strategies for elastic full waveform inversion of time-lapse ocean bottom cable (OBC) seismic data, *in* 81st Annual International Meeting, SEG, Expanded Abstracts: 41954200.

APPENDIX A

A

```
!PARBASE25_7

!SEGYPREP
PROBLEM      : tomography
GEOMETRY     : segy
RECIPROCITY  : no
DEPTH TYPE   : fixed
UNIQUE       : no
FIXED ARRAY  : no
IO       : segy
DELAY SOURCE : -200.0

! Size and resolution of model
DX : 25.0
NX1   : 661    ! IL
NX2   : 1      ! XL
NX3   : 201    ! DEPTH

! Resolution of data
DTMS       : 2.0       ! TIMESTEP
TOTAL TIME : 6002.0


SOU DEPTH      : 5.0
SOU DX          : 25.0
SOU DY   : 1.0
SOU NX   : 521
SOU NY   : 1
SOU X ORIGIN   : 1000.0
SOU Y ORIGIN   : 0.0

REC DEPTH    : 10.0
REC GEOMETRY : source
REC DX        : 12.5
REC DY       : 0.0
REC NX       : 801
REC NY        : 1
REC X ORIGIN : 190.0
REC Y ORIGIN : 0.0

!RUNFILE

 ! Skeleton Runfile for FULLWAVE3D
 ! Generated on June 28, 2019, at 09:57:31
 ! By SEGYPREP, Version 3.10
 ! For project PARBASE25_7

 Problem      : tomography

 ! Problem type
 Units        : metric
 Equation     : acoustic
 Domain       : time
 Probdims     : 2D
 IO           : SEGY

! Model size
NX1     :         661
NX2     :           1
NX3     :         201
DX      :   25.000000

! Numbers of shots and receivers
NCOMP   :         521
NSHOT   :         521
MAXPS   :           1
NRECS   :      322149
MAXRC   :         801

! Free surface boundary at top, absorbing elsewhere
B top         : 100
B left        : 100
B right       : 100
B front       : 20
B back        : 20
B bottom      : 100

ETOP          : 100
EBOT          : 100
ELEF          : 100
ERIG          : 100

! Control density model
Water velocity : 1500
Water density  : 1000
Gardner cutoff : 1500

! Inversion parameters
Slowness             : yes
Normalise            : yes
Amplitude            : no
Spatial              : yes
Global Gaussian Width : 0

 ! Inversion contraints
Vel Cut Off : 1501.0  ! Original velocities below
    this are never changed
Vel Con Min : 1501.0  ! Velocity updates are
    clipped to this lower bound
Vel Con Max : 6237.5  ! Velocity updates are
    clipped to this upper bound

! Control shots per iteration - affects both
    runtime and quality
NumCStoSkip  : 4      ! Skip four shots, i.e. take
    every fifth
NumCSperIter : 104

 ! Iteration blocks
NumIterBlocks : 13

Frequency     : 3.0
Iterations    : 10
```

```
Frequency     : 3.4
Iterations    : 10

Frequency     : 3.9
Iterations    : 10

Frequency     : 4.5
Iterations    : 10

Frequency     : 5.2
Iterations    : 10

Frequency     : 6.0
Iterations    : 10

Frequency     : 6.9
Iterations    : 10

Frequency     : 7.9
Iterations    : 10

Frequency     : 9.0
Iterations    : 10

Frequency     : 10.2
Iterations    : 10

Frequency     : 11.5
Iterations    : 10

Frequency     : 12.9
Iterations    : 10

Frequency     : 14.4
Iterations    : 10

! All done
```

APPENDIX B

B

```
!PARBASE25_12

!SEGYPREP:
PROBLEM      : tomography
GEOMETRY     : segy
RECIPROCITY  : no
DEPTH TYPE   : fixed
UNIQUE       : no
FIXED ARRAY  : no
IO       : segy
DELAY DATA   : 200.0

! Size and resolution of model
DX : 25.0
NX1   : 661    ! IL
NX2   : 1        ! XL
NX3   : 201    ! DEPTH

! Resolution of data
DTMS       : 2.0       ! TIMESTEP
TOTAL TIME : 6002.0

SOU DEPTH      : 5.0
SOU DX          : 25.0
SOU DY   : 1.0
SOU NX   : 521
SOU NY   : 1
SOU X ORIGIN   : 1000.0
SOU Y ORIGIN   : 0.0

REC DEPTH    : 10.0
REC GEOMETRY : source
REC DX        : 12.5
REC DY       : 0.0
REC NX       : 801
REC NY        : 1
REC X ORIGIN : 190.0
REC Y ORIGIN : 0.0

!RUNFILE
 ! Skeleton Runfile for FULLWAVE3D
 ! Generated on August 02, 2019, at 17:12:24
 ! By SEGYPREP, Version 3.10
 ! For project PARBASE25_12

 Problem      : tomography

 ! Problem type
 Units        : metric
 Equation     : acoustic
 Domain       : time
 Probdims     : 2D
 IO           : SEGY

 ! Model size
```

```
NX1     :         661
NX2     :           1
NX3     :         201
DX      :   25.000000

! Numbers of shots and receivers
NCOMP   :         521
NSHOT   :         521
MAXPS   :           1
NRECS   :      322149
MAXRC   :         801

! Free surface boundary at top, absorbing elsewhere
B top         : 100
B left        : 100
B right       : 100
B front       : 20
B back        : 20
B bottom      : 100

ETOP          : 150
EBOT          : 100
ELEF          : 100
ERIG          : 100

! Add model smoothing in X1 and X3
SX1 = 1.3
SX3 = 0.5

! Control density model
Water velocity : 8000
Water density  : 1000

! Inversion parameters
Slowness             : yes
Normalise            : yes
Amplitude            : no
Spatial              : yes
Global Gaussian Width : 0

 ! Inversion contraints
Vel Cut Off : 1501.0  ! Original velocities below
    this are never changed
Vel Con Min : 1501.0  ! Velocity updates are
    clipped to this lower bound
Vel Con Max : 6237.5  ! Velocity updates are
    clipped to this upper bound

! Control shots per iteration - affects both
    runtime and quality
NumCStoSkip  : 4      ! Skip four shots, i.e. take
    every fifth
NumCSperIter : 104

 ! Iteration blocks
NumIterBlocks : 13

Frequency     : 3.0
Iterations    : 10

Frequency     : 3.4
Iterations    : 10

Frequency     : 3.9
Iterations    : 10

Frequency     : 4.5
Iterations    : 10

Frequency     : 5.2
Iterations    : 10

Frequency     : 6.0
Iterations    : 10

Frequency     : 6.9
Iterations    : 10

Frequency     : 7.9
Iterations    : 10

Frequency     : 9.0
Iterations    : 10

Frequency     : 10.2
Iterations    : 10

Frequency     : 11.5
Iterations    : 10

Frequency     : 12.9
Iterations    : 10

Frequency     : 14.4
Iterations    : 10

! All done
```

APPENDIX C

C

```
!PARBASE25_18

!SEGYPREP

PROBLEM      : tomography
GEOMETRY     : segy
RECIPROCITY  : no
DEPTH TYPE   : fixed
UNIQUE       : no
FIXED ARRAY  : no
IO       : segy
DELAY DATA   : 200.0

! Size and resolution of model
DX : 25.0
NX1   : 661    ! IL
NX2   : 1      ! XL
NX3   : 201    ! DEPTH

! Resolution of data
DTMS       : 2.0       ! TIMESTEP
TOTAL TIME : 6002.0


SOU DEPTH      : 5.0
SOU DX          : 25.0
SOU DY   : 1.0
SOU NX   : 521
SOU NY   : 1
SOU X ORIGIN   : 1000.0
SOU Y ORIGIN   : 0.0

REC DEPTH    : 10.0
REC GEOMETRY : source
REC DX        : 12.5
REC DY       : 0.0
REC NX       : 801
REC NY        : 1
REC X ORIGIN : 1190.0
REC Y ORIGIN : 0.0

!RUNFILE
 ! Skeleton Runfile for FULLWAVE3D
 ! Generated on August 09, 2019, at 09:38:14
 ! By SEGYPREP, Version 3.10
 ! For project PARBASE25_18

 Problem      : tomography

 ! Problem type
 Units        : metric
 Equation     : acoustic
 Domain       : time
 Probdims     : 2D
 IO           : SEGY

! Model size
NX1     :         661
NX2     :           1
NX3     :         201
DX      :    25.000000

! Numbers of shots and receivers
NCOMP   :         521
NSHOT   :         521
MAXPS   :           1
NRECS   :      322149
MAXRC   :         801

 ! Free surface boundary at top, absorbing elsewhere
B top         : 100
B left        : 100
B right       : 100
B front       : 20
B back        : 20
B bottom      : 100

ETOP          : 150
EBOT          : 100
ELEF          : 100
ERIG          : 100

! Add model smoothing in X1 and X3
SX1 = 1.3
SX3 = 0.25

! Control density model
Water velocity : 8000
Water density  : 1000

! Inversion parameters
Slowness             : yes
Normalise            : yes
Amplitude            : no
Spatial              : yes
Global Gaussian Width : 0

 ! Inversion contraints
Vel Cut Off : 1501.0  ! Original velocities below
    this are never changed
Vel Con Min : 1460.0  ! Velocity updates are
    clipped to this lower bound
Vel Con Max : 6237.5  ! Velocity updates are
    clipped to this upper bound

! Control shots per iteration - affects both
    runtime and quality
NumCStoSkip  : 4      ! Skip four shots, i.e. take
    every fifth
NumCSperIter : 104

 ! Iteration blocks
NumIterBlocks : 13
```

```
Frequency     : 3.0
Iterations    : 10

Frequency     : 3.4
Iterations    : 10

Frequency     : 3.9
Iterations    : 10

Frequency     : 4.5
Iterations    : 10

Frequency     : 5.2
Iterations    : 10

Frequency     : 6.0
Iterations    : 10

Frequency     : 6.9
Iterations    : 10

Frequency     : 7.9
Iterations    : 10

Frequency     : 9.0
Iterations    : 10

Frequency     : 10.2
Iterations    : 10

Frequency     : 11.5
Iterations    : 10

Frequency     : 12.9
Iterations    : 10

Frequency     : 14.4
Iterations    : 10

! All done
```