UNIVERSIDADE FEDERAL FLUMINENSE
INSTITUTO DE GEOCIÊNCIAS
DEPARTAMENTO DE GEOLOGIA E GEOFÍSICA

Ammir Ayman Karsou

Aceleração dos algoritmos de modelagem acústica e elástica e da inversão do campo de onda completo (FWI) em GPU utilizando OpenACC

DISSERTAÇÃO DE MESTRADO

PROGRAMA DE PÓS-GRADUAÇÃO DINÂMICA DOS OCEANOS E DA TERRA (DOT)

Niterói, RJ, Brasil

2020

Ammir Ayman Karsou

Aceleração dos algoritmos de modelagem acústica e elástica e da inversão do campo de onda completo (FWI) em GPU utilizando OpenACC

Dissertação apresentada à Universidade Federal Fluminense como requisito parcial do Programa de Pós-Graduação em Dinâmica dos Oceanos e da Terra para a obtenção do título de Mestre em Ciências.

Área de concentração: Geologia e Geofísica

Orientador
Prof. Marco Antonio Cetale Santos

Niterói, RJ, Brasil
2020

Ammir Ayman Karsou

Aceleração dos algoritmos de modelagem acústica e elástica e da inversão do campo de onda completo (FWI) em GPU utilizando OpenACC

Dissertação apresentada à Universidade Federal Fluminense como requisito parcial do Programa de Pós-Graduação em Dinâmica dos Oceanos e da Terra para a obtenção do título de Mestre em Ciências.

Aprovada em 29/12/2020 pela banca examinadora abaixo:

Prof. Marco Antonio Cetale Santos, DSc.
(Orientador)
UFF / DOT / GISIS

Bruno Pereira Dias, DSc.
Petrobrás

Prof. Luiz Alberto Santos, DSc.
UFF / GGO / GISIS

Prof. Roger Matsumoto Moreira, DSc.
UFF / GISIS

Niterói, RJ, Brasil
2020

Lista de ilustrações

Lista de tabelas

Índice de algoritmos

Lista de abreviaturas e siglas

FWI	*Full Waveform Inversion*
RTM	*Reverse Time Migration*
CPU	*Central Processing Unit*
GPU	*Graphic Processing Unit*
HPC	*High Performance Computing*
ABC	*Absorbing Boundary Condition*
PML	*Perfectly Matched Layers*
CPML	*Convolutional Perfectly Matched Layers*
SPML	*Split-Field Perfectly Matched Layers*
NPML	*Nearly Perfectly Matched Layers*
RAM	*Random Access Memory*
CUDA	*Compute Unified Device Architecture*
MDF	Método de Diferenças Finitas

Sumário

Agradecimentos

Quero primeiramente agradecer à pesssoa mais importante da minha vida, a mulher mais linda do mundo que atende pelo nome de Elisa Helena Crespo Bragança. Obrigado por ter me dado uma ótima criação, da qual teve 100 % de influência na minha vida acadêmica. Te amo muito! Agradecer também ao meu irmão Ibrahim Karsou, que apesar de todos os problemas hoje é meu parceiro. Ao meu pai Ayman Karsou, que me deu o suporte durante a minha graduação. À minha vó Miriam Ibrahim Karsou e minha tia Amany Karsou, que fizeram da minha infância a mais feliz do mundo, cheia de amor e carinho. Também quero agradecer às minhas mães postiças Miriam Aparecida e Anna Procópio, sou muito feliz em poder conviver com vocês e perturbar as duas com o meu jeito chato de ser. Aos meus pais postiços Ivan do Vale e Marcius Oliveira, que me apoiaram incondicionalmente no decorrer da minha vida. À minha tia Balquees Karsou mas que mais parece minha irmã, obrigado pelo apoio incondicional, mesmo quando eu desapareço por um tempo. Aos meus Abdallah e Mohammad Karsou. À minha vó Heloísa Bragança, que nos acolheu quando viemos para o Brasil. E quero agradecer todos da minha (grande) família, que tiveram uma interferência positiva, de maneira direta ou indireta. Aos meus amigos, os quais escolhi irmãos para a vida: Adelino Fontoura, Állan Soares, Cayo Fonseca, Diego Rodrigues, Eloíse Policarpo, Ewerton Lópes, Fellipe Ferreira, Flávio Louzada, Gabriel Câmara, Gabriel Matos, Henrique Nogueira, Ilson Filho, Israeli Rodrigo, Juan Carlos, Júlio Santana, João Boarato, Lael Filho, Lucas Navarro, Luiz Paulo, Maria Fernanda Alpoim, Maria Theresa Alpoim, Matheus Francis, Matheus Klatt, Pedro Campos, Rayan Thadeu, Thiago Araújo, Twanny Nunes, Vanessa Cunha, Victor Marum, Wanderson Souza e Yuri Nascimento, muito obrigado por me acompanharem durante a minha vida e vocês tem um lugar especial no meu coração. Agora quero agradecer aos que tornaram o meu mestrado possível, primeiramente ao meu orientador Marco Cetale por me dar a oportunidade de iniciar no mestrado e por me guiar no decorrer da pesquisa. Ao Jonatan Dias, por confiar no meu potencial e me indicar à vaga. Aos professores Luiz Alberto e Guilherme Hernandez por me confiarem as cartas de recomendação. À equipe do GISIS, do qual me orgulho muito de fazer parte, em especial aos meus mentores Felipe Capuzzo, Felipe Timóteo e Rodrigo Stern. Também para a Bruna Carbonesi, Danielle Tostes, Marília Carneiro e ao Roberto Miyamoto, vocês são sensacionais! Agradeço imensamente aos membros da banca examinadora Bruno Pereira, Luiz Alberto e Roger Matsumoto, por aceitarem o meu convite e avaliar o meu trabalho em uma data tão adversa. Agradeço à Universidade Federal Fluminense por me abrigar desde a graduação e agradeço à Petrobrás e Fundação Euclides da Cunha pelo financiamento do meu curso de mestrado. Agradeço a todos os amigos e colegas que não foram mencionados mas me ajudaram em alguma fase deste processo.

Um agradecimento especial vai à Luciana Ferreira, muito obrigado por acreditar no meu potencial, eu não teria conseguido sem sjua ajuda. Gratidão! E por fim, agradeço a mim por não desistir e continuar mesmo em meio as diversidades que me foram apresentadas. Concluir um curso de mestrado é um processo duro e desgastante, que demandou um pouco de minha saúde mental mas valeu a pena.

Resumo

O método sísmico é amplamente utilizado na exploração de óleo e gás. Utilizando técnicas de processamento e tomografia, consegue-se gerar modelos de velocidade capazes de gerar boas imagens de subsuperfície. Com a expansão das fronteiras exploratórias, necessita-se de modelos com maior resolução para imagear zonas de geologia complexa. A inversão FWI é uma técnica que consegue gerar modelos de alta resolução, sendo atualmente amplamente empregada na etapa de imageamento sísmico. Por ser um método iterativo, a solução depende da qualidade do modelo inicial e da convergência do método de otimização. A inversão FWI é uma técnica que exige alto tempo de computação e memória, em meio a este contexto, o trabalho desenvolve estratégias para otimizar o tempo computacional utilizando placas gráficas (GPUs). Devido à alta complexidade que as interfaces de paralelização podem oferecer, optou-se por utilizar o OpenACC, um modelo de programação baseado em diretivas para paralelizar as regiões mais custosas nos códigos de modelagem e inversão. Logo, são paralelizados os códigos de modelagens acústica, elástica e da inversão FWI. Os resultados são gerados utilizando os modelos de Marmousi e o modelo de Búzios. Estes mostram ganho de performance considerável, dando liberdade de se utilizar altas ordens de diferença finita sem perda de tempo de máquina. Também é mostrado que há ganhos de performance de até 37x para a quarta ordem no caso elástico. Os tempos de execução na geração dos dados sintéticos observados e da inversão FWI utilizando a GPU são medidos e comparados com as demais configurações, com estes valores sendo consideravelmente menores que os valores estimados utilizando arquiteturas em CPU. A inversão FWI foi capaz de recuperar feições geológicas que não eram visíveis no modelo inicial para o modelo de Marmousi, aumentando a resolução do modelo final. Para o modelo de Búzios, bons resultados são mostrados. Mesmo com o algoritmo de inversão não convergindo como no modelo de Marmousi, o modelo de velocidade recuperado mostra a maioria das feições complexas presentes no modelo real. As imagens finais migradas também de ambos os modelos mostram um melhor imageamento das estruturas geológicas, apresentando horizontes sísmicos mais contínuos e uma maior definição e detalhe destas estruturas em ambos os modelos.

Abstract

The seismic method is widely used in the oil and gas exploration. Using processing techniques and tomography, it is possible to obtain velocity models able to generate good images in a less complex geological setting. With the expansion of the exploratory borders, models with a higher resolution are needed to image zones of complex geology. The full waveform inversion (FWI) is a technique which can generate high resolution models, being highly employed in the seismic imaging stage. However, the full waveform inversion suffers of being a high computation cost technique. Given to the fact of being a iterative method, the solution depends on the quality of the initial guess and the convergence of the optimization method. The full waveform inversion is a technique that demands a high computation time and memory, in this context, the work develops strategies to optimize processing time using graphic processing units (GPUs). Due to the high complexity that the parallel interfaces can offer to the programmer, OpenACC was chosen as a programming model based on directives to parallelize the most computationally costly regions in the modeling and inversion algorithms. Therefore, the work parallelizes the acoustic and elastic modeling and the full waveform inversion codes. The results are generated using the Marmousi and Búzios geologic models. These results show considerable speedup, giving the freedom of choosing a high finite difference order with a low loss of computation time. It also shows that there are speedups of 37x with the fourth order in the elastic case. The execution times in the generation of the observed synthetic data and the full waveform inversion are measured using GPU and compared with the CPU, with these values being considerably lower than the ones estimated. The full waveform inversion was able to recover geologic features that were not visible in the initial guess for the Marmousi model, improving the quality of the velocity model and improving final migrated image. Using the Búzios model, good results are shown. Even though the inversion algorithm does not converge as in the Marmousi model, the velocity model retrieved shows most of the complex features present in the real model. The final migrated images of both models show a better imaging of the geologic structures, presenting more continuous seismic horizons and a better resolution and detail of these structures in both models.

1 Introdução

O método sísmico é amplamente empregado na exploração de hidrocarbonetos para imageamento de subsuperfície devido à sua alta resolução, confiabilidade em estimar parâmetros em profundidade de investigação. O mecanismo de aquisição sísmica na indústria petrolífera, essencialmente, consiste na injeção de ondas mecânicas num meio de subsuperfície, no qual parte do campo de onda é transmitido e parte refletido dentre as interfaces com contrastes de impedância, e registrados na superfície. Uma combinação de sensores sísmicos fica localizada em pontos estratégicos na superfície, de modo a ter suficientes traços com informações do meio para o processamento e imageamento sísmico. Com os sismógrafos na superfície, registram-se alguns segundos da informação do tempo e amplitude em superfície. Tais registros são respostas do campo de onda ao meio geológico, fornecendo informações sobre propriedades elásticas do meio geológico como velocidade compressional, velocidade cisalhante e densidade (SHERIFF; GELDART, 1995).

À medida que se explora campos petrolíferos em regiões geologicamente mais complexas, técnicas de processamento sísmico são necessariamente aprimoradas a fim de diminuir incertezas para produção. Em regiões com feições como falhamentos, dobras, domos de sal, intrusões de rochas ígneas e heterogeinidades de rochas como as carbonáticas, houve a necessidade de aprimoramento das técnicas de imageamento até então utilizadas (VIRIEUX; OPERTO, 2009; KALITA et al., 2019). Uma das técnicas que tem sido bastante estudada e aprimorada para o imageamento e modelos de velocidade foi a inversão do campo de onda completo ou *Full Waveform Inversion* (FWI). A FWI é uma técnica de imageamento de alta resolução, que utiliza todo o contéudo do registro sísmico (tempo de registro e amplitude) para gerar modelos de maior resolução. Lailly et al. (1983) e Tarantola (1984) desenvolveram esta técnica de modo a ser computacionalmente factível de ser implementada ao demonstrarem que o gradiente pode ser reduzido ao cálculo de um campo de onda descendente correlacionado com o campo de onda modelado.

Esta técnica tem sido aprimorada nas últimas décadas, sendo amplamente utilizada na indústria na melhoria dos modelos de velocidade. Porém, a inversão de campo de onda completo foi por muito tempo considerada uma técnica ineficiente de imageamento, uma vez que demonstrava maus resultados devido ao *cycle skipping* (salto de ciclo) (VIRIEUX; OPERTO, 2009). Como o problema é altamente não-linear, ao se utilizar um modelo de velocidade relativamente longe do ideal, a solução do problema se encontra em mínimos locais. Estes mínimos são criados por conta do salto de ciclo e eventos próximos no sismograma que são correlacionados, obtendo modelos de velocidade que não correspondem à Terra.

A solução mais empregada na indústria, que melhora os resultados das inversões e foi proposta por Bunks et al. (1995) é dividir o dado em bandas de frequências. Frequências mais baixas tem função objetivo mais suave e tem menos mínimos locais, com estes ficando relativamente distantes uns dos outros. Ao otimizar para cada banda de frequência, pode-se chegar cada vez mais próximo ao modelo de velocidade apropriado. Para seguir a abordagem multiescala, a inversão necessita de um bom modelo inicial contendo informações de baixa frequência. A qualidade da inversão por FWI pode ser afetada dependendo do arranjo da aquisição. Espaçamentos longos, azimutes e corberutas amplas contribem para maior resolução e precisão dos modelos. Aquisições sísmicas máritimas convencionais do Brasil se desenvolveram principalmente por aquisição de *Streamers*, que possuem cabos de 6-10km e azimute restrito à direção de aquisição do navio, o que pode limitar a qualidade do resultados. Os arranjos de OBN (*Ocean Bottom Nodes*) estão sendo cada vez mais utilizados para avanços de imageamento em regiões complexas como a Bacia de Santos. Isto se dá devido a seu arranjo mais amplo em *offsets*, azimutes e mais larga banda de frequência. Sirgue, Etgen e Albertin (2007) verificaram que aquisições do tipo *Narrow Azimuth* possuíam muitas limitações em ambientes de geologia complexa e demonstraram melhores resultados utilizando *Multi Azimuth*. Após os avanços de arranjos de aquisição e um maior conteúdo de frequência no dado sísmico, modelos podem ser gerados através da tomografia para servir de um bom modelo inicial. Modelos com altos comprimentos de onda a partir da tomografia (BISHOP et al., 1985; PRATT; GOULTY, 1991) ou modelos advindos da fase de análise de velocidade podem ser utilizados como entrada para a FWI.

Apesar de ser iniciada com bons modelos, a FWI tem alto custo computacional associado à convergência do método de otimização e o tempo para se calcular a equação da onda em todos os pontos do modelo dificultavam a sua aplicação. Com os avanços de tecnologias de *hardware* e HPC (*High Performance Computing* ou Computação de alta Performance), este problema pôde ser contornado. *Clusters* foram utilizados a partir dos anos 2000 e permaneceram como ferramenta principal no processamento e imageamento até os dias de hoje (BRANDSBERG-DAHL, 2017). Com o aumento da capacidade dos microprocessadores e a utilização de *clusters*, tornou-se viável o uso de técnicas mais sofisticadas e custosas como FWI. No entanto, a velocidade dos microprocessadores se estagnou devido à limitações tecnológicas. Para aumentar a velocidade das aplicações utilizadas necessitava-se aumentar o número de núcleos, tendo a necessidade de obter *clusters* cada vez maiores, mais caros e de maior custo de manutenção (BARLAS, 2014). Em meio a este contexto, uma alternativa para a aceleração de aplicações foi o uso de placas gráficas (*Graphic Processing Units* ou GPUs). Inicialmente usadas apenas como ferramentas para visualização gráfica, agora as GPUs passaram a servir como unidades de processamento auxiliares da CPU, sendo utilizadas em problemas de baixa complexidade lógica e alta intensidade aritmética (MISIC; DURDEVIC; TOMASEVIC, 2012). Desde então as GPUs são empregadas em diversos problemas, incluindo problemas do imageamento sísmico.

Porém, para se utilizar desta vantagem, era necessário aprender todo um novo paradigma de programação, a complexidade do entendimento e de implementação em placas gráficas era um impedimento para uma maior utilização no meio científico. O OpenACC é uma abordagem para contornar este tipo de problema, sendo um modelo de programação paralela baseado em diretivas que fornece portabilidade em distintas plataformas. Este foi inicialmente desenvolvido para cientistas e engenheiros obterem vantagem da computação de alta performance com menor esforço de programação, tornando mais acessível a utilização de GPUs nos problemas de simulações físicas. Baseado neste conceito, o objetivo deste trabalho é demonstrar o uso do OpenACC como modelo de programação em GPU para acelerar os códigos de modelagem acústica, modelagem elástica e inversão FWI. Desta forma, aumentando a eficiência de computação e diminuindo o tempo de máquina necessário para obter resultados de modelagens e inversões sísmicas. Complementarmente, este trabalho aborda a teoria matemática da inversão FWI, descreve arquiteturas paralelas e fornece um fluxo de paralelização em diretivas para otimizar o tempo de execução dos códigos de modelagem e inversão FWI.

O trabalho é organizado da seguinte forma: No capítulo 2 são detalhadas as questões do problema direto e inverso, explicando a equação da onda e descrevendo as formas mais comuns da minimização do problema sísmico. O problema inverso é abordado matematizando o problema de Newton, Gauss-Newton, cálculo das derivadas de Fréchet e o cálculo do gradiente via método adjunto. Também são feitas considerações sobre aspectos da modelagem e a inversão. No capítulo 3 é dada uma introdução aos algoritmos de otimização númerica, falando brevemente sobre os algoritmos mais comuns utilizados e detalhando como é feito o cálculo do passo para o algoritmo L-BFGS. No capítulo 4 é dada uma introdução à máquinas paralelas, comentando as diferenças físicas entre a CPU e a GPU. Depois são comentadas as diferentes diretivas do modelo de programação OpenACC e são dados exemplos de paralelização de *loops* utilizando diretivas de controle de dados e de computação. No capítulo 5 é descrita a geometria de aquisição e são comentadas as características principais dos modelos geológicos utilizados neste trabalho, sendo estes o modelo de Marmousi (VERSTEEG, 1994) e o modelo de Búzios (KARSOU et al., 2019). No capítulo 6 é apresentada a metodologia, onde são dados o fluxo de como a inversão FWI é feita no trabalho e o fluxo de paralelização dos códigos de modelagem e inversão. Também é comentado o fluxo de testes utilizado neste trabalho para avaliar o impacto da paralelização no tempo de execução e a qualidade da inversão FWI. Por fim, são descritos os *hardwares* utilizados para executar os códigos paralelizados, comentando sua capacidade teórica de computação e suas especificações. O capítulo 7 se destina a seção de resultados. Descrevendo assim os resultados e apresentando gráficos para a avaliação do ganho de performance e imagens de comparação do sucesso de obtenção de modelos de velocidade a partir da inversão, analisando a função objetivo e analisando o impacto de um modelo mais detalhado na migração reversa no tempo. No capítulo 8 é feita uma conclusão com

base nos resultados apresentados no capítulo 7. Para complementar, são anexados os apêndices A, B e C. Com o apêndice A descrevendo a teoria da discretização da equação da onda para os casos acústico e elástico, também descrevendo os critérios de instabilidade e dispersão levados em conta no trabalho para obter uma solução estável e confiável. Já o apêndice B se destina a dar uma breve explicação sobre a história e a formulação teórica e matemática do algoritmo da migração reversa no tempo. Por fim, o apêndice C apresenta os pseudocódigos para a paralelização dos algoritmos das modelagens acústica e elástica e da inversão FWI.

2 Inversão do campo de onda completo (FWI)

O algoritmo de inversão FWI consiste de um esquema de otimização não-linear, cujo principal objetivo é a obtenção de propriedades físicas do meio, através da comparação entre a modelagem do campo de ondas e dados sísmicos adquiridos. Portanto, faz-se necessário a apresentação da formulação de problemas direto e inverso (TARANTOLA, 2006).

2.1 Problema direto

O problema direto é definido como a previsão quantitativa de valores de dados d à partir de um modelo de parâmetros m. Desta forma, o problema direto se baseia em calcular o dado que deve ser observado à partir do modelo determinado. Ele pode ser representado por:

$$L(\bar{m}) = \bar{d}, \tag{2.1}$$

onde $L(\bar{m}) = \bar{d}$ é a notação para o conjunto de equações $L^i(m_1, m_2, ..) = d^i, i = 1, 2, 3...$ O operador L é denotado de operador direto ou operador de modelagem direta, representando a lei física do problema proposto. O vetor $\bar{m}$ são os parâmetros do modelo ou as propriedades físicas de um determinado meio. O vetor $\bar{d}$ é referente aos dados observados resultantes da interação com o modelo. No contexto da FWI acústica isotrópica, pode-se afirmar que:

1. Os parâmetros $\bar{m}$ são o modelo de propriedades: Velocidade compressional, velocidade cisalhante e densidade.
2. A modelagem de L simula a aquisição sísmica, gerando o conjunto de dados $\bar{d}$ de uma simulação ou da aquisição real do campo de onda através de um meio geológico de parâmetros $\bar{m}$.
3. A equação da onda é o operador de modelagem direta L.
4. Os dados observados são os sismogramas obtidos da aquisição simulada da modelagem.

O problema direto sísmico é considerado um problema bem posto, significando que a matriz de dados é suficiente para que o problema seja possível e determinado.

Este capítulo não se atém à discretização da equação da onda e tópicos específicos da modelagem. O apêndice A está dedicado para a discussão da discretização da equação da onda elástica, acústica e a abordagem do tamanho necessário de malha para respeitar os critérios de instabilidade e dispersão.

2.1.1 Equação da onda elástica

A lei física L que rege o problema direto no método sísmico é a equação da onda. A equação da onda é derivada da segunda lei de Newton e da lei de Hooke. Este é um sistema de equações que descreve ondas mecânicas em meios sólidos e líquidos a partir do deslocamento de partícula. A propagação da onda em um meio é influenciada por diversas propriedades físicas do meio, sendo estes os parâmetros elásticos, densidade, anisotropia e continuidade do meio. Neste trabalho, parte-se do pressuposto de continuidade do meio, isto é, que ele não possui poros e preenche todo o espaço que ocupa. Outro pressuposto é o de que o parâmetro elástico não possui dependência a um eixo preferencial, sendo igual para todas as direções do eixo cartesiano.

Uma formulação da equação da onda elástica é dada por Virieux (1986). Esta formulação supõe que o meio é bidimensional nos eixos x e z, somente propaga ondas compressionais e cisalhantes verticais (SV) e as tensões cisalhantes são iguais e independentes da direção. A equação em função do deslocamento de partícula e da tensão, Virieux (1986) expressa um sistema de equações diferenciais do tipo:

$$\begin{cases} \dfrac{\partial^2 u_x}{\partial t^2} = & b\left(\dfrac{\partial \sigma_{xx}}{\partial x} + \dfrac{\partial \sigma_{xz}}{\partial z}\right), \\ \dfrac{\partial^2 u_z}{\partial t^2} = & b\left(\dfrac{\partial \sigma_{zz}}{\partial z} + \dfrac{\partial \sigma_{xz}}{\partial x}\right), \\ \sigma_{zz} = & (\lambda + 2\mu)\dfrac{\partial u_z}{\partial z} + \lambda \dfrac{\partial u_x}{\partial x}, \\ \sigma_{xx} = & (\lambda + 2\mu)\dfrac{\partial u_x}{\partial x} + \lambda \dfrac{\partial u_z}{\partial z}, \\ \sigma_{xz} = & \mu\left(\dfrac{\partial u_z}{\partial x} + \dfrac{\partial u_x}{\partial z}\right), \end{cases} \tag{2.2}$$

com u_x e u_z sendo, respectivamente, o deslocamentos de partícula nos eixos x e y, σ_{xx} e σ_{zz} sendo as tensões normais e σ_{xz}, a tensão cisalhante, b sendo a flutuabilidade, que é o inverso da densidade e λ e μ são o módulo de rigidez e o módulo de cisalhamento, respectivamente. Este sistema pode ser modificado utilizando $v = \frac{\partial u}{\partial t}$. Logo, o sistema se transforma em:

$$\begin{cases} \dfrac{\partial v_x(\vec{x},t)}{\partial t} = & b\left(\dfrac{\partial \sigma_{xx}(\vec{x},t)}{\partial x} + \dfrac{\partial \sigma_{xz}(\vec{x},t)}{\partial z}\right), \\ \dfrac{\partial v_z(\vec{x},t)}{\partial t} = & b\left(\dfrac{\partial \sigma_{zz}(\vec{x},t)}{\partial z} + \dfrac{\partial \sigma_{xz}(\vec{x},t)}{\partial x}\right), \\ \dfrac{\partial \sigma_{zz}(\vec{x},t)}{\partial t} = & (\lambda+2\mu)\dfrac{\partial v_z(\vec{x},t)}{\partial z} + \lambda\dfrac{\partial v_x(\vec{x},t)}{\partial x}, \\ \dfrac{\partial \sigma_{xx}(\vec{x},t)}{\partial t} = & (\lambda+2\mu)\dfrac{\partial v_x(\vec{x},t)}{\partial x} + \lambda\dfrac{\partial v_z(\vec{x},t)}{\partial z}, \\ \dfrac{\partial \sigma_{xz}(\vec{x},t)}{\partial t} = & \mu\left(\dfrac{\partial v_z(\vec{x},t)}{\partial x} + \dfrac{\partial v_x(\vec{x},t)}{\partial z}\right), \end{cases} \tag{2.3}$$

onde v_x e v_z são a velocidade de partícula em seus respectivos eixos. Tal simplificação é feita para transformar o sistema hiperbólico de segunda ordem em primeira ordem e facilitar o cálculo das derivadas parciais. Estas equações descrevem o comportamento das ondas P e SV em duas dimensões.

2.1.2 Equação da onda acústica

Devido à sua simplicidade e baixo custo computacional e menor necessidade de memória, o trabalho utiliza A equação da onda acústica de segunda ordem com densidade constante, uma vez que o campo de pressão é calculada apenas uma vez e necessita de menos variáveis para armazenar as informações dos campos e de parâmetros.

A equação da onda acústica de segunda ordem pode ser descrita por:

$$\frac{1}{v^2(\vec{x})}\frac{\partial^2 P(\vec{x},t)}{\partial t^2} - \nabla^2 P(\vec{x},t) = R(t)\delta(\vec{x}-\vec{x_s}), \tag{2.4}$$

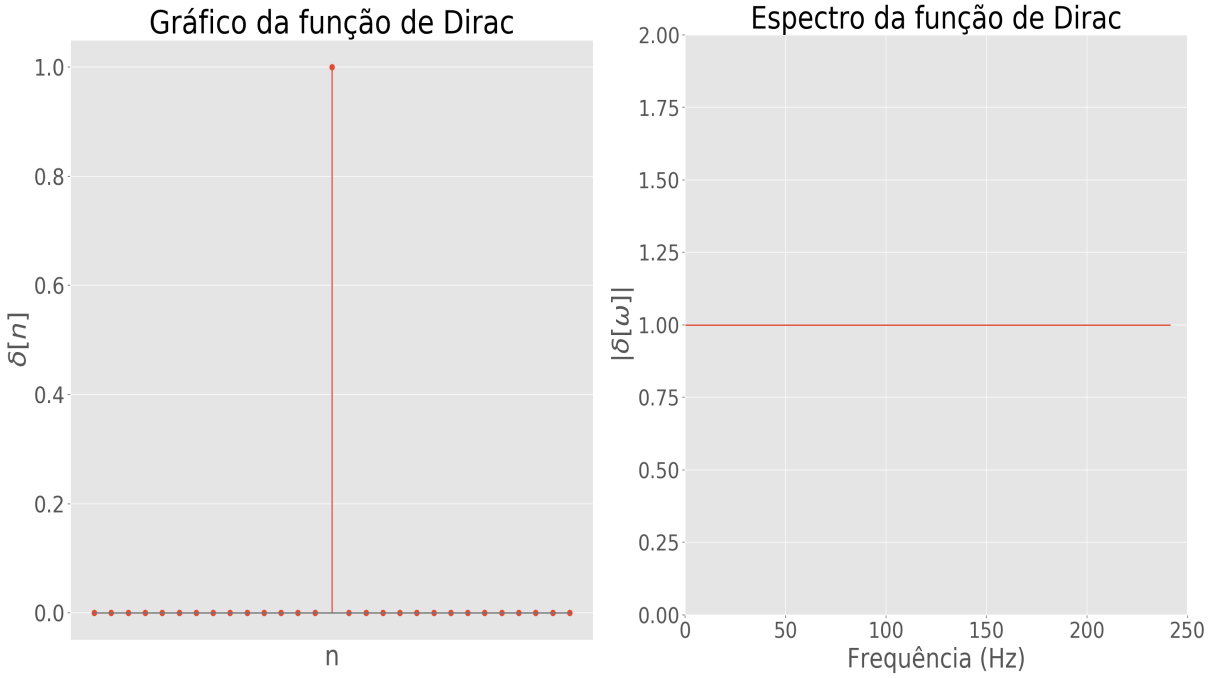
com $P(\vec{x},t)$ sendo a pressão em cada ponto do modelo, $v(\vec{x})$ a velocidade de propagação da onda acústica no meio e $R(t)\delta(\vec{x}-\vec{x_s})$ o termo fonte em cada posição dos tiros.

2.1.3 Fonte Sísmica

A fonte ideal para todas as aquisições é a fonte na forma da função de Dirac, onde a equação é descrita por:

$$\int_{-\infty}^{\infty} \delta(x)dx = 1. \tag{2.5}$$

A figura 1 contém o gráfico da função de Dirac e seu espectro. Analisando a função, pode-se perceber que ela tem valor diferente de zero somente em um instante de tempo, sendo nula para o restante. Ao verificar seu espectro, pode-se perceber que esta tem espectro branco, ou seja, possui valores não nulos e iguais para todas as frequências.



(a) (b)

Figura 1 – a) Gráfico da amplitude da função discreta do delta de Dirac. b) Espectro do pulso, tendo amplitude igual para todas as frequências do pulso.

No entanto, o conceito do delta de Dirac se baseia em emitir toda a energia em um único instante de tempo, algo fisicamente impossível. Logo, uma abordagem adotada é utilizar a função Gaussiana ou suas derivadas com relação ao tempo. A sua função é dada por:

$$G(t) = \frac{A_0}{2\pi(\pi f_c)^2} \exp\left(-\pi(\pi f_c)^2\ t\right), \tag{2.6}$$

onde A_0 é um fator de amplitude da fonte, f_c é um parâmetro de referência que é função da frequência de corte. O termo frequência de corte pode ser utilizado de diversas maneiras. Na área de processamento de sinais, refere-se à frequência de corte como o valor correspondente a $-3d\beta$ com relação ao valor de referência. Logo, f_c se define por:

$$f_c = \frac{f_{corte}}{3\sqrt{\pi}}, \tag{2.7}$$

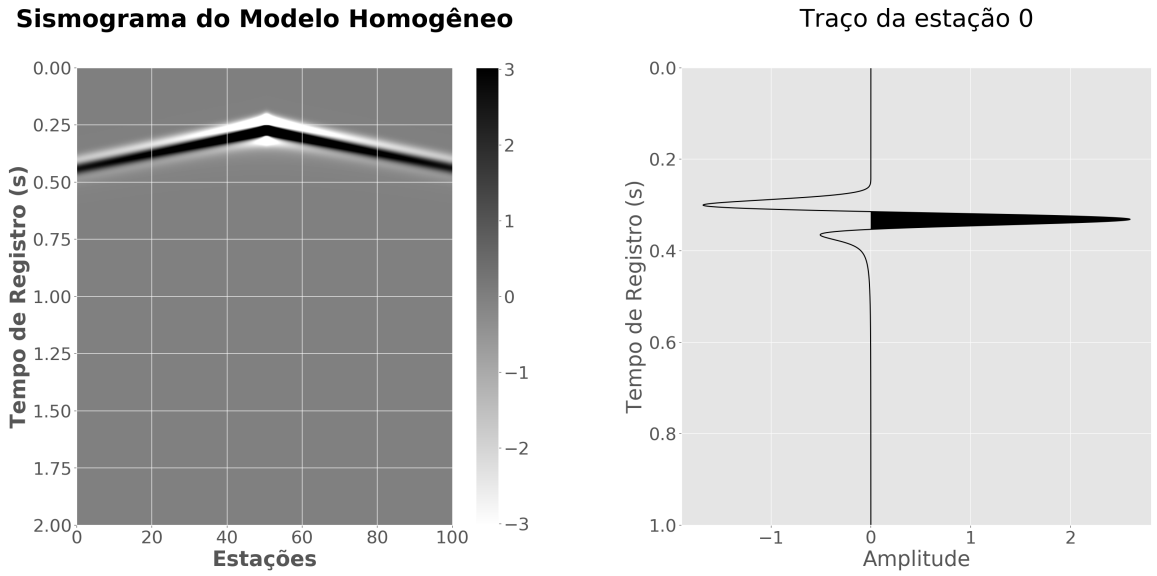
com f_{corte} sendo a frequência de corte. A partir da função Gaussiana, pode-se obter a suas derivadas, representadas pelas funções:

$$G'(t) = -A_0\ 2(\pi(\pi f_c)^2)\ t\ \exp\left(-\pi(\pi f_c t)^2\right), \tag{2.8}$$

e

$$R(t) = A_0(2\pi(\pi f_c t)^2 - 1)\exp\left(-\pi(\pi f_c t)^2\right), \tag{2.9}$$

onde $R(t)$ é a função *Ricker* (RICKER, 1953). A primeira derivada da função Gaussiana é utilizada para o caso elástico e é considerada como uma fonte explosiva, sendo injetada nas tensões normais seguindo a formulação proposta por Virieux (1986). Já a função *Ricker* é utilizada para o caso acútico, sendo injetada no campo de pressão. A figura 2 ilustra o sismograma gerado em um modelo homogêneo e um traço deste sismograma.



(a) (b)

Figura 2 – a) Sismograma gerado a partir de um tiro na metade do modelo, demonstrando uma diferença de amplitudes com relação aos vales da *wavelet*. b) Traço extraído da estação 0.

2.1.4 Correção de fase e amplitude de sismogramas

Apesar das fontes serem simétricas, a forma de onda encontrada no sismograma não é simétrica. Este fenômeno ocorre pois o espalhamento geométrico não é esférico para o caso 2D, mas cilíndrico. Mesmo ao injetar uma fonte simétrica e sem fase, o sismograma resultante da modelagem terá uma forma de onda defasada com relação ao seu centro, como mostra a figura 2. Uma forma de adaptar o sismograma é utilizando o método proposto por Pica, Diet e Tarantola (1990). O método consiste em aplicar uma correção de fase e amplitude ao sismograma, convolvendo-o por $\frac{1}{\sqrt{t}}$ para corrigir a fase e o multiplicando por $\sqrt{t}$ para corrigir a amplitude dos sismograma 3D para o sismograma 2D. Neste trabalho, faz-se o caminho inverso, corrigindo a fase e amplitude dos sismogramas 2D para os 3D.

Antes da aplicação da correção de amplitude, equação de correção dos sismogramas é dada por:

$$\phi^{3D}(t,\vec{x}) \circledast \frac{1}{\sqrt{t}} = \phi^{2D}(t,\vec{x}), \tag{2.10}$$

onde S^{2D} e S^{3D} são os sismogramas 2D e 3D, respectivamente. Para simplificar o cálculo, aplica-se a transformada de Fourier para passar do domínio do tempo para o domínio da frequência e aplicar a correção de fase:

$$\Phi^{3D}(\omega,\vec{x})\mathcal{F}\left\{\frac{1}{\sqrt{t}}\right\} = \Phi^{2D}(\omega,\vec{x}), \tag{2.11}$$

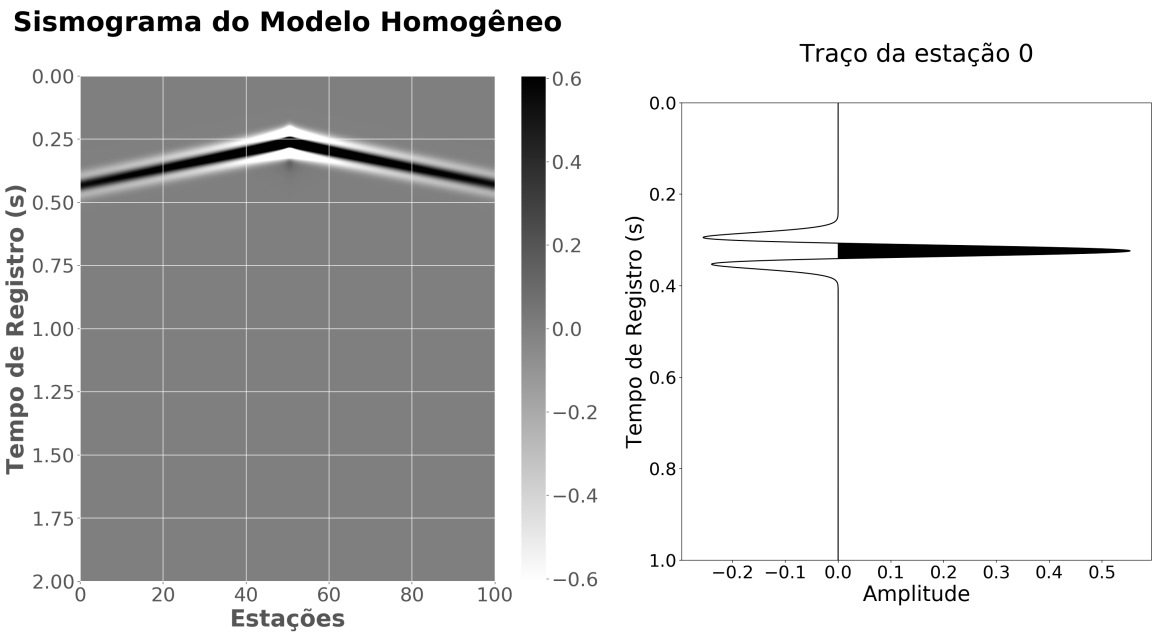
onde $\mathcal{F}\left\{\frac{1}{\sqrt{t}}\right\}$ é a transformada de Fourier da funrção $\frac{1}{\sqrt{t}}$. Com isso, pode-se encontrar a equação que corrige a fase do sismograma da forma:

$$\Phi^{3D}(\omega,\vec{x}) = \frac{\Phi^{2D}(\omega,\vec{x})}{\mathcal{F}\left\{\frac{1}{\sqrt{t}}\right\}}. \tag{2.12}$$

Calculando a transformada inversa e corrigindo a amplitude, obtém-se a equação:

$$\phi^{3D}(t,\vec{x}) = \mathcal{F}^{-1}\left(\frac{\Phi^{2D}(\omega,\vec{x})}{\mathcal{F}\left\{\frac{1}{\sqrt{t}}\right\}}\right)\ \frac{1}{\sqrt{t}}. \tag{2.13}$$

Este processo é feito traço a traço no sismograma, demandando um custo computacional adicional. Uma forma de diminuir este custo é aplicar o processo de correção de fase na fonte sísmica e a correção de amplitude na modelagem. Ao fazer desta forma, não se torna necessário o ajuste para todos os sismogramas. Este processo também é utilizado para o caso elástico. A figura 3 mostra o sismograma e a wavelet resultante da correção aplicada usando a equação 2.13.



(a) (b)

Figura 3 – a) Sismograma gerado a partir de um tiro na metade do modelo, já com a diferença de fase e amplitude nos vales corrigida. b) Traço extraído da estação 0.

2.1.5 Bordas absortivas

Um dos problemas da simulação numérica é que o modelo utilizado é apenas um recorte na área de interesse da Terra. Em uma aquisição real, ao explodir a fonte em campo, a onda propaga-se até se dissipar ao longo do meio geológico, como o caso de um modelo infinito. Já em uma aquisição numérica, seria proibitivo considerar um meio infinito. Portanto, ao se criar um modelo, o campo propagado será refletido nos limites do mesmo. Isto ocorre pois a simulação é conservativa, preservando a energia do sistema. Para contornar este problema, bordas não reflexivas são necessárias para eliminar eventos do sismograma que não correspondem às reflexões oriundas da heterogeneidade do modelo utilizado.

Uma alternativa para contornar este problema é extrapolar o modelo para todas as direções, levando em conta o tempo de aquisição e a velocidade do modelo, sendo chamado de borda infinita. Desta maneira, a reflexão gerada pela borda do modelo não chega aos receptores durante a aquisição. Tal método não é o mais eficiente pois necessitaria de um número altíssimo de pontos de borda, tornando alto o custo computacional e a capacidade de memória para armazenar a informação nos pontos extras deste modelo.

Com a necessidade de uma borda absortiva eficiente, Cerjan et al. (1985) propuseram multiplicar um fator à amplitude, fazendo o valor da amplitude decair exponencialmente

ao longo da borda. Este fator é dado pela seguinte função:

$$d(i) = \exp\left(-f\ (N-i)^2\right), \tag{2.14}$$

onde i é a posição na borda absortiva, N é o numero de pontos da borda e f é um fator referente ao poder de absorção da borda. O valor de f depende do número de pontos da borda utilizados, no trabalho de Cerjan et al. (1985) foi proposto um valor de $0,015$ para 20 pontos de borda. Bording (2005) analisou valores de número de camadas absortivas de 20 a 60 pontos e valores de f variando de $0,0005$ a $0,0100$. Ao analisar os valores de contorno de pseudo-energia, Bording (2005) concluiu que há um mínimo de energia ao utilizar 45 pontos e um fator de $0,0053$. Estes valores de números de borda e fator de amortecimento são calculados em um contexto de baixo poder computacional, com a necessidade de poupar memória e tempo de processamento.

Com o intuito de diminuir os efeitos de bordas para o eletromagnetismo, Berenger (1994) apresentou as Camadas perfeitamente ajustadas ou *Perfectly Matched Layers* (PML). A PML tem se mostrado muito efetiva para efeitos acústicos e elásticos isotrópicos. A teoria da PML prova que antes de haver a discretização, não há reflexão na região da borda para o modelo, absorvendo todas as ondas independentemente da frequência e do ângulo de incidência (figura 4).

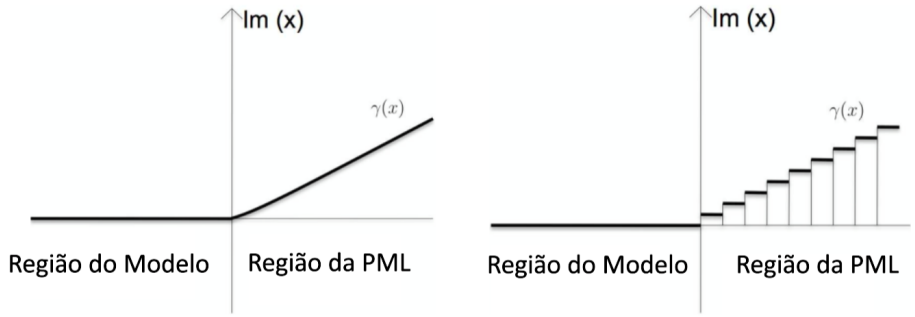


Figura 4 – Ilustração do crescimento do fator de amortecimento (*damping*). À esquerda o fator de amortecimento da PML cresce suavemente, inibindo reflexões advindas da borda. À direita, devido à discretização para simulações numéricas, a função de amortecimento tem passos entre seus valores. Modificado de Guasch (2011).

No entanto, tal característica é perdida durante a discretização, especialmente para ângulos oblíquos, tendo considerável retorno de energia nas quinas das bordas. Para contornar estes problemas, adaptações como a *Split-Field Perfectly Matched Layers* (SPML) (CARCIONE; HERMAN; KROODE, 2002), *Nearly Perfectly Matched Layers* (NPML) (MCGARRY; MOGHADDAM, 2009; METIN, 2013) e *Convolutional Perfectly Matched Layers* (CPML) (RODEN; GEDNEY, 2000; KOMATITSCH; MARTIN, 2007; PASALIC; MCGARRY, 2010) foram desenvolvidas para a equações da onda elástica e acústica com variados graus de sucesso. A SPML possui alta eficácia, porém necessita atualizar o cálculo

das variáveis auxiliares em todo o modelo, aumentando o custo computacional do método. Já a NPML existe apenas nas bordas absortivas, diminuindo o custo computacional mas esta apresentou alta instabilidade para altos tempos de modelagem (WANG et al., 2013). Diferentemente dos outros métodos, a CPML se mostrou muito eficiente na atenuação da energia de borda e estável para altos passos de tempo de modelagem, sendo o tipo de PML utilizado neste trabalho. A formulação da CPML se dá no princípio de extensão do sistema de coordenadas, utilizando adicionalmente o parâmetro α_k para controlar o conteúdo de baixa frequência e reflexões com alto ângulo de incidência. A extensão do sistema de coordenadas é dado no domínio da frequência por:

$$S_k = 1 + \frac{\sigma_k}{\alpha_k + i\omega}, \tag{2.15}$$

onde S_k é o parâmetro de extensão do sistema de coordenadas, σ_k é o fator de amorcecimento da borda, $k \in \{x, z\}$, ω a frequência angular e $i^2 = -1$. O parâmetro é multiplicado em cada derivada parcial espacial, ao aplicar a transformada de Fourier inversa, obtém-se a derivada parcial alongada da forma:

$$\frac{\partial}{\partial k*} = \mathcal{F}^{-1}\left\{\frac{1}{S_k}\right\}\frac{\partial}{\partial k}. \tag{2.16}$$

Ao fazer a transformada inversa de Fourier de S_k, obtém-se a solução da forma:

$$\mathcal{F}^{-1}(t) = \delta(t) - \sigma_k \exp\left(-(\sigma_k + \alpha_k)t\right) U(t), \tag{2.17}$$

onde $\delta(t)$ é o delta de Dirac, $U(t)$ é a função degrau e σ_k é um fator de amortecimento dado por Collino e Tsogka (2001) e α_k por Roden e Gedney (2000) nas equações:

$$\sigma_k(i) = -\frac{(M+1)}{2\ N\ d_k} v(i) \log(R_t) \left(\frac{i}{N}\right)^M, \tag{2.18}$$

$$\alpha_k(i) = \pi f_{pico}\left(1 - \left(\frac{i}{N}\right)^2\right), \tag{2.19}$$

onde N é o número de pontos da borda, i é a posição na borda $1 < i < N$, d_k é o espaçamento da direção do k-ésimo eixo, $v(i)$ é a velocidade da propagação da onda no ponto da borda, R_t é o coeficiente de refletividade da borda e f_{pico} é a frequência de pico do sinal. Apesar do sistema analítico estar bem resolvido, a convolução possui uma solução alternativa mais eficiente do que a tradicional forma analítica. A alternativa mais eficiente proposta por Roden e Gedney (2000) é a utilização de uma variável de memória, que é

atualizada para cada passo de tempo na borda. A equação para convolução recursiva se transforma em:

$$\frac{\partial P}{\partial k*} = \frac{\partial P(\vec{x},t)}{\partial k} + \psi_k, \tag{2.20}$$

onde ψ_k é uma variável de memória, resultante do alongamento das coordenadas em cada eixo de coordenadas das bordas absortivas e $\frac{\partial}{\partial k}$ é um operador de derivada parcial para cada eixo, com $k \in \{x, z\}$. A variável de memória é dada pela equação:

$$\psi_k^n = a_k \psi_k^{n-1} + b_k \left(\frac{\partial P(\vec{x},t)}{\partial k} \right)^n, \tag{2.21}$$

a_k e b_k são coeficientes que multiplicam a variável de memória e o campo de pressão no ponto, cada eixo coordenado possui seu determinado coeficiente. Estes coeficientes podem ser obtidos pelas equações:

$$a_k = \exp\left(-(\sigma_k + \alpha_k)\ \Delta t\right) \tag{2.22}$$

e

$$b_k = \frac{\sigma_k}{(\sigma_k + \alpha_k)}(a_k - 1). \tag{2.23}$$

Com o conceito de alongamento do sistema de coordenadas, a equação da onda na borda passa a ficar da seguinte maneira:

$$\frac{1}{v^2(\vec{x})}\frac{\partial^2 P(\vec{x},t)}{\partial t^2} = \frac{\partial^2 P(\vec{x},t)}{\partial x_*^2} + \frac{\partial^2 P(\vec{x},t)}{\partial z_*^2}, \tag{2.24}$$

como o alongamento de coordenada é feito na derivada de segunda ordem, a equação 2.20 tem que ser replicada duas vezes, havendo a necessidade do cálculo de mais uma variável de memória. Desta forma, equação 2.24 se transforma em:

$$\frac{1}{v^2(\vec{x})}\frac{\partial^2 P(\vec{x},t)}{\partial t^2} = \frac{\partial^2 P(\vec{x},t)}{\partial x^2} + \frac{\partial^2 P(\vec{x},t)}{\partial z^2} + \frac{\partial \psi_x}{\partial x} + \frac{\partial \psi_z}{\partial z} + \xi_x + \xi_z, \tag{2.25}$$

onde ξ_x e ξ_z são produtos da segunda aplicação do processo de convolução recursiva por conta da derivada de segunda ordem. Os novos campos auxiliares são dados pela equação:

$$\xi_k^n = a_k \xi_k^{n-1} + b_k \left(\frac{\partial \psi_k}{\partial k} + \frac{\partial^2 P(\vec{x},t)}{\partial k^2} \right)^n. \tag{2.26}$$

Um fato interessante da borda CPML é similaridade da equação na borda com a equação da onda, sendo apenas acrescentadas variáveis de memória em cada região específica da borda como mostra a figura 5.

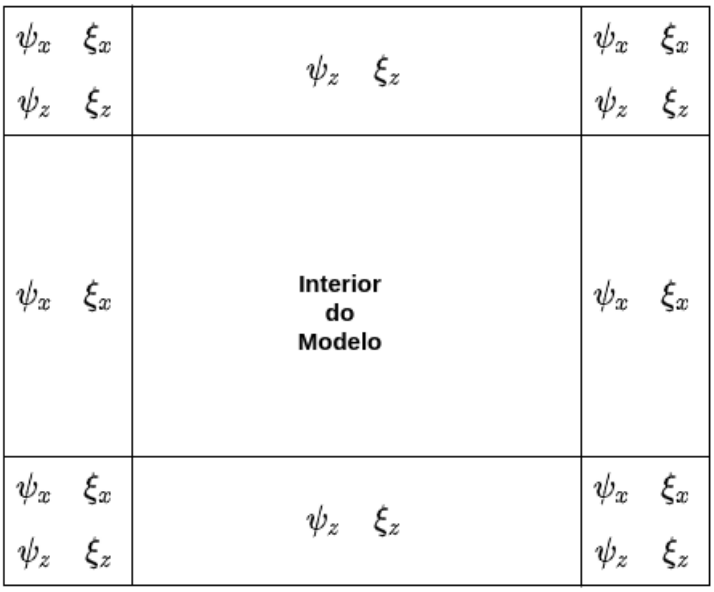


Figura 5 – Ilustração das variáveis de memória e sua respectiva localização na borda absortiva. Modificado de Liu e Tao (1997).

Apesar de ter quatro variáveis auxiliares além das varíaveis de campos de pressão, as variáveis da CPML apenas existem na borda absortiva, o que diminue o custo computacional para o cálculo do campo resultante. Outro detalhe importante é que a equação é completa apenas nas quinas do modelo como mostra a figura 5, quando a equação está no domínio x das bordas. A equação neste domínio se torna:

$$\frac{1}{v^2(\vec{x})}\frac{\partial^2 P(\vec{x},t)}{\partial t^2} = \frac{\partial^2 P(\vec{x},t)}{\partial x^2} + \frac{\partial^2 P(\vec{x},t)}{\partial z^2} + \frac{\partial \psi_x}{\partial x} + \xi_x. \tag{2.27}$$

2.2 Problema inverso

Ao passo que a modelagem age como o problema direto, o método de FWI é um problema inverso, ou seja, busca a partir dos dados e da lei física do problema estimar o vetor de parâmetros do modelo. Uma vez que a equção 2.1 é um problema direto, um problema inverso poderia ter simplesmente a forma:

$$\bar{m} = L^{-1}\bar{d}. \tag{2.28}$$

No entanto, a inversão de dados sísmicos é um problema altamente não-linear e mal posto, com infinitas soluções, o que inviabiliza o cálculo da matriz L^{-1}. Para resolver

o problema inverso neste contexto, usa-se as informações a priori para gerar um modelo inicial mais realista a fim de diminuir a não-linearidade do problema.

De maneira similar aos problemas de regressão linear, a inversão FWI busca minimizar o quadrado da diferença entre o dado registrado $\vec{d}$ e o dado sintético obtido por um modelo inicial $\vec{m}_0$. O objetivo é obter um modelo de propriedades tal que o dado sintético seja o mais próximo possível do dado registrado. Para tal, define-se a função a ser minimizada, chamada função objetivo ou funcional em função de $\Gamma = (\frac{1}{v})^2$:

$$r(\vec{\Gamma}) = \frac{1}{2} \sum_{s=1}^{Ns} \sum_{t=1}^{Nt} \sum_{r=1}^{Nr} (P_{obs}(\vec{x}, t) - P_{calc}(\vec{x}, t; \vec{\Gamma}))^2 \tag{2.29}$$

onde Ns é o numero de fontes, Nt é o numero de amostras de tempo da simulação, Nr é o numero de receptores utilizados, $\vec{x}$ é o vetor de dimensões do modelo, $\vec{\Gamma}$ é o vetor do modelo de vagarosidade ao quadrado, t é o tempo, P_{obs} é o campo de onda captado no sismograma da aquisição e P_{calc} é o campo de onda captado no sismograma calculado. A função escolhida é a mesma utilizada nos problemas de física clássica, uma vez que a mesma é contínua, diferenciável e de fácil manipulação e foi escolhida em função de $\vec{\Gamma}$ pois esta oferece um cálculo de gradiente da função mais conveniente para o trabalho.

O método FWI busca modelar sismogramas sintéticos com a mesma amostragem, posição de tiros e receptores e minimizar sua diferença com o dado registrado. Uma forma de obter um vetor de parâmetros que minimiza a função objetiva é por:

$$\vec{\Gamma} = \vec{\Gamma}_0 - \alpha \, \nabla r(\vec{\Gamma}_0), \tag{2.30}$$

onde α é um número real ou uma aproximação da matriz Hessiana, $\vec{\Gamma}_0$ é um modelo inicial de vagarosidade ao quadrado e $\nabla r(\vec{\Gamma}_0)$ é o gradiente da função objetivo, um vetor que seu negativo aponta para a direção de maior decréscimo da função. Esta equação é a equação atualiza os parâmetros. O cálculo de α é melhor comentado nas seções seguintes.

2.3 Método de Newton

O método de Newton é um algoritmo que busca iterativamente encontrar o mínimo de uma função. O método consiste em aproximar iterativamente a função objetivo (azul) a uma parábola (caso 2D) ou um parabolóide em 3D (verde), até atingir o mínimo da função aproximada. O processo de encontrar a direção de Newton é mostrado na figura 6.

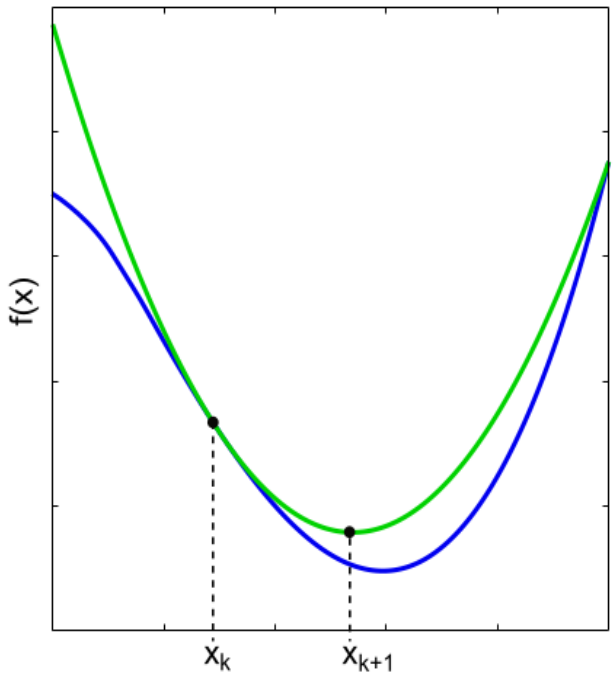


Figura 6 – Ilustração da convergência do método de Newton. A função objetivo (azul) é aproximada por uma parábola (verde).

Porém, a função objetivo não é analiticamente conhecida. Uma solução é fazer a expansão em série de Taylor da função, a qual é uma ótima aproximação local da função objetivo. Fazendo a expansão em torno de $\vec{\Gamma}_0$, tem-se:

$$r(\Gamma) = r(\Gamma_0) + \sum_{k=1}^{N} \frac{\partial r(\Gamma_{0k})}{\partial \Gamma_k}(\Gamma_k - \Gamma_{0k}) + \frac{1}{2} \sum_{k=1}^{N}\sum_{l=1}^{N} \frac{\partial^2 r(\Gamma_0)}{\partial \Gamma_k \partial \Gamma_l}(\Gamma_k - \Gamma_{0k})(\Gamma_l - \Gamma_{0l}) + O^3, \quad (2.31)$$

onde N é o número de termos do modelo de velocidade e O^3 é o erro de ordem 3 das derivadas parciais. Para encontrar um mínimo, deve-se fazer a derivada parcial ser nula. Aplicando a derivada parcial na função objetivo em função do parâmetro m, obtém-se:

$$\begin{aligned} \frac{\partial r(\Gamma)}{\partial \Gamma_m} = \frac{\partial r(\Gamma_0)}{\partial \Gamma_m} &+ \sum_{k=1}^{N} \frac{\partial^2 r(\Gamma_0)}{\partial \Gamma_k \partial \Gamma_m}(\Gamma_k - \Gamma_{0k}) + \frac{\partial r(\Gamma_0)}{\partial \Gamma_m} \\ &+ \frac{1}{2} \sum_{k=1}^{N}\sum_{l=1}^{N} \frac{\partial^3 r(\Gamma_0)}{\partial \Gamma_k \partial \Gamma_l \partial \Gamma_m}(\Gamma_k - \Gamma_{0k})(\Gamma_l - \Gamma_{0l}) \\ &+ \frac{1}{2} \sum_{k=1}^{N} \frac{\partial^2 r(\Gamma_0)}{\partial \Gamma_k \partial \Gamma_m}(\Gamma_k - \Gamma_{0k}) + \frac{1}{2} \sum_{l=1}^{N} \frac{\partial^2 r(\Gamma_0)}{\partial \Gamma_m \partial \Gamma_l}(\Gamma_l - \Gamma_{0l}). \end{aligned} \quad (2.32)$$

Ao truncar a série de Talyor na terceira ordem e considerando uma função contínua

e diferenciável, assume-se que:

$$\frac{\partial^3 r(\Gamma_0)}{\partial\Gamma_k \partial\Gamma_l \partial\Gamma_m} = 0 \tag{2.33}$$

e

$$\frac{\partial^2 r(\Gamma_0)}{\partial\Gamma_k \partial\Gamma_m} = \frac{\partial^2 r(\Gamma_0)}{\partial\Gamma_m \partial\Gamma_l}. \tag{2.34}$$

Logo, somando os fatores semelhantes, fazendo $(\Gamma_k - \Gamma_{0k}) = \Delta\Gamma_k$, usando a mesma base de somatórios e igualando a zero obtém-se:

$$\sum_{k=1}^{N} \frac{\partial^2 r(\Gamma_0)}{\partial\Gamma_k \partial\Gamma_m} \Delta\Gamma_k = -\frac{\partial r(\Gamma_0)}{\partial\Gamma_m}. \tag{2.35}$$

Ao expandir o conjunto de equações 2.35, esta se torna:

$$\begin{cases} \frac{\partial^2 r(\Gamma_0)}{\partial\Gamma_1 \partial\Gamma_1}\Delta\Gamma_1 + \frac{\partial^2 r(\Gamma_0)}{\partial\Gamma_2 \partial\Gamma_1}\Delta\Gamma_2 + ... + \frac{\partial^2 r(\Gamma_0)}{\partial\Gamma_N \partial\Gamma_1}\Delta\Gamma_N = -\frac{\partial r(\Gamma_0)}{\partial\Gamma_1}, \\ \frac{\partial^2 r(\Gamma_0)}{\partial\Gamma_1 \partial\Gamma_2}\Delta\Gamma_1 + \frac{\partial^2 r(\Gamma_0)}{\partial\Gamma_2 \partial\Gamma_2}\Delta\Gamma_2 + ... + \frac{\partial^2 r(\Gamma_0)}{\partial\Gamma_N \partial\Gamma_2}\Delta\Gamma_N = -\frac{\partial r(\Gamma_0)}{\partial\Gamma_2}, \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ \frac{\partial^2 r(\Gamma_0)}{\partial\Gamma_1 \partial\Gamma_N}\Delta\Gamma_1 + \frac{\partial^2 r(\Gamma_0)}{\partial\Gamma_2 \partial\Gamma_N}\Delta\Gamma_2 + ... + \frac{\partial^2 r(\Gamma_0)}{\partial\Gamma_N \partial\Gamma_N}\Delta\Gamma_N = -\frac{\partial r(\Gamma_0)}{\partial\Gamma_N}. \end{cases}$$

O sistema da equação pode ser escrito de forma matricial, onde cada derivada parcial de segunda ordem representa o elemento da matriz Hessiana H_{ij}, $\Delta\Gamma_j$ representa cada elemento do vetor de vagarosidade ao quadrado e $\frac{\partial r(\Gamma_0)}{\partial\Gamma_i}$ representa cada elemento do vetor gradiente. Desta forma, escreve-se a equação da forma:

$$H\Delta\vec{\Gamma} = -\nabla r(\vec{\Gamma}_0), \tag{2.36}$$

onde H é a matriz Hessiana da função objetivo, $\Delta\vec{\Gamma}$ é o vetor de diferença do inverso da velocidade ao quadrado e $\nabla r(\vec{\Gamma}_0)$ representa o gradiente da função objetivo. Assumindo que a matriz Hessiana possui inversa, é positivamente definida e fazendo a equação em função de s, obtém-se:

$$\vec{\Gamma} = \vec{\Gamma}_0 - H^{-1}\nabla r(\vec{\Gamma}_0), \tag{2.37}$$

Por fim, $\vec{\Gamma}$ é a solução desejada para o sistema. Com H^{-1} sendo a inversa da matriz Hessiana. Portanto, a partir de um modelo inicial de vagarosidade ao quadrado $\vec{\Gamma}_0$, a inversa da matriz Hessiana e do gradiente da função no modelo inicial, é possivel iterativamente minimizar a função objetivo a fim de encontrar o mínimo global ou local da função utilizando o método de Newton.

2.4 Método de Gauss-Newton

Apesar de ter alta convergência, calcular a matriz Hessiana é extremamente caro para problemas com alto número de parâmetros. Outro fator limitador é que o problema apresenta natureza altamente não-linear, tornando a matriz Hessiana mal posta, não possuindo uma inversa definida. Para contornar tal problema, sabe-se que substituindo a função objetivo, para um tiro, um receptor e uma amostra de tempo, tem-se que:

$$\frac{\partial r(\Gamma_0)}{\partial \Gamma_m} = \frac{\partial \Delta P(\vec{x}, t; \vec{\Gamma})}{\partial \Gamma_m} \Delta P(\vec{x}, t; \vec{\Gamma}), \tag{2.38}$$

e aplicando a regra da cadeia na função objetivo:

$$Hij = \left(\frac{\partial^2 P(\vec{x}, t; \vec{\Gamma})}{\partial \Gamma_j \partial \Gamma_i} \Delta P(\vec{x}, t; \vec{\Gamma}) + \frac{\partial P(\vec{x}, t; \vec{\Gamma})}{\partial \Gamma_j} \frac{\partial P(\vec{x}, t; \vec{\Gamma})}{\partial \Gamma_i} \right). \tag{2.39}$$

Para contornar o problema de não linearidade do sistema proposto, é feita uma linearização da forma:

$$P(\vec{\Gamma}) = P(\vec{\Gamma}_0) + F\Delta\vec{\Gamma}, \tag{2.40}$$

onde o campo modelado $P(\vec{\Gamma})$ é uma função linear em relação ao modelo de vagarosidade ao quadrado $\vec{\Gamma}$. Ao derivar o campo em relação à $\vec{\Gamma}$, obtém-se:

$$\frac{\partial P(\vec{\Gamma})}{\partial \vec{\Gamma}} = F, \tag{2.41}$$

e

$$\frac{\partial^2 P(\vec{x}, t; \vec{\Gamma})}{\partial \Gamma_j \partial \Gamma_i} = 0, \tag{2.42}$$

com F sendo uma matriz chamada de derivada de Fréchet. Utilizando a informação à partir das derivadas de Fréchet, obtém-se:

$$H = \left(F^T F \right), \tag{2.43}$$

$$\nabla r(\vec{\Gamma}_0) = F^T \ \Delta P(\vec{x}, t; \vec{\Gamma}). \tag{2.44}$$

Por fim, a equação 2.45 encontra o campo de vagarosidade a partir das equações linearizadas usando as derivadas de Fréchet que é dada por:

$$\vec{\Gamma} = \vec{\Gamma}_0 - (F^T \ F)^{-1} \ F^T \ \Delta P(\vec{x}, t; \vec{\Gamma}). \tag{2.45}$$

2.5 Derivadas de Fréchet

Utilizando a abordagem proposta por Born (1926), deseja-se obter o campo de pressão total como a soma de um campo incidente suave e de um campo espalhado.

$$P(\vec{x}, t; \vec{\Gamma}) = P_0(\vec{x}, t; \vec{\Gamma}_0) + \Delta P(\vec{x}, t; \vec{\Gamma}), \tag{2.46}$$

e utilizando a definição de refletividade, onde o contraste é a diferença da forma:

$$\Delta\Gamma = \Gamma - \Gamma_0. \tag{2.47}$$

Utilizando esta abordagem, pode-se definir a equação da onda para o campo total e campo incidente, respectivamente, da seguinte forma:

$$\nabla^2 P(\vec{x}, t; \vec{\Gamma}) - (\Gamma_0 + \Delta\Gamma) \ \frac{\partial^2 P(\vec{x}, t; \vec{\Gamma})}{\partial t^2} = R(t)\delta(\vec{x} - \vec{x_s}), \tag{2.48}$$

$$\nabla^2 P(\vec{x}, t; \vec{\Gamma_0}) - \Gamma_0 \ \frac{\partial^2 P(\vec{x}, t; \vec{\Gamma_0})}{\partial t^2} = R(t)\delta(\vec{x} - \vec{x_s}). \tag{2.49}$$

De acordo com a linearização do campo de onda dada pela equação 2.46, o campo espalhado é a subtração do campo total pelo campo incidente, dado por:

$$\nabla^2 P(\vec{x}, t; \vec{\Gamma}) - \nabla^2 P(\vec{x}, t; \vec{\Gamma_0}) - [(\Gamma_0 + \Delta\Gamma) - \Gamma_0] \ \frac{\partial^2 P(\vec{x}, t; \vec{\Gamma}) - P(\vec{x}, t; \vec{\Gamma_0})}{\partial t^2} = 0. \tag{2.50}$$

Modificando a equação 2.50, tem-se:

$$\nabla^2 \Delta P(\vec{x}, t; \vec{\Gamma}) - \Gamma_0 \ \frac{\partial^2 \Delta P(\vec{x}, t; \vec{\Gamma})}{\partial t^2} = \Delta\Gamma \frac{\partial^2 P(\vec{x}, t; \vec{\Gamma})}{\partial t^2}. \tag{2.51}$$

Considerando a modelagem de Born (1926), onde o contraste é baixo, pode-se aproximar $P(\vec{x},t;\vec{\Gamma}) \approx P(\vec{x},t;\vec{\Gamma}_0)$, transformando a equação 2.51 em:

$$\nabla^2 \Delta P(\vec{x},t;\vec{\Gamma}) - \Gamma_0 \frac{\partial^2 \Delta P(\vec{x},t;\vec{\Gamma})}{\partial t^2} = \Delta\Gamma \frac{\partial^2 P(\vec{x},t;\vec{\Gamma}_0)}{\partial t^2}. \tag{2.52}$$

Aplicando a solução para o campo espalhado a partir da equação 2.52, obtém-se:

$$\Delta P(\vec{x},t;\vec{\Gamma}) = \int_V g(\vec{x},t;\vec{\Gamma}_0) * \Delta\Gamma \frac{\partial^2 P(\vec{x},t;\vec{\Gamma}_0)}{\partial t^2}\, dV, \tag{2.53}$$

onde a $g(\vec{x},t;\vec{\Gamma}_0)$ é a função de Green. Logo, substituindo em 2.40 na linearização do campo, observa-se:

$$P(\vec{\Gamma}_k) = P(\vec{\Gamma}_{0k}) + \sum_{k=1}^{N} g(x_k,t;\Gamma_{0k}) * \frac{\partial^2 P(x_k,t;\Gamma_{0k})}{\partial t^2} \Delta\Gamma_k. \tag{2.54}$$

Logo, chega-se a conclusão que a derivada de Fréchet é dada por:

$$F(\Gamma_{0k}) = \sum_{k=1}^{N} g(x_k,t;\Gamma_{0k}) * \frac{\partial^2 P(x_k,t;\Gamma_{0k})}{\partial t^2}, \tag{2.55}$$

como a derivada de Fréchet é a convolução da função de Green com o termo $\frac{\partial^2 P(x_k,t;\Gamma_{0k})}{\partial t^2}$, pode-se usar este como termo fonte da equação diferencial para obter a derivada de Fréchet a partir da solução da equação diferencial, ficando da forma:

$$\nabla^2 F(x_k,t;\Gamma_{0k}) - \Gamma_0 \frac{\partial^2 F(x_k,t;\Gamma_{0k})}{\partial t^2} = \frac{\partial^2 P(x_k,t;\Gamma_{0k})}{\partial t^2}. \tag{2.56}$$

Apesar de demonstrar o cálculo das derivadas de Fréchet, o trabalho não as calcula. Uma razão é a utilização do método adjunto aliado à técnicas de otimização local para encontrar os modelos de velocidade. Outro fator é a qualidade da inversão, como o método de Gauss-Newton negligencia o fator de segunda ordem da Hessiana, os resultados se mostram inferiores à métodos Newton ou quasi-Newton como demonstrado por Métivier et al. (2012).

2.6 Cálculo do gradiente via método adjunto

O método adjunto é uma alternativa para obter o gradiente sem calcular explicitamente as derivadas de Fréchet. Utilizando a derivada de Fréchet na equação 2.54, o gradiente da função fica da forma:

$$\nabla r(\vec{\Gamma}_0) = \sum_{r=1}^{Nr} \sum_{k=1}^{N} g(x_k,x_i,t;\Gamma_{0k}) * \frac{\partial^2 P_i(x_k,t;\Gamma_{0k})}{\partial t^2} \Delta P_i(t;s_k), \tag{2.57}$$

aplicando a transformada de Fourier no gradiente, obtém-se:

$$\nabla r(\vec{\Gamma}_0) = \frac{1}{2\pi}\sum_{r=1}^{Nr}\sum_{k=1}^{N} G(x_k, x_i, \omega; \Gamma_{0k})(i\omega)^2 P_k(x_k, \omega; \Gamma_{0k})\Delta P_i(\omega; \Gamma_k). \tag{2.58}$$

Para dar uma solução no domínio do tempo, é introduzida a fórmula de Plancharel. Duas funções $a(t)$ e $b(t)$ podem ser dadas na frequência pela relação:

$$\frac{1}{2\pi}\int_{-\infty}^{\infty} \overline{A}\ (\omega)B(\omega)\ d\omega = \int_{-\infty}^{\infty} a(t)b(t)\ dt, \tag{2.59}$$

onde $\overline{F}(\omega)$ é o complexo conjugado de $F(\omega)$. Aplicando a fórmula de Plancherel nas equações 2.55 e 2.57, tem-se:

$$\begin{aligned}&\frac{1}{2\pi}\sum_{r=1}^{Nr}\sum_{k=1}^{N}\int_{-\infty}^{\infty} \overline{F}\ (x_k, x_r, \omega; \Gamma_{0k})\Delta P_r(\omega; \Gamma_k)\ d\omega = \\ &\sum_{r=1}^{Nr}\sum_{k=1}^{N}\int_{-\infty}^{\infty} F(x_k, x_r, t; \Gamma_{0k})\Delta P_r(t; \Gamma_k)\ dt,\end{aligned} \tag{2.60}$$

substituindo a derivada de Fréchet, observa-se que o gradiente se dá por:

$$\nabla r(\vec{\Gamma_0}) = \frac{1}{2\pi}\sum_{r=1}^{Nr}\sum_{k=1}^{N}\int_{-\infty}^{\infty} \overline{(G(x_k, x_r, \omega; \Gamma_{0k})(i\omega)^2 P_k(x_k, \omega; \Gamma_{0k}))}\ \Delta P_r(\omega; \Gamma_k)\ d\omega. \tag{2.61}$$

Aplicando o operador conjugado e separando o somatório da vagarosidade e o campo, tem-se:

$$\nabla r(\vec{\Gamma_0}) = \frac{1}{2\pi}\sum_{k=1}^{N}\int_{-\infty}^{\infty} \overline{((i\omega)^2 P_k(x_k, \omega; \Gamma_{0k}))}\ \sum_{r=1}^{Nr}\overline{G}(x_k, x_r, \omega; \Gamma_{0k})\Delta P_r(\omega; \Gamma_k)\ d\omega. \tag{2.62}$$

Utilizando a fórmula de Plancherel, tem-se que:

$$A(\omega) = (i\omega)^2 P_k(x_k, \omega; \Gamma_{0k})\ \rightarrow\ a(t) = \frac{\partial^2 P_r(x_k, -t; \Gamma_{0k})}{\partial t^2}, \tag{2.63}$$

e que:

$$B(\omega) = \sum_{r=1}^{Nr}\overline{G}(x_k, x_r, \omega; \Gamma_{0k})\Delta P_r(\omega; \Gamma_k)\ \rightarrow\ b(t) = \sum_{r=1}^{Nr} G(x_k, x_r, -t; \Gamma_{0k}) * \Delta P_r(t; \Gamma_k). \tag{2.64}$$

Ao passar o complexo conjugado da função de Green para o domínio do tempo, esta passa a estar sob condições finais de contorno. Logo, a modelagem passa a acontecer reversamente no tempo. O termo $b(t)$ passa a ser chamado de campo adjunto, definido como:

$$P_k^{\dagger} = \sum_{r=1}^{Nr} G(x_k, x_r, -t; \Gamma_{0k}) * \Delta P_r(t; \vec{\Gamma}). \tag{2.65}$$

Como o resíduo está convolvido com a função de Green, basta utilizá-lo como termo fonte para calcular o campo adjunto, ficando:

$$\nabla^2 P^{\dagger}(\vec{x}, t; \vec{\Gamma}_0) - \Gamma_0 \frac{\partial^2 P^{\dagger}(\vec{x}, t; \vec{\Gamma}_0)}{\partial t^2} = \sum_{r=1}^{Nr} \Delta P_r(t; \vec{\Gamma}). \tag{2.66}$$

onde o campo $P^{\dagger}(\vec{x}, t; \vec{\Gamma}_0)$ é calculado reversamente no tempo. Substituindo na equação 2.30 e simplificando a derivada no tempo por:

$$\frac{\partial^2 P(\vec{x}, t; \vec{\Gamma}_0)}{\partial t^2} = P"(x_k, t; \vec{\Gamma}), \tag{2.67}$$

o gradiente passa a ser calculado pela expressão:

$$\nabla r(\vec{\Gamma}_0) = \sum_{k=1}^{N} \int P"(x_k, t; \Gamma_{0k}) P_k^{\dagger}(\vec{x}, t; \vec{\Gamma}_0)\ dt. \tag{2.68}$$

Substituindo na equação 2.30 se obtém:

$$\vec{\Gamma} = \vec{\Gamma}_0 - \alpha \int P"(\vec{x}, t; \vec{\Gamma}_0) P^{\dagger}(\vec{x}, t; \vec{\Gamma}_0)\ dt. \tag{2.69}$$

Para melhor manipulação, deixando a equação em função da velocidade se obtém:

$$\frac{1}{\vec{v}^2} = \frac{1}{\vec{v_0}^2} - \alpha \int P"(\vec{x}, t; \vec{\Gamma}_0) P^{\dagger}(\vec{x}, t; \vec{\Gamma}_0)\ dt. \tag{2.70}$$

Por fim, isolando o termo de velocidade:

$$\vec{v} = \frac{\vec{v_0}}{\sqrt{1 - \vec{v_0}^2 \alpha \int P"(\vec{x}, t; \vec{\Gamma}_0) P^{\dagger}(\vec{x}, t; \vec{\Gamma}_0)\ dt}}. \tag{2.71}$$

Assumindo uma atualização menor que 10% o valor máximo do modelos de velocidade, pode-se utilizar o pré-condicionamento da pseudo-Hessiana como proposto por Carneiro et al. (2018), deixando a velocidade da forma:

$$\vec{v} = \vec{v_0} + \alpha \, \frac{\vec{v_0}^3}{2} \, \frac{\int P"(\vec{x}, t; \vec{\Gamma}_0) P^{\dagger}(\vec{x}, t; \vec{\Gamma}_0) \, dt}{\int P"(\vec{x}, t; \vec{\Gamma}_0) P"(\vec{x}, t; \vec{\Gamma}_0) \, dt}. \tag{2.72}$$

Desta forma, o método adjunto fornece o vetor gradiente que aponta para a direção de decrescimento. Para o cálculo de α, lança-se mão de métodos de otimização para estimar um passo que minimiza a função objetivo.

2.7 Considerações sobre a modelagem e a inversão

Na de aquisição sísmica marinha, diversos fatores influenciam negativamente o sinal sísmico: ruídos randômicos e coerentes, *swell*, interferência sísmica, bolhas, *ghost* e múltiplas. Estes problemas são conhecidos e existem tratamentos específicos para cada um deles. De maneira similar à aquisição real, estes fenômenos tem que ser eliminados do dado pois não são levados em consideração na simulação numérica. Por conta de dados sintéticos não representarem com exatidão os problemas encontrados em campo, deve-se fazer algumas considerações quanto à geração e inversão destes dado.

Diversas abordagens podem ser utilizadas para a inversão, este trabalho utiliza a abordagem de Pica, Diet e Tarantola (1990) e Bunks et al. (1995), que divide o dado sísmico em bandas de frequência. Isto é necessário por conta do fenômeno de salto de ciclo (*cycle skipping*). A figura 7 ilustra o problema de salto de ciclo. A linha sólida representa um sismograma monocromático de período T, os sismogramas tracejados são os modelados. O sismograma inferior possui defasagem menor e o superior maior que $\frac{T}{2}$.

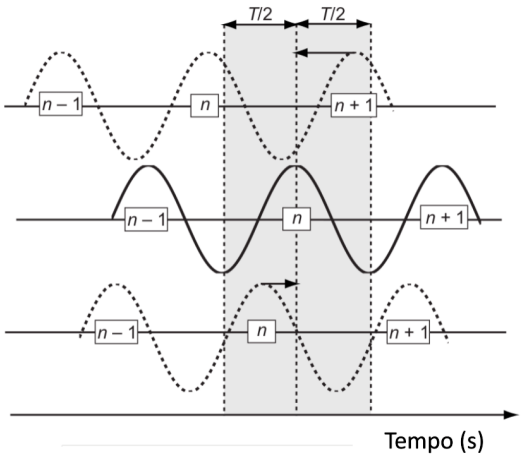


Figura 7 – Ilustração do fenômeno de salto de ciclo. As linhas tracejadas representam sismogramas monocromáticos defasados e a linha sólida representa o sismograma observado. Modificado de Virieux e Operto (2009).

Analisando a figura, o sismograma inferior não sofrerá salto de ciclo pois os picos e vales estão mais próximos e serão correlacionados. Já no sismograma superior, devido à fase ser maior que $\frac{T}{2}$, há uma correlação do pico de amostra $n+1$ do sismograma superior com a amostra n sismograma observado, correlacionando eventos de natureza diferente. Ao iniciar a inversão com altas frequências, eventos distintos podem ser correlacionados. A função objetivo nas altas frequências possui uma maior variabilidade e um número maior de mínimos locais, dificultando a otimização ao mínimo global. A figura 8 ilustra este problema.

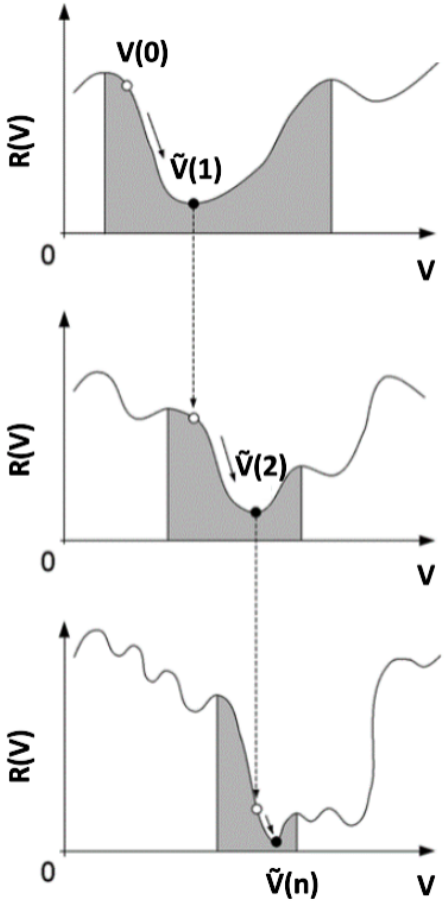


Figura 8 – Ilustração da variação da função objetivo de acordo com o conteúdo de frequências. Modificado de Fichtner (2009).

Pode-se observar que em baixas frequências, a função objetivo é suave, bem comportada e possui poucos mínimos locais. Isto facilita a linha de busca do algoritmo, podendo encontrar o mínimo global daquela banda de frequência. Uma vez terminada a otimização neste espectro, o modelo de velocidade é atualizado e é utilizado como modelo inicial para a próxima banda de frequência, obtendo iterativamente modelos que vão sendo atualizados até a frequência desejada. Apesar de possuir poucos mínimos locais, ainda é necessário um bom modelo inicial para a inversão, uma vez que o algoritmo pode continuar otimizando por volta de um mínimo local e gerar um modelo de velocidade que não condiz

com a geologia do dado.

Outro fenômeno presente no imageamento sísmico é o *ghost* (fantasma). O *ghost* é o resultado de interferência destrutiva de reflexões espúrias reverberadas da superfície marinha que são sobrepostas com reflexões primárias, agindo como um filtro *Notch* no dado (SHERIFF; GELDART, 1995; HAMARBITAN; MARGRAVE, 2001). Para não sofrer com este tipo de problema, este trabalho aplica bordas absortivas na porção superior do modelo, desta forma, a onda que seria refletida pela superfície livre será absorvida e não causará interferência no dado. Neste contexto, é necessário se fazer uma escolha da borda utilizada nos códigos de modelagem e inversão.

2.7.1 Análise de absorção das bordas

Devido à necessidade de uma borda absortiva, deseja-se encontrar a borda mais eficiente quanto à diminuição da energia refletida nas bordas. Para medir a efetividade da borda absortiva, deve-se medir o conteúdo de energia refletido. A equação de energia cinética da onda é dada por:

$$E_c = \frac{\rho\omega^2 A^2}{2}, \tag{2.73}$$

onde E_c é a energia cinética da onda, ρ é a densidade do meio, ω é a frequência do pulso e A é a amplitude da onda. Como os testes serão feitos no mesmo modelo e os pulsos possuem mesmo espectro, pode-se fazer a comparação de energia utilizando o quadrado da amplitude. Para a energia refletida, é modelada uma onda direta sem a presença de borda absortiva com um modelo homogêneo e utilizando borda infinita. Logo, pode-se subtrair o sismograma do modelo com borda absortiva e analisar a energia refletida da borda. Para obter o conteúdo de energia no sismograma, basta somar termo a termo o quadrado da amplitude da forma:

$$E_{sismograma} = \sum_{i=1}^{N_r}\sum_{j=1}^{N_t} A_{ij}^2, \tag{2.74}$$

onde N_r é o número de receptores e N_t é o número de amostras de tempo. Para encontrar o melhor coeficiente para a borda Cerjan, modelou-se o sismograma para valores de f de $0,0053$, $0,0040$, $0,0027$, $0,0014$ e $0,0010$ para a equação 2.14. O valor da energia refletida é apresentado na tabela 1.

Tabela 1 – Valores de energia refletida para cada configuração da borda Cerjan.

	0,0010	0,0014	0,0027	0,0040	0,0053
100 pontos	1,3257294	0,0583778	0,6975196	3,0108169	8,1795391
125 pontos	0,0171657	0,0532984	0,6975213	3,0108161	8,1795418
150 pontos	0,0142816	0,0532989	0,6975216	3,0108163	8,1795419
175 pontos	0,0142808	0,0532989	0,6975216	3,0108163	8,1795419
200 pontos	0,0142808	0,0532989	0,6975215	3,0108163	8,1795419

Analisando os valores da tabela 1, pode-se constatar que para 7 casas decimais, o valor mínimo de energia refletida se encontra utilizando 175 pontos e fator $f = 0,0010$. O valor com 200 pontos tem a mesma energia refletida mas necessita de mais pontos de borda. Tendo em vista este resultado, será utilizada essa configuração para as simulações utilizando borda Cerjan. Apesar de ser eficiente, a borda Cerjan necessita de um número grande de pontos de borda para ser efetiva em mitigar os efeitos de reflexão de borda. Tal fato aumenta o custo computacional devido à necessidade de calcular a equação da onda nestes pontos adicionais. Para um modelo de dimensões 400x400 e bordas 175x175, haverá um aumento de 106% no número de pontos a serem calculados em relação ao modelo original, aumentando proporcionalmente o custo computacional e de memória.

Visando a utilizar menos pontos de borda, busca-se analisar a efetividade da borda CPML, cuja a proposta é utilizar um baixo valor de número de pontos. Para a borda CPML é feito o mesmo estudo de energia refletida utilizado para a borda Cerjan. Para estes testes, são feitas modelagens de 10 a 50 pontos de borda e valores de R_t da equação 2.18 que variam de 10^{-3} a 10^{-7}. Os valores de energia refletida são expressos pela tabela 2.

Tabela 2 – Valores de energia refletida para cada valor de refletividade da borda CPML.

	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
10 pontos	1531,5486996	1048,6437801	752,6380221	556,5054837	420,1754473
20 pontos	30,1329534	7,4418262	2,0515162	0,6658399	0,2688626
30 pontos	2,5787090	0,3544189	0,0713057	0,0239352	0,0119083
40 pontos	0,5077210	0,0545641	0,0114999	0,0048039	0,0028092
50 pontos	0,1654954	0,0159091	0,0038357	0,0018462	0,0011338

Analisando a tabela 2, a configuração com menor conteúdo de energia refletida é utilizando 50 pontos e $Rt = 10^{-7}$. Para efeito de comparação, a figura 9 mostra os sismogramas com a onda direta e os sismogramas residuais para as melhores configurações das bordas Cerjan e CPML.

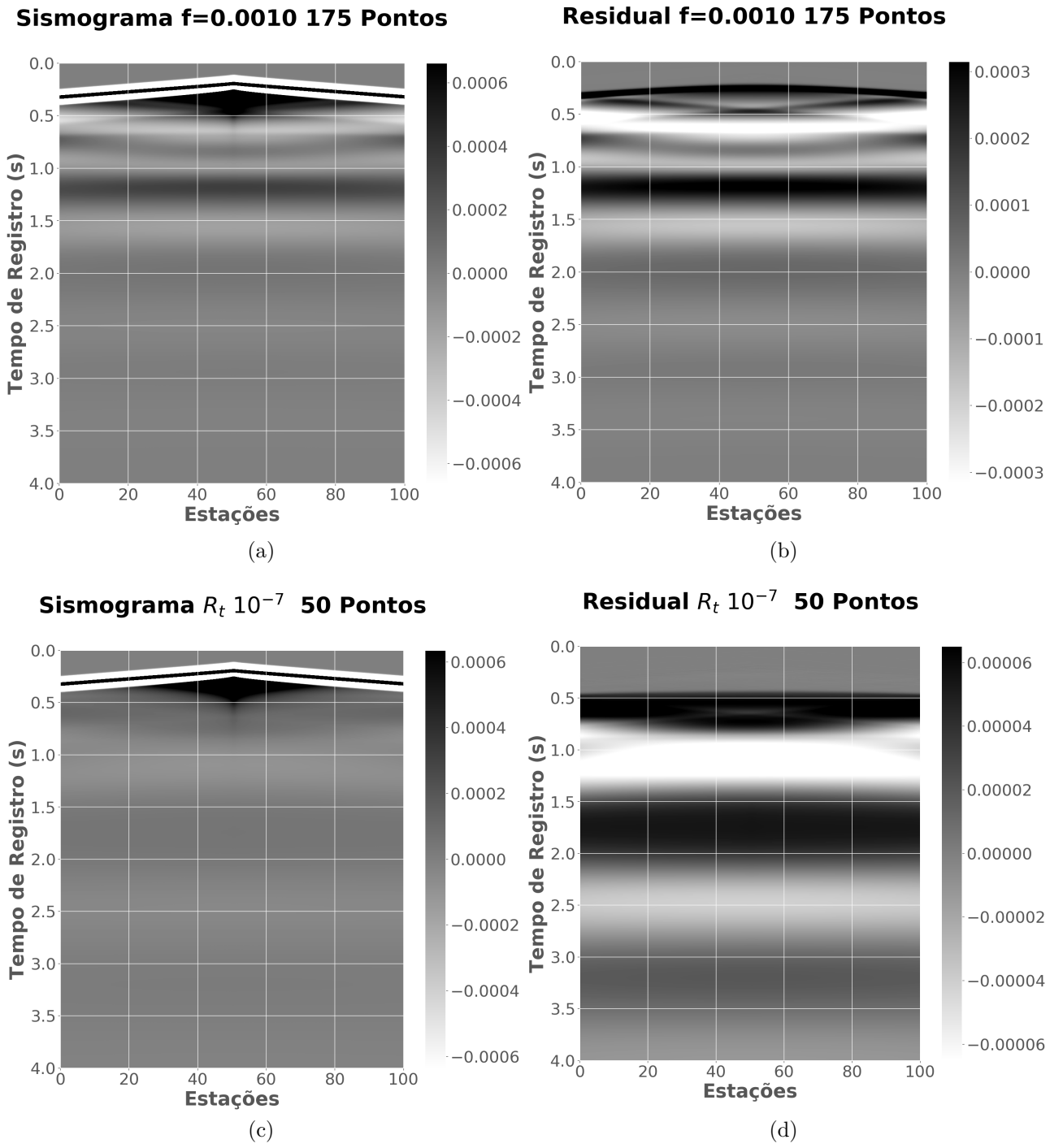


Figura 9 – a) Sismograma gerado utilizando a borda Cerjan com fator $f = 0,0010$ e 175 pontos de borda. b) Sismograma residual, resultante da subtração do sismograma gerado utilizando a borda Cerjan com a onda direta. c) Sismograma gerado utilizando a borda CPML com fator $R_t = 10^{-7}$ e 50 pontos de borda. d) Sismograma residual, resultante da subtração do sismograma gerado usando CPML com a onda direta.

Ao analisar a figura 9, nota-se significativa diferença na energia refletida. O sismograma da borda CPML reflete menos energia e isso é evidenciado nas amplitudes do sismograma residual. Observa-se que a escala do sismograma residual da CPML (Figura

9.d) possui escala de ordem de grandeza menor que a do sismograma residual da Cerjan. Para comparação, seleciona-se o traço da estação 0 da onda direta e dos sismogramas das bordas CPML e Cerjan, ilustrado na figura 10.

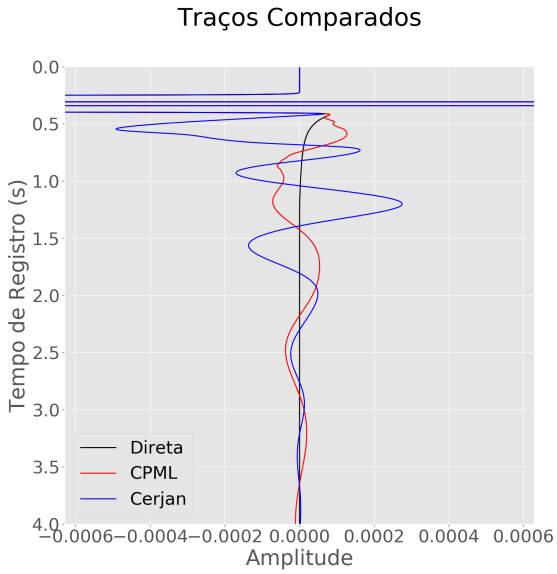


Figura 10 – Comparação do primeiro traço dos sismogramas da onda direta (preto) gerada utilizando borda infinita, CPML (vermelho), Cerjan (Azul).

Observa-se que a borda CPML respeita mais o desenho da onda direta do que a borda Cerjan, outra comparação a ser feita é com relação à energia refletida. Enquanto que a borda Cerjan retornou uma energia de 0.014 280 J, a borda CPML refletiu apenas 0.001 133 8 J. Tendo em vista estes resultados, a borda utilizada neste trabalho será a borda CPML. No entanto, a CPML é apenas utilizada para a equação da onda acústica. Esta decisão se dá pois a borda CPML apresenta instabilidade para longos períodos no caso elástico, havendo a necessidade de apenas usar a continuação do campo através da borda infinita.

Outro fator presente na FWI utilizando dados sintéticos é o crime da inversão. O crime da inversão ocorre quando os mesmos operadores de inversão são os mesmos utilizados na geração do dado (WIRGIN, 2004). Apesar de tentar evitar tal prática, o trabalho comete o crime da inversão pois o operador de diferença finita e as *wavelets* usadas na modelagem são os mesmos utilizados no *kernel* de inversão. Uma consideração deste trabalho é a razão de se utilizar a décima ordem no espaço para a implementação do MDF. Devido ao alto paralelismo da GPU, a utilização de altas ordens acaba sendo de baixo ou nenhum impacto no tempo de execução. Dando assim liberdade de obter soluções menos susceptíveis à dispersão mas sem um impacto negativo no tempo de execução do código.

3 Otimização numérica

A otimização numérica é uma ferramenta importante na análise de sistemas físicos. O uso desta ferramenta é feito a partir da minimização de uma função que define a energia ou potencial de um sistema físico. Uma vez formulado o modelo, um algoritmo de otimização pode ser utilizado para encontrar o vetor de parâmetros que melhor minimiza a função. Não há um algoritmo de otimização universal para todos os problemas, mas sim um conjunto de algoritmos desenhados para cada tipo de problema. Este conjunto de algoritmos pode ser subdividido em algoritmos de otimização global e de otimização local.

Os algoritmos de otimização global como o de Monte Carlo, utilizam um elevado número de modelagens, modificando os valores dos parâmetros e calculando o valor da função para cada parâmetro modificado para chegar à solução com menor valor de função objetivo. Em um conjunto de parâmetros da ordem de 10^9, esta abordagem torna-se impraticável para o caso sísmico, mesmo para os computadores modernos empregados na indústria. Algoritmos de otimização locais são mais computacionalmente acessíveis. Com a garantia de estar próximo do mínimo global e ter uma acurada direção de atualização, os algoritmos locais fornecem a solução menos computacionalmente custosa e satisfatória no contexto do imageamento sísmico, uma vez que o algoritmo iterativamente atualiza o modelo de parâmetros até chegar uma tolerância aceitável. O conjunto de algoritmos de otimização local pode ser separado em algoritmos de linha de busca (*line search*) e regiões de confiança (*Trust-Region*). Este trabalho utiliza e aborda apenas os de linha de busca. Exemplos de algoritmos de linha de busca são os de máxima descida (*Steepest Descent*) (PETROVA; SOLOV'EV, 1997), Gradiente Conjugado (*Conjugate Gradient*) (SHEWCHUK, 1994), BFGS e L-BFGS (NOCEDAL; WRIGHT, 2006). O algoritmo de linha de busca tenta encontrar iterativamente um passo α positivo escalar ou matricial que se aproxima do mínimo da função objetivo, atualizando o valor do parâmetro $\vec{\Gamma}_k$ atualizado do FWI ao longo da direção de atualização p_k da forma:

$$\vec{\Gamma}_{k+1} = \vec{\Gamma}_k - \alpha\ \vec{p}_k. \tag{3.1}$$

Estes algoritmos precisam satisfazer algumas condições matemáticas para continuar o processo de otimização. A primeira condição é que $r(\vec{\Gamma}_k - \alpha\ \vec{p}_k) < r(\vec{\Gamma}_k)$, podendo apenas atualizar o valor da vagarosidade ao quadrado quando a função objetivo diminuir. Outra condição básica dos algoritmos de linha de busca é satisfazer a condição de direção descendente, sendo:

$$p_k^T \nabla r(\vec{\Gamma}) < 0. \tag{3.2}$$

Apesar de satisfazer estas condições básicas, alguns algoritmos requerem condições adicionais como as condições de Wolfe (NOCEDAL; WRIGHT, 2006). As condições de Wolfe estipulam o passo α, onde a primeira condição verifica se o mesmo oferece um decréscimo satisfatório e a segunda verifica a curvatura da função, analisando se o passo é suficiente para produzir uma mudança significativa na função objetivo. As condições de Wolfe são dadas pelas equações:

$$r(\vec{\Gamma}_k - \alpha\ \vec{p}_k) \leq r(\vec{\Gamma}_k) + c_1\alpha\nabla r^T(\vec{\Gamma}_k)\ p_k, \tag{3.3}$$

e

$$\mid \nabla r^T(\vec{\Gamma}_k - \alpha\ \vec{p}_k)\ \vec{p}_k \mid \leq \mid c_2\nabla r^T(\vec{\Gamma}_k)\ p_k \mid\ . \tag{3.4}$$

onde c_1 e c_2 são constantes com valores no intervalo $0 < c_1 < c_2 < 1$. O valor exato de cada constante varia para cada algoritmo, os coeficientes utilizados neste trabalho são $c_1 = 10^{-3}$ e $c_2 = 0.9$. Uma propriedade interessante destas condições é que estas são invariantes com relação à escala, não sendo alteradas devido à multiplicação de uma constante à função objetivo.

3.1 O algoritmo L-BFGS

O método L-BFGS é um algoritmo iterativo para resolver problemas não lineares de otimização. O método é uma versão de memória limitada do método BFGS, muito utilizado em problemas de larga escala. BFGS é um algoritmo desenvolvido por Broyden, Fletcher, Goldfarb e Shanno (NOCEDAL; WRIGHT, 2006), sendo considerado um método quasi-Newton de otimização local. Algoritmos quasi-Newton não calculam ou utilizam a matriz Hessiana completa, estes códigos exploram a propriedade de simetria da matriz Hessiana para calcular apenas elementos específicos e a truncando. Para entender o método L-BFGS, primeiro se deve analisar como é a estimativa da matriz truncada no método BFGS, cada passo no método BFGS é dado por:

$$\vec{\Gamma}_{k+1} = \vec{\Gamma}_k - \alpha H_k^{-1}\nabla r(\vec{\Gamma}_k), \tag{3.5}$$

com H_k^{-1} sendo a aproximação da matriz Hessiana na iteração k, onde sua fórmula é dada por:

$$H_{k+1}^{-1} = V_k^T H_k^{-1} V_k + z_k x_k x_k^T, \tag{3.6}$$

onde os vetores $x_k,\ y_k,\ z_k$ e V_k são dados à partir de:

$$x_k = \vec{\Gamma}_{k+1} - \vec{\Gamma}_k, \quad y_k = \nabla r(\vec{\Gamma}_{k+1}) - \nabla r(\vec{\Gamma}_k), \tag{3.7}$$

$$z_k = \frac{1}{y_k^T x_k} \quad e \quad V_k = I + z_k y_k x_k^T, \tag{3.8}$$

onde I é a matriz identidade. Como dito anteriormente, o problema do método BFGS está no tamanho da matriz H_k^{-1}, que torna o problema custoso e necessita de alta memória para computar todos os seus valores. Uma maneira de contornar este problema é fazer o produto $H_k^{-1}\nabla r(\vec{\Gamma}_k)$ a partir de produto interno e soma de pares de vetores armazenados de m interações anteriores. Este método é denominado *Limited memory BFGS* ou L-BFGS. O esquema de calcular a matriz Hessiana é dado pela seguinte equação:

$$\begin{aligned} H_k^{-1} = {} & (V_{k-1}^T \cdots V_{k-m}^T) H_0^{-1} (V_{k-1} \cdots V_{k-m}) \\ & + z_{k-m+1}(V_k^T \cdots V_{k-m+1}^T) x_{k-m+1} x_{k-m+1}^T (V_k \cdots V_{k-m+1}) \\ & + z_{k-m+2}(V_{k+1}^T \cdots V_{k-m+2}^T) x_{k-m+2} x_{k-m+2}^T (V_{k+1} \cdots V_{k-m+2}) \\ & + \cdots \\ & + z_{k-1} x_{k-1} x_{k-1}^T. \end{aligned} \tag{3.9}$$

onde H_0^{-1} é um chute inicial da inversa da matriz Hessiana, sendo obtida a cada iteração à partir da equação:

$$H_0^{-1} = \gamma_k I, \tag{3.10}$$

com γ_k sendo dado por:

$$\gamma_k = \frac{x_{k-1} y_{k-1}}{y_{k-1} y_{k-1}}. \tag{3.11}$$

O coeficiente γ_k funciona como um fator de escala para diagonal da inversa da matriz Hessiana, garantindo que a direção de atualização esteja sempre bem escalada, aceitando o valor de $\alpha_k = 1$ para a primeira e outras iterações. A estratégia tem funcionado bem e o seu custo computacional depende do valor de m vetores armazenados para estimar a inversa da Hessiana, este valor m é variável para cada tipo de problema encontrado.

Portanto, o fluxo utilizado para calcular o novo valor do parâmetro Γ_k é dado por:

Algoritmo 1: Algoritmo L-BFGS

Utilizar ponto inicial s_0;
$m \leftarrow 20$;
$k \leftarrow 0$;
while *($k <$ número de iterações)* **do**
  Calcular H_k^0;
  $q \leftarrow \nabla r(\vec{\Gamma}_k)$;
  for $i = k-1,\ k-m$ **do**
    $\alpha_i \leftarrow z_i x_i^T q$;
    $q \leftarrow q + \alpha_i y_i$;
  end
  $r \leftarrow H_k^0 q$;
  for $i = k-m,\ k-1$ **do**
    $\beta \leftarrow z_i y_i^T r$;
    $r \leftarrow r + x_i(\alpha_i - \beta)$;
  end
  $\vec{\Gamma}_{k+1} \leftarrow \vec{\Gamma}_k - r$;
  if *($k > m$)* **then**
    Deletar $[x_{k-m}, y_{k-m}]$;
  end
  $x_k \leftarrow \vec{\Gamma}_{k+1} - \vec{\Gamma}_k$;
  $y_k \leftarrow \nabla r(\vec{\Gamma}_{k+1}) - \nabla r(\vec{\Gamma}_k)$;
  $k \leftarrow k + 1$;
end

O algoritmo tem deficiências quanto à convergência quando o problema é muito mal-posto, mas converge de maneira mais eficaz que algoritmos que não utilizam a Hessiana como um fator para a atualização do parâmetro (NOCEDAL; WRIGHT, 2006).

4 Programação em placas gráficas utilizando OpenACC

A evolução dos computadores ajudou a propulsar o desenvolvimento tecnológico e científico dos últimos anos. A capacidade e velocidade do processamento tem crescido exponencialmente, possibilitando resolver problemas complexos e de larga escala (BARLAS, 2014).

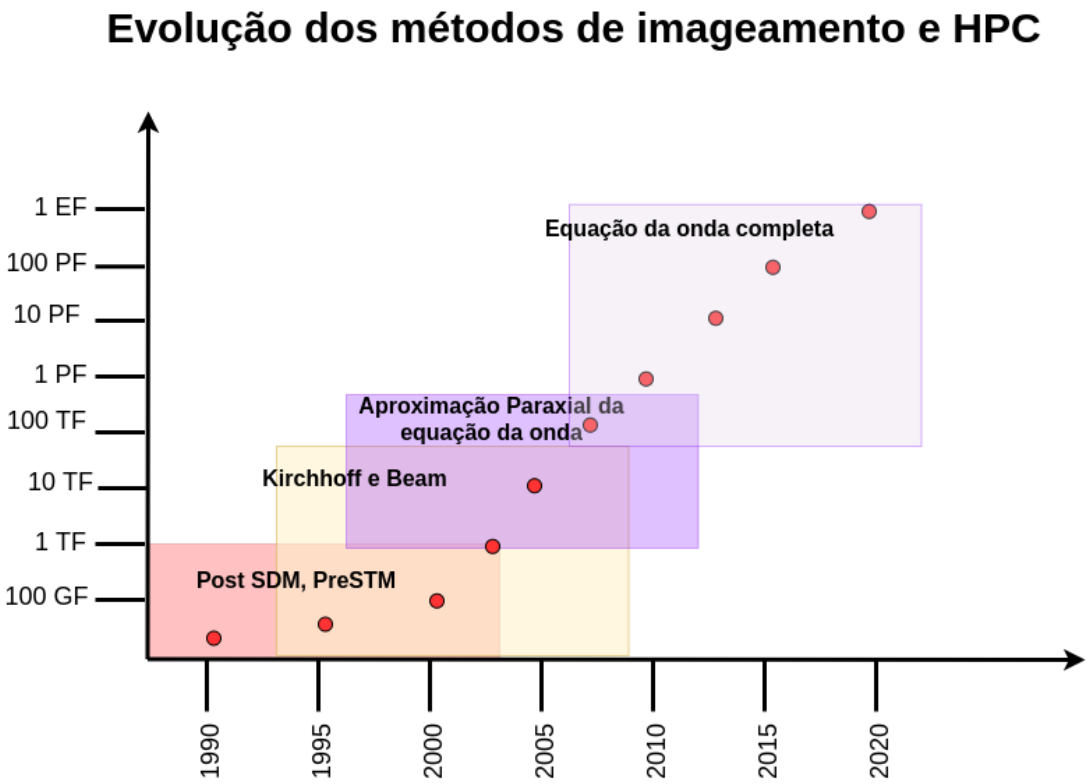


Figura 11 – Evolução dos métodos de imageamento sísmo e da computação de alta performance (HPC) ao longo dos anos. O eixo y é a capacidade de operações de *floats* por segundo e o eixo x é o de tempo em anos. Modificado de Brandsberg-Dahl (2017).

A figura 11 ilustra a evolução das técnicas de imageamento e da computação de alta performance ao longo dos anos. O número de [1]transistores aumentou exponencialmente em trinta anos, sendo o motor do aumento da capacidade computacional e da computação científica. A dificuldade de aumentar o [2]*clock* e o número de transistores em um uníco núcleo dificultou o aumento do poder computacional. A solução foi aumentar a quantidade

[1] Transistor: dispositivo semicondutor usado para amplificar ou trocar sinais eletrônicos e potência elétrica.
[2] *Clock*: número de ciclos por segundo de um sinal de sincronismo usado dentro do processador.

de núcleos que um processador pode comportar. Estes processadores agora podem hospedar diversas CPUs em um [3]*chipset*.

Instintivamente, pode-se pensar que para aumentar a capacidade de computação de uma máquina, deve-se empregar mais transistores em um núcleo e aumentar mais ainda o número de núcleos. No entanto uma grande mudança neste paradigma computacional ocorreu por conta do surgimento da arquitetura heterogênea CPU-GPU, decorrente da melhor empregabilidade de GPUs em problemas computacionalmente custosos (CHENG; GROSSMAN; MCKERCHER, 2014). Apesar de um núcleo de CPU ser bem mais robusto que um núcleo de GPU, a GPU se beneficia de ter uma arquitetura massivamente paralela, chegando a ter dezenas ou milhares de núcleos conectados a uma alta banda de frequência e uma alta velocidade de RAM, diminuindo significativamente o tempo de processamento. Outra vantagem da GPU está em seu consumo de energia, fornecendo uma razão de cálculos por Watt maior que a CPU. Tal característica é um grande estímulo para a maior empregabilidade de GPUs em *clusters* e serviços de nuvem.

Os paradigmas de programação podem ser divididos em programação serial e paralela. A programação serial, ilustrada na figura 12 é o paradigma tradicional da computação, onde cálculos e execuções são feitas em série. Já a figura 13 ilustra o conceito de programação paralela, que se baseia em cálculos sendo feitos simultaneamente, partindo do princípio de que grandes problemas podem ser divididos em problemas menores. Apesar de seus benefícios, a programação paralela não é aconselhável para todos os problemas encontrados na ciência. Problemas onde a instrução atual depende de uma anterior impossibilitam a paralelização, obrigando a recorrer à programação serial. Para minimizar o tempo de execução ao máximo, o programador tem a tarefa de mapear as regiões mais paralelizáveis e aplicar a lógica de programação paralela em tais regiões.

Figura 12 – Ordem de execução de um algoritmo serial. Modificado de Cheng, Grossman e McKercher (2014).

[3] *Chipset*: conjunto de componentes eletrônicos de baixa capacidade, em um circuito integrado, que gerencia o fluxo de dados entre o processador, memória e periféricos.

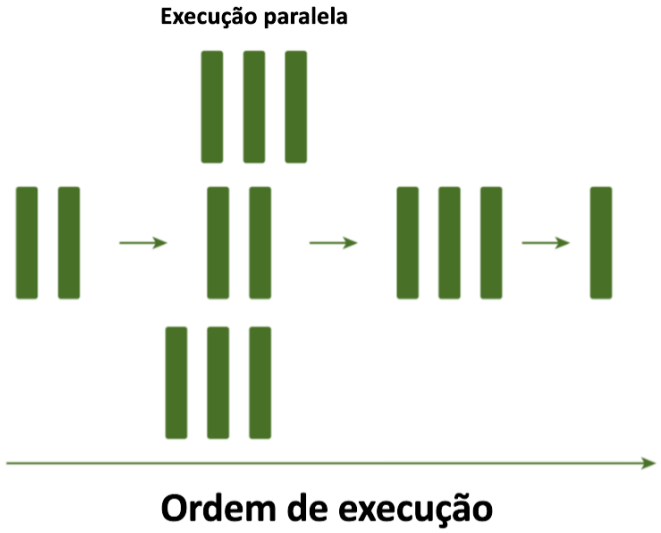


Figura 13 – Ordem de execução de um algoritmo paralelizado. Modificado de Cheng, Grossman e McKercher (2014).

4.1 Arquitetura de máquinas paralelas

As arquiteturas de computadores podem ser classificadas pela quantidade de dados e instruções que estas processam concomitantemente. Para classificar as máquinas de acordo com cada tipo de arquitetura, Flynn (1972) elaborou um esquema de classificação de máquinas paralelas, sendo diferenciadas pelo tipo de instruções e tipo de fluxo de dados. Quatro tipos de arquiteturas de máquinas podem ser diferenciadas, sendo estas:

1. Instrução Única Dado Único (*Single Instruction Single Data* ou SISD). Uma simples máquina que executa uma instrução de cada vez.

2. Instrução Única Múltiplos Dados (*Single Instruction Multiple Data* ou SIMD) se refere a um tipo de arquitetura paralela. Esta arquitetura executa a mesma instrução em dados diferentes. CPUs modernos e o multiprocessador de transmissão (*Streaming Multiprocessor* ou SM) de uma GPU seguem esta linha de arquitetura.

3. Múltiplas Instruções Único Dado (*Multiple Instruction Single Data* ou MISD). Este tipo de arquitetura é utilizado quando a tolerância à falha é necessária em um sistema, a partir daí dados podem ser processados por diversas máquinas e decisões são feitas pelo princípio da maioria. Esta arquitetura não é comumente utilizada na ciência, sendo empregada em ramos militares.

4. Múltiplas Instruções Múltiplos Dados (*Multiple Instruction Multiple Data ou MIMD*). Arquitetura mais versátil, sendo definida por máquinas multinúcleos como GPUs. Como um SM é de arquitetura SIMD e uma GPU é uma coleção de SMs, esta pode ser definida como MIMD.

A CPU e a GPU tiveram funções distintas desde o início de sua utilização na computação. A CPU funciona como o cérebro do computador, responsável por executar todas as tarefas. Já a GPU era responsável por acelerar o processamento de imagens. Por conta desta dinâmica a CPU é chamada de *Host* (anfitriã) e a GPU é chamada de *device* (acessório). Apesar de ambas possuírem diversos núcleos, o núcleo da CPU possui características diferentes em relação ao núcleo da GPU. A figura 14 ilustra a composição básica de cada máquina paralela.

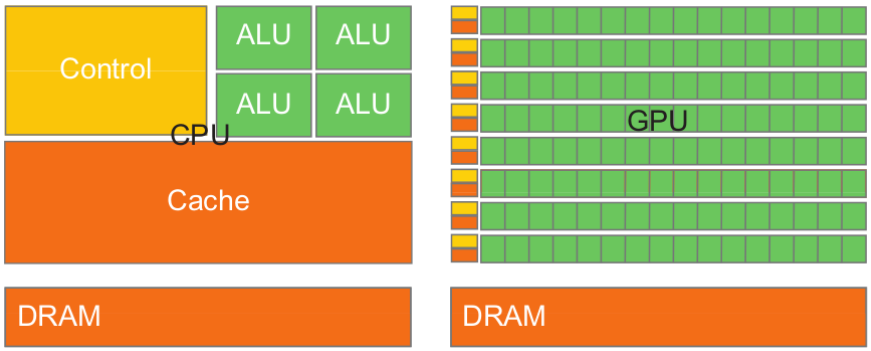


Figura 14 – Ilustração da arquitetura da CPU, à esquerda e da GPU, à direita. Retirado de Kirk e Hwu (2010).

Analisando a figura 14, pode-se observar a presença das seguintes feições:

- ALU: A *Arithmetic Logic Unit* (Unidade Lógica Aritmética) é a parte do processador com a finalidade de executar operações lógicas e artiméticas, tendo acesso direto à memória do processador.
- *Control*: Mais conhecida como Unidade de Controle (*Control Unit* ou CU), é o componente da CPU que direciona a operação do processador. Esta unidade diz à memória do computador, ALU e *devices* como responder às instruções enviadas para o processador.
- *Cache*: A memória *Cache* é uma memória de acesso rápido, apesar de possuir baixa capacidade de armazenamento, esta unidade é bem rápida e armazena as informações que são mais utilizadas da CPU ou GPU.
- DRAM ou RAM: Sigla para *Dynamic Random Access Memory* (Memória dinâmica de Acesso Aleatório). Esta é uma memória de acesso aleatório que armazena cada *bit* do dado em uma célula de memória, tendo como função o armazenamento de memória principal do computador.

A arquitetura da CPU e da GPU são visivelmente diferentes. A CPU é feita para ter pesados e grandes caches de memória, com poucos e complexos ALUs e uma decodificação

de instrução para que a instrução não fique parada esperando enquanto o dado chega na memória principal (DRAM). Tal configuração torna a CPU extremamente eficiente, sendo otimizada para programas seriais e de alta complexidade. Já a GPU é composta de diversas fileiras de SMs, onde cada fileira possui *Cache* e *Control* significativamente mais leves que na CPU, tornando-a muito menos efetiva ao lidar com problemas de alta complexidade lógica. A ALU também é consideravelmente mais leve, tornando sua computação serial inferior à CPU. No entanto, um SM possui diversos ALUs, possibilitando paralelismo massivo e aumento drástico na performance computacional. A figura 15 ilustra a efetividade da CPU e da GPU com relação ao paralelismo do problema e do tamanho do dado.

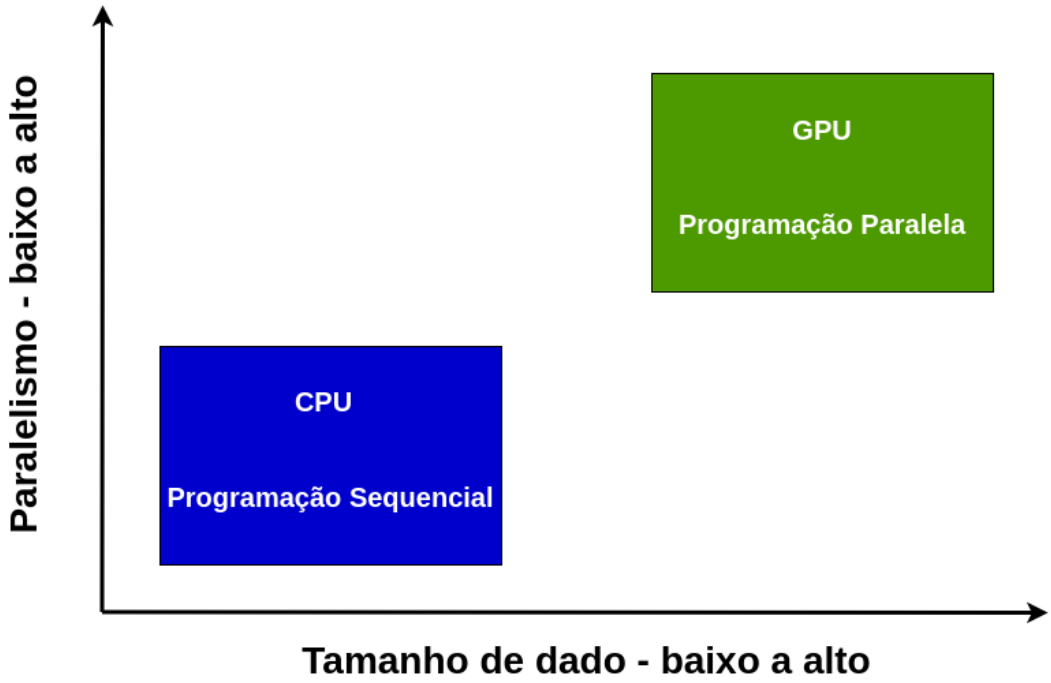


Figura 15 – Ilustração de cada paradigma de programação e os problemas em que são recomendadas. Modificado de Kirk e Hwu (2010).

Apesar dos grandes avanços em suas arquiteturas e performance, a GPU não veio para substituir a CPU. Sendo cada uma útil para cada tipo de abordagem utilizada. A CPU é ótima para problemas que exigem intenso controle e a GPU é ótima para problemas paralelizáveis e cálculos pesados. Uma clara vantagem da GPU é, uma vez que o dado está em sua memória, o acesso a este dado é muito rápido e a sua capacidade de cálculos é muito superior à CPU. No entanto, uma fraqueza da GPU é a memória, sendo consideravelmente menor comparada à CPU. Outro fator que afeta a perfomance da GPU é a largura de banda ou *memory bandwidth*. Largura de banda é a taxa em que o dado pode ser lido ou armazenado na memória pelo processador, sendo medida em bytes por segundo. Para obter uma boa aplicação da GPU nos problemas científicos, é necessário identificar regiões de maior paralelismo e diminuir a taxa de transferência entre as memórias da CPU e da GPU. Caso contrário, o tempo de execução é limitado pela largura de banda da memória

da GPU empregada.

Como um código é feito de regiões que variam em graus de paralelismo e tamanho do problema, o ideal é utilizar a computação heterogênea e alocar a CPU e a GPU para onde possuem a sua maior utilidade. A lei de Amdahl (KIRK; HWU, 2010) estima o ganho de performance de programas paralelos, a equação é a razão do tempo de um programa escrito de forma serial e do tempo de um código com regiões paralelizadas. A equação é dada por:

$$P = \frac{t_{serial}}{t_{paralelo}}, \tag{4.1}$$

onde P é a performance adquirida com a paralelização e t é o tempo de execução do código. Considerando que um código pode apenas ter certas regiões paralelizadas, estima-se que $t_{paralelo}$ a partir de:

$$t_{paralelo} = (1 - \alpha_p)t + \frac{\alpha_p\, t}{n}, \tag{4.2}$$

sendo α_p uma constante proporcional à porcentagem do código paralelizável, assumindo $0 \leq \alpha \leq 1$. A constante n se refere ao número de núcleos paralelos de uma GPU e t é o tempo de execução serial do código. Esta equação é importante para constatar que mesmo possuindo uma GPU potente, parte significativa do código depende da execução serial. Logo, torna-se necessário o uso de um conjunto de potentes CPUs e GPUs integradas para obter o máximo desempenho. Para obter o ganho de performance no código paralelizado, aplica-se a equação 4.2 na razão da performance:

$$P = \frac{t}{(1 - \alpha_p)t + \dfrac{\alpha_p\, t}{n}} = \frac{1}{(1 - \alpha_p) + \dfrac{\alpha_p}{n}}\ . \tag{4.3}$$

Uma característica importante é vista ao aplicar o limite:

$$\lim_{n \to \infty} P = \frac{1}{(1 - \alpha_p)}, \tag{4.4}$$

fica visível, que mesmo com infinitos núcleos paralelizados, existe um limite teórico para o aumento de performance, que é dependente apenas da porcentagem do código paralelizada. A figura 16 mostra o que seria o aumento teórico de performance para diversos valores de α_p.

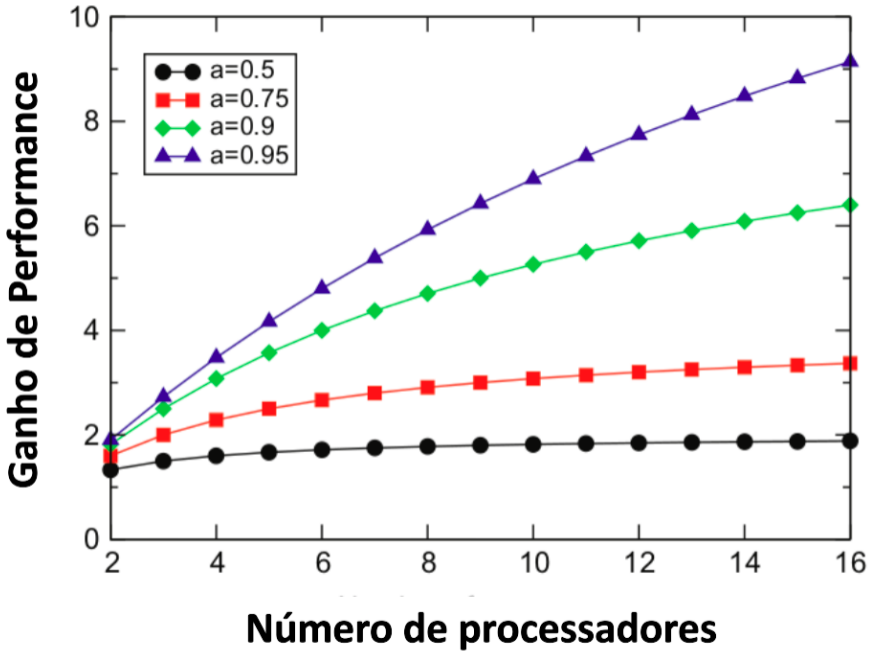


Figura 16 – Gráfico ilustrando o ganho de performânce com relação aos processadores para diversos valores de α_p. Modificado de Barlas (2014).

Analisando o gráfico, pode-se observar que os ganhos de performance são limitados, principalmente para valores baixos de α_p. A partir dos valores teóricos, a paralelização mostra-se insuficiente para gerar grandes diferenças nos tempos de execução comparado à execução serial. No entanto, o ganho de performance na prática é muito superior ao valor teórico, mostrando uma falha na lei de Amdahl para descrever os computadores modernos (BARLAS, 2014). Para descrever melhor o problema de performance, a lei de Gustafson-Barsis formula uma equação mais realista para máquinas paralelas. A proposta de Gustafson (1988) é, ao invés de analisar a relação de um programa paralelo ao serial, analisar a relação de uma máquina paralela em relação à uma sequencial para o mesmo problema.

Considerando uma aplicação paralela que demora t segundos para ser executada, o tempo de uma aplicação sequencial com n núcleos seria:

$$t_{sequencial} = (1 - \alpha_p)t + n\ .\alpha_p\ .t, \tag{4.5}$$

onde o ganho de performance pode ser calculado por:

$$P = \frac{(1 - \alpha_p)t + n\ .\alpha_p\ .t}{t} = (1 - \alpha_p) + n\ .\alpha_p \tag{4.6}$$

A figura 17 mostra o aumento da performance com relação ao número de processadores para diversos valores de α_p a partir da lei de Gustafson-Barsis.

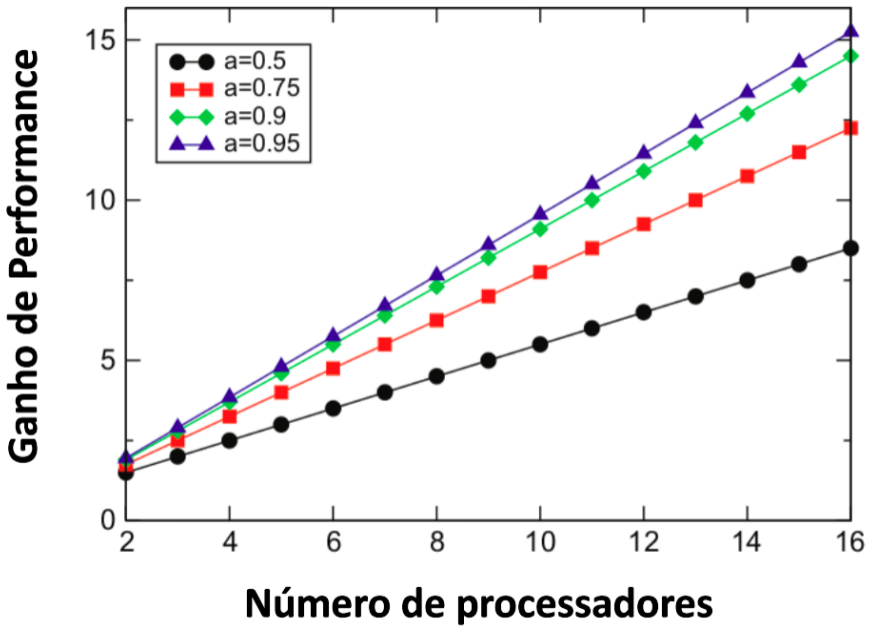


Figura 17 – Gráfico ilustrando o ganho de performânce com relação aos processadores para diversos valores de α_p para a lei de Gustafson-Barsis. Modificado de Barlas (2014).

Ao analisar o gráfico, pode-se perceber uma tendência linear para os valores de α_p como previsto na equação 4.6, demonstrando que o aumento de performance está diretamente correlacionado com o número de processadores.

Um fator importante para ressaltar é que a lei de Amdahl é pessimista, assumindo que a fração paralelizável do código é fixa e não depende do tamanho do problema. Já a lei de Gustafson-Barsis propõe que o paralelismo aumenta em uma aplicação conforme o tamanho do problema aumenta. Tal visão prevê que máquinas massivamente paralelas executam operações que antes eram consideradas impraticáveis, não só diminuindo o tempo de processamento como permitindo trabalhar com problemas grandes, tornando-a mais realista para os problemas de paralelização atual.

4.2 Programação em diretivas utilizando OpenACC

Com o aumento da aplicação de placas gráficas em computação científica, a NVIDIA® contribuiu para o avanço desta com a criação do CUDA, sigla que corresponde a *Compute Unified Device Architecture* (Arquitetura de Dispositivo de Computação Unificada) (NVIDIA; VINGELMANN; FITZEK, 2020). CUDA é uma extensão da linguagem de programação C e permite ao programador utilizar a arquitetura massivamente paralela das placas da NVIDIA®, podendo criar códigos paralelizados e executá-los quando necessários. Apesar do controle e autonomia que dá ao programador, esta extensão pode ser complexa para programadores iniciantes. A presença de variadas bibliotecas e o controle

das diversas hierarquias de memória tornam complexa a tarefa de produzir um código de alta performance. Para contornar estas complexidades, programadores têm recorrido a modelos de programação portáteis e que entregam performance. Com esta filosofia de programação, OpenACC é desenhado para dar aos pesquisadores a habilidade de programar códigos de alta performance e obter um modelo de alto nível (FARBER, 2016).

O OpenACC é um modelo que consiste em utilizar diretivas do compilador para expor paralelismos nos códigos e executá-los nos aceleradores. Inicialmente, o desenvolvedor em CUDA deveria explicitamente alocar as variáveis na GPU, transferí-las da CPU, manipulá-las na sua hierarquia de memória e trazer o resultado para a CPU de volta. Já em OpenACC, o desenvolvedor necessita apenas mapear as regiões de maior custo e paralelismo do código e inserir as diretivas para informar ao compilador a otimização desejada, sem a necessidade de explicitamente escrever novas linhas para executar esta função. Outra vantagem é que ao utilizar as diretivas para paralelizar o código, este pode ser utilizado paralelizado tanto em CPU quanto na GPU, havendo apenas a necessidade de informar esta configuração durante a compilação. OpenACC pode ser usada em linguagens de programação como C, C++ e Fortran e necessita de um compilador próprio, fornecido pelo Portland Group®.

As diretivas do OpenACC são baseadas em modelos de programação intencionalmente criados para se assemelhar ao modelo do OpenMP, usando *pragmas* para invocar o comando específico da diretiva. *Pragmas* são instruções que indicam ao compilador a tomar uma atitude a partir de determinada linha do código. Uma diretiva segue um modelo conforme:

```
1 #pragma acc <diretiva> [clausula]
```

A diretiva é uma instrução de comando para o compilador executar a ação requerida no bloco de código abaixo. No OpenACC, são definidas três tipos de diretivas:

- Diretivas de computação: São diretivas que exploram o paralelismo de um bloco de código, sendo compostos por *kernels*, *parallel*, *routine* e *loop*.
- Diretivas de controle de dados: a transferência de dados entre a CPU e a GPU pode ser executada pelo compilador. No entanto, o compilador na grande maioria das vezes não consegue encontrar a solução mais otimizada de transferência de dados. Neste contexto, esta diretiva permite ao programador o controle do fluxo de dados entre a GPU e a CPU. Dentre estas diretivas estão a data, *update*, *cache*, *atomic*, *declare*, *enter data* e *exit data*.
- Diretivas de sincronização: esta diretiva oferece ao programador a possibilidade de

executar porções do código de maneira assíncrona, permitindo otimizar regiões onde não há dependência de dado entre ambas. Estas diretivas são as *async* e *wait*.

As diretivas de computação utilizadas neste trabalho são *kernels*, *parallel* e *loop*. A *kernels* é uma diretiva que dá ao compilador total liberdade para paralelizar uma área do código, o compilador analisa esta área e decide quais as melhores otimizações a serem aplicadas. Apesar de atrativa, esta diretiva não é indicada na maioria dos casos, uma vez que não será executado nenhum tipo de paralelização caso o compilador não tenha certeza sobre a dependência de dados na região. Uma alternativa é a diretiva *parallel*. Diferentemente da *kernels*, esta diretiva é usada quando o programador ter certeza do paralelismo de uma certa região do código. Por último, a diretiva *loop* pode ser utilizada dentro de *Kernels* e *parallel*, sendo mais utilizada junto com a *parallel*. Ao indicar estas duas diretivas, indica-se que o *loop* abaixo não possui dependência de dados e pode ser paralelizado. Este tipo de combinação é usado em todos os *loops* custosos computacionalmente, podendo ser utilizada da forma:

```
1 #pragma acc parallel loop
2 for (i=0;i<N;i++){
3
4     C[i] = A[i] + B[i];
5
6 }
```

É comum encontrar *loops* duplos nos códigos, não sendo diferente neste trabalho. Uma abordagem que foi bem sucedida foi utilizar a cláusula *Collapse*. Esta cláusula colapsa o número de *loops* entrelaçados em um, assim paralelizando ambos os loops quando há independência de dados entre estes. Outras cláusulas importantes são *num worker* e *vector length*, estas cláusulas são utilizadas em todos os *loops* paralelizados, quando não chamadas são deixadas nos valores padrão de 4 e 128, respectivamente. Estas duas últimas cláusulas podem ser modificadas e o desempenho da otimização pode variar dependendo da arquiteturas de GPU utilizada. Um exemplo de uso das diretivas e cláusulas para um *loop* duplo pode ser dado por:

```
1 #pragma acc parallel loop num_worker(4) vector_length(128) collapse(2)
2 for (i=0;i<N;i++){
3     for (j=0;j<M;j++){
4
5         C[j + N*i] = A[j + N*i] + B[j + N*i];
6
7     }
8 }
```

Mesmo paralelizando o *loop*, há um problema de transferência de dados. Com esta paralelização na CPU, a execução não será afetada pois as matrizes já estão na memória RAM do processador. No entanto, na GPU, o dado é copiado e o resultado é extraido para cada execução do *loop*. Logo, o código paralelizado terá maior tempo de execução na transferência de dados que na computação da soma das matrizes. Uma solução para este problema é utilizar diretivas de controle de dados. A forma mais otimizada de controle de dados é transferir as matrizes antes da computação, declarar a presença destes dados no *loop* paralelizado e desalocar da GPU ao final do processo. Um exemplo de código de soma de matrizes na GPU é:

```
#pragma acc enter data copyin(A[0:N*M],B[0:N*M],C[0:N*M])

#pragma acc parallel loop num_worker(4) vector_length(128) \
present(A[0:N*M],B[0:N*M],C[0:N*M]) collapse(2)
for (i=0;i<N;i++){
    for (j=0;j<M;j++){

        C[j + N*i] = A[j + N*i] + B[j + N*i];

    }
}

#pragma acc exit data copyout(C[0:N*M])
#pragma acc exit data delete(A[0:N*M],B[0:N*M],C[0:N*M])
```

O presente trabalho segue este modelo de diretivas para alocar as matrizes e paralelizar os *loops*, otimizando os códigos de modelagem e inversão. A utilização das diretivas e alocação das matrizes no código de inversão FWI é melhor detalhada no capítulo de Metodologia.

5 Descrição dos modelos e geometria de aquisição

Este capítulo se destina a descrever os dois modelos utilizados no trabalho e suas respectivas geometrias de aquisição. Ambos são utilizados para as modelagens dos dados para o caso acústico e elástico, possuindo volumes de velocidade compressional, velocidade cisalhante e densidade. Os modelos escolhidos para as aplicações numéricas foram os modelos de Marmousi (VERSTEEG, 1994) e de Búzios adaptado (KARSOU et al., 2019).

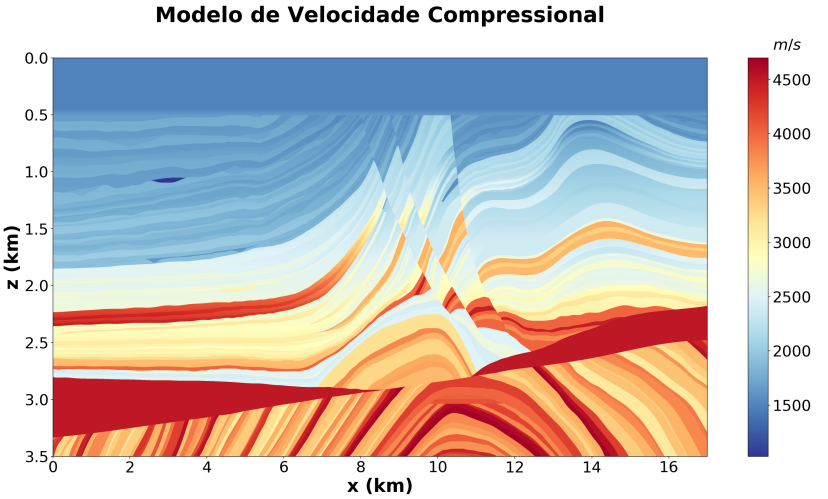
5.1 Descrição do modelo de Marmousi

No contexto de exploração sísmica, diversos algoritmos de inversão e imageamento são criados, mas sem um modelo de referência, não é possível quantificar a eficácia de cada método utilizado. Em meio ao contexto de imageamento sísmico, o modelo de Marmousi foi criado para testar a efetividade de cada nova técnica empregada. Este modelo foi baseado na geologia da bacia sedimentar de Cuanza, localizada ao norte de Angola. O modelo possui estruturas complexas como falhamentos, dobras, rochas ígneas e altas variações laterais e verticais de velocidade (VERSTEEG, 1994). Inicialmente o modelo de Marmousi possuía apenas volumes de velocidade compressional e densidade, negligenciando conversões entre ondas compressionais e cisalhantes. Sua seguinte atualização veio no trabalho de Martin, Wiley e Marfurt (2006), onde foram adicionados novos reservatórios e o volume de velocidade cisalhante. As características do grid do modelo são dadas pela tabela 3.

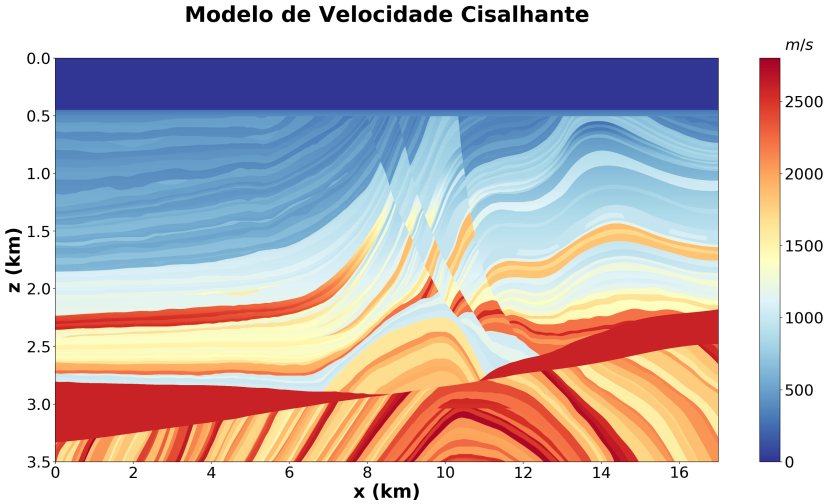
Tabela 3 – Dimensões espaciais do modelo Marmousi 2.

Dimensões Espaciais	
Nx	13601 Amostras
Nz	2801 Amostras
dx	1,25 m
dz	1,25 m
Extensão	17000 m
Profundidade	3500 m

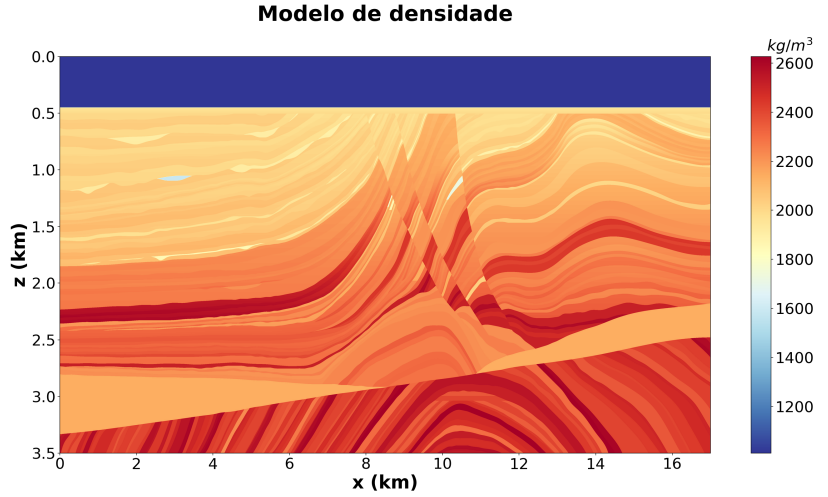
Os modelos de propriedades são representados pela figura 18, sendo estes a velocidade compressional, velocidade cisalhante e densidade, respectivamente.



(a)



(b)



(c)

Figura 18 – a) Modelo de velocidade compressional de Marmousi. b) Modelo de velocidade cisalhante de Marmousi. c) Modelo de densidade de Marmousi.

Apesar destes modelos serem uma ótima base para o modelo de Marmousi, eles não podem ser utilizados na modelagem e na inversão pois têm um grande número de amostras e um espaçamento dx e dz muito pequenos. Uma alternativa é reescalar o modelo para dimensões utilizadas na escala sísmica. O redimensionamento do modelo de velocidade é realizado utilizando uma interpolação linear, diminuindo o número de amostras do modelo sem perder as dimensões físicas. A tabela 4 mostra as dimensões do modelo de velocidade reescalado.

Tabela 4 – Dimensões espaciais do modelo Marmousi 2 reescalado.

Dimensões Espaciais	
Nx	1361 Amostras
Nz	701 Amostras
dx	12,5 m
dz	5,0 m
Extensão	17000 m
Profundidade	3500 m

As dimensões da tabela 4 foram escolhidas pois estas respeitam os critérios de instabilidade e dispersão da modelagem para uma frequência de corte de 30 Hz. As equações dos critérios de instabilidade e dispersão são apresentadas no capítulo A Utilizando o modelo reescalado, foi gerado um arranjo geométrico de aquisição sísmica para a obtenção do dado observado. A tabela 5 contém os valores utilizados para a representação da aquisição do dado observado.

Tabela 5 – Informações da aquisição feita para o modelo Marmousi 2.

Dimensões de Aquisição		
Descrição	Posição na malha	Posição (m)
Posição do primeiro tiro	11	125,0 m
Posição do último tiro	1347	16825,0 m
Espaçamento entre tiros	4	50,0 m
Profundidade da fonte	3	15,0 m
posição do primeiro receptor	0	0 m
posição do último receptor	1361	17000,0 m
Espaçamento entre receptores	1	12,5 m
Profundidade dos receptores	3	15,0 m

Informações do registro	
Número de tiros	335
Número de receptores	1361
Número de passos de tempo	16000
amostragem de tempo (dt)	0,0005 s
Tempo de registro	8 s

5.2 Descrição do modelo de Búzios

Com a descoberta de grandes reservas de óleo no pré-sal brasileiro, houve um aumento no estudo da área para a exploração de novos campos. A principal bacia produtora do Brasil é a bacia de Santos, produzindo o equivalente a 1.303 milhões de barris de petróleo por dia em 2019 (ANP, 2019). Apesar de promissoras reservas, a bacia de Santos impõe desafios para o imageamento sísmico por conta de sua geologia. A presença de rochas ígneas, falhamentos, diápiros de sal e *overhangs* dificultam o imageamento dos reservatórios do pré-sal. Apesar do modelo de Marmousi ser um bom modelo para testes de algoritmos de imageamento, a geologia encontrada na bacia de Santos se apresenta muito mais desafiadora para os algoritmos do que no modelo de Marmousi. Em meio a este contexto, este trabalho propõe utilizar um modelo de velocidade referência para a bacia de Santos, especificamente para o campo de Búzios. Para a modelagem acústica, se utiliza uma versão atualizada do modelo utilizado em Karsou et al. (2019). Este trabalho não busca descrever a geologia ou a criação do modelo, sendo detalhado no trabalho citado. O modelo utilizado mantém algumas características do modelo original, mas conta com a adição de algumas feições geológicas. A figura 19 mostra o modelo original e sua modificação com as novas feições.

Foram adicionadas as feições denotadas pelas letras A, B, C e D:

A) Decidiu-se deixar um intervalo homogêneo na região do sal com estratificação apenas em um setor.

B) Na porção B foi simulado um *overhang*, presente em algumas seções da bacia de Santos.

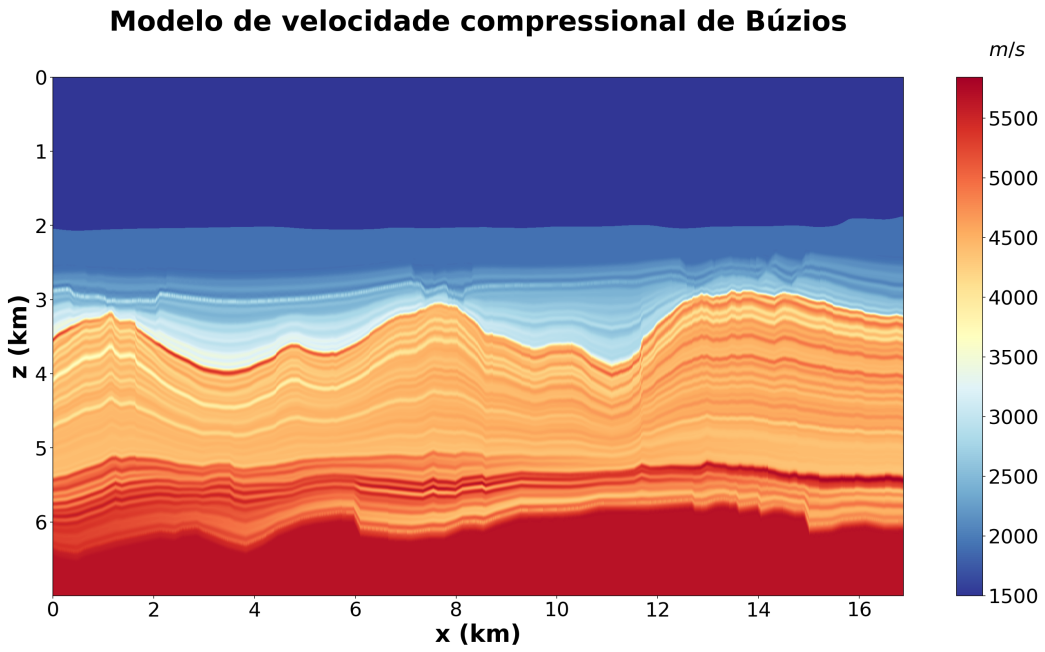
C) Foi gerado um reservatório turbidítico na região do pós-sal.

D) Por fim, simulou-se um reservatório do pré-sal abaixo do *overhang* com a intenção de verificar se a inversão FWI consegue imageá-lo.

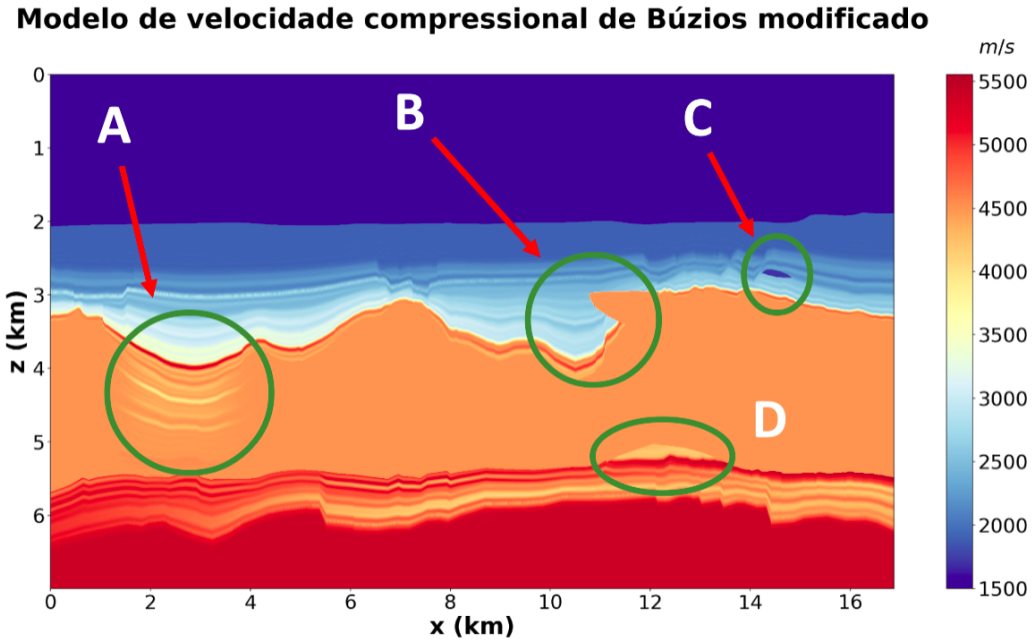
A dimensões do modelo modificado são menores que a do modelo original, representando um recorte numa área específica. A tabela 6 mostra as dimensões espaciais do modelo de Búzios.

Tabela 6 – Dimensões espaciais do modelo de Búzios.

Dimensões Espaciais	
Nx	1351 Amostras
Nz	1400 Amostras
dx	12,5 m
dz	5,0 m
Extensão	16875,0 m
Profundidade	7000 m



(a)



(b)

Figura 19 – a) Modelo de velocidade compressional de Búzios inicial. b) Modelo de velocidade compressional de Búzios modificado.

O modelo de Búzios possui quase 17 km de extensão, sendo quase idêntico em extensão mas tendo o dobro de profundidade. Por ter uma extensão similar, optou-se adotar uma geometria de aquisição do tipo *split-spread*. A tabela 7 mostra as características da geometria de aquisição utilizada.

Tabela 7 – Informações da aquisição feita para o modelo Marmousi 2.

Dimensões de Aquisição		
Descrição	Posição na malha	Posição (m)
Posição do primeiro tiro	11	125,0 m
Posição do último tiro	1335	16675,0 m
Espaçamento entre tiros	4	50,0 m
Profundidade da fonte	3	15,0 m
posição do primeiro receptor	1	0 m
posição do último receptor	1351	16875,0 m
Espaçamento entre receptores	1	12,5 m
Profundidade dos receptores	3	15,0 m

Informações do registro	
Número de tiros	332
Número de receptores	1351
Número de passos de tempo	16000
amostragem de tempo (dt)	0,0005 s
Tempo de registro	8 s

Esta geometria aquisição serve também para a modelagem elástica. No entanto, no momento em que o teste foi realizado, apenas existia o modelo de velocidade compressional, sendo necessária a criação de um modelo de velocidade cisalhante e densidade. Para isto, o modelo de velocidade compressional foi separado em quatro regiões, como mostra a figura 20.

Na figura 20 são mostrados quatro horizontes, sendo eles o fundo do mar (azul), topo do sal (verde), base do sal (amarelo) e embasamento econômico (vermelho). Desta forma, são separadas quatro zonas geológicas:

- Pós-sal: região composta de rochas sedimentares siliciclásticas.
- Sal: zona composta de rochas evaporíticas.
- Pré-sal: composta de rochas carbonáticas.
- Embasamento econômico: composta de rochas ígneas.

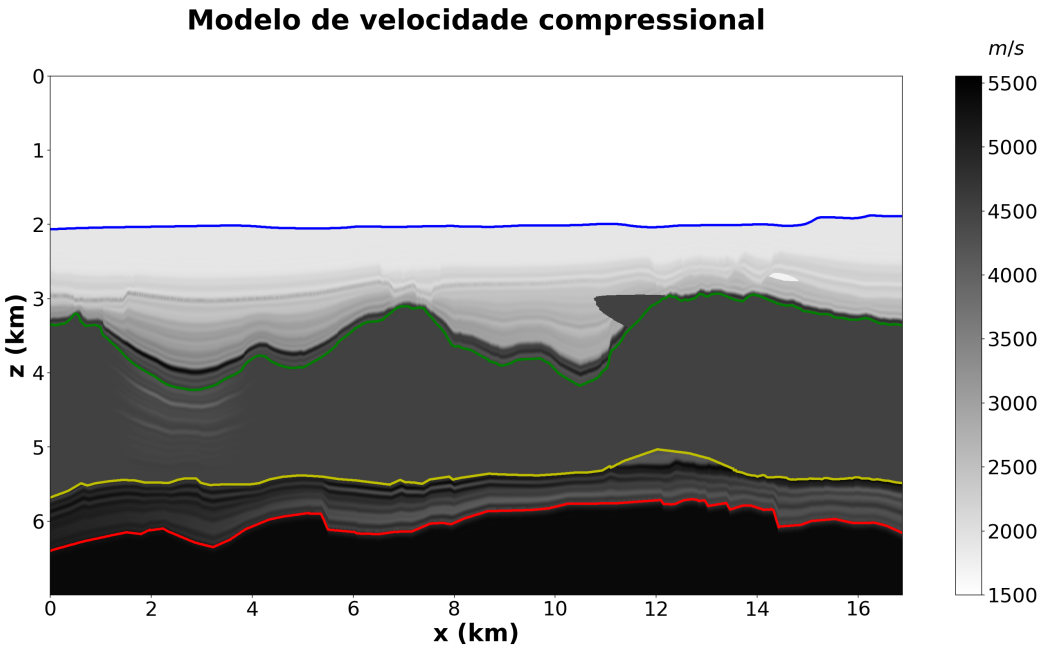


Figura 20 – Ilustração do modelo de velocidade compressional e os horizontes interpretados. Um detalhe importante é a característica do horizonte do topo do sal (verde), o qual representa o topo do sal no modelo original.

Uma abordagem utilizada comumente para gerar valores de velocidades cisalhantes é utilizar a equação 5.1:

$$v_s = \frac{v_p}{\sqrt{3}}, \tag{5.1}$$

onde v_p é a velocidade de propagação da onda compressional e v_s é a velocidade da onda cisalhante. Esta relação pode ser utilizada para a predição da velocidade cisalhante. No entanto, este tipo de abordagem pode gerar uma dependência dos eventos sísmicos com a velocidade compressional. Para contornar este problema, utiliza-se uma abordagem diferente entre v_p e v_s para cada respectiva zona. A zona do início do modelo ao fundo do mar recebe um valor nulo. Na região do pós-sal se utiliza a relação dada por Han, Nur e Morgan (1986), cuja a relação é utilizada para encontrar a velocidade cisalhante de arenitos, sendo demonstrada pela equação:

$$v_s = 0,794 v_p - 0,894, \tag{5.2}$$

onde v_p e v_s são dados em $\mathrm{km\,s^{-1}}$. Para a região do sal, utilizou-se uma relação dada por Teixeira et al. (2017). A proposta é que há uma correlação de segunda ordem, com alta confiança entre a velocidade compressional e a velocidade cisalhante. Esta relação foi selecionada pois ela encontra uma relação de v_s para rochas evaporíticas da bacia de

Santos, sendo uma região similar à do modelo. A equação para o cálculo da velocidade cisalhante é:

$$v_s = -2.407\,76 \times 10^{-4}\ v_p^2 + 2,7710\ v_p - 5099,61, \tag{5.3}$$

onde v_p e v_s são dados em $\mathrm{m\,s^{-1}}$. Para a região do reservatório do pré-sal, utilizou-se a relação de GREENBERG e CASTAGNA (1992) para carbonatos, dada por:

$$v_s = -0,0551\ v_p^2 + 1,0168\ v_p - 1,0305, \tag{5.4}$$

com v_p e v_s sendo dados em $\mathrm{km\,s^{-1}}$. Por fim, para o embasamento econômico se utiliza da relação de Carroll (1969), que correlaciona velocidade compressional e cisalhante em rochas ígneas. A equação regente desta correlação é:

$$v_s = 0,937562\ v_p^{0,81846}, \tag{5.5}$$

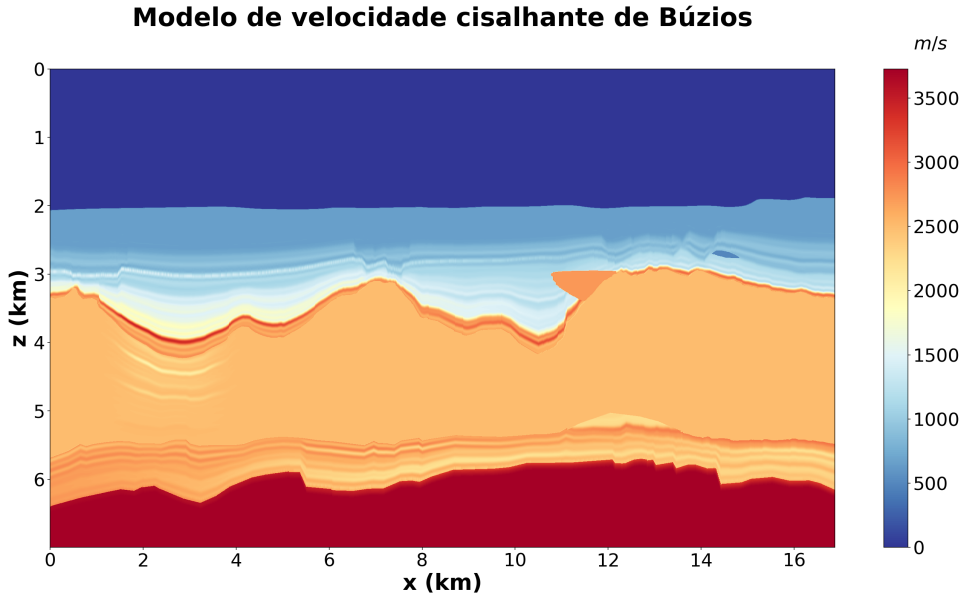
com v_p e v_s sendo dados em $\mathrm{km\,s^{-1}}$. Para gerar o modelo de densidade, o processo é mais simples. Potter e Stewart (1998) argumenta que a equação de Gardner, Gardner e Gregory (1974) é uma boa aproximação para todas as rochas sedimentares, com exceção de rochas evaporíticas. A equação de Gardner, Gardner e Gregory (1974) é expressa por:

$$\rho = 0,31\ v_p^{0,25}, \tag{5.6}$$

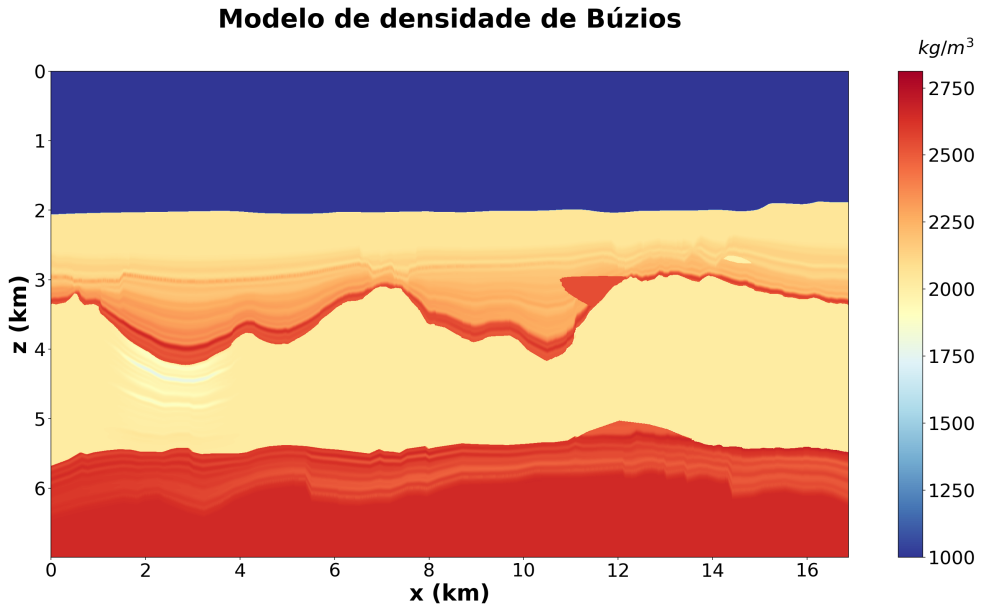
onde ρ é dado em $\mathrm{g\,cm^{-3}}$. Por conta desta equação não descrever o comportamento da densidade em rochas evaporíticas, é utilizada a equação dada por Teixeira et al. (2017), demonstrada por:

$$\rho = 2.1395 \times 10^{-7}\ v_p^2 - 1.394 \times 10^{-3}\ v_p + 3,959, \tag{5.7}$$

onde ρ é dado em $\mathrm{g\,cm^{-3}}$. Utilizando estas relações para cada correspondente região, são gerados os modelos de velocidade cisalhantes e densidade, ilustrados na figura 21.



(a)



(b)

Figura 21 – a) Modelo de velocidade cisalhante de Búzios modificado. b) Modelo de velocidade de densidade de Búzios modificado.

6 Metodologia

Este capítulo descreve os fluxos de inversão, paralelismo e de testes utilizados no trabalho, além de abordar brevemente sobre os *softwares* e *hardwares* usados para a execução e paralelização dos algoritmos de modelagem acústica, elástica e inversão FWI.

6.1 Fluxo de inversão e paralelismo

A inversão do campo de onda completo segue um fluxo pré-determinado para a obtenção de modelos de velocidade, o método resolve iterativamente diminuindo a diferença entre o dado observado e dado calculado. Para isto é encontrado um gradiente que seu negativo aponta na direção de decréscimo da função e é encontrado um passo para um modelo estimado que diminui a função objetivo. A metodologia empregada para a inversão do campo de onda completo é detalhada no fluxo da figura 22.

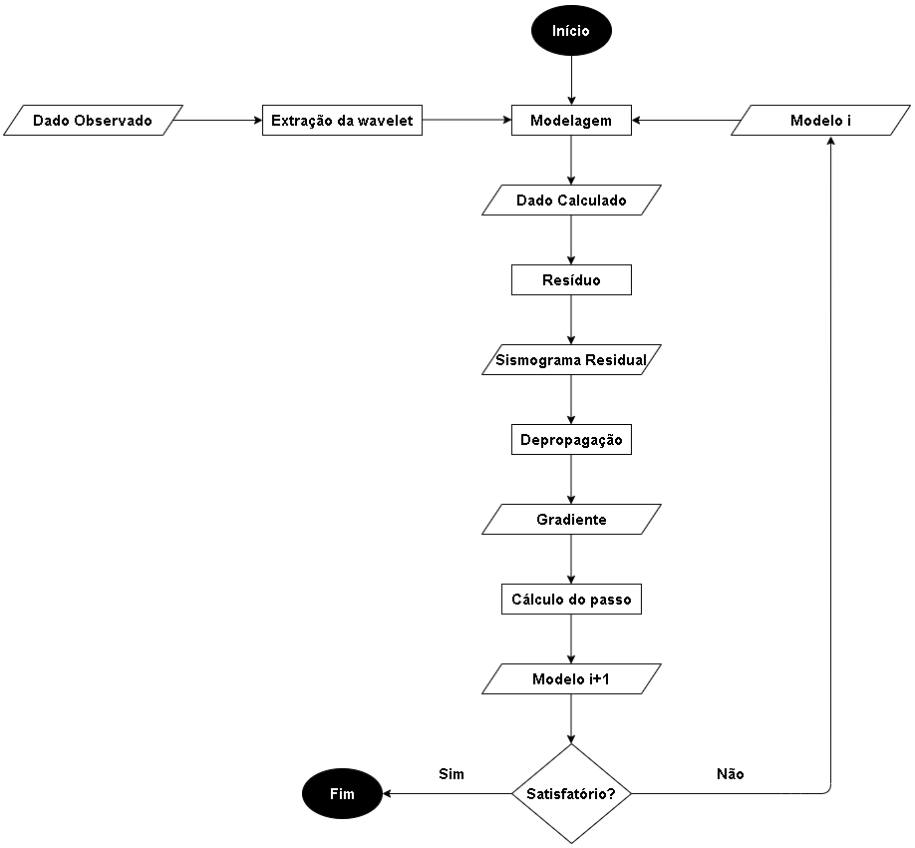


Figura 22 – Ilustração do fluxo utilizado na inversão do campo de onda completo.

O fluxo começa com a extração da a partir do dado observado. Estão presentes diversos algoritmos de estimativa de na literatura (YI et al., 2013). Porém este trabalho modela uma característica para cada frequência. Em ambientes com perda de energia por

propriedades viscoelásticas ou com variação lateral de velocidade, esta metodologia se torna falha pois esta acaba por mudar sua forma no decorrer da simulação (ALKHALIFAH, 2000; ROBERTSSON, 1994). Mas como as simulações não levam em conta os efeitos de viscoelasticidade e anisotropia, foi adotada esta forma de estimativa da *wavelet*. Com a *wavelet* estimada, utiliza-se um modelo inicial para gerar o dado calculado. A maneira mais comum de obter um modelo inicial é utilizando a tomografia de tempo de trânsito. A inversão tomográfica é capaz de gerar um modelo de velocidade suave, que respeita a cinemática do problema e próximo suficiente da solução. Para simular um modelo advindo da tomografia, aplica-se um filtro média móvel de 100 termos nos dois modelos de velocidade utilizados. A suavização é feita na vagarosidade, uma vez que a suavização na velocidade não honra o tempo de trânsito da geologia do modelo de velocidade (FARIA, 1986).

Após a obtenção do dado calculado, e consequentemente o resíduo, este é depropagado ou modelado reversamente no tempo para obter o gradiente. O gradiente obtido serve como controle de qualidade para cada iteração do algoritmo de FWI, servindo como critério de parada da inversão quando este não oferece perturbação suficiente ao modelo de velocidade. Os critérios de parada utilizados foram de 7 iterações ou 22 avaliações da função objetivo para cada banda de frequência, ao satisfazer uma destas duas condições, passa-se para a banda de frequência seguinte e o modelo de velocidade da última iteração da banda anterior é utilizado como modelo inicial para a próxima. Desta forma, o algoritmo resolve iterativamente para cada banda de frequência escolhida. Por se tratar de dados sintéticos, optou-se por fazer a inversão a cada 5 Hz, uma vez que a convergência em dados sintéticos é maior que em dados reais. Outro fator para tal abordagem é que fazer em intervalos menores impactará negativamente no custo computacional, necessitando um maior tempo de máquina. Portanto, optou-se por utilizar de frequência máxima de 20 Hz.

Dado o fluxo de inversão, necessita-se de uma estratégia para paralelização do código de inversão FWI. Utilizando as diretivas fornecidas pelo OpenACC, é possível montar um algoritmo que controla o fluxo de dados entre o *Host* e a GPU. Os fluxos de paralelização dos códigos de modelagem acústica, modelagem elástica e inversão FWI são descritos no apêndice C. Na etapa de paralelização do código de inversão FWI, todos as variáveis são inicializadas, alocadas e zeradas na CPU. Após esta etapa, são transferidas para GPU, de onde não saem até o cálculo do último tiro, quando se retira o vetor de gradiente e são deletados todas as outras variáveis na GPU. Para o cálculo da matriz de derivadas, decidiu-se gravar alguns campos no intervalo de correlação e calcular as derivadas a partir destes campos de onda gravados. A correlação para obtenção do gradiente é feita para a cada 20 amostras de tempo (0,01 s). Para otimizar a questão da memória da GPU, que se tornaria problemática para modelos muito grandes, a região do fundo do mar é ignorada no cálculo do gradiente e não se é gravado o campo na região do mar, desta forma economizando parte da memória da GPU. Esta estratégia também é empregada no

código de migração RTM, visto que ambos possuem *kernel* parecido do ponto de vista computacional. Para a migração reversa no tempo (*Reverse Time Migration* ou Migração RTM), é utilizada a condição de imagem compensação de iluminação, descrita no apêndice B. Durante esta etapa, é modelada a onda direta para remover a onda direta contida nos sismogramas do dado observado, com um posterior *mute* das informações de ondas refratadas presentes em longos espaçamentos. Esta etapa é utilizada para aprimorar a qualidade da migração, retirando os efeitos de ondas refratadas que agem com interferência negativa no algoritmo. Para aprimorar a imagem obtida, é aplicado o filtro laplaciano da Gaussiana (SOTAK; BOYER, 1989) para realçar os contrastes presentes na imagem.

6.2 Metodologia de testes

Para quantificar e analisar os resultados obtidos das paralelizações e da inversão FWI, são utilizados critérios de análise de performance e dos modelos de velocidade obtidos. Inicialmente é feito um estudo do impacto da diferença finita no tempo de execução, executando o código de modelagem acústica utilizando CPML e medindo seu tempo de execução para cada ordem de diferença finita. São medidos os tempos de execução da segunda até a décima ordem da diferença finita para os códigos serial denominado de *Host*, CPU paralelizado denominado de *Multicore* e para a GPU. Após isto, é utilizada a equação 4.1 para analisar o ganho de performance de cada configuração com relação ao código serial. Em seguida, esta mesma rotina de testes é feita para o caso elástico até a quarta ordem.

Após a fase de teste da diferença finita, são expressos os valores de tempo de execução de um tiro e o tempo estimado para uma aquisição de múltiplos tiros utilizando o *Host*, *Multicore* e GPU. Após uma breve discussão sobre os resultados, é feita uma geração dos dados utilizando a GPU e se faz um breve comentário sobre o tempo estimado da aquisição para a GPU e seu tempo medido, com este fluxo sendo feito tanto para o caso acústico quanto para o caso elástico. Dados os testes para aquisição, é feita a mesma comparação mas para uma iteração do código de inversão FWI, medindo o tempo de execução de um tiro e de uma aquisição completa para uma iteração, comentando e comparando o valor real medido na GPU com o estimado anteriormente a partir de um único tiro. Calculados e comparados os tempos de execução, é mostrada e comentada uma tabela da memória necessária dos algoritmos de modelagem acústica, modelagem elástica e inversão FWI.

Passando a etapa de testes do tempo de execução e memória, são feitos testes para avaliar os resultados dos modelos de velocidade advindos da inversão FWI. Para obter o modelo inicial, suaviza-se o modelo original na vagarosidade utilizando um filtro média móvel de 100 pontos. Após a obtenção do modelo inicial, é feita a inversão FWI utilizando

quatro bandas de frequência, sendo as bandas de 0-5 Hz, 0-10 Hz, 0-15 Hz e de 0-20 Hz. O modelo final e o gráfico da função objetivo de acordo com as iterações são mostrados e analisados para todas as bandas até a frequência de corte de 20 Hz. Ao se obter o último modelo, é utilizada a equação 6.1 para analisar a diferença entre o modelo de velocidade advindo da inversão e o modelo inicial com relação modelo real. É calculada a diferença entre os modelos obtidos e o verdadeiro a partir da equação do erro percentual relativo:

$$D = 100\frac{\|M_r - M_o\|}{M_r}, \tag{6.1}$$

onde M_r é o modelo real e M_o é o modelo obtido, sendo o modelo inicial ou o modelo final da inversão. A diferença é utilizada para analisar as regiões onde há maior discrepância entre ambos modelos. Devido ao desconhecimento do modelo real em aquisições não sintéticas, é feita a diferença entre o modelo final e o modelo inicial para analisar o quanto o modelo inicial foi modificado. É feita uma migração RTM para analisar a qualidade da inversão a partir da estimativa inicial utilizada e uma migração RTM no modelo final obtido da inversão. Estes resultados são comentados e são discutidos os impactos da definição do modelo de velocidade na imagem final. Por fim, são apresentadas uma tabela com o número de avaliações da função objetivo e o tempo de execução para cada banda e uma tabela com os valores de tempo de execução estimados baseados no número de avaliações da função objetivo. Os valores de tempo de execução real e estimado são comparados e são feitos comentários quanto à proximidade da estimativa e do valor real medido. O fluxo de teste descrito é feito de forma idêntica para os modelos de Marmousi e de Búzios.

6.3 *Softwares* e *hardwares* utilizados

Para a execução e compilação dos códigos de modelagem e inversão, são utilizados diversas linguagens de programação e interfaces em combinação. O código fonte da modelagem e inversão FWI são escritos em C, sendo paralelizados e compilados utilizando o OpenACC da Portland Group®. São utilizadas as linguagens *Python* e C conjuntamente. Os códigos em C geram os dados e fazem as computações mais custosas, já o código em *Python* é responsável por passar os parâmetros de aquisição e realizar o processo de otimização. O processo de otimização é feito utilizando a biblioteca *Scipy* do *Python*, usando a função *minimize* para gerar os modelos de velocidade atualizados e requerer a avaliação das funções objetivo.

Em relação aos *hardwares* utilizados no experimento, é utilizada um processador Intel® Core^{TM} i7-9700 e uma placa gráfica NVIDIA® GeForce^{TM} RTX 2060. As especificações do processador são dadas pela tabela 8.

Tabela 8 – Especificações do processador utilizado.

Processador	Intel® Core^{TM} i7-9700
Número de *cores*	8
Memória	16 Gb
Frequência do Processador	3 GHz
Largura de banda	até 41,6 Gb/s

Este processador é de 9ª geração da Intel® e com boa capacidade de lidar com os problemas 2D utilizados no trabalho. Possui uma memória RAM de 16 Gb, 3 GHz de frequência (*clock*) e largura de banda de até 41.6 Gb/s. A tabela 9 mostra as especificações da placa de vídeo RTX 2060 utilizada.

Tabela 9 – Principais características da placa gráfica empregada.

Placa	GeForce^{TM} RTX 2060
CUDA *Cores*	1920
Memória	6 Gb
Frequência do Processador	1,365 GHz
Largura de Banda	até 336 Gb/s

A placa gráfica possui menor memória e uma menor frequência (*clock*) que o processador. No entanto, a placa gráfica possui um número bem mais elevado de núcleos e largura de banda, possibilitando uma alta capacidade de cálculo e processamento de um grande volume de dados. A partir dos *hardwares* descritos, estes são utilizados em conjunto para executar os códigos de modelagem acústica, modelagem elástica, inversão FWI e migração RTM utilizados no trabalho.

7 Resultados e discussão

Este capítulo é separado em resultados de performance computacional e resultados de inversão. Ambos os resultados são apresentados utilizando os modelos de Marmousi e o modelo de Búzios, sendo discutidos individualmente em cada subseção.

7.1 Resultados de performance computacional

Modelo Marmousi

A ordem da diferença finita é um fator importante na modelagem. Como o método usa a aproximação da derivada por série de Taylor, a ordem define o quão próximo da solução real a solução numérica está. Portanto, uma ordem maior possui maior precisão que as menores, mas com um custo computacional atrelado. Baseado nisto, deseja-se verificar o poder da paralelização no tempo de execução em cada configuração. Para isto, é gerada uma tabela com valores de tempos de execução para cada ordem de diferença finita. A tabela 10 mostra uma tabela de valores de tempo de execução de ordens de diferença finita correspondente a cada paralelização. São designados como *Host* o código serial e *Multicore* é o código paralelizado na CPU. Os valores de tempo de execução da tabela 10 correspondem ao modelo de Marmousi utilizando a borda CPML.

Tabela 10 – Tempos de execução de cada ordem de diferença finita para as diferentes configuração de paralelização no caso acústico.

	Host	Multicore	GPU
2ª ordem	1m 53,13s	33,35s	8,32s
4ª ordem	2m 10,89s	35,93s	8,46s
6ª ordem	2m 54,84s	39,97s	8,31s
8ª ordem	3m 08,74s	45,88s	8,31s
10ª ordem	3m 32,13s	47,50s	8,40s

Analisando a tabela 10, verifica-se um aumento do tempo de execução com relação ao aumento da ordem da diferença finita para o *Host* e *Multicore*, um resultado esperado para arquiteturas em CPU. Utilizando o *Multicore*, há um ganho de performance em relação ao *host*, porém tendo o mesmo crescimento com relação ao tempo de execução. Já analisando os valores da GPU, pode-se verificar que a ordem da diferença finita basicamente não altera o tempo de execução. No entanto, é necessário utilizar uma métrica para analisar o ganho em cada configuração. Para analisar o ganho de performance de cada configuração, usa-se a equação 4.1 para gerar os valores mostrados na figura 23.

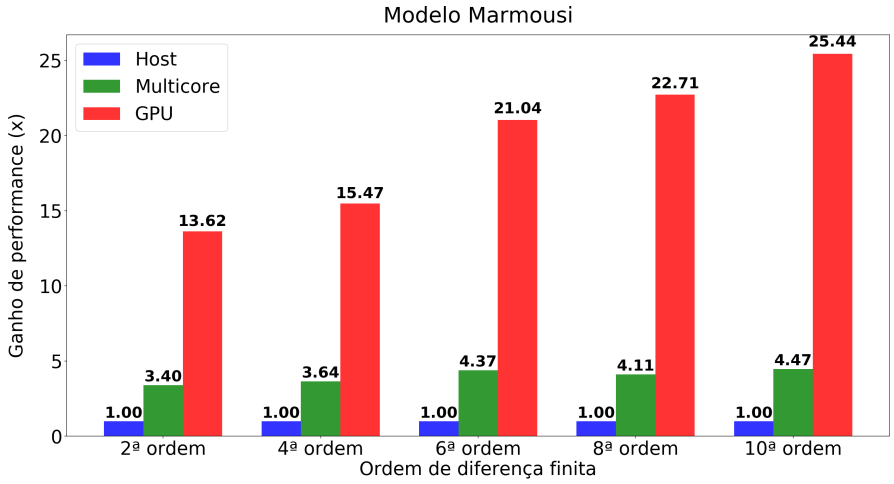


Figura 23 – Ganho de performance de cada paralelização em relação ao código serial para o caso acústico.

O gráfico demonstra uma tendência de aumento do ganho de performance da GPU em relação à CPU conforme se aumenta a ordem, isso ocorre pois o tempo de execução das outras configurações aumenta com a ordem enquanto que o da GPU se mantém relativamente constante, oscilando em torno de 4x. Os valores de ganho de performance para a GPU variam entre aproximadamente 13x para a segunda ordem e até 25x o código serial para a décima ordem. Este é um ótimo resultado, pois dá a liberdade de se usar uma ordem de alta precisão sem afetar no tempo de execução. Outro detalhe importante a ressaltar é a constância do ganho de performance da *Multicore*, isto se dá pois a CPU é um dispositivo de baixo paralelismo comparado à GPU, otimizando proporcionalmente o tempo de execução de acordo com o número de núcleos. Já a GPU, diferentemente da CPU, não possui linearidade pois este dispositivo se desempenha bem para uma quantidade maior de dados e operações aritméticas. Devido ao comportamento da GPU com relação às ordens, decidiu-se utilizar a ordem 10 para as modelagens, inversões e migrações.

Com os testes da tabela 10 sendo realizados para a equação da onda acústica, são necessários testes para verificar o impacto das ordens na execução da equação da onda elástica. Os testes são feitos no modelo elástico de Marmousi. A tabela 11 contém os valores do tempo de execução para cada ordem de diferença finita para o modelo elástico até a quarta ordem.

Tabela 11 – Tempos de execução de cada ordem de diferença finita para as diferentes configuração de paralelização no caso elástico.

	Host	Multicore	GPU
2ª ordem	32m 53,54s	9m 8,38s	1m 11,80s
4ª ordem	46m 16,11s	11m 51,11s	1m 14,13s

Este caso demanda tanto computacionalmente que um tiro utilizando a quarta

ordem leva pouco mais de 46 minutos para ficar pronto no *Host* e 11 minutos para o *Multicore*. A figura 24 ilustra o ganho de performance a partir dos valores da tabela 11.

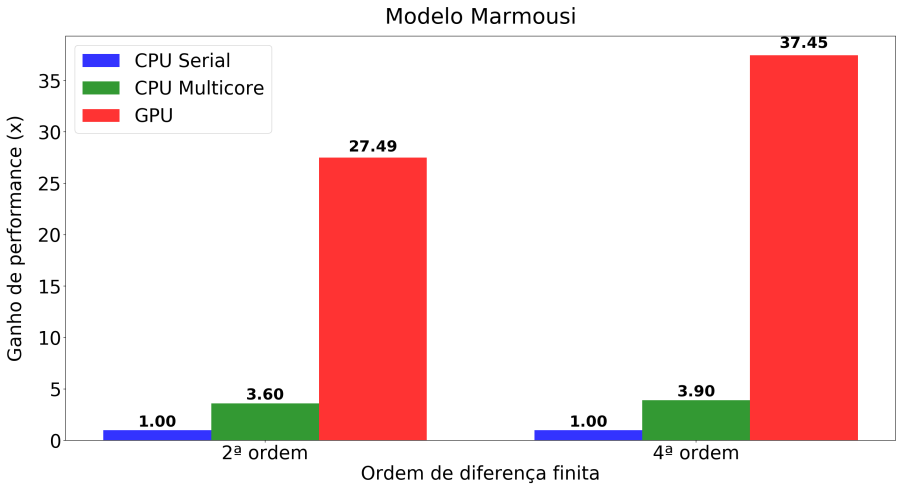


Figura 24 – Ganho de performance de cada paralelização em relação ao código serial para o caso elástico.

Verifica-se um ganho de performance de 27x para a segunda ordem, diferentemente do caso acústico onde o ganho era de 13x. Isto é resultado da eficiência da GPU para grandes problemas e grandes dados, otimizando em até 37x para a quarta ordem. Estes resultados incentivam o uso de arquiteturas heterogêneas para otimização dos códigos de modelagem elástica.

Passando do problema de diferenças finitas, deseja-se comparar o tempo de execução para a modelagem de todos os sismogramas da aquisição. Logo, modela-se um tiro da aquisição e multiplica-se o tempo de execução desse tiro pelo total de tiros da aquisição para se ter um tempo estimado. Isto é feito para a *Host*, *Multicore* e GPU. A tabela 12 ilustra o tempo estimado e o tempo de execução para cada configuração na geração do dado observado acústico.

Tabela 12 – Tempos de execução de cada configuração para o dado observado acústico.

	Host	Multicore	GPU
Tempo 1 tiro	3m 33,26s	47,65s	8,02s
Tempo estimado 335 tiros	19h 50m 42,10s	4h 26m 2,75s	44 m 46,70s

Analisando a tabela 12, verifica-se que os tempos estimados para o *Host*, *Multicore* e GPU são discrepantes, com valores variando de aproximadamente 20 horas para ser completo no código serial e de apenas 44 minutos para a GPU. Não se é gerada aquisição completa para o *Host* e o *Multicore* pois estas exigiriam um custo computacional adicional para seu cálculo. No entanto, é calculado o tempo de aquisição utilizando a GPU, completando a geração do dado em 33 minutos e 57,14 segundos, valor aproximadamente 11

minutos menor que o estimado pela tabela. Para o caso elástico, a tabela 13 mostra os valores do tempo de execução de um tiro e o estimado para aquisição.

Tabela 13 – Tempos de execução de cada configuração para o dado observado elástico.

	Host	Multicore	GPU
Tempo 1 tiro	46m 16,11s	11m 51,11s	1m 14,13s
Tempo estimado 335 tiros	10d 16h 1m 8,51s	2d 17h 34m 48,51s	6h 50m 11,15s

Analisando a tabela 13, observa-se que o valor estimado para o dado sintético elástico pode chegar a aproximadamente 10 dias e 16 horas, valor acima da configuração *Multicore*, que ficou em torno de 2 dias e 17 horas aproximadamente. Para a GPU foi estimado um valor de 6 horas e 50 minutos. Ao fazer a modelagem do dado sintético, o valor medido de tempo de execução foi de 6 horas, 45 minutos e 29,68 segundos, valor aproximadamente 5 minutos menor que o estimado.

Após o estudo do tempo de execução para cada configuração na geração dos dados de aquisição, é analisado o resultado da inversão FWI para o modelo de Marmousi para uma avaliação da função objetivo. A tabela 14 contém os valores dos tempos estimados e adquiridos de tempo de execução.

Tabela 14 – Tempos de execução de cada configuração de paralelização para uma avaliação da função objetivo da inversão FWI.

	Host	Multicore	GPU
Tempo 1 tiro	7m 47,96s	1m 45,75s	16,88s
Tempo estimado 84 tiros	10h 55m 8,64s	2h 28m 3.0s	23m 37,92s

Observando os tempos de execução, pode-se verificar que o valor estimado para o *Host* necessita de aproximadamente 11 horas para calcular uma avaliação da função objetivo, valor superior às aproximadas 2 horas do *Multicore* e 23 minutos da GPU. Pode-se notar que a GPU oferece um ganho no custo computacional, pois não se necessita horas para avaliar a função objetivo e pode-se utilizar de mais iterações devido ao baixo custo. O tempo real medido da aquisição foi de 19 minutos e 15,39 segundos, tempo aproximadamente 4 minutos inferior que o valor estimado, amplificando a portabilidade do uso de GPU no problema de FWI. Um ponto importante é o espaço necessário na memória da GPU para cada código, uma vez que problemas muito grandes não caberiam em sua memória. A tabela 15 mostra em megabytes o espaço necessário para cada algoritmo.

Tabela 15 – Memória necessária na GPU dos algoritmos de modelagem acústica, modelagem elástica e inversão FWI para o modelo de Marmousi.

	Modelagem Acústica	Modelagem Elástica	Inversão FWI
Utilização na memória	235 Mb	601 Mb	2997 Mb

Verificando a tabela, pode-se perceber como esperado, que a modelagem acústica é o algoritmo com menor ocupação de memória. A modelagem elástica possui aproximadamente o triplo de espaço ocupado, ocupando 601 Mb da GPU. Por último, a FWI é o algoritmo que mais requer memória. Isto ocorre pois este requer a gravação das derivadas do campo modelado, sendo guardadas na memória da GPU para agilizar a operação do cálculo do gradiente.

Modelo Búzios

De forma similar ao que foi realizado no modelo de Marmousi, é feito um estudo do impacto da ordem da diferença finita no modelo de Búzios. Os resultados são mostrados na tabela 16.

Tabela 16 – Tempo de execução em segundos de cada ordem de diferença finita para cada configuração de paralelização para o caso acústico.

	Host	Multicore	GPU
2ª ordem	3m 28,08s	1m 6,72s	11,18s
4ª ordem	4m 4,6s	1m 12,72s	11,27s
6ª ordem	5m 20,47s	1m 18,87ss	11,32s
8ª ordem	5m 47,92s	1m 28,54s	11,40s
10ª ordem	6m 31,33s	1m 31,60s	11,80s

Analisando a tabela 16, verifica-se que estes resultados se assemelham muito ao caso acústico do modelo de Marmousi, exibindo uma tendência linear do tempo de execução com a ordem quando utilizando o *Host* e *Multicore*, mas mantendo a semelhança nos valores utilizando a GPU. Para analisar melhor o ganho de performance de cada configuração, a figura 25 mostra o gráfico de ganho de performance para cada ordem de diferença finita.

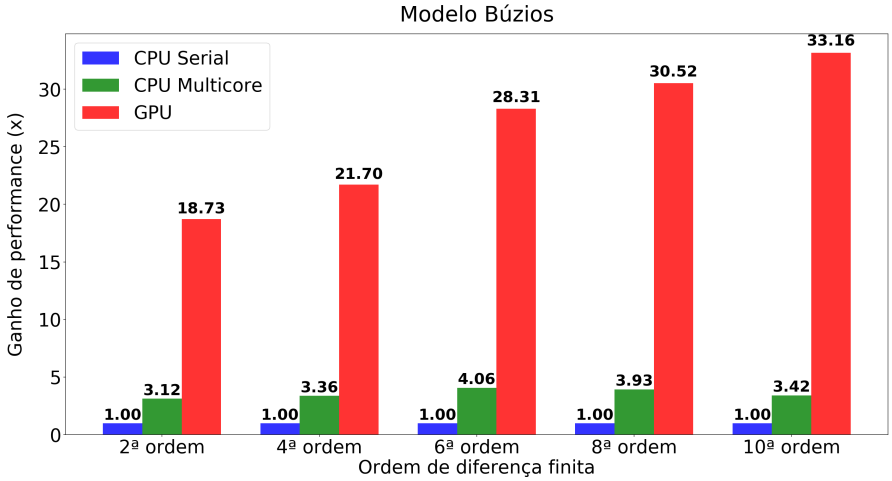


Figura 25 – Ganho de performance de cada paralelização em relação ao código serial para o caso acústico.

Pode-se analisar que os valores do ganho de performance estão maiores que para o modelo de Marmousi. Isto ocorre pois a GPU possui desempenho muito melhor para problemas maiores, entregando valores de ganho de performance variando de 18x para a segunda ordem até 33x para a décima ordem de diferença finita. Fazendo a mesma análise, mas para o caso elástico, gera-se uma tabela de tempo de execução para cada ordem dado pela tabela 17.

Tabela 17 – Tempo de execução em segundos de cada ordem de diferença finita para cada configuração de paralelização para o caso elástico.

	Host	Multicore	GPU
2ª ordem	37m 1,01s	11m 2,37s	1m 23,12s
4ª ordem	52m 38,54s	13m 31,02s	1m 25,23s

De acordo com a tabela 17, os valores para o *Host* crescem consideravelmente, atingindo aproximadamente 52 minutos para a quarta ordem. Já há uma diminuição para o *Multicore*, chegando até aproximadamente 13 minutos. A GPU ofereceu um aumento, mas não tão elevado quanto às outras configurações. A figura 26 mostra o ganho de performance para cada ordem de diferença finita.

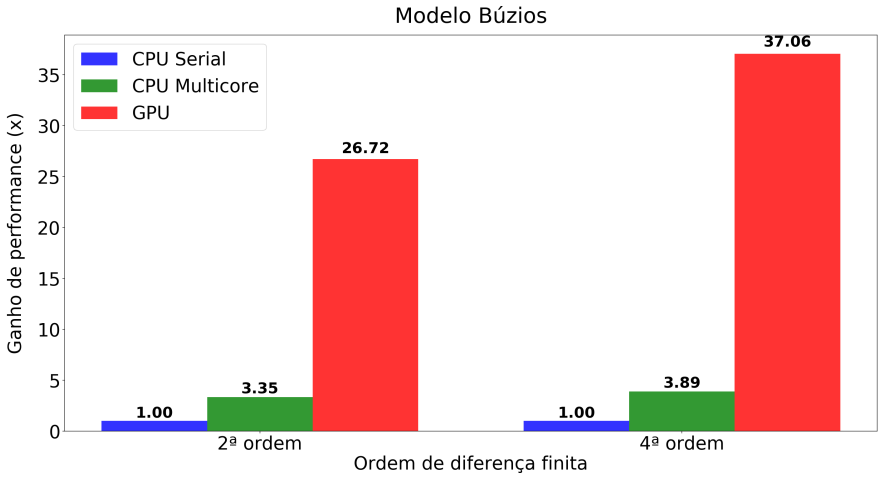


Figura 26 – Ganho de performance de cada paralelização em relação ao código serial para o caso elástico.

O ganho de performance ficou similar ao modelo de Marmousi, variando entre aproximadamente 26x a 37x para a GPU. Devido ao número de pontos, esperava-se um maior ganho de performance por conta da não linearidade do ganho de performance da GPU com relação ao tamanho do problema. Passando para o problema da geração do dado observado, a tabela 18 contém os valores de tempo de execução medidos de um tiro e os estimados para aquisição para o dado sintético acústico.

Tabela 18 – Tempos de execução de cada configuração de paralelização para o dado observado.

	Host	Multicore	GPU
Tempo 1 tiro	6min 29,77s	1m 31,36s	11,41s
Tempo estimado 332 tiros	1d 11h 56m 43,64s	8h 25m 31,51s	1h 3m 8,12s

Analisando a tabela 18, pode-se verificar que foi estimado um total de aproximadamente 1 dia e 12 horas para a geração do dado sintético utilizando o *Host*, 8 horas e 25 minutos para o *Multicore* e 1 hora e 3 minutos para a GPU. O valor medido para o tempo da execução da GPU foi de 52 minutos e 1,49 segundos, valor aproximadamente 11 minutos menor. Fazendo a mesma comparação para uma avaliação da função objetivo da inversão FWI, gera-se a tabela 19

Tabela 19 – Tempos de execução de cada configuração de paralelização para a inversão FWI.

	Host	Multicore	GPU
Tempo 1 tiro	14m 22,37s	3m 25,90s	29,75s
Tempo estimado 83 tiros	19h 52m 56,71s	4h 44m 49,70s	41m 9,25s

Observando a tabela, os valores mostram um estimado de 19 horas e 52 minutos para o código serial, 4 horas e 44 minutos para o código *Multicore* e de 41 minutos para a GPU. O valor medido foi de 27 minutos e 5,23 segundos, valor bem abaixo do estimado, mostrando a não linearidade da GPU para modelos grandes como o de Búzios. Por fim, a mesma análise é feita para a modelagem elástica, com os valores de tempo de execução dados pela tabela 20.

Tabela 20 – Tempos de execução em segundos de cada ordem de diferença finita para cada configuração de paralelização para o caso elástico.

	Host	Multicore	GPU
Tempo 1 tiro	52m 38,54s	13m 31,02s	1m 25,23s
Tempo estimado 332 tiros	12d 3h 17m 15,28s	3d 2h 47m 38,64s	7h 51m 36,36s

Observando os valores da tabela, o tempo estimado da aquisição utilizando a configuração serial seria de 12 dias e 3 horas aproximadamente. Para o *Multicore* seria de 3 dias e 2 horas aproximadamente, valor relativamente superior ao de 7 horas e 51 minutos estimado pela GPU. O valor medido da aquisição foi de 7 horas, 49 minutos e 53,20 se, aproximadamente 2 minutos a menos que o estimado pela tabela. Para o caso elástico, o valor estimado e o calculado para a GPU ficaram bem próximos, diferentemente dos resultados para a aquisição acústica. Analisando o consumo de memória de cada algoritmo

para o modelo de Búzios, a tabela 21 exibe os valores de memória para cada algoritmo na GPU.

Tabela 21 – Memória necessária na GPU dos algoritmos de modelagem acústica, modelagem elástica e inversão FWI para o modelo de Marmousi.

	Modelagem Acústica	Modelagem Elástica	Inversão FWI
Utilização na memória	279 Mb	665 Mb	4845 Mb

Pode-se notar que os valores aumentaram levemente em relação ao modelo de Marmousi para os códigos de modelagem acústica e elástica. Já para a inversão FWI houve um aumento significativo, uma vez que em um modelo maior, mais pontos da matriz de derivada do campo de onda modelado são armazenados, aumentando proporcionalmente o custo de memória.

7.2 Resultados da inversão FWI

Esta seção é destinada à análise dos resultados da inversão do campo de onda completo, obtendo os modelos de velocidade para cada banda de frequência e analisando-os para os modelos de Marmousi e Búzios.

Modelo Marmousi

Para se iniciar a inversão FWI, deve-se escolher um modelo inicial para se otimizar. O modelo inicial utilizado é uma suavização do modelo de Marmousi, obtido a partir da suavização na vagarosidade. O modelo suavizado é dado pela figura 27.

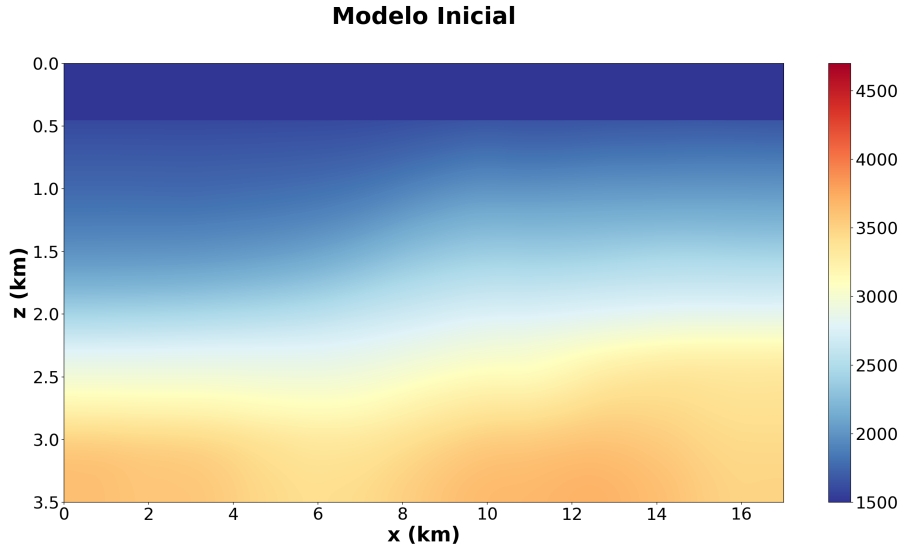
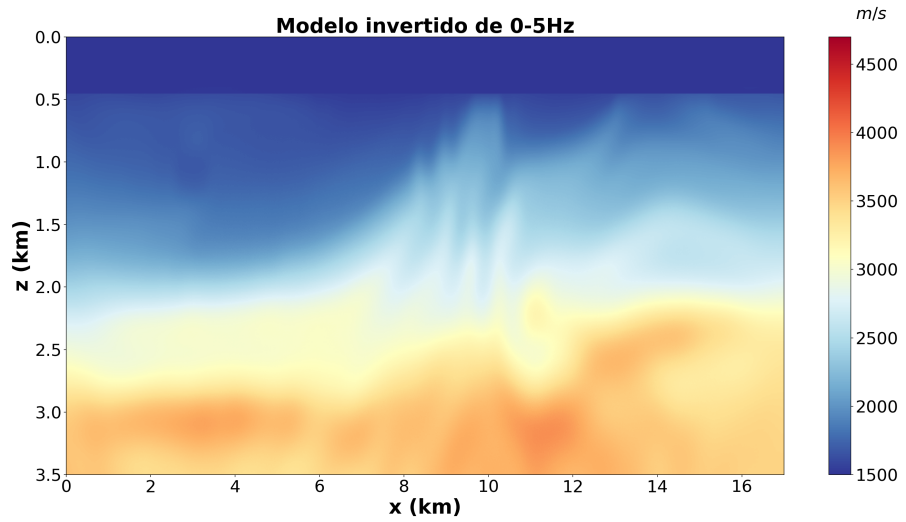


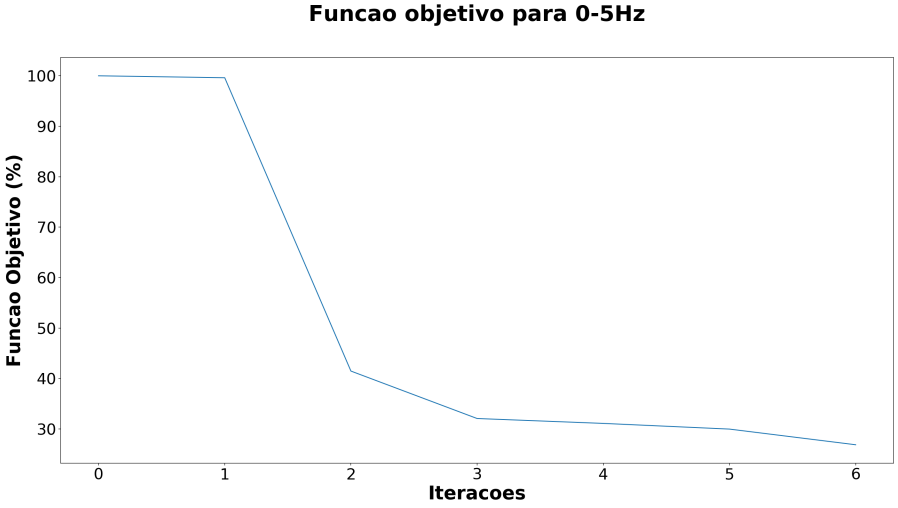
Figura 27 – Modelo de velocidade inicial utilizado para a inversão FWI no modelo de Marmousi.

Pode-se notar que o modelo suavizado aparentemente pouco se assemelha ao modelo de Marmousi, preservando apenas informações de baixa frequência espacial. Esta abordagem foi utilizada para simular um modelo inicial advindo da análise de velocidade ou da tomografia de tempo de trânsito, sendo etapas do processamento e imageamento sísmico.

Com o intuito de recuperar um modelo mais verossímel, é feita a inversão FWI para gradativamente obter valores menores de diferença relativa. A figura 28 mostra o resultado da inversão utilizando 0-5Hz e o gráfico de sua função objetivo.



(a)

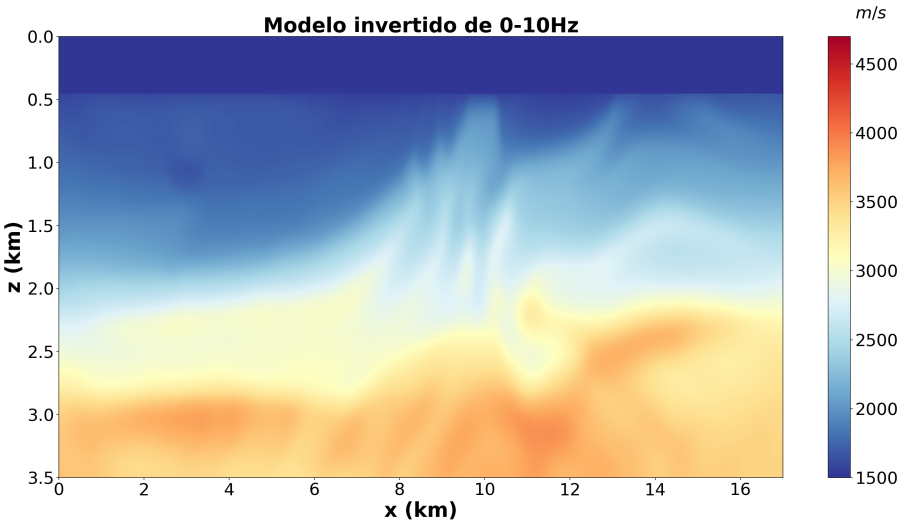


(b)

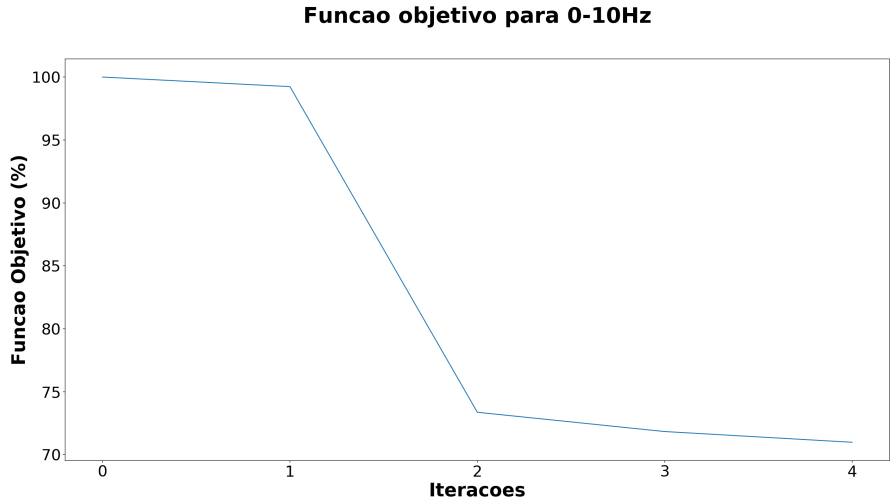
Figura 28 – a) Modelo de velocidade obtido da inversão para a banda de 0-5Hz. b) Gráfico da função objetivo para a banda de 0-5Hz.

Pode-se notar uma diferença do modelo obtido a partir desta inversão com o da figura 27, estruturas de falhas do modelo passam a ser aparentes, também é mostrada a região do sal nas laterais do modelo. A função objetivo mostra que 22 avaliações foram

suficientes para 6 iterações. A função objetivo decresceu consideravelmente e atingiu um valor mínimo de $26,85\%$ do valor inicial. Continuando o processo de inversão, obtém-se um modelo de velocidade para a banda de 0-10Hz, dado pela figura 29.



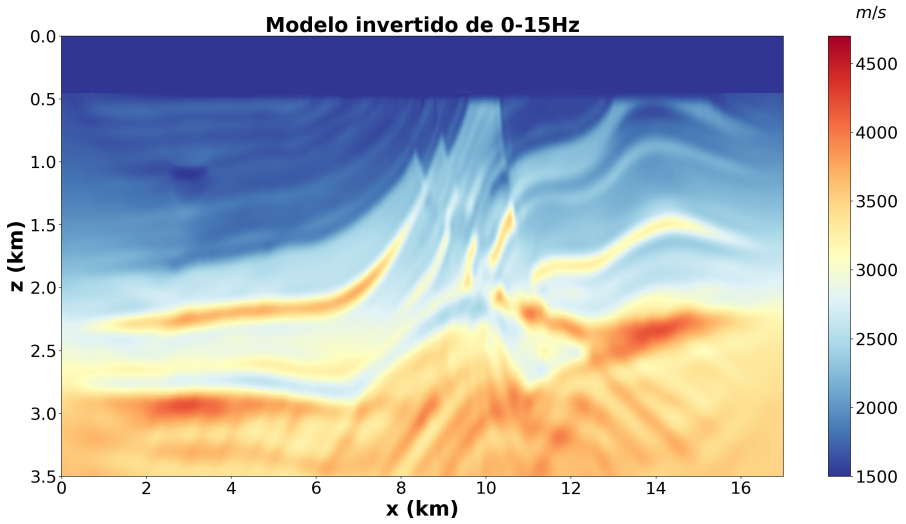
(a)



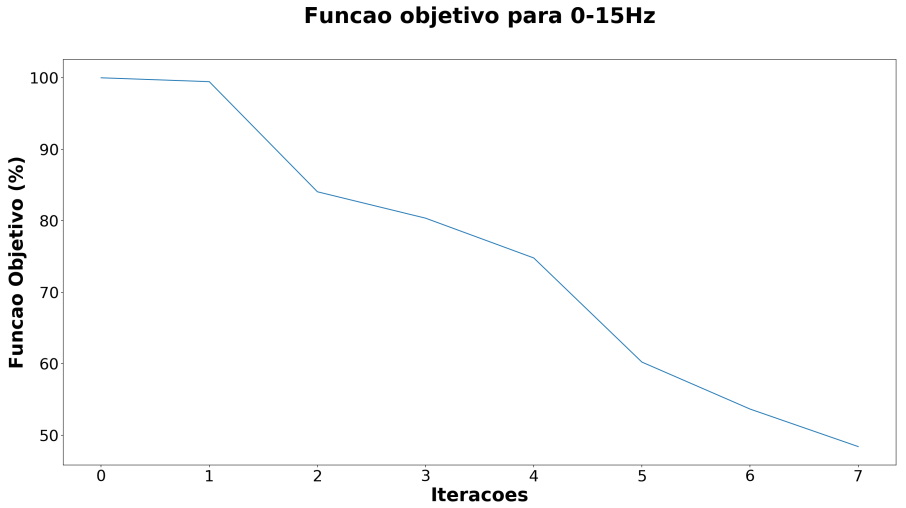
(b)

Figura 29 – a) Modelo de velocidade obtido da inversão para a banda de 0-10Hz. b) Gráfico da função objetivo para a banda de 0-10Hz.

As feições da falha estão melhor definidas, camadas sedimentares mais rasas e a camada de sal estão mais aparentes. As dobras do modelo também ficam mais definidas e é possível verificar uma mudança na velocidade no topo da dobra, como esperado para o reservatório carbonático. A função objetivo atingiu 4 iterações, diminuindo para aproximadamente $70,97\%$. Analisando o gráfico da função objetivo, é possível verificar que esta encontrou-se em um platô e o método não conseguiu minimizar seu valor, mesmo obtendo 22 avaliações da função objetivo. Passando para a próxima banda, a figura 30 ilustra o resultado da inversão para a banda de 0-15Hz.



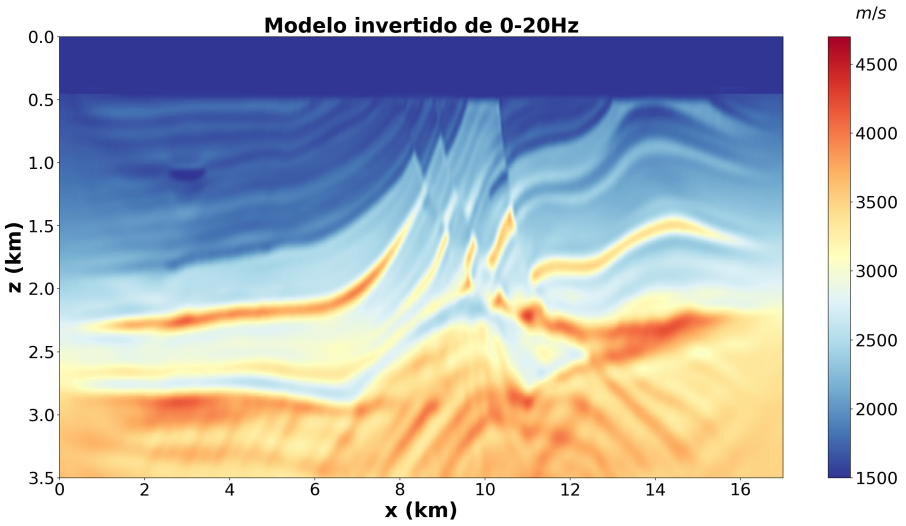
(a)



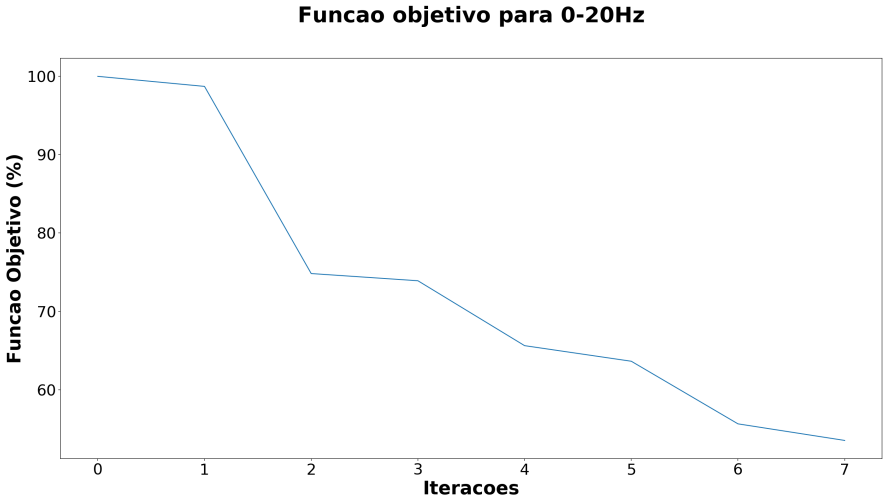
(b)

Figura 30 – a) Modelo de velocidade obtido da inversão para a banda de 0-15Hz. b) Gráfico da função objetivo para a banda de 0-15Hz.

Por conta do aumento da frequência, as camadas finas ficam cada vez mais definidas e as falhas do modelo delimitadas. Pode-se notar também uma considerável melhora na definição do modelo, obtendo um modelo muito mais próximo do modelo real de Marmousi. Analisando a função objetivo, pode-se observar que esta decaiu consideravelmente, atingindo um mínimo de $48,41\%$ do valor original. Nesta banda de frequência, o critério de parada foi acionado ao atingir 7 iterações, sendo necessárias 18 avaliações da função objetivo. Por fim, a figura 31 mostra o resultado utilizando a banda 0-20Hz.



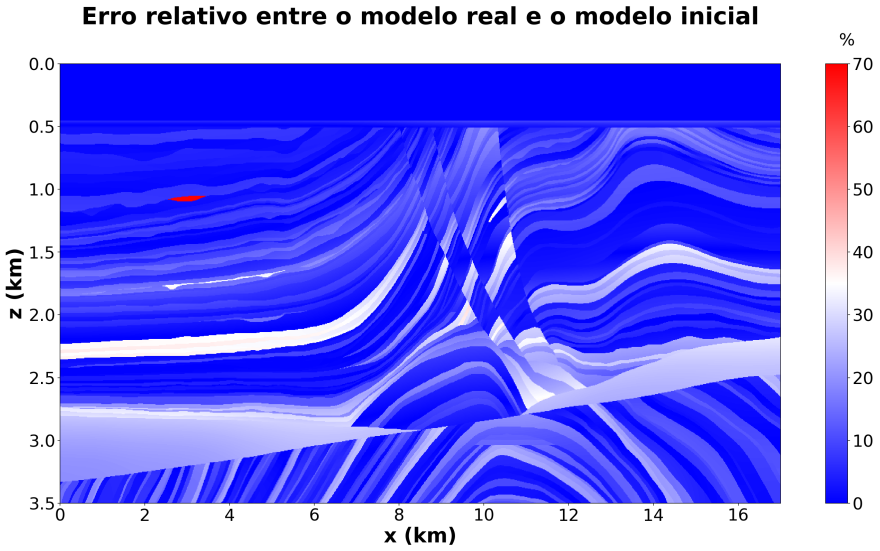
(a)



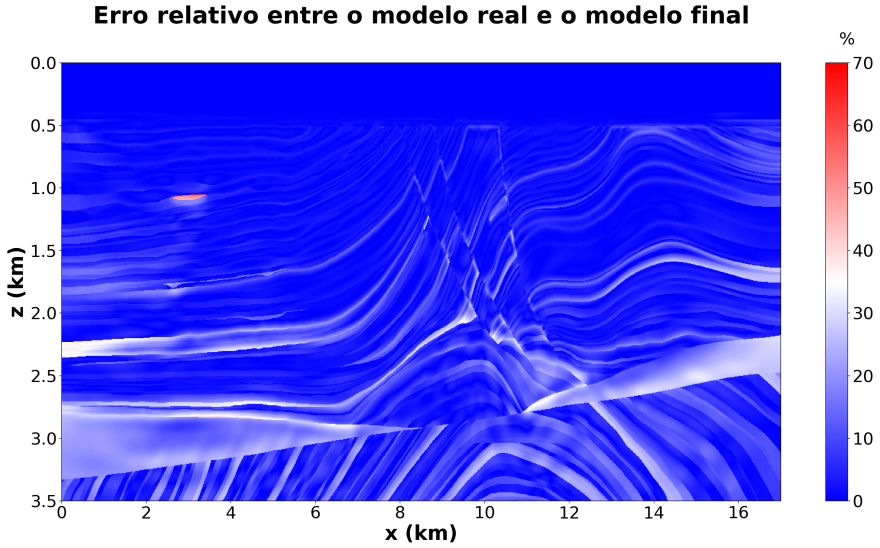
(b)

Figura 31 – a) Modelo de velocidade obtido da inversão para a banda de 0-20Hz. b) Gráfico da função objetivo para a banda de 0-20Hz.

O modelo final obtido representa melhor a geologia que o modelo inicial, realçando camadas, falhas, dobras e os reservatórios rasos e profundos. Avaliando a função objetivo, esta decaiu lentamente até o patamar de $53,52\%$ do valor original. Para quantificar a diferença, a figura 32.a ilustra a diferença percentual relativa entre o modelo real e o modelo inicial e a figura 32.b a diferença relativa entre o modelo real e o modelo final obtido da inversão.



(a)



(b)

Figura 32 – a) Matriz de diferença relativa obtida comparando o modelo inicial com o real. b) Matriz de diferença relativa obtida comparando o modelo obtido da inversão com o real.

Pode-se notar uma diminuição da diferença comparado com a figura 32.b, inclusive melhorando os valores de velocidade no canal, onde a diferença chegava a 70% (figura 32.a), passando para valores de aproximadamente 50%. Uma forma de analisar a qualidade da inversão é comparando a migração RTM utilizando o modelo inicial e utilizando o modelo obtido da inversão FWI. Devido ao fato de em condições reais de imageamento não se saber o modelo real, faz-se a diferença entre o modelo final e o inicial, indicado pela figura 33.

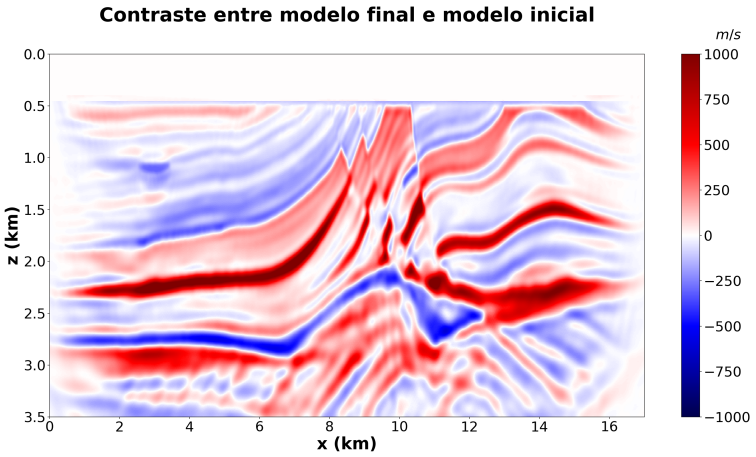


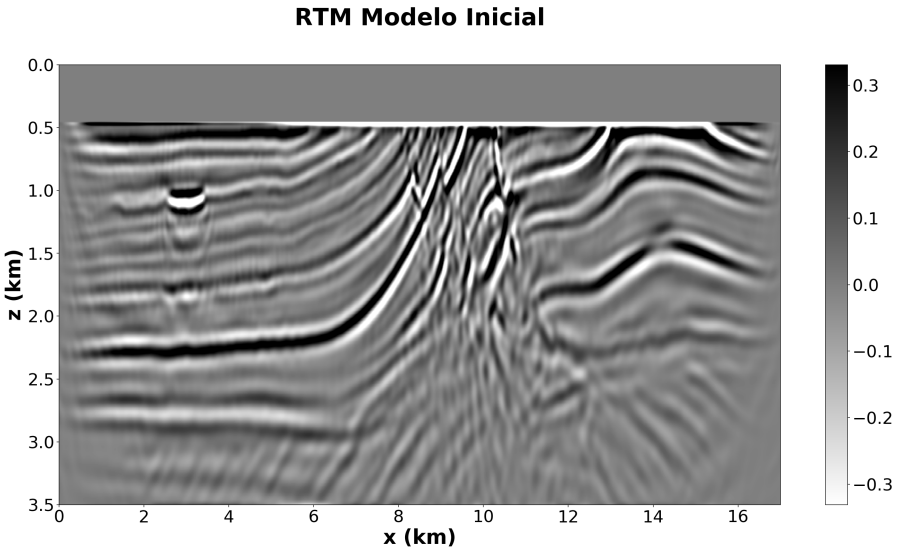
Figura 33 – Ilustração da diferença entre o modelo final obtido da FWI e o modelo utilizado como entrada para a inversão.

Analisando a figura 33, pode-se verificar que o modelo mudou significativamente, sendo modificado positivamente (aumento nos valores da velocidade) ou negativamente (diminuição dos valores de velocidade) em diversas camadas do modelo. Nota-se que há uma diminuição da velocidade nas camadas mais rasas e em algumas sequências sedimentares. Há um aumento das velocidades em algumas regiões do conjunto de falhas, nas regiões do sal (laterais do modelo) e na região das dobras. Por fim, a figura 34 ilustra a migração utilizando o modelo inicial e a migração utilizando o modelo obtido da inversão.

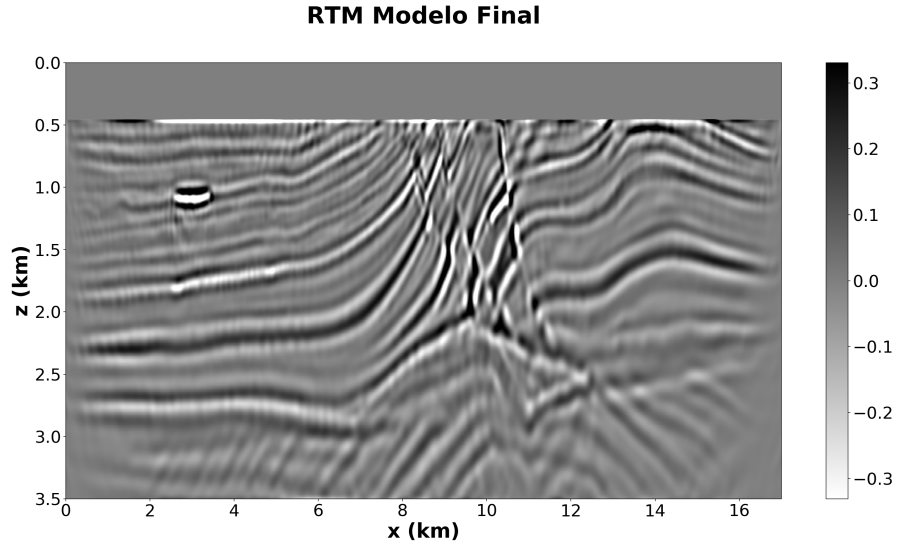
Analisando ambas migrações, pode-se notar uma melhora considerável na seção migrada utilizando o modelo advindo da inversão. Como o modelo representou melhor o modelo de velocidade real, este conseguiu imagear os detalhes das camadas, especialmente das regiões mais profundas. Pode-se notar que a migração utilizando o modelo inicial não consegue colapsar a energia na região das dobras, mantendo uma região caótica na região do reservatório carbonático do modelo de Marmousi. Uma característica importante a ser ressaltada é a robustez do algoritmo de migração RTM, imageando as estruturas do modelo mesmo com um modelo inicial com baixo detalhe da geologia de subsuperfície. Para o tempo e o número de avaliações da função objetivo para cada estágio da inversão, a tabela 22 mostra seus respectivos valores para cada banda de frequência da inversão FWI.

Tabela 22 – Números de avaliação da função objetivo e tempo de execução medido para cada banda da inversão FWI.

	Avaliações	Tempo
0-5Hz	22	6h 14m 10,98s
0-10Hz	22	6h 18m 51,08s
0-15Hz	18	5h 5m 9,60s
0-20Hz	18	5h 5m 40,30s
Total	80	22h 43m 51,96s



(a)



(b)

Figura 34 – a) Seção migrada utilizando o modelo inicial dado como entrada da inversão FWI. b) Seção migrada utilizando o modelo final obtido pela inversão.

Analisando a tabela 22, pode-se notar que as duas primeiras faixas de frequência atingiram o critério de parada de 22 iterações, pode-se notar também que ambas possuem tempo de execução bem similar, se diferenciando em apenas 4 minutos. Já as bandas seguintes atingiram o critério de 7 iterações, ambas necessitaram do mesmo número de avaliações e tiveram tempo quase idêntico, com alguns segundos de diferença. Ao se fazer o acumulado, a inversão necessitou de 80 avaliações da função objetivo e levou cerca de aproximadamente 22 horas e 44 minutos para ficar pronta. Este resultado pode ser comparado ao utilizar os valores estimados da tabela 23 e multiplicá-los pelo número de avaliações da função objetivo.

Tabela 23 – Tempos de execução estimado de cada configuração de paralelização para a inversão FWI utilizando 80 avaliações da função objetivo.

	Host	Multicore	GPU
Tempo estimado	36d 09h 31m 31,20s	8d 05h 24m 0.0s	1d 7h 30m 33,60s

Comparando o valor estimado e calculado da GPU, pode-se notar uma diferença de aproximadamente 9 horas entre ambas. Isto é um bom resultado, uma vez que os valores de tempo de execução não necessariamente são lineares de acordo com o número de avaliações da função objetivo. Outro fator é o tempo estimado para a inversão utilizando *Host*, onde seriam estimados 36 dias e 9 horas para completar a inversão, ressaltando o problema da programação serial quanto ao tempo de máquina necessário para obter o resultado da inversão. Já para o *Multicore*, apesar de se estimar um tempo relativo de aproximadamente 8 dias e levando em conta uma margem de erro de algumas horas, ainda é um tempo inferior ao apresentado pela GPU, demonstrando a eficiência e a economia no tempo computacional que o emprego da GPU pode proporcionar.

Modelo Búzios

Passando para os resultados da inversão FWI, necessita-se de um modelo inicial como entrada para o algoritmo. Seguindo a metodologia descrita anteriormente, o modelo utilizado é ilustrado na figura 35.

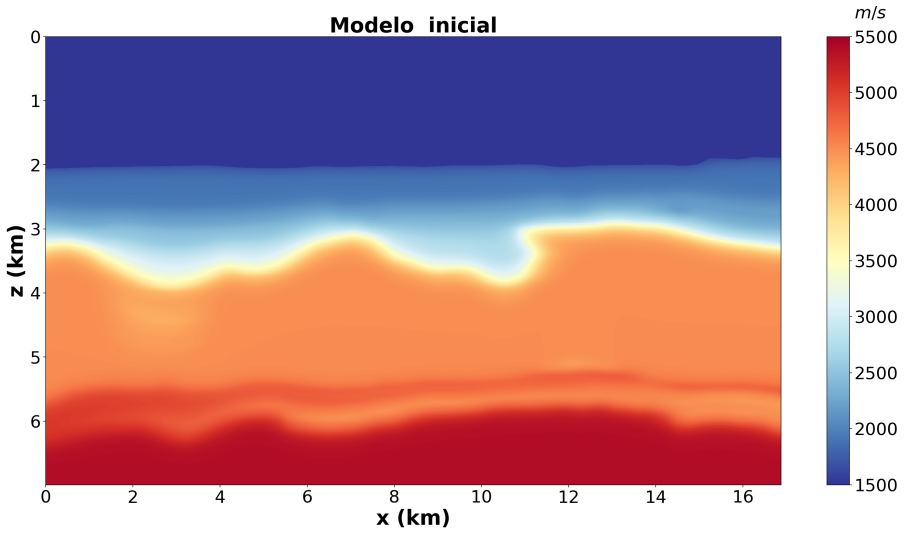
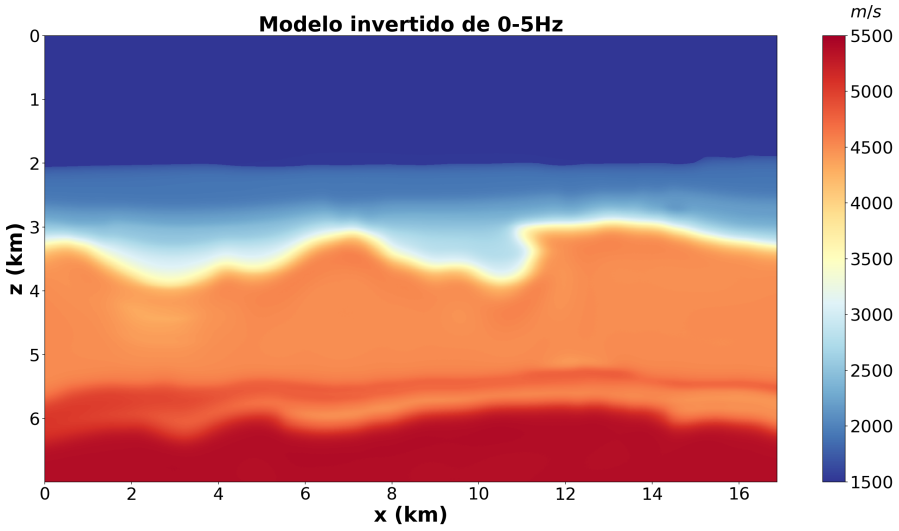


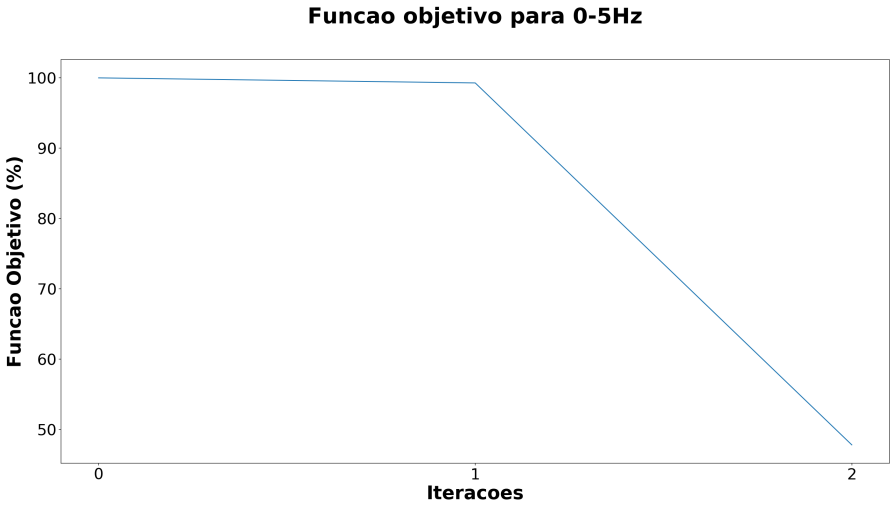
Figura 35 – Modelo inicial utilizado como entrada para a inversão FWI.

Analisando a figura 35, pode-se notar que o modelo possui as macro estruturas da geologia de Búzios. Diferentemente do modelo de Marmousi, este modelo possui características mais próximas com relação ao modelo original. A partir do modelo inicial,

é iniciada a inversão FWI para obter modelos com mais detalhes. A figura 36 mostra o modelo obtido para a banda de 0-5Hz e sua função objetivo.



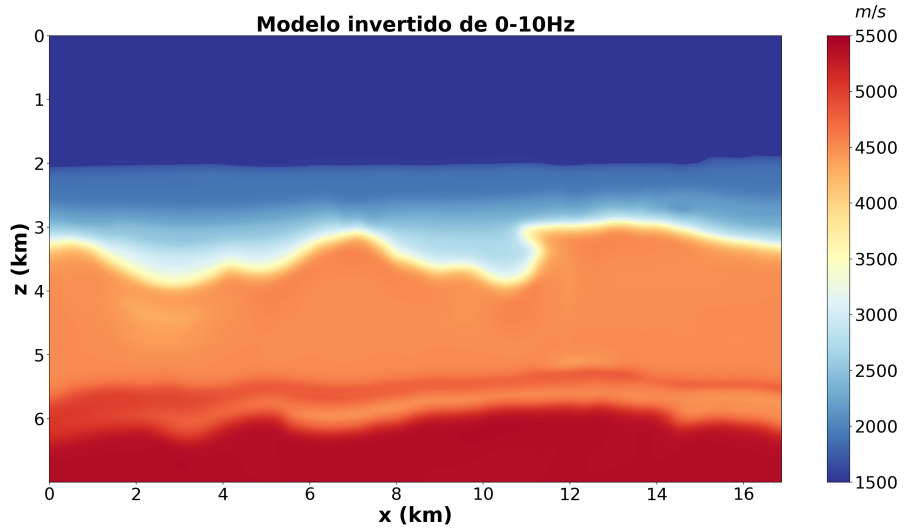
(a)



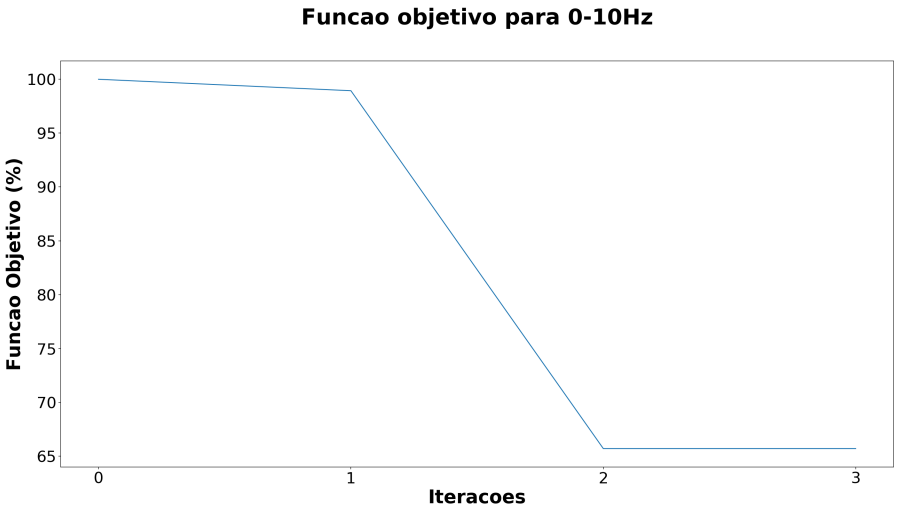
(b)

Figura 36 – a) Modelo de velocidade obtido da inversão para a banda de 0-5Hz. b) Gráfico da função objetivo para a banda de 0-5Hz.

Observando a figura, não é possível notar uma diferença significativa no modelo de velocidade, isto ocorre pois a função objetivo foi minimizada apenas 2 vezes, provocando pouca alteração no resultado da inversão para esta banda. A complexidade do modelo também oferece dificuldade maior para o algoritmo, que busca encontrar um modelo de velocidade que seja mais coerente mas acaba por falhando na etapa de minimização. Passando para a banda de 0-10Hz, o resultado é dado pela figura 37.



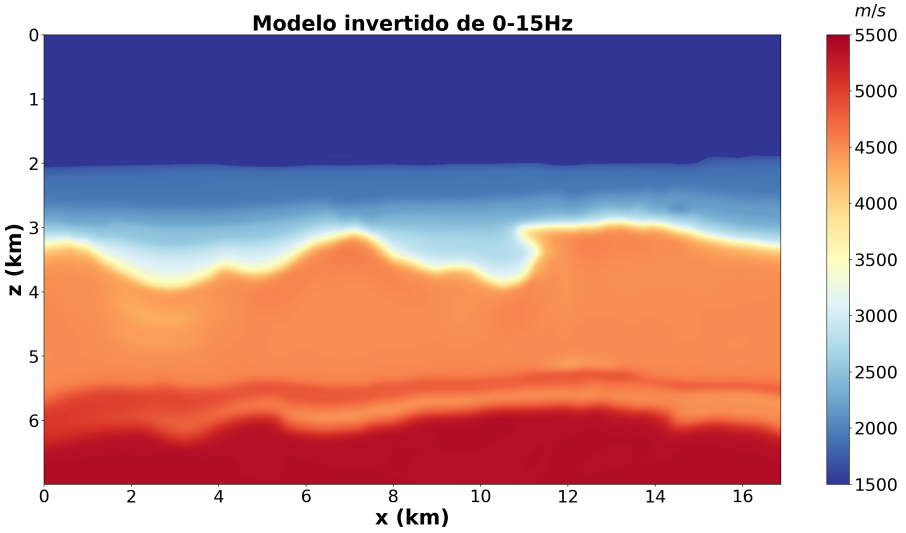
(a)



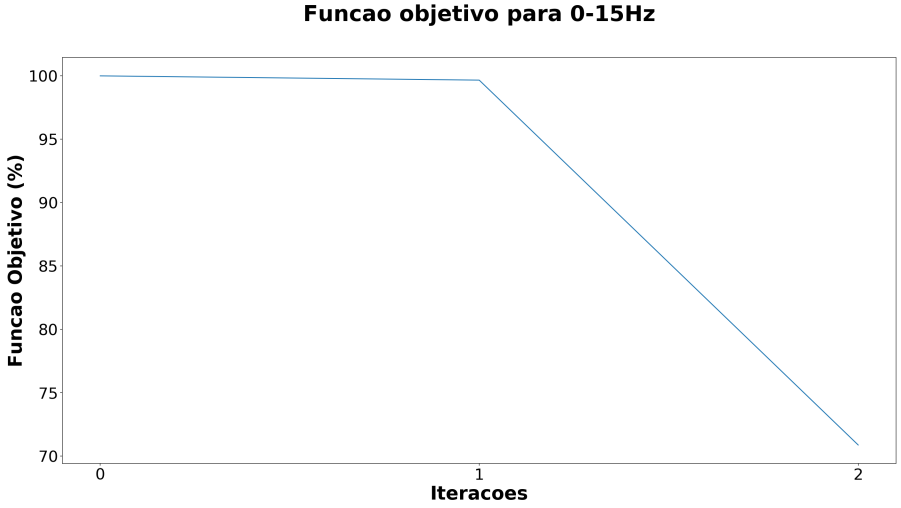
(b)

Figura 37 – a) Modelo de velocidade obtido da inversão para a banda de 0-10Hz. b) Gráfico da função objetivo para a banda de 0-10Hz.

Algumas características são mais evidentes, como as falhas no topo do sal e o reservatório é pouco mais visível. Ao verificar a função objetivo, pode-se observar que a função atingiu um platô em 3 iterações, sendo o motivo de haver pouca alteração no modelo de velocidade. A figura 38 ilustra o resultado para a banda de 0-15Hz.



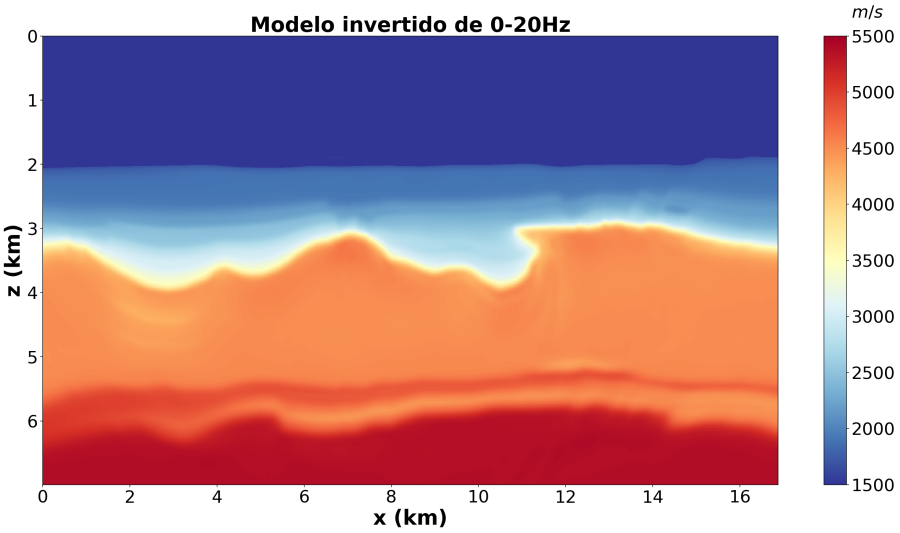
(a)



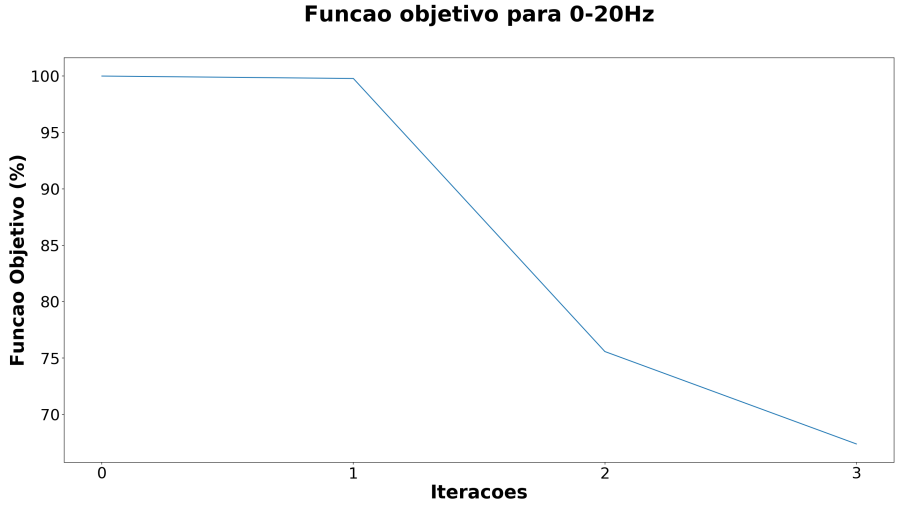
(b)

Figura 38 – a) Modelo de velocidade obtido da inversão para a banda de 0-15Hz. b) Gráfico da função objetivo para a banda de 0-15Hz.

A partir desta banda, é possível ter um maior detalhe das falhas do pós-sal, do contraste da região do *overhang* com o sal e uma definição maior na região do sal estratificado. A função obteve duas iterações, mesmo utilizando o critério de 22 avaliações da função objetivo. Por fim, é feita a inversão para a banda de 0-20Hz, dada pela figura 39.



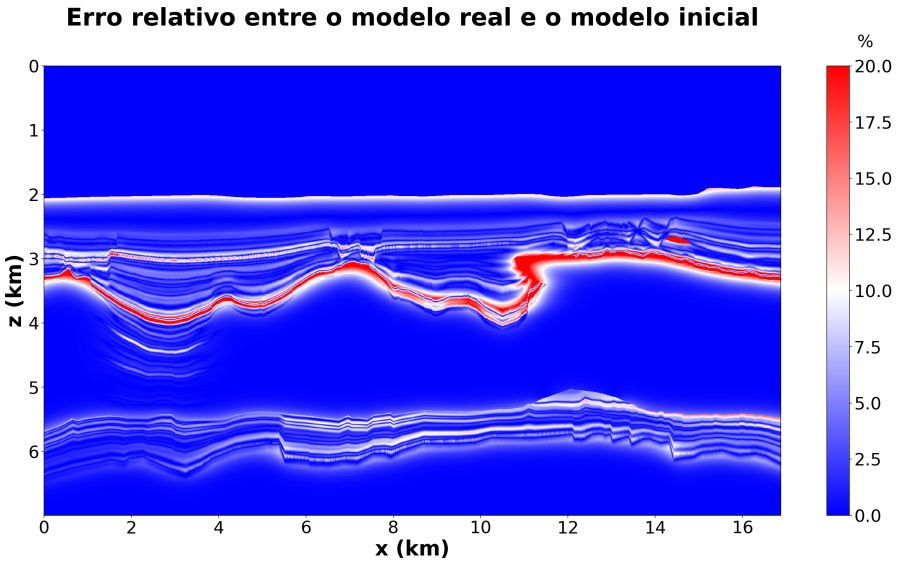
(a)



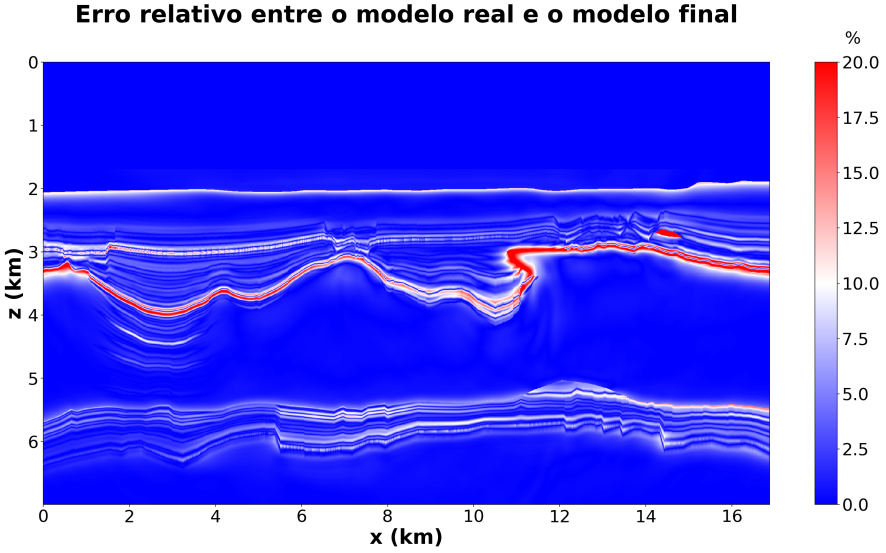
(b)

Figura 39 – a) Modelo de velocidade obtido da inversão para a banda de 0-20Hz. b) Gráfico da função objetivo para a banda de 0-20Hz.

Esta foi a banda com maior impacto no modelo de velocidade, aprimorando a região do reservatório do pós-sal e definindo melhor a região do *overhang*. A função objetivo foi minimizada 3 vezes, atingindo um valor de aproximadamente $67,38\%$ do valor original. Um possível motivo para a baixa alteração e diminuição no valor da função objetivo é a presença de um mínimo local. Uma vez que o modelo possui altas complexidades geológicas, o algoritmo acabou não minimizando de forma satisfatória, diferentemente da inversão no modelo de Marmousi. A figura 40 ilustra a matriz de diferença do modelo invertido em relação ao modelo real.



(a)



(b)

Figura 40 – a) Matriz de diferença relativa entre o modelo real e o modelo inicial. b) Matriz de diferença relativa entre o modelo real e o modelo obtido da inversão.

Pode-se notar que modelo possui diferença relativa média em torno de 5% a 10%, mudando significativamente apenas na região do reservatório pós-sal e no topo do sal. Um problema encontrado também é que o modelo possui diferença relativa baixa com relação ao modelo original. Isto acarretou em baixas atualizações do modelo, especialmente para as baixas frequências, uma vez que as macro estruturas já estavam presentes no modelo inicial e novas atualizações se tornariam mais difíceis. Para analisar a progressão do modelo de velocidade, é feita a diferença entre o modelo obtido da inversão pelo modelo inicial (figura 41).

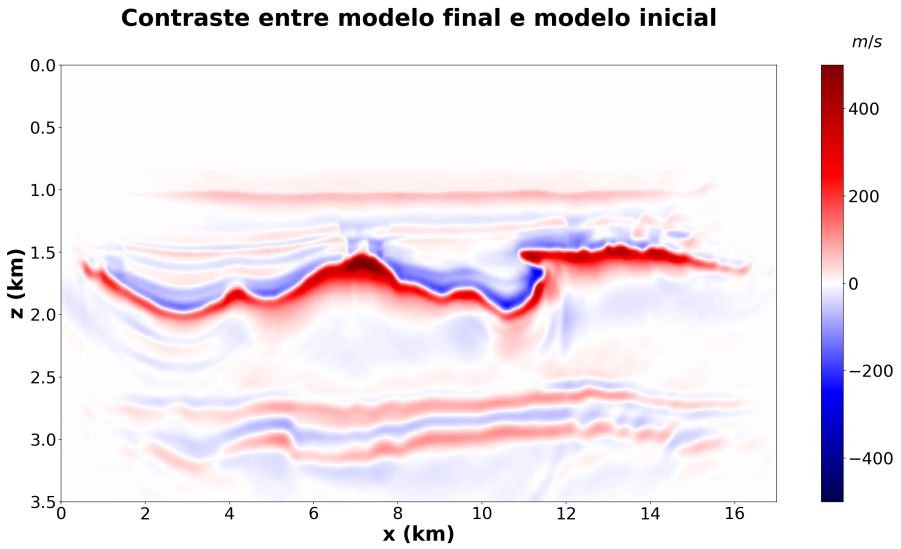
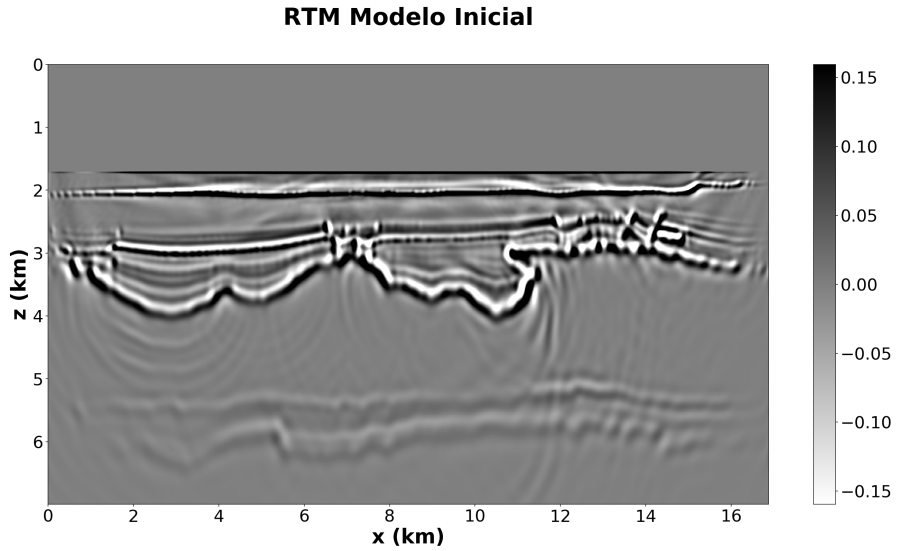
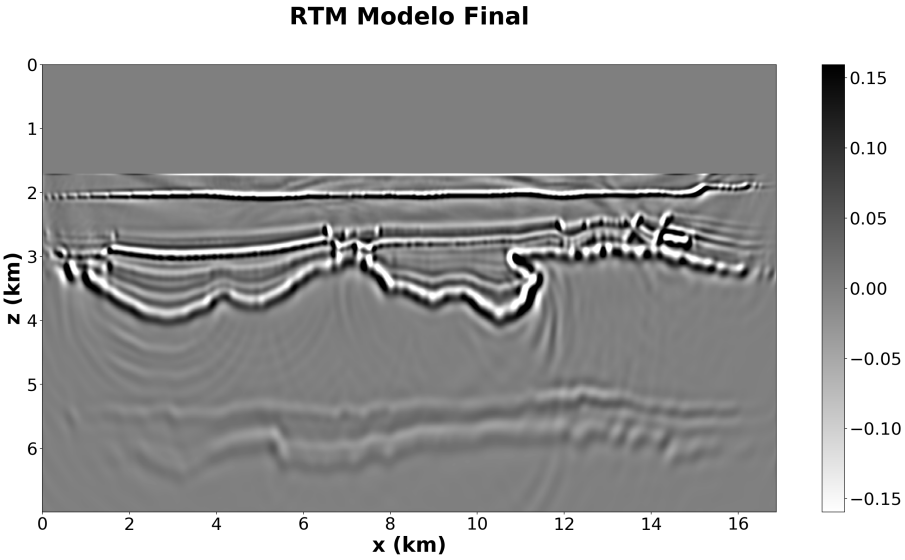


Figura 41 – Diferença entre o modelo final obtido da inversão pelo modelo inicial de Búzios.

Observando a figura, pode-se observar que não há uma mudança tão drástica quanto no modelo de Marmousi, mas há realces nas feições do modelo de velocidade. As falhas do pós-sal são modificadas, tendo aumento de velocidade nas camadas mais contínuas e diminuição na região da falha. Há também um aumento drástico da velocidade no topo do sal. Um fato curioso é a baixa modificação da velocidade na região do sal, modificando baixos valores apenas para a região do sal estratificado. A região do reservatório sofre diminuição leve nos valores de velocidade e há um padrão intercalado de aumento e diminuição na região do pré-sal. Outro fato interessante é falta de variação na região do embasamento econômico. Por fim, compara-se a imagem migrada utilizando o modelo inicial e o modelo final (figura 42).



(a)



(b)

Figura 42 – a) Seção migrada utilizando o modelo inicial dado como entrada da inversão FWI. b) Seção migrada utilizando o modelo final obtido pela inversão.

Comparando o resultado das duas migrações, não se é possível notar uma grande diferença. Isto se dá pois o algoritmo obteve dificuldade em atualizar o modelo de velocidade, causando pouco impacto na imagem final migrada. A maior diferença notada é uma melhor delimitação do topo do sal e uma diminuição nos artefatos de migração, realçando melhor as amplitudes das reflexões. A tabela 24 exibe os valores de avaliações da função objetivo e o tempo de execução em cada banda.

Tabela 24 – Números de avaliação da função objetivo e tempo de execução medido para cada banda da inversão FWI.

	Avaliações	Tempo
0-5Hz	22	9h 36m 46,31s
0-10Hz	22	9h 36m 14,24s
0-15Hz	22	9h 34m 54,90s
0-20Hz	22	9h 28m 40,50s
Total	88	1d 14h 16m 35,95s

Pode-se observar que o critério de parada em todas as bandas de frequência foi de 22 iterações. Os tempos de execução são bem similares, oscilando poucos minutos entre estes. A inversão de todas as bandas necessitou de 1 dia, 14 horas e 16 minutos aproximadamente. Para comparar o valor medido da inversão com o tempo estimado, se é multiplicado o total de avaliações da função objetivo pelo tempo estimado da FWI dado pela tabela 19, obtendo a tabela 25.

Tabela 25 – Tempos de execução estimado de cada configuração de paralelização para a inversão FWI utilizando 88 avaliações da função objetivo.

	Host	Multicore	GPU
Tempo estimado	72d 21h 39m 10,48s	17d 9h 44m 53,60s	2d 12h 21m 34s

Observando a tabela, pode-se ver que o tempo estimado pela GPU é quase duas vezes maior que o tempo medido, isto mostra que a estimativa feita para GPUs não deve ser feita de maneira linear, como é feita para CPUs, uma vez que a GPU não reage linearmente de acordo com o tamanho do problema. Outro fator importante é que, considerando o valor estimado para a arquitetura em CPU, a GPU oferece um ganho de 70 dias de máquina para o código serial e 15 dias para o código *Multicore*. Este resultado mostra que a computação em placas gráficas consegue diminuir consideravelmente o tempo de execução, gerando o resultados preciso em menor tempo.

8 Conclusão

Os valores dados pelas tabelas mostram que o tempo de execução cresceu proporcionalmente ao acesso à memória em cada ordem de diferença finita, exceto para a GPU, que manteve um valor constante. Isto ocorre tanto para o modelo de Marmousi, quanto para o modelo de Búzios. Pode-se notar que devido ao alto paralelismo da GPU, é possível usufruir de altas ordens no MDF sem comprometer a performance do código para o caso acústico. Na geração do dado sintético observado para o caso acústico, obteve-se um tempo de execução medido menor do que o estimado. Isto ocorre em ambos os modelos, sendo uma particularidade do caso acústico. Os dados sintéticos elásticos obtiveram resultados diferentes, estando com valores medidos próximos aos estimados. Ao analisar os resultados estimados utilizando o *Host*, *Multicore* e o valor medido da GPU, pode-se dizer que a GPU conseguiu otimizar significativamente o tempo de execução em cada algoritmo, diminuindo o problema de custo computacional atrelado à inversão FWI. Pode-se concluir que paralelizar códigos utilizando OpenACC é uma abordagem prática e efetiva na diminuição do tempo computacional. Devido à baixa complexidade quanto às diretivas e uma maior automatização da paralelização, os resultados obtidos foram otimistas quanto ao uso do OpenACC em problemas científicos, como para a inversão FWI.

Quanto aos resultados da inversão FWI no modelo de Marmousi, pode-se dizer que a inversão FWI adquiriu um modelo de alta resolução. No geral, as bandas de frequência chegaram a um valor aceitável de iterações, diminuindo consideravelmente a função objetivo e recuperando um modelo de velocidade mais próximo do modelo real. Os resultados da migração RTM mostram melhora significativa no imageamento das estruturas, especialmente estruturas associadas ao sal nas bordas do modelo e nas dobras das regiões mais profundas do modelo. A inversão para o modelo de Marmousi necessitou de menos de um dia para ser realizada na GPU, valor inferior ao de 36 dias estimados utilizando programação serial e de 8 dias estimados utilizando *Multicore*. No entanto, não se pode obter a mesma conclusão para o modelo de Búzios, onde somente se conseguiu atualizar o modelo três vezes. Uma possível causa é a complexidade do modelo, uma vez que este possui feições geológicas desafiadoras na indústria petrolífera. Apesar de não haver mudanças bruscas no modelo de velocidade, a inversão FWI conseguiu recuperar feições no sal e no reservatório do pós-sal, melhorando a imagem migrada final.

São sugeridos trabalhos futuros para lidar com os problemas encontrados neste modelo. Dentre estes trabalhos, pode-se utilizar um fluxo de inversão FWI utilizado para regiões do sal, uma vez que o fluxo utilizado neste trabalho é um fluxo proposto por trabalhos clássicos. Outra sugestão seria a integração da tomografia de reflexão com a FWI. Desta forma, os modelos iniciais podem ser obtidos a partir da inversão tomográfica

e diminuir o problema atrelado à força da suavização do modelo. Deseja-se também paralelizar os códigos de modelagem e inversão utilizando CUDA, uma vez que esta oferece mais liberdade ao programador quanto à paralelização e pode fornecer códigos ainda mais otimizados.

Referências

ALKHALIFAH, T. An acoustic wave equation for anisotropic media. *GEOPHYSICS*, v. 65, n. 4, p. 1239–1250, 2000. Disponível em: <https://doi.org/10.1190/1.1444815>. Citado na página 70.

ANP. *Boletim da Produção de Petróleo e Gás Natural*. [S.l.], 2019. Citado na página 63.

BARLAS, G. *Multicore and GPU Programming: An Integrated Approach*. [S.l.: s.n.], 2014. ISBN 978-0124171374. Citado 5 vezes nas páginas 4, 17, 49, 55 e 56.

BEDNAR, J. A brief history of seismic migration. *Geophysics*, v. 70, 05 2005. Citado na página 112.

BERENGER, J.-P. A perfectly matched layer for the absorption of electromagnetic waves. *Journal of Computational Physics*, v. 114, n. 2, p. 185 – 200, 1994. ISSN 0021-9991. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0021999184711594>. Citado na página 27.

BISHOP, T. N. et al. Tomographic determination of velocity and depth in laterally varying media. *Geophysics*, v. 50, n. 6, p. 903–923, 1985. Citado na página 17.

BLEISTEIN, N.; GRAY, S. From the hagedoom imaging technique to kirchhoff migration and inversion. *Geophysical Prospecting*, v. 49, p. 629 – 643, 07 2008. Citado na página 112.

BORDING, R. P. Finite difference modeling - nearly optimal sponge boundary conditions. In: _____. *SEG Technical Program Expanded Abstracts 2004*. [s.n.], 2005. p. 1921–1924. Disponível em: <https://library.seg.org/doi/abs/10.1190/1.1845189>. Citado na página 27.

BORN, M. Quantenmechanik der stoßvorgänge. *Zeitschrift für Physik*, v. 38, n. 11, p. 803–827, Nov 1926. ISSN 0044-3328. Disponível em: <https://doi.org/10.1007/BF01397184>. Citado 2 vezes nas páginas 35 e 36.

BRANDSBERG-DAHL, S. High-performance computing for seismic imaging; from shoestrings to the cloud. *SEG Technical Program Expanded Abstracts 2017*, p. 5273–5277, 2017. Citado 3 vezes nas páginas 3, 17 e 49.

BUNKS, C. et al. Multiscale seismic wave-form inversion. *Geophysics*, v. 60, p. 1457–1473, 09 1995. Citado 2 vezes nas páginas 17 e 39.

CARCIONE, J. M.; HERMAN, G. C.; KROODE, a. P. E. ten. Review Article: Seismic modeling. *Review Literature And Arts Of The Americas*, v. 67, n. 4, p. 1304 – 1325, 2002. ISSN 00168033. Citado na página 27.

CARNEIRO, M. et al. On the scaling of the update direction for multi-parameter full waveform inversion: Applications to 2d acoustic and elastic cases. *Pure and Applied Geophysics*, v. 175, p. 1–25, 01 2018. Citado na página 39.

CARROLL, R. The determination of the acoustic parameters of volcanic rocks from compressional velocity measurements. *International Journal of Rock Mechanics and Mining Sciences Geomechanics Abstracts*, v. 6, n. 6, p. 557 – 579, 1969. ISSN 0148-9062. Disponível em: <http://www.sciencedirect.com/science/article/pii/0148906269900229>. Citado na página 67.

CERJAN, C. et al. A nonreflecting boundary condition for discrete acoustic and elastic wave equations. *GEOPHYSICS*, v. 50, n. 4, p. 705–708, 1985. Disponível em: <https://doi.org/10.1190/1.1441945>. Citado 2 vezes nas páginas 26 e 27.

CHATTOPADHYAY, S.; MCMECHAN, G. Imaging conditions for prestack reverse-time migration. *Geophysics*, v. 73, 05 2008. Citado na página 113.

CHENG, J.; GROSSMAN, M.; MCKERCHER, T. *Professional CUDA C Programming*. 1st. ed. GBR: Wrox Press Ltd., 2014. ISBN 1118739329. Citado 4 vezes nas páginas 3, 4, 50 e 51.

CLAERBOUT, J. F.; DOHERTY, S. M. Downward continuation of moveout-corrected seismograms. *GEOPHYSICS*, v. 37, n. 5, p. 741–768, 1972. Disponível em: <https://doi.org/10.1190/1.1440298>. Citado na página 112.

COLLINO, F.; TSOGKA, C. Application of the perfectly matched absorbing layer model to the linear elastodynamic problem in anisotropic heterogeneous media. *Geophysics*, v. 66, p. 294–307, 01 2001. Citado na página 28.

DABLAIN, M. A. The application of high-order differencing to the scalar wave equation. *GEOPHYSICS*, v. 51, n. 1, p. 54–66, 1986. Disponível em: <https://doi.org/10.1190/1.1442040>. Citado na página 108.

FARBER, R. *Parallel Programming with OpenACC*. 1st. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2016. ISBN 0124103979. Citado na página 57.

FARIA, E. L. *Migração antes do empilhamento utilizando propagação reversa no tempo.* Dissertação (Mestrado) — Universidade Federal da Bahia, Salvador, Bahia, Brasil, 1986. Citado na página 70.

FICHTNER, A. *Full Seismic Waveform Modelling and Inversion*. [S.l.]: Springer, 2009. 615 p. ISSN 1098-6596. ISBN 0724-3111\r0945-4292. Citado 2 vezes nas páginas 3 e 40.

Flynn, M. J. Some computer organizations and their effectiveness. *IEEE Transactions on Computers*, C-21, n. 9, p. 948–960, 1972. Citado na página 51.

GARDNER, G. H. F.; GARDNER, L. W.; GREGORY, A. R. Formation velocity and density—the diagnostic basics for stratigraphic traps. *GEOPHYSICS*, v. 39, n. 6, p. 770–780, 1974. Disponível em: <https://doi.org/10.1190/1.1440465>. Citado na página 67.

GREENBERG, M. L.; CASTAGNA, J. P. Shear-wave velocity estimation in porous rocks: Theoretical formulation, preliminary verification and applications1. *Geophysical Prospecting*, v. 40, n. 2, p. 195–209, 1992. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-2478.1992.tb00371.x>. Citado na página 67.

GUASCH, L. *3D elastic time-frequency full-waveform inversion*. Tese (Doctor of Philosophy) — Imperial College London, 2011. Citado 2 vezes nas páginas 3 e 27.

GUSTAFSON, J. L. Reevaluating amdahl's law. *Commun. ACM*, Association for Computing Machinery, New York, NY, USA, v. 31, n. 5, p. 532–533, maio 1988. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/42411.42415>. Citado na página 55.

HAMARBITAN, N. S.; MARGRAVE, G. F. Spectral analysis of a ghost. *GEOPHYSICS*, v. 66, n. 4, p. 1267–1273, 2001. Citado na página 41.

HAN, D.; NUR, A.; MORGAN, D. Effects of porosity and clay content on wave velocities in sandstones. *GEOPHYSICS*, v. 51, n. 11, p. 2093–2107, 1986. Disponível em: <https://doi.org/10.1190/1.1442062>. Citado na página 66.

KALITA, M. et al. Regularized full-waveform inversion with automated salt flooding. *Geophysics*, v. 84, n. 4, p. R569–R582, 2019. ISSN 0016-8033. Citado na página 16.

KARSOU, A. et al. Construction of a velocity model of the brazilian pre salt based on bios field - preliminary results. In: . [S.l.]: The 16th International Congress of the Brazilian Geophysical Society and EXPOGEf, 2019. p. 1–5. Citado 3 vezes nas páginas 18, 60 e 63.

KIRK, D. B.; HWU, W.-m. W. *Programming Massively Parallel Processors: A Hands-on Approach*. 1st. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2010. ISBN 0123814723. Citado 4 vezes nas páginas 4, 52, 53 e 54.

KOMATITSCH, D.; MARTIN, R. An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation. *Geophysics*, v. 72, 09 2007. Citado na página 27.

LAILLY, P. et al. The seismic inverse problem as a sequence of before stack migrations, in Conference on Inverse Scattering: Theory and Application. 1983. Citado na página 16.

LINES, L. R.; SLAWINSKI, R.; BORDING, R. P. A recipe for stability of finite-difference wave-equation computations. *GEOPHYSICS*, v. 64, n. 3, p. 967–969, 1999. Disponível em: <https://doi.org/10.1190/1.1444605>. Citado na página 108.

LIU, Q.; TAO, J. The perfectly matched layer for acoustic waves in absorptive media. *Journal of The Acoustical Society of America - J ACOUST SOC AMER*, v. 102, p. 2072–2082, 10 1997. Citado 2 vezes nas páginas 3 e 30.

LIU, Y.; SEN, M. K. Time–space domain dispersion-relation-based finite-difference method with arbitrary even-order accuracy for the 2d acoustic wave equation. *Journal of Computational Physics*, v. 232, n. 1, p. 327 – 345, 2013. ISSN 0021-9991. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0021999112004718>. Citado na página 107.

MADARIAGA, R. Dynamics of an expanding circular fault. *Bulletin of the Seismological Society of America*, v. 66, p. 639–666, 06 1976. Citado na página 109.

MARTIN, G. S.; WILEY, R.; MARFURT, K. J. Marmousi2: An elastic upgrade for marmousi. *The Leading Edge*, v. 25, n. 2, p. 156–166, 2006. Disponível em: <https://doi.org/10.1190/1.2172306>. Citado na página 60.

MéTIVIER, L. et al. Applying gauss-newton and exact newton method to full waveform inversion. *European Association of Geoscientists and Engineers*, p. 1–5, 06 2012. Citado na página 36.

MCGARRY, R.; MOGHADDAM, P. Npml boundary conditions for second-order wave equations. In: _____. *SEG Technical Program Expanded Abstracts 2009*. [s.n.], 2009. p. 3590–3594. Disponível em: <https://library.seg.org/doi/abs/10.1190/1.3255611>. Citado na página 27.

METIN, G. Application of nearly perfectly matched layer with second-order acoustic equations in seismic numerical modeling. *Journal of Geology Geosciences*, v. 02, 01 2013. Citado na página 27.

MISIC, M.; DURDEVIC, D.; TOMASEVIC, M. Evolution and trends in gpu computing. In: . [S.l.: s.n.], 2012. p. 289–294. ISBN 978-1-4673-2577-6. Citado na página 17.

NOCEDAL, J.; WRIGHT, S. *Numerical optimization, series in operations research and financial engineering*. [s.n.], 2006. ISSN 10969101. ISBN 0387987932. Disponível em: <http://www.lavoisier.fr/livre/notice.asp?id=O2SWL2AO2O2OWB>. Citado 3 vezes nas páginas 45, 46 e 48.

NVIDIA; VINGELMANN, P.; FITZEK, F. H. *CUDA, release: 10.2.89*. 2020. Disponível em: <https://developer.nvidia.com/cuda-toolkit>. Citado na página 56.

PASALIC, D.; MCGARRY, R. Convolutional perfectly matched layer for isotropic and anisotropic acoustic wave equations. In: _____. *SEG Technical Program Expanded Abstracts 2010*. [s.n.], 2010. p. 2925–2929. Disponível em: <https://library.seg.org/doi/abs/10.1190/1.3513453>. Citado na página 27.

PETROVA, S. S.; SOLOV'EV, A. D. The origin of the method of steepest descent. *Historia Mathematica*, v. 24, n. 4, p. 361 – 375, 1997. ISSN 0315-0860. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0315086096921461>. Citado na página 45.

PICA, A.; DIET, J. P.; TARANTOLA, A. Nonlinear inversion of seismic reflection data in a laterally invariant medium. *Geophysics*, v. 55, n. 3, p. 284–292, 1990. ISSN 00168033. Citado 2 vezes nas páginas 24 e 39.

POTTER, C.; STEWART, R. Density predictions using v p and v s sonic logs. *CREWES Research Report*, 01 1998. Citado na página 67.

PRATT, R. G.; GOULTY, N. R. Combining wave-equation imaging with traveltime tomography to form high-resolution images from crosshole data. *Geophysics*, v. 56, n. 2, p. 208–224, 1991. Citado na página 17.

RICKER, N. Wavelet contraction, wavelet expansion, and the control of the seismic resolution. *Geophysics*, v. 18, n. 4, p. 769–792, 1953. Disponível em: <https://doi.org/10.1190/1.1437927>. Citado na página 24.

ROBERTSSON, J. Viscoelastic finite-difference modeling. *Geophysics*, v. 59, 01 1994. Citado na página 70.

RODEN, A.; GEDNEY, S. Convolutional pml (cpml): an efficient fdtd implementation of the cfs-pml for arbitrary media. *Microwave and Optical Technology Letters*, v. 27, p. 334–339, 12 2000. Citado 2 vezes nas páginas 27 e 28.

SHERIFF, R. E.; GELDART, L. P. *Exploration Seismology*. [S.l.]: Cambridge University Press, 1995. Citado 2 vezes nas páginas 16 e 41.

SHEWCHUK, J. R. *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. USA, 1994. Citado na página 45.

SIRGUE, L.; ETGEN, J.; ALBERTIN, U. 3D full waveform inversion: Wide versus narrow azimuth acquisitions. *Society of Exploration Geophysicists - 77th SEG International Exposition and Annual Meeting, SEG 2007*, p. 1760–1764, 2007. Citado na página 17.

SOTAK, G.; BOYER, K. The laplacian-of-gaussian kernel: A formal analysis and design procedure for fast, accurate convolution and full-frame output. *Computer Vision, Graphics, and Image Processing*, v. 48, n. 2, p. 147 – 189, 1989. ISSN 0734-189X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0734189X89800362>. Citado na página 71.

TARANTOLA, A. Inversion of seismic reflection data in the acoustic approximation. *Geophysics*, v. 49, n. 8, p. 1259–1266, 1984. Citado na página 16.

TARANTOLA, A. *Inverse Problem Theory*. [s.n.], 2006. v. 120. 1–358 p. ISSN 0001-4966. ISBN 0898715725. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/17069280>. Citado na página 20.

TEIXEIRA, L. et al. Rock physics and seismic inversion to identify stratification within salt section supporting velocity, facies modeling and geomechanical analysis. In: . [S.l.: s.n.], 2017. Citado 2 vezes nas páginas 66 e 67.

VERSTEEG, R. The Marmousi experience; velocity model determination on a synthetic complex data set. *The Leading Edge*, v. 13, n. 9, p. 927–936, 09 1994. ISSN 1070-485X. Citado 2 vezes nas páginas 18 e 60.

VIRIEUX, J. P-SV wave propagation in heterogeneous media: Velocity-stress finite-difference method. *Geophysics*, v. 51, n. 4, p. 889–901, 1986. Citado 5 vezes nas páginas 5, 21, 24, 109 e 110.

VIRIEUX, J.; OPERTO, S. An overview of full-waveform inversion in exploration geophysics. *Geophysics*, v. 74, n. 6, p. WCC1–WCC26, 2009. ISSN 00168033. Citado 3 vezes nas páginas 3, 16 e 39.

WANG, H. et al. Stability of finite difference numerical simulations of acoustic logging-while-drilling with different perfectly matched layer schemes. *Applied Geophysics*, v. 10, n. 4, p. 384–396, Dec 2013. ISSN 1993-0658. Disponível em: <https://doi.org/10.1007/s11770-013-0400-6>. Citado na página 28.

WIRGIN, A. The inverse crime. *Math Phys*, 02 2004. Citado na página 44.

YI, B. et al. Comparison of wavelet estimation methods. *Geosciences Journal*, v. 17, 03 2013. Citado na página 69.

YILMAZ, O. Seismic data analysis. In: _____. [s.n.], 2012. cap. 4. Migration, p. 463–654. Disponível em: <https://library.seg.org/doi/abs/10.1190/1.9781560801580.ch4>. Citado na página 112.

ZHOU, H.-W. et al. Reverse time migration: A prospect of seismic imaging methodology. *Earth-Science Reviews*, v. 179, p. 207 – 227, 2018. ISSN 0012-8252. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0012825217306256>. Citado na página 112.

Apêndices

APÊNDICE A – Discretização da equação da onda

Uma vez que a equação da onda não possui solução analítica para modelos heterogêneos, é necessário utilizar de simulações numéricas para modelar o campo de onda. Como neste trabalho são usadas as aproximações acústica e elástica da equação da onda, são usadas duas abordagens diferentes para a discretização, em ambos os casos é utilizado o método de diferença finita.

A.1 Equação da onda acústica

A discretização para o caso acústico utiliza malha regular, isto ocorre pois os fenômenos acústicos possuem baixa instabilidade ou dispersão por conta da malha. Utilizando o método de diferenças finitas, pode-se obter a derivada dos campos à partir de uma equação de diferenças. Como a derivada no tempo com ordem maior que a segunda apresenta algumas instabilidades, utiliza-se sempre a segunda ordem no tempo para a derivada temporal mas variando a ordem da derivada espacial. Discretizando as derivadas, tem-se que:

$$\frac{\partial^2 P(\vec{x},t)}{\partial t^2} \approx \frac{\left(P^{n+1}_{[i,j]} - 2P^{n}_{[i,j]} + P^{n-1}_{[i,j]}\right)}{dt^2} \tag{A.1}$$

onde $P^{n+1}_{[i,j]}$ é o campo de onda futuro, $P^{n}_{[i,j]}$ é o campo de onda presente, $P^{n-1}_{[i,j]}$ é o campo de onda passado e dt é a amostragem em tempo. Discretizando agora a derivada espacial dada por Liu e Sen (2013), obtém-se:

$$\begin{aligned}\nabla^2 P(\vec{x},t) = &\sum_{m=0}^{M} a_m \left(\frac{P^{n}_{[i-m,j]} + P^{n}_{[i+m,j]}}{dx^2}\right) + \\ &\sum_{m=0}^{M} a_m \left(\frac{P^{n}_{[i,j-m]} + P^{n}_{[i,j+m]}}{dz^2}\right),\end{aligned} \tag{A.2}$$

onde a_m é o coeficiente da diferença finita ao respectivo termo, dx é o espaçamento em x e dz é o espaçamento em z. Chamando:

$$D_x = \sum_{m=0}^{M} a_m \left(\frac{P^{n}_{[i-m,j]} + P^{n}_{[i+m,j]}}{dx^2}\right), \tag{A.3}$$

e

$$D_z = \sum_{m=0}^{M} a_m \left(\frac{P^n_{[i,j-m]} + P^n_{[i,j+m]}}{dz^2} \right), \tag{A.4}$$

pode-se obter a equação da onda a partir destes coeficientes. Sabendo que equação da onda acústica é dada pela equação 2.4, discretizando a equação em função de diferença tem-se:

$$P^{n+1}_{[i,k]} = 2P^n_{[i,k]} - P^{n-1}_{[i,k]} + v^2_{[i,k]} dt^2 \left(D_x + D_z \right). \tag{A.5}$$

Para cada passo de tempo, os campos são atualizados e assim a simulação continua até o fim do tempo de aquisição. Junto a discretização, vem os problemas de instabilidade e dispersão. A instabilidade ocorre quando se escolhe um dt grande, tal que a simulação torna-se instável e acaba gerando valores tendendo ao infinito. Já a dispersão ocorre quando a amostragem espacial é grande tal qual não consegue amostrar o menor comprimento de onda da modelagem, gerando uma repetição no sinal. Para contornar este problema, são definidos critérios de instabilidade e dispersão necessários para que a simulação possua solução confiável. O primeiro critério é a lei de Courant de instabilidade, que pode ser calculado como mostra Lines, Slawinski e Bording (1999):

$$\frac{v_{max} dt}{\sqrt{dx^2 + dz^2}} \leq \frac{2}{\sqrt{A_1}} \tag{A.6}$$

onde V_{max} é a velocidade máxima do modelo de velocidade e A_1 é a soma dos valores absolutos dos coeficientes a_m da diferença finita. Já o critério de dispersão é dado por (DABLAIN, 1986) da forma:

$$dx \ \leq \ \frac{v_{min}}{S f_n}, \tag{A.7}$$

onde V_{min} é a menor velocidade do modelo de velocidade, f_n é a frequência de Nyquist ou maior frequência presente na ondaleta e S é um valor empírico que depende da ordem da diferença finita utilizada. Os valores de S podem ser obtidos pela tabela 26:

Tabela 26 – Valores de S respectivos à cada ordem de diferença finita.

	2ª ordem	4ª ordem	6ª ordem	8ª ordem	10ª ordem
S	$\backsim 8,0$	4,61-5,92	4,28-3,58	3,64-3,15	$\backsim 3,0$

A.2 Equação da onda elástica

Diferentemente da discretização para o caso acústico, a discretização para o caso elástico não utiliza a malha regular. Isto ocorre pois há uma instabilidade na simulação na interface água-rocha utilizando malha regular, sendo um problema para modelagens elásticas. Uma solução para este problema foi proposta por Virieux (1986), que consiste em utilizar a malha intercalada proposta por Madariaga (1976) para representar as tensões e velocidades de partícula. A figura 43 ilustra como ficam as variáveis para cada ponto da malha.

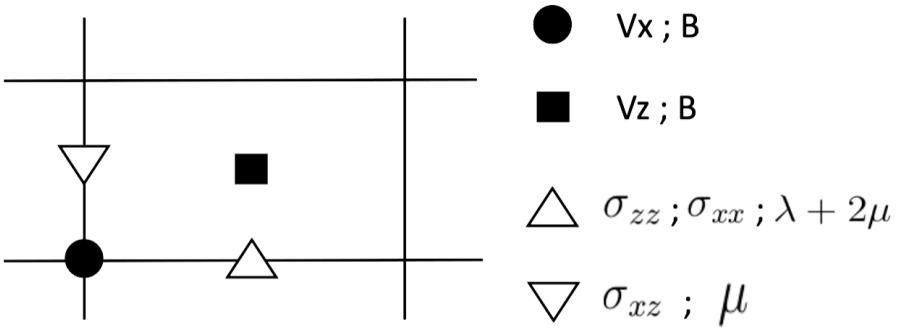


Figura 43 – Ilustração da posição de cada varíavel na malha intercalada. Modificado de Virieux (1986).

Ao discretizar a equação da onda, as derivadas parciais não necessitam de três campos como na malha regular, sendo mais uma vantagem da malha da malha intercalada com relação à malha regular. Sabendo que a derivada temporal é dada por:

$$\frac{\partial v_x(\vec{x},t)}{\partial t} = \frac{v^{n+\frac{1}{2}}_{x[i,j]} - v^{n-\frac{1}{2}}_{x[i,j]}}{dt}, \tag{A.8}$$

e que a derivada espacial é dada por:

$$\frac{\partial v_x(\vec{x},t)}{\partial x} = \sum_{m=1}^{M} a'_m \left(\frac{v^n_{x[i+\frac{m}{2},j]} - v^n_{x[i-\frac{m}{2},j]}}{dx} \right), \tag{A.9}$$

onde a'_m é o coeficiente da diferença finita da malha intercalada respectiva a cada elemento m. Utilizando essas propriedades, pode-se obter o sistema de equações diferenciais discretizado da forma:

$$\begin{aligned} v^{n+\frac{1}{2}}_{x[i,j]} = v^{n-\frac{1}{2}}_{x[i,j]} + dt\ B_{[i,j]} \sum_{m=1}^{M} a'_m \left(\frac{\sigma^n_{xx[i+\frac{m}{2},j]} - \sigma^n_{xx[i-\frac{m}{2},j]}}{dx} \right) + \\ dt\ B_{[i,j]} \sum_{m=1}^{M} a'_m \left(\frac{\sigma^n_{xz[i,j+\frac{m}{2}]} - \sigma^n_{xz[i,j-\frac{m}{2}]}}{dz} \right), \end{aligned} \tag{A.10}$$

$$v^{n+\frac{1}{2}}_{z[i+\frac{1}{2},j+\frac{1}{2}]} = v^{n-\frac{1}{2}}_{z[i+\frac{1}{2},j+\frac{1}{2}]} + dt\ B_{[i+\frac{1}{2},j+\frac{1}{2}]} \sum_{m=1}^{M} a'_m \left(\frac{\sigma^n_{zz[i+\frac{1}{2},j+\frac{1+m}{2}]} - \sigma^n_{zz[i+\frac{1}{2},j+\frac{1-m}{2}]}}{dz} \right) + dt\ B_{[i+\frac{1}{2},j+\frac{1}{2}]} \sum_{m=1}^{M} a'_m \left(\frac{\sigma^n_{xz[i+\frac{1+m}{2},j+\frac{1}{2}]} - \sigma^n_{xz[i+\frac{1-m}{2},j+\frac{1}{2}]}}{dx} \right), \tag{A.11}$$

$$\sigma^{n+1}_{xx[i+\frac{1}{2},j]} = \sigma^n_{xx[i+\frac{1}{2},j]} + dt\ [\lambda_{[i+\frac{1}{2},j]} + 2\mu_{[i+\frac{1}{2},j]}] \sum_{m=1}^{M} a'_m \left(\frac{v^{n+\frac{1}{2}}_{x[i+\frac{1+m}{2},j]} - v^{n+\frac{1}{2}}_{x[i+\frac{1-m}{2},j]}}{dx} \right) + dt\ \lambda_{[i+\frac{1}{2},j]} \sum_{m=1}^{M} a'_m \left(\frac{v^{n+\frac{1}{2}}_{z[i+\frac{1}{2},j+\frac{1+m}{2}]} - v^{n}_{z[i+\frac{1}{2},j+\frac{1-m}{2}]}}{dz} \right), \tag{A.12}$$

$$\sigma^{n+1}_{zz[+\frac{1}{2},j]} = \sigma^n_{zz[i+\frac{1}{2},j]} + dt\ [\lambda_{[i+\frac{1}{2},j]} + 2\mu_{[i+\frac{1}{2},j]}] \sum_{m=1}^{M} a'_m \left(\frac{v^{n+\frac{1}{2}}_{z[i,j+\frac{1+m}{2}]} - v^{n+\frac{1}{2}}_{z[i,j-\frac{1-m}{2}]}}{dz} \right) + dt\ \lambda_{[i+\frac{1}{2},j]} \sum_{m=1}^{M} a'_m \left(\frac{v^{n+\frac{1}{2}}_{x[i+\frac{1+m}{2},j]} - v^{n+\frac{1}{2}}_{x[i+\frac{1-m}{2},j]}}{dx} \right), \tag{A.13}$$

e

$$\sigma^{n+1}_{xz[i,j+\frac{1}{2}]} = \sigma^n_{xz[i,j+\frac{1}{2}]} + dt\ \mu_{[i,j+\frac{1}{2}]} \sum_{m=1}^{M} a'_m \left(\frac{v^{n+\frac{1}{2}}_{x[i,j+\frac{1+m}{2}]} - v^{n+\frac{1}{2}}_{x[i,j+\frac{1-m}{2}]}}{dz} \right) + dt\ \mu_{[i,j+\frac{1}{2}]} \sum_{m=1}^{M} a'_m \left(\frac{v^{n+\frac{1}{2}}_{z[i+\frac{1+m}{2},j]} - v^{n}_{z[i+\frac{1-m}{2},j]}}{dx} \right). \tag{A.14}$$

Por conta do arranjo de malha intercalada, não há a necessidade de atualizar os campos como há na malha regular. Pode-se perceber também que os campos de tensão e de velocidade estão deslocados não só em espaço como em tempo também, devendo ser calculados em instantes diferentes na modelagem. Como na malha regular, há a necessidade de se calcular os critérios de instabilidade e dispersão. Um critério de instabilidade é dado por Virieux (1986), sendo descrito pela equação:

$$\frac{dt\sqrt{v^2_{pmax} + v^2_{smax}}}{\sqrt{dx^2 + dz^2}} < 1, \tag{A.15}$$

sendo V_{pmax} a velocidade compressional máxima do modelo e V_{smax} a velocidade cisalhante máxima do modelo. Já para a dispersão é utilizada a mesma relação da equação A.7, mas desconsiderando os valores nulos de velocidade cisalhante que o modelo de velocidade possui.

APÊNDICE B – Migração RTM

A migração é uma técnica de imageamento extremamente robusta, que busca colapsar as difrações e mover refletores inclinados para sua posição verdadeira (YILMAZ, 2012). Uma técnica de migração muito utilizada na indústria é a migração Kirchhoff (BLEISTEIN; GRAY, 2008), ela é muito utilizada por ser o método que provou ser um dos mais flexíveis e robustos no imageamento sísmico (BEDNAR, 2005). Por muito tempo, a migração Kirchhoff foi largamente empregada na indústria como ferramenta principal de migração, mas esta sofre de alguns problemas. O modelo de velocidade necessita ser suave, sem muito contraste de velocidade lateral. Por conta desta característica, regiões de alta complexidade geológica são dificilmente imageados com precisão, sendo deslocados ou até não imageados. Com o conceito de Claerbout e Doherty (1972) e o aumento do poder computacional, a migração reversa no tempo (*Reverse Time Migration* ou RTM) se popularizou cada vez mais. A migração RTM se baseia em depropagar reversamente no tempo o sismograma obtido na aquisição (campo de onda ascendente) e correlacioná-lo ao campo de onda gerado na modelagem (campo de onda descendente). Diferentemente da migração Kirchhoff, que se utiliza de uma aproximação de alta frequência da equação da onda, a migração RTM utiliza a equação da onda completa para depropagar o campo ascendente. A figura 44 mostra como é o imageamento do refletor a partir do campo descendente e ascendente.

Quando se obtém o modelo de velocidade correto, o campo descendente coincide com o campo ascendente no refletor que gerou a reflexão, qualquer outro tipo de reflexão ou amplitude não será imageado caso não haja sobreposição dos campos. Para calcular a contribuição de cada campo no imageamento dos refletores, diversas condições de imagem são adotadas na indústria, sendo as mais comuns a condição de imagem de correlação cruzada e a de compensação de iluminação.

Uma fase chave na migração RTM é a condição de imagem, a condição de imagem serve para calcular a melhor relação entre os campos descendente e ascendente (ZHOU et al., 2018). A condição de imagem mais comum é a de correlação cruzada ou correlação *Zero-lag*, sendo dado por:

$$Im(\vec{x}) = \sum_{i=0}^{N_t} A(\vec{x}, i) D(\vec{x}, i), \tag{B.1}$$

onde $Im(\vec{x})$ é a imagem gerada, N_t é o número de amostras de tempo, $A(\vec{x}, i)$ é o campo de onda ascendente e $D(\vec{x}, i)$ é o campo de onda descendente. Uma denominação comum é chamar o campo ascendente de campo do receptor e o campo descendente de campo da

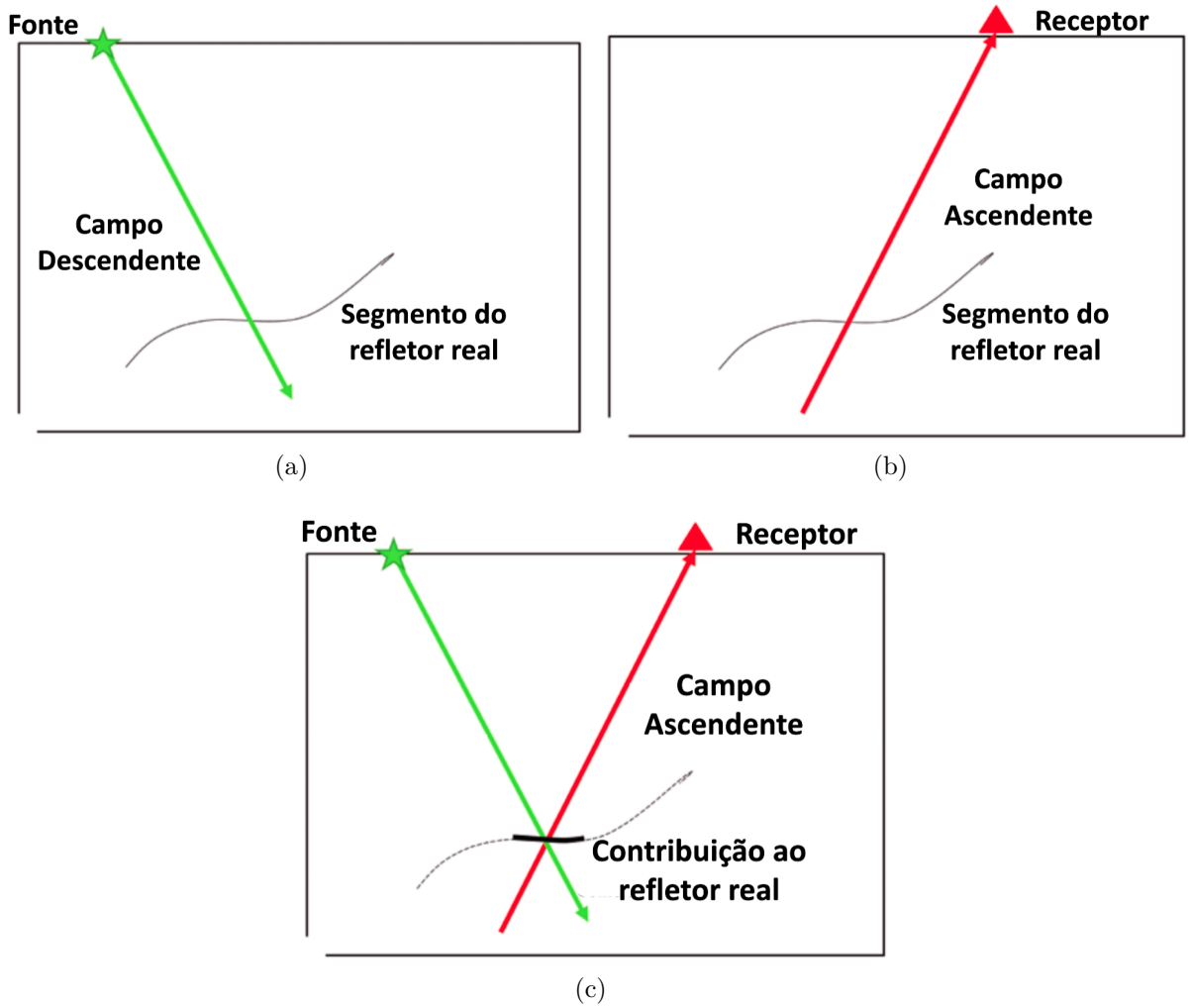


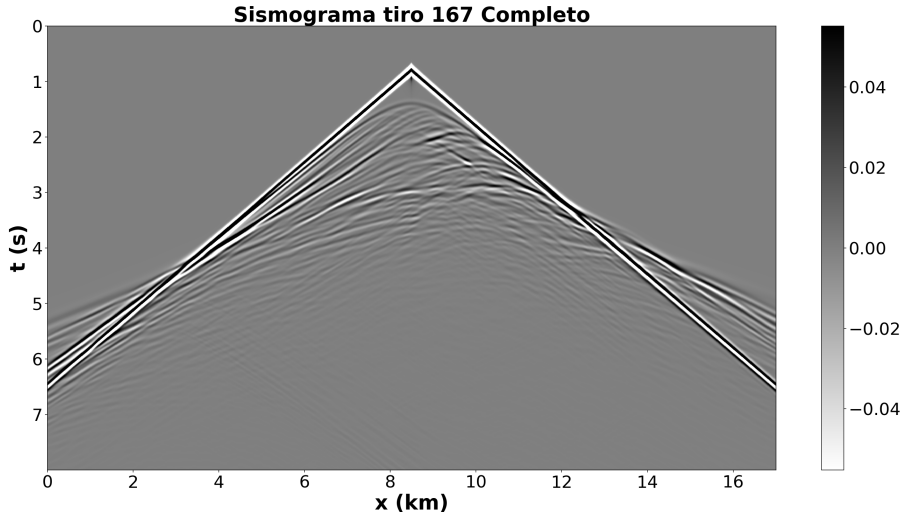
Figura 44 – a) Continuação do campo descendente. b) Depropagação do campo ascendente obtido no receptor. c) Coincidência dos campos descendentes e ascendentes no refletor real.

fonte. Apesar de facilmente aplicável e identificar as estruturas geológicas, este tipo de condição de imagem não possui relação física com o coeficiente de reflexão real do dado (CHATTOPADHYAY; MCMECHAN, 2008). Para contornar este problema, uma forma de normalizar é calculando o somatório do campo descendente e escalando a correlação cruzada por ela, ficando da forma:

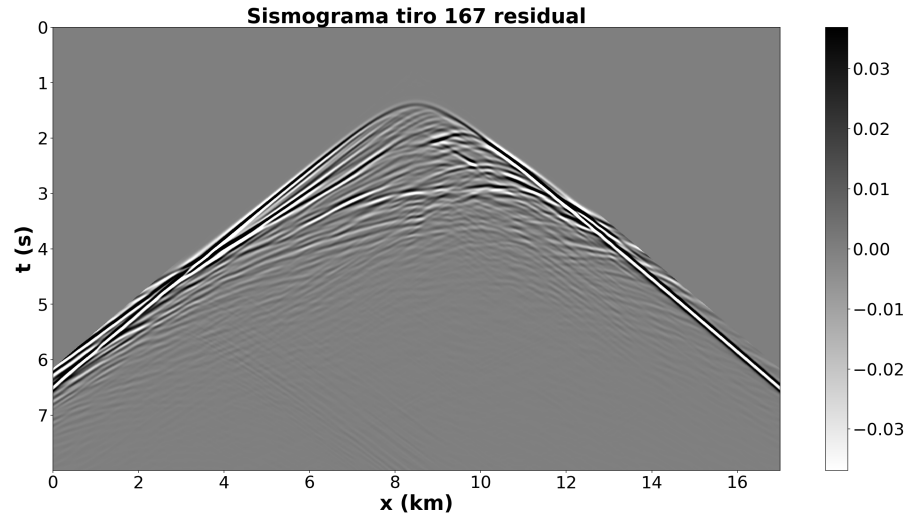
$$Im(\vec{x}) = \frac{\sum_{i=0}^{N_t} A(\vec{x}, i) D(\vec{x}, i)}{\sum_{i=0}^{N_t} D(\vec{x}, i) D(\vec{x}, i) + \epsilon}, \tag{B.2}$$

onde ϵ é uma constante arbitrária para evitar divisão por zero onde não há cobertura do campo descendente. Desta forma é possível escalar fisicamente as amplitudes encontradas na migração, obtendo uma imagem facilmente manipulável para filtragens e outras técnicas de processamento de imagens. Este trabalho utiliza esta abordagem para analisar a melhoria na imagem da inversão FWI. Como a teoria da migração RTM visa imagear eventos a

partir de reflexões, são feitas etapas adicionais para amplificar a qualidade da imagem final migrada. Estas etapas são as de remoção da onda direta e o *mute* do conteúdo de ondas refradas contidas no registro sísmico. A figura 45 mostra o sismograma do tiro 167 para o modelo de Marmousi e o resultado da remoção da onda direta e da remoção das ondas refratadas presentes nos longos espaçamentos.



(a)



(b)

Figura 45 – a) Sismograma contendo todos as informações da aquisição sísmica. b) Sismograma resultante da eliminação da onda direta e da aplicação do *mute*.

Sendo removido todo o conteúdo presente após a onda direta. O sismograma da figura 45.b) é utilizado como entrada para a migração e geração da imagem final empilhada.

APÊNDICE C – Pseudocódigos de Modelagem e Inversão

Neste capítulo são escritos os pseudo-códigos utilizados na modelagem acústica, modelagem elástica e modelagem FWI, estes descrevem os *loops* principais paralelizados e como é o fluxo de dados entre o *Host* e a GPU. O código da modelagem acústica é dado pelo pseudocódigo C.1.

```
1
2  Iniciar;
3
4  ler parametros de geometria;
5  alocar variaveis (Campos de Pressao, fonte, velocidade ...);
6  ler variaveis de arquivos binarios (Velocidade,fonte, dado observado);
7  zerar variaveis de pressao e de memoria da CPML;
8
9  #pragma acc enter data copyin(Variaveis de pressao, velocidade ...)
10
11 For(n = 0, Nt)
12 {
13   #pragma acc kernels
14   {
15     Inserir fonte
16   }
17   #pragma acc parallel loop present(Campos de pressao, variaveis CPML,
       ...) collapse(2)
18   For(x = 0, Nx_borda)
19   {
20     For(x = 0$, Nz_borda)
21     {
22       Calculo das variaveis de memoria da CPML;
23     }
24   }
25   #pragma acc parallel loop present(Campos de Pressao, variaveis CPML)
      collapse(2)
26   For(x = 0, Nx)
27   {
28     For(z = 0, Nz)
29     {
30       Calculo campo de pressao interior do modelo ;
31     }
32    }
33
```

```
34   #pragma acc parallel loop present(Campos de Pressao) collapse(2)
35   For(x = 0, Nx)
36   {
37     For(z = 0, Nz)
38     {
39       Atualizacao dos campos de pressao ;
40     }
41   }
42
43
44   #pragma acc parallel loop present(Campos de Pressao,sismograma)
45   For(i = 0, Nrec)
46   {
47     Calcular o sismograma;
48   }
49 }
50 #pragma acc exit data copyout(Sismograma)
51 #pragma acc exit data delete(Variaveis de pressao, velocidade ...)
```

Listing C.1 – Pseudocódigo de modelagem acústica

De forma muito similar à paralelização da modelagem acústica, a modelagem elástica também recebe o mesmo fluxo. No entanto, como não há o cálculo das variáveis de memória da CPML, não se é necessário calculá-las para a borda, calculando a equação da onda no modelo todo. O pseudo código da modelagem elástica é dado por:

```
1
2 Iniciar;
3
4 ler parametros de geometria;
5 alocar variaveis (Campos de tensao e velocidade, fonte, velocidade ...);
6 ler variaveis de arquivos binarios (Velocidade,fonte);
7 zerar variaveis de tens o e velocidade;
8
9 #pragma acc enter data copyin(Variaveis de tens o , velocidade,
     densidade ...)
10
11 For(n = 0, Nt)
12 {
13   #pragma acc kernels
14   {
15     Inserir fonte nas tensoes
16   }
17   #pragma acc parallel loop present(Campos de tensao, variaveis de
      velocidade, ...) collapse(2)
18   For(x = 0, Nx_borda)
19   {
```

```
20      For(x = 0$, Nz_borda)
21      {
22        Calculo das variaveis de memoria da CPML;
23      }
24    }
25    #pragma acc parallel loop present(Campos de tensao e velocidade)
       collapse(2)
26    For(x = 0, Nx)
27    {
28      For(z = 0, Nz)
29      {
30        Calculo campo de tensao e velocidade;
31      }
32    }
33
34
35    #pragma acc parallel loop present(Campos de tensao e velocidade,dado
       sismograma ..)
36    For(i = 0, Nrec)
37    {
38      Calcular o sismograma;
39    }
40    }
41 #pragma acc exit data copyout(Sismograma)
42 #pragma acc exit data delete(Variaveis de tensao, velocidade ...)
```

Listing C.2 – Pseudocódigo de modelagem elástica

Por fim, é apresentado o pseudocódigo de inversão FWI, dado por:

```
1
2 Iniciar;
3
4 ler parametros de geometria;
5 alocar variaveis (Campos de Pressao, fonte, velocidade ...);
6 ler variaveis de arquivos binarios (Velocidade,fonte, dado observado);
7 zerar variaveis de pressao e de memoria da CPML;
8
9 #pragma acc enter data copyin(Variaveis de pressao, velocidade ...)
10
11 For(n = 0, Nt)
12 {
13   #pragma acc kernels
14   {
15     Inserir fonte
16   }
17   #pragma acc parallel loop present(Campos de pressao, variaveis CPML,
       ...) collapse(2)
```

```
18   For(x = 0, Nx_borda)
19   {
20     For(x = 0$, Nz_borda)
21     {
22       Calculo das variaveis de memoria da CPML;
23     }
24   }
25   #pragma acc parallel loop present(Campos de Pressao, variaveis CPML)
      collapse(2)
26   For(x = 0, Nx)
27   {
28     For(z = 0, Nz)
29     {
30       Calculo campo de pressao interior do modelo ;
31     }
32   }
33   if(resto(n/fator) = 0)
34   {
35     #pragma acc parallel loop present(Campos de Pressao,matriz derivada)
       collapse(2)
36     For(x = 0, Nx)
37     {
38       For(z = 0, Nz)
39       {
40         Calcular derivada temporal do campo de pressao;
41       }
42     }
43   }
44   #pragma acc parallel loop present(Campos de Pressao,dado calculado ..)
45   For(i = 0, Nrec)
46   {
47     Calcular o sismograma do dado calculado ;
48   }
49 }
50 #pragma acc parallel loop present(dado observado,dado calculado,residuo)
       collapse(2)
51 For(i = 0, Nrec)
52 {
53   For(n = 0, Nt)
54   {
55     Calcular residuo;
56     calcular funcao objetivo;
57   }
58 }
59 zerar variaveis de pressao e de memoria da CPML;
60 For(n = Nt, 0,-1)
61 {
```

```
62   #pragma acc parallel loop present(residuo)
63   For(i = 0, Nrec)
64   {
65     depropagar o residuo
66   }
67   #pragma acc parallel loop present(Campos de pressao, variaveis CPML,
       ...) collapse(2)
68   For(x = 0, Nx_borda)
69   {
70     For(x = 0, Nz_borda)
71     {
72     Calculo das variaveis de memoria da CPML;
73     }
74   }
75   #pragma acc parallel loop present(Campos de Pressao, variaveis CPML)
      collapse(2)
76   For(x = 0, Nx)
77   {
78     For(z = 0, Nz)
79     {
80       Calculo campo de pressao interior do modelo ;
81     }
82   }
83   if(resto(n/fator) = 0)
84   {
85     #pragma acc parallel loop present(Campos de Pressao,matriz derivada)
       collapse(2)
86     For(x = 0, Nx)
87     {
88       For(z = 0, Nz)
89       {
90         calcular o gradiente;
91       }
92     }
93   }
94
95 }
96
97 #pragma acc update host(gradiente)
98
99 #pragma acc data delete(Variaveis de pressao, velocidade ...)
```

Listing C.3 – Pseudocódigo da inversão FWI