

DISSERTAÇÃO DE MESTRADO

**FWI MULTIESCALA: UMA
IMPLEMENTAÇÃO EM GPU**

VICTOR KOEHNE

SALVADOR – BAHIA
MARÇO – 2018

FWI multiescala: uma implementação em GPU

por

VICTOR KOEHNE

Geofísica (Universidade Federal da Bahia – 2014)

Orientador: Reynam da Cruz Pestana

DISSERTAÇÃO DE MESTRADO

Submetida em satisfação parcial dos requisitos ao grau de

MESTRE EM CIÊNCIAS

EM

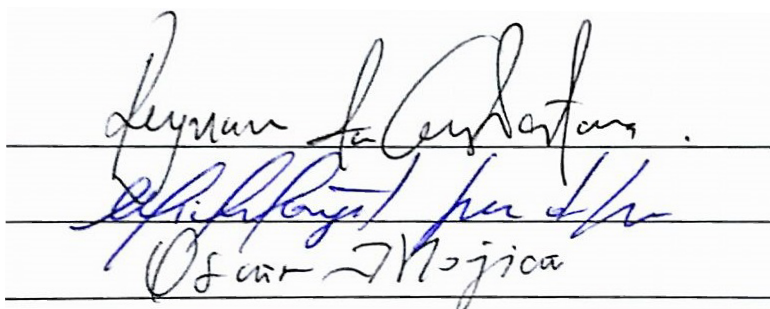
GEOFÍSICA

ao

Conselho Acadêmico de Ensino

da

Universidade Federal da Bahia



Comissão Examinadora

Dr. Reynam da Cruz Pestana

Dr. Michelângelo Gomes da Silva

Dr. Oscar Fabian Mojica Ladino

Aprovada em 05 de março de 2018

A presente pesquisa foi desenvolvida no Centro de Pesquisa em Geofísica e Geologia da UFBA, com recursos próprios, da CAPES, da CNPq, CTPETRO, ANP, PETROBRAS

Koehne, Victor,

FWI multiescala: uma implementação em GPU / Victor Koehne. — Salvador, 2018.

172 f.: il.

Orientador: Reynam da Cruz Pestana

Dissertação (Mestrado) - Pós-Graduação em Geofísica. Instituto de Geociências da Universidade Federal da Bahia, 2018.

1. FWI. 2. GPU. 3. Modelagem Sísmica. 4. RTM. 5. REM.
I. Pestana, Reynam. II. Título.

Resumo

A inversão completa da forma de onda (FWI - do inglês *full-waveform inversion*) é atualmente uma das principais ferramentas para determinar modelos de velocidades da subsuperfície com alta resolução. Nessa dissertação, a FWI no domínio do tempo é introduzida e desenvolvida sob a ótica da implementação, na seguinte sequência: modelagem sísmica, migração reversa no tempo (ou *reverse time migration*, RTM) e FWI. É mostrado que a RTM, do ponto de vista computacional, equivale a duas modelagens sísmicas, ou três modelagens se for usada a implementação com borda efetiva. A abordagem da FWI como um problema iterativo que visa minimizar o resíduo dos dados sísmicos mostra que o gradiente de cada iteração é, pelo método adjunto, equivalente à RTM do resíduo. Dessa maneira, usando RTM com borda efetiva e um método de estimação do passo, é mostrado que uma iteração da FWI é computacionalmente equivalente a quatro modelagens sísmicas.

Como se sabe, a modelagem é um processo muito intensivo computacionalmente, e dentre as formas eficientes de resolver esse problema se destaca o uso de computação paralela. Nessa dissertação, se escolheu a paralelização utilizando placas gráficas (ou *graphics processing unit*, GPU) que possui alta capacidade de cálculo de pontos flutuantes, porém baixa eficiência na transferência de dados. Tais características se adequam muito bem ao problema de extrapolação de campos de onda no tempo, em especial no cálculo do Laplaciano da equação da onda acústica em cada ponto do modelo, que correspondem a quase todo o tempo computacional da modelagem, custo esse que, paralelizado em GPU, mais que compensa as transferências de memória CPU-GPU inerentes ao problema.

A GPU, todavia, tem uma restrição de memória, tipicamente variando entre 2, 5 e 12 GB. Nessa dissertação, então se focou em técnicas que permitem a economia de memória em troca de processamento. Dentre essas implementações, destaca-se o uso de borda efetiva na RTM e o uso do método de expansão rápida (REM), que permite extrapolação a maiores intervalos de tempo, reduzindo o número total de amostras temporais que precisam ser armazenadas e transferidas na memória do equipamento utilizado.

A implementação em GPU também permitiu testar em tempo hábil alguns dos fatores

mais importantes que influenciam a FWI. Foram testados, numa malha regular: quatro operadores de modelagem - diferenças finitas (DF), pseudo-espectral (PS), REM-DF e REM-PS-; as condições de borda absorvedora *taper* e *perfectly matching layer* (PML); e os métodos de inversão conjugado gradiente não linear (CGNL) e L-BFGS. Os resultados desses testes permitiram selecionar os melhores critérios para execução da FWI em modelos de velocidades sintéticos, porém de geologia complexa. O uso da metodologia multiescala, essencial para evitar convergência a mínimos locais, associado a uma modelagem extra para obtenção de um passo adequado, permitiu a obtenção de resultados finais de alta resolução para os três modelos testados.

Abstract

Full-waveform inversion (FWI) is nowadays one of the main tools for estimating high resolution subsurface velocity models. In this dissertation, time-domain FWI is introduced from an algorithmic point of view: seismic modeling, reverse time migration (RTM), and FWI. It is shown that RTM, from a computational point of view, is equivalent to two seismic modeling processes, or three if the effective boundaries implementation is used. The approach of FWI as an iterative problem (which aims to minimize the seismic data residue) shows that the gradient of each iteration, using the adjoint-state method, is equivalent to the RTM of the residue. In this manner, using RTM with effective boundaries and a step length estimation method, it is shown in this thesis that one iteration of FWI is computationally equivalent of four seismic modelling processes.

It is known that seismic modeling is a highly intensive computational process, and among the techniques to mitigate this cost the use of parallel computing stands out. In this dissertation we chose the parallelization using the graphics processing unit (GPU) which has high floating point computation capability, but low efficiency in data transfer. These characteristics fit very well to the problem of wave field extrapolation in time, especially in the calculation of the Laplacian of the acoustic wave equation at each point of the model, whose computational costs in GPU more than compensates the data transfers inherent to the problem.

The GPU, however, has a memory constraint, typically ranging from 2, 5 and 12 GB. In this thesis, we then focused in techniques that allowed memory savings in exchange of processing. Among these implementations, we highlight the use of effective boundaries in RTM and the rapid expansion method (REM) for time extrapolation, which allows marching at longer time steps, reducing the total time samples that need to be stored and transferred.

The GPU implementation also enabled to test in a timely manner some of the most important factors influencing FWI. We tested, in a regular grid: four modeling operators - finite-differences (FD), pseudo-spectral (PS), REM-FD, REM-PS -; the absorbing boundary conditions taper and perfectly matching layer (PML); and the inversion methods L-BFGS

and non-linear conjugate gradient (NLCG). The results of these tests allowed to select the best criteria for FWI execution in synthetic velocity models of complex geology. The use of the multiscale methodology, essential to avoid convergence to local minima, in conjunction with an extra modeling step to ensure an efficient step length, allowed achieving final results of high resolution for the three tested models.

Índice

Resumo	3
Abstract	5
Índice	7
Índice de Tabelas	10
Índice de Figuras	11
Introdução	17
1 Modelagem Sísmica	21
1.1 Equação da onda acústica	23
1.2 Soluções numéricas da equação da onda acústica	27
1.2.1 Diferenças finitas em tempo e espaço	27
1.2.2 Método pseudo-espectral	35
1.2.3 Método de Expansão Rápida (REM)	38
1.3 Reflexão nos limites do modelo	42
2 Migração Reversa no Tempo	45
2.1 Modificações na condição de imagem	54
2.2 Borda efetiva	55
2.3 Artefatos da condição de imagem de correlação cruzada	58
3 Inversão da Forma de Onda completa (FWI)	60
3.1 Os métodos de Newton, quasi-Newton, Gauss-Newton e <i>Steepest-Descent</i>	64
3.2 Métodos <i>steepest-descent</i>	66
3.2.1 Cálculo do gradiente $\nabla E(\mathbf{m})$	68
3.2.2 Cálculo da Jacobiana J na direção $\mathbf{s}$	69
3.2.3 Método Conjugado Gradiente Não Linear	71
3.3 Método L-BFGS	71

3.4	Normalização das amplitudes	74
3.5	Determinação da assinatura da fonte	75
3.6	Pré-condicionamento do gradiente	75
3.7	Abordagem Multiescala	78
3.7.1	Filtragem do dado observado	80
3.7.2	Etapas da abordagem multiescala e exemplificação	82
4	Implementação em placas gráficas (GPUs)	88
4.1	A unidade de processamento gráfico - GPU	88
4.2	Arquitetura da GPU - CUDA	90
4.3	Memória da GPU	95
4.4	Observações sobre custo computacional	97
4.5	Bibliotecas úteis: cuBLAS e cuFFT	99
5	Metodologia e resultados	101
5.1	Metodologia	101
5.2	Ajustes à implementação da FWI	103
5.2.1	Normalização das amplitudes	104
5.2.2	Influência da onda direta	107
5.3	Análise dos operadores de modelagem	113
5.3.1	Comparação entre diferentes operadores de DF	114
5.3.2	Comparação entre operadores DF, pseudo-espectral, REM-DF e REM-pseudo-espectral	114
5.4	Análise dos métodos de inversão	117
5.4.1	L-BFGS com passo variável vs. L-BFGS com passo unitário	119
5.4.2	L-BFGS vs. Conjugado Gradiente não-linear	124
5.5	Notas sobre $\mathbf{d}_{\text{obs}}$ sintético	126
5.5.1	Influência do operador de modelagem	126
5.5.2	Influência da borda absorvedora	129
5.6	Resultados	134
5.6.1	Modelo Marmousi	136
5.6.2	Modelo <i>Overthrust</i>	141
5.6.3	Modelo Falhas	145
6	Conclusões	149
	Agradecimentos	151

Apêndice A	Dedução das equações matriciais do método inverso	152
A.1	Formulação matricial de $\Delta \mathbf{m}$ em função de $\mathbf{d}_{\text{cal}}$	152
A.2	Matriz sensibilidade: forma usual	156
Apêndice B	Cálculo do gradiente pelo método adjunto	158
B.1	Aplicação à FWI	160
Apêndice C	Pseudocódigos	164
Referências Bibliográficas		171

Índice de Tabelas

2.1	<i>Tradeoff</i> computacional, RTM (1 tiro).	55
4.1	Tempo de processamento para a RTM em GPU.	99
4.2	Tempo de processamento para a RTM em GPU.	99
5.1	Modelagem na GPU de 1 tiro, para diferentes operadores de DF. $\Delta t = 0,001s$, $nt = 4000$	114
5.2	Modelagem na GPU de 1 tiro, operadores DF, PS e REM. $\Delta t = 0,001s$, $nt = 4000$	115
5.3	Modelagem na GPU de 1 tiro, operadores DF, PS e REM, para um $\mathbf{d}_{\text{obs}}$ de 0,004 s.	116
5.4	Tempos de processamento para a FWI utilizando seis métodos gradiente com critérios de passo inicial diferentes, e para o método L-BFGS com passo uni- tário. Retirado de dos Santos (2013).	118
5.5	Tempos de processamento da FWI para diferentes passos do L-BFGS, e um mesmo valor final da função objetivo. Retirado de dos Santos (2013).	119
5.6	Tempos de processamento (GPU) de 120 iterações da FWI para os três testes com o passo do L-BFGS, e operador DF.	121
5.7	Tempos de processamento (GPU) de 120 iterações da FWI para os três testes com o passo do L-BFGS, e operador PS.	121
5.8	Tempos de processamento (GPU) para 120 iterações duma FWI multiescala, para os métodos CG não linear e L-BFGS.	124
5.9	Tempos medidos e estimados pela equação 5.4	136
5.10	Tempos computacionais para FWI com DF16 e REM-DF16.	138
5.11	Tempos de processamento, FWI do modelo <i>Overthrust</i> , para diferentes quan- tidades de tiros. Os tempos em série foram calculados por 5.4, a partir do tempo de 1 tiro.	142
5.12	Custos computacionais para a FWI do modelo Falhas.	147

Índice de Figuras

1.1	a) Campo de velocidades com duas camadas, de 1500 m/s e 3500 m/s. Propagação da onda, após: b) 2 ms, c)10 ms e d)35 ms.	22
1.2	a) <i>Wavelet</i> Ricker com $f_{peak} = 15Hz$. b) Espectro de amplitudes. f_{max} ultrapassa $30Hz$ ($2f_{peak}$), chegando próximo de $50Hz$, porém as altas frequências tem baixa amplitude.	26
1.3	Modelagem de um tiro, num modelo de duas camadas, usando uma Ricker de $f_{peak} = 15$ Hz como fonte. a) Sismograma para 101 geofones, split-spread. b)Espectro de amplitude dos traços mais próximos à fonte (traço 51). A frequência máxima ultrapassa $2f_{peak} = 30$ Hz.	27
1.4	Propagação da onda num modelo de duas camadas. a) Propagação livre de dispersão. b) Dispersão temporal (à frente da frente de onda principal) e espacial (atrás da frente de onda principal).	33
1.5	Propagação da onda num modelo de duas camadas. a) Propagação livre de dispersão. b) Dispersão espacial (atrás da frente de onda principal).	34
1.6	Quebra da primeira condição de dispersão para o REM. a) $f_{max} = 23,10$ Hz, $\Delta t = 4$ ms. b) $f_{max} = 46,20$ Hz, $\Delta t = 4$ ms.	41
1.7	Quebra da segunda condição de dispersão para o REM. a) $f_{max} = 23,10$ Hz, $\Delta t = 4$ ms. b) $f_{max} = 23,10$ Hz, $\Delta t = 24$ ms.	41
1.8	Propagação num modelo de 2 camadas, após $20ms$. a) Sem borda absorvedora; b) com <i>taper</i> ; c) com PML (Moreira et. al., 2011); d) com PML ("Collino modificado", de Collino e Tsogka, 2011).	44
2.1	Propagação reversa no tempo de um tiro, visto em (a). Início da propagação reversa, fatia 3,8 s do sismograma é inserida na superfície, como indicado (b).	46
2.2	Continuação da propagação reversa do tiro. Em (a), é inserida a fatia 3,2 s do sismograma, onde se pode perceber o evento decorrente da borda (não atenuado completamente) sendo depropagado. O processo continua sequencialmente; em (b) é inserida a fatia 2,08 s do sismograma, onde vemos o evento da reflexão começando a ser depropagado.	47

2.3	Continuação da depropagação. Fatia 1,04 s em (a), fatia próxima de 0 s em (b).	48
2.4	Construção da imagem de um tiro pela RTM, para um modelo de duas camadas.	49
2.5	Construção da imagem de um tiro pela RTM, para um modelo de um ponto difrator. Retirado de Lines et. al. (2010)	50
2.6	Imagem da RTM (pós filtro Laplaciano) num modelo de duas camadas, para (a) 1 tiro, (b) 2 tiros, (c) 3 tiros e (d) 25 tiros. Pode-se observar que cada tiro ilumina melhor o ponto do refletor diretamente abaixo de si. Além da imagem do refletor, há artefatos na superfície decorrentes da onda direta de cada tiro.	52
2.7	Imagens da RTM para o modelo Marmousi. (a) Condição de imagem usual. (b) Condição de imagem com compensação de iluminação do campo de onda da fonte.	54
2.8	Esquematização da malha de modelagem, mostrando a borda absorvedora $A_1A_2A_3A_4 - C_1C_2C_3C_4$; a borda efetiva $D_1D_2D_3D_4 - A_1A_2A_3A_4$ e o modelo (zona de interesse) $A_1A_2A_3A_4$. Note que a borda efetiva está dentro do modelo. Retirado de Yang et. al. (2014).	56
2.9	Depropagação do campo da fonte utilizando borda efetiva. (a) 3,92 s; (b) 2,2 s; (c) 1 s; (d) 0,02 s.	57
2.10	Migração (RTM) de um tiro para o modelo Sigsbee 2D. Pode-se ver as correlações indesejadas dos campos acima do refletor, ao longo de todo o caminho fonte-refletor-receptores, em (a); e a atenuação desses artefatos por um filtro laplaciano, em (b). Retirada de Liu et. al. (2011).	59
3.1	Modelo de velocidades original (a) e suavizado (b).	61
3.2	Esboço da função objetivo não-linear $E(\mathbf{m})$ para um caso de 2 parâmetros (m_1 e m_2). $E(\mathbf{m} + \Delta\mathbf{m})$ é um parabolóide em função de $\Delta\mathbf{m}$. Em vermelho, derivadas nulas em cada parâmetro dão o ponto de mínimo.	63
3.3	a) Modelo verdadeiro com velocidade de fundo 2000 m/s e dois espalhadores de velocidade 2500 m/s. b) Modelo inicial, com apenas um espalhador. Retirado de dos Santos (2013).	69
3.4	Sismogramas sintéticos gerados com: a) modelo verdadeiro; b) modelo inicial. c) Diferença entre (a) e (b) (resíduo). Retirado de dos Santos (2013).	70
3.5	Migração do resíduo da Figura 3.4 c, que é o gradiente da função objetivo E pelo método adjunto. Retirado de dos Santos (2013).	70

3.6	Influência do pré-condicionamento do gradiente na FWI. a) Resultado sem pré-condicionamento. b) Resultado com pré-condicionamento. c) Erro RMS total para os resultados (a) e (b).	77
3.7	Interpretação heurística da abordagem multiescala. Cada painel representa o gráfico de uma função objetivo, desde a menor escala (maior frequência) (a) até a maior escala (menor frequência) (e). Adaptado de Bunks et. al. (1995).	79
3.8	Causas de mínimos locais na função objetivo para FWI. (a) Erros cinemáticos grandes fazem que o gradiente seja erroneamente nulo; (b) Altas frequências geram máximos locais na correlação da função objetivo (<i>cycle-skipping</i>). Retirado de Bunks et. al. (1995).	80
3.9	Formação de artefatos de <i>cycle-skipping</i> , para uma componente de onda monocromática, na FWI. A linha sólida representa uma componente do dado observado, e as linhas pontilhadas, do dado modelado, ambas com um <i>delay</i> maior que $T/2$. No caso da modelagem superior, a FWI vai atualizar o modelo como se houvesse correlação entre o ciclo $(n+1)$ dos sismogramas modelados e o ciclo n do dado observado, gerando um modelo errôneo. No caso da modelagem inferior, o ciclo n do dado modelado vai correlacionar com o ciclo n do dado observado, gerando uma atualização correta do modelo. O limite máximo do delay para que não ocorra <i>cycle-skipping</i> é de $T/2$. Retirado de Virieux e Operto (2009).	81
3.10	Efeitos do filtro de Wiener: nos sismogramas, (a) 30 Hz e (d) filtrado a 6 Hz; no espectro de amplitude dos sismogramas, (b) 30 Hz e (e) filtrado 6 Hz; <i>wavelet</i> de 30 Hz usada para gerar o dado observado, em (c); <i>wavelet</i> de 6 Hz, usada no filtro de Wiener. Retirado de dos Santos (2013).	82
3.11	Modelo Marmousi de 125×383 amostras, a) verdadeiro e b) suavizado. . . .	84
3.12	FWI de toda a banda de frequência, de 0 – 30 Hz, do dado observado. a) Função objetivo, por iteração; b) Erro RMS entre modelo verdadeiro e estimado, por iteração; c) modelo verdadeiro e d) modelo estimado final.	85
3.13	FWI da banda de frequência de 0 – 7,5 Hz. a) Função objetivo, por iteração; b) Erro RMS entre modelo verdadeiro e estimado, por iteração; c) modelo verdadeiro e d) modelo estimado final.	86
3.14	Resultado multiescala para 4 bandas de frequência, até 30 Hz. a) Função objetivo, por iteração; b) Erro RMS entre modelo verdadeiro e estimado, por iteração; c) modelo verdadeiro e d) modelo estimado final.	87
4.1	Operações em ponto flutuante (GFLOPS) - CPU e GPU ao longo dos últimos anos, retirado de NVIDIA CUDA C Programming Guide (2017).	89

4.2	Diferenças nas arquiteturas da CPU e GPU, retirado de NVIDIA CUDA C Programming Guide (2017).	89
4.3	Grupos de threads são organizados em blocks; grupos de blocks são organizados em grids, retirado de NVIDIA CUDA C Programming Guide (2017). . .	91
4.4	Somando dois vetores.	92
4.5	Esquematização das memórias em CUDA.	96
4.6	Performance no uso de diferentes memórias em implementação em GPU do problema de modelagem sísmica. Retirado de Sadahiro (2014).	98
4.7	Tempos de processamento numa implementação de modelagem sísmica em GPU. Retirado de Sadahiro (2014).	99
5.1	Modelo Marmousi de 125×383 amostras, a) verdadeiro e b) suavizado duas vezes.	104
5.2	Dado observado para o modelo Marmousi, 28 tiros.	105
5.3	10 iterações da FWI para 5 tiros, com e sem normalização dos dados calculados. a) Resultado sem normalização e sua função objetivo (c). b) Resultado com normalização e sua função objetivo (d).	106
5.4	10 iterações da FWI, 5 tiros, sem eliminar a onda direta. a) Modelo de velocidades após 10 iterações; b) Primeiro gradiente; c) Função objetivo. . .	109
5.5	10 iterações da FWI, 5 tiros, eliminando a onda direta. a) Modelo de velocidades após 10 iterações; b) Primeiro gradiente; c) Função objetivo.	110
5.6	Sismogramas para o primeiro tiro da primeira iteração da FWI. a) $\mathbf{d}_{\text{obs}}$ com onda direta. b) $\mathbf{d}_{\text{cal}}$ com onda direta. c) $\mathbf{d}_{\text{res}}$ (resíduo) com onda direta. d) $\mathbf{d}_{\text{obs}}$ sem onda direta. e) $\mathbf{d}_{\text{cal}}$ sem onda direta. f) $\mathbf{d}_{\text{res}}$ (resíduo) calculado com $\mathbf{d}_{\text{obs}}$ e $\mathbf{d}_{\text{cal}}$ sem onda direta.	111
5.7	Resultado de 10 iterações da FWI, 5 tiros, eliminando o artefato da onda direta pela zeragem das primeiras amostras do gradiente. a) Gradiente original da iteração 1, com artefato da onda direta. b) Gradiente da iteração 1 com as 10 primeiras amostras zeradas. c) Modelo de velocidades ao fim das iterações, com fatia do modelo verdadeiro colada no topo. d) Função objetivo.	112
5.8	Convergência da função objetivo para o método gradiente. Retirado de Nocedal e Wright (2006).	117
5.9	Resultados da FWI para os testes 1, 2 e 3 para o L-BFGS, operador DF. a) Teste 1: $\alpha = 1$ em todas as iterações (exceto nas de inicialização). b) Teste 3: α variável nas iterações 1 a 10, $\alpha = 1$ nas iterações 11 a 30 (de cada rodada). c) Teste 2: α variável em todas as iterações. d) Erro RMS para cada um dos testes. e) Valores de α ao longo das iterações para cada um dos testes. . . .	122

5.10	Resultados da FWI para os testes 1, 2 e 3 para o L-BFGS, operador PS. a) Teste 1: $\alpha = 1$ em todas as iterações (exceto nas de inicialização). b) Teste 3: α variável nas iterações 1 a 10, $\alpha = 1$ nas iterações 11 a 30 (de cada rodada). c) Teste 2: α variável em todas as iterações. d) Erro RMS para cada um dos testes. e) Valores de α ao longo das iterações para cada um dos testes.	123
5.11	Comparação entre métodos Conjugado Gradiente não linear e L-BFGS, 120 iterações da FWI multiescala. a) Resultado para CG. b) Resultado para L-BFGS. c) Erro RMS para os resultados em (a), (b).	125
5.12	FWI multiescala, 4 rodadas de 30 iterações cada, L-BFGS, para diferentes operadores de DF. a) DF de ordem 10. b) DF de ordem 12. c) DF de ordem 16. d) Erro RMS para cada um dos operadores.	127
5.13	Diferenças na inversão decorrentes do uso de diferentes operadores para $\mathbf{d}_{\text{obs}}$ e $\mathbf{d}_{\text{cal}}$. a) $\mathbf{d}_{\text{obs}}$ com operador DF12, $\mathbf{d}_{\text{cal}}$ com DF12. b) $\mathbf{d}_{\text{obs}}$ com operador DF12, $\mathbf{d}_{\text{cal}}$ com operador DF16. c) $\mathbf{d}_{\text{obs}}$ com operador PS, $\mathbf{d}_{\text{cal}}$ com operador DF12. d) Erro RMS para os resultados em (a), (b), (c).	128
5.14	a) 28 tiros modelados sem borda. b) Eventos de borda destacados: reflexão na borda superior (1); reflexões na borda da esquerda (2); reflexões na borda da direita (3).	131
5.15	a) Tiros com borda de tamanho 256, sem eventos de borda. b) Tiros sem borda absorvedora. c) Tiros com <i>taper</i> . d) Tiros com PML.	132
5.16	Resultados da FWI multiescala, borda absorvedora de tamanho 32. a) Modelo invertido para $\mathbf{d}_{\text{obs}}$ com <i>taper</i> , $\mathbf{d}_{\text{cal}}$ com <i>taper</i> . b) Modelo invertido para $\mathbf{d}_{\text{obs}}$ com PML, $\mathbf{d}_{\text{cal}}$ com PML. c) Modelo invertido para $\mathbf{d}_{\text{obs}}$ com PML, $\mathbf{d}_{\text{cal}}$ com <i>taper</i> . d) Erros RMS para os resultados (a), (b) e (c).	133
5.17	Resultados da FWI para um $\mathbf{d}_{\text{obs}}$ com "borda infinita". a) Modelo invertido para $\mathbf{d}_{\text{cal}}$ com <i>taper</i> . b) Modelo invertido para $\mathbf{d}_{\text{cal}}$ com PML. c) Erros RMS para os resultados (a) e (b).	134
5.18	Perfis de velocidades na posição $x = 6,6\text{km}$: verdadeiro, inicial e invertido. .	138
5.19	Modelos Marmousi. a) 751×2301 amostras. a) 250×767 amostras. a) 125×383 amostras.	139
5.20	a) Modelo Marmousi 250×767 inicial (entrada da FWI). b) Resultado da FWI utilizando operador REM-DF16. c) Erro RMS total ao longo das iterações. .	140
5.21	Modelo <i>Overthrust</i> 2-D. a) Modelo original. b) Modelo suavizado, campo inicial da FWI.	142
5.22	a) Perfil na posição 14 km do modelo, para FWI de 70 tiros. b) Perfil para FWI de 140 e 350 tiros. c) Erro RMS em cada iteração da FWI, para 70, 140 e 350 tiros.	143

5.23	FWIs do modelo <i>Overthrust</i> , para: a) 70 tiros, b) 140 tiros e c) 350 tiros. . .	144
5.24	Modelo falhas. a) Modelo original. b) Modelo suavizado, campo inicial da FWI. c) Resultado final da FWI.	146
5.25	a) Perfil de velocidades na posição 3 km do modelo. b) Erro RMS ao longo das iterações.	147
5.26	Funções objetivo normalizadas para cada uma das 5 rodadas da FWI (modelo Falhas). a) De 0 – 10, 12 Hz. b) De 0 – 20, 24 Hz. c) De 0 – 30, 36 Hz. d) De 0 – 40, 48 Hz. e) De 0 – 50, 82 Hz.	148

Introdução

A inversão da forma de onda completa (FWI - do inglês *full waveform inversion*) (Tarantola, 1984) vem se tornando a ferramenta padrão da indústria para estimar modelos de velocidades de alta resolução. Através da minimização entre dados sísmicos sintéticos e dados de campo (dados observados), a FWI é capaz de produzir um modelo fiel aos dados observados da subsuperfície, com alta resolução e a um nível de detalhe que não é alcançado por outras técnicas comumente utilizadas, como a tomografia de tempos de trânsito (Liu et al., 2012). A FWI depende de duas etapas fundamentais: resolução de um problema direto (modelagem), para modelar os dados sintéticos; e resolução de um problema inverso, para atualizar o modelo de velocidades. Devido à alta não-linearidade e não unicidade da solução, o principal desafio da FWI é evitar convergência a mínimos locais, e ao mesmo tempo manter o custo computacional a um nível aceitável (dos Santos e Pestana, 2015). Uma forma de evitar mínimos locais é utilizar uma abordagem multiescala do problema (Bunks et al., 1995); o problema de custo computacional pode ser reduzido através da paralelização do algoritmo em placas gráficas (GPU) (Wang et al., 2011, Mao et al., 2012).

A divisão em capítulos desta dissertação baseou-se nos principais desafios encontrados na implementação de um algoritmo de FWI: o método de modelagem (Capítulo 1), o método de migração (Capítulo 2) e o método de inversão (Capítulo 3). A migração faz parte do processo de inversão, mas do ponto de vista da sequência de implementação, vem depois da modelagem. A estratégia multiescala, para evitar mínimos locais, é também tratada no Capítulo 3. No Capítulo 4 é abordada a paralelização em placas gráficas, como forma de reduzir o custo computacional da FWI e permitir testes em tempo hábil. No Capítulo 5 é apresentada a metodologia, os principais problemas práticos encontrados na implementação da FWI em placas gráficas, e resultados para três modelos de velocidades.

A modelagem sísmica, no contexto acústico, consiste em obter o campo de pressões correspondente ao campo de velocidades, que é perturbado por algum tipo de fonte. O campo é obtido em todos os pontos no espaço, para cada instante de um intervalo de tempo pré-determinado. O processo iterativo de extrapolar o campo de ondas por soluções numéricas da equação completa da onda acústica é computacionalmente intensivo, pode demandar muita

memória computacional, e apresentar problemas numéricos conhecidos como dispersão e instabilidade. Uma ferramenta muito útil para mitigar esses problemas de memória, dispersão e instabilidade, abordada nessa dissertação, é o método de expansão rápida (REM, do inglês *rapid expansion method*) (Pestana e Stoffa, 2010). Como mostrado no Capítulo 1, o processo de modelagem consome a maior parte do tempo de computação tanto dos algoritmos de migração, quanto de inversão; seu entendimento e implementação são fundamentais para executar a FWI.

O método de imageamento utilizado foi a migração reversa no tempo (RTM, do inglês *reverse time migration*, Baysal et al., 1983; Whitmore, 1983; McMechan, 1983; Loewenthal e Mufti, 1983). A RTM, para um tiro, consiste em: modelagem sintética do campo da fonte; depropagação do campo dos receptores, injetado na superfície; e condição de imagem, que é a correlação cruzada no *lag* zero do campo da fonte e campo dos receptores, em todos os tempos. Tanto a propagação do campo da fonte quanto a depropagação do campo dos receptores são consideradas modelagens sísmicas, a diferença é que no primeiro caso a fonte é uma *wavelet* injetada em apenas um ponto, e no segundo caso temos várias fontes, uma em cada receptor, cuja amplitude injetada é correspondente à amplitude no sismograma do tiro. O custo computacional da condição de imagem é muito pequeno, se comparado à propagação e depropagação dos campos. Assim, do ponto de vista computacional, a RTM pode ser entendida como duas modelagens sísmicas.

A inversão da forma da onda completa (FWI - do inglês *full-waveform Inversion*) é um processo de inversão que visa obter o campo de velocidades sísmicas (nesta dissertação, velocidades acústicas) usando como dado de entrada todas as informações contidas no sismograma - tempos de chegada, amplitude e fase - e um modelo de velocidades inicial. Na formulação mínimos quadrados se deseja minimizar uma função objetivo, que mede o resíduo entre dados observados (sismograma) e dados calculados (modelagem sintética do sismograma utilizando o campo de velocidades invertido). Os métodos de busca desse mínimo podem ser locais ou globais. Métodos globais ainda são inviáveis à FWI devido ao grande número de parâmetros a ser invertido. Dentre os métodos locais, se destacam os métodos iterativos, que consistem em linearizar o problema em torno do modelo corrente e encontrar, no espaço de modelos dessa linearização local, aquele que corresponda ao menor erro entre dado calculado e dado observado (Virieux e Operto, 2009). Esse modelo atualizado será o modelo corrente da próxima iteração. O processo é então repetido até que se chegue à convergência.

A linha que conecta o modelo corrente ao modelo atualizado é a direção de busca, e a distância entre os dois é o passo. Dentre os métodos para encontrar essa direção de busca

e o passo se destacam: métodos gradiente (*steepest-descent*), de Newton, Gauss-Newton e quasi-Newton (Virieux e Operto, 2009). Dentre eles, vale salientar que o custo do método de Newton também é proibitivo, dado o grande número de parâmetros da FWI (capítulo 3). Todos esses métodos fazem uso do gradiente da função objetivo, como primeira estimativa da direção de busca. Como o cálculo do gradiente pela matriz sensibilidade (ou matriz de Fréchet) tem custo computacional proibitivo à FWI, ele é calculado pelo método adjunto (Tarantola, 1984). É mostrado no capítulo 3 que o cálculo do gradiente pelo método adjunto corresponde à RTM do resíduo, a cada iteração. Assim, computacionalmente, o cálculo do gradiente corresponde à uma RTM, que como visto anteriormente, tem o custo de duas modelagens. Para o cálculo do passo utilizamos a abordagem de Pica et al. (1990), cujo custo computacional é de uma modelagem.

Nesta dissertação comparamos três métodos de inversão: dois *steepest-descent* (gradiente e conjugado-gradiente) e um quasi-Newton (L-BFGS) (Nocedal e Wright, 2006). Todos eles custam apenas as modelagens intrínsecas ao cálculo do gradiente, e uma referente ao passo. Apesar do conjugado-gradiente e do L-BFGS fazerem uso de gradientes anteriores (Capítulo 3), que tem um custo de memória, nesse problema 2D essa etapa é irrisória, se comparada ao tempo de modelagem. Assim, pode-se resumir o custo computacional da FWI a três modelagens por iteração.

Dado o exposto, basta analisar o custo computacional da modelagem sísmica utilizada para obter o custo computacional da FWI. Essa foi a abordagem utilizada por Yang et al. (2015). Nessa dissertação fez-se uso de algoritmos em placas gráficas (*Graphics Processing Unit*, ou GPU), paralelizando o problema e reduzindo o custo computacional. No Capítulo 4 apresentamos a arquitetura da GPU, a metodologia de uso e as vantagens e desvantagens em relação à computação em série. No Capítulo 5 são comparados os custos de uma modelagem em série e uma paralelizada em GPU; o custo da FWI é então obtido imediatamente, multiplicando pelo número de modelagens por iteração e pelo número total de iterações. Também é apresentado nesse capítulo os custos de memória da GPU/CPU na execução de cada algoritmo (conjugado-gradiente não linear e L-BFGS). A metodologia apresentada se resumiu a:

- reduzir o máximo possível o modelo de velocidades Marmousi inicial (decimando o original), para acelerar os testes;
- selecionar o algoritmo de modelagem mais adequado;
- selecionar o melhor algoritmo de inversão;
- executar a inversão no modelo Marmousi original, e em modelos diferentes (*Overthrust*,

falhas).

O uso de GPUs se mostrou altamente eficiente, pois permitiu o teste e posterior seleção dos diversos algoritmos de modelagem e inversão em poucos minutos. Definiu-se como modelo teste o campo de velocidades Marmousi de dimensões $n_z = 125$ e $n_x = 383$ amostras, espaçamentos $\Delta z = 24$ m e $\Delta x = 24$ m, que aqui será denominado modelo Marmousi-Teste. O dado observado base desse modelo tem 28 tiros, cobrindo todo o modelo, com 101 receptores num arranjo *split-spread* em cada tiro; cada tiro é espaçado de 10 amostras (240 m) e cada receptor é separado de 1 amostra (24 m).

Para o modelo Marmousi-Teste, testes pequenos (menos de 10 tiros, apenas uma rodada multiescala com menos de 10 iterações) custaram em média 5 minutos, e testes completos (28 tiros cobrindo todo o modelo, 4 rodadas de 30 iterações cada) duraram em média 45 minutos. Os resultados para os modelos testados foram satisfatórios e coerentes com os resultados da literatura (Yang et al., 2015; dos Santos, 2013).

1

Modelagem Sísmica

A modelagem sísmica é o processo de reproduzir, sinteticamente, a propagação de uma onda na terra, decorrente duma perturbação mecânica causada por uma fonte sísmica, localizada num ponto da subsuperfície da Terra.

Para realizar a modelagem é preciso:

- Um campo de propriedades físicas que represente as propriedades e contrastes da Terra. Exemplo: campo de velocidades acústicas (Figura 1.1(a));
- Uma representação matemática de uma fonte sísmica (Figura 1.2), a ser inserida num ponto da Terra de forma a simular a perturbação mecânica. Normalmente, esse ponto se encontra na superfície;
- Uma relação física que descreva a propagação da onda resultante da fonte. Exemplo: equação da onda acústica.
- Uma solução numérica dessa relação física, que permita uma implementação computacional estável. Exemplos: método de diferenças finitas, método pseudo-espectral, método de expansão rápida.

Cada um dos itens acima será discutido nas seções seguintes.

Na formulação acústica da modelagem sísmica, realizada neste trabalho, a subsuperfície é representada como uma malha 2D, onde cada ponto (z, x) desta malha assume um valor de pressão $P(z, x, t)$ e velocidade sísmica $v(z, x)$. Inicialmente (tempo zero), $P(x, z, t) = 0$ em toda a malha, exceto na posição na qual se introduz o pulso sísmico $w(t)$; em outras

palavras, sendo (z_s, x_s) a posição espacial do pulso sísmico na malha, $P(z_s, x_s, t) = s(t)$. O pulso ideal seria uma delta de Dirac, a qual possui amplitude apenas quando $t = 0$ s. No caso de modelar um pulso real, este tem uma pequena duração temporal, por exemplo, se for usado um pulso (ou *wavelet*) do tipo Ricker, como aproximação da fonte, com frequência máxima 35 Hz, $t_{pulso} = 1/35 \approx 28$ ms, o que significa que o tempo para que toda a forma do pulso seja inserida no modelo é de 28 ms. Ou seja, durante os primeiros 28 ms de modelagem, o pulso é tanto inserido no modelo quanto propagado por ele; após 28 ms há apenas propagação.

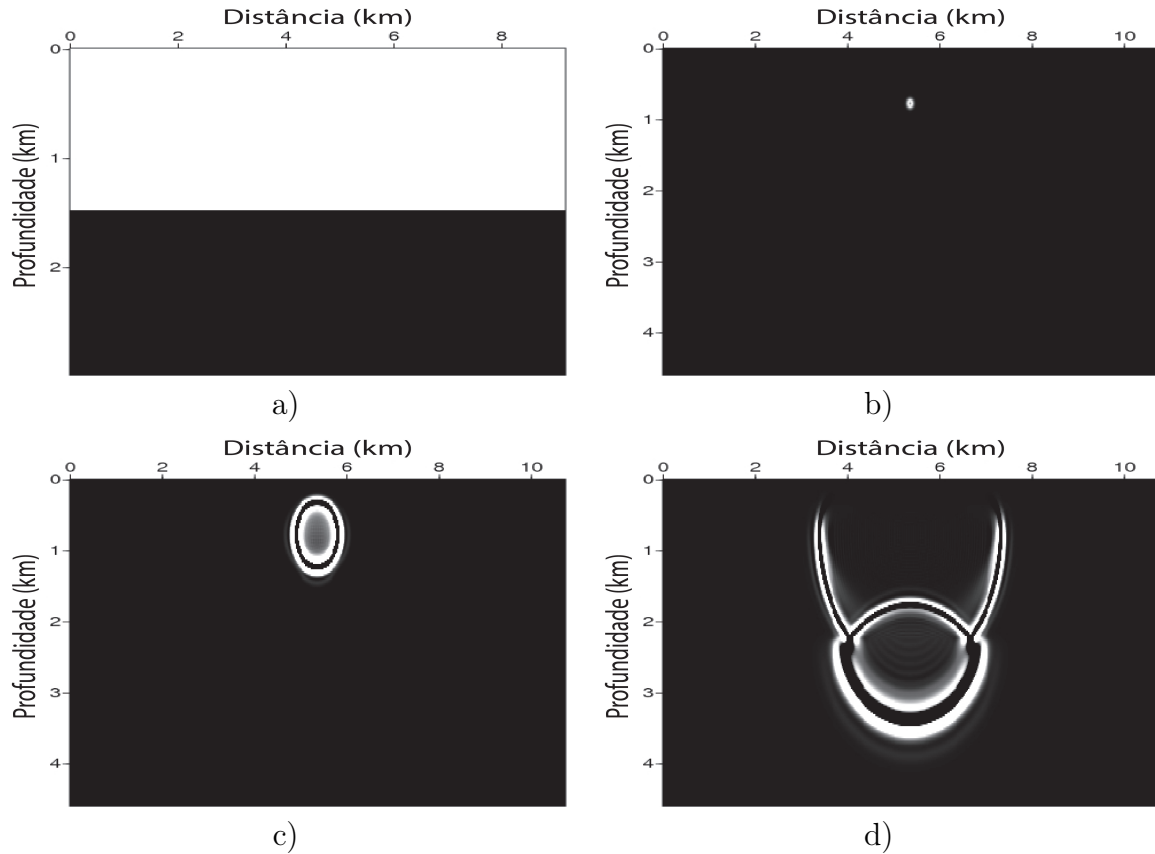


Figura 1.1: a) Campo de velocidades com duas camadas, de 1500 m/s e 3500 m/s. Propagação da onda, após: b) 2 ms, c) 10 ms e d) 35 ms.

A informação proveniente da modelagem acústica realizada desta maneira pode ser salva de diversas formas, dependendo do foco de estudo ou do método que se quer implementar. Na modelagem de um tiro sísmico, por exemplo, só são salvas informações do campo na superfície, ou seja, a matriz $P(x, z = 0, t)$ (Figura 1.3-a). No caso de um Perfil Sísmico Vertical (PSV, ou VSP - do inglês *Vertical Seismic Profile*), seria salva a matriz $P(z, x = x_{VSP}, t)$. Uma terceira forma é armazenar todo o campo de pressão em todos os tempos, ou seja, a matriz 3D completa $P(x, z, t)$, denominados instantâneos da propagação da onda ou *snapshots*.

A modelagem sísmica é o cerne de toda essa dissertação. Como será visto nos próximos capítulos, a migração reversa no tempo, que tem por objetivo gerar uma imagem da subsuperfície em termos da refletividade sísmica, e consiste de duas modelagens sísmicas e uma condição de imagem. A inversão da forma de onda completa, que tem por objetivo estimar um modelo de propriedades físicas da subsuperfície (velocidades da onda-P, no caso acústico), em sua formulação iterativa, tem por etapa mais importante a migração do resíduo entre dado observado e dado calculado, a cada iteração.

1.1 Equação da onda acústica

Estritamente falando, meios acústicos são fluidos (líquidos ou gases) nos quais o módulo de cisalhamento (ou em inglês, *shear modulus*) é zero. Nesses meios pode-se definir ondas de pressão que obedecem à equação da onda escalar, e se propagam à velocidade de propagação acústica. Essas ondas são sempre longitudinais. Anisotropia também é um fenômeno raro em meios acústicos (Verschuur e Gisolf, 2010).

Por outro lado, em meios elásticos, não há como definir uma pressão escalar, e a propagação depende de um tensor de *stress*. O meio anisotrópico mais geral é descrito por 21 parâmetros independentes de rigidez elástica. Um meio elástico totalmente isotrópico tem 2 parâmetros de rigidez: o módulo de incompressibilidade (ou em inglês, *bulk modulus*), que é o fator de proporcionalidade entre a média dos elementos de *stress* da diagonal do tensor e a mudança relativa de volume; e o módulo de cisalhamento, que é o fator de proporcionalidade entre o *stress* cisalhante e o ângulo de cisalhamento. Num meio elástico temos ondas longitudinais e transversais. Durante a reflexão ou espalhamento dessas ondas, conversões de longitudinal para transversal ou vice-versa podem ocorrer (Verschuur e Gisolf, 2010).

Apesar da equação completa da onda elástica ser mais complicada (e uma melhor representação da resposta da terra a uma excitação mecânica) que a equação da onda acústica, a solução da equação elástica completa pode ser decomposta em uma onda longitudinal e duas ondas transversais, que individualmente obedecem equações de onda do tipo acústico, embora com diferentes velocidades de propagação, dado o modo de polarização (P, SH, SV). Esta é a razão pela qual todo o maquinário de imageamento acústico pode ser aplicado a problemas de ondas elásticas, como por exemplo a propagação de ondas na terra, uma vez que se decompõe o campo de onda em ondas longitudinais e ondas transversais (Verschuur e Gisolf, 2010).

A dedução da equação da onda acústica parte da análise da perturbação de pressão num elemento infinitesimal de volume, ou de área, no caso 2D. Como a presente dissertação

trata do caso 2D, apresentamos as equações básicas que fundamentam a equação dessa onda acústica. Uma dedução mais detalhada pode ser encontrada em Verschuur e Gisolf, 2010.

Como dito anteriormente, num meio acústico, aqui suposto, o módulo de cisalhamento é nulo, e a única constante elástica é o módulo de incompressibilidade (*bulk modulus*) K . A lei de Hooke dada por:

$$P(z, x, t) = -K(z, x) \nabla \cdot \mathbf{u}(z, x, t) \quad (1.1)$$

mostra que a pressão P e a velocidade de partícula $\mathbf{u}$ tem uma relação linear, cuja constante de proporcionalidade é o módulo de incompressibilidade K . Incluindo um termo fonte $q(z, x, t)$ adimensional (área injetada por unidade de área), temos:

$$P(z, x, t) = -K(z, x) \nabla \cdot \mathbf{u}(z, x, t) - q(z, x, t). \quad (1.2)$$

A 2ª lei de Newton aplicada ao elemento de área perturbado, após algumas considerações algébricas, resulta em:

$$-\nabla P(z, x, t) = \rho(z, x) \frac{\partial^2 \mathbf{u}(z, x, t)}{\partial t^2}, \quad (1.3)$$

onde ρ é a densidade do meio. Adicionando uma força externa à equação, que deve ter dimensão de força por unidade de área, temos:

$$\mathbf{f}(z, x, t) - \nabla P(z, x, t) = \rho(z, x) \frac{\partial^2 \mathbf{u}(z, x, t)}{\partial t^2}, \quad (1.4)$$

Podemos então eliminar $\mathbf{u}$ das equações 1.2 e 1.4, dividindo 1.4 por ρ , e tomando seu divergente ($\nabla \cdot$)¹; diferenciando 1.2 duas vezes no tempo ($\partial^2/\partial t^2$) e subtraindo as equações de forma a eliminar $\mathbf{u}$, resulta na equação da onda acústica para P :

$$\nabla \cdot \left[\frac{1}{\rho(z, x, t)} \nabla P(z, x, t) \right] - \frac{1}{K(z, x)} \frac{\partial^2 P(z, x, t)}{\partial t^2} = -s(z, x, t), \quad (1.5)$$

onde

$$s(z, x, t) = \frac{\partial^2 q(z, x, t)}{\partial t^2} - \nabla \cdot \left[\frac{\mathbf{f}(z, x, t)}{\rho(z, x)} \right]. \quad (1.6)$$

¹Onde $\nabla \cdot$ é o operador divergente: $\nabla \cdot P = \partial P / \partial z + \partial P / \partial x$, em duas dimensões.

A equação 1.6, que expressa a fonte s de perturbação de pressão em termos das fontes q e $\mathbf{f}$, das leis de Hooke e Newton, tem um papel teórico na solução da equação da onda para meios não homogêneos (Verschuur e Gisolf, 2010). Nesta dissertação essas fontes não serão mais abordadas, e $s(z, x, t)$ será expressada em termos duma *wavelet* e de sua posição num modelo 2D, como veremos a seguir.

Assumindo um meio de densidade constante, e sabendo que a velocidade de propagação duma onda longitudinal de pressão (onda P) é dada por $c_0(z, x) = \sqrt{K(z, x)/\rho(z, x)}$, obtemos a equação da onda acústica 2D para o caso de densidade constante:

$$\frac{1}{c_0^2(z, x)} \frac{\partial^2 P(z, x, t)}{\partial t^2} = \nabla^2 P(z, x, t) - s(z, x, t), \quad (1.7)$$

onde $\nabla^2 = \partial^2/\partial z^2 + \partial^2/\partial x^2$ é o operador Laplaciano em 2D.

A equação 1.7 é conhecida como equação da onda acústica 2D para o caso de densidade constante. De forma a tornar a nomenclatura mais enxuta, a partir de então utiliza-se o termo "equação da onda acústica" como substitutivo de "equação da onda acústica 2D para o caso de densidade constante".

Um conceito muito importante à teoria do imageamento acústico é o campo de onda de uma fonte pontual. Esse será o único tipo de fonte utilizado para esta dissertação. A equação da onda acústica, na presença de uma fonte pontual localizada em (z_s, x_s) , é dada por:

$$\frac{1}{c_0^2(z, x)} \frac{\partial^2 P(z, x, t)}{\partial t^2} = \nabla^2 P(z, x, t) - w(t)\delta(z - z_s, x - x_s), \quad (1.8)$$

onde $w(t)$ é a *wavelet* fonte. Um tipo de *wavelet* muito usual na literatura, e também utilizada para esta dissertação, é a Ricker, definida matematicamente como:

$$w(t) = (1 - 2t_r^2)e^{-t_r^2}, \quad (1.9)$$

onde t_r é dado por:

$$t_r = \pi f_{peak} \left(t - \frac{1}{f_{peak}} \right). \quad (1.10)$$

Analisando a transformada de Fourier da *wavelet* é possível entender os conceitos de frequência de pico e frequência máxima. A frequência de pico f_{peak} é o componente monofrequência de maior amplitude da *wavelet*; na Figura 1.2-b, é a frequência de 15 Hz. A frequência máxima f_{max} é o último componente monofrequência que possui correlação (amplitude relevante) com a *wavelet*; no exemplo do espectro de amplitude na Figura 1.2-b, corresponde à frequência de aproximadamente 48 Hz. A fórmula analítica que descreve a Ricker é construída com base na frequência de pico escolhida; a frequência máxima é então determinada por consequência, através da transformada de Fourier dessa Ricker. Na literatura (Dablain, 1986) se recomenda como regra prática para descobrir a frequência máxima f_{max} da Ricker a relação $f_{max} \approx 2f_{peak}$. Todavia, Wang (2015) mostrou que f_{max} não corresponde a $2f_{peak}$, como mostra a Figura 1.2: para $f_{peak} = 15$ Hz. Neste caso, há frequências de quase 50 Hz, ou seja, o fator é maior que 2.

Argumenta-se que a regra $f_{max} \approx 2f_{peak}$ é válida também devido ao fato das frequências acima de $2f_{peak}$ terem baixas amplitudes, e serem quase que totalmente atenuadas no processo de modelagem (devido às perdas de energia por divergência esférica). A Figura 1.3 mostra que a regra prática é aproximadamente válida. Todavia, de forma a garantir a análise de dispersão (tratada nas seções seguintes), se adotou o critério $f_{max} \approx 2.3f_{peak}$ para as modelagens desta dissertação. Uma outra prática na modelagem, visando reduzir a dispersão, é a de fazer uma suavização gaussiana no ponto da fonte e nos pontos próximos, ao inserir a *wavelet*.

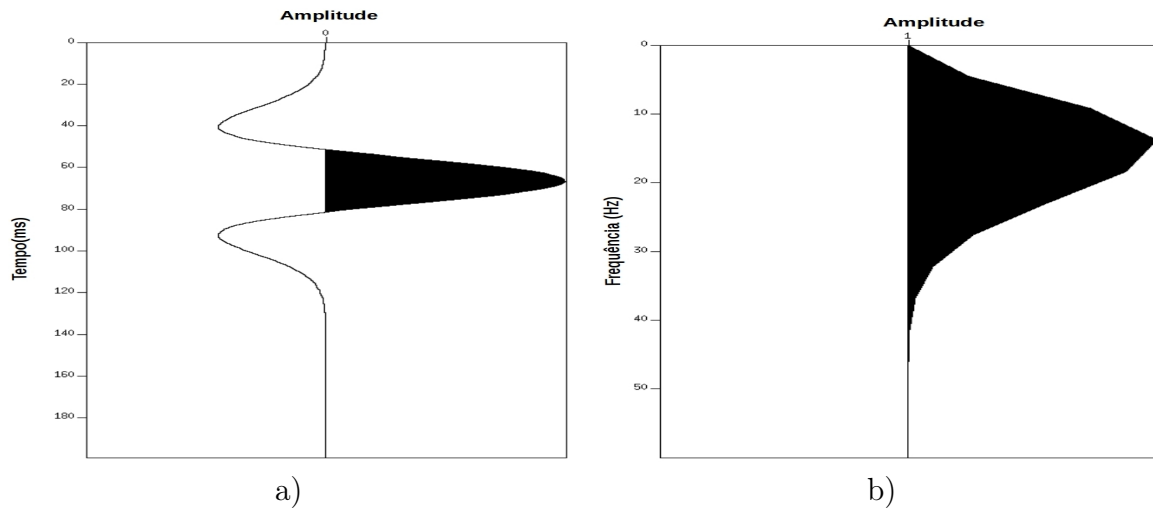


Figura 1.2: a) *Wavelet* Ricker com $f_{peak} = 15\text{Hz}$. b) Espectro de amplitudes. f_{max} ultrapassa 30Hz ($2f_{peak}$), chegando próximo de 50Hz , porém as altas frequências tem baixa amplitude.

Para executar a modelagem sísmica (acústica) é necessário resolver a equação 1.7. A

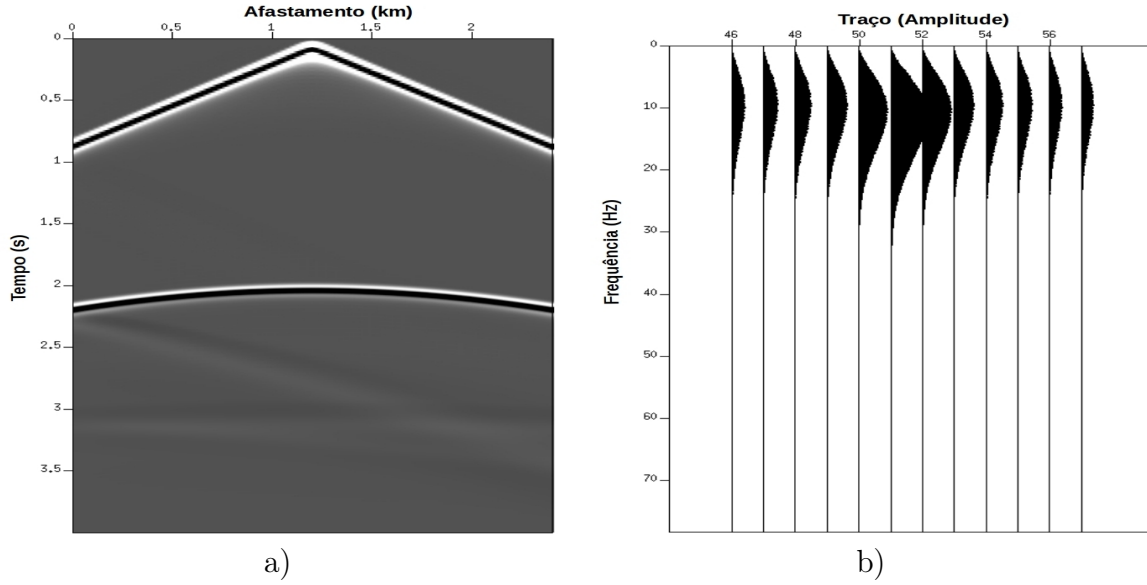


Figura 1.3: Modelagem de um tiro, num modelo de duas camadas, usando uma Ricker de $f_{peak} = 15$ Hz como fonte. a) Sismograma para 101 geofones, split-spread. b) Espectro de amplitude dos traços mais próximos à fonte (traço 51). A frequência máxima ultrapassa $2f_{peak} = 30$ Hz.

seguir, apresentamos algumas técnicas numéricas de solução da equação da onda acústica: diferenças finitas, método pseudo-espectral e método de expansão rápida.

1.2 Soluções numéricas da equação da onda acústica

1.2.1 Diferenças finitas em tempo e espaço

Introduzindo o método de maneira heurística, sabe-se que o conceito de derivada de uma função num ponto está associado à taxa de variação da função nesse ponto. A reta tangente que representa a derivada no ponto pode ser aproximada pela reta secante que conecta pontos vizinhos, antes e depois do ponto alvo. Considerando espaçamentos fixos Δz dos pontos vizinhos ao ponto alvo, temos uma aproximação de diferenças finitas para derivada primeira:

$$\frac{\partial P(z)}{\partial z} \approx \frac{P(z + \Delta z) - P(z - \Delta z)}{\Delta z}. \quad (1.11)$$

Uma dedução mais estruturada da aproximação de derivadas por diferenças finitas pode ser feita através do truncamento de séries de Taylor (Araujo, 2009). O objetivo é achar uma aproximação para a derivada segunda em z de $P(z, x, t)$. Omitimos as variáveis (x, t) de forma a tornar a notação mais enxuta. Partimos então da série de Taylor para $P(z + \Delta z)$, aproximada em torno do ponto $P(z)$:

$$P(z + \Delta z) = P(z) + \frac{\partial P(z)}{\partial z} \Delta z + \frac{\partial^2 P(z)}{\partial z^2} \frac{\Delta z^2}{2!} + \frac{\partial^3 P(z)}{\partial z^3} \frac{\Delta z^3}{3!} + \dots, \quad (1.12)$$

e da série de Taylor para $P(z - \Delta z)$, aproximada em torno do ponto $P(z)$:

$$P(z - \Delta z) = P(z) - \frac{\partial P(z)}{\partial z} \Delta z + \frac{\partial^2 P(z)}{\partial z^2} \frac{\Delta z^2}{2!} - \frac{\partial^3 P(z)}{\partial z^3} \frac{\Delta z^3}{3!} + \dots \quad (1.13)$$

Somando as equações 1.12 e 1.13, temos:

$$P(z + \Delta z) + P(z - \Delta z) = 2P(z) + 2\frac{\partial^2 P(z)}{\partial z^2} \frac{\Delta z^2}{2!} + 2\frac{\partial^4 P(z)}{\partial z^4} \frac{\Delta z^4}{4!} + \dots \quad (1.14)$$

Desprezando os termos de ordem superior a 2 (ou maiores que $O(\Delta z^2)$) a derivada segunda em z pode ser aproximada por:

$$\frac{\partial^2 P(z)}{\partial z^2} \approx \frac{P(z + \Delta z) - 2P(z) + P(z - \Delta z)}{\Delta z^2}, \quad (1.15)$$

que é a aproximação de diferenças finitas centrada de segunda ordem, uma vez que foram desprezados os termos de ordem superior a $O(\Delta z^2)$. A aproximação é dita centrada visto que utilizam-se os pontos vizinhos de forma que $P(z)$ está no centro. Também é possível encontrar aproximações utilizando apenas pontos anteriores $P(z - \Delta z), P(z - 2\Delta z), \dots$, tomando as correspondentes expansões de Taylor, obtendo uma aproximação de diferenças regressivas; ou analogamente utilizar apenas $P(z + \Delta z), P(z + 2\Delta z), \dots$ e obter uma aproximação de diferenças progressivas; todavia, a diferença centrada apresenta uma acurácia superior às diferenças regressivas/progressivas (Burden e Faires, 2010).

Apesar da dedução em 1.15 ter sido feita para a variável z , ela é também válida para as variáveis x e t . Podemos também encontrar a aproximação de quarta ordem para a derivada segunda por um procedimento similar, porém desprezando agora apenas os termos acima de $O(\Delta z^4)$ (Araujo, 2009),

$$\frac{\partial^2 P(z)}{\partial z^2} \approx \frac{1}{12\Delta z^2} [-P(z - 2\Delta z) + 16P(z - \Delta z) - 30P(z) + 16P(z + \Delta z) - P(z + 2\Delta z)]. \quad (1.16)$$

A aproximação de quarta ordem é mais precisa que a de segunda ordem, porém, como mostra a equação 1.16, são necessários mais pontos vizinhos para efetuar o cálculo. A aproximação de segunda ordem utiliza 3 pontos, a de quarta ordem utiliza 5; como regra geral, dada

uma aproximação de diferenças finitas de ordem n , são necessários $n + 1$ pontos para sua computação (Burden e Faires, 2010).

Uma vez definidas as derivadas em função dos pontos próximos espaçados por intervalos regulares $(\Delta z, \Delta x, \Delta t)$, a transição para o problema discreto (inerente à implementação computacional) é imediata. Definindo a notação de discretização, temos:

$$P(z, x, t) = P(i\Delta z, j\Delta x, n\Delta t) = P_{i,j}^n, \quad (1.17)$$

onde $i \in \{0, 1, \dots, nz\}$, $j \in \{0, 1, \dots, nx\}$, $n \in \{0, 1, \dots, nt\}$, sendo $(\Delta z, \Delta x)$ os intervalos de amostragem espacial, em metros, (nz, nx) o número total de pontos na dimensão vertical e horizontal, adimensionais (por exemplo, num modelo de 3 km verticais podemos ter $\Delta z = 24.0$ m, $nz = 125$ pontos); Δt o intervalo de amostragem temporal, em segundos, e nt o número máximo de pontos no tempo, adimensional (por exemplo, para um tempo total de 4s podemos ter $\Delta t = 0,001$ s e $nt = 4000$).

De maneira geral, em modelagem sísmica, se utiliza quase sempre aproximação de segunda ordem para $\partial^2/\partial t^2$, pois o campo de pressão $P(z, x, t)$ representa uma matriz de tamanho nz por $n x$, que deve ser armazenada na memória para cálculo das diferenças. O método de segunda ordem no tempo exige armazenamento de duas matrizes, $P_{i,j}^{n-1}$ e $P_{i,j}^{n+1}$, para calcular $P_{i,j}^{n+1}$. Já as aproximações das derivadas espaciais não tem esse problema, pois se coletam os pontos vizinhos ao ponto alvo na mesma malha espacial. Assim, pode-se usar aproximações de oitava, décima, décima segunda ordem, etc, sem que se esgote a memória computacional. Nesta dissertação, utilizou-se aproximação de 2ª ordem para a derivada temporal e de 14ª ordem para as derivadas espaciais.

Discretizando a equação da onda acústica para diferenças finitas de 2ª ordem no tempo, e substituindo o Laplaciano ∇^2 por um termo D^k , que representa os operadores de diferenças finitas espaciais de ordem k para z e x , temos a equação utilizada na extrapolação do campo de onda, pelo método de diferenças finitas:

$$\frac{P_{i,j}^{n+1} - 2P_{i,j}^n + P_{i,j}^{n-1}}{\Delta t^2} = c_0^2 D^k(P_{i,j}^n) + f^n(z_s, x_s).$$

$$P_{i,j}^{n+1} = 2P_{i,j}^n - P_{i,j}^{n-1} + c_0^2 \Delta t^2 D^k(P_{i,j}^n) + f^n(z_s, x_s). \quad (1.18)$$

A equação 1.18 pode ser interpretada da seguinte maneira (Figura 1.1): no instante inicial, os campos $P_{i,j}^n$ e $P_{i,j}^{n-1}$ são nulos. O campo $P_{i,j}^{n+1}$ tem um único valor, injetado na

posição da fonte (z_s, x_s) . No instante seguinte, $P_{i,j}^n$ se torna $P_{i,j}^{n+1}$, $P_{i,j}^{n-1}$ se torna $P_{i,j}^n$, e $P_{i,j}^{n+1}$ será calculado. Nesse caso a diferença temporal $2P_{i,j}^n - P_{i,j}^{n-1}$ é não-nula, e as diferenças espaciais $D_k(P_{i,j}^n)$, próximas ao ponto (z_s, x_s) também são não-nulas. O processo então se repete até que se atinjam nt iterações temporais, gerando a propagação da onda devido à injeção duma fonte pontual.

Estabilidade numérica

Como visto anteriormente, a extrapolação do campo de onda, fundamentada na equação da onda acústica com densidade constante, é feita através do método de diferenças finitas, para aproximar as derivadas espaciais e temporal. A aproximação de uma equação diferencial (envolvendo quantidades infinitesimais $\partial/\partial t, \partial/\partial z, \partial/\partial x$) por diferenças finitas (envolvendo quantidades finitas $\Delta t, \Delta z, \Delta x$) pode gerar resultados irreais, que não correspondem ao comportamento descrito pela equação da onda acústica. Um desses resultados irreais é a instabilidade numérica, que ocorre quando as amplitudes crescem em demasia na extrapolação, gerando um campo de onda com *NaN* e/ou *Inf* (*Not a Number*, decorrente de tentativa de divisão por zero, e *Infinity*, quando o resultado duma operação gera um número maior que a precisão máxima do processador), o que impede a visualização da propagação.

Uma maneira aproximada de determinar se os espaçamentos $\Delta t, \Delta z, \Delta x$ escolhidos fazem a solução estável foi desenvolvida heurísticamente por Lines et al. (1999):

$$\frac{c_0 \Delta t}{h} \leq \sqrt{\frac{a_1}{a_2}}, \quad (1.19)$$

sendo c_0 a velocidade sísmica do meio, substituída normalmente por $\max(c_0)$ para garantir estabilidade em todos os pontos, h o intervalo espacial máximo do grid $\max(\Delta z, \Delta x)$, a_1 a soma dos pesos do operador de diferenças finitas temporal e a_2 a soma dos pesos dos operadores de diferenças finitas espaciais. Essa soma é definida por (modificado de Lines et al., 1999):

$$a = \sum_{m=-M}^M |w_m|, \quad (1.20)$$

Onde M é a metade da ordem do operador escolhido. Por exemplo, para o operador de segunda ordem $\mathbf{w} = (1, -2, 1)$, resultando $a = 4$. Como citado anteriormente, a aproximação da derivada temporal $\partial/\partial t^2$ é quase sempre a de segunda ordem (inclusive nesta dissertação). Daí, se modifica a equação 1.19, fazendo $a_1 = 4$, de forma que:

$$\frac{c_0 \Delta t}{h} \leq \frac{2}{\sqrt{a_2}}. \quad (1.21)$$

Rearrmando e isolando Δt , temos a condição de estabilidade, dependente da velocidade máxima do modelo, espaçamentos espaciais, e tamanho do operador de diferenças finitas:

$$\Delta t \leq \frac{2 \max(\Delta z, \Delta x)}{\max(c_0) \sqrt{a_2}}. \quad (1.22)$$

Como a_2 envolve dois operadores espaciais (em x e z), sua equação é (modificado de Lines et al., 1999):

$$a_2 = \sum_{m=-M}^M |w_{zm}| + |w_{xm}|. \quad (1.23)$$

Por exemplo, para um modelo Marmousi de dimensões $nz = 125$ e $nx = 383$ amostras, $dz = 24$ m, $dx = 24$ m e $\max(c_0) = 5500$ m/s, usando um operador espacial de segunda ordem $(1; -2; 1)$, temos $a_2 = 8$, resultando em $\Delta t \leq 0,0036$ s. Acima desse valor, a solução é instável. Como garantia duma boa propagação, normalmente se escolhe um valor abaixo do limite; nesse caso, $\Delta t = 0,003$ s garante uma solução estável.

Dispersão numérica

A dispersão é um dos efeitos numéricos decorrentes da aproximação da equação da onda acústica por um operador discreto, como o de diferenças finitas. Esse artefato numérico faz com que a velocidade de fase de cada harmônico de P seja função do intervalo de discretização. Seguimos a abordagem de Dablain (1986) que, por motivos de clareza, analisa a equação 1D da onda acústica:

$$\frac{\partial^2 P}{\partial t^2} = c_0^2 \frac{\partial^2 P}{\partial x^2}, \quad (1.24)$$

onde P é o campo de pressões, t o tempo, x o espaço e c_0 a velocidade acústica do meio.

Existem dois tipos de dispersão numérica: temporal, onde o erro numérico decorre da aproximação discreta de $\frac{\partial^2 P}{\partial t^2}$, e espacial, que decorre da aproximação de $\frac{\partial^2 P}{\partial x^2}$.

Analisando somente a dispersão temporal, para uma aproximação por diferenças finitas de 2ª ordem no tempo, temos:

$$\frac{P^{n+1} - 2P^n + P^{n-1}}{\Delta t^2} = c_0^2 \frac{\partial^2 P}{\partial x^2}, \quad (1.25)$$

onde

$$P^n = P(x, n\Delta t).$$

Lembrando que o campo P , em termos da análise de Fourier, pode ser visto como uma superposição de harmônicos mono-frequência, onde a frequência de cada harmônico tem seu correspondente no espectro de P , a análise do harmônico de P de frequência máxima, solução da equação da onda 1.24, se estende para o campo P (pelo princípio da superposição). Assim, substituindo o harmônico $e^{i(k_x x + \omega t)}$ em 1.25, e fazendo uso da identidade de Euler:

$$\frac{2[\cos(\omega\Delta t) - 1]}{\Delta t^2} = -c_0^2 k_x^2. \quad (1.26)$$

A velocidade de fase é, por definição:

$$c = \frac{\omega}{k_x}. \quad (1.27)$$

Num meio livre de dispersão (tanto física quanto numérica), a velocidade de fase é sempre a velocidade de propagação do meio, ou seja, $\frac{c}{c_0} = 1$. Todavia, devido à aproximação temporal de DF no tempo, temos:

$$\frac{c^2}{c_0^2} = \frac{-(\omega^2 \Delta t^2)}{2[\cos(\omega\Delta t) - 1]}. \quad (1.28)$$

Dablain (1986) sugere então, para um entendimento intuitivo da equação 1.28, fazer uso da expansão de Taylor do cosseno, supondo $\omega\Delta t \approx 0$ (Δt pequeno), de forma que:

$$\frac{c^2}{c_0^2} \approx 1 + \frac{(\omega^2 \Delta t^2)}{12}. \quad (1.29)$$

A equação 1.29 indica que conforme Δt aumenta ou a frequência aumente, a tendência é que partes da wavelet (fonte do campo P) fiquem avançadas em relação à velocidade de fase verdadeira. Na prática, no caso de dispersão numérica temporal aparecem artefatos à *frente* da frente de onda principal, como mostra a Figura 1.4-b.

Similarmente, analisamos na equação 1.24 o efeito da dispersão decorrente da aproximação da derivada espacial:

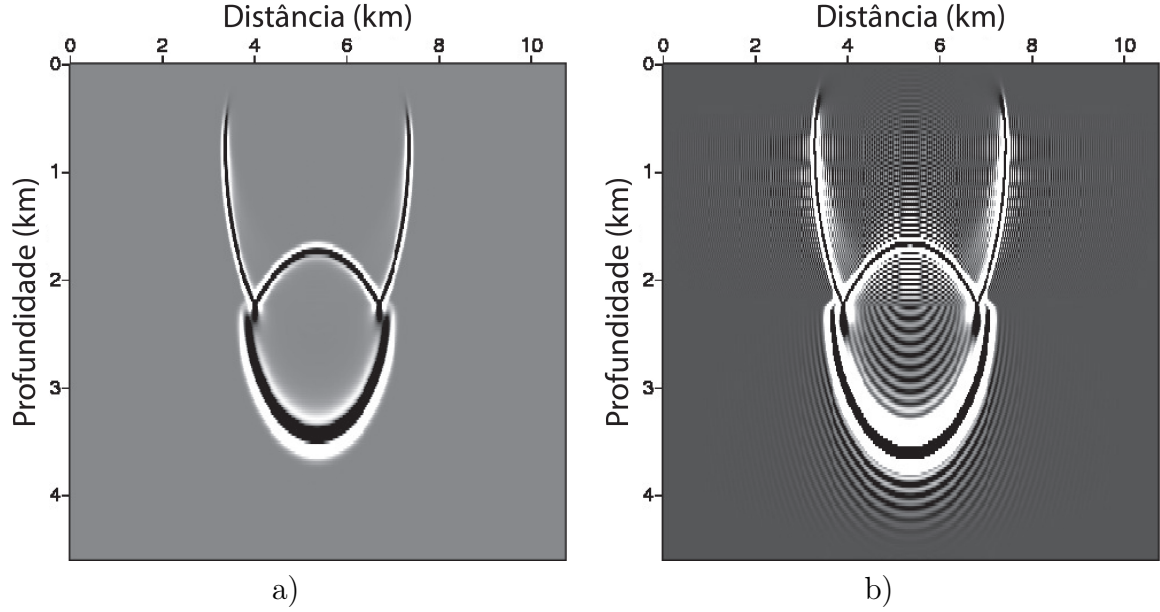


Figura 1.4: Propagação da onda num modelo de duas camadas. a) Propagação livre de dispersão. b) Dispersão temporal (à frente da frente de onda principal) e espacial (atrás da frente de onda principal).

$$\frac{\partial^2 P}{\partial t^2} = \frac{c_0^2}{\Delta x^2} (P_{i+1} - 2P_i + P_{i-1}), \quad (1.30)$$

onde

$$P_i = P(i\Delta x, t).$$

Seguindo o mesmo processo, encontramos

$$\frac{c^2}{c_0^2} \approx 1 - \frac{(k_x^2 \Delta x^2)}{12}. \quad (1.31)$$

Comparando as equações 1.29 e 1.31 vê-se que a dispersão espacial age na direção oposta da temporal. A dispersão espacial atrasa a propagação das frequências mais altas, gerando artefatos numéricos *atrás* da frente de onda principal, como mostra a Figura 1.5-b.

Na literatura (Kosloff e Baysal, 1982, Chu e Stoffa, 2012, Araujo, 2009) é comum analisar graficamente o desvio de $\frac{c}{c_0}$ do valor ideal 1, variando Δt e Δx . Todavia essa análise gráfica não é muito prática à modelagem, salvo casos em que a própria dispersão é o foco de estudo (Chu e Stoffa, 2012). É preferível fazer uso, por praticidade, duma fórmula que nos permita saber a frequência máxima não dispersiva f_{max} com base nos parâmetros da modelagem.

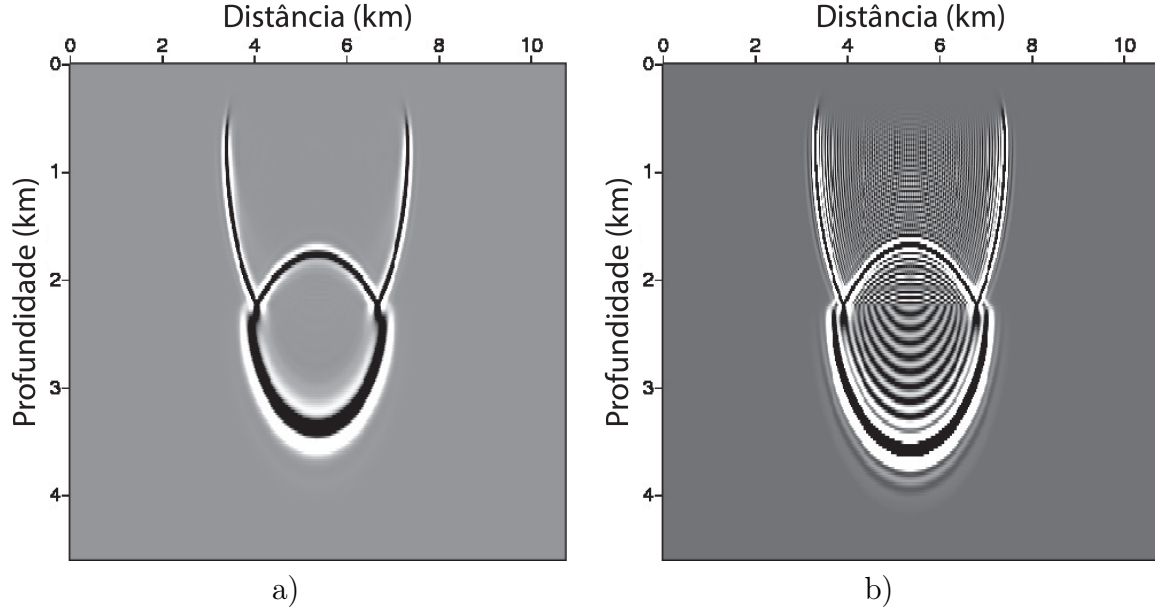


Figura 1.5: Propagação da onda num modelo de duas camadas. a) Propagação livre de dispersão. b) Dispersão espacial (atrás da frente de onda principal).

Analisando novamente um harmônico de P , comprimento de onda λ , frequência f , se propagando num meio de velocidade acústica c_0 , temos:

$$f = \frac{c_0}{\lambda}. \quad (1.32)$$

Num esquema de extrapolação numérica das derivadas espaciais da equação da onda, o comprimento de onda λ passa a ser função do espaçamento da malha Δx e do número mínimo de pontos G necessário para recuperar λ a partir da amostragem (Wu et al., 1996):

$$f = \frac{c_0}{G\Delta x}.$$

O teorema da amostragem de Shannon-Nyquist mostra que 2 pontos por comprimento de onda é suficiente para recuperar uma senóide (Oppenheim e Schaffer, 2009). Essa constatação também é válida para o presente caso de modelagem sísmica, porém apenas quando se calculam as derivadas espaciais por transformada de Fourier (Kosloff e Baysal, 1982), que é um operador considerado exato (ver seção seguinte). Todavia, no caso de aproximações das derivadas espaciais por diferenças finitas, são necessários mais pontos por comprimento de onda para que não ocorram erros numéricos. O valor de G é normalmente encontrado empiricamente (testando e verificando os limites dispersivos), por depender do esquema de diferenças finitas utilizado (Dablain, 1986). Wu et al. (1996) mostraram que, comparando resultados de DF de diversas ordens com resultados analíticos, para um erro RMS normali-

zado da ordem de 10^{-2} nas amplitudes, deve-se seguir:

- $G = 4.61$ a 5.92 para DF de 4ª ordem no espaço;
- $G = 3.58$ a 4.28 para DF de 6ª ordem no espaço;
- $G = 3.15$ a 3.64 para DF de 8ª ordem no espaço.

Diferentemente da instabilidade numérica, que é um fenômeno que, quando ocorre, inviabiliza a modelagem, a dispersão sempre ocorre, mas em diferentes intensidades. Cabe àquele realizando a modelagem escolher o nível de dispersão, ajustando os parâmetros do esquema de diferenças finitas.

Sabendo que λ varia com o meio, a velocidade $c_0 = c_{min}$ garante que estamos tratando do menor λ possível. Estendendo para 2D, substituir Δx por $\max(\Delta z, \Delta x)$ garante o limite dispersivo da análise. Assim,

$$f_{disp} = \frac{c_{min}}{G \max(\Delta z, \Delta x)}. \quad (1.33)$$

Por exemplo, para um modelo Marmousi de dimensões $nz = 125$ e $nx = 383$ amostras, $dz = 24$ m, $dx = 24$ m e $\max(c_0) = 1500$ m/s, usando um operador espacial de segunda ordem, com $G = 10$, resulta em $f_{disp} = 6,25$ Hz. Modelar com uma *wavelet* de frequência máxima acima desse valor gera uma propagação dispersiva.

É possível modelar com frequências além da máxima, e posteriormente reduzir ou mitigar a dispersão, como por exemplo filtrando o campo de onda a cada iteração temporal (Manning e Margrave, 2004) ou modificando a equação da onda acústica para atenuar a energia dispersiva (dos Santos e Pestana, 2012, Liu et al., 2008).

Outras abordagens incluem melhorar o operador de extrapolação espacial, de forma que G se aproxime de 2, e o operador seja menos custoso computacionalmente que o espectral (Chu e Stoffa, 2011, Chu e Stoffa, 2012, Liang et al., 2015).

Na metodologia dessa dissertação seguiu-se a regra prática da equação 1.33. É importante ressaltar que deve-se sempre conferir os resultados da modelagem para verificar o nível de dispersão. As fórmulas apresentadas nessa seção são apenas guias, visto que o erro numérico pode variar dependendo da *wavelet* usada e da implementação feita.

1.2.2 Método pseudo-espectral

Dada a equação da onda acústica com densidade constante

$$\frac{1}{c_0^2} \frac{\partial^2 P(z, x, t)}{\partial t^2} = \frac{\partial^2 P(z, x, t)}{\partial z^2} + \frac{\partial^2 P(z, x, t)}{\partial x^2}, \quad (1.34)$$

o método consiste em resolver as derivadas espaciais via Transformada de Fourier, e a temporal por diferenças finitas de 2ª ordem (sendo portanto espectral apenas no espaço, daí o nome *pseudo-espectral*).

Definindo um par de transformadas de Fourier 2D no espaço por (Telford et al., 1976):

$$\tilde{P}(k_z, k_x, t) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} P(z, x, t) e^{-i(k_z z + k_x x)} dz dx, \quad (1.35)$$

$$P(z, x, t) = \frac{1}{4\pi^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \tilde{P}(k_z, k_x, t) e^{i(k_z z + k_x x)} dk_z dk_x, \quad (1.36)$$

e a propriedade de derivação (Telford et al., 1976):

$$\frac{d^n g(x)}{dx^n} \leftrightarrow (ik_x)^n G(k_x), \quad (1.37)$$

onde $\leftrightarrow$ significa, da esquerda para a direita, transformada direta de Fourier, e da direita para a esquerda, transformada inversa de Fourier. Adaptando a equação 1.37 para as duas derivadas espaciais, temos:

$$\frac{\partial^2 P(z, x, t)}{\partial z^2} + \frac{\partial^2 P(z, x, t)}{\partial x^2} \leftrightarrow (-k_z^2 - k_x^2) \tilde{P}(k_z, k_x, t). \quad (1.38)$$

As derivadas espaciais da equação 1.34 são então resolvidas da seguinte maneira:

- Transforma o campo $P(z, x, t)$ para o domínio (k_z, k_x) ;
- Multiplica cada ponto pelo seu $(-k_z^2 - k_x^2)$ correspondente;
- Transforma $P(k_z, k_x, t)$ de volta para o domínio (z, x) .

Sendo $\mathcal{F}$ notação para aplicar a equação 1.35 a uma função P e $\mathcal{F}^{-1}$ notação para aplicar a equação 1.36, pode-se sintetizar o processo como:

$$\frac{1}{c_0^2} \frac{\partial^2 P(z, x, t)}{\partial t^2} = \mathcal{F}^{-1} \left\{ (-k_z^2 - k_x^2) \mathcal{F} \{P(z, x, t)\} \right\}, \quad (1.39)$$

Por fim, aproximando a derivada temporal por diferenças finitas de 2ª ordem (equação 1.15 seção de DF), temos a equação que permite achar o campo futuro $P(z, x, t + \Delta t)$ pelo método pseudo-espectral:

$$P(z, x, t + \Delta t) = 2P(z, x, t) - P(z, x, t - \Delta t) + \Delta t^2 c_0^2 \mathcal{F}^{-1} \left\{ (-k_z^2 - k_x^2) \mathcal{F} \{P(z, x, t)\} \right\}. \quad (1.40)$$

Estabilidade numérica

A metodologia pseudo-espectral para a equação da onda acústica com densidade constante, como citado anteriormente, consiste em aproximar $\partial^2 P / \partial t^2$ por diferenças finitas de 2ª ordem e calcular $\partial^2 P / \partial z^2$, $\partial^2 P / \partial x^2$ exatamente por meio da transformada de Fourier espacial 2D, usando a propriedade 1.37 de derivação. Visto que o campo P pode ser expresso em termos de cada onda plana que o compõe (Oliveira, 2014), os efeitos das aproximações do método pseudo-espectral (diferenças finitas em $\partial^2 P / \partial t^2$, espectral em $\nabla^2 P$) para um componente de onda plana foram analisados por Araujo (2009).

Já que P é composto de ondas planas, a análise da estabilidade do componente de onda plana com a frequência máxima de P garante a estabilidade de P . Sendo assim, para um componente de onda plana, a seguinte relação (encontrada após algumas manipulações algébricas) deve ser obedecida (Araujo, 2009):

$$\frac{\omega \Delta t}{2} = \arcsin \left(\frac{v \Delta t}{2} (k_z^2 + k_x^2)^{1/2} \right). \quad (1.41)$$

Pela definição de arco-seno,

$$-1 \leq \frac{v \Delta t}{2} (k_z^2 + k_x^2)^{1/2} \leq 1. \quad (1.42)$$

Como $\omega > 0$ e $\Delta t > 0$, se restringe 1.42 novamente a:

$$\frac{v \Delta t}{2} (k_z^2 + k_x^2)^{1/2} \leq 1. \quad (1.43)$$

O limite da estabilidade para Δt é encontrado quando analisamos os valores limite de k_z e

k_x . Esses valores são as frequências de Nyquist espaciais $k_z^{max} = \frac{\pi}{\Delta z}$ e $k_x^{max} = \frac{\pi}{\Delta x}$. Assim,

$$\frac{v\Delta t}{2} \left(\frac{\pi^2}{\Delta z^2} + \frac{\pi^2}{\Delta x^2} \right)^{1/2} \leq 1. \quad (1.44)$$

De forma a encontrar uma fórmula prática que garanta a estabilidade nas modelagens sísmicas, podemos fazer, na equação 1.44, $v = \max(v)$ e $\Delta z = \Delta x = \max(\Delta z, \Delta x)$. Aplicando essas condições a 1.44 e isolando Δt , a relação

$$\Delta t \leq \frac{2}{\pi \max(v) \sqrt{\frac{1}{\Delta z^2} + \frac{1}{\Delta x^2}}} \quad (1.45)$$

garante a estabilidade do método pseudo-espectral. Por exemplo, para um modelo Marmousi de dimensões $n_z = 125$ e $n_x = 383$ amostras, $dz = 24$ m, $dx = 24$ m e $\max(v) = 5500$ m/s, usando um operador pseudo-espectral temos $\Delta t \leq 0,00196$ s. Acima desse valor, a solução é instável. Como garantia duma boa propagação, normalmente se escolhe um valor abaixo do limite; nesse caso, $\Delta t = 0,0018$ s garante uma solução estável.

Dispersão numérica

A discussão teórica da dispersão numérica já foi feita na seção anterior. Aqui apresentamos novamente a regra prática que determina a frequência máxima não-dispersiva, onde $G = 2$ para o operador pseudo-espectral em questão (Kosloff e Baysal, 1982):

$$f_{disp} = \frac{c_{min}}{2 \max(\Delta z, \Delta x)}. \quad (1.46)$$

1.2.3 Método de Expansão Rápida (REM)

O método de expansão rápida, ou REM (*rapid expansion method*, Pestana e Stoffa, 2010) fornece uma maneira altamente estável de resolver a equação da onda acústica, ao custo de memória de um esquema de diferenças finitas temporal de segunda ordem. Por ser muito estável, o método permite flexibilidade na escolha do passo temporal Δt , o que é de grande utilidade na realização de testes na RTM e FWI (ver capítulo "Metodologia e Resultados").

O REM parte da solução exata no tempo da equação da onda, escrita na forma:

$$\frac{\partial^2 P}{\partial t^2} = -L^2 P. \quad (1.47)$$

onde

$$-L^2 = c^2 \nabla^2. \quad (1.48)$$

Em Pestana e Stoffa (2010) é mostrada a dedução da solução temporal da equação da onda 1.47, avaliada em $t + \Delta t$ e $t - \Delta t$. Tal solução provê uma forma de marchar no tempo semelhante à aproximação de diferenças finitas de segunda ordem no tempo, que para calcular $P(t + \Delta t)$ necessita apenas dos dois campos anteriores $P(t)$ e $P(t - \Delta t)$, fator chave num esquema de extrapolação, pois como discutido anteriormente (seção 1.2.1) o uso de mais campos pode inviabilizar a implementação computacional devido ao alto custo de memória. A equação exata com termo cosseno (Pestana e Stoffa, 2010), usando notação discreta (equação 1.17), é:

$$P_{i,j}^{n+1} = P_{i,j}^{n-1} + 2 \cos(L\Delta t) P_{i,j}^n. \quad (1.49)$$

Apesar de utilizar apenas 3 pontos no tempo, como na DF temporal de 2ª ordem, a equação 1.49 pode ter a precisão temporal ajustada para qualquer ordem, desde que se utilize um número de termos suficiente ao expandir o operador $\cos(L\Delta t)$. Em outras palavras, o aumento da precisão temporal é contemplado pelo cálculo de mais Laplacianos associados ao termo cosseno, dada a equivalência entre derivada temporal e derivadas espaciais dada por 1.47 (Pestana e Stoffa, 2010).

O cosseno na equação 1.49 pode ser expandido utilizando os polinômios de Chebyshev (Kosloff et al., 1989):

$$\cos(L\Delta t) = \sum_{k=0}^M C_{2k} J_{2k}(\Delta t R) Q_{2k} \left(\frac{iL}{R} \right) \quad (1.50)$$

onde $C_{2k} = 1$ para $k = 0$ e $C_{2k} = 2$ para $k > 0$. O valor J_{2k} representa a função de Bessel de ordem $2k$, e $Q_{2k} \left(\frac{iL}{R} \right)$ são os polinômios modificados de Chebyshev, calculados segundo a relação recursiva:

$$\begin{aligned}
Q_0(w) &= 1 \\
Q_2(w) &= 1 + 2w^2 \\
Q_{k+2}(w) &= 2(1 + 2w^2)Q_k(w) - Q_{k-2}(w),
\end{aligned} \tag{1.51}$$

onde $w = iL/R$.

O termo R é um escalar associado ao maior autovalor de $-L^2$. Para o caso 2-D, o valor de R é dado por:

$$R = \pi c \sqrt{(1/\Delta x^2) + (1/\Delta z^2)}, \tag{1.52}$$

onde em geral o valor de c deve ser substituído por c_{max} , o maior valor de velocidade do modelo (Pestana e Stoffa, 2010).

A estabilidade do REM é garantida pelo somatório 1.50, computando o número máximo de termos adequado ao problema. A série converge exponencialmente para $M > R\Delta t$, isto é, o número de termos usados na expansão é diretamente proporcional ao intervalo de amostragem no tempo utilizado. Ou seja, para qualquer Δt , o número de termos calculados será suficiente para garantir a convergência da expansão, e, conseqüentemente, a estabilidade numérica do método (Tal-Ezer et al., 1987).

Na solução numérica pelo REM, os Laplacianos da expansão cosseno podem ser calculados pelas técnicas de solução descritas nessa seção: podemos ter REM-diferenças-finitas, onde os Laplacianos são calculados pelo método de diferenças finitas, ou REM-pseudo-espectral, onde são calculados por transformada de Fourier. Qualquer técnica de solução numérica que resolva o Laplaciano pode ser aplicada no método de expansão rápida; todavia deve-se atentar ao fato de que a condição de dispersão do REM será a condição de dispersão dada pelo método utilizado para aproximação do Laplaciano.

Estabilidade e dispersão do REM

O REM é sempre estável desde que se escolha o número de termos M maior que $R\Delta t$. Quanto à dispersão, existem duas condições:

- $f_{max} < f_{disp}$. f_{max} é a frequência máxima da fonte, e f_{disp} é dada por 1.33, com G correspondente ao método usado para o cálculo dos Laplacianos do termo cosseno em 1.49;

- $\Delta t < 1/2f_{disp}$, ou seja, menor que o tempo de Nyquist correspondente ao limite dispersivo f_{disp} .

Por exemplo, calculando os Laplacianos do REM por um operador de DF de 12^a ordem, onde $G \approx 2$ na equação de dispersão, e usando um modelo de duas camadas com $c_{min} = 1500$ m/s e $\max(\Delta z, \Delta x) = 24.0$ m, f_{disp} fica:

$$f_{disp} = \frac{c_{min}}{2 \max(\Delta z, \Delta x)} = \frac{1500}{2.24} = 31,25 Hz. \quad (1.53)$$

Usando uma fonte de $f_{peak} = 10$ Hz, $f_{max} \approx 23,10$ Hz (pela regra $f_{max} \approx 2.31f_{peak}$ citada no fim da seção 1.1). Assim, na Figura 1.6-a, $f_{max} < f_{disp}$, pois $23,10 < 31,25$, a regra é respeitada e a propagação é não dispersiva. Na Figura 1.6-b, modelamos com $f_{peak} = 20$ Hz, $f_{max} \approx 46,20$ Hz, desrespeitando a primeira condição e gerando uma propagação dispersiva. Em ambos os casos, foi usado $\Delta t = 4$ ms.

Na Figura 1.7-a, repetimos a modelagem: $f_{peak} = 10$ Hz, $f_{max} \approx 23,10$ Hz, $\Delta t = 4$ ms, respeitando as duas condições de dispersão. Na Figura 1.7-b, a frequência da fonte é a mesma e $\Delta t = 24$ ms, o que desobedece a segunda condição de dispersão: $\Delta t < 1/2f_{disp} = 1/(2 \times 31,25) = 16$ ms; conseqüentemente, a propagação é dispersiva.

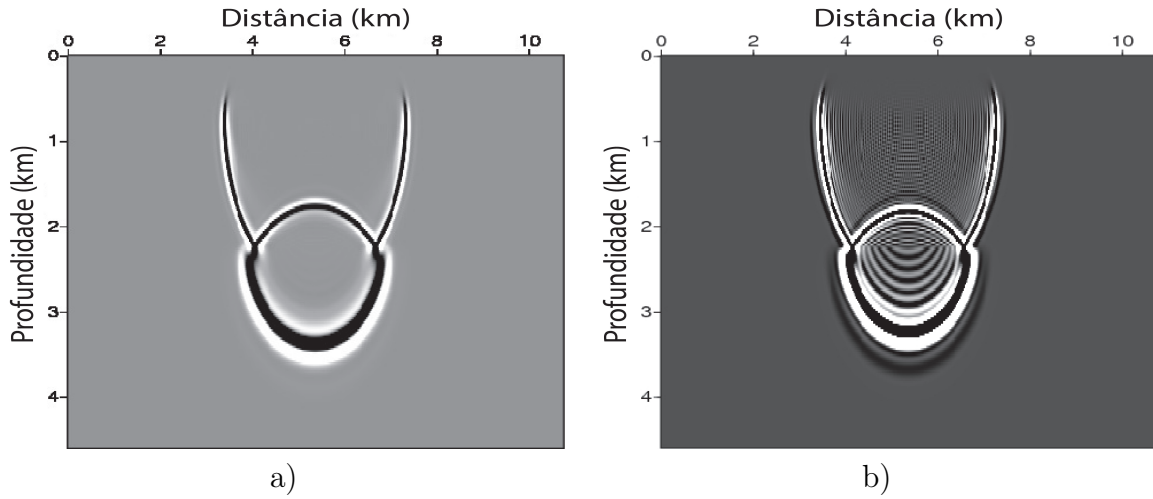


Figura 1.6: Quebra da primeira condição de dispersão para o REM. a) $f_{max} = 23,10$ Hz, $\Delta t = 4$ ms. b) $f_{max} = 46,20$ Hz, $\Delta t = 4$ ms.

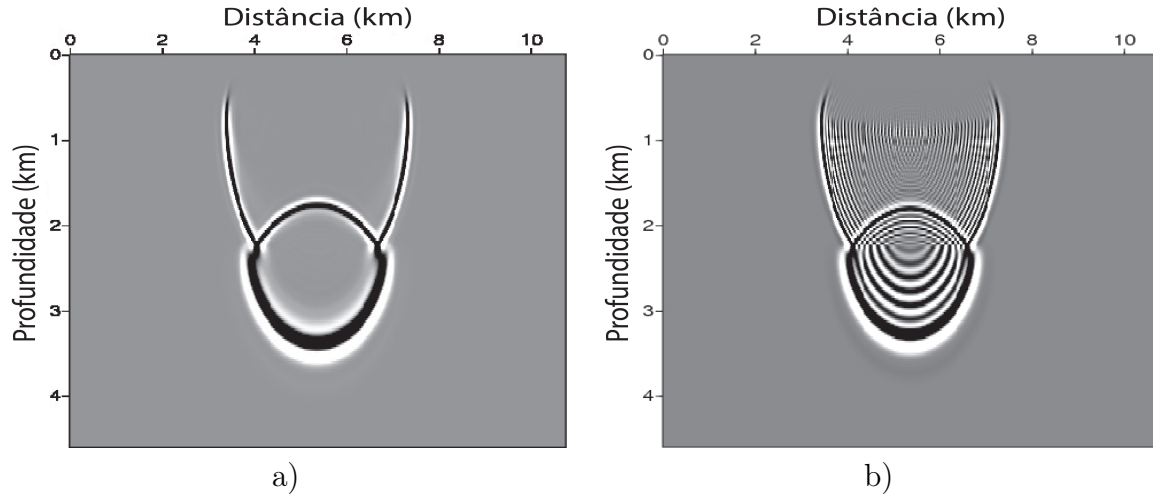


Figura 1.7: Quebra da segunda condição de dispersão para o REM. a) $f_{max} = 23, 10$ Hz, $\Delta t = 4$ ms. b) $f_{max} = 23, 10$ Hz, $\Delta t = 24$ ms.

1.3 Reflexão nos limites do modelo

Todos os métodos de solução numérica da equação da onda acústica fazem uso de pontos vizinhos para calcular as derivadas espaciais em cada ponto. Um problema surge nos limites do modelo: não há pontos vizinhos. Nesses limites os valores de pressão P são sempre zero, visto que não há como calcular as derivadas espaciais; o contraste entre o valor zero e a frente de onda que chega nos limites provoca uma reflexão da onda de volta ao modelo (Figura 1.8-a). Esse fenômeno numérico é altamente indesejável, visto que as reflexões não correspondem a uma simulação real da terra. Desta forma, para realizar uma modelagem sísmica satisfatória, é necessário algum método para atenuar essas reflexões nos limites do modelo.

Uma forma eficiente de evitar o problema consiste em inserir bordas após os limites do modelo, e apenas dentro dessa borda inserir uma condição de atenuação da energia da onda. Dessa forma, quando a onda atingir o novo limite e refletir, sua energia será tão pequena que os eventos de reflexão nos limites terão amplitude desprezível quando comparados aos eventos reais. Nesse trabalho foram testadas duas condições de borda absorvedora: condição de borda esponja, ou *taper* (Cerjan et al., 1985), e a PML (Moreira et al., 2011, Collino e Tsogka, 2001), ou *Perfectly Matched Layer* ("Camada Perfeitamente Adaptada" em tradução livre).

O método do *taper* consiste em atenuar gradualmente as amplitudes que chegam à borda, multiplicando por um fator T dado por (Cerjan et al., 1985):

$$T(ib) = e^{-[0.015(nb-ib)]^2}, \quad (1.54)$$

onde ib é a distância (unidimensional) do limite estendido do modelo ao limite do modelo original; nb é o tamanho da borda. Por exemplo, no limite original do modelo, $ib - nb = 0$ (sem atenuação); no limite estendido, $nb - ib = nb$ (atenuação máxima); e dentro da borda, uma variação suave dada pela equação 1.54. Note que a equação 1.54 é unidimensional, de forma que, ao implementar o taper, se deve construir a atenuação vertical (ib como índice vertical, no topo e base do modelo estendido), separadamente da atenuação horizontal (ib como índice horizontal, na esquerda e direita do modelo estendido).

Já a PML consiste em aplicar, nas bordas, a equação da onda amortecida, dada por (Moreira et al., 2011):

$$\frac{\partial^2 P}{\partial t^2} + 2\alpha B \frac{\partial P}{\partial t} + \alpha^2 B^2 P = c_0^2 \nabla^2 P, \quad (1.55)$$

que, resolvida por diferenças finitas centradas 2ª ordem para $\frac{\partial^2 P}{\partial t^2}$ (equação 1.15), e diferença finita progressiva para $\frac{\partial P}{\partial t} \approx \frac{P(t+\Delta t) - P(t)}{\Delta t}$, fornece a equação de extrapolação na borda:

$$P(t+\Delta t) = \left[\frac{1}{1 + 2\alpha B \Delta t} \right] \left\{ 2P(t) - P(t - \Delta t) + c_0^2 \Delta t^2 \nabla^2 P(t) + [2\alpha B \Delta t - \alpha^2 B^2 \Delta t^2] P(t) \right\}, \quad (1.56)$$

onde, por Moreira et al. (2011),

$$B = c_0^2$$

$$\alpha(ib) = \frac{1}{c_0^2 \Delta t} \ln \left(\frac{1}{r_{pml}} \right) \left(\frac{ib}{nb} \right)^k, \quad (1.57)$$

e r_{pml} , o coeficiente máximo de absorção, tem melhor valor 0.8, e o expoente k tem melhor valor 4. Nessa dissertação testamos o proposto por Moreira et al. (2011), e uma forma modificada para α , que apresentou melhores resultados, adaptada de Collino e Tsogka (2001):

$$\alpha(ib) = \frac{3}{2 nb c_0} \ln \left(\frac{1}{r_{pml}} \right) \left(\frac{ib}{nb} \right)^k, \quad (1.58)$$

$$r_{pml} = 0.5, \quad k = 2.$$

A Figura 1.8 mostra a comparação entre propagação sem borda (a), com *taper* (b), com a PML (Moreira et al., 2011) e (d) com PML (Collino e Tsogka, 2001 modificado). Analisando

a reflexão na borda inferior do modelo vê-se que, para o modelo de duas camadas, o *taper* produziu os melhores resultados.

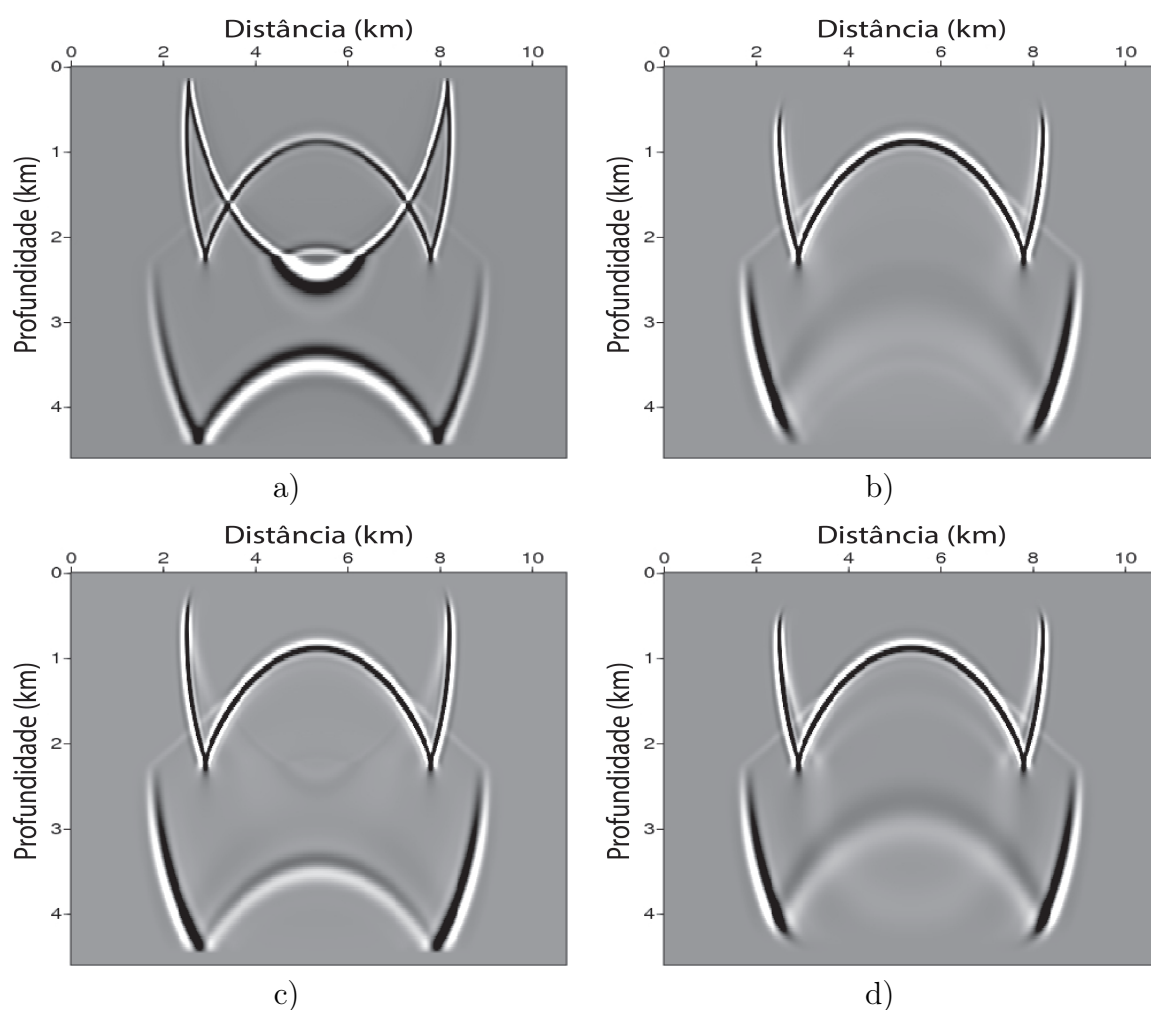


Figura 1.8: Propagação num modelo de 2 camadas, após 20ms. a) Sem borda absorvedora; b) com *taper*; c) com PML (Moreira et. al., 2011); d) com PML ("Collino modificado", de Collino e Tsogka, 2011).

2

Migração Reversa no Tempo

A migração reversa no tempo, ou RTM (do inglês, *reverse time migration*), foi introduzida inicialmente por Baysal et al. (1983), Whitmore (1983), McMechan (1983), Loewenthal e Mufti, 1983, como um método de migração, usando a equação completa da onda, onde o sismograma é utilizado como condição de contorno na superfície ($z = 0$) do modelo geológico, num esquema de extrapolação reversa no tempo da equação da onda.

A RTM foi vista como muito promissora, por permitir imagear estruturas complexas, de altos ângulos e com variações laterais abruptas de velocidade; todavia, durante muito tempo, o custo de memória e computacional foi um empecilho para implementação do método pela indústria.

O algoritmo da RTM pré-empilhamento tem três etapas: para cada tiro, (a) extrapolação direta no tempo do campo da fonte $P_s(z, x, t)$; (b) extrapolação reversa no tempo do campo dos receptores $P_r(z, x, t)$, onde o sismograma do tiro corrente é injetado nas posições dos receptores; e (c) aplicação de uma condição de imagem que delineie onde ocorreram as reflexões. A forma mais usual, e até hoje utilizada, de extrapolar os campos, é resolver numericamente a equação da onda acústica, em geral utilizando diferenças finitas de segunda ordem no tempo e de mais altas ordens no espaço (ou método pseudo-espectral). Nesse sentido, com base na equação 1.18, o campo da fonte $P_s(z, x, t)$ é extrapolado por:

$$P_{i,j}^{n+1} = 2P_{i,j}^n - P_{i,j}^{n-1} + c_0^2 \Delta t^2 D_k(P_{i,j}^n) + f_s^n(z_s, x_s), \quad (2.1)$$

onde a fonte $f_s^n(z_s, x_s)$ é normalmente uma *wavelet* (Ricker, por exemplo). A Figura 1.1, no início do capítulo anterior, ilustra a propagação do campo da fonte. O campo dos recepto-

res $P_r(z, x, t)$ é extrapolado com a mesma equação, mas de maneira reversa no tempo, ou seja:

$$P_{i,j}^{n-1} = 2P_{i,j}^n - P_{i,j}^{n+1} + c_0^2 \Delta t^2 D_k(P_{i,j}^n) + f_r^n(z_r, \mathbf{x}_r), \quad (2.2)$$

onde $f_r^n(z_r, \mathbf{x}_r) = S_r \mathbf{d}_{\text{obs}}^k$, sendo d_{obs}^k o k -ésimo tiro, e S_r um operador que insere o sismograma linha a linha nas posições dos receptores. As Figuras 2.1, 2.2, 2.3 ilustram esse processo; o sismograma é inserido, fatia a fatia, na superfície, desde o tempo máximo até o tempo zero, e seus eventos são sequencialmente depropagados para dentro da terra.

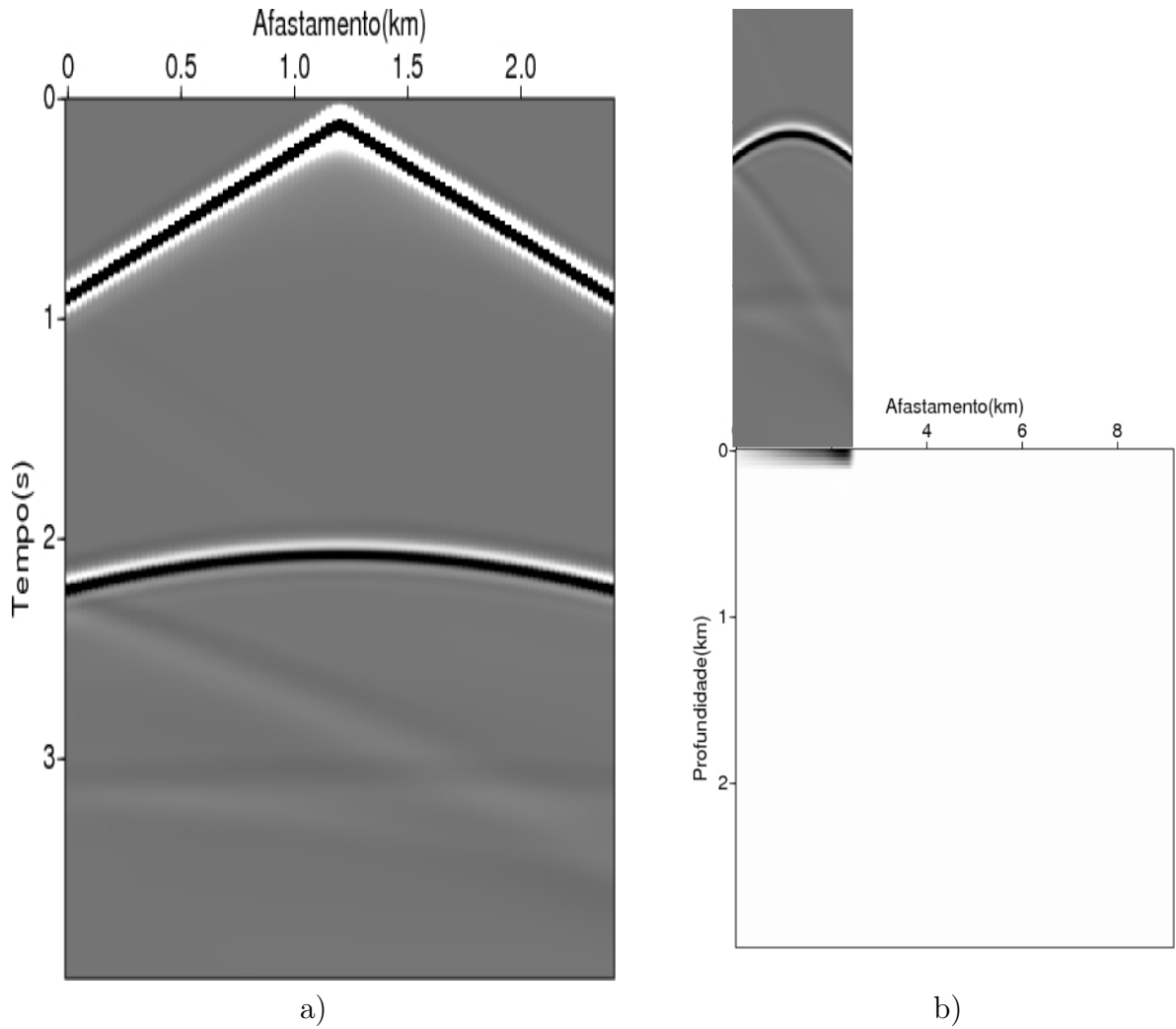


Figura 2.1: Propagação reversa no tempo de um tiro, visto em (a). Início da propagação reversa, fatia 3, 8 s do sismograma é inserida na superfície, como indicado (b).

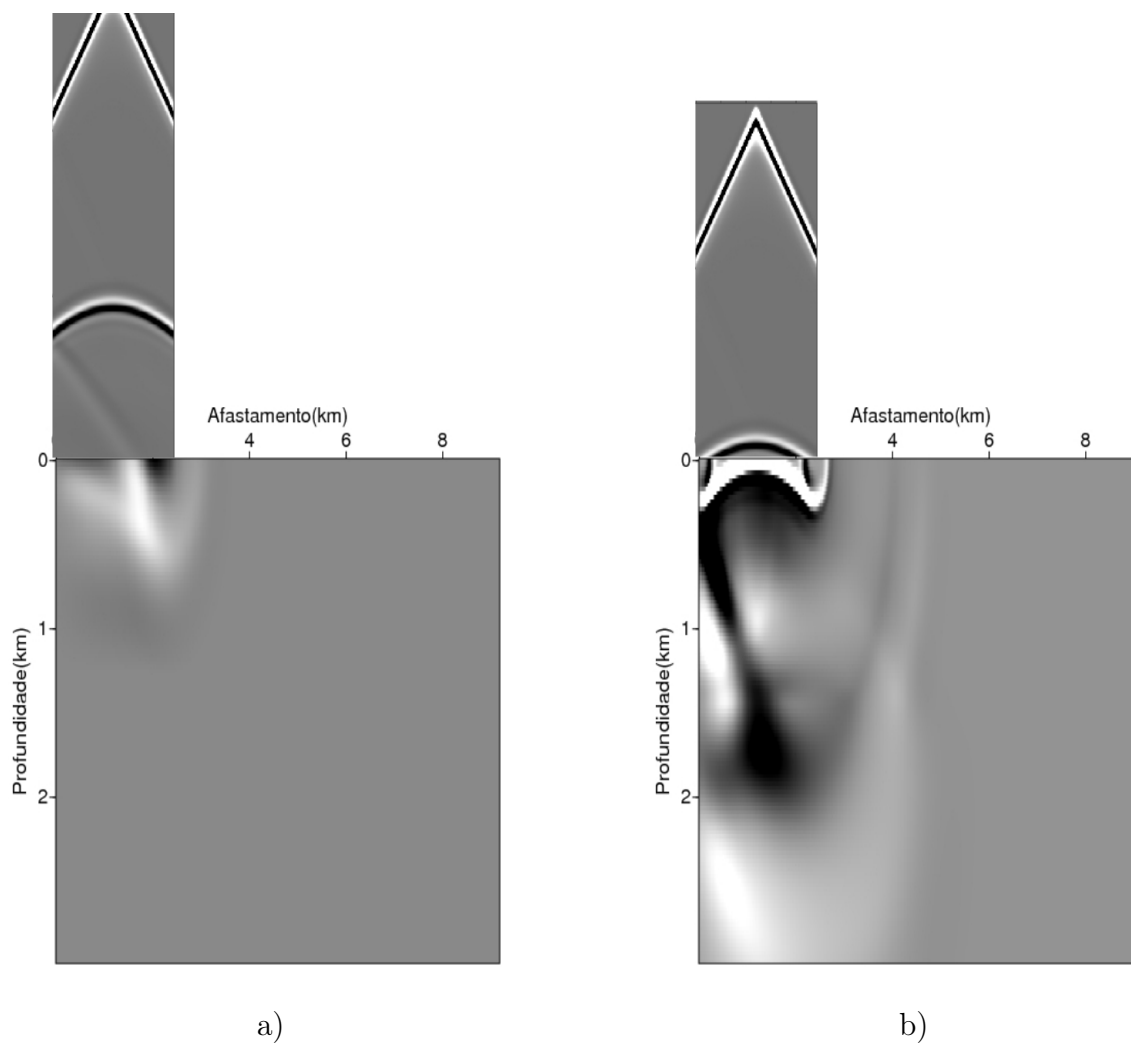


Figura 2.2: Continuação da propagação reversa do tiro. Em (a), é inserida a fatia 3,2 s do sismograma, onde se pode perceber o evento decorrente da borda (não atenuado completamente) sendo depropagado. O processo continua sequencialmente; em (b) é inserida a fatia 2,08 s do sismograma, onde vemos o evento da reflexão começando a ser depropagado.

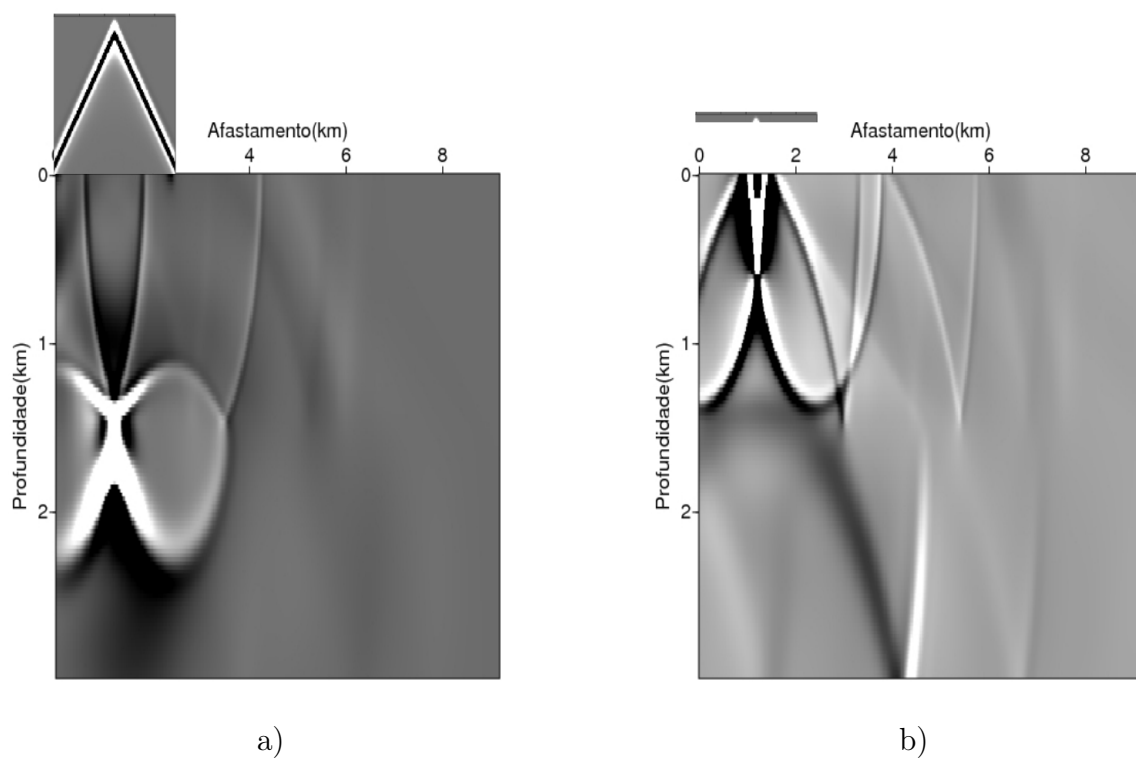


Figura 2.3: Continuação da depropagação. Fatia 1,04 s em (a), fatia próxima de 0 s em (b).

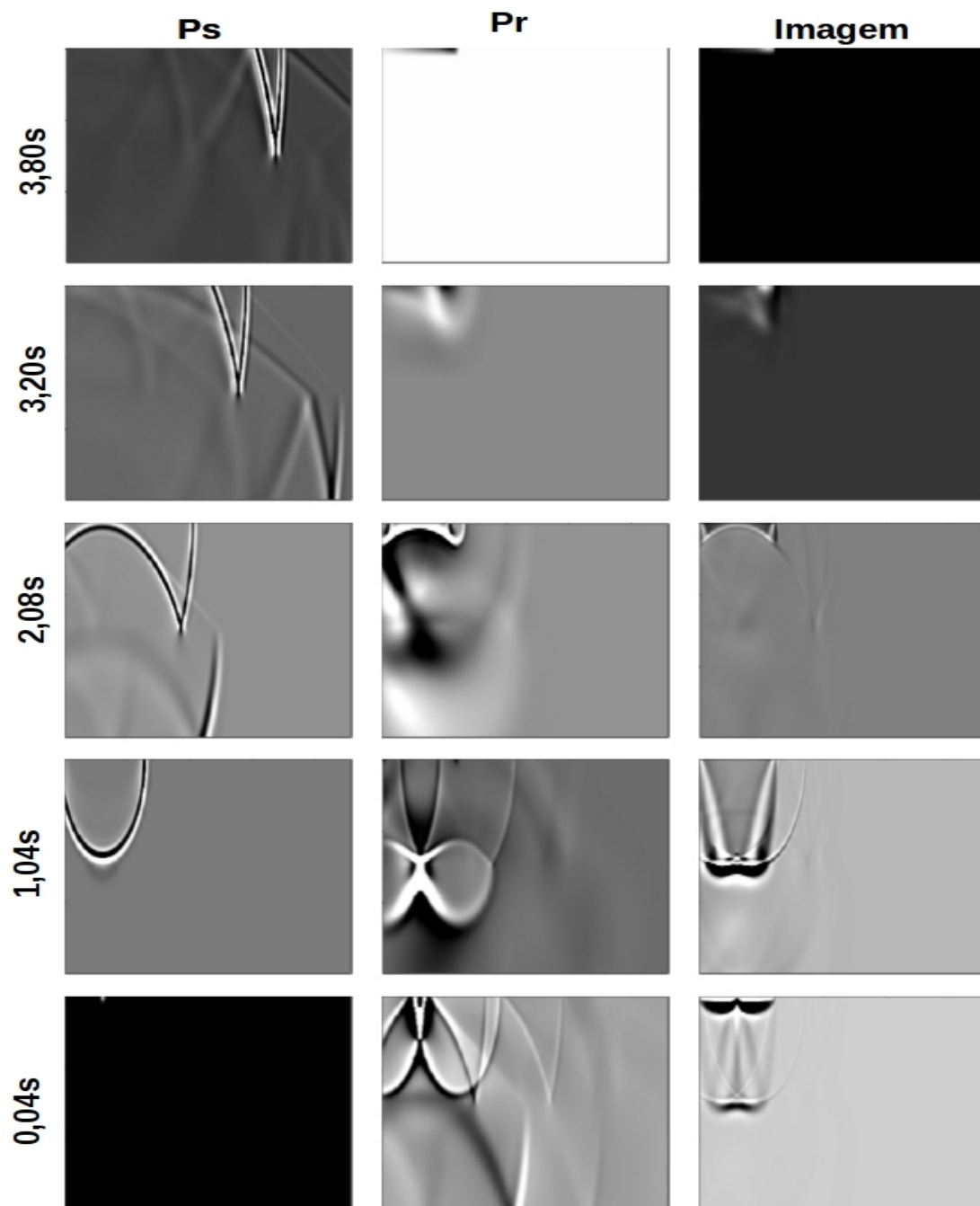


Figura 2.4: Construção da imagem de um tiro pela RTM, para um modelo de duas camadas.

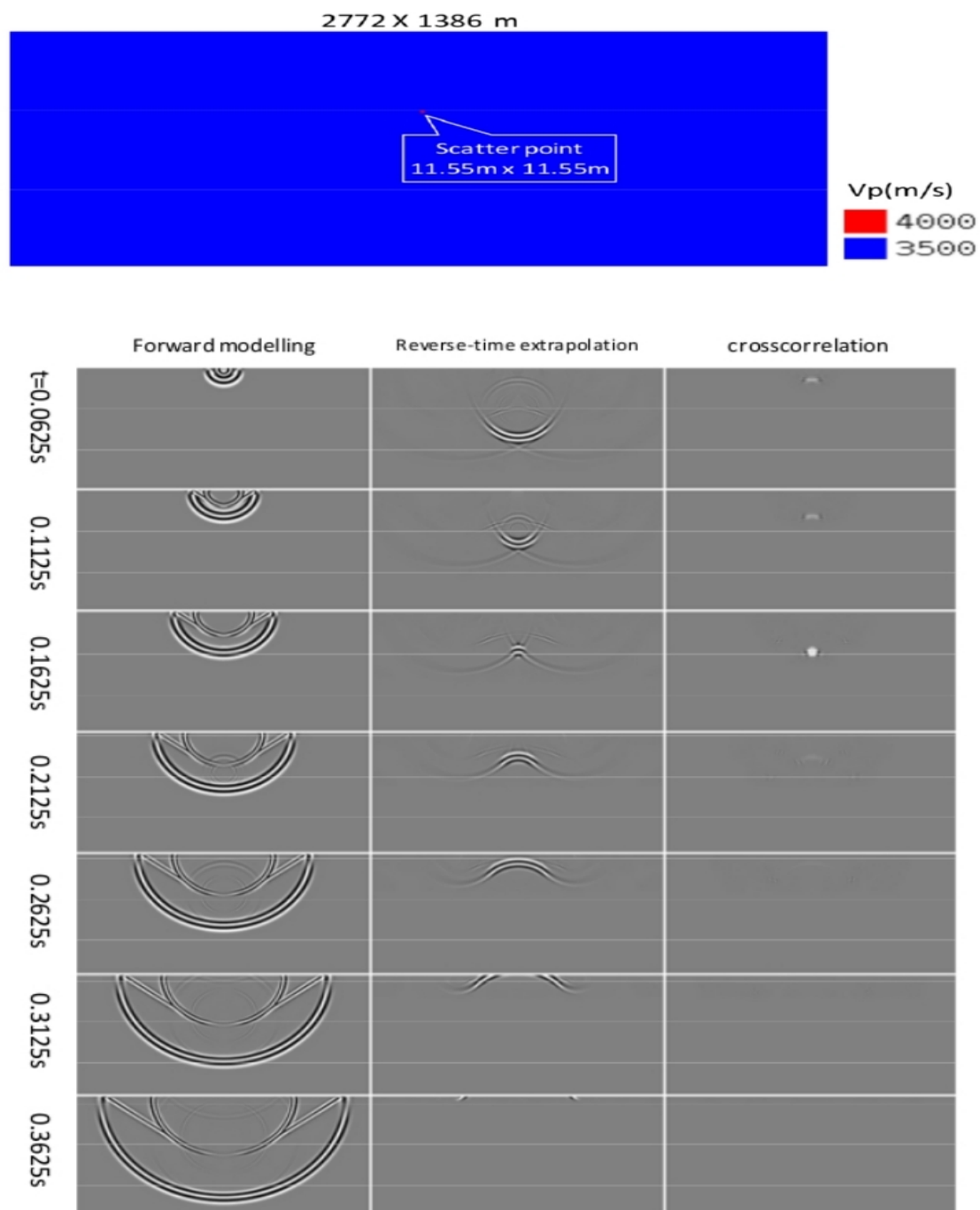


Figura 2.5: Construção da imagem de um tiro pela RTM, para um modelo de um ponto difrator. Retirado de Lines et. al. (2010)

A imagem é normalmente construída pela seguinte condição de imagem de correlação cruzada no *lag* zero (Claerbout, 1971):

$$I_s(z, x) = \int_0^T P_s(z, x, t) P_r(z, x, t), \quad (2.3)$$

tiro a tiro. A imagem final é dada pela soma das imagens de cada tiro, ou

$$I(z, x) = \sum_s I_s(z, x). \quad (2.4)$$

A condição de imagem pode ser entendida observando a Figura 2.4, que representa a RTM de um tiro para o modelo de duas camadas. Os campos P_s e P_r , de $t = T$ a $t = 0$, tem correlação cruzada máxima na posição do refletor. A Figura 2.5, retirada de Lines et al. (2010), mostra o mesmo para um modelo de velocidades com um ponto espalhador; novamente, a correlação cruzada é máxima em cima do ponto espalhador (que é o único refletor do modelo). Observar que na Figura 2.4 a imagem é dada pela soma das correlações até o instante representado (acumulada), enquanto que na Figura 2.5 a imagem mostra apenas o valor da correlação cruzada no instante representado.

A condição de imagem de correlação cruzada pode ser melhorada, por exemplo, para corrigir o efeito de divergência esférica. Isso é discutido na seção seguinte, "Modificações na condição de imagem". Além disso, essa condição de imagem gera artefatos de baixa frequência e alta amplitude que devem ser atenuados; a correção deles é discutida na seção "Artefatos da condição de imagem de correlação cruzada".

Enquanto que as Figuras 2.4 e 2.5 ilustram a equação 2.3, ou seja, para apenas um tiro, a Figura 2.6 ilustra a equação 2.4, para vários tiros, num modelo de duas camadas.

Pode-se observar que o campo $P_s(z, x, t)$ e o campo $P_r(z, x, t)$ são extrapolados em direções opostas no eixo do tempo, isto é, o campo de onda da fonte é extrapolado à frente no tempo, começando no tempo zero e na posição da fonte, enquanto que o campo de onda do receptor é extrapolado de forma reversa no tempo, começando no tempo máximo e nas posições dos receptores. A inserção dos campos na equação 2.3 para o posterior cálculo da correlação cruzada podem ser abordados de três maneiras: salvar todos os *snapshots* do campo de onda $P_s(z, x, t)$ e durante o cálculo reverso no tempo de $P_r(z, x, t)$, acessar na memória as posições previamente salvas de $P_s(z, x, t)$; ou salvar *snapshots* em intervalos de tempo discretos maiores que o passo temporal, recuperando os *snapshots* restantes de $P_s(z, x, t)$ durante a condição da imagem, através de interpolação ou extrapolação (técnica de *checkpointing*, Symes, 2007); ou uma terceira opção, que é salvar os valores do campo

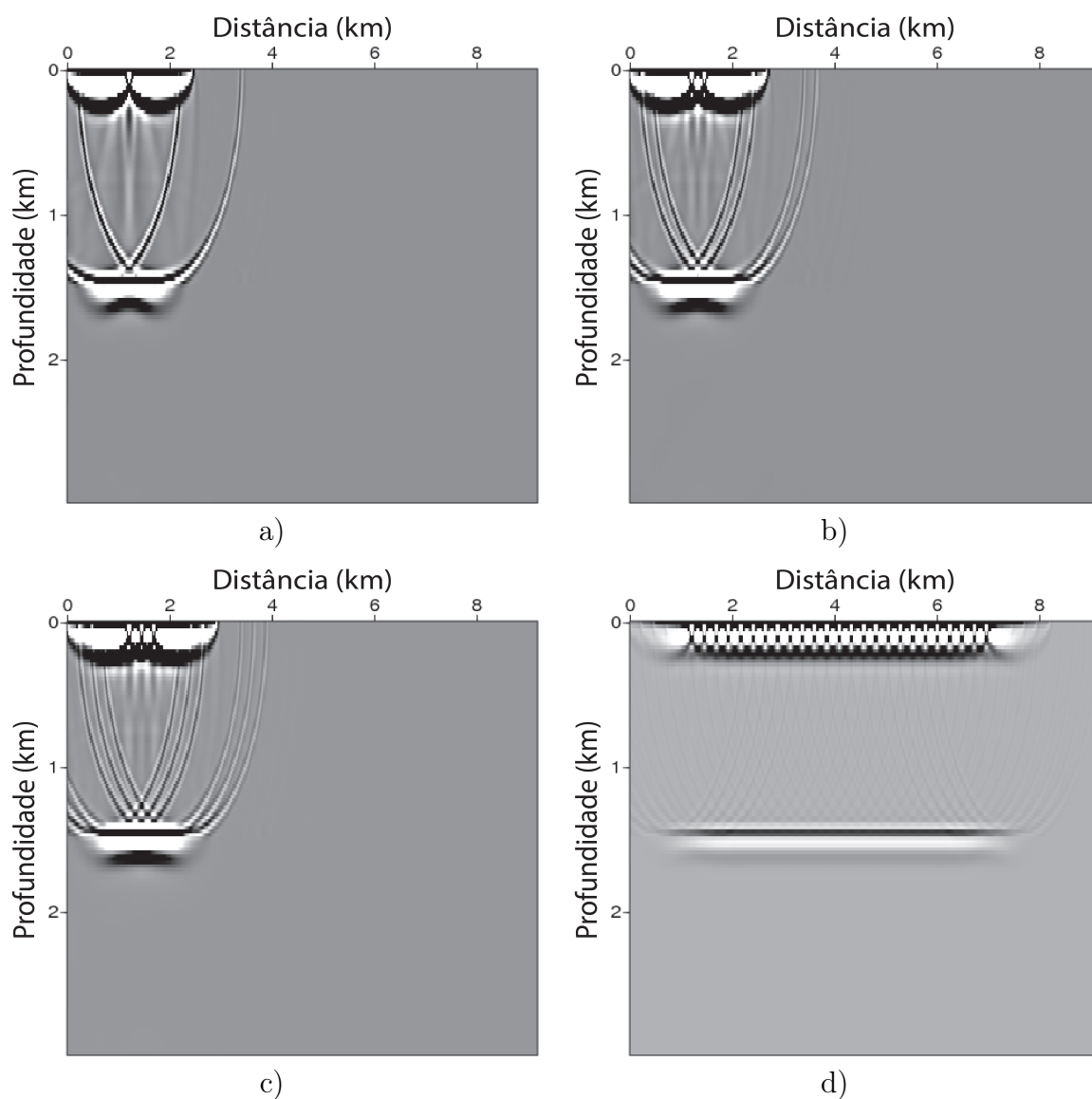


Figura 2.6: Imagem da RTM (pós filtro Laplaciano) num modelo de duas camadas, para (a) 1 tiro, (b) 2 tiros, (c) 3 tiros e (d) 25 tiros. Pode-se observar que cada tiro ilumina melhor o ponto do refletor diretamente abaixo de si. Além da imagem do refletor, há artefatos na superfície decorrentes da onda direta de cada tiro.

de onda apenas nas bordas da malha para todos os passos de tempo, e depois computar o campo de onda nos pontos da parte interna da malha utilizando os valores salvos das bordas como condições de contorno. Cada uma destas formas tem suas vantagens e desvantagens, a depender da configuração do sistema (Dussaud et al., 2008). Nessa dissertação, se optou pela última técnica, a chamada borda efetiva, apresentada e explicada na seção "Borda Efetiva".

Como um resumo do ponto de vista teórico, Symes (2007) aponta que a RTM equivale ao primeiro passo do algoritmo gradiente para uma inversão mínimos quadrados, computada pelo método de estado adjunto (Lailly, 1983, Tarantola, 1984).

Tanto na modelagem do campo da fonte quanto do campo dos receptores, é necessário saber o valor das velocidades sísmicas em cada ponto (z, x) do modelo. O próprio campo de velocidades (verdadeiro), usado pra gerar uma imagem, já é uma imagem melhor que a imagem da RTM. Esse aparente paradoxo é discutido por Whitmore (1983):

"A sensibilidade aumentada dos procedimentos de migração em profundidade às velocidades intervalares traz consigo um paradoxo: se o modelo de velocidades é conhecido, então a migração não é necessária, e se o modelo de velocidades não é conhecido, então um pré-requisito essencial para a migração está ausente. Por causa desse paradoxo, a própria migração não é um método inverso. A migração em profundidade é, no entanto, uma ferramenta muito útil para desvendar estruturas complexas. Oferece, ao menos, um procedimento no qual um modelo inicial pode ser testado, comparando o resultado da sua migração em profundidade com um modelo em profundidade. Se a comparação não for favorável, então devem ser feitas modificações no modelo. No caso em que a sequência vertical de velocidades é conhecida em alguns locais (por exemplo, a partir de controle de poço, análise de velocidade ou migração pré-empilhamento), então a migração pode ser usada iterativamente para ajudar a desenvolver um modelo geológico melhorado".

No seu artigo, Whitmore (1983) dá as bases da RTM e da migração como método de análise de velocidades. O paradoxo é esclarecido pelo procedimento de melhoria iterativa do modelo de velocidades, através de ajustes manuais. Na formulação mínimos quadrados da FWI o procedimento é semelhante, mas o ajuste iterativo do modelo de velocidades é automático, definido pela teoria da otimização. Na FWI por métodos gradiente, a RTM é o ponto fundamental de cada iteração, pois fornece o gradiente que será usado para atualizar o modelo de velocidades corrente (capítulo 3).

2.1 Modificações na condição de imagem

A condição de imagem de correlação cruzada, para todos os tiros, é dada por:

$$I(z, x) = \sum_s^{N_{sht}} \sum_{t=0}^T P_s(z, x, t) P_r(z, x, t). \quad (2.5)$$

Essa condição de imagem pode ser modificada com diversos objetivos, como atenuar os artefatos de baixa frequência (Liu et al., 2011), ou realçar refletores rasos e/ou profundos (Kaelin e Guitton, 2006, Apraez, 2015), ou corrigir as amplitudes atenuadas por divergência esférica. Nessa dissertação se usa uma modificação da condição de imagem do tipo (Gauthier et al., 1986; Bai et al., 2014):

$$I(z, x) = \frac{\sum_s^{N_{sht}} \sum_{t=0}^T P_s(z, x, t) P_r(z, x, t)}{\sum_s^{N_{sht}} \sum_{t=0}^T P_s(z, x, t)^2}. \quad (2.6)$$

ou seja, uma compensação da iluminação do campo da fonte P_s , que visa corrigir efeitos de divergência esférica. Na Figura 2.7 são mostradas duas seções migradas do modelo Marmousi-Teste (definido no penúltimo parágrafo da Introdução), com $f_{max} \approx 30$ Hz. É possível perceber a diferença nas imagens da Figura 2.7; a segunda, com a condição de imagem 2.6, melhora o efeito de divergência esférica e resalta os refletores da imagem.

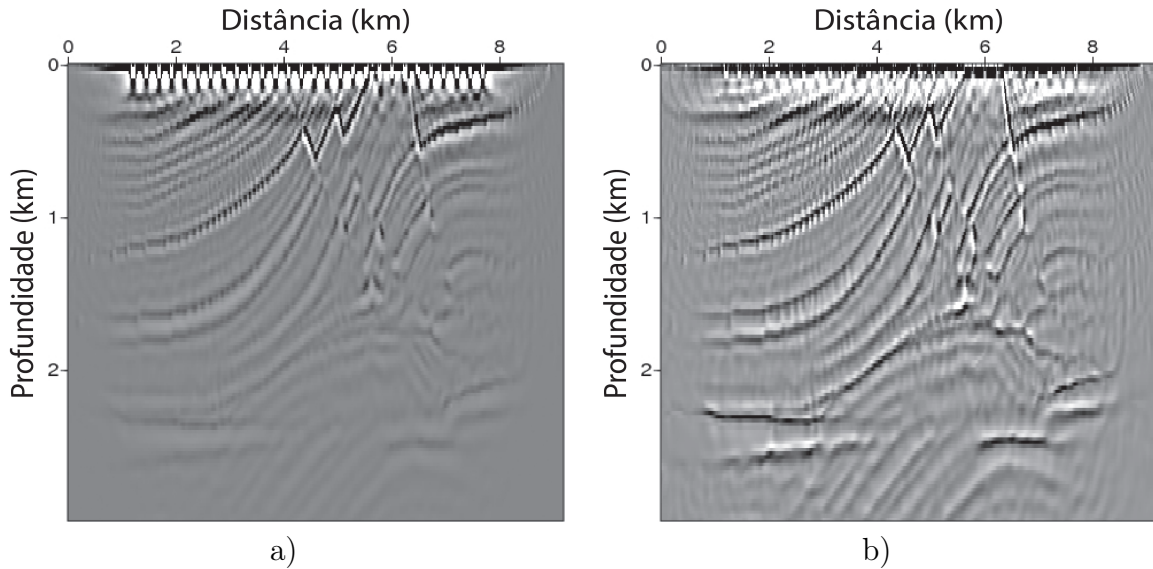


Figura 2.7: Imagens da RTM para o modelo Marmousi. (a) Condição de imagem usual. (b) Condição de imagem com compensação de iluminação do campo de onda da fonte.

2.2 Borda efetiva

Como citado anteriormente, a inserção dos campos na equação 2.3 para o posterior cálculo da correlação cruzada podem ser abordados de três maneiras (Dussaud et al., 2008):

1. salvar todos os *snapshots* do campo de onda $P_s(x, z, t)$ e durante o cálculo reverso no tempo de $P_r(x, z, t)$, acessar na memória as posições previamente salvas de $P_s(x, z, t)$;
2. salvar *snapshots* em intervalos de tempo discretos maiores que o passo temporal, recuperando os *snapshots* restantes de $P_s(x, z, t)$ durante a condição da imagem, através de interpolação ou extrapolação (técnica de *checkpointing*, Symes, 2007);
3. salvar os valores do campo de onda apenas nas bordas da malha para todos os passos de tempo, e depois computar o campo de onda nos pontos da parte interna da malha utilizando os valores salvos como condições de contorno.

A abordagem 1, a mais intuitiva, representa o menor custo computacional, e o maior custo de memória. Na opção 3, da borda efetiva, é necessário apenas o armazenamento do campo nas bordas, mas deve ser realizada mais uma modelagem de forma a reconstruir P_s a partir das bordas salvas. Nesse sentido, a opção gasta menos memória e tem maior custo computacional.

Na Tabela 2.2 comparamos as opções 1 e 3 em termos de memória e tempo computacional, para o modelo Marmousi-Teste.

	Memória	Tempo computacional
RTM (1)	975.5 MB	0.46 s
RTM (3)	420.9 MB	0.70 s

Tabela 2.1: *Tradeoff* computacional, RTM (1 tiro).

Nesta dissertação baseamos-nos na abordagem e códigos de Yang et al. (2007), adaptados de uma malha *staggered-grid* para a malha regular descrita no capítulo 1. A estratégia da borda efetiva consiste em, na propagação direta, salvar apenas as bordas do campo da fonte P_s , e seus último dois *snapshots*. Na propagação reversa, o campo P_s é remodelado e reconstruído a partir das bordas previamente salvas. A Figura 2.8 mostra o tamanho e localização da borda efetiva. Sua espessura é de metade do tamanho do operador espacial (por exemplo, se o operador for de oitava ordem, a borda tem espessura 4), a altura é a profundidade do modelo, e a largura é a largura do modelo. A Figura 2.9 mostra a depropagação do campo P_s a partir das bordas.

Em resumo, o algoritmo da RTM por bordas efetivas consiste em:

1. Modela o campo P_s de $t = 0$ a $t = T$, pela equação 2.1 salvando apenas a borda efetiva em cada instante, e os últimos dois instantes completos;
2. Faz a propagação reversa de P_s utilizando as bordas efetivas, e de P_r concomitantemente, ambas segundo a equação 2.2, de $t = T$ a $t = 0$. A cada iteração, acumula a correlação dos campos P_s e P_r (condição de imagem).

Logo, o passo 1 consiste de uma modelagem, e o passo 2 de mais duas modelagens, totalizando o custo computacional de 3 modelagens para a RTM utilizando borda efetiva.

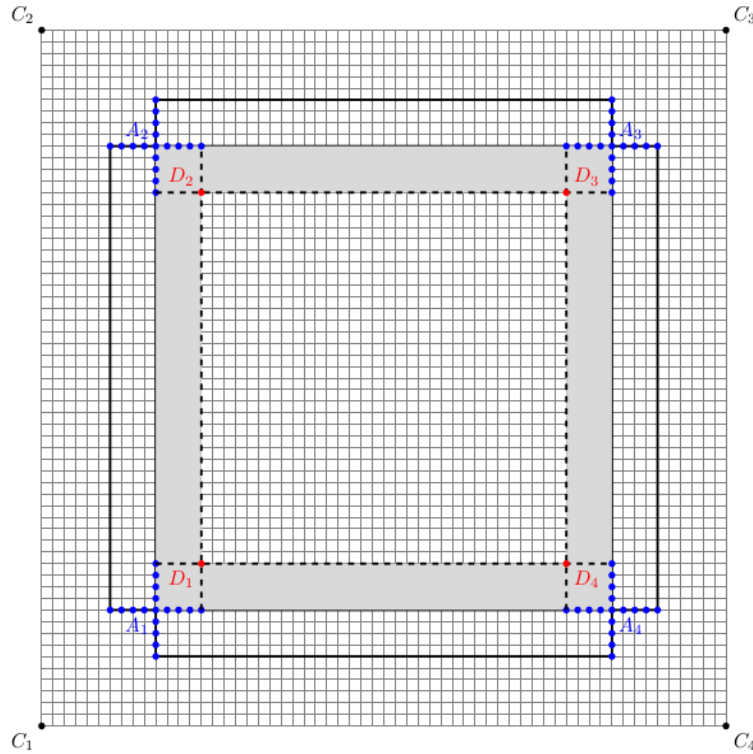


Figura 2.8: Esquematização da malha de modelagem, mostrando a borda absorvedora $A_1A_2A_3A_4 - C_1C_2C_3C_4$; a borda efetiva $D_1D_2D_3D_4 - A_1A_2A_3A_4$ e o modelo (zona de interesse) $A_1A_2A_3A_4$. Note que a borda efetiva está dentro do modelo. Retirado de Yang et. al. (2014).

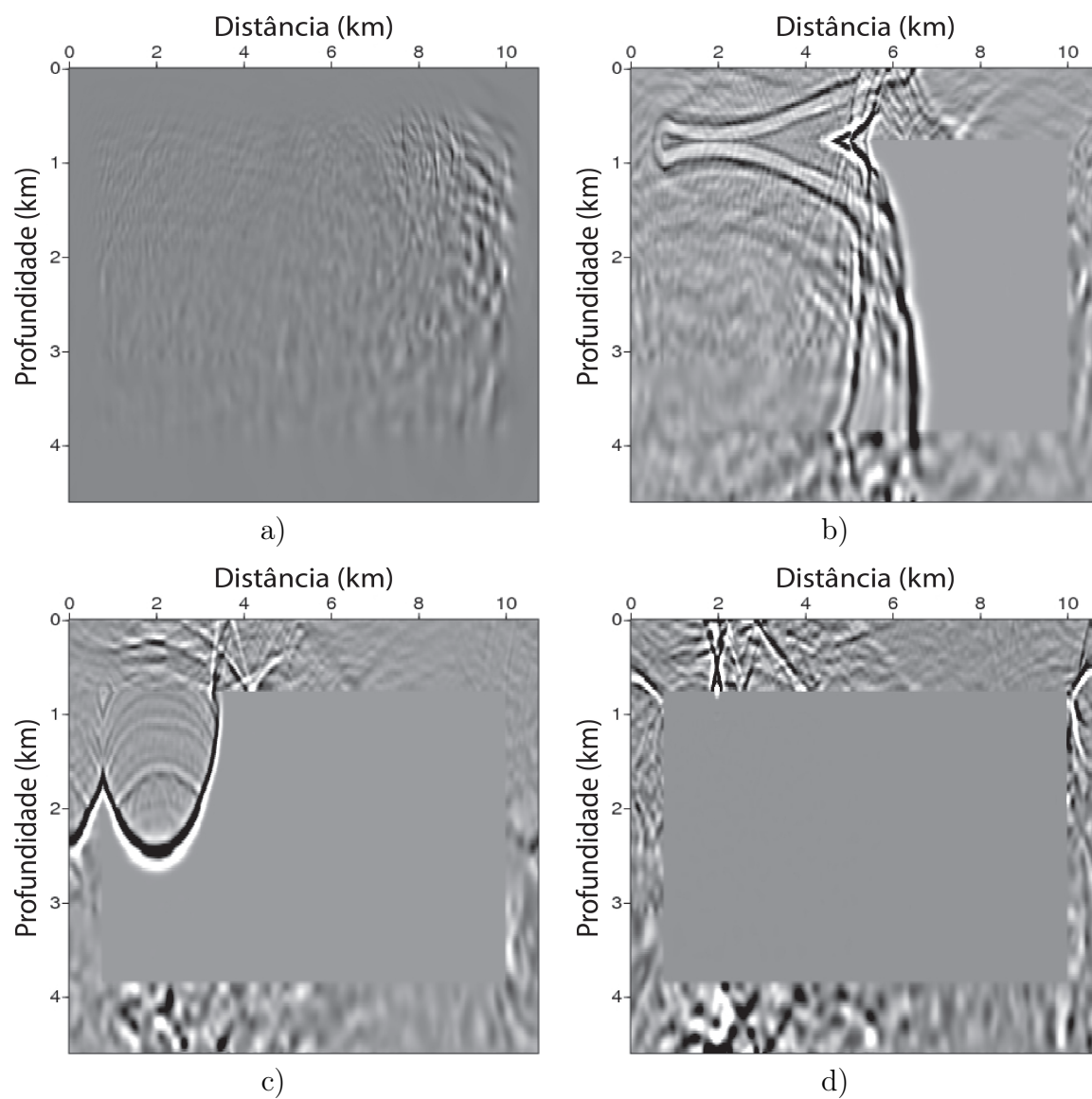


Figura 2.9: Depropagação do campo da fonte utilizando borda efetiva. (a) 3,92 s; (b) 2,2 s; (c) 1 s; (d) 0,02 s.

2.3 Artefatos da condição de imagem de correlação cruzada

A imagem na RTM é tradicionalmente construída pela correlação cruzada no *lag-zero* dos campos extrapolados da fonte e dos receptores. Essa condição de imagem fornece a cinemática correta dos eventos e é muito simples de implementar. Todavia, ela normalmente produz uma grande quantidade de ruído de alta amplitude e baixa frequência que contaminam a imagem (Figura 2.10(a)). Esses ruídos ocorrem porque há a correlação "correta" dos campos (da fonte e dos receptores) no refletor, mas também há correlação desses campos em todo o caminho fonte-refletor-receptores, que podem ser vistos claramente na Figura 2.10(a).

Existem diversas formas de corrigir esse ruído da RTM, e uma das mais simples e efetivas (implementada nesta dissertação) é a aplicação de um filtro Laplaciano na imagem. Esse filtro consiste em substituir a amplitude da imagem em cada ponto pelo seu Laplaciano:

$$I_{corr}(z, x) = \frac{\partial^2 I(z, x)}{\partial z^2} + \frac{\partial^2 I(z, x)}{\partial x^2}. \quad (2.7)$$

A Figura 2.10 -b mostra a efetividade do filtro laplaciano na atenuação das correlações indesejadas (fora dos refletores). Nesta dissertação, em todas as imagens envolvendo RTM, foi aplicado o filtro Laplaciano.

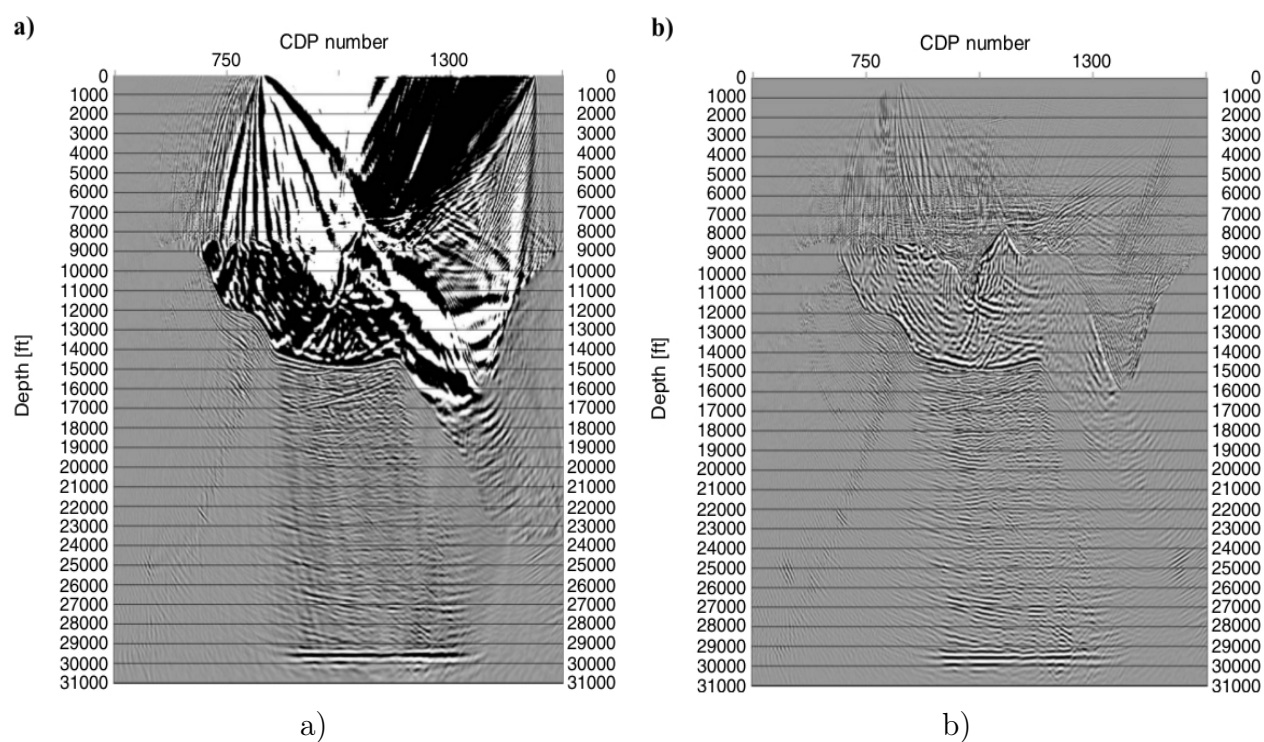


Figura 2.10: Migração (RTM) de um tiro para o modelo Sigsbee 2D. Pode-se ver as correlações indesejadas dos campos acima do refletor, ao longo de todo o caminho fonte-refletor-receptores, em (a); e a atenuação desses artefatos por um filtro laplaciano, em (b). Retirada de Liu et. al. (2011).

3

Inversão da Forma de Onda completa (FWI)

Nesse capítulo é introduzida a teoria da FWI com base na excelente discussão encontrada em Virieux e Operto (2009). Em adição ao desenvolvimento do artigo, são apresentadas as deduções matemáticas para as equações matriciais do método inverso e o gradiente pelo método adjunto (apêndices A e B). No corpo do texto, foi dada ênfase apenas aos pontos fundamentais necessários para entender a implementação da FWI feita nesta dissertação.

A inversão da forma de onda completa, ou *Full Waveform Inversion* (FWI), tem por objetivo obter os parâmetros $\mathbf{m}$ do modelo que descrevem a subsuperfície, a partir do campo $\mathbf{u}$ resultante da excitação desse modelo. O problema é descrito pela equação do problema direto não-linear (Menke, 1984):

$$\mathbf{u} = G(\mathbf{m}), \quad (3.1)$$

onde G descreve a relação físico-matemática entre o campo e os parâmetros do modelo. Na inversão clássica o problema seria resolvido encontrando o operador G^{-1} , de forma que $\mathbf{m}$ seria determinado a partir de $\mathbf{u}$ pela relação $\mathbf{m} = G^{-1}(\mathbf{u})$. Todavia, no caso sísmico em questão, o problema é altamente não-linear, impossibilitando encontrar analiticamente o operador G^{-1} . O caso mais simples do problema 3.1 corresponde à aproximação acústica monoparamétrica (abordado nesta dissertação), onde o $\mathbf{m}$ se restringe às velocidades da onda-P em cada ponto da malha usada para discretizar o problema inverso. No caso mais extremo, $\mathbf{m}$ corresponde aos 21 módulos elásticos que caracterizam um meio triclinico elástico, a densidade, e algumas variáveis que caracterizam o comportamento anelástico do modelo de subsuperfície da terra (Virieux e Operto, 2009).

No contexto da s smica de explora  o, normalmente o campo $\mathbf{u}$   medido ao longo do tempo apenas na superf cie, gerando sismogramas, que denominaremos aqui de dado observado, ou $\mathbf{d}_{\text{obs}}$. Normalmente tamb m se tem acesso a uma primeira estimativa do modelo $\mathbf{m}$, dada pelo conhecimento *a priori* da geologia do local, dados de po o, entre outros; denominamos essa estimativa inicial de $\mathbf{m}_0$. Nesta disserta  o, onde s  foram testados modelos e dados sint ticos, $\mathbf{m}_0$ foi dado por uma suaviza  o do modelo verdadeiro $\mathbf{m}$, como mostra a Figura 3.1. Assim, com base nos dados iniciais dispon veis do problema, $\mathbf{d}_{\text{obs}}$ e $\mathbf{m}_0$, se constr i a formula  o m nimos quadrados da FWI, que consiste em achar o melhor $\mathbf{m}$, em itera  es sucessivas a partir de $\mathbf{m}_0$, que minimize uma fun  o objetivo do tipo (Tarantola, 1984):

$$E(\mathbf{m}) = \frac{1}{2} \Delta \mathbf{d}^T \Delta \mathbf{d}, \quad (3.2)$$

onde

$$\Delta \mathbf{d} = \mathbf{d}_{\text{obs}} - \mathbf{d}_{\text{cal}},$$

e $\mathbf{d}_{\text{cal}} = S_r \mathbf{u}$, sendo $\mathbf{u}$ o campo de ondas modelado sobre o modelo corrente, e S_r o operador que o restringe   geometria dos sismogramas $\mathbf{d}_{\text{obs}}$ em $(\mathbf{x}_r, t)$.

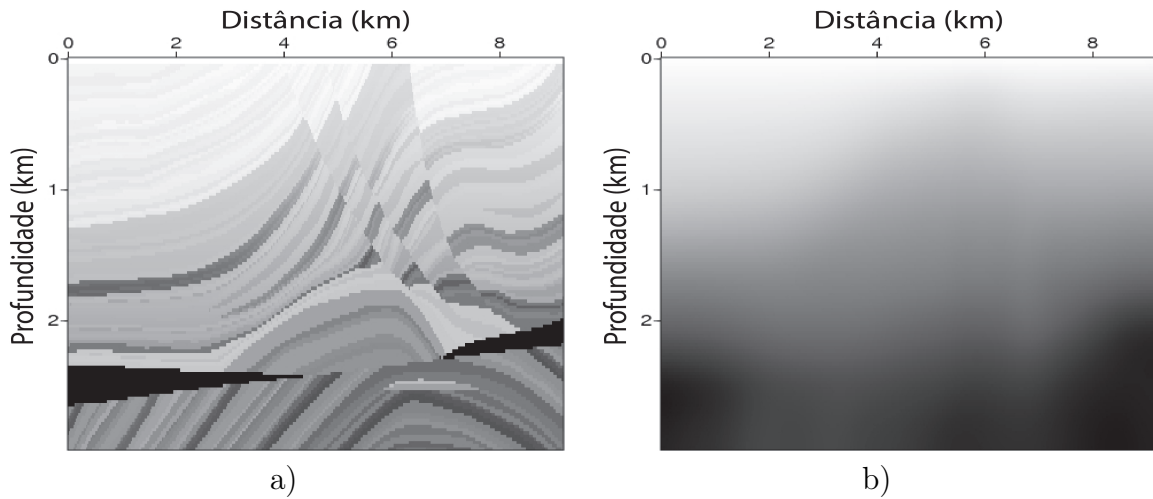


Figura 3.1: Modelo de velocidades original (a) e suavizado (b).

A FWI   um problema altamente n o-linear; por m, fazendo uso da teoria da perturba  o, e utilizando a aproxima  o de Born, pode-se formular a FWI como um problema de otimiza  o local iterativo (Virieux e Operto, 2009). O m todo da perturba  o se assenta na premissa de que uma perturba  o linear no modelo, da forma $\mathbf{m} = \mathbf{m}_0 + \Delta \mathbf{m}$, causa uma perturba  o linear no campo de press o, da forma $\mathbf{u} = \mathbf{u}_0 + \Delta \mathbf{u}$. A aproxima  o de

Born utiliza o campo incidente em lugar do campo total gerado por um espalhador (no caso em questão, espalhadores serão inseridos iterativamente no campo suavizado da Figura 3.1). Uma dedução detalhada por teoria da perturbação, à luz da aproximação de Born, para FWI acústica, pode ser encontrada em Yang et al. (2015).

A matriz das derivadas primeiras de $\mathbf{u}$ em relação aos parâmetros do modelo é chamada matriz sensibilidade, ou matriz de Fréchet, ou matriz Jacobiana. Como será visto a seguir, a Jacobiana de $\mathbf{u}$ é um dos termos fundamentais para se encontrar a perturbação ótima $\Delta\mathbf{m}$, que somada ao modelo corrente, resulta num modelo mais próximo do modelo verdadeiro. O objetivo da inversão, como citado anteriormente, é chegar o mais perto possível desse modelo verdadeiro.

A teoria da perturbação e a aproximação de Born permitem então a formulação mínimos quadrados para a FWI (Tarantola, 1984). Essa formulação consiste em ancorar $\Delta\mathbf{m}$ na função objetivo 3.2, que deve ser minimizada. A ancoragem consiste em aproximar $E(\mathbf{m}_0 + \Delta\mathbf{m})$ em torno de $E(\mathbf{m}_0)$, utilizando uma expansão de Taylor de 2ª ordem:

$$E(\mathbf{m}_0 + \Delta\mathbf{m}) = E(\mathbf{m}_0) + \sum_{j=1}^N \frac{\partial E(\mathbf{m}_0)}{\partial m_j} \Delta m_j + \sum_{l=1}^N \sum_{j=1}^N \frac{\partial^2 E(\mathbf{m}_0)}{\partial m_l \partial m_j} \Delta m_j \Delta m_k + O(\Delta\mathbf{m}^3),$$

$$E(\mathbf{m}_0 + \Delta\mathbf{m}) \approx E(\mathbf{m}_0) + \sum_{j=1}^N \frac{\partial E(\mathbf{m}_0)}{\partial m_j} \Delta m_j + \sum_{l=1}^N \sum_{j=1}^N \frac{\partial^2 E(\mathbf{m}_0)}{\partial m_l \partial m_j} \Delta m_j \Delta m_k, \quad (3.3)$$

sendo N o número total de parâmetros do modelo. Os termos $O(\Delta\mathbf{m}^3)$ são desprezados. Observe que a equação (3.3) descreve um parabolóide em $\Delta\mathbf{m}$. O mínimo da aproximação em torno de $\mathbf{m}_0$ é o ponto no qual:

$$\frac{\partial E(\mathbf{m})}{\partial m_i} = 0, \quad (3.4)$$

para todos os i de $1, \dots, N$ (ver Figura 3.2, para o caso de 2 parâmetros).

Derivando (3.3) com relação a m_i , e aplicando 3.4, obtemos:

$$\frac{\partial E(\mathbf{m}_0 + \Delta\mathbf{m})}{\partial m_i} = \frac{\partial E(\mathbf{m}_0)}{\partial m_i} + \sum_{j=1}^N \frac{\partial^2 E(\mathbf{m}_0)}{\partial m_i \partial m_j} \Delta m_j + \sum_{l=1}^N \sum_{j=1}^N \frac{\partial^3 E(\mathbf{m}_0)}{\partial m_i \partial m_l \partial m_j} \Delta m_j \Delta m_k = 0; \quad (3.5)$$

como as derivadas terceiras são nulas para a aproximação de Taylor de segunda ordem,

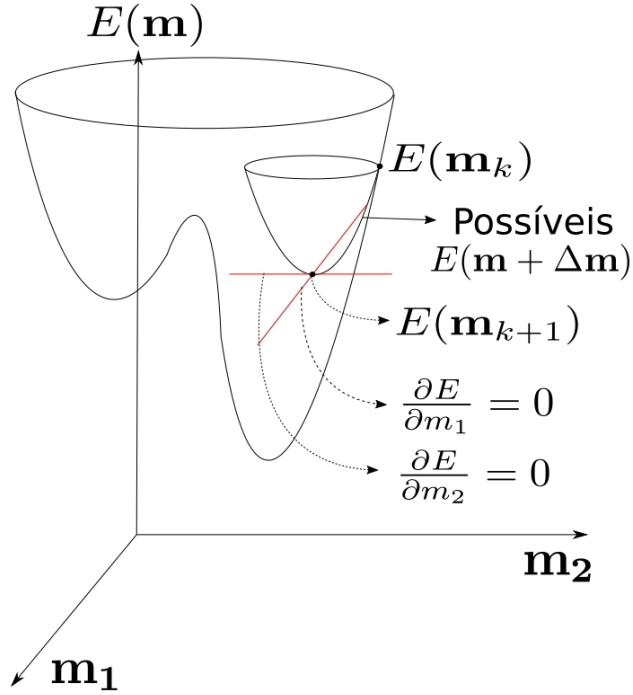


Figura 3.2: Esboço da função objetivo não-linear $E(\mathbf{m})$ para um caso de 2 parâmetros (m_1 e m_2). $E(\mathbf{m} + \Delta \mathbf{m})$ é um parabolóide em função de $\Delta \mathbf{m}$. Em vermelho, derivadas nulas em cada parâmetro dão o ponto de mínimo.

$$\sum_{j=1}^N \frac{\partial^2 E(\mathbf{m}_0)}{\partial m_i \partial m_j} \Delta m_j = -\frac{\partial E(\mathbf{m}_0)}{\partial m_i}. \quad (3.6)$$

Organizando para todos os m_i , em notação matricial, tem-se:

$$\begin{bmatrix} \frac{\partial^2 E(m_0)}{\partial m_1^2} & \frac{\partial^2 E(m_0)}{\partial m_1 \partial m_2} & \cdots & \frac{\partial^2 E(m_0)}{\partial m_1 \partial m_N} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 E(m_0)}{\partial m_N \partial m_1} & \frac{\partial^2 E(m_0)}{\partial m_N \partial m_2} & \cdots & \frac{\partial^2 E(m_0)}{\partial^2 m_N} \end{bmatrix} \begin{bmatrix} \Delta m_1 \\ \vdots \\ \Delta m_N \end{bmatrix} = - \begin{bmatrix} \frac{\partial E(m_0)}{\partial m_1} \\ \vdots \\ \frac{\partial E(m_0)}{\partial m_N} \end{bmatrix}, \quad (3.7)$$

$$\frac{\partial^2 E(\mathbf{m}_0)}{\partial \mathbf{m}^2} \Delta \mathbf{m} = -\frac{\partial E(\mathbf{m}_0)}{\partial \mathbf{m}}, \quad (3.8)$$

$$\Delta \mathbf{m} = \left[\frac{\partial^2 E(m_0)}{\partial \mathbf{m}^2} \right]^{-1} \left(-\frac{\partial E(\mathbf{m}_0)}{\partial \mathbf{m}} \right) + O(\mathbf{m}^3), \quad (3.9)$$

sendo $\frac{\partial^2 E(\mathbf{m}_0)}{\partial \mathbf{m}^2}$ e $\frac{\partial E(\mathbf{m}_0)}{\partial \mathbf{m}}$ a matriz Hessiana H_E e o gradiente $\nabla E(\mathbf{m}_0)$ da função objetivo, respectivamente. A direção $\left(-\frac{\partial E(\mathbf{m}_0)}{\partial \mathbf{m}} \right)$ é oposta à direção de variação máxima no ponto $E(\mathbf{m}_0)$ (denominada gradiente, ou *steepest ascent*) e é chamada *steepest descent*. Pode-se

concluir então que a perturbação $\Delta \mathbf{m}$ em (3.9), que leva ao mínimo do problema mínimos quadrados local, é "iniciada" na direção *steepest descent*, e sofre uma transformação linear dada por H_E^{-1} , gerando a direção que minimiza o problema local.

No caso de um problema linear $\mathbf{d} = \mathbf{G}\mathbf{m}$, a aproximação (3.3) seria exata, e o passo $\Delta \mathbf{m} = H_E^{-1}(-\nabla E(\mathbf{m}_0))$ levaria ao mínimo global em apenas uma iteração (Virieux e Operto, 2009). Todavia a FWI é não-linear, necessitando de diversas iterações (com uma aproximação de segunda ordem a cada iteração, em torno do mínimo do parabolóide da iteração anterior) para (possivelmente) convergir ao mínimo global.

3.1 Os métodos de Newton, quasi-Newton, Gauss-Newton e *Steepest-Descent*

Para entender cada método primeiro precisamos expressar o passo $\Delta \mathbf{m}$ dado por (3.9) em termos do dado calculado $\mathbf{d}_{cal}$, explicitando e derivando $E(\mathbf{m}) = \frac{1}{2}(\mathbf{d}_{obs} - \mathbf{d}_{cal})^T(\mathbf{d}_{obs} - \mathbf{d}_{cal})$.

O gradiente $\nabla E(\mathbf{m})$ é dado por:

$$\nabla E(\mathbf{m}) = \frac{\partial E(\mathbf{m})}{\partial \mathbf{m}} = \frac{1}{2} \frac{\partial}{\partial \mathbf{m}} \Delta \mathbf{d}^T \Delta \mathbf{d} = \frac{1}{2} \left(-\frac{\partial \mathbf{d}_{cal}^T}{\partial \mathbf{m}} \Delta \mathbf{d} - \Delta \mathbf{d}^T \frac{\partial \mathbf{d}}{\partial \mathbf{m}} \right),$$

então

$$\nabla E(\mathbf{m}) = -\frac{\partial \mathbf{d}_{cal}^T}{\partial \mathbf{m}} \Delta \mathbf{d}, \quad (3.10)$$

ou

$$\nabla E(\mathbf{m}) = -J^T \Delta \mathbf{d}, \quad (3.11)$$

sendo J a Jacobiana dos dados calculados, ou matriz sensibilidade, ou matriz de Fréchet.

Derivando o gradiente $\nabla E(\mathbf{m})$, que é uma matriz coluna, em relação a cada parâmetro de $\mathbf{m}$, e organizando essas colunas ordenadamente em uma matriz (NxN), temos a Hessiana de $E(\mathbf{m})$ em função de $\mathbf{d}_{cal}$ (para uma dedução completa, ver apêndice C):

$$H_E = \frac{\partial^2 E(\mathbf{m})}{\partial \mathbf{m}^2} = J^T J + \frac{\partial J^T}{\partial \mathbf{m}^T} [\Delta \mathbf{d}, \dots, \Delta \mathbf{d}]. \quad (3.12)$$

Substituindo as expressões de $\nabla E(\mathbf{m})$, H_E em função de $\mathbf{d}$ em $\Delta\mathbf{m}$, temos $\Delta\mathbf{m}$ em função de $\mathbf{d}_{cal}$:

$$\Delta\mathbf{m} = - \left\{ J^T J + \frac{\partial J^T}{\partial \mathbf{m}^T} [\Delta\mathbf{d}, \dots, \Delta\mathbf{d}] \right\}^{-1} J^T \Delta\mathbf{d}. \quad (3.13)$$

Quando $\Delta\mathbf{m}$ é calculado pela expressão (3.13) a cada iteração, temos o **Método de Newton**. Todavia para FWI o método de Newton é proibitivo, pois para os N parâmetros em $\mathbf{m}$ seriam necessárias N perturbações e N modelagens para o cálculo de $\nabla E(\mathbf{m})$, e N^2 modelagens para H_E (ver seção B.2 do apêndice B).

Para exemplificar, o modelo Marmousi-Teste (definido ao final da Introdução), de dimensões $n_z = 125$ e $n_x = 383$ amostras possui $N = 47875$ parâmetros. Para um tiro, usando o método de diferenças finitas de décima ordem, o tempo de modelagem, já paralelizado em GPU, é de aproximadamente 0,2s. Como a Jacobiana requer uma modelagem completa (todos os 28 tiros) para cada um dos 47875 parâmetros, o tempo necessário seria de $N \times n_{shots} \times 0,2 = 47875 \times 28 \times 0,2 = 74,4$ h. Já a Hessiana necessitaria de N^2 modelagens; nesse exemplo, o cálculo levaria aproximadamente 400 anos ($N^2 \times n_{shots} \times 0,2$). Dessa forma, executar a FWI calculando a Hessiana é inviável, sendo necessário o uso de aproximações, como será visto adiante. O cálculo da Jacobiana diretamente também é contraprodutivo, visto que cada iteração da FWI, nesse exemplo, já paralelizado em GPU, levaria 74 h; por isso, também são necessários métodos de aproximação da Jacobiana, que também serão abordados nessa seção.

Em casos mais complexos da FWI, muitas vezes N é da ordem de 10^6 , tornando a inversão pelo Método de Newton inviável computacionalmente, mesmo em supercomputadores e/ou clusters de GPUs. Daí a necessidade do uso de métodos que calculem $\Delta\mathbf{m}$ de maneira aproximada, como os métodos de **Gauss-Newton**, **quasi-Newton** e **Steepest-Descent**.

Os **métodos de quasi-Newton** visam aproximar a hessiana inversa tanto em informações das primeiras derivadas (termo $J^T J$) quanto das segundas derivadas (termo $\frac{\partial J^T}{\partial \mathbf{m}^T} [\Delta\mathbf{d}, \dots, \Delta\mathbf{d}]$), e por conterem informações dessas segundas derivadas, normalmente são mais precisos que os métodos de Gauss-Newton e *steepest-descent*. Nos testes dessa dissertação foi aplicado o método quasi-Newton L-BFGS, apresentado adiante nesse capítulo.

Quando o termo com segundas derivadas ($\frac{\partial J^T}{\partial \mathbf{m}^T}$) é desprezado (Yang et al., 2015; Pratt et al., 1998) se tem a seguinte expressão para $\Delta\mathbf{m}$:

$$\Delta\mathbf{m} = - \left\{ J^T J \right\}^{-1} J^T \Delta\mathbf{d}, \quad (3.14)$$

ou

$$\Delta \mathbf{m} = -H_a^{-1} J^T \Delta \mathbf{d}, \quad (3.15)$$

onde $H_a = J^T J$ é chamada Hessiana aproximada. Quando se calcula $\Delta \mathbf{m}$ pela equação acima, a cada iteração, tem-se o **Método de Gauss-Newton** (Virieux e Operto, 2009). Nos testes dessa dissertação não foi utilizado o método de Gauss-Newton, apenas quasi-Newton e *steepest-descent*.

3.2 Métodos *steepest-descent*

Outra alternativa é substituir H_a^{-1} por um escalar α (Virieux e Operto, 2009):

$$\Delta \mathbf{m} = -\alpha \nabla E(\mathbf{m}) = -\alpha (J^T \Delta \mathbf{d}). \quad (3.16)$$

Quando $\Delta \mathbf{m}$ é calculado pela expressão (3.16) a cada iteração, temos o **Método *Steepest-Descent***, base dos demais Métodos *Steepest-Descent*. Desta forma o problema de encontrar o mínimo do parabolóide em $\Delta \mathbf{m}$ se reduz a encontrar o mínimo da parábola, função de α , na direção *steepest-descent*.

Substituindo então (3.16) em (3.3), e usando notação matricial, resulta em:

$$E(\mathbf{m}_0 + \Delta \mathbf{m}) = E(\mathbf{m}_0) - \alpha \nabla E(\mathbf{m})^T \nabla E(\mathbf{m}) + \frac{\alpha^2}{2} \nabla E(\mathbf{m})^T H_E \nabla E(\mathbf{m}). \quad (3.17)$$

Utilizando a condição de mínimo $\frac{\partial E(\mathbf{m})}{\partial \alpha} = 0$, e a aproximação $H_E \approx H_a = J^T J$, temos que:

$$\alpha = \frac{\nabla E(\mathbf{m})^T \nabla E(\mathbf{m})}{(\nabla E(\mathbf{m})^T J)^T (J \nabla E(\mathbf{m}))}, \quad (3.18)$$

ou

$$\alpha = \frac{\langle \nabla E(\mathbf{m}), \nabla E(\mathbf{m}) \rangle}{\langle J \nabla E(\mathbf{m}), J \nabla E(\mathbf{m}) \rangle}, \quad (3.19)$$

onde o termo $\langle \cdot, \cdot \rangle$ significa produto interno. A equação 3.19 é a expressão para o α no Método *Steepest-Descent*. Substituir a Hessiana pelo escalar α faz com que a função objetivo

E seja função de α , ou seja, $E = E(\alpha)$ no contexto dessa aproximação local. Em problemas não lineares, que é o caso da FWI, $E(\alpha)$ apresenta vários mínimos locais, de forma que a equação 3.19 pode gerar pouca convergência (ou até divergência) do processo. Os métodos que otimizam o α , que gere um menor $E(\alpha)$, são denominados *métodos de busca em linha*, e os critérios mais utilizados nessa busca são as condições de Wolfe (dos Santos, 2013). A equação padrão 3.19 nos dá uma primeira estimativa do α , que pode ser otimizado pelos métodos de busca em linha. Além disso, é possível ainda otimizar essa estimativa por diversos métodos (que podem usar ou não informações de iterações anteriores) como por exemplo interpolação quadrática, método de Barzilai-Borwein, comprimento do passo máximo e médio, entre outros (Nocedal e Wright, 2006).

Para os testes realizados, utilizou-se o método gradiente, conjugado-gradiente e L-BFGS. Nos testes dos métodos gradiente e conjugado-gradiente não foi utilizado nenhum método para melhorar o passo inicial, nem nenhum critério de busca em linha. Por esse motivo, não apresentamos sua fundamentação teórica; leitores interessados devem consultar dos Santos (2013), que contém uma excelente discussão teórica e prática dos métodos e sua comparação com o L-BFGS. Como mostrado em sua dissertação, o L-BFGS, um método quasi-Newton, é mais eficiente e menos custoso que os métodos gradiente otimizados, e é atualmente o método mais utilizado. Por esse motivo, os principais resultados foram obtidos utilizando o L-BFGS, e os resultados pelo método gradiente e conjugado-gradiente foram executados para efeito de comparação. Além disso, como será visto a seguir, o L-BFGS precisa ser inicializado por algum método gradiente, justificando a importância prática do entendimento dessas abordagens.

Além da otimização do α , é possível melhorar a direção de busca $\mathbf{s}$. Nos métodos Pré-condicionado Gradiente, Conjugado Gradiente e Pré-condicionado Conjugado Gradiente (Tarantola, 1987) a direção de busca é otimizada por

$$\Delta \mathbf{m} = -\alpha \mathbf{s}, \quad (3.20)$$

sendo $\mathbf{s}$ a direção otimizada. Uma dedução similar nos dá então, para os demais métodos gradiente:

$$E(\mathbf{m}_0 + \Delta \mathbf{m}) = E(\mathbf{m}_0) - \alpha \nabla E^T \mathbf{s} + \frac{\alpha^2}{2} \mathbf{s}^T H_E \mathbf{s}.$$

$$\alpha = \frac{\langle \nabla E(\mathbf{m}), \mathbf{s} \rangle}{\langle J\mathbf{s}, J\mathbf{s} \rangle},$$

Recapitulando, nos métodos gradiente o modelo $\mathbf{m}$ é atualizado por:

$$\nabla E(\mathbf{m}) = -J^T \Delta \mathbf{d}, \quad (3.21)$$

$$\alpha = \frac{\langle \nabla E(\mathbf{m}), \mathbf{s} \rangle}{\langle J\mathbf{s}, J\mathbf{s} \rangle}, \quad (3.22)$$

$$\mathbf{m} = \mathbf{m}_0 - \alpha \mathbf{s}. \quad (3.23)$$

Então a cada iteração, deve-se calcular $\nabla E(\mathbf{m})$, $\mathbf{s}$ e o termo $J\mathbf{s}$. Com isso se calcula o α e o modelo é atualizado por (3.23).

3.2.1 Cálculo do gradiente $\nabla E(\mathbf{m})$

Como mostra a equação (3.21), para o cálculo exato do gradiente seria necessária a matriz sensibilidade. Um modelo de N parâmetros exigiria N perturbações e N modelagens. Tal procedimento é muito custoso; uma alternativa mais prática e viável computacionalmente consiste em calcular o gradiente pelo método adjunto (Tarantola, 1987). A partir dele, o gradiente pode ser calculado no domínio do tempo através da seguinte relação (ver Apêndice C):

$$\nabla E(\mathbf{m}) = \frac{2}{v^3} \int_0^{t_{max}} \lambda \frac{\partial^2 P_{cal}}{\partial t^2} dt \quad (3.24)$$

onde P_{cal} são os campos de onda modelados por alguma solução numérica da equação da onda acústica (capítulo 1) e λ é a variável adjunta, que corresponde aos campos de onda do resíduo $\Delta \mathbf{d}$ injetado na superfície e depropagado pela relação (ver Apêndice C):

$$\frac{1}{v(\mathbf{x})^2} \frac{\partial^2 \lambda(\mathbf{x}, t)}{\partial t^2} - \nabla^2 \lambda(\mathbf{x}, t) = \Delta \mathbf{d}(x, z = 0, t). \quad (3.25)$$

A relação (3.24) é análoga a uma migração reversa no tempo, na qual ao invés da correlação cruzada usual do campo propagado da fonte e campo depropagado dos receptores, temos a correlação cruzada da derivada segunda do campo propagado da fonte e o campo depropagado do resíduo dos receptores. Assim, a cada iteração da FWI é realizado um processo de custo computacional similar à RTM, para o cálculo de $\nabla E(\mathbf{m})$.

De forma simplória, pode-se dizer que o gradiente (RTM do resíduo) contém as estruturas que estão no modelo verdadeiro, mas não estão no modelo inicial, já que as estruturas presentes nos dois modelos produzirão os mesmos eventos nos dados observados e calculados, e, conseqüentemente, não estarão presentes no resíduo entre eles. As Figuras 3.3, 3.4 e 3.5,

retiradas de dos Santos (2013), exemplificam como o gradiente recupera feições do modelo verdadeiro, que não estão no modelo inicial.

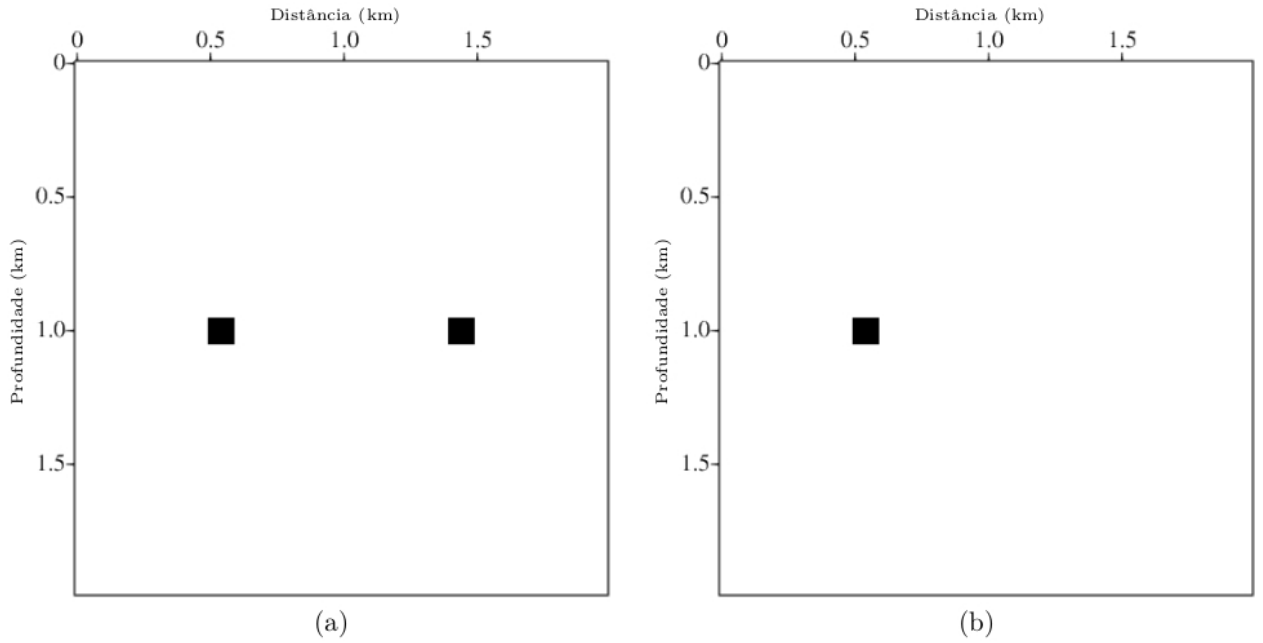


Figura 3.3: a) Modelo verdadeiro com velocidade de fundo 2000 m/s e dois espalhadores de velocidade 2500 m/s. b) Modelo inicial, com apenas um espalhador. Retirado de dos Santos (2013).

3.2.2 Cálculo da Jacobiana J na direção $\mathbf{s}$

A determinação do passo α exige o cálculo de $J\mathbf{s}$ (equação 3.22). Como discutido anteriormente, o cálculo da matriz sensibilidade J é muito custoso, e por vezes inviável, na FWI. A matriz sensibilidade completa necessitaria do cálculo da sensibilidade em todas as direções, exigindo N perturbações e, portanto, N modelagens. Todavia é possível calcular o termo $J\mathbf{s}$ fazendo apenas uma modelagem, observando que o mesmo se trata da sensibilidade de $\mathbf{d}_{cal}$ apenas na direção $\mathbf{s}$. Sendo G o operador de modelagem, o termo $J\mathbf{s}$ pode ser aproximado pela expansão de Taylor de primeira ordem de $G(\mathbf{m} + \epsilon\mathbf{s})$ em torno de $G(\mathbf{m})$:

$$J\mathbf{s} = \frac{\partial G(\mathbf{m})}{\partial \mathbf{m}} \mathbf{s} \approx \frac{G(\mathbf{m} + \epsilon\mathbf{s}) - G(\mathbf{m})}{\epsilon}, \quad (3.26)$$

e o passo ϵ dessa modelagem dado por (Pica et al., 1990):

$$\max(\epsilon|\mathbf{s}|) \leq \frac{\max(|\mathbf{m}|)}{100}. \quad (3.27)$$

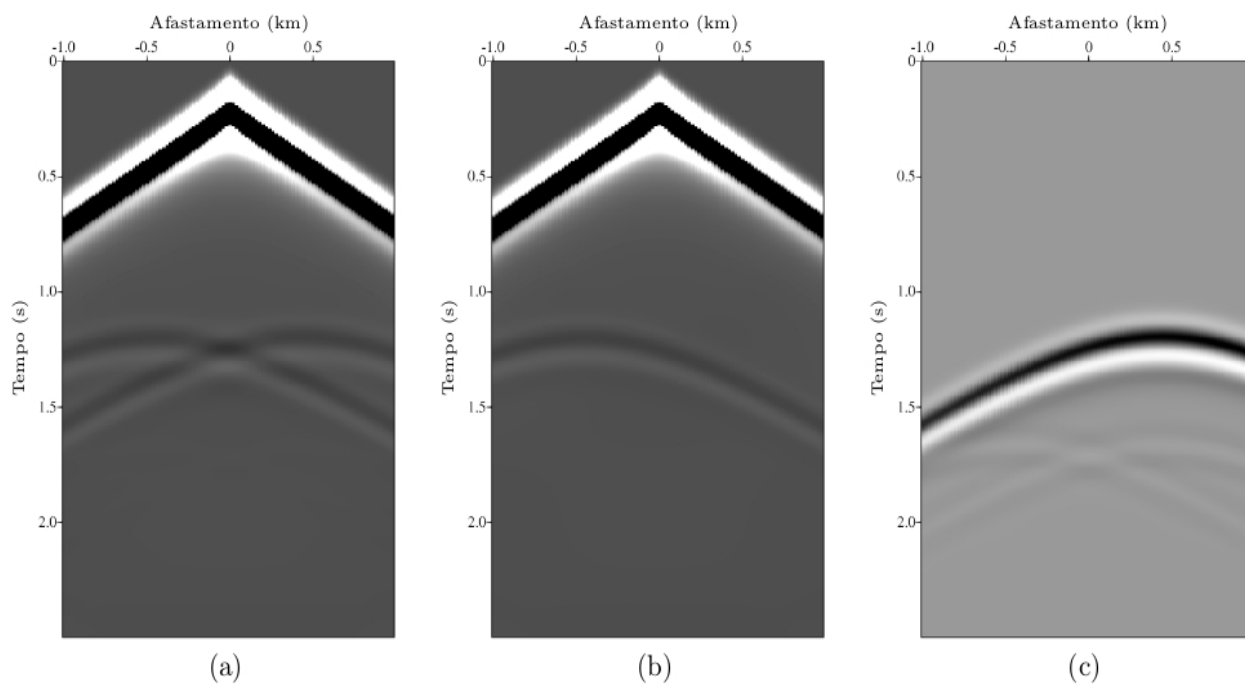


Figura 3.4: Sismogramas sintéticos gerados com: a) modelo verdadeiro; b) modelo inicial. c) Diferença entre (a) e (b) (resíduo). Retirado de dos Santos (2013).

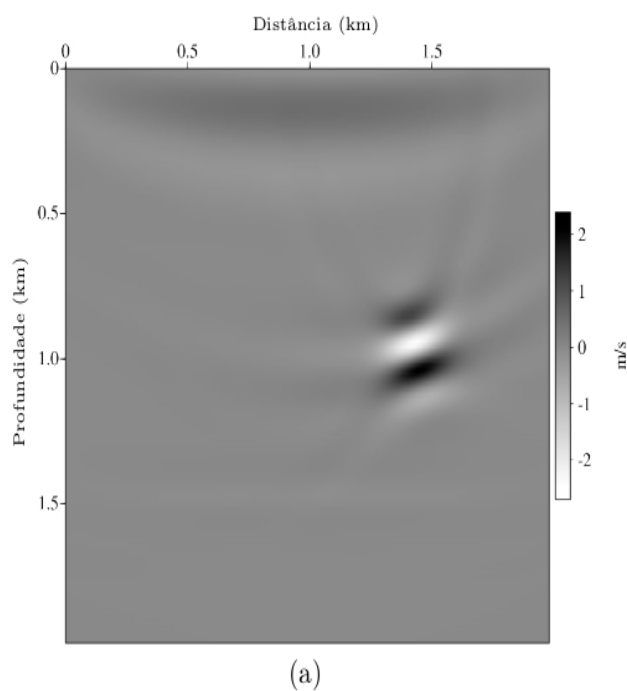


Figura 3.5: Migração do resíduo da Figura 3.4 c, que é o gradiente da função objetivo E pelo método adjunto. Retirado de dos Santos (2013).

Assim, a cada iteração da FWI é realizado um processo de custo idêntico a uma modelagem sísmica para o cálculo do termo J_s . Essa etapa, mais a etapa do cálculo do gradiente

(equivalente em custo à uma RTM), respondem por quase todo o custo computacional por iteração da FWI.

Pode-se então resumir uma iteração dum método gradiente como:

- calcula o gradiente pelo método adjunto (equações 3.24 e 3.25);
- a partir do gradiente, calcula a direção $\mathbf{s}$ (dada por $\nabla E(\mathbf{m})$ no método gradiente; dada pela equação 3.28 no método conjugado-gradiente);
- calcula o termo $J\mathbf{s}$ pelas equações 3.26 e 3.27;
- calcula α pela equação 3.22;
- atualiza $\mathbf{m}$ pela equação 3.23.

3.2.3 Método Conjugado Gradiente Não Linear

Para uma dedução mais detalhada, recomenda-se a leitura de Porsani (2008) ou Nocedal e Wright (2006); aqui são apresentadas apenas as equações essenciais à implementação. Nesse método, o modelo é atualizado iterativamente na direção $\mathbf{s}_k$, que é combinação linear do gradiente na iteração corrente $\nabla E(\mathbf{m}_k)$ e da direção anterior $\mathbf{s}_{k-1}$ (na primeira iteração, a direção é o próprio gradiente):

$$\mathbf{s}_k = \nabla E(\mathbf{m}_k) + \beta_k \mathbf{s}_{k-1}. \quad (3.28)$$

O escalar β_k garante que $\mathbf{s}_k$ e $\mathbf{s}_{k-1}$ são conjugados. Dentre as várias formas de calcular o escalar β_k , a fórmula de Polak-Ribière é a mais usada para FWI (Virieux e Operto, 2009):

$$\beta_k = \frac{\langle \nabla E(\mathbf{m}_k) - \nabla E(\mathbf{m}_{k-1}), \nabla E(\mathbf{m}_k) \rangle}{\langle \nabla E(\mathbf{m}_k), \nabla E(\mathbf{m}_k) \rangle}, \quad (3.29)$$

Apenas três vetores de dimensão N , $\nabla E(\mathbf{m}_k)$, $\nabla E(\mathbf{m}_{k-1})$ e $\mathbf{s}_{k-1}$ são necessários a cada iteração.

3.3 Método L-BFGS

Nessa seção são apresentadas apenas as equações básicas e o algoritmo do método quasi-Newton L-BFGS, baseado em dos Santos (2013). Uma discussão teórica aprofundada pode ser encontrada em Nocedal e Wright (2006).

Os métodos de Gauss-Newton fazem uso duma Hessiana aproximada H_a sem contribuição das derivadas segundas da função objetivo E (no caso da FWI, as derivadas de $\mathbf{d}_{\text{cal}}$), usando apenas o termo $J^T J$, ou uma aproximação dele. Já os métodos de quasi-Newton usam uma Hessiana aproximada $\mathbf{B}$ que contém aproximações tanto das derivadas primeiras quanto das derivadas segundas, mas sem precisar calculá-las explicitamente. Essas abordagens quasi-Newton fazem uso das informações obtidas nas iterações anteriores, uma vez que as mudanças apresentadas no gradiente ao longo das iterações carregam informação sobre as segundas derivadas de E (Nocedal e Wright, 2006). Dessa forma, uma aproximação $\mathbf{B}_k$ é atualizada na k -ésima iteração, utilizando informação de gradientes e modelos de iterações anteriores a k para aproximar a matriz Hessiana.

O método BFGS (nomeado a partir dos seus criadores, Broyden, Fletcher, Goldfarb e Shanno) define a atualização da matriz $\mathbf{B}_{k+1}^{-1}$ (aproximação da Hessiana), calculada na k -ésima iteração, como:

$$\mathbf{B}_{k+1}^{-1} = \mathbf{B}_k^{-1} - \frac{\mathbf{B}_k^{-1} \mathbf{y}_k \mathbf{y}_k^T \mathbf{B}_k^{-1}}{\mathbf{y}_k^T \mathbf{B}_k^{-1} \mathbf{y}_k} + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}. \quad (3.30)$$

Entretanto, a matriz $\mathbf{B}$ terá N^2 elementos, sendo N o número de parâmetros do modelo. No caso da FWI, este valor normalmente está muito acima da capacidade de armazenamento de dados dos computadores atuais. Assim, uma versão do BFGS foi criada para que pudesse ser aplicada a problemas de larga escala. Essa versão ficou conhecida como L-BFGS (BFGS de memória limitada, ou *limited-memory* BFGS). Com essa metodologia, não é preciso ter a matriz $\mathbf{B}$ explicitamente armazenada, apenas um número m de iterações passadas, i.e., os modelos e gradientes. Portanto, o custo de armazenamento é reduzido de N^2 para $2mN$, com m sendo normalmente um número menor que 30 (normalmente, se utilizam as 10 iterações anteriores, $m = 10$). Por exemplo, para um modelo de velocidades 3D com $100 \times 100 \times 100$ elementos e $m = 20$, o método L-BFGS demandará 25 mil vezes menos armazenamento que o BFGS (dos Santos, 2013).

Define-se, para a k -ésima iteração,

$$\begin{aligned} \mathbf{x}_k &= \mathbf{m}_{k+1} - \mathbf{m}_k, \\ \mathbf{y}_k &= \mathbf{g}_{k+1} - \mathbf{g}_k, \end{aligned} \quad (3.31)$$

onde $\mathbf{m}$ é o modelo de parâmetros e $\mathbf{g}$ o gradiente, e $\mathbf{x}$ e $\mathbf{y}$ são cubos contendo m diferenças entre modelos/gradientes consecutivos, respectivamente. O algoritmo para o L-BFGS é apresentado a seguir, adaptado de Nocedal e Wright (2006). Note que os *loops* do algoritmo sempre começam/terminam em $k - 1$, de forma que as expressões 3.31 sempre tratam da

informação corrente k e da iteração anterior $k - 1$.

Algoritmo 1: Método L-BFGS

```

q  $\leftarrow$  gk;
para  $i = k - 1, k - 2, \dots, k - m$  faça
     $\epsilon_i \leftarrow \sigma_i \mathbf{x}_i^T \mathbf{q}$ ;
    q  $\leftarrow$  q  $- \epsilon_i \mathbf{y}_i$ ;
fim para
r  $\leftarrow \gamma_k \mathbf{q}$ ;
para  $i = k - m, k - m + 1, \dots, k - 1$  faça
     $\beta \leftarrow \sigma_i \mathbf{y}_i^T \mathbf{r}$ ;
    r  $\leftarrow$  r  $+ \mathbf{x}_i (\epsilon_i - \beta)$ ;
fim para

```

Onde:

$$\sigma_k = \frac{1}{\mathbf{y}_k^T \mathbf{x}_k}, \quad (3.32)$$

$$\gamma_k = \frac{\mathbf{x}_{k-1}^T \mathbf{y}_{k-1}}{\mathbf{y}_{k-1}^T \mathbf{y}_{k-1}}. \quad (3.33)$$

O fator γ_k serve como um operador de escala, que tenta aproximar os valores de $\mathbf{B}_k$ dos valores da Hessiana. Dessa forma, o método L-BFGS em geral é muito melhor escalonado que o *steepest-descent*, e o comprimento do passo $\alpha_k = 1$ é normalmente a melhor escolha (dos Santos, 2013).

Ao fim da execução do algoritmo, a direção de busca $\mathbf{s}_k$ será dada pelo vetor $-\mathbf{r}$.

3.4 Normalização das amplitudes

A solução do problema direto através da equação da onda acústica não é capaz de reproduzir corretamente as amplitudes do dado obtido em uma aquisição sísmica (Virieux e Operto, 2009). Portanto, para que se possa comparar os dois dados é preciso aplicar um operador de normalização a um deles, de forma que as amplitudes sejam compatíveis e o resíduo entre eles tenha significado consistente (dos Santos, 2013).

Uma forma simples de obter esse efeito é aplicar um fator escalar (Wang, Gao, Zhang e Zhao, 2011):

$$\rho = \frac{\mathbf{d}_{cal}^T \mathbf{d}_{obs}}{\mathbf{d}_{cal}^T \mathbf{d}_{cal}}, \quad (3.34)$$

de forma que o dado calculado normalizado é $\mathbf{d}_{calN} = \rho \mathbf{d}_{cal}$, e o resíduo deve ser dado por $\Delta \mathbf{d} = \mathbf{d}_{obs} - \mathbf{d}_{calN}$.

3.5 Determinação da assinatura da fonte

Um dos aspectos que devem ser considerados na solução do problema direto na FWI é a determinação da assinatura do termo fonte, ou *wavelet*. A modelagem de um dado com uma fonte diferente da usada no dado observado produzirá diferenças nos sismogramas que não são provenientes de erros do modelo de velocidades, o que pode gerar artefatos indesejados no resultado da inversão. Portanto, é preciso dispor de um meio para estimar, a partir do dado observado, a *wavelet* usada na sua geração, quando esta informação não estiver disponível (dos Santos, 2013).

A alternativa mais simples para isso é assumir que a primeira reflexão registrada nos receptores mais próximos à fonte conseguem manter a forma de onda da *wavelet*. Assim, se for possível isolar este evento no traço sísmico, assume-se que ele é uma boa representação da *wavelet*. Outras estratégias mais sofisticadas envolvem a estimação da *wavelet* usando um conjunto maior de dados, seja através de um problema inverso (Shin et al., 2007) ou de métodos estatísticos (Baan, 2008).

Neste trabalho a *wavelet* usada na geração dos dados, tanto observado quanto calculado, é sempre conhecida, dispensando a aplicação dos métodos citados anteriormente.

3.6 Pré-condicionamento do gradiente

Como visto anteriormente (Capítulo 3 e apêndice C), o cálculo do gradiente da FWI pelo método adjunto equivale à RTM do resíduo (a menos de um fator $\frac{2}{v^3}$). Sendo o gradiente uma RTM, é possível melhorá-lo, ou em outras palavras, pré-condicioná-lo com uma condição de imagem diferente da usual, como por exemplo a condição de imagem com compensação do campo de onda da fonte (equação 2.6).

Assim, com base na equação da condição de imagem para a RTM com compensação do campo da fonte (equação 2.6, seção 2.1), e a equação do gradiente pelo método adjunto (equação 3.24, seção 3.2.1), tem-se a equação para o gradiente pré-condicionado $\nabla E_{PC}(\mathbf{m})$ (Yang et al., 2015), utilizada nesta dissertação:

$$\nabla E_{PC}(\mathbf{m}) = \frac{\nabla E(\mathbf{m})}{\sqrt{\sum_s \int_0^{t_{max}} P_{cal}^2 dt + \gamma^2}}, \quad (3.35)$$

onde γ é um fator de estabilização para evitar divisão por zero.

Na Figura 3.6 são mostrados resultados de 120 iterações da FWI multiescala, para 28 tiros no modelo Marmousi-Teste, com e sem pré-condicionamento. Pode-se ver que os resultados são muito parecidos, mas analisando o erro RMS de cada um em relação ao modelo verdadeiro (Figura 3.6-c), percebe-se a superioridade da inversão utilizando o gradiente pré-condicionado. Outros testes com condições de imagem diferentes no pré-condicionamento do gradiente podem ser encontrados em Santos (2017), Kaelin et al. (2007).

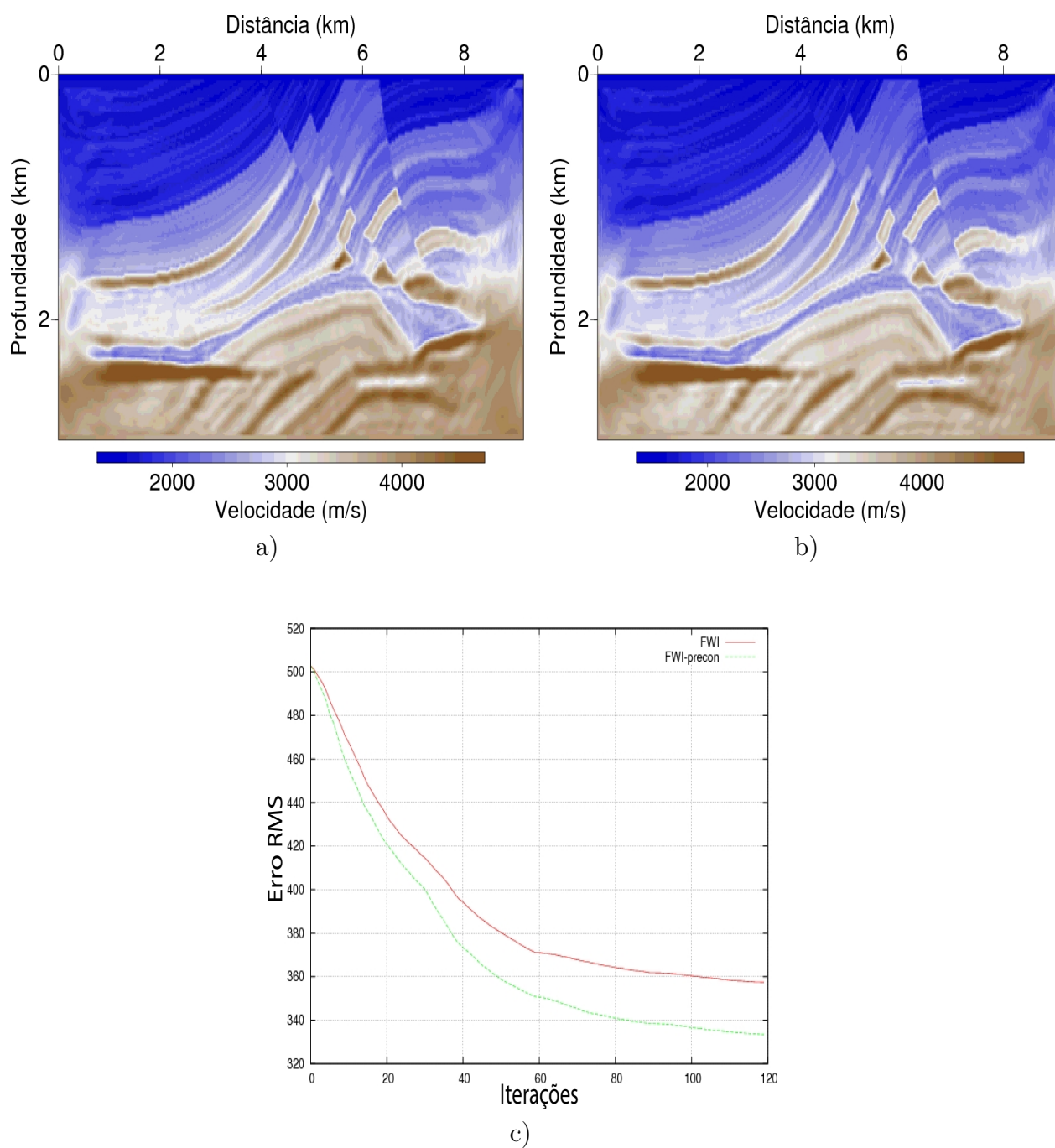


Figura 3.6: Influência do pré-condicionamento do gradiente na FWI. a) Resultado sem pré-condicionamento. b) Resultado com pré-condicionamento. c) Erro RMS total para os resultados (a) e (b).

3.7 Abordagem Multiescala

A inversão de forma de onda no domínio do tempo, tal como descrita no capítulo anterior, está sujeita à convergência para mínimos locais, devido à alta não-linearidade do problema (Virieux e Operto, 2009). Entretanto, devido ao alto custo computacional da FWI, a utilização de métodos de busca global, como, por exemplo, o algoritmo genético e o *Simulated Annealing* (Kirkpatrick et al., 1983), ainda é inviável. Dessa forma, é preciso fazer uso de estratégias de inversão capazes de reduzir as chances de convergência para mínimos locais, a um custo computacional aceitável. Dentre estas estratégias, está a abordagem multiescala.

É válido observar que a abordagem multiescala não é um método de inversão distinto, é apenas uma forma de organizar/selecionar os parâmetros do modelo e do dado observado. Na estratégia apresentada por Bunks et al. (1995), a escala é a escala do comprimento de onda λ , de λ_{max} a λ_{min} , que envolve então decompor v de v_{min} a v_{max} e/ou f de f_{min} a f_{max} , visto que $\lambda = v/f$. Sabe-se que um campo de onda responde a variações na subsuperfície cujo tamanho mínimo é aproximadamente metade do comprimento de onda λ (Virieux e Operto, 2009); assim, se espera que, numa abordagem multiescala que inicie a inversão nos comprimentos de onda maiores, sejam delineadas as feições maiores do modelo, e progressivamente as feições menores, à medida que se reduza a escala de λ . A forma de delimitar o λ , e consequentemente as escalas do problema, utilizada nesta dissertação (e também por Bunks et al., 1995) é delimitar a frequência f do dado observado; Bunks também propõe reduzir o modelo de velocidades v , mas nesta dissertação foi utilizada apenas a seleção de bandas de frequência, ou seja, a multiescala na frequência.

A vantagem do método consiste no fato de que nas escalas maiores há muito menos mínimos locais, de forma que é possível se aproximar com certa segurança do mínimo global, e posteriormente refinar o processo, chegando próximo a esse mínimo global e obtendo um modelo de alta resolução (como mostra a Figura 3.7, retirada de Bunks et al., 1995). A desvantagem é que o custo computacional é maior, visto que o processo convencional realiza a inversão da banda completa de frequências para um certo número k de iterações, e a abordagem multiescala realiza k iterações para cada escala (banda de frequência), em média. Por exemplo, para uma FWI de 30 iterações, um processo multiescala que utilize 4 bandas de frequência realizará 120 iterações, e levará 4 vezes mais tempo. Todavia, como mostra a seção "Etapas da abordagem multiescala e exemplificação" a superioridade dos resultados compensa o custo computacional.

Os mínimos locais da função objetivo são causados principalmente por dois efeitos (ver Figura 3.8, retirada de Bunks et al., 1995). Primeiro, quando o modelo inicial de parâmetros

(no caso aqui tratado de FWI acústico, os parâmetros se reduzem às velocidades da onda P) origina grandes erros cinemáticos entre dado observado e dado calculado (uma perturbação no modelo de velocidades não afeta a soma quadrada dos resíduos), fazendo com que o gradiente seja zero sem que se esteja no mínimo global. Segundo, os componentes de alta frequência da fonte geram uma função correlação que é multimodal (Bunks et al., 1995). Em outras palavras, são geradas correlações errôneas de componentes de onda plana do ciclo posterior/anterior dos campos de onda observado/calculado, fenômeno denominado *cycle skipping*, o que explica boa parte dos mínimos locais na função objetivo (ver Figura 3.9 retirada de Virieux e Operto, 2009).

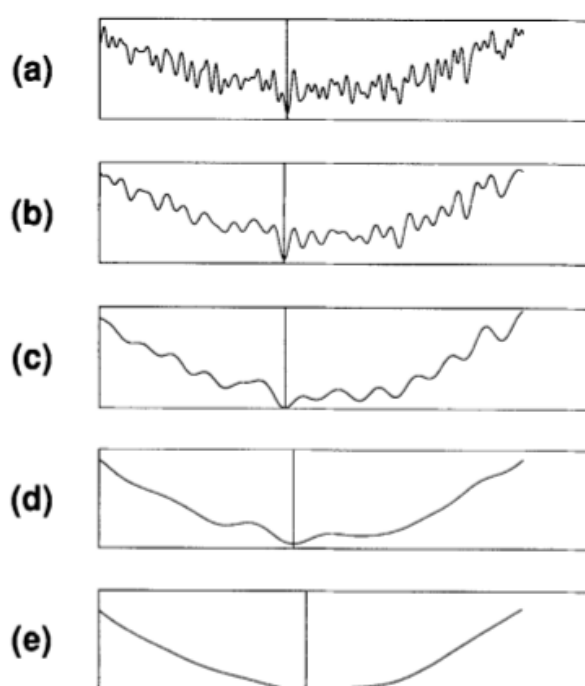


Figura 3.7: Interpretação heurística da abordagem multiescala. Cada painel representa o gráfico de uma função objetivo, desde a menor escala (maior frequência) (a) até a maior escala (menor frequência) (e). Adaptado de Bunks et. al. (1995).

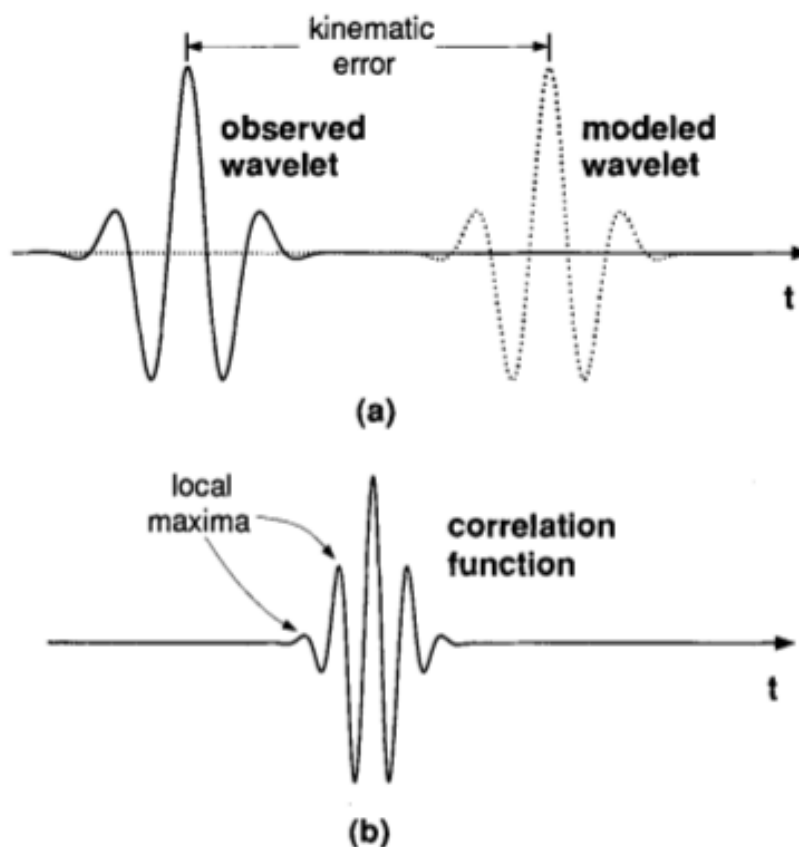


Figura 3.8: Causas de mínimos locais na função objetivo para FWI. (a) Erros cinemáticos grandes fazem que o gradiente seja erroneamente nulo; (b) Altas frequências geram máximos locais na correlação da função objetivo (*cycle-skipping*). Retirado de Bunks et. al. (1995).

3.7.1 Filtragem do dado observado

No domínio do tempo, a seleção das frequências requer uma filtragem no dado observado, a fim de delimitar a banda que será utilizada em cada etapa. Também é preciso garantir que o dado modelado tenha o mesmo conteúdo de frequência do dado observado, para que o resíduo entre eles seja significativo. Uma forma de garantir isso é filtrando a fonte sísmica usada na modelagem, de forma que tenha o mesmo conteúdo de frequência do dado observado filtrado (ou se a *wavelet* for conhecida, utilizá-la na geração do dado calculado com uma frequência máxima que corresponda à banda da rodada atual). Dessa forma, os dados modelados não necessitarão ser filtrados novamente (Bunks et al., 1995).

Existem diversos tipos de filtros que podem ser usados para limitar a banda de frequência de um sinal. Neste trabalho, se pretende utilizar o filtro de Wiener, que é definido no domínio da frequência por Boonyasiriwat et al. (2010):

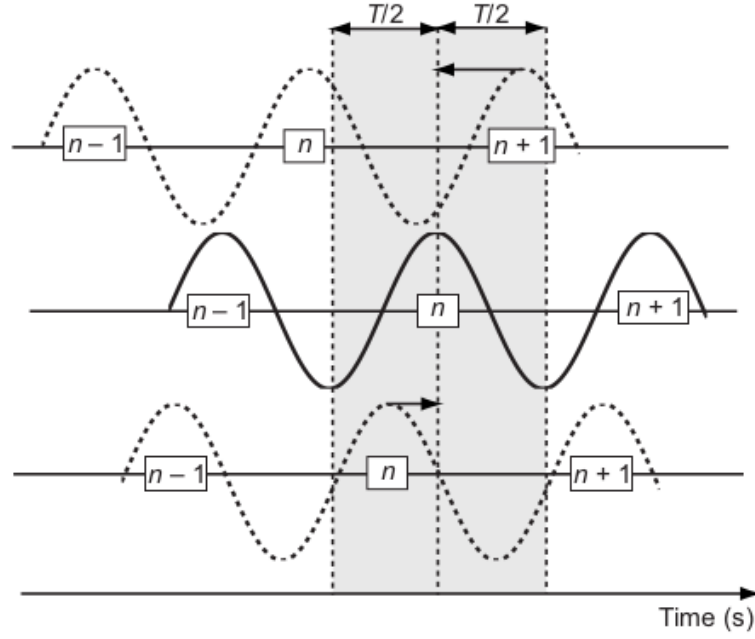


Figura 3.9: Formação de artefatos de *cycle-skipping*, para uma componente de onda monocromática, na FWI. A linha sólida representa uma componente do dado observado, e as linhas pontilhadas, do dado modelado, ambas com um *delay* maior que $T/2$. No caso da modelagem superior, a FWI vai atualizar o modelo como se houvesse correlação entre o ciclo ($n+1$) dos sismogramas modelados e o ciclo n do dado observado, gerando um modelo errôneo. No caso da modelagem inferior, o ciclo n do dado modelado vai correlacionar com o ciclo n do dado observado, gerando uma atualização correta do modelo. O limite máximo do *delay* para que não ocorra *cycle-skipping* é de $T/2$. Retirado de Virieux e Operto (2009).

$$f_{Wiener}(\omega) = \frac{W_{alvo}W_{original}^*(\omega)}{|W_{original}(\omega)|^2 + \epsilon^2}, \quad (3.36)$$

onde f_{Wiener} é o filtro que será aplicado ao dado, ω a frequência temporal, $W_{original}$ é a *wavelet* original do dado, W_{alvo} é a *wavelet* com conteúdo de frequência limitado, $*$ é o índice que indica complexo conjugado e ϵ é um número pequeno usado para estabilizar a divisão (dos Santos, 2013). Aqui, neste trabalho, para cada banda de frequência, o fator f_{Wiener} foi aplicado traço a traço.

A Figura 3.10, retirada de dos Santos (2013), mostra o efeito do filtro de Wiener num tiro dum dado observado de frequência máxima 30 Hz. Observa-se, pelos sismogramas e bandas de frequência, que o dado filtrado é equivalente a um dado modelado com uma *wavelet* de 6 Hz.

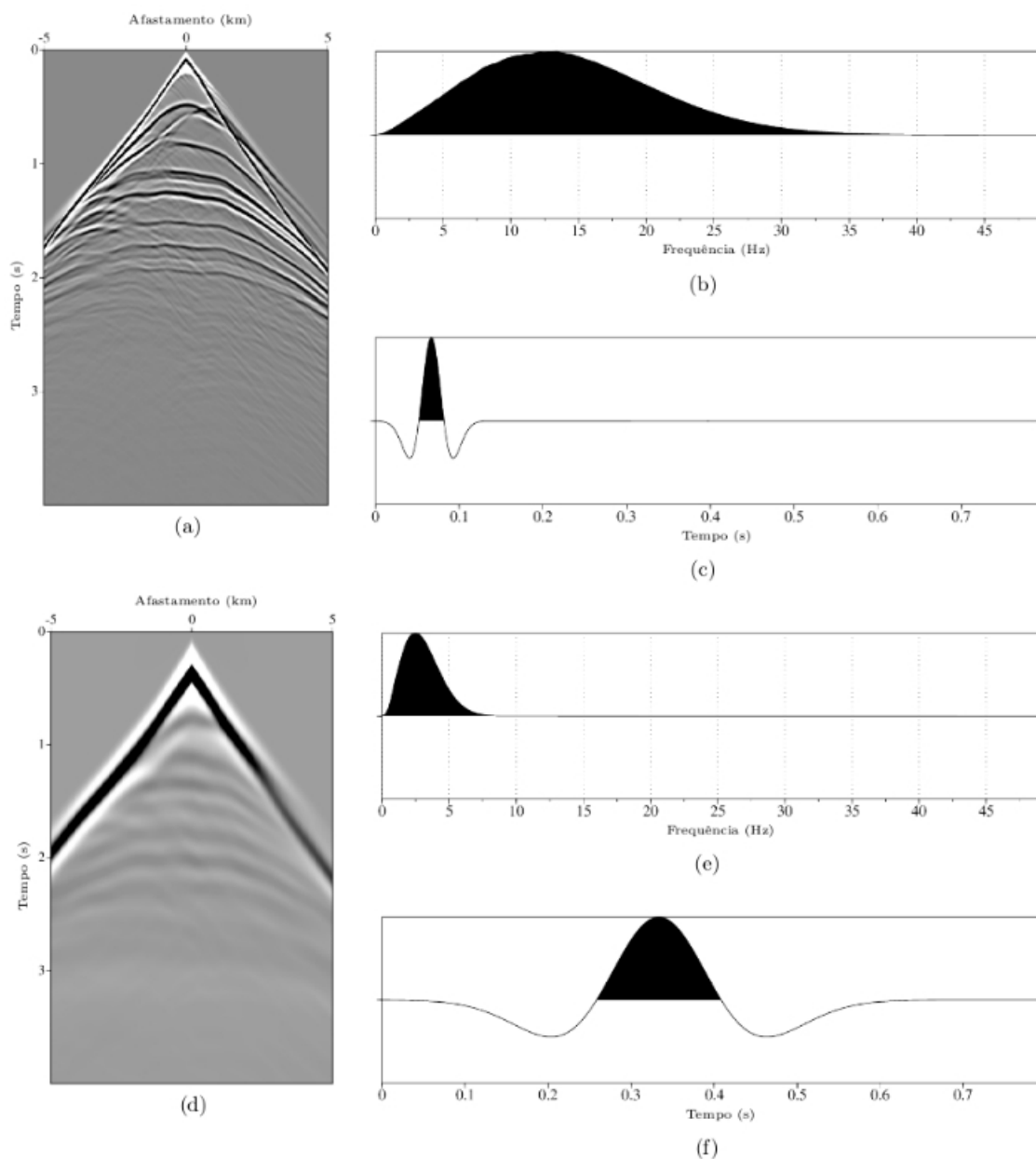


Figura 3.10: Efeitos do filtro de Wiener: nos sismogramas, (a) 30 Hz e (d) filtrado a 6 Hz; no espectro de amplitude dos sismogramas, (b) 30 Hz e (e) filtrado 6 Hz; *wavelet* de 30 Hz usada para gerar o dado observado, em (c); *wavelet* de 6 Hz, usada no filtro de Wiener. Retirado de dos Santos (2013).

3.7.2 Etapas da abordagem multiescala e exemplificação

As etapas para realizar a abordagem multiescala para a FWI no domínio do tempo podem ser resumidas em (adaptado de dos Santos, 2013):

- Para a banda de frequências atual:
 - (a) Filtrar o dado observado;

- (b) Utilizar uma *wavelet* com frequência máxima correspondente à banda atual, para gerar os dados modelados;
- (c) Realizar k iterações da FWI;
- Definir o resultado da rodada atual como modelo inicial;
- Aumentar a banda de frequências.

Esse ciclo é repetido até que o número de bandas de frequência planejado tenha sido atingido ou algum outro critério de parada tenha sido satisfeito.

Para exemplificar como o processo de inversão de toda a banda de frequências acaba convergindo para um mínimo local, e que a abordagem multiescala contorna o problema, foram feitas três inversões para um modelo Marmousi-Teste suavizado (Figura 3.11-b), visando obter o modelo Marmousi verdadeiro da Figura 3.11-a. Se utilizou o dado observado base do Marmousi-Teste (ver Introdução), com frequência máxima de 30 Hz, em cada um dos três processos abaixo:

1. 30 iterações da banda completa, de 0 – 30 Hz;
2. 30 iterações numa banda de baixa frequência, de 0 – 7,5 Hz;
3. abordagem multiescala, onde foram feitas 120 iterações, 30 para cada banda: de 0 – 7,5 Hz; de 0 – 15 Hz; de 0 – 22,5 Hz e de 0 – 30 Hz.

A Figura 3.11 mostra o modelo verdadeiro e o modelo inicial para a FWI. Na Figura 3.12 são mostrados resultados da inversão da banda de frequência completa, de 0 – 30 Hz, 30 iterações. É possível perceber que a função objetivo converge, mas o erro RMS entre modelo verdadeiro e estimado diverge; isso indica que a função objetivo, na verdade, convergiu para um mínimo local. Percebe-se que o resultado nas partes mais profundas do modelo é ruim comparado ao modelo verdadeiro.

Na Figura 3.13 é mostrada a inversão para a banda $(0 - 7,5)Hz$, com o intuito de exemplificar como as baixas frequências primeiro delineam as grandes feições do modelo (3.13-d), contornando os mínimos locais: a função objetivo converge, e o erro RMS decresce, indicando proximidade do mínimo global, como esquematizado na Figura 3.7. Nessa dissertação conveciona-se o erro RMS como a raiz quadrada da soma total dos erros entre cada ponto do modelo verdadeiro e do modelo estimado, ou seja, o erro RMS total (sem normalização), como mostrado na equação:

$$E_{RMS}(\mathbf{m}^{inv}) = \sqrt{\sum_i^N (m_i^{inv} - m_i^{ver})^2}. \quad (3.37)$$

Por fim, na Figura 3.14 é mostrada a inversão para as 4 bandas de frequência, até $30Hz$. Há convergência da função objetivo em cada rodada, e todas as rodadas (120 iterações, 30 por rodada) são plotadas juntas, na Figura 3.13-a. O erro RMS também decresce, indicando que não houve "aprisionamento" em mínimo local. O resultado final, na Figura 3.14-d, é muito mais próximo do modelo verdadeiro que a inversão de toda a banda (sem multiescala) apresentada na Figura 3.11-d, justificando o custo computacional mais elevado da abordagem multiescala.

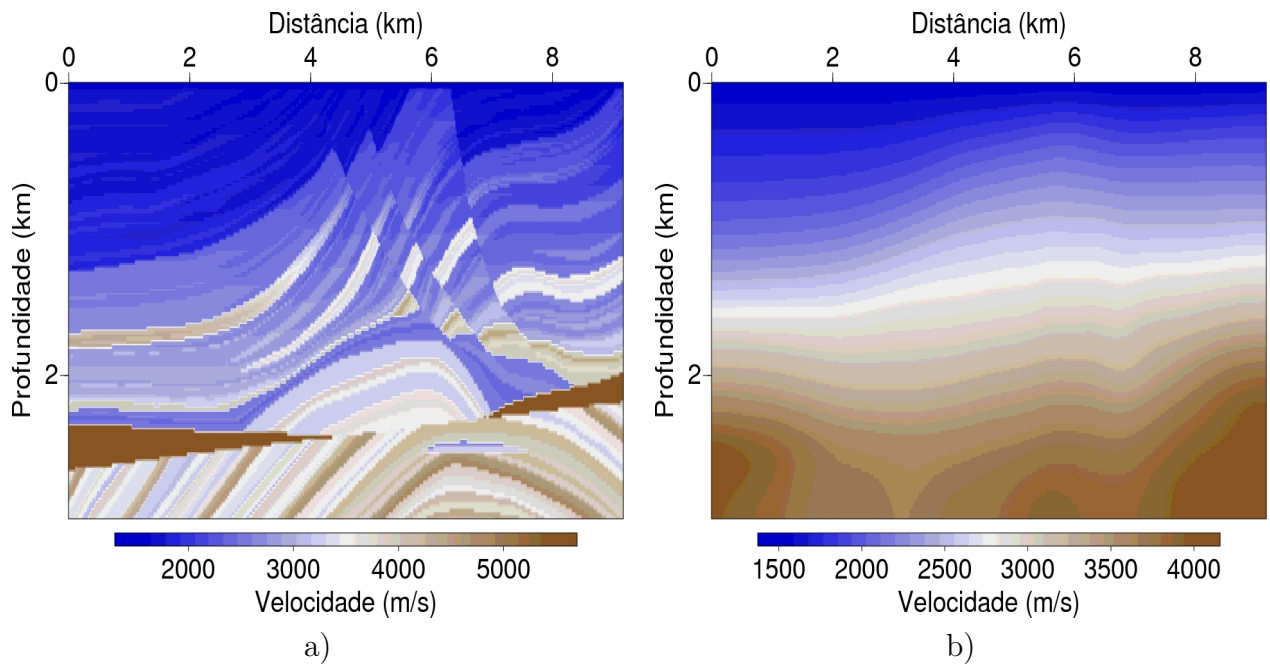


Figura 3.11: Modelo Marmousi de 125×383 amostras, a) verdadeiro e b) suavizado.

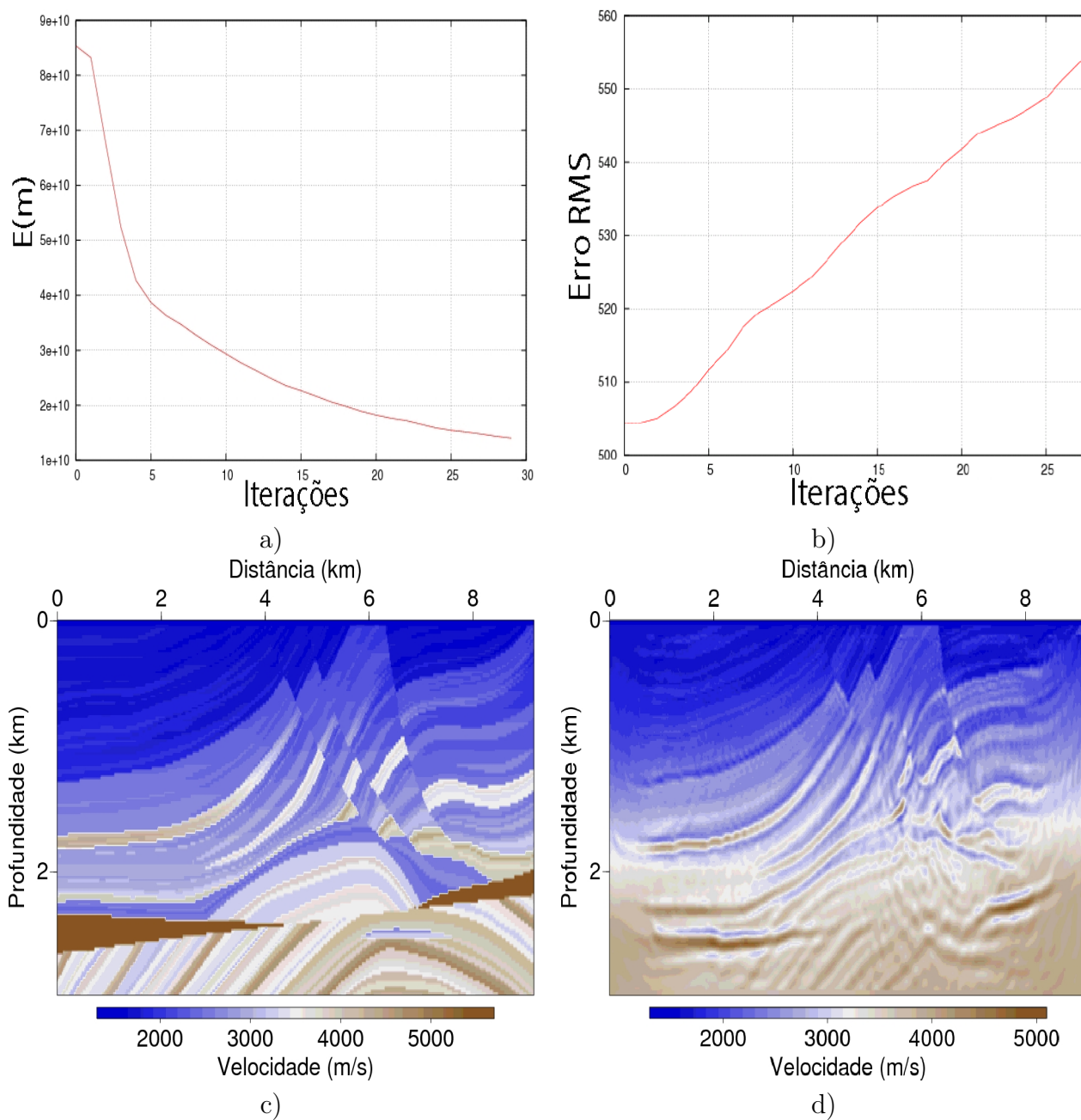


Figura 3.12: FWI de toda a banda de frequência, de 0 – 30 Hz, do dado observado. a) Função objetivo, por iteração; b) Erro RMS entre modelo verdadeiro e estimado, por iteração; c) modelo verdadeiro e d) modelo estimado final.

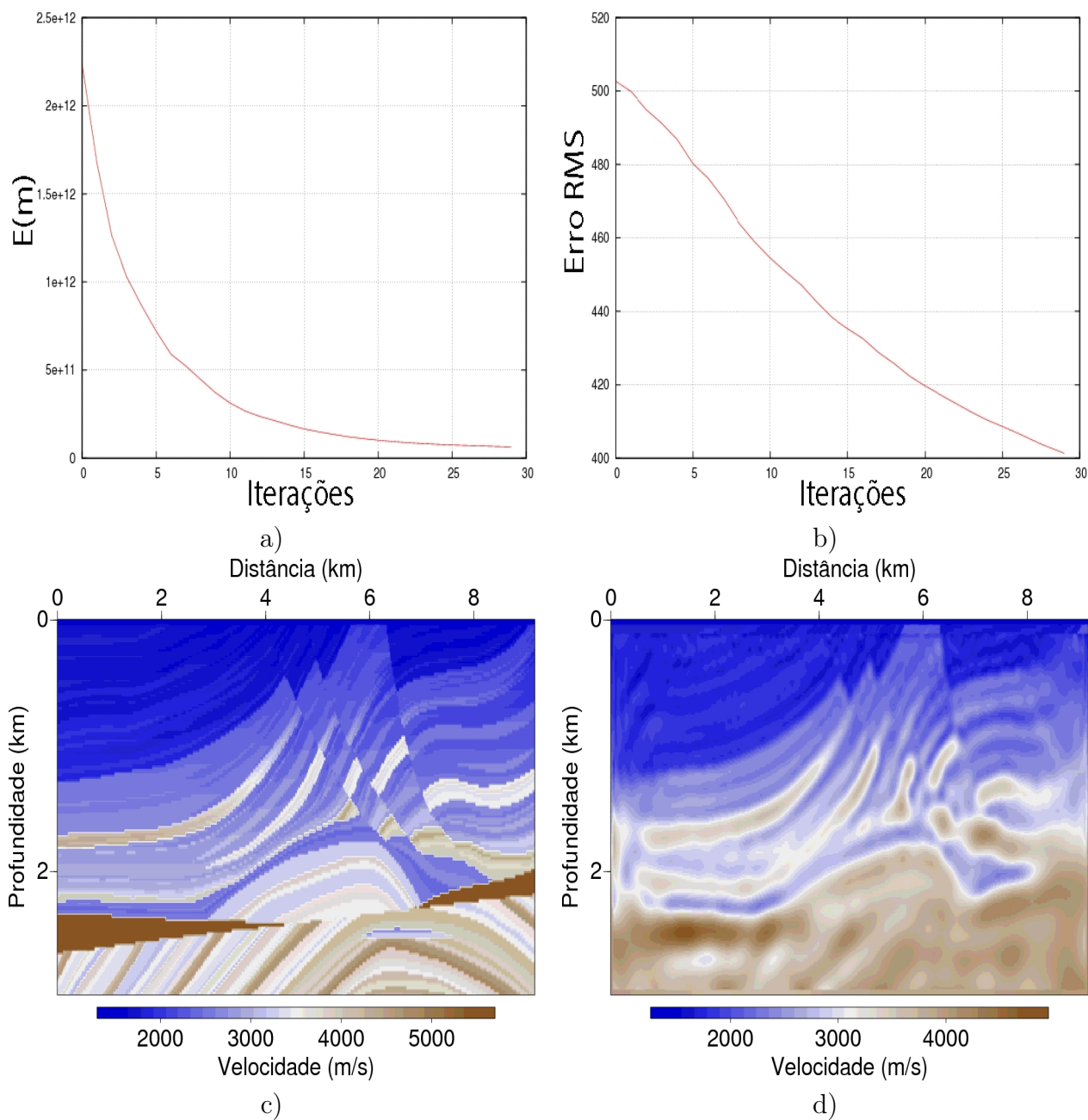


Figura 3.13: FWI da banda de frequência de 0 – 7,5 Hz. a) Função objetivo, por iteração; b) Erro RMS entre modelo verdadeiro e estimado, por iteração; c) modelo verdadeiro e d) modelo estimado final.

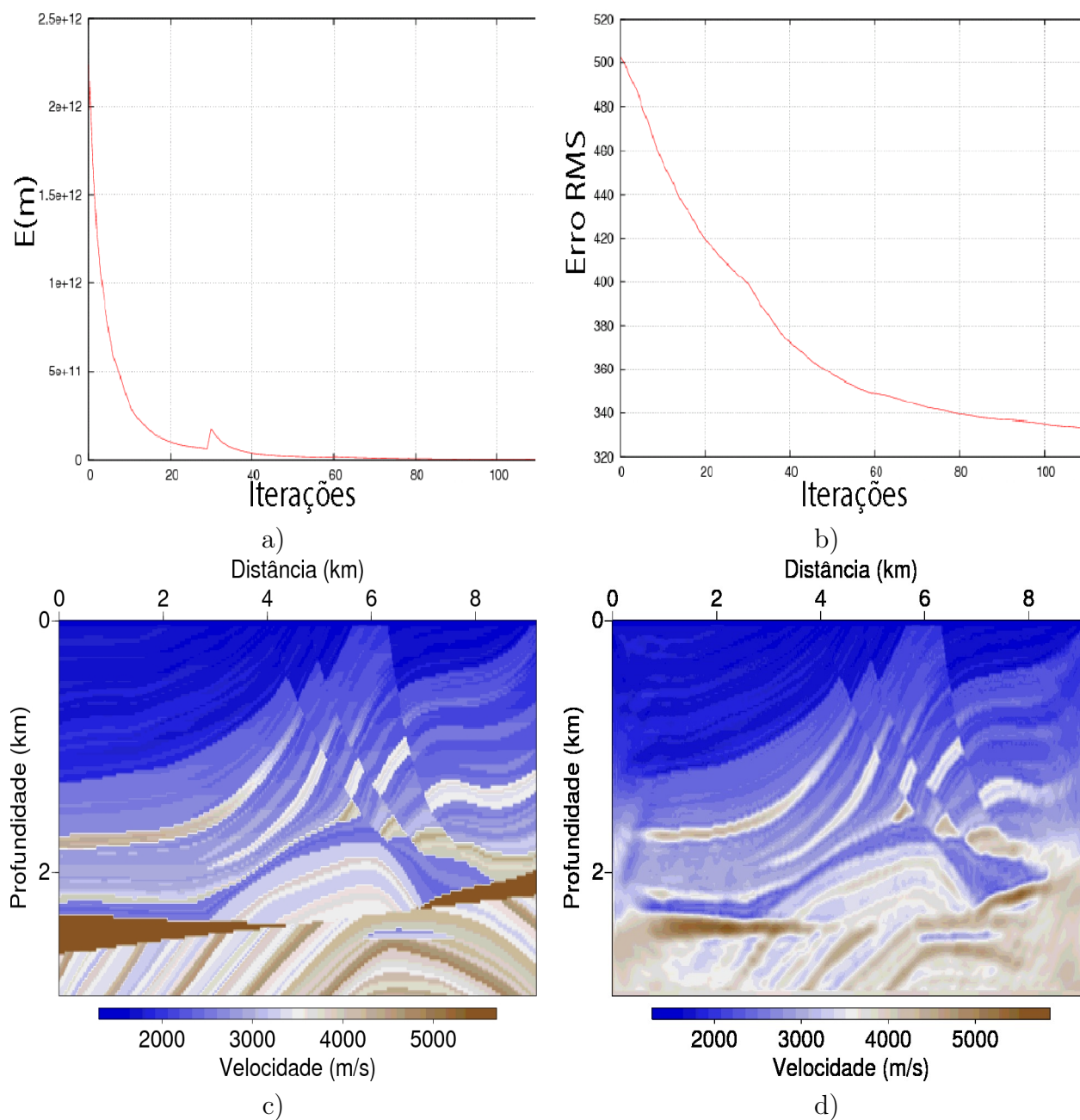


Figura 3.14: Resultado multiescala para 4 bandas de frequência, até 30 Hz. a) Função objetivo, por iteração; b) Erro RMS entre modelo verdadeiro e estimado, por iteração; c) modelo verdadeiro e d) modelo estimado final.

4

Implementação em placas gráficas (GPUs)

A modelagem sísmica é um processo altamente intensivo do ponto de vista computacional. Como visto nos capítulos 1, 2 e 3, a RTM com a metodologia de borda efetiva equivale a 3 modelagens sísmicas, e a FWI custa 1 RTM e 1 modelagem (ou seja, 4 modelagens) por iteração. Apesar do alto custo computacional, o problema é altamente paralelizável. Pode ser paralelizado tiro a tiro e/ou ponto a ponto, ou seja, cada ponto do campo de onda pode ser computado em paralelo. O primeiro tipo de paralelização (tiro a tiro) é usualmente aproveitado fazendo uso de diversos processadores, e linguagem MPI (dos Santos, 2013); o segundo tipo, ponto a ponto, e um tiro de cada vez, é mais bem aproveitado por uma implementação em placas gráficas, foco desta dissertação, e introduzida neste capítulo. Pseudocódigos com algumas das principais implementações utilizadas nesta dissertação são apresentados no Apêndice C.

4.1 A unidade de processamento gráfico - GPU

Devido à crescente demanda do mercado por imagens de cada vez mais alta-definição, associada à necessidade de transmissão em tempo real, a unidade de processamento gráfico (*Graphics Processing Unit*, ou GPU) evoluiu para uma estrutura altamente paralelizada, multifilamentada (por filamento ou *thread* entende-se sub-unidade de processamento) e com diversos núcleos, resultando numa tremenda capacidade de processamento de ponto-flutuante. A Figura 4.1 (NVIDIA, 2017b) ilustra a superioridade das GPUs em velocidade de processamento (GFLOP/s):

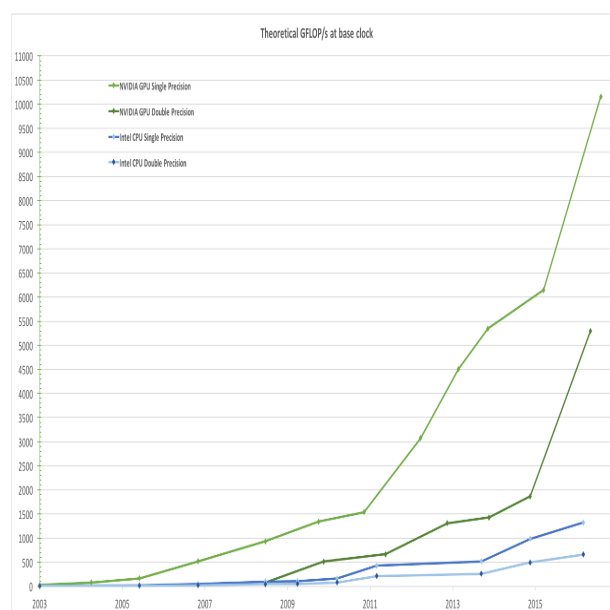


Figura 4.1: Operações em ponto flutuante (GFLOPS) - CPU e GPU ao longo dos últimos anos, retirado de NVIDIA CUDA C Programming Guide (2017).

A razão por trás dessa discrepância entre CPU e GPU, no cálculo de pontos-flutuantes, é que a GPU é justamente especializada em computação intensiva e altamente paralelizada, já que seu propósito original é a renderização gráfica, onde cada pixel de imagem é constantemente atualizado em paralelo (NVIDIA, 2017b). Seu *design* difere primariamente da CPU por dedicar mais transistores ao processamento de dados (ALUs - *Arithmetic Logical Units*) em detrimento do cache de dados e controle de fluxo, como ilustra a Figura 4.2.

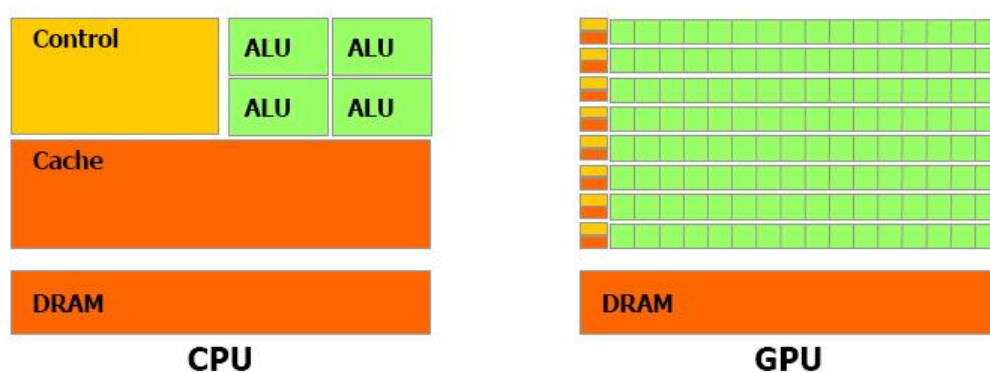


Figura 4.2: Diferenças nas arquiteturas da CPU e GPU, retirado de NVIDIA CUDA C Programming Guide (2017).

Essencialmente, as GPUs do início da década de 2000 foram designadas para produzir uma certa cor para cada pixel na tela, usando unidades aritméticas programáveis conhecidas como *pixel shaders* (sombreadores de pixel). Em geral, um pixel shader usa a posição (x,y) do pixel na tela, além de informações adicionais - cores, coordenadas de texturas, atributos

gráficos - para computar a cor final. Devido ao fato das operações matemáticas realizadas nas cores e texturas de entrada serem completamente controladas pelo programador, pesquisadores concluíram que estas "cores" de entrada poderiam ser na verdade qualquer tipo de dado (Sanders e Kandrot, 2010).

Inicialmente, a programação direcionada à GPU estava restrita a linguagens (APIs) gráficas, como OpenGL e DirectX. Tal situação dificultava bastante a pesquisa em áreas não relacionadas a área gráfica, pois o programador precisava inicialmente converter seus dados para um ambiente gráfico, processá-lo, e por fim reconvertê-lo. Esse processo, além de trabalhoso, poderia apresentar erros inesperados (já que não se estava sendo feita uma computação gráfica em si). A situação mudou em 2006, quando a NVIDIA lançou a GeForce 8800 GTX, a primeira placa gráfica a ser construída com arquitetura CUDA (do inglês, *Compute Unified Device Architecture*). De forma resumida, esta nova arquitetura permitia que as ALUs (Unidades Lógicas Aritméticas), antes focadas em processamento de pixels, possuíssem a liberdade de realizar computações de uso geral. Em conjunto com esta nova arquitetura, a NVIDIA criou o CUDA C (e seu compilador `nvcc`), que nada mais é que a linguagem C com novas bibliotecas e comandos específicos para operar na GPU e CPU (Sanders e Kandrot, 2010).

Quanto às aplicações a GPU é bem adaptada a problemas que podem ser expressos por computação paralela - o mesmo programa é executado em unidades de processamento em paralelo - e com alta intensidade aritmética, que é a razão entre operações aritméticas e operações de memória. Como mostrado, a GPU atual é especializada em cálculos de ponto-flutuante, deixando em segundo plano o controle de fluxo; por esta razão, a GPU tem capacidade inferior de transferência de memória comparada à CPU (NVIDIA, 2017b). Problemas com baixa necessidade de transferência de memória e alta necessidade de cálculos são os que melhor se adequam à programação em GPU, como é o caso da modelagem sísmica.

4.2 Arquitetura da GPU - CUDA

A unidade gráfica é organizada em sub-unidades de processamento denominadas *threads*. As *threads*, por sua vez, podem ser organizadas em blocos (*blocks*), e os *blocks* podem ser organizados em *grids*, como ilustra a Figura 4.3 (NVIDIA, 2017b).

O número de *grids*, *blocks* por *grid* e *threads* por *block* é definido pelo programador. Cada *thread* tem um identificador único, relacionada à sua posição em certo bloco de certo *grid*. Ao rodar um programa, todas as *threads* solicitadas executam o mesmo código, porém como cada uma tem um identificador único, os resultados são paralelos (no cálculo de um

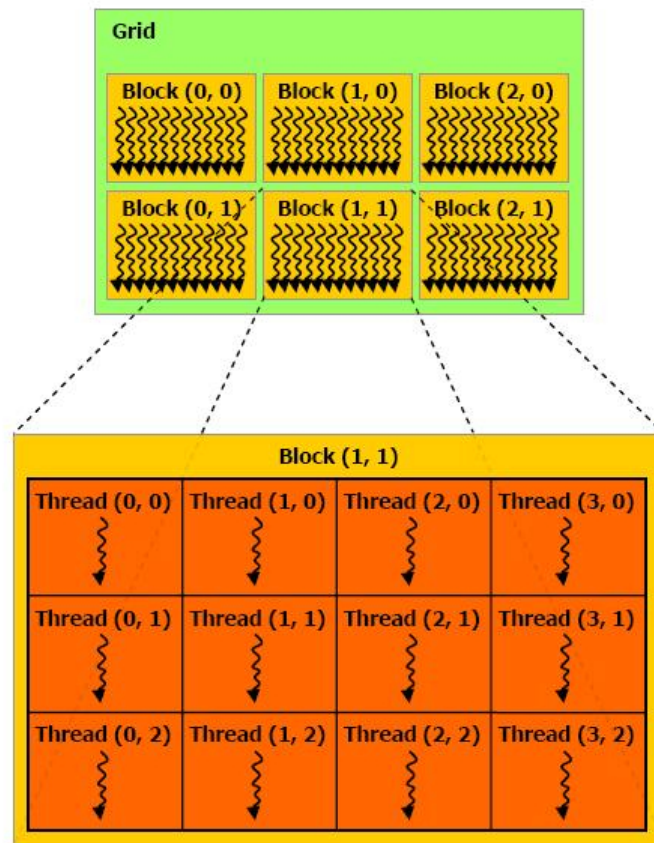


Figura 4.3: Grupos de threads são organizados em blocks; grupos de blocks são organizados em grids, retirado de NVIDIA CUDA C Programming Guide (2017).

Laplaciano em cada ponto de uma malha, por exemplo; a equação é a mesma para todas as *threads*, mas o resultado é distinto devido aos identificadores únicos de cada *thread*).

O processo pode ser ilustrado por um exemplo simples, como a soma de dois vetores de mesma dimensão. O objetivo é somar o vetor 1 (a) com o vetor 2 (b), de forma a obter um vetor 3 (c), onde $a(i) = b(i) + c(i)$:

Num pseudocódigo serial, o problema pode ser programado da seguinte maneira:

```

Defina a dimensão N dos vetores
Dimensiona a(N), b(N), c(N)
Preenche a e b; preenche c com zeros
Faça para index de 1 a N {
    c(index) = a(index) + b(index)
}
Imprime c
Fim

```

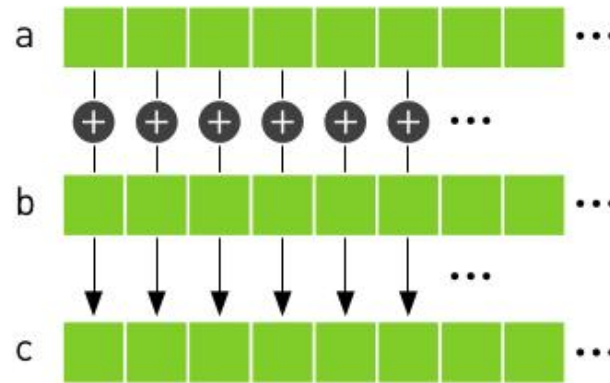


Figura 4.4: Somando dois vetores.

O seu equivalente em linguagem C seria:

```
#define N 10    //Neste exemplo, N=10
int main() {
    int a[N], b[N], c[N];
    int index;

    //Preenche v1 e v2
    for (int i=0; i<N; i++) {
        a[i] = -i;
        b[i] = i * i;
    }

    //Cálculo da soma
    int index = 0;
    while (index < N) {
        c[index] = a[index] + b[index];
        index += 1;
    }

    //Imprime c
    for (int i=0; i<N; i++) {
        printf( "%d + %d = %d\n", a[i], b[i], c[i] );
    }
}
```

```
return 0;
} //Fim
```

Na etapa "Cálculo da soma", o processador executa N iterações. A reestruturação do código utilizando a GPU permite que esta soma seja feita em apenas 1 iteração, executada por N *threads*. Assim, o código paralelizado, em pseudocódigo seria:

```
Defina a dimensão N dos vetores
Dimensiona a(N), b(N), c(N) na CPU
Dimensiona a_d(N), b_d(N), c_d(N) na GPU
Preenche a e b; preenche c com zeros na CPU
Copia a, b, c para a_d, b_d, c_d (CPU para GPU)
Execute a função "adição" para N threads, 1 block, 1 grid
Copia c_d para c (GPU para CPU)
Imprime c
Fim

Função adição(a[N], b[N], c[N]) {
    c[index_thread] = a[index_thread] + b[index_thread]
}
```

A diferença fundamental é a variável `index_thread`, que é única para cada *thread*. Como são chamadas N threads, este índice é preenchido automaticamente: para a primeira *thread* `index_thread = 1`, para a segunda Thread `index_thread = 2`, até a *thread* N , onde `index_thread = N`. Este processo pode ser entendido mais detalhadamente analisando o código em CUDA C (adaptado de Sanders e Kandrot, 2010):

```
#define N 10
int main( void ) {
    int a[N], b[N], c[N];
    int *dev_a, *dev_b, *dev_c;
    // Aloca a memória na GPU
    cudaMalloc( (void**)&dev_a, N * sizeof(int) );
    cudaMalloc( (void**)&dev_b, N * sizeof(int) );
    cudaMalloc( (void**)&dev_c, N * sizeof(int) );
    // Preenche 'a' e 'b' na CPU
    for (int i=0; i<N; i++) {
```

```
    a[i] = -i;
    b[i] = i * i;
}
//Copia os vetores 'a' e 'b' para a GPU
cudaMemcpy( (dev_a, a, N*sizeof(int), cudaMemcpyHostToDevice) );
cudaMemcpy( (dev_b, b, N*sizeof(int), cudaMemcpyHostToDevice) );

//Cálculo da soma na GPU
add<<<N,1>>>( dev_a, dev_b, dev_c);

//Copia 'c' da GPU para a CPU
cudaMemcpy( (c, dev_c, N*sizeof(int), cudaMemcpyDeviceToHost ) );

//Imprime os resultados
for(int i=0; i<N; i++) {
    printf( "%d + %d = %d\n", a[i], b[i], c[i] );
}

//Libera a memória alocada na GPU
cudaFree (dev_a);
cudaFree (dev_b);
cudaFree (dev_c);

return 0;

}
```

É interessante observar que na etapa "Cálculo da soma na GPU" a definição de N threads, 1 bloco e 1 grid é feita dentro da chamada da função add, entre os delimitadores <<<>>>. Todas as N threads executarão a função add, mas como o identificador *blockIdx.x* é diferente para cada uma, cada *thread* executará uma soma $c(i) = a(i) + b(i)$, reduzindo o número de iterações de N para apenas 1.

4.3 Memória da GPU

A maior desvantagem da GPU em relação à CPU é em relação à memória. Enquanto CPUs podem manipular centenas de GB de memória, a GPU tem tipicamente 2 GB de memória em média (ou 5 GB como a placa Tesla C0275 do CPGG-UFBA, utilizada nos testes desta dissertação; as placas mais modernas podem chegar a ter 12 GB). Saber se a GPU terá memória para os dados relacionados ao problema de modelagem/RTM/FWI (para pelo menos 1 tiro) é muito importante, pois dividir os dados e recarregá-los na GPU pode custar muito tempo. O ideal é que a GPU tenha memória para o modelo de velocidades, os campos de pressão, e o tiro corrente. À medida que o volume de dados ultrapassa a memória da GPU, as transferências de memória CPU-GPU crescem muito, de forma que são necessárias estratégias de escalabilidade para que a paralelização em GPU seja vantajosa (por exemplo, o uso de múltiplas GPUs).

Numa implementação em GPU, devemos nos preocupar com 4 tipos de memória:

1. memória da CPU, ou *host memory*;
2. memória global da GPU, ou *global memory*;
3. memória constante da GPU, ou *constant memory*;
4. memória compartilhada da GPU, ou *shared memory*.

Cada tipo de memória tem uma latência de acesso (quão "rápida" é a memória) diferente; na lista acima, a latência de acesso é decrescente, ou seja, a *shared memory* é a mais rápida, porém é a que tem menos espaço disponível na GPU. Uma maneira óbvia de aumentar a performance seria sempre usar ao máximo a memória de menor latência, mas como veremos o pequeno espaço disponível pode limitar essa abordagem (Sadahiro, 2014).

Quaisquer dados lidos de arquivos para a memória da CPU devem ser copiados inicialmente para a *global memory* da GPU. Uma vez disponíveis na *global memory*, os dados podem ser copiados para as memórias de menor latência. A Figura 4.5 ilustra a hierarquia das 4 memórias citadas anteriormente, e outras memórias que serão citadas ou explicadas a seguir, dependendo de sua relevância.

A memória global é a maior e mais lenta das memórias na GPU, com tamanhos típicos de até 6 GB (Tesla C2075 do Cluster Águia CPGG-UFBA, por exemplo); os modelos mais modernos da série Tesla podem ter 12, 16 e até 24 GB. A memória fica na placa, mas não no *chip*. Os dados na *global memory* são automaticamente *cacheados* para o chip, e eventualmente substituídos por novos dados. O conceito de *cache*, muito usual em computação,

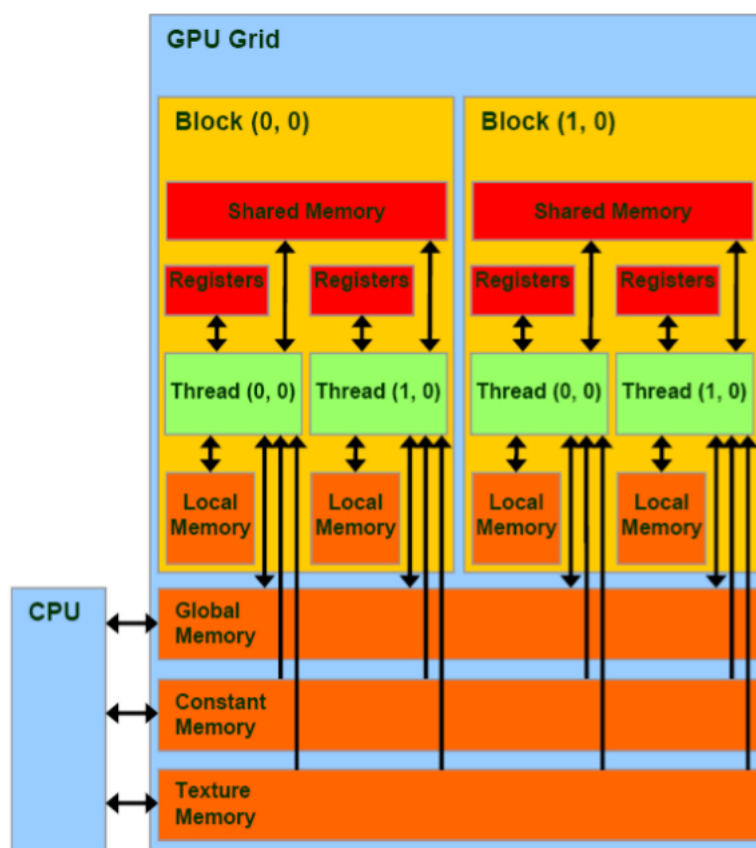


Figura 4.5: Esquematização das memórias em CUDA.

pode ser resumido aqui como um espaço limitado, mas de rápido acesso, onde dados são armazenados. Assim, no contexto da *global memory*, elementos dos dados que sejam usados repetidamente tem boa chance de aproveitar a rapidez dessa memória *cache*. Devido ao fato da *global memory* da GPU não ser tão grande quanto a memória da CPU, *clusters* de computadores baseados na arquitetura GPU não são tão populares na indústria do petróleo como os *clusters* convencionais de CPUs. A memória global tem a maior latência dentre as memórias da GPU, porém é muito mais rápida que a latência da memória da CPU. Como o nome sugere, o escopo da *global memory* é global dentro da GPU (ou seja, pode ser acessada a partir de qualquer área da GPU). A *global memory* é também a única que pode ser acessada pela CPU através de comandos de cópia de memória em CUDA, como mostra a Figura 4.5 (Sadahiro, 2014).

A *shared memory* é uma memória de baixa latência *on-chip* (localizada fisicamente no *chip*), da GPU. É também conhecida como *user-controlled L1 cache*. A maioria das GPUs tem 16 KB de *shared memory* disponível por *streaming multiprocessor*, e placas mais potentes, como a Tesla C2075 (Cluster Águia CPGG-UFBA), tem 48 KB por *streaming multiprocessor*. O *streaming multiprocessor*, ou SM, é uma das unidades hierárquicas do *hardware* da GPU, e para fazer uso eficiente da *shared memory* é preciso conhecer o número

de SMs da placa utilizada, e implementar de acordo (Rubio et al., 2012). A *shared memory* tem uma velocidade de acesso cerca de 100 vezes maior que a *global memory*, de forma que "o conceito de *shared memory* é algo que deve ser conhecido e utilizado, sempre que possível, por todo programador em GPU que se importe com a performance"(Cook, 2012).

A implementação da modelagem sísmica utilizando operadores de diferenças finitas e a *shared memory* é superior à implementação utilizando apenas a *global memory*, como mostra (Micikevicius, 2009). Todavia, é muito mais complicada, pois depende do conhecimento do *hardware* da placa (e seus SMs), para um aproveitamento completo. Como mostrado a seguir por Sadahiro (2014), o ganho em velocidade da implementação em *shared memory* é apenas ligeiramente superior, de forma que, analisando a razão "esforço de implementação"versus "ganho de performance", optou-se, nesta dissertação, pelo uso exclusivo da memória global para os operadores de modelagem. O uso da *shared memory* limitou-se apenas à aceleração de aplicações simples, como o cálculo de produtos escalares necessários à FWI.

Sadahiro (2014), em sua implementação para o problema de modelagem sísmica em GPU, analisa diferentes implementações para o problema de extrapolação do campo de onda, inclusive comparando implementação com *global memory* e *shared memory*. Na Figura 4.6, retirada de sua dissertação, podemos ver que o ganho pelo uso da *shared memory*, para o problema de modelagem sísmica, não é muito significativo. Desta forma, nas implementações desta dissertação se optou pelo uso da memória global da GPU para a extrapolação dos campos de onda.

A título de entendimento da Figura 4.5, a *local memory*, ou memória local, é um espaço privado de memória interno a cada *thread*. Ela tem a mesma latência da memória global. Qualquer variável alocada num *kernel* (ver seção anterior) é armazenada na memória local.

A memória constante tem menor latência que a memória global, e também tem escopo global. Normalmente a memória constante das GPUs tem 64 KB. Se tentarmos acessar blocos de memória consecutivos via *threads*, a tarefa é serializada, degradando a eficiência do acesso de memória. No caso da memória constante, os valores são transmitidos para todas as *threads*, tornando-a mais rápida (Sadahiro, 2014). A memória constante é útil para armazenar constantes muito utilizadas no processamento, como por exemplo os coeficientes dos operadores de diferenças finitas.

4.4 Observações sobre custo computacional

Uma característica da GPU que se deve conhecer para otimizar as implementações é o conceito de *warp*, que é a forma que a arquitetura CUDA executa seus processos. Cada *warp*

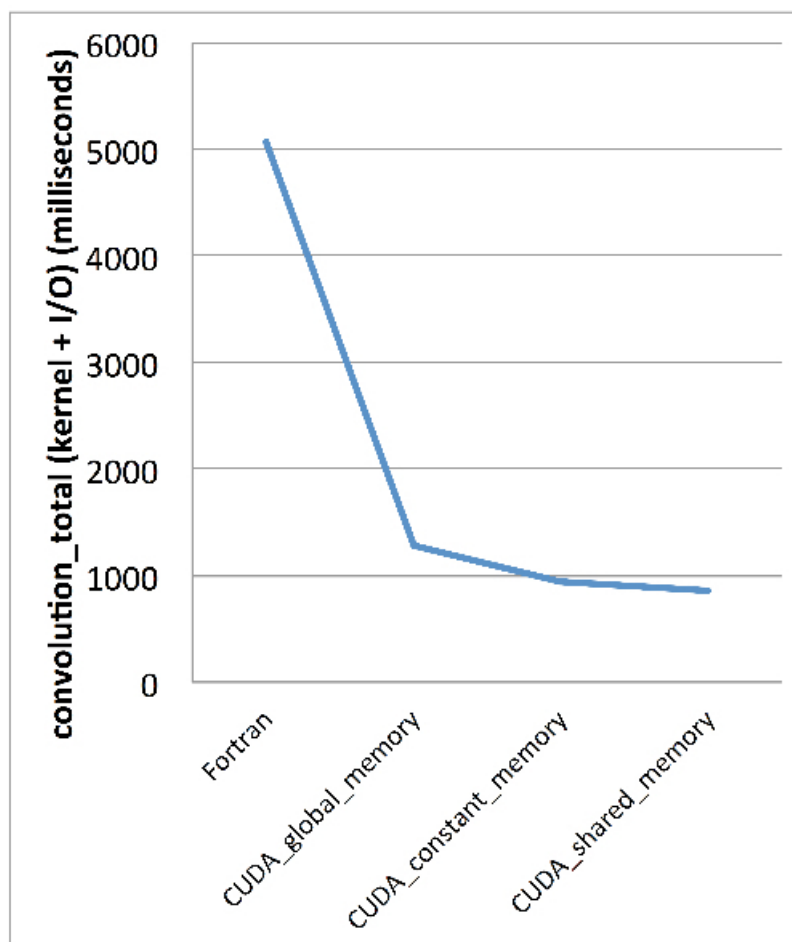


Figura 4.6: Performance no uso de diferentes memórias em implementação em GPU do problema de modelagem sísmica. Retirado de Sadahiro (2014).

consiste de 32 threads, e 32 posições de memória consecutivas de um vetor são acessadas de uma única vez. O acesso à memória global (e compartilhada) usa esse esquema. Quando o número de elementos de um vetor ou matriz sendo processado é múltiplo de 32, normalmente as operações são executadas mais rapidamente que em um vetor não múltiplo de 32. Dessa forma, uma boa otimização, no exemplo da modelagem sísmica, é expandir o campo até o múltiplo mais próximo de 32 (e no final coletar os resultados apenas na área do campo original). Por exemplo, se o campo de velocidades (e conseqüentemente o campo de pressão associado ao problema) tem 125×383 amostras, ele seria expandido para 128×384 , onde cada dimensão é múltipla de 32. Esse ajuste representa um pequeno ganho na velocidade de execução do código.

Uma segunda característica para se levar em conta é o tempo que levam as transferências de memória CPU-GPU (ou *host-device*) e GPU-CPU (ou *device-host*). Na dissertação de Sadahiro (2014) são analisados os tempos das transferências, inicialização, e processamento em si do *kernel* de modelagem sísmica. Como se pode observar na Figura 4.7, cerca de

50% do tempo de processamento é devido à transferência de memória CPU-GPU/GPU-CPU, e os outros 50% à execução do *kernel* em si. Dessa forma, conclui-se que uma boa implementação deve ter o mínimo possível de transferência de memória entre CPU e GPU. O ideal é que sempre dê para copiar todos os dados para a GPU e processá-los, sem que haja necessidade de retransferência. No problema de modelagem 2D tratado nesta dissertação, isso é possível; todavia, dados 3D superam a memória global de 5 GB da GPU, fato que degradaria a velocidade do código. Nesse caso, clusters de GPUs seriam mais adequados ao problema.

Time(%)	Time	Calls	Avg	Min	Max	Name
=====	Profiling result:					
25.16	320.38ms	7996	40.07us	928ns	174.69us	[CUDA memcpy HtoD]
22.95	292.20ms	1999	146.17us	145.95us	167.17us	[CUDA memcpy DtoH]
1.55	19.73ms	1999	9.87us	9.38us	10.34us	initializerKernel
50.34	640.99ms	1999	320.66us	306.34us	345.38us	vtconvlKernelXZopY_GMEB

Figura 4.7: Tempos de processamento numa implementação de modelagem sísmica em GPU. Retirado de Sadahiro (2014).

Seguindo o mesmo raciocínio, medimos a RTM de 1 tiro, para o modelo Marmousi-Teste (Tabela 4.4). Nesse caso, há transferência do modelo de velocidades do *host* para o *device*, e da imagem migrada do *device* para *host*. Vemos que a transferência toma 20% do tempo de processamento do código. Na RTM de 10 tiros, a transferência é praticamente a mesma, mas o tempo de processamento é 10 vezes maior (Tabela 4.4); nesse caso, o tempo de transferência passa a ser de apenas 2,5% do código, mostrando que a RTM é um problema adequado à implementação em GPU, dada a baixa transferência de dados entre GPU e CPU.

	Tempo(s)	Tempo (%)
Transferência H2D e D2H	0.16s	20%
RTM (1 tiro)	0.61s	80%

Tabela 4.1: Tempo de processamento para a RTM em GPU.

	Tempo(s)	Tempo (%)
Transferência H2D e D2H	0.16s	2.5%
RTM (10 tiros)	6.1s	97.5%

Tabela 4.2: Tempo de processamento para a RTM em GPU.

4.5 Bibliotecas úteis: cuBLAS e cuFFT

Nesta dissertação duas bibliotecas disponibilizadas gratuitamente pela NVIDIA foram utilizadas: cuBLAS e cuFFT. As bibliotecas possuem funções de alta performance para CUDA

C, fazendo uso de otimizações direcionadas à arquitetura da GPU, de forma que suas implementações para certa tarefa quase sempre serão superiores às realizadas por um usuário. A biblioteca cuBLAS (NVIDIA, 2015), é uma implementação em CUDA C da clássica biblioteca BLAS (Basic Linear Algebra Subprograms), originalmente escrita em FORTRAN. Ela foi muito útil às implementações desta dissertação, especialmente no cálculo de produtos escalares necessários aos métodos de inversão (Capítulo 3), e em transposições e multiplicações de matriz na GPU.

Já a biblioteca cuFFT (NVIDIA, 2017a) é uma implementação em CUDA C para a transformada rápida de Fourier, adaptada da biblioteca FFTW. Nesta dissertação suas funções foram utilizadas para cálculo do filtro de Wiener (Capítulo 3) e nas transformadas de Fourier do operador pseudo-espectral (Capítulo 1).

5

Metodologia e resultados

Este capítulo tem por objetivo mostrar a metodologia aplicada na implementação da FWI acústica 2D em GPU e os problemas encontrados. De posse da melhor metodologia e solução dos problemas práticos, apresentamos os resultados para diversos modelos de velocidade.

5.1 Metodologia

A metodologia inicial consistiu em realizar testes da FWI para um modelo 2D reduzido, com poucos tiros, e um operador de diferenças finitas de baixa ordem, de forma a executar o processo e corrigir problemas em tempo hábil. O modelo inicial escolhido para teste foi um modelo de velocidades Marmousi-Teste, e os dados observados/modelados com geometria do dado observado base (definidos no final da Introdução). As dimensões espaciais do modelo Marmousi devem ser mantidas sempre em aproximadamente 3 km verticais e 9,2 km horizontais, que são as dimensões geológicas "reais" pretendidas para o Marmousi. Para esse caso, os intervalos são $\Delta z = 24$ m e $\Delta x = 24$ m. Tipicamente o tempo de cada teste nesse modelo variou de 5 min (até 10 tiros, até 10 iterações, sem multiescala) até 45 min (28 tiros e 4 rodadas multiescala de 30 iterações). O curto tempo de espera nesses testes, proporcionado pela implementação paralela em GPU, foi fundamental ao avanço do trabalho.

Corrigidos os problemas, são testados os diversos operadores abordados no capítulo 1, e apresentados os critérios para a melhor escolha do operador, dependendo do problema (seção "Análise dos operadores de modelagem"). Depois são analisados os métodos de inversão citados no capítulo 3, o conjugado-gradiente e o L-BFGS, concluindo qual o melhor método para os testes desta dissertação (seção "Análise dos métodos de inversão"). Na última seção

antes dos resultados se discute a influência do uso de um dado observado sintético (caso desta dissertação) no processo de inversão (seção "Notas sobre $\mathbf{d}_{\text{obs}}$ sintético"). Assim, resolvidos os problemas e determinados os critérios de escolha dos métodos de modelagem e inversão, são gerados os resultados (seção "Resultados") para modelos maiores, com maior conteúdo de frequência e maior quantidade de tiros, de forma a obter a melhor imagem possível. Antes das seções comparativas, nos parágrafos a seguir há um breve resumo dos capítulos 1, 2 e 3 (modelagem, RTM e inversão), sobre o que deve ser levado em consideração para os testes práticos.

Quanto à modelagem, ao testar um modelo, deve-se considerar sempre as condições de dispersão e estabilidade necessárias para uma boa simulação numérica. Deve-se também considerar um tempo máximo de propagação que permita chegada na superfície das reflexões decorrentes dos refletores mais profundos do modelo; no caso do modelo Marmousi, e dos demais modelos testados nesta dissertação, o tempo de 4 s foi suficiente. Esse tempo máximo delimita então o número de iterações que deve ser realizada pelo algoritmo de modelagem; por exemplo, se as condições de estabilidade só permitem um $\Delta t = 0,002$ s, o número de iterações deve ser necessariamente 2000, de forma a completar 4 s de propagação dum tiro.

Quanto à migração (RTM), deve-se inicialmente garantir uma boa modelagem, da forma citada no parágrafo anterior. Além disso, deve-se considerar a memória utilizada para armazenar os campos de pressão. Como a GPU utilizada nesta implementação tem memória máxima de 5,2 GB, as dimensões (nz, nx) do modelo de velocidades (que determina as dimensões dos campos de pressão) e o número máximo de iterações temporais nt são proporcionais à memória que deve ser alocada. Como citado no Capítulo 2, na seção sobre RTM, o cubo de pressões da fonte em todos os tempos é necessário para gerar a imagem; ele tem dimensões $nz \times nx \times nt$. No caso do modelo 125×383 amostras, com 2000 iterações temporais, um cubo de campos tem então aproximadamente $125 \times 383 \times 2000 \times 4 = 365$ MB (visto que um número real ocupa 4 bytes). Esse valor está distante da capacidade máxima de 5,2 GB da Tesla C2075, mas levando em consideração, por exemplo, o maior campo Marmousi 2D disponibilizado, de dimensões 751×2301 , a memória necessária para o cubo seria de 13GB, inviável para a GPU. Desta forma, pensando em testes futuros de maiores dimensões, se faz necessária uma estratégia de economia de memória, como a borda efetiva citada no Capítulo 2. A borda efetiva permite armazenar, ao invés de um cubo $nz \times nx \times nt$, um cubo $2(nz + nx) \times nt$. A economia de memória é compensada pela necessidade de mais uma modelagem, para reconstruir o campo da fonte, durante a condição de imagem, totalizando 3 modelagens para a RTM: uma para o campo da fonte, uma para o campo dos receptores, e uma para reconstruir o campo da fonte.

Como mostrado no Capítulo 3, pelo método adjunto, o gradiente da função objetivo na FWI é obtido através da RTM do resíduo. Desta forma, com um algoritmo de RTM implementado, se tem praticamente toda a estrutura da FWI, do ponto de vista de memória. Na prática a FWI consiste em realizar uma RTM por iteração, atualizando o modelo de velocidades. Alguns métodos de inversão demandam uma modelagem a mais para delimitar o passo, de forma que o custo da FWI por iteração passa a ser de 4 modelagens: 3 da RTM (com borda efetiva) e uma para estimar o passo. Assim, a estimativa do custo da FWI pode ser obtida diretamente através do custo de modelagem, abordagem que foi seguida na seção Resultados. As implementações de modelagem e RTM parecem ser suficientes para implementar a FWI, mas não são; alguns problemas que inviabilizam a inversão surgem no processo, sendo os principais:

- balanceamento das amplitudes do dado modelado, para obtenção de um resíduo significativo;
- influência da onda direta no resíduo e no gradiente.

Esses problemas são abordados na seção seguinte.

5.2 Ajustes à implementação da FWI

Na etapa inicial de testes foi utilizado o modelo Marmousi-Teste, 28 tiros, fonte de $f_{max} \approx 30$ Hz ($f_{peak} = 13$ Hz), e 2000 iterações temporais com $\Delta t = 0,002$ s (totalizando 4 s). O dado observado é mostrado na Figura 5.2. O modelo inicial para FWI, mostrado na Figura 5.1-b, foi gerado da suavização dupla (suavização da suavização) do modelo 5.1-a, usando a função *smooth2* do pacote *Seismic Unix*¹ (Stockwell e Cohen, 2008), que utiliza uma técnica de mínimos quadrados amortecido.

O método de inversão utilizado foi o conjugado-gradiente, da forma descrita na seção 3.2.3, e o passo α calculado pelo método descrito em 3.2.2. A inversão foi feita para toda a banda de frequência de 0 – 30 Hz, sem abordagem multiescala.

Durante a fase de testes os problemas da normalização das amplitudes e da influência da onda direta surgiram simultaneamente, mas aqui serão abordados separadamente.

¹Pacote de ferramentas sísmicas, em código aberto, criado e mantido pela *Colorado School of Mines*.

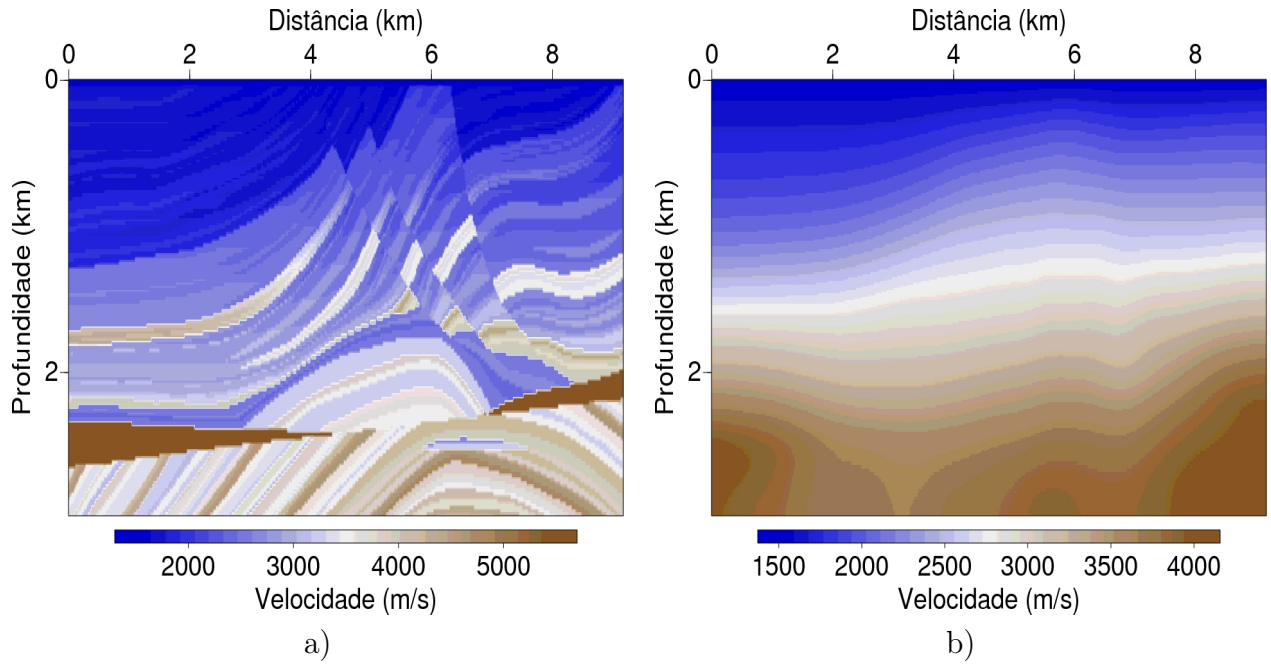


Figura 5.1: Modelo Marmousi de 125×383 amostras, a) verdadeiro e b) suavizado duas vezes.

5.2.1 Normalização das amplitudes

Quando não se normalizam os dados calculados pela amplitude dos dados observados, o resíduo fica desbalanceado (não localiza bem os eventos cinemáticos) e a função objetivo diverge. Nesse exemplo, foram feitas 10 iterações para 5 tiros (dentre os 28 do dado observado); a *wavelet* utilizada no dado calculado é a mesma do dado observado, mas com amplitude 10 vezes menor. O resultado mostrado na Figura 5.3-a parece coerente, mas na verdade os contrastes de camada estão ligeiramente deslocados de suas posições verdadeiras, levando a função objetivo a divergir (Figura 5.3-c). Na Figura 5.3-b vemos os resultados para amplitudes normalizadas do dado calculado, que apresenta uma função objetivo (Figura 5.3-d) convergente. A equação utilizada na normalização é apresentada na seção 3.4.

Foram realizados testes semelhantes, gerando o dado calculado com a mesma *wavelet*, mesma amplitude, só que com operadores diferentes (DF de diferentes ordens, REM DF, pseudo-espectral, REM pseudo-espectral). A mudança de operador não afetou as amplitudes dos sismogramas, e houve convergência do processo.

Pensando em aplicações práticas, a normalização é uma etapa fundamental, visto que os dados observados reais normalmente passarão por diversos pré-processamentos e filtragens, que irão alterar suas amplitudes. É importante, então, que as amplitudes dos dados calculados estejam sempre no mesmo intervalo que as do dado observado, de forma a garantir a convergência do processo.

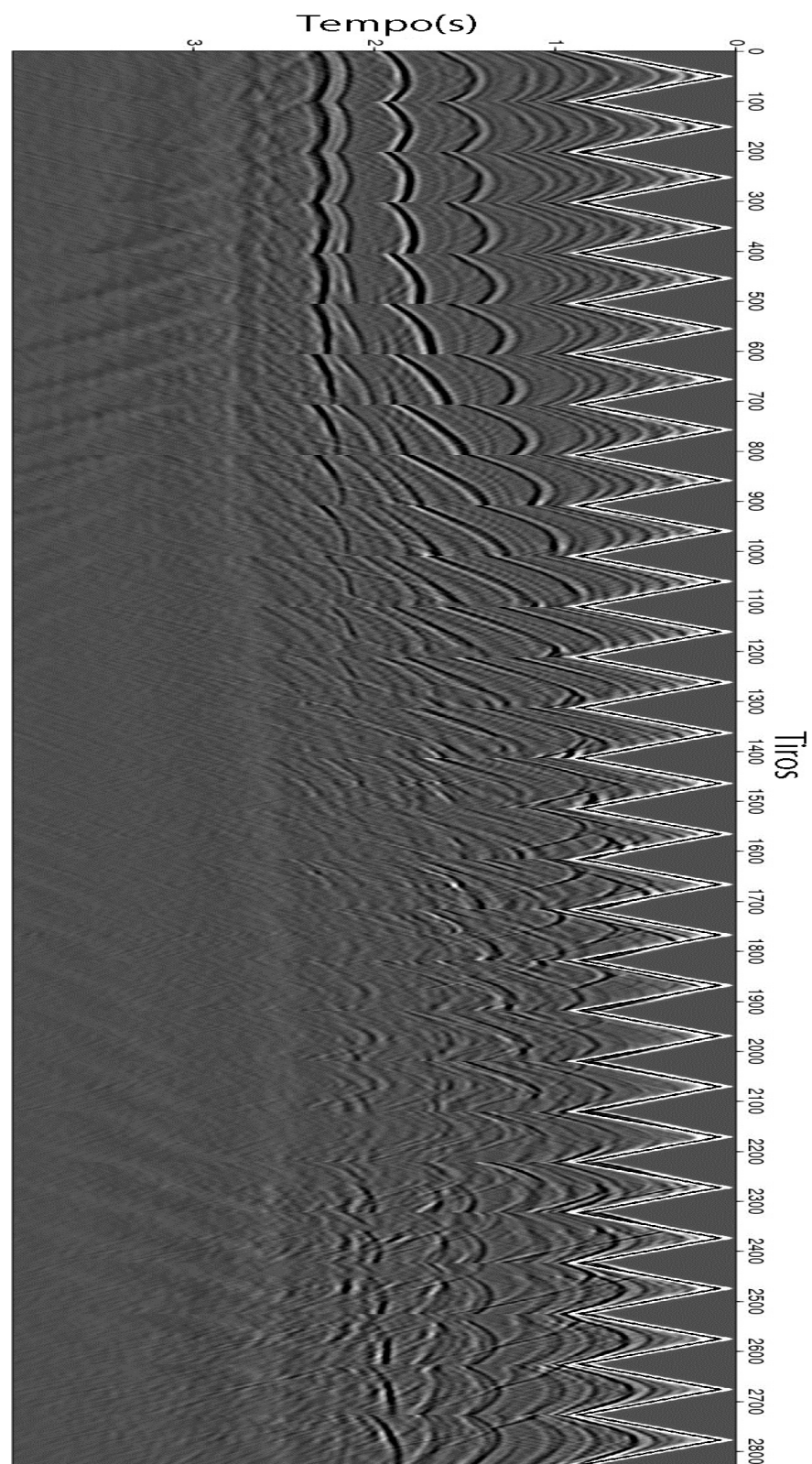


Figura 5.2: Dado observado para o modelo Marmousi, 28 tiros.

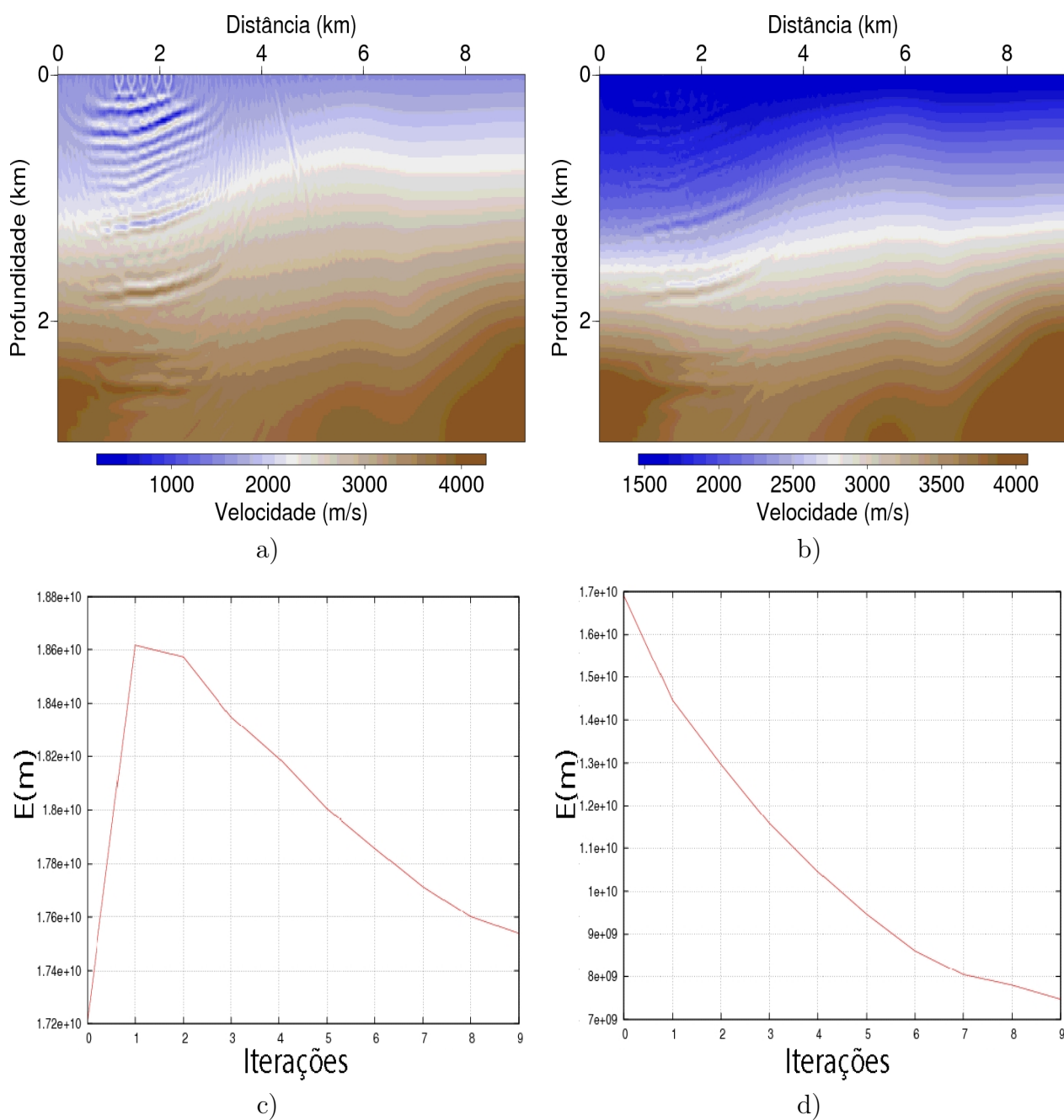


Figura 5.3: 10 iterações da FWI para 5 tiros, com e sem normalização dos dados calculados. a) Resultado sem normalização e sua função objetivo (c). b) Resultado com normalização e sua função objetivo (d).

5.2.2 Influência da onda direta

Se o operador de modelagem utilizado para o cálculo de $\mathbf{d}_{\text{obs}}$ for idêntico ao utilizado para o cálculo de $\mathbf{d}_{\text{cal}}$, e a fatia da superfície do modelo de velocidades utilizado em ambos for também idêntica, as amplitudes da onda direta serão exatamente iguais em ambos os dados, sendo completamente eliminada pela subtração intrínseca ao resíduo $\mathbf{d}_{\text{obs}} - \mathbf{d}_{\text{cal}}$, como exemplificado em Park et al. (2014). Por exemplo, no caso de operadores de diferenças finitas idênticos de ordem NJ , a onda direta será idêntica em ambos os dados se, no mínimo, as primeiras $NJ/2$ amostras verticais de cada um dos modelos for idêntica.

Na prática, isso não ocorre; no caso dessa dissertação, as fatias dos modelos de velocidade verdadeiro e inicial são ligeiramente diferentes, e muitas vezes o operador de modelagem utilizado em $\mathbf{d}_{\text{obs}}$ é também diferente do usado em $\mathbf{d}_{\text{cal}}$. Consequentemente, a onda direta ainda aparece no resíduo, e sua amplitude é muito maior que a de todos os outros eventos, de forma que o gradiente (Figura 5.4-b), que é a migração do resíduo, acaba apresentando um evento de altíssima amplitude decorrente da onda direta. A atualização do modelo corrente por esse gradiente acaba realçando apenas o evento advindo da onda direta, negligenciando todos os outros refletores, e gerando os resultados ruins para o modelo de velocidades invertido, como mostrado na Figura 5.4-a. Nesse caso, há convergência da função objetivo, mas essa minimização gira em torno apenas do evento da onda direta, que por ter as maiores amplitudes, domina o processo (Figura 5.4-c).

Como mostrado na seção sobre RTM, Figura 2.6, a condição de imagem de correlação cruzada da RTM gera artefatos decorrentes da onda direta do sismograma (que é injetado no modelo reversamente no tempo). O artefato gerado é de alta amplitude, e sua extensão pode contaminar a imagem dos refletores mais rasos. Visto que o gradiente (calculado pelo método adjunto) é a RTM do resíduo, a correção desse problema depende então de eliminar (ou atenuar) o evento da onda direta nesse gradiente. Uma maneira de mitigar o problema é eliminá-la dos dados observados e calculados, de forma que ela não aparecerá no resíduo, e consequentemente também não haverá artefatos no gradiente (ou haverá, mas atenuados). Essa foi a abordagem adotada nesta dissertação, e os resultados podem ser vistos na Figura 5.5. A diferença entre os sismogramas com e sem onda direta são mostradas na Figura 5.6.

Outra forma de corrigir o problema é zerar as primeiras amostras do gradiente, eliminando o evento da onda direta, e no modelo inicial colar as primeiras amostras do modelo verdadeiro (Klimm, 2013). Essa forma de correção é mostrada na Figura 5.7. Apesar de efetiva e de implementação mais simples, essa abordagem implica perda de informação dos refletores mais rasos, visto que devem ser zeradas todas as amostras do gradiente que contém este artefato (Figura 5.7-b), o que acaba zerando a imagem dos refletores mais rasos tam-

bém. Aliado a isso, colar as amostras do modelo verdadeiro no modelo inicial, na extensão do artefato da onda direta (Figura 5.7-c), acaba por limitar a inversão do campo de velocidades na prática. Por exemplo, nesse caso do modelo Marmousi-Teste, o artefato da onda direta chega a ocupar as 10 primeiras amostras do gradiente, de forma que se ignora do processo de inversão cerca 240 m da subsuperfície ($10 \times 24,0$ m). Nesta dissertação optou-se por filtrar a onda direta dos tiros, e quando necessário zerar as primeiras amostras do gradiente e inserir uma fatia de velocidades verdadeiras no topo do modelo inicial (tipicamente, a fatia tem altura de 2 a 5 pontos).

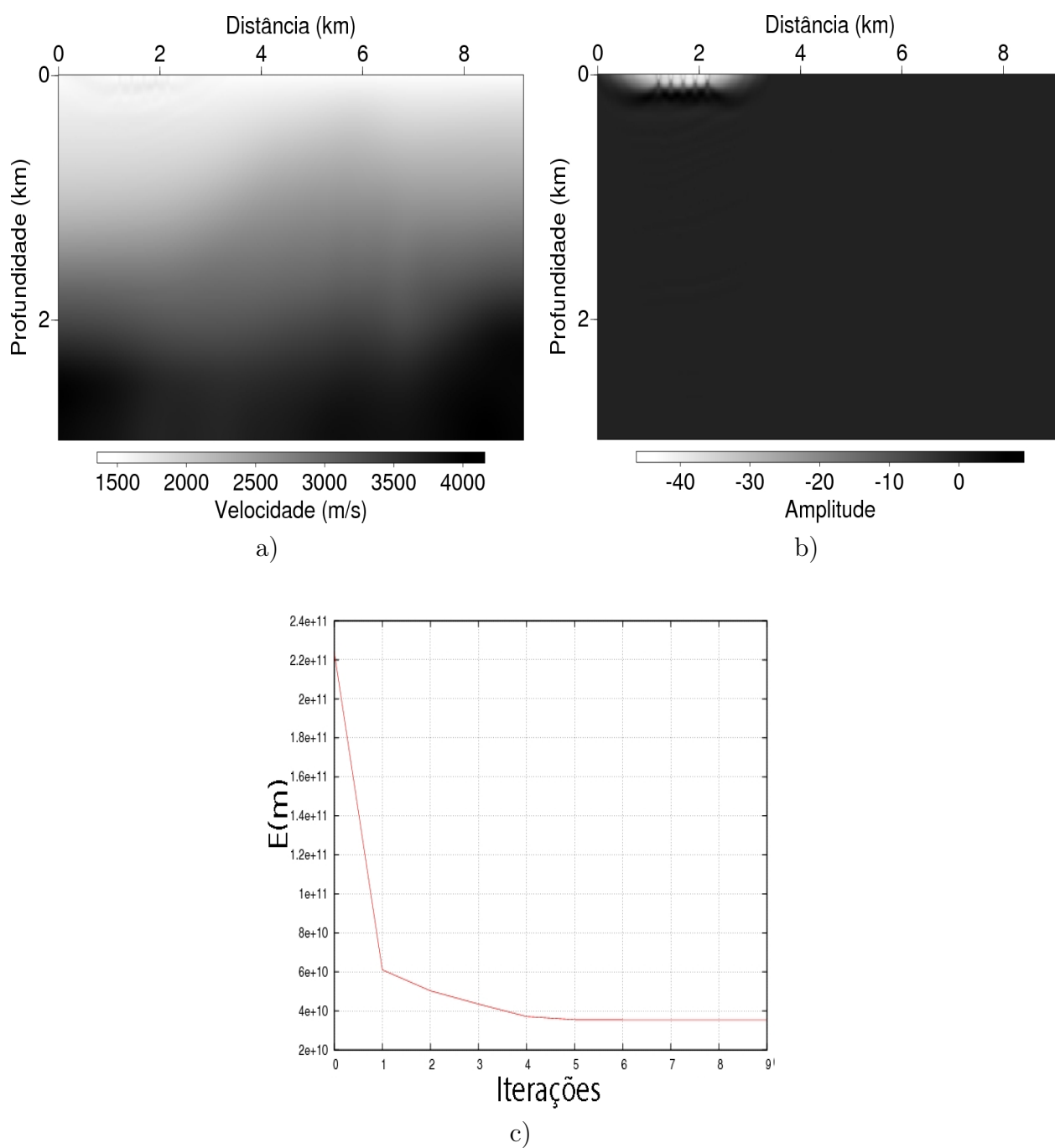


Figura 5.4: 10 iterações da FWI, 5 tiros, sem eliminar a onda direta. a) Modelo de velocidades após 10 iterações; b) Primeiro gradiente; c) Função objetivo.

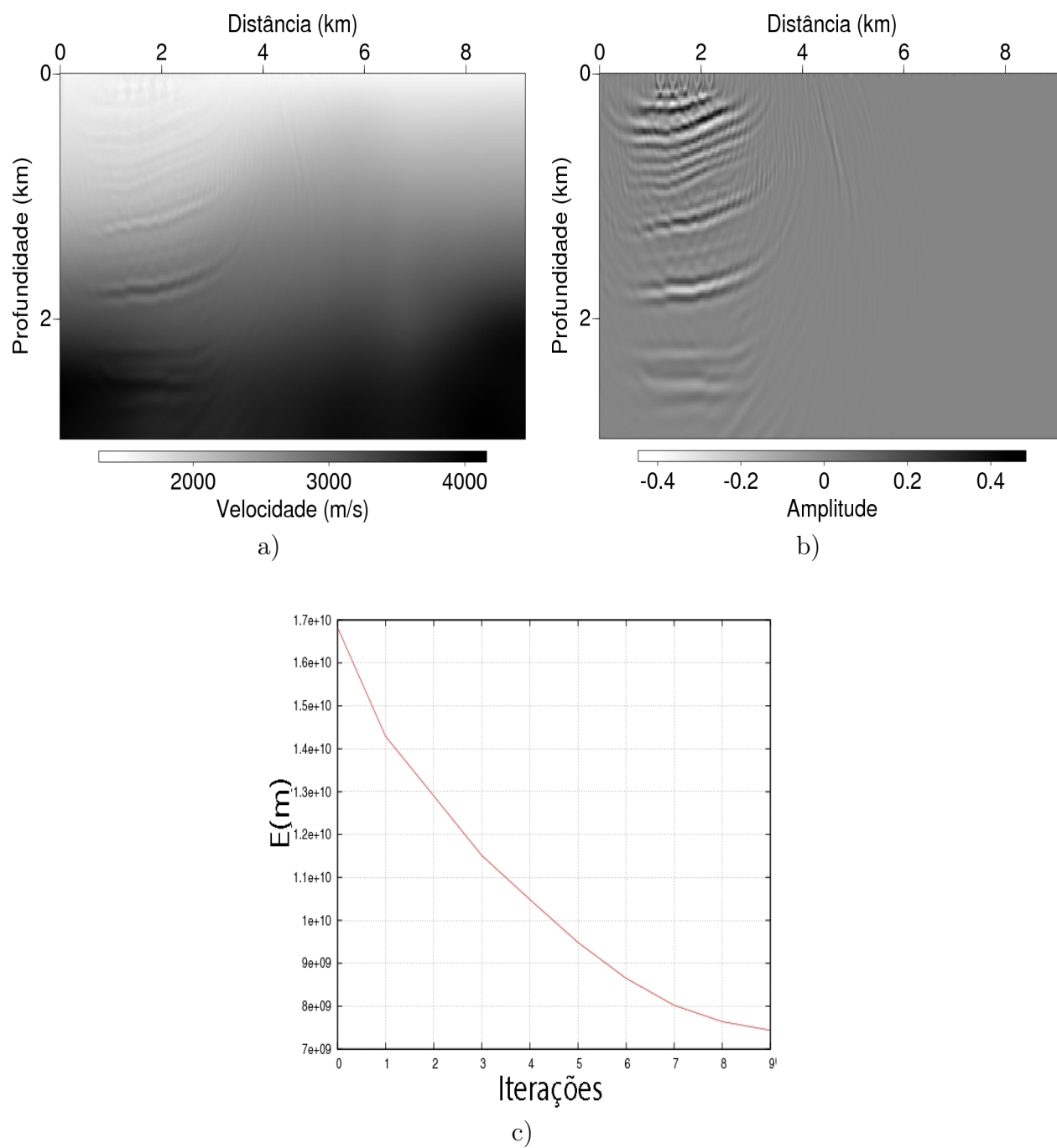


Figura 5.5: 10 iterações da FWI, 5 tiros, eliminando a onda direta. a) Modelo de velocidades após 10 iterações; b) Primeiro gradiente; c) Função objetivo.

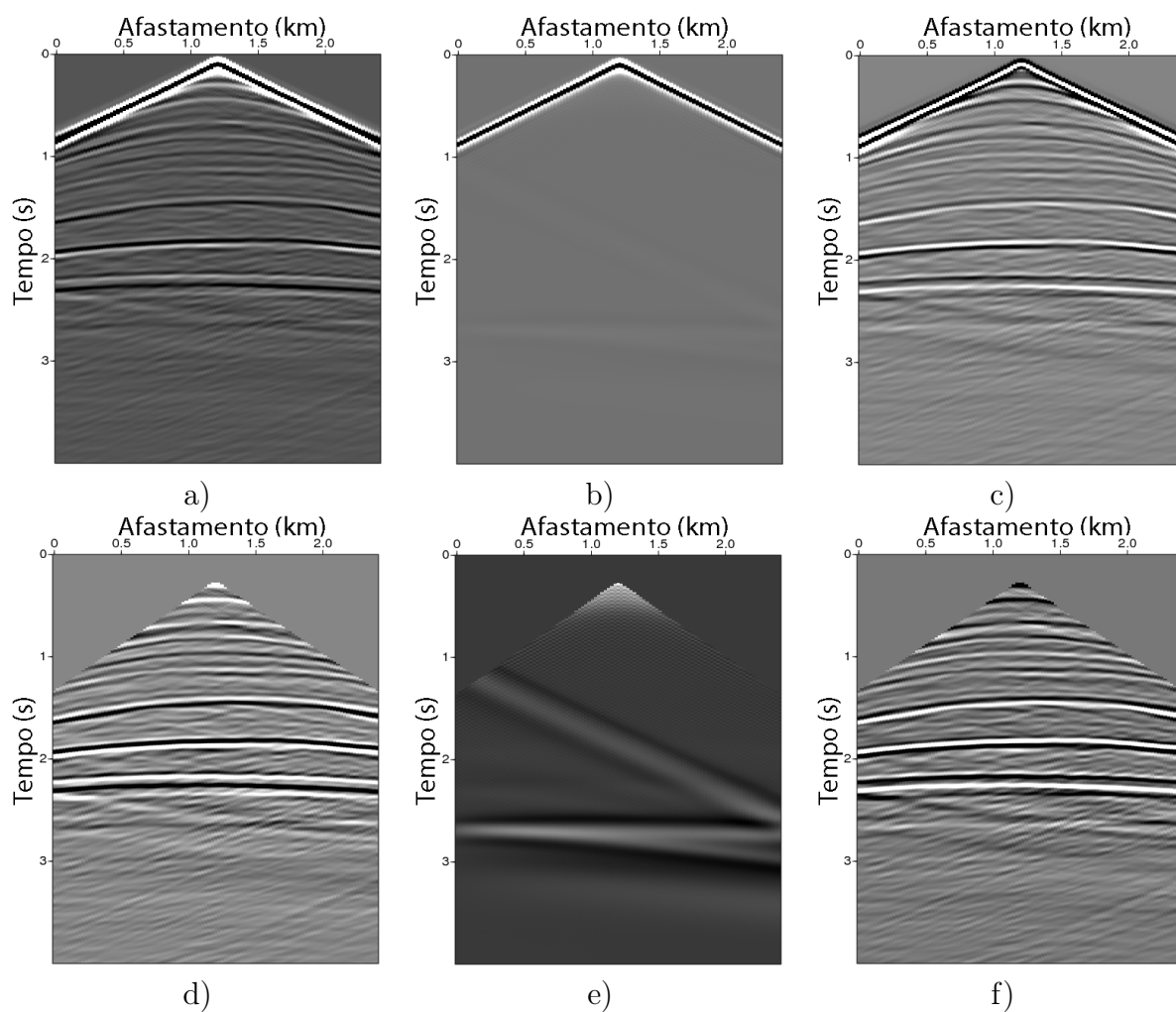


Figura 5.6: Sismogramas para o primeiro tiro da primeira iteração da FWI. a) $\mathbf{d}_{\text{obs}}$ com onda direta. b) $\mathbf{d}_{\text{cal}}$ com onda direta. c) $\mathbf{d}_{\text{res}}$ (resíduo) com onda direta. d) $\mathbf{d}_{\text{obs}}$ sem onda direta. e) $\mathbf{d}_{\text{cal}}$ sem onda direta. f) $\mathbf{d}_{\text{res}}$ (resíduo) calculado com $\mathbf{d}_{\text{obs}}$ e $\mathbf{d}_{\text{cal}}$ sem onda direta.

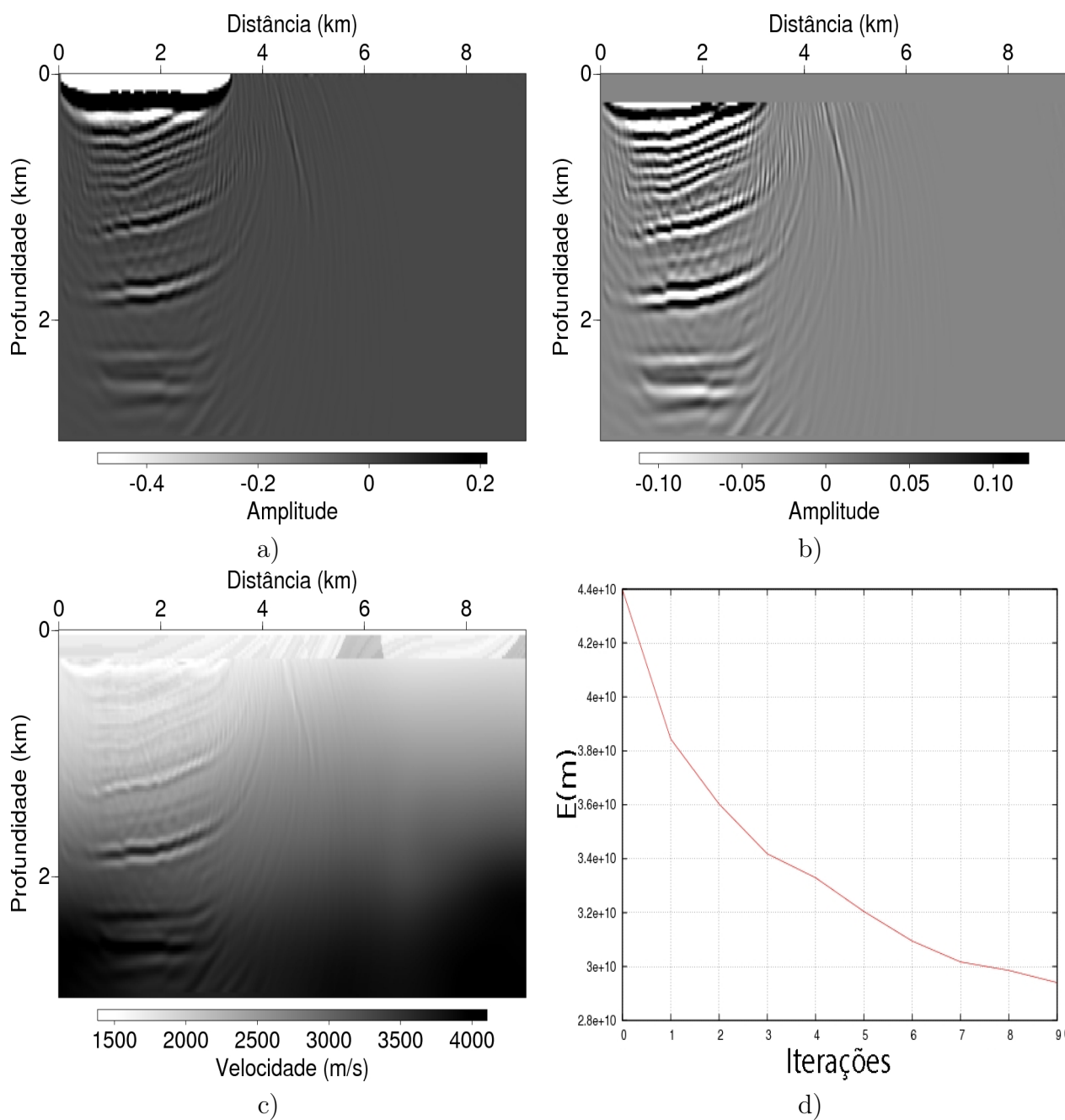


Figura 5.7: Resultado de 10 iterações da FWI, 5 tiros, eliminando o artefato da onda direta pela zeragem das primeiras amostras do gradiente. a) Gradiente original da iteração 1, com artefato da onda direta. b) Gradiente da iteração 1 com as 10 primeiras amostras zeradas. c) Modelo de velocidades ao fim das iterações, com fatia do modelo verdadeiro colada no topo. d) Função objetivo.

5.3 Análise dos operadores de modelagem

Para a modelagem sísmica foram testados os operadores de diferenças finitas, de ordem 2 a 14, o operador pseudo-espectral, o REM-DF e o REM-pseudo-espectral. Em resumo, o operador de DF é normalmente o mais rápido, mas não permite uma frequência máxima não dispersiva tão grande quanto o operador pseudo-espectral, que é mais lento. Sabe-se que para calcular o resíduo e fazer a FWI, a amostragem do dado calculado deve ser a mesma do dado observado. É uma ocorrência comum, quando se testam vários modelos de velocidades de dimensões diferentes, que a condição de estabilidade exija um passo temporal (Δt) muito menor que a amostragem do dado observado (por exemplo, o dado observado é amostrado em 0,004 s, mas a condição de estabilidade exige um $\Delta t = 0,001$ s). Normalmente se resolve esse problema modelando o dado calculado a esse Δt menor, e depois reamostrando-o na migração e nos tiros, o que exige uma reimplementação de várias etapas do código para incluir reamostragem. Todavia, com o método de expansão rápida, isso não é necessário; pode-se modelar os campos a grandes Δt 's (até o limite de Nyquist), tanto para o REM-DF quanto para o REM-pseudo-espectral (seção 1.2.3). Como será visto nessa seção, utilizar a abordagem REM ao invés da abordagem de reamostrar gera economia memória e simplicidade de implementação, a um tempo computacional apenas ligeiramente maior. O REM é então uma ferramenta que permite alta versatilidade nos testes de modelagem (e consequentemente, RTM e FWI). Além disso, é importante ressaltar que, como mostrado por dos Santos e Pestana (2015), nos tempos mais avançados os operadores de DF e pseudo-espectral acabam distorcendo a forma da onda, o que não ocorre quando se usa os operadores REM. Trata-se, então, de um método mais preciso. Essa conclusão já foi abordada na seção 1.2.3, onde se mostrou que os operadores usuais sempre usam DF de 2ª ordem para aproximar a derivada temporal, enquanto o REM equivale a aproximações temporais de mais alta ordem, dependendo do número de termos M utilizado na expansão. Neste trabalho, a distorção da forma de onda nos tempos mais tardios não afetou os resultados da inversão.

A escolha do operador mais adequado a cada problema depende então das variáveis memória, custo computacional, tamanho do modelo, frequência máxima desejada (que afeta a resolução final do modelo de velocidades), e a amostragem do dado observado. A análise de cada operador à luz desses fatores é discutida a seguir.

Os testes para os operadores foram feitos no modelo Marmousi-Teste (Figura 5.1(a)).

Operador	Tempo por tiro	Máximo Δt estável	Máxima f não dispersiva
DF ordem 2	0,288s	0,00350s	14 Hz
DF ordem 4	0,307s	0,00270s	16 Hz
DF ordem 6	0,325s	0,00230s	22 Hz
DF ordem 8	0,344s	0,00227s	24 Hz
DF ordem 10	0,362s	0,00222s	25 Hz
DF ordem 12	0,380s	0,00220s	26 Hz
DF ordem 14	0,399s	0,00218s	26 Hz

Tabela 5.1: Modelagem na GPU de 1 tiro, para diferentes operadores de DF. $\Delta t = 0,001s$, $nt = 4000$.

5.3.1 Comparação entre diferentes operadores de DF

Primeiro foram analisados os operadores de diferenças finitas (DF) para diversas ordens. Para a condição de dispersão, utilizou-se a fórmula 1.32 como base e uma análise visual (se modelou além do limite da fórmula 1.33, analisando visualmente os sismogramas para verificar eventos dispersivos). Para análise de estabilidade foi usada a equação 1.22.

Supondo um dado observado amostrado a $0,001s$, se modelou 1 tiro a $0,001s$, $nt = 4000$ (iterações temporais), para vários operadores de DF, mostrados na Tabela 5.3.1.

Se pôde perceber que conforme aumenta o tamanho do operador, aumenta o tempo de processamento, reduz o máximo Δt estável e aumenta a máxima frequência não-dispersiva. A frequência máxima não dispersiva não difere muito entre os operadores de ordem 10, 12, e 14, enquanto o tempo de processamento cresce sempre de maneira linear. A partir dessa comparação, se escolheu o operador de DF de ordem 10 para os testes, visto que tem a mesma frequência máxima que operadores maiores, e um tempo menor de processamento.

5.3.2 Comparação entre operadores DF, pseudo-espectral, REM-DF e REM-pseudo-espectral

A partir do teste anterior, se adotou como base o operador DF de ordem 10. Aqui são comparados então os operadores DF (ordem 10), pseudo-espectral (ou PS), REM-DF (ordem 10) e REM pseudo-espectral (ou REM-PS). As condições de dispersão e estabilidade do operador de DF são dadas por 1.33 e 1.22, respectivamente. A estabilidade do operador PS é dada pela equação 1.45, e a dispersão pela equação 1.46. O REM-DF e o REM-PS são quase sempre estáveis para grandes intervalos de Δt (tipicamente até 32 ms), e se deve respeitar apenas às duas condições de dispersão dadas na seção 1.2.3. Novamente, as análises de dispersão foram feitas de maneira visual, modelando acima do limite das regras

Operador	Tempo por tiro	Máximo Δt estável (REM: máx. Δt não dispersivo)	Máxima f não dispersiva
DF 10	0,362s	0,00223s	25 Hz
REM-DF 10	1,043s	0,0240s	25 Hz
PS	0,741s	0,00196s	40 Hz
REM-PS	2,165s	0,012s	40 Hz

Tabela 5.2: Modelagem na GPU de 1 tiro, operadores DF, PS e REM. $\Delta t = 0,001$ s, $nt = 4000$.

de dispersão.

Supondo mais uma vez um dado observado amostrado a 0,001 s, se modelou 1 tiro a 0,001 s, $nt = 4000$, para os quatro operadores citados, mostrados na Tabela 5.3.2.

Analisando esses resultados, se percebe que os operadores PS permitem modelar a frequências máximas não dispersivas cerca de 37% maiores que os operadores DF. Todavia, o tempo de processamento é superior; conclui-se então que a escolha entre DF e PS dependerá se o problema exige maior conteúdo frequência dos dados calculados, ou menor tempo de processamento.

Já para os operadores REM, percebe-se que o tempo de processamento é sempre maior que o dos seus "operadores base"(PS ou DF); as frequências máximas são as mesmas; e o passo máximo temporal não dispersivo é cerca de 10 vezes maior. Desta forma, se conclui que, se for possível modelar ao Δt do dado observado de maneira estável e sem dispersão, não é vantajoso utilizar os operadores REM. Todavia, isso nem sempre é possível. Muitas vezes a amostragem do dado observado é maior que o Δt estável dos métodos DF e PS, de forma que deve-se executar a modelagem ao Δt estável, e depois reamostrar ao Δt do dado observado. Isso demanda mais tempo de processamento e mais memória utilizada. O operador REM permite resolver este problema sem necessidade de reamostragem, a tempos de processamento na mesma ordem de grandeza e a um custo de memória inferior. O teste a seguir elucida essas vantagens.

Supondo um dado observado amostrado a 0,004 s, foram testados novamente os 4 operadores, de forma a obter um dado calculado (tanto cubo de pressões quanto sismogramas) a 0,004 s, foi gerada a Tabela 5.3.2.

Por esses resultados, concluímos que se as condições de estabilidade impuserem uma modelagem a um Δt menor que o Δt do dado observado, é mais vantajoso usar o operador REM (tanto DF quanto PS): ele permite modelar exatamente ao Δt desejado, com economias

Operador	Δt	nt	Tempo por tiro	Máximo Δt estável (REM: máx. Δt não dispersivo)	Máxima f não dispersiva	Memória do cubo P ($nz \times nx \times nt \times 4$)
DF 10	0,002s	2000	0,181s	0,00223s	25 Hz	656 MB
REM-DF 10	0,004s	1000	0,486s	0,024s	25 Hz	328 MB
PS	0,0015s	2667	0,491s	0,00196s	40 Hz	875 MB
REM-PS	0,004s	1000	1.048s	0,012s	40 Hz	328 MB

Tabela 5.3: Modelagem na GPU de 1 tiro, operadores DF, PS e REM, para um d_{obs} de 0,004 s.

de 50% da memória utilizada, a um tempo 60% maior (para esse teste). Além disso, há outra grande vantagem, que é a implementação. Incluir a reamostragem a Δt 's mais baixos na modelagem e na correlação cruzada da RTM implica numa implementação mais complexa, com mais variáveis auxiliares e mais memória utilizada. Dessa maneira, o uso do REM permite um código enxuto e com menor gasto de memória, a um tempo de computação ligeiramente maior.

A memória do cubo P na última coluna da tabela foi calculada pois (como mostrado no Capítulo 2) ele é parte fundamental do algoritmo da RTM. Quanto menor o tamanho desse cubo, melhor, pois há maior liberdade de testes com modelos de velocidades maiores (visto que a GPU usada tem memória de 5,2 GB). Aqui ele foi calculado já com adição das bordas absorvedoras (de tamanho 32 amostras); o tamanho total do modelo vai então de 125×383 para 192×448 amostras, de forma que o tamanho do cubo (em número real de 4 *bytes*) é de $192 \times 448 \times 4 \times nt$.

Em resumo, os operadores PS são os que permitem a maior frequência não dispersiva, mas são também mais lentos que os operadores DF. Os operadores REM tem maior custo computacional que seus "operadores base", menor custo de memória, e são boas soluções para problemas que exijam modelagem a grandes passos no tempo (Δt).

5.4 Análise dos métodos de inversão

Em problemas lineares a direção do gradiente da iteração atual é sempre perpendicular à direção do gradiente da iteração subsequente, gerando uma convergência em "zigue-zague" para o mínimo local (Figura 5.8; Nocedal e Wright, 2006). No caso não-linear, pode-se ainda usar o método gradiente, fazendo esse mesmo zigue-zague decorrente do uso da direção gradiente a cada iteração; todavia, de forma a considerar a não linearidade, procura-se buscar os α (o quanto que se "caminha" na direção daquela iteração) que, para aquela iteração, fazem a função objetivo diminuir; se aumentar, faz-se a busca novamente. Existem diversos critérios para realizar essa busca, dados pelos chamados métodos de busca em linha (Nocedal e Wright, 2006).

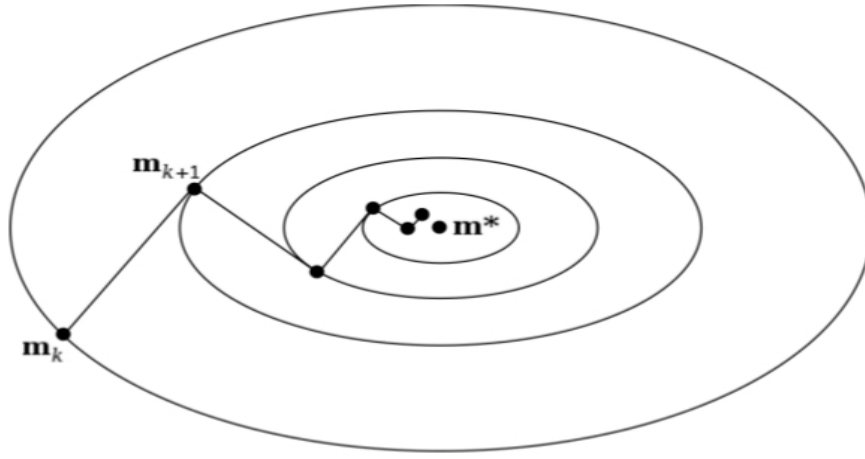


Figura 5.8: Convergência da função objetivo para o método gradiente. Retirado de Nocedal e Wright (2006).

Nos métodos gradiente com busca em linha, se usa a direção do gradiente a cada iteração, algum critério para determinar o α inicial da iteração (como Barzilai-Borwein, entre outros; dos Santos, 2013), e métodos de busca em linha (como as condições de Wolfe, entre outros; dos Santos, 2013) para garantir que esse α vai gerar uma redução da função objetivo na iteração seguinte. Ou seja, a direção usada a cada iteração é a direção do problema linear (gradiente da função objetivo na iteração atual) e a não linearidade é incorporada apenas no passo α . Para a iteração atual k , a equação que atualiza o modelo $\mathbf{m}$ é dada por:

$$\mathbf{m}_{k+1} = \mathbf{m}_k - \alpha \nabla E(\mathbf{m}_k). \quad (5.1)$$

O uso de métodos gradientes com busca em linha é muito intensivo computacionalmente, pois toda vez que se tenta um α diferente pela busca em linha, é necessário remodelar $\mathbf{d}_{\text{cal}}$ e comparar com $\mathbf{d}_{\text{obs}}$, de forma a verificar se o resíduo (e consequentemente, a função objetivo) diminui realmente. Na tabela 5.4, retirada e adaptada de dos Santos (2013), cada

Tabela 5.4: Tempos de processamento para a FWI utilizando seis métodos gradiente com critérios de passo inicial diferentes, e para o método L-BFGS com passo unitário. Retirado de dos Santos (2013).

Método	Total de iterações de busca em linha	Tempo total de processamento
Gradiente com α^{med}	203	22 h 49 min 28 s
Gradiente com α^{max}	287	33 h 54 min 03 s
Gradiente com α^{intp}	209	23 h 59 min 39 s
Gradiente com α^{bb1}	36	4 h 8 min 13 s
Gradiente com α^{bb2}	68	7 h 36 min 39 s
Gradiente com α^{abb}	38	4 h 27 min 39 s
L-BFGS com α^1	30	3 h 21 min 35 s

teste da FWI partiu de condições iniciais idênticas e obteve a mesma convergência final para a função objetivo. Pode-se observar que os métodos gradiente com busca em linha - cada um com um critério de passo inicial: médio, máximo, Barzilai-Borwein, entre outros; consultar dos Santos (2013) para maiores detalhes - tem um tempo computacional muito superior ao método não linear L-BFGS (a ser discutido a seguir). Vale observar que nos testes dessa tabela foi usada uma implementação paralela (que usou *cluster* de computadores) distinta da desta dissertação (que usou placas gráficas); sua inclusão serve apenas para elucidar que o método não linear L-BFGS obteve o mesmo resultado que os métodos lineares de busca em linha, a um custo computacional muito inferior.

Outros métodos, como o conjugado-gradiente não linear e o L-BFGS, incluem a não linearidade do problema tanto na direção de busca, quanto no passo α , atualizando $\mathbf{m}$ na iteração atual k por:

$$\mathbf{m}_{k+1} = \mathbf{m}_k - \alpha \mathbf{s}_k, \quad (5.2)$$

sendo $\mathbf{s}$ a direção otimizada de busca. No método conjugado-gradiente, a direção $\mathbf{s}_k$ é dada por uma combinação do gradiente atual $\nabla E(\mathbf{m}_k)$ e a direção da iteração anterior $\mathbf{s}_{k-1}$. Ou seja, faz uso de apenas uma iteração anterior ao otimizar a direção de busca (equações 3.28, 3.29, seção 3.2.3). Já para o passo α a otimização para incluir não linearidade poderia ser feita da mesma forma que para os métodos gradiente vistas no parágrafo anterior; todavia, escolheu-se a abordagem de Yang et al. (2015), baseada em Pica et al. (1990), no cálculo de α :

Tabela 5.5: Tempos de processamento da FWI para diferentes passos do L-BFGS, e um mesmo valor final da função objetivo. Retirado de dos Santos (2013).

Método	Total de iterações de busca em linha	Tempo total de processamento
L-BFGS com α^{med}	170	19 h 2 min 17 s
L-BFGS com α^{max}	280	31 h 24 min 40 s
L-BFGS com α^{intp}	202	22 h 53 min 17 s
L-BFGS com α^1	30	3 h 21 min 35 s

$$\alpha_k = \frac{\langle \nabla E(\mathbf{m}_k), \mathbf{s}_k \rangle}{\langle J_k \mathbf{s}_k, J_k \mathbf{s}_k \rangle},$$

$$J_k \mathbf{s}_k \approx \frac{G(\mathbf{m}_k + \epsilon \mathbf{s}_k) - G(\mathbf{m}_k)}{\epsilon},$$

$$\epsilon \leq \frac{\max(|\mathbf{m}_k|)}{100 \max(|\mathbf{s}_k|)}. \quad (5.3)$$

que são as equações 3.22, 3.26, 3.27, respectivamente, já mostradas no Capítulo 3. Nessa abordagem não é feita busca em linha (que pode demandar diversas modelagens) e o comprimento inicial de α é dado pelas equações 5.3 (que demanda apenas uma modelagem na direção $\mathbf{s}$).

O método L-BFGS também atualiza $\mathbf{m}$ de acordo com a equação 5.2, onde $\mathbf{s}$ é dado por $-\mathbf{r}$ (descrito na seção 3.3), e o α pode ser dado por diversos critérios. Argumenta-se que o método L-BFGS é bem escalado, e portanto deve ter $\alpha = 1$ (Nocedal e Wright, 2006). dos Santos (2013) testou essa hipótese e analisou o L-BFGS com diversos métodos de busca em linha e passos iniciais, como mostra a Tabela 5.7, retirada da mesma referência. A hipótese de que o passo unitário resulta na melhor convergência (em detrimento do uso de passo α inicial e busca em linha) para o L-BFGS foi confirmada. Na seção a seguir, testaremos a mesma hipótese, mas verificando o L-BFGS com $\alpha = 1$ e o L-BFGS com passo inicial α dado pelas equações 5.3 (Yang et al., 2015; Pica et al., 1990).

5.4.1 L-BFGS com passo variável vs. L-BFGS com passo unitário

Como já mencionado anteriormente, a implementação da FWI nesta dissertação requer 4 modelagens por iteração: 3 para o cálculo do gradiente (RTM com borda efetiva) e 1 para estimar o passo α pelas equações 5.3. Se confirmado que o melhor passo para o L-BFGS

é $\alpha = 1$, a quarta modelagem por iteração da FWI não seria necessária, gerando uma redução de custo computacional de aproximadamente 25% por iteração. Os testes desta seção pretendem verificar essa hipótese. Eles são similares aos realizados por dos Santos (2013) (Tabela 5.7), com a diferença de que aqui não é feita a busca em linha; são testados o L-BFGS para $\alpha = 1$ e para α dado pelas equações 5.3, e analisadas as convergências das funções objetivo de cada um.

Como mostrado no capítulo 3, o L-BFGS aproxima a Hessiana fazendo uso dos gradientes das iterações anteriores; tipicamente, se utiliza $m = 10$ (Nocedal e Wright, 2006), ou seja, 10 gradientes anteriores. Na primeira iteração, é sempre necessário outro método para inicializar o L-BFGS; utilizou-se aqui o método gradiente, com o passo dado pelas equações 5.3. Dessa maneira, a primeira iteração do método L-BFGS sempre demandará 4 modelagens, devido à inicialização. Tomando como exemplo uma rodada da FWI com 30 iterações, nas iterações 2 a 10, o cálculo da direção pelo L-BFGS primeiro utiliza 1 gradiente anterior ($m = 1$), 2 gradientes anteriores ($m = 2$), 3 ($m = 3$), 4 ($m = 4$), e assim por diante, até chegar à iteração 11, onde estão disponíveis 10 gradientes anteriores; da iteração 11 a 30, se usa $m = 10$, como pretendido. De forma a responder à pergunta de qual o melhor passo α em cada uma dessas iterações, foram feitos três testes:

1. $\alpha = 1$ em todas as iterações (exceto nas de inicialização);
2. α calculado por 5.3 nas iterações 1 a 10, $\alpha = 1$ nas iterações 11 a 30;
3. α calculado por 5.3 em todas as iterações.

Para cada um desses 3 testes, foi feita a FWI multiescala com 4 rodadas de 30 iterações cada (totalizando 120 iterações). É importante ressaltar que não se deve utilizar gradientes de rodadas anteriores no início da rodada seguinte, pois os resíduos de uma rodada (que geram os gradientes) tem conteúdo de frequência distintos dos resíduos de outra rodada, de forma que não devem ser incluídos no mesmo processo.

Os resultados são mostrados na Figura 5.9. O resultado final dos 3 testes é muito parecido; com base no erro RMS de cada um, é possível observar que o uso de α variável em todas as iterações dá os melhores resultados, seguido pelo uso do α nas iterações de 1 a 10, seguido pelo $\alpha = 1$ em todas as iterações. Na Figura 5.9(e) são plotados os α para cada um dos testes; pode-se observar que ele oscila em torno do valor 1 nas iterações 11 a 30 de cada rodada (11-30, 31-40, 61-70, 91-100), sendo menos próximo de 1 nas iterações 1 a 9 de cada rodada. Isso pode ser atribuído ao fato de que nas primeiras iterações, onde o L-BFGS usa menos gradientes, a aproximação da Hessiana ainda não é tão boa; conforme

são incorporados mais gradientes, a estimativa do passo pelas 5.3 passa a oscilar em torno de 1.

A Figura 5.10 mostra os mesmos testes para o método pseudo-espectral (PS), com as mesmas conclusões do parágrafo anterior. Vale ressaltar que esses testes são apenas um guia para a FWI usando mais tiros e modelos maiores; as possibilidades 1, 2 e 3 devem ser avaliadas em cada caso. Em resumo, o uso de passo variável dado pelas equações 5.3 no L-BFGS melhora os resultados, mas tem o custo computacional de 1 modelagem a mais por iteração. Fica a critério daquele que testa escolher se o custo computacional extra para o passo compensa a melhoria (ou é imprescindível à convergência).

Tabela 5.6: Tempos de processamento (GPU) de 120 iterações da FWI para os três testes com o passo do L-BFGS, e operador DF.

Método	Tempo iteração α variável	Tempo por iteração $\alpha = 1$	Tempo total de processamento (GPU)
L-BFGS (1)	22,6 s	16,7 s	33 min 46 s
L-BFGS (2)	22,6 s	16,7 s	37 min 20s
L-BFGS (3)	22,6 s	-	45 min 14 s

Tabela 5.7: Tempos de processamento (GPU) de 120 iterações da FWI para os três testes com o passo do L-BFGS, e operador PS.

Método	Tempo iteração α variável	Tempo por iteração $\alpha = 1$	Tempo total de processamento (GPU)
L-BFGS (1)	48,4 s	36,1 s	59 min 06 s
L-BFGS (2)	48,4 s	36,1 s	1 h 3 min 20s
L-BFGS (3)	48,4 s	-	1 h 16 min 58 s

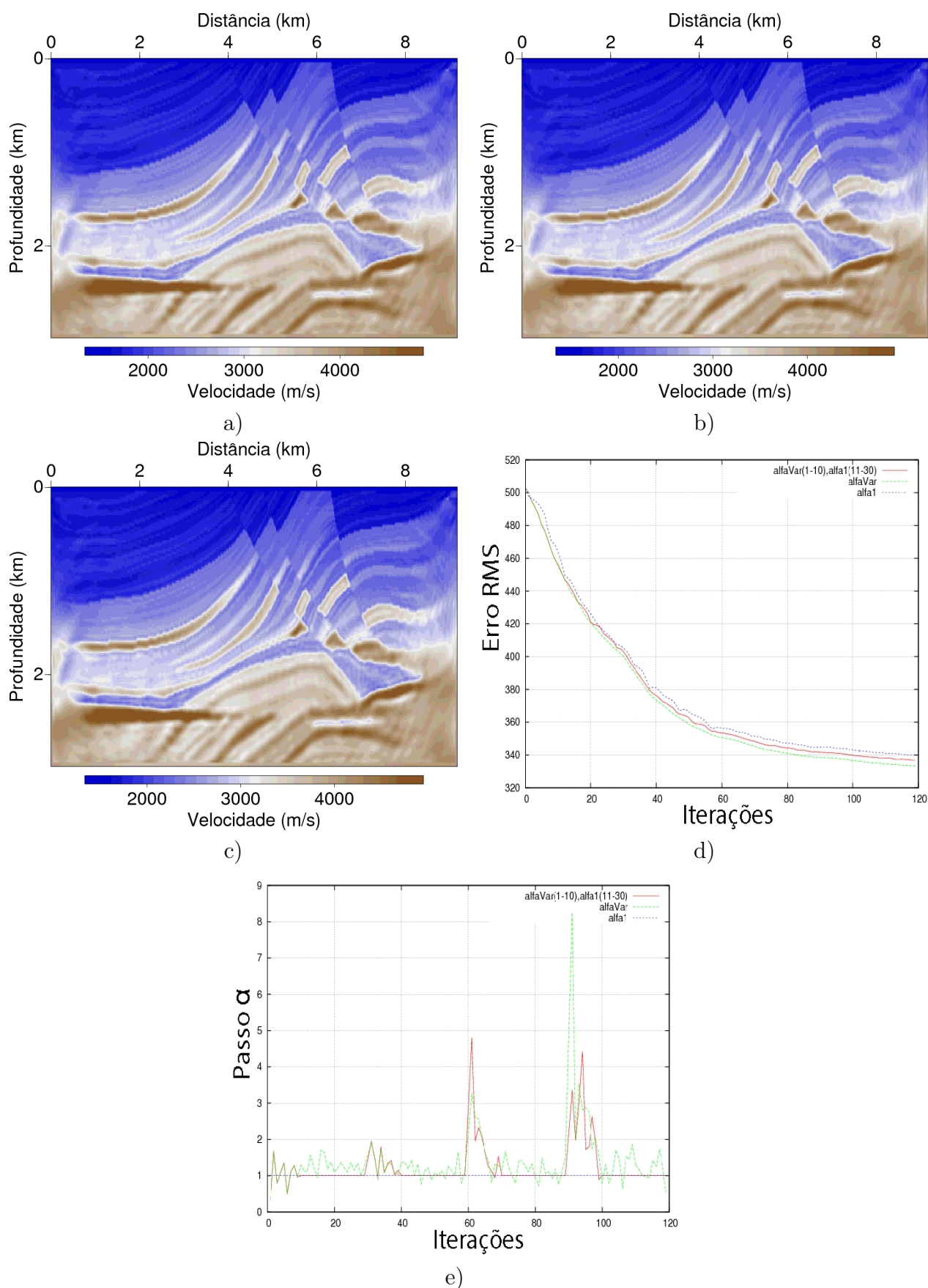


Figura 5.9: Resultados da FWI para os testes 1, 2 e 3 para o L-BFGS, operador DF. a) Teste 1: $\alpha = 1$ em todas as iterações (exceto nas de inicialização). b) Teste 3: α variável nas iterações 1 a 10, $\alpha = 1$ nas iterações 11 a 30 (de cada rodada). c) Teste 2: α variável em todas as iterações. d) Erro RMS para cada um dos testes. e) Valores de α ao longo das iterações para cada um dos testes.

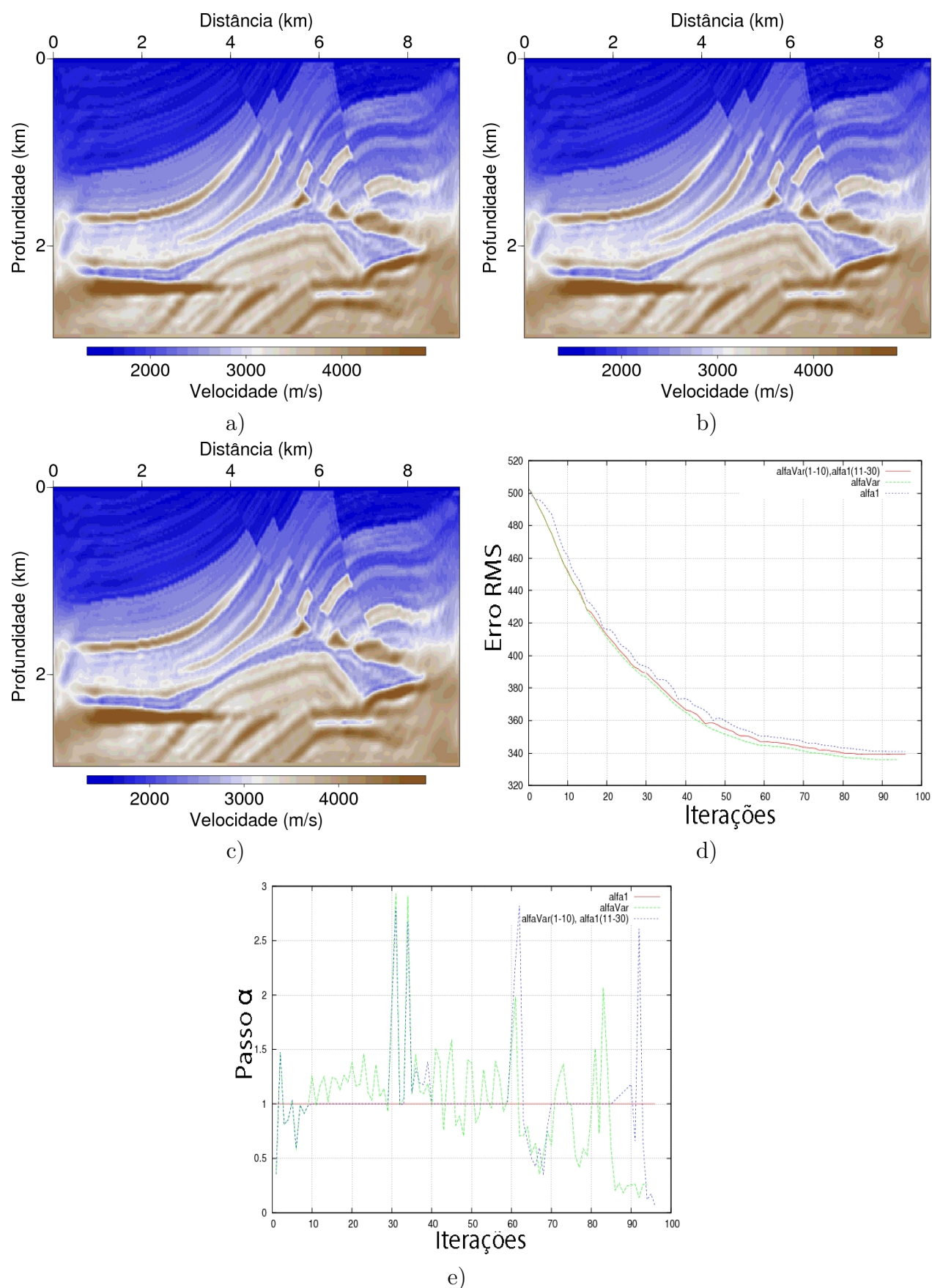


Figura 5.10: Resultados da FWI para os testes 1, 2 e 3 para o L-BFGS, operador PS. a) Teste 1: $\alpha = 1$ em todas as iterações (exceto nas de inicialização). b) Teste 3: α variável nas iterações 1 a 10, $\alpha = 1$ nas iterações 11 a 30 (de cada rodada). c) Teste 2: α variável em todas as iterações. d) Erro RMS para cada um dos testes. e) Valores de α ao longo das iterações para cada um dos testes.

5.4.2 L-BFGS vs. Conjugado Gradiente não-linear

Nesta subseção são comparados os resultados entre o método conjugado gradiente não linear (seção 3.2.3), com passo α variável (dado pelas equações 5.3), e L-BFGS com $\alpha = 1$ (exceto na iteração inicial, onde α é variável; seções 3.3 e 5.4.1). Os resultados são mostrados na Figura 5.11. Percebe-se que o conjugado gradiente tem resultado pior, pois converge para um mínimo local, como mostra o erro RMS comparativo em 5.11-c. Além disso, como requer 4 modelagens por iteração (3 para o gradiente, 1 para o passo α), versus 3 modelagens por iteração do L-BFGS (exceto na primeira iteração de cada rodada), tem um tempo computacional superior, como mostra a Tabela 5.8.

Tanto o conjugado gradiente não linear quanto o L-BFGS incorporam a não linearidade na direção $\mathbf{s}$ (como mostrado no início desta seção). A superioridade do L-BFGS pode ser atribuída ao fato de que ele incorpora vários gradientes anteriores na direção atual (tipicamente, 10 gradientes anteriores), enquanto que o conjugado gradiente incorpora apenas o gradiente da iteração anterior na direção atual. Dado que o L-BFGS apresentou melhores resultados a um tempo computacional inferior, ele foi o método de escolha em todas as FWI da seção "Resultados".

Tabela 5.8: Tempos de processamento (GPU) para 120 iterações numa FWI multiescala, para os métodos CG não linear e L-BFGS.

Método	Tempo total de processamento (GPU)
CG não linear	42 min 15 s
L-BFGS	37 min 20s

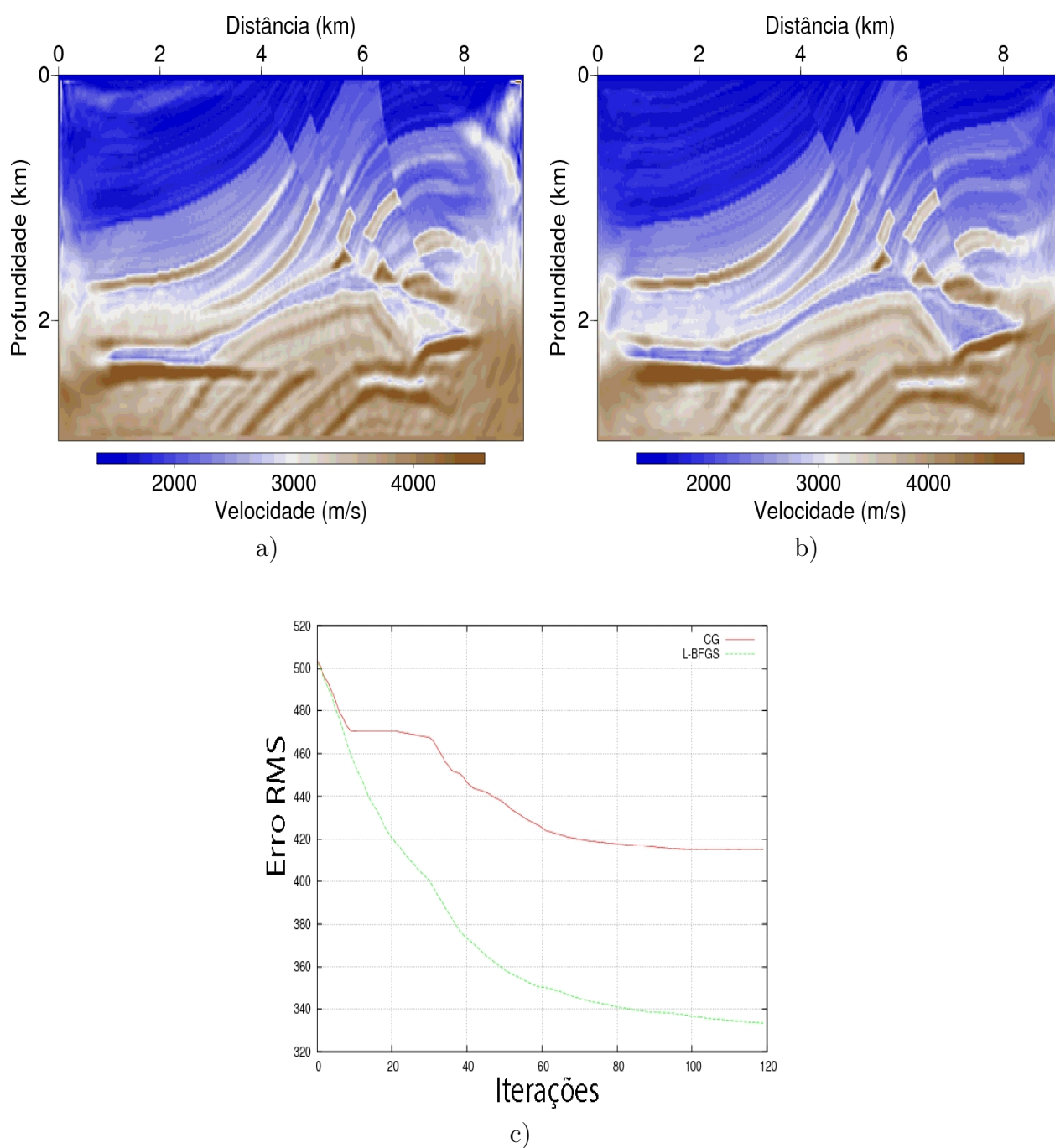


Figura 5.11: Comparação entre métodos Conjugado Gradiente não linear e L-BFGS, 120 iterações da FWI multiescala. a) Resultado para CG. b) Resultado para L-BFGS. c) Erro RMS para os resultados em (a), (b).

5.5 Notas sobre $\mathbf{d}_{\text{obs}}$ sintético

Nesta dissertação não se trabalhou com dados reais, ou seja, o dado observado $\mathbf{d}_{\text{obs}}$ é sintético. Esta seção confirma a hipótese esperada de que, quanto mais próximos o operador e condições de borda utilizados para gerar $\mathbf{d}_{\text{obs}}$ dos utilizados para gerar os $\mathbf{d}_{\text{cal}}$ na inversão, melhor o resultado da FWI; todavia, também se mostra que o uso de operadores e condições de borda diferentes para $\mathbf{d}_{\text{obs}}$ e $\mathbf{d}_{\text{cal}}$ não inviabiliza a inversão.

5.5.1 Influência do operador de modelagem

Primeiro são comparadas inversões com $\mathbf{d}_{\text{obs}}$ e $\mathbf{d}_{\text{cal}}$ sob as mesmas condições de borda e mesmo operador, para verificar se há maior precisão no resultado, se aumentado o tamanho do operador. Foram testados os 28 tiros do modelo Marmousi com frequência máxima de 25 Hz. Como todos os operadores (de ordem 10, 12 e 16) obedecem à condição de dispersão (Tabela 5.3.1), se esperam resultados muito próximos, e é o que efetivamente ocorre, como mostra a Figura 5.12.

A seguir é testada a FWI para:

1. $\mathbf{d}_{\text{obs}}$ gerado com operador pseudo-espectral (PS), $\mathbf{d}_{\text{obs}}$ gerado com operador DF12
2. $\mathbf{d}_{\text{obs}}$ gerado com operador DF12, $\mathbf{d}_{\text{obs}}$ gerado com operador DF12
3. $\mathbf{d}_{\text{obs}}$ gerado com operador DF12, $\mathbf{d}_{\text{obs}}$ gerado com operador DF16

Talvez fosse de se esperar que a maior precisão no $\mathbf{d}_{\text{cal}}$ gerasse melhores resultados; mas como se vê na Figura 5.13, prevalece o quão parecidos são os operadores. Conclui-se então que operadores mais parecidos para $\mathbf{d}_{\text{obs}}$ e $\mathbf{d}_{\text{cal}}$ geram uma melhor inversão, mas operadores diferentes também não a inviabilizam, gerando resultados similares. A inversão usando operadores diferentes (Figura 5.13(b)) é considerada boa, visto que o Erro RMS para o modelo invertido final é menor que o erro RMS para o modelo suavizado inicial (Figura 5.13(c)).

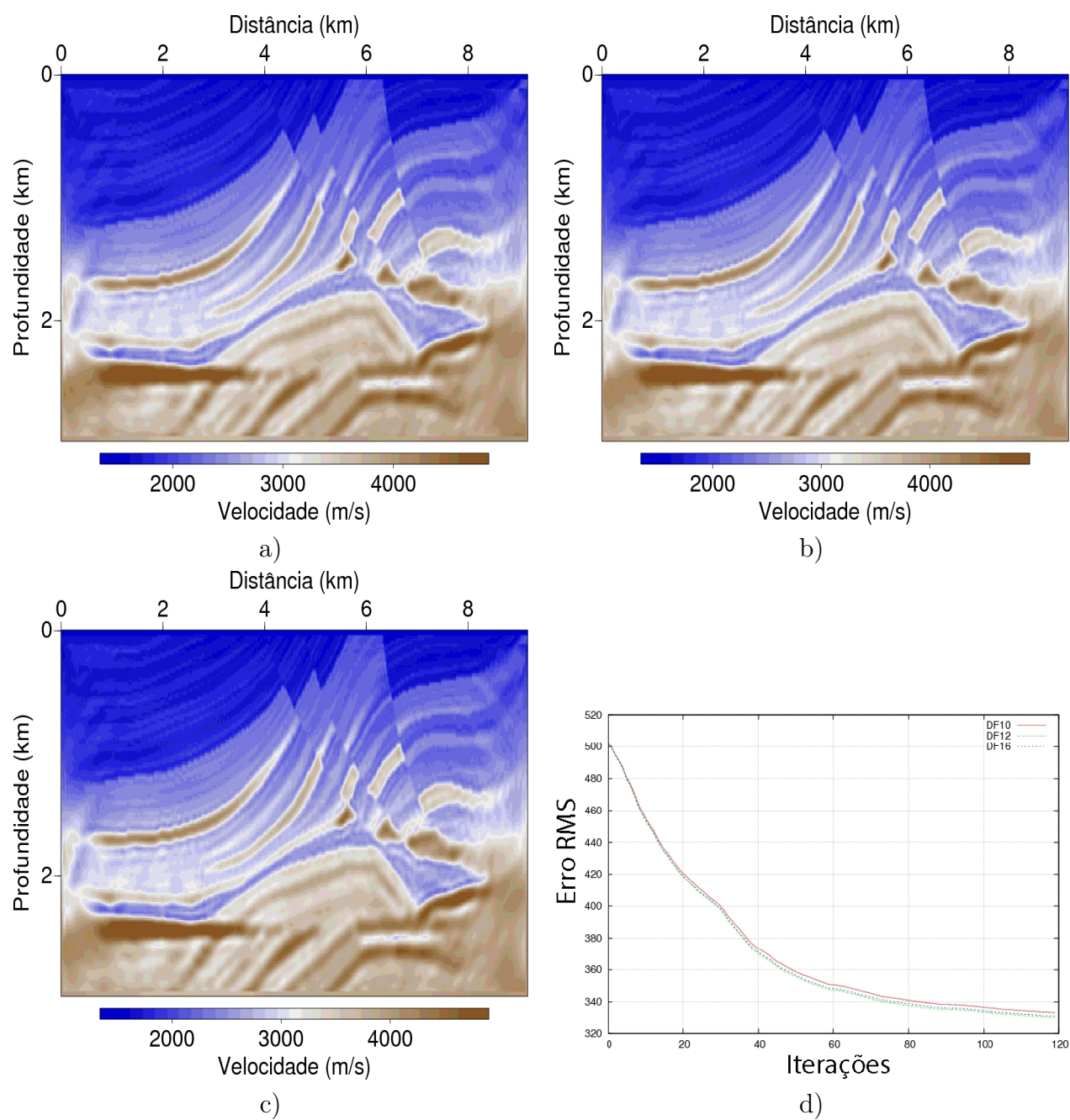


Figura 5.12: FWI multiescala, 4 rodadas de 30 iterações cada, L-BFGS, para diferentes operadores de DF. a) DF de ordem 10. b) DF de ordem 12. c) DF de ordem 16. d) Erro RMS para cada um dos operadores.

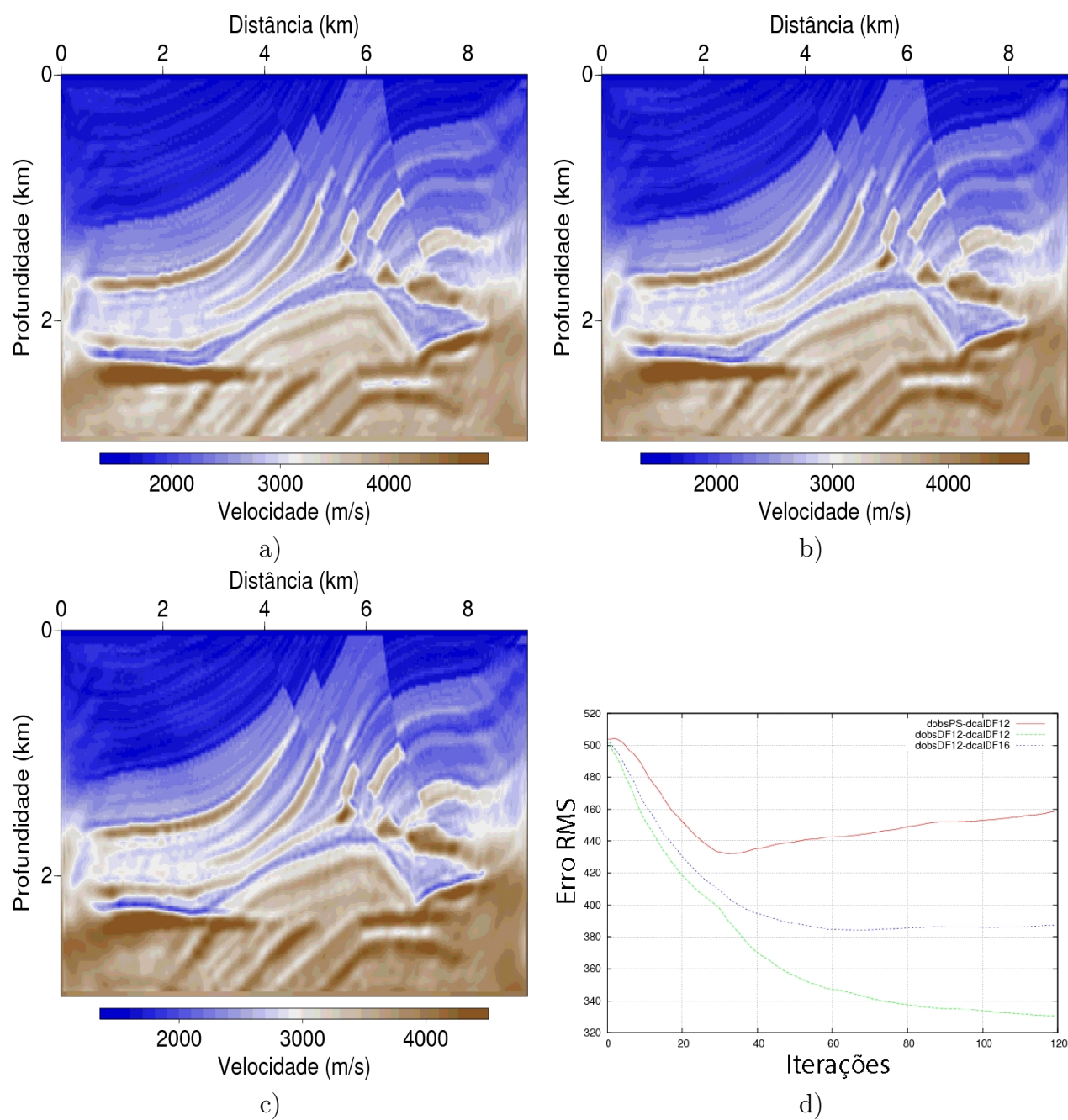


Figura 5.13: Diferenças na inversão decorrentes do uso de diferentes operadores para $\mathbf{d}_{obs}$ e $\mathbf{d}_{cal}$. a) $\mathbf{d}_{obs}$ com operador DF12, $\mathbf{d}_{cal}$ com DF12. b) $\mathbf{d}_{obs}$ com operador DF12, $\mathbf{d}_{cal}$ com operador DF16. c) $\mathbf{d}_{obs}$ com operador PS, $\mathbf{d}_{cal}$ com operador DF12. d) Erro RMS para os resultados em (a), (b), (c).

5.5.2 Influência da borda absorvedora

Para entender a influência da borda absorvedora em $\mathbf{d}_{\text{obs}}$ e $\mathbf{d}_{\text{cal}}$, e depois na FWI, iniciamos a análise verificando os dados para o arranjo previamente citado de 28 tiros, que cobrem todo o modelo de velocidades Marmousi-Teste, modelados sem borda na Figura 5.14(a). Os eventos mais marcantes decorrentes de reflexão nas bordas são elucidados na Figura 5.14(b): pode-se perceber que há uma forte reflexão na borda superior, presente em todos os tiros; reflexões no lado esquerdo do modelo nos tiros iniciais; e reflexões no lado direito do modelo nos tiros finais.

Uma forma eficiente de gerar uma "borda infinita", que é a situação de uma aquisição real, é modelar com uma borda duas vezes maior que a dimensão vertical do modelo (no caso, $n_z = 125$, e a borda teria tamanho $n_b = 250$ amostras) de forma que o sismograma registre até os eventos decorrentes dos refletores mais profundos, mas não dê tempo das reflexões de borda serem registradas. O resultado é um sismograma sem eventos de borda, mas o tempo computacional e a memória utilizada para gerá-lo, nesse caso, é cerca de 6 vezes maior. Portanto, nem sempre é possível gerar sismogramas sem eventos de borda, pois como mostrado anteriormente, a memória da GPU usada é de 5,2 GB.

Na Figura 5.15 são mostrados os eventos de borda para uma modelagens com borda "infinita" (Figura 5.15(a)), sem borda (Figura 5.15(b)), com *taper* (Figura 5.15(c)) de borda tamanho 32 amostras e com PML (Figura 5.15(d)) de borda tamanho 32. A Figura 5.15(b) é utilizada como base para reconhecer os eventos nas Figuras 5.15(c) e 5.15(d). O tamanho 32 foi escolhido arbitrariamente, por proporcionar baixo custo computacional e de memória, e uma atenuação razoável dos eventos de borda.

No primeiro teste, verifica-se a influência da borda absorvedora em $\mathbf{d}_{\text{obs}}$ e $\mathbf{d}_{\text{cal}}$. O intuito é ver se, dado $\mathbf{d}_{\text{obs}}$ gerado com borda de tamanho 32, e $\mathbf{d}_{\text{cal}}$ também gerado com borda de tamanho 32, qual resulta na melhor inversão: *taper* ou PML. Além disso são verificados os resultados para um $\mathbf{d}_{\text{obs}}$ com PML e um $\mathbf{d}_{\text{cal}}$ com *taper*. Realizou-se então a FWI multiescala (120 iterações) para:

1. $\mathbf{d}_{\text{obs}}$ e $\mathbf{d}_{\text{cal}}$ com *taper*,
2. $\mathbf{d}_{\text{obs}}$ e $\mathbf{d}_{\text{cal}}$ com PML,
3. $\mathbf{d}_{\text{obs}}$ com PML e $\mathbf{d}_{\text{cal}}$ com *taper*.

Os resultados são mostrados na Figura 5.16. Como se pode observar nas Figuras 5.16 (a), (b), e (c), os resultados são bastante similares. A análise do erro RMS (Figura 5.16(b)) de

cada inversão mostra com mais detalhe que a inversão PML-PML é melhor nas primeiras 30 iterações, que corresponde à rodada multiescala de menor frequência; porém nas rodadas seguintes onde são usadas frequências maiores a performance *taper-taper* é melhor, já que apresentou o menor erro RMS total ao final das iterações. Como esperado, o resultado PML-*taper* é o pior de todos, mas ainda assim gera uma inversão razoável (Figura 5.16(c)), apesar do reservatório no anticlinal do modelo, a $z = 2,5$ km e $x = 6,5$ km, não ser muito bem imageado. Em conclusão, esse teste mostra que os eventos de borda, e a condição de borda que os caracteriza em $\mathbf{d}_{\text{obs}}$ e $\mathbf{d}_{\text{cal}}$, apresentam melhores resultados se forem gerados igualmente; e que o *taper* em ambos os dados gerou os melhores resultados (análise geral do erro RMS total da Figura 5.16(b)).

Por fim, de forma a avaliar a inversão com um $\mathbf{d}_{\text{obs}}$ mais próximo do real, onde não há eventos de borda, é testada a inversão utilizando o $\mathbf{d}_{\text{obs}}$ da Figura 5.15(a) com borda "infinita" (borda de tamanho 256, que tem um tempo de modelagem 6 vezes maior), e a modelagem usual para $\mathbf{d}_{\text{cal}}$, com borda de tamanho 32, usando *taper* no primeiro teste e PML no segundo. Os resultados são mostrados na Figura 5.17(a) para o $\mathbf{d}_{\text{cal}}$ com *taper* e na Figura 5.17(b) para o $\mathbf{d}_{\text{cal}}$ com PML. A inversão de $\mathbf{d}_{\text{cal}}$ com PML apresentou melhores resultados que o *taper*, mas nenhum dos testes imageou corretamente o reservatório no anticlinal a $z = 2,5$ km e $x = 6,5$ km. Conclui-se então que os eventos de borda influenciam o processo de inversão, visto que afetam o resíduo entre $\mathbf{d}_{\text{obs}}$ e $\mathbf{d}_{\text{cal}}$, mas não ao ponto de inviabilizá-lo. Todavia, a inversão é considerada ruim, visto que o erro RMS total para o modelo invertido final é maior que o erro RMS para o modelo suavizado inicial (Figura 5.17(c)).

Nessa implementação em GPU, a duração de 120 iterações da FWI multiescala para um $\mathbf{d}_{\text{cal}}$ modelado com borda 256 seria de aproximadamente $3h49min$, usando os 28 tiros do modelo Marmousi-Teste. Para um dado observado de 280 tiros, que cobre todo o modelo Marmousi-Teste, o tempo seria de aproximadamente $33h20min$; num modelo Marmousi maior, de tamanho 250×767 , cobertura completa de 650 tiros (para esse arranjo *split-spread* de 101 geofones) o tempo seria de aproximadamente 4,5 dias. Dessa forma, devido ao alto custo computacional em gerar resultados com "borda infinita", nos testes da seção "Resultados", $\mathbf{d}_{\text{obs}}$ e $\mathbf{d}_{\text{cal}}$ foram modelados com as mesmas condições de borda, e com borda tamanho 32. Uma sugestão de trabalho futuro seria a análise da FWI para um $\mathbf{d}_{\text{obs}}$ sem eventos de borda, e como gerar um $\mathbf{d}_{\text{cal}}$ sem esses eventos a um custo computacional viável.

Uma observação final sobre $\mathbf{d}_{\text{obs}}$ sintético é que num dado observado real, ocorre reflexão na camada superior (o ar). Por isso há argumentos de que não deveria haver borda absorvedora na parte superior do modelo (Yang et al., 2007, Yang et al., 2015). A reflexão na borda

superior muda a forma da *wavelet* e gera múltiplas indesejadas, dificultando a FWI. Dessa forma, nesse trabalho utilizou-se borda superior absorvedora, sob a suposição de que, numa situação real, o $\mathbf{d}_{\text{obs}}$ de entrada para a FWI já estaria tratado das múltiplas e das alterações na forma da *wavelet*.

De todos esses testes, para diferentes operadores de modelagem e condições de borda, a conclusão é que quanto mais próximos $\mathbf{d}_{\text{obs}}$ e $\mathbf{d}_{\text{cal}}$ no operador de modelagem e na condição de borda, melhor o resultado da FWI, mas que operadores e condições de borda diferentes não inviabilizam a inversão; que para um $\mathbf{d}_{\text{obs}}$ gerado com borda, a PML tem melhor performance para dados de menor frequência, mas o *taper* dá o melhor resultado para todas as bandas; e que para um $\mathbf{d}_{\text{obs}}$ sem eventos de borda (mais próximo de um dado real) a PML dá melhores resultados.

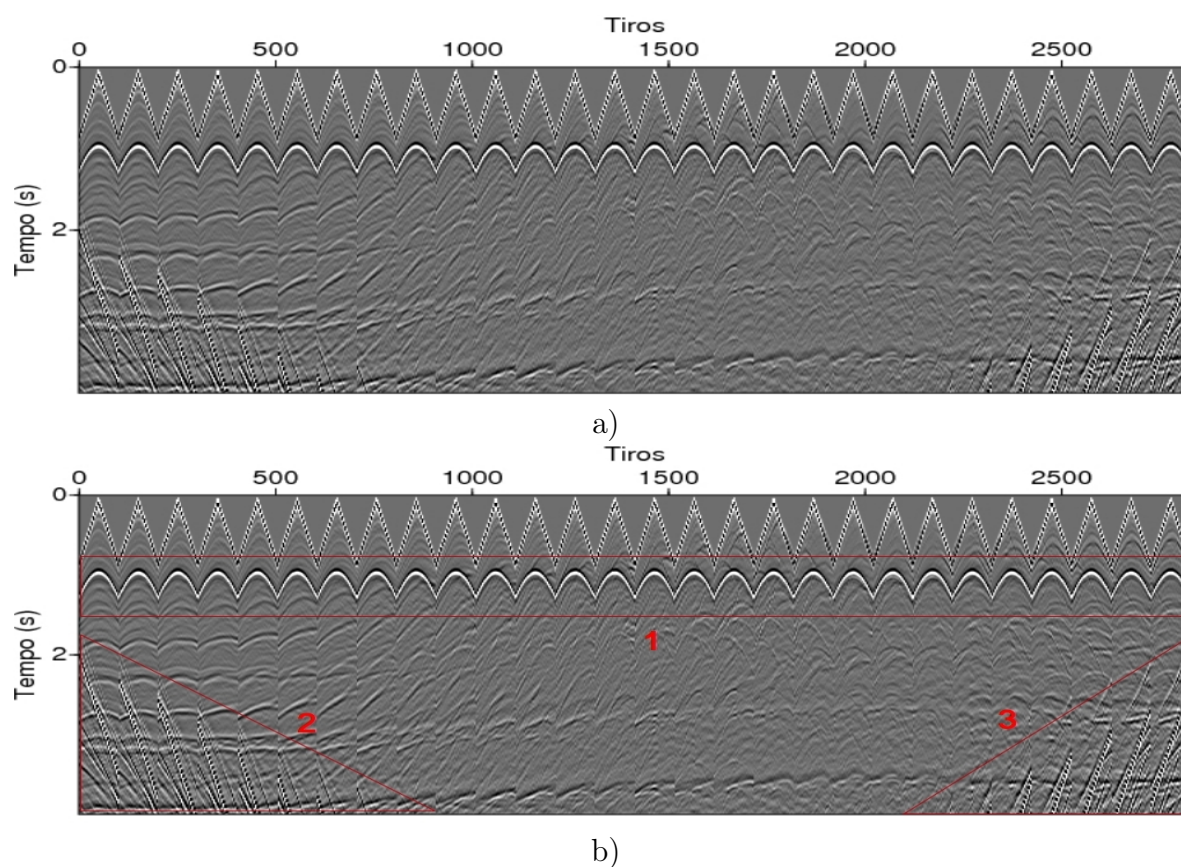


Figura 5.14: a) 28 tiros modelados sem borda. b) Eventos de borda destacados: reflexão na borda superior (1); reflexões na borda da esquerda (2); reflexões na borda da direita (3).

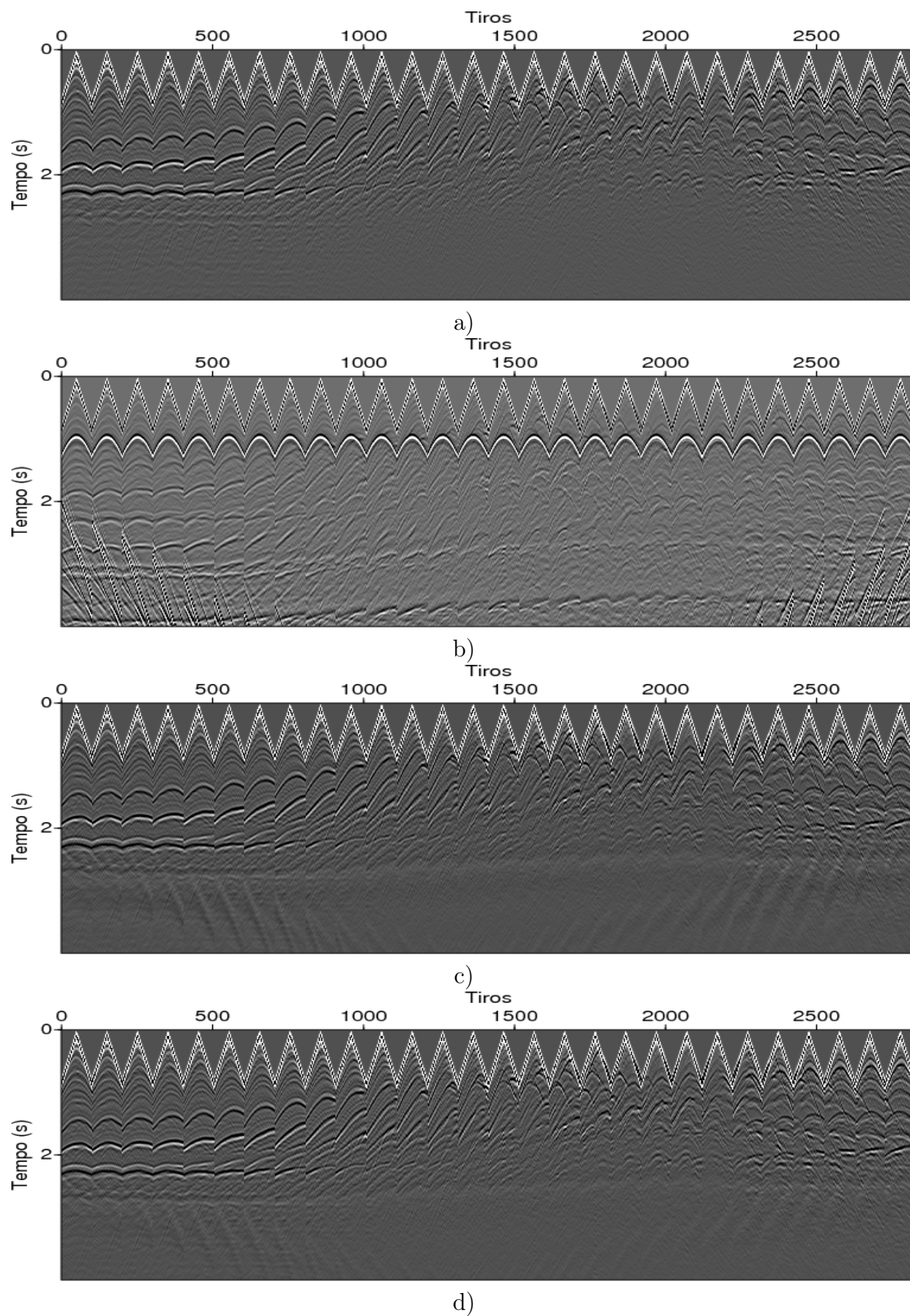


Figura 5.15: a) Tiros com borda de tamanho 256, sem eventos de borda. b) Tiros sem borda absorvedora. c) Tiros com *taper*. d) Tiros com PML.

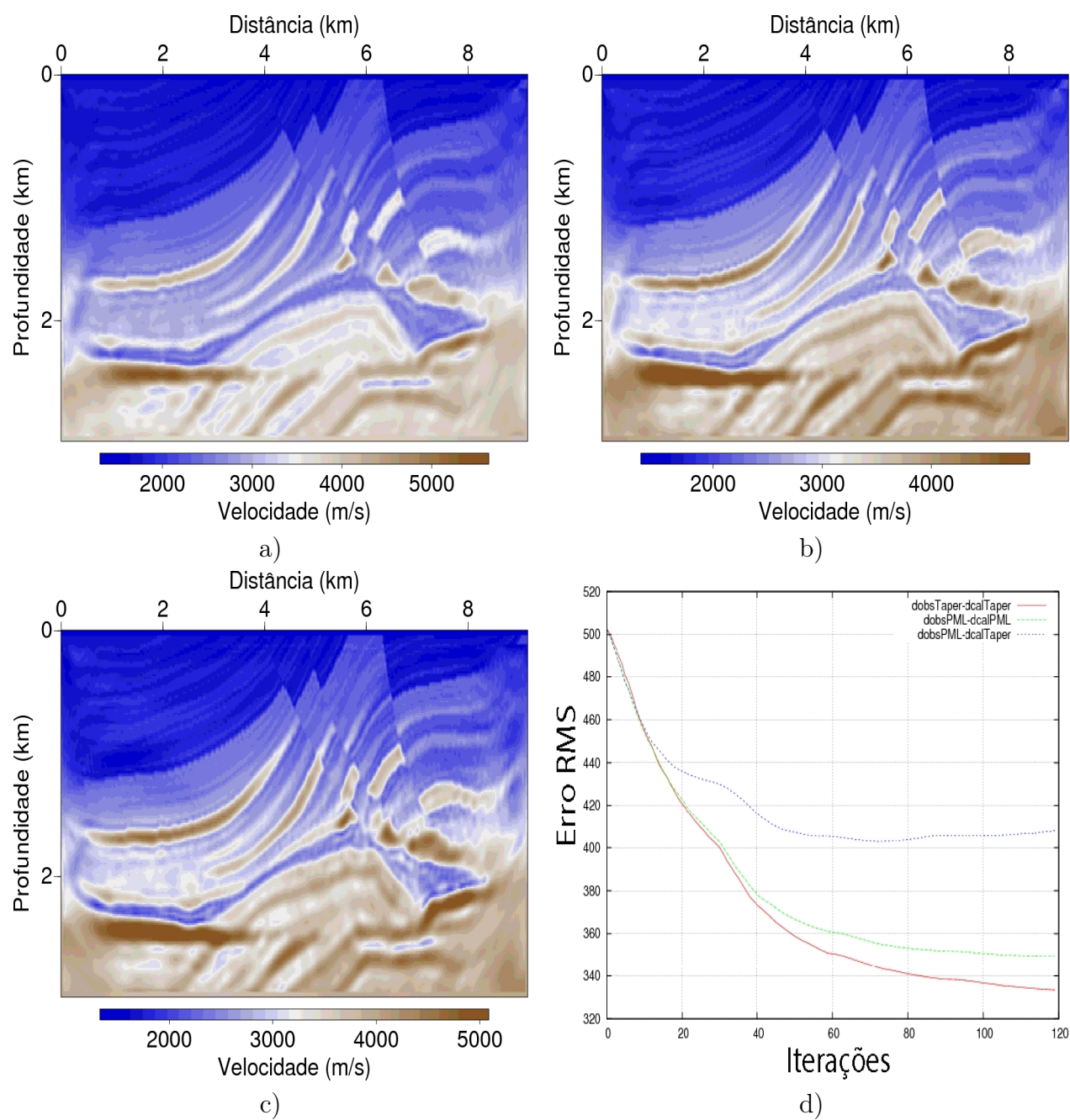


Figura 5.16: Resultados da FWI multiescala, borda absorvedora de tamanho 32. a) Modelo invertido para d_{obs} com *taper*, d_{cal} com *taper*. b) Modelo invertido para d_{obs} com PML, d_{cal} com PML. c) Modelo invertido para d_{obs} com PML, d_{cal} com *taper*. d) Erros RMS para os resultados (a), (b) e (c).

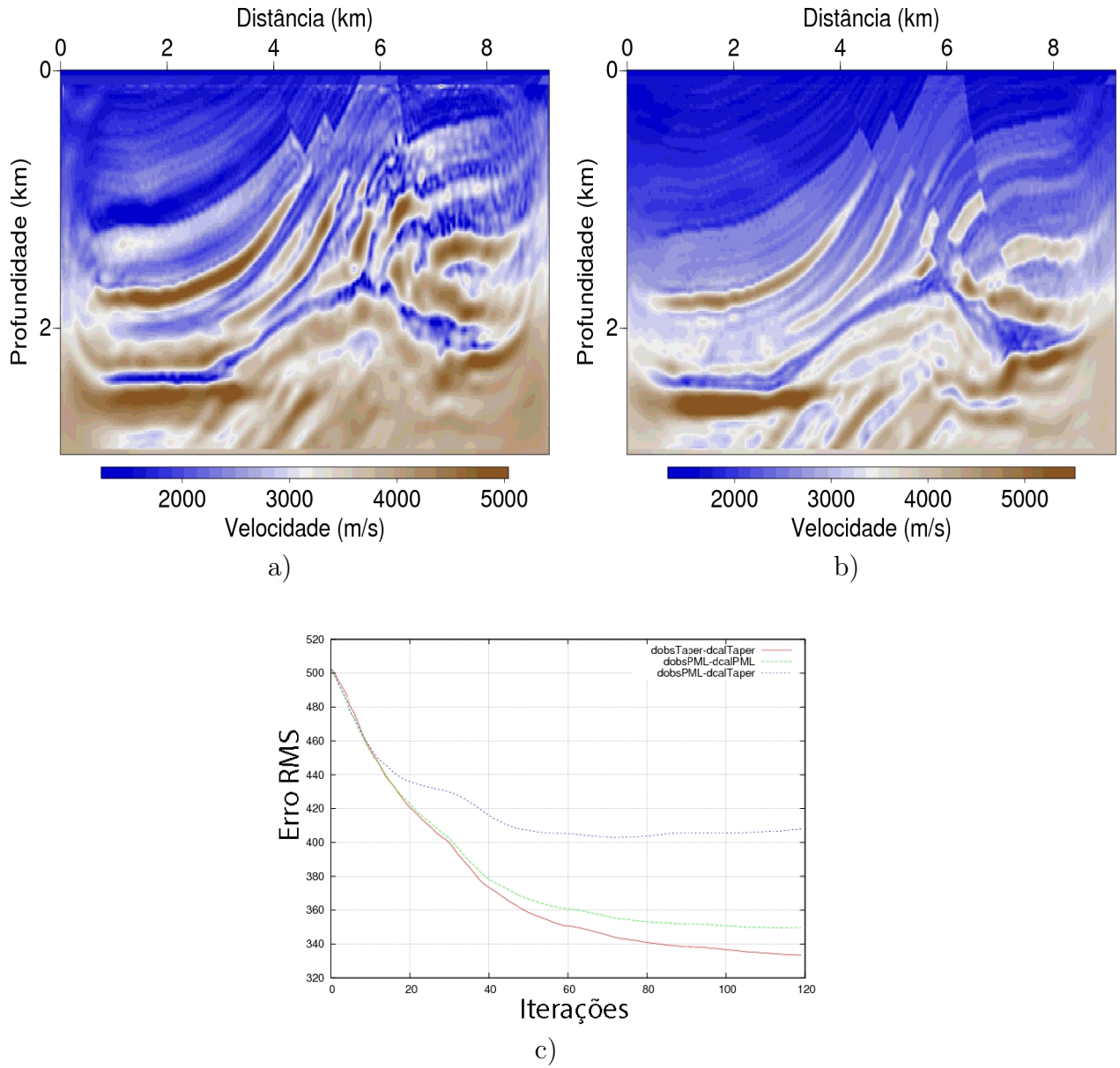


Figura 5.17: Resultados da FWI para um $\mathbf{d}_{\text{obs}}$ com "borda infinita". a) Modelo invertido para $\mathbf{d}_{\text{cal}}$ com *taper*. b) Modelo invertido para $\mathbf{d}_{\text{cal}}$ com PML. c) Erros RMS para os resultados (a) e (b).

5.6 Resultados

Nesta seção são mostrados resultados para diversos modelos de velocidades diferentes, utilizando os métodos de modelagem e inversão mais adequados a cada caso, com base em toda a discussão realizada nos capítulos precedentes. Além disso, o tempo de computação (em GPU) de cada resultado é comparado ao tempo numa implementação em série. Visto que praticamente todo o tempo computacional da FWI é dado pelo tempo das modelagens sísmicas envolvidas, a base de comparação entre implementação em GPU e implementação

em série foi o tempo da modelagem de 1 tiro. Sabendo que cada iteração da FWI (como discutido no capítulos 1, 2 e 3 realiza 4 modelagens (3 para o gradiente, 1 para o passo), o tempo de uma iteração da FWI pode ser estimado por:

$$t_{iter} = n_{shots} \times n_{mod} \times t_{1shot}, \quad (5.4)$$

$$t_{total} = n_{iter} \times t_{iter}. \quad (5.5)$$

onde n_{shots} é o número total de tiros, n_{mod} é o número de modelagens por iteração (nesta dissertação, ou 3 modelagens no caso de passo unitário, ou 4 modelagens no caso de passo variável), e t_{1shot} o tempo de modelagem de 1 tiro. Para obter o t_{total} , deve-se somar cada t_{iter} ; se o n_{mod} foi o mesmo em todas as iterações, basta multiplicar t_{iter} pelo número total de iterações (equação 5.5); se n_{mod} variar (por exemplo, passo variável, e passo unitário), deve-se calcular cada t_{iter} e somar os tempos adequadamente.

De forma a confirmar a validade da equação 5.4, mediu-se o tempo de modelagem t_{1shot} para os modelos Marmousi-Teste e *Overthrust* (70 tiros), e o tempo t_{iter} da FWI desses modelos. Foi utilizado o método L-BFGS com passo variável, então $n_{mod} = 4$ em cada iteração. O tempo medido é comparado ao t_{iter} estimado pela equação 5.4.

O resultado é mostrado na Tabela 5.9. Comparando t_{iter} estimado e medido, observa-se que o medido é 3% maior no modelo Marmousi-Teste, e 10% maior no modelo *Overthrust*. Essa diferença pode ser explicada pela maior transferência de memória CPU-GPU que ocorre na FWI, em relação à modelagem. Quanto maior o tempo das modelagens (o Marmousi-Teste tem apenas 28 tiros, enquanto o *Overthrust* tem 70), melhor a precisão da fórmula (10% para 3% nesse caso), pois a parte do tempo total que corresponde a processos fixos (transferências CPU-GPU, alocação inicial, inicializações) fica cada vez menos relevante para o tempo total, à medida que o número de modelagens cresce.

Mesmo não contabilizando essa diferença no tempo de transferência CPU-GPU, as equações 5.4 e 5.5 são boas estimadoras do tempo computacional. Desta forma, para comparar tempo em série e tempo em paralelo, basta comparar t_{1shot} de cada uma das abordagens, e utilizar a equação 5.4 para comparar os tempos estimados totais. Nas seções seguintes as comparações de tempo são realizadas dessa maneira.

Tabela 5.9: Tempos medidos e estimados pela equação 5.4 .

	t_{1shot} (GPU, medido)	t_{iter} (GPU, estimado)	t_{iter} (GPU, medido)
Marmousi 28 tiros	0,18 s	20,16 s	22,5 s
Overthrust 70 tiros	0,43 s	2 min 4 s	2min 8 s

5.6.1 Modelo Marmousi

O modelo Marmousi foi criado pelo *Institute Français du Pétrole* em 1988. As feições deste modelo são baseadas num perfil que corta a bacia de Cuanza e a parte Norte de Quenguela, em Angola. A geometria e modelo de velocidades foram criados de forma a produzir dados sísmicos complexos, que requerem técnicas avançadas de processamento para obter uma imagem correta da subsuperfície (Irons, 1988). O modelo de velocidades deve ter 9,225 km de extensão e 3 km de profundidade de forma a representar a geologia citada. Dessa maneira, o modelo de velocidades pode ter diferentes quantidades de amostras (nz, nx), desde que se compense nos intervalos de amostragem Δz e Δx .

O maior modelo Marmousi disponibilizado tem $nz = 751$ e $nx = 2301$ amostras, com $\Delta z = 4$ m e $\Delta x = 4$ m. Quanto maior o número de amostras do modelo de velocidades, melhor tende a ser a imagem da subsuperfície, mas também maior o custo computacional, visto que os campos de pressão, onde são propagadas as ondas (processo que responde por quase todo o custo computacional) deve ter as mesmas dimensões do modelo de velocidade. Assim, é comum que se reamostre o campo Marmousi 751×2301 para versões menores, como por exemplo o campo Marmousi-Teste 125×383 (6 vezes menor que o "original"), com $\Delta z = 24$ m e $\Delta x = 24$ m, utilizado em todos os capítulos anteriores, e o modelo Marmousi 250×787 (3 vezes menor que o "original"), com $\Delta z = 12$ m e $\Delta x = 12$ m, a ser utilizado nessa seção. A Figura 5.19 mostra essas 3 versões do modelo.

O modelo 125×383 (Figura 5.19-c) já foi extensamente testado no capítulo 3 e nesse capítulo. A FWI do modelo 751×2301 tem um alto tempo de processamento (51 min por iteração, 220 tiros), de forma que optou-se então por testar o modelo 250×767 , em um caso específico que exemplifica as vantagens do REM.

Marmousi 250×767

Supondo que nesta situação só se dispõe de um $\mathbf{d}_{obs}$ amostrado a 4 ms (amostragem típica da sísmica), com número de amostras $nt = 1000$, em 133 tiros espaçados de 5 amostras, e frequência máxima de aproximadamente 42 Hz. Deseja-se, então, obter o melhor campo de velocidades possível, partindo de um modelo inicial mostrado na Figura 5.20-a (que na

verdade é o Marmousi 250×767 suavizado duas vezes), e dos dados disponibilizados. Para modelar $\mathbf{d}_{\text{cal}}$ se escolheu o operador DF de ordem 16 (DF 16), que num teste visual não mostrou dispersão para uma $f_{\text{peak}} = 18$ Hz ($f_{\text{max}} \approx 41,6$ Hz). Já a condição de estabilidade (equação 1.22) obriga que $\Delta t < 0,001165$ s.

Do ponto de vista da dispersão numérica, o operador DF 16 é capaz de simular um $\mathbf{d}_{\text{cal}}$ de frequência máxima similar ao $\mathbf{d}_{\text{obs}}$, mas do ponto de vista da amostragem (estabilidade), não, pois não é possível gerar diretamente um dado amostrado a 0,004 s, sem que se altere as amostragens espaciais, devido à condição de estabilidade. Uma solução possível é, então, modelar $\mathbf{d}_{\text{cal}}$ a 0,001 s, com $nt = 4000$ amostras (de forma que o campo se propague por todos os 4s necessários), e interpolar $\mathbf{d}_{\text{obs}}$ a 0,001 s para calcular o resíduo e migrá-lo na obtenção do gradiente.

Outra solução, na qual não é necessário interpolar $\mathbf{d}_{\text{obs}}$, gerando um $\mathbf{d}_{\text{cal}}$ diretamente com a mesma amostragem de 0,004 s, é utilizando o operador REM-DF16. Como discutido na seção 5.3, ele permite uma extrapolação a grandes Δt , gerando economia de memória a um custo computacional na mesma ordem de grandeza. A Tabela 5.10 compara as duas abordagens para esse problema (REM e interpolação). Para a implementação em GPU foi utilizada a placa Tesla C0275, e na implementação em série utilizou-se um processador Intel Xeon E5620 (ambos do Cluster Águia CPGG-UFBA).

É muito importante observar que o número máximo de termos M do REM, a ser utilizado na modelagem dos $\mathbf{d}_{\text{cal}}$ de cada iteração da FWI, depende da velocidade máxima do modelo (equação 1.52). Durante a FWI, as velocidades de todo o modelo variam a cada iteração, e a velocidade máxima pode ultrapassar aquela utilizada para calcular inicialmente o número de termos M do REM. Se isso ocorrer, o método pode se tornar instável nas iterações subsequentes, gerando amplitudes infinitas e inviabilizando a inversão. Desta forma, deve-se saber de antemão a velocidade máxima que o modelo pode alcançar, e utilizá-la para determinar o número de termos M usado no REM.

Nesse exemplo, o modelo inicial (suavizado, Figura 5.20) tem $v_{\text{max}} = 4538$ m/s, que se utilizada para o cálculo dos termos do REM, resulta em $M = 9$. Ao executar a FWI com 9 termos, houve instabilidade na iteração 56, inviabilizando a continuidade da inversão. Ao utilizar a velocidade máxima do modelo verdadeiro, $v_{\text{max}} = 5500$ m/s, se obtém $M = 11$, e a inversão estável para todas as iterações. É razoável supor que, numa situação real, se tenham informações geológicas suficientes para estimar a velocidade máxima da região, justificando o uso da velocidade máxima do modelo verdadeiro para esse cálculo.

Para a FWI é utilizado o método L-BFGS com passo variável (4 modelagens por iteração), e 5 rodadas multiescala de 30 iterações cada (totalizando 150 iterações) divididas em:

de 0 – 8,3 Hz, de 0 – 16,6 Hz, de 0 – 25,0 Hz, de 0 – 33,3 Hz, de 0 – 41,6 Hz. O resultado da inversão é apresentado na Figura 5.20(b). Pode-se perceber que a inversão foi bem sucedida, recuperando as feições do modelo e o reservatório localizado na posição $z = 2,5$ km e $x = 6,6$ km. O perfil de velocidades na posição $x = 6,6$ km mostra os resultados com mais detalhe (Figura 5.18). O erro RMS total, ao longo das 150 iterações, é apresentado na Figura 5.20(c), e ilustra a convergência do processo.

Tabela 5.10: Tempos computacionais para FWI com DF16 e REM-DF16.

	t_{1shot} (GPU, medido)	t_{iter} (GPU, estimado)	t_{total} (GPU, estimado)	t_{total} (série, estimado)	Memória (FWI)
FWI com DF16 $\Delta t = 0,001s$ $nt = 4000$	1,18s	10min 28s	1d 02h 09min	117d 15h 48min	827,73 MB
FWI com REM-DF16 $\Delta t = 0,004s$ $nt = 1000$	2,72s	24min 07s	2d 12h 17min	312d 18h 31min	325,45 MB

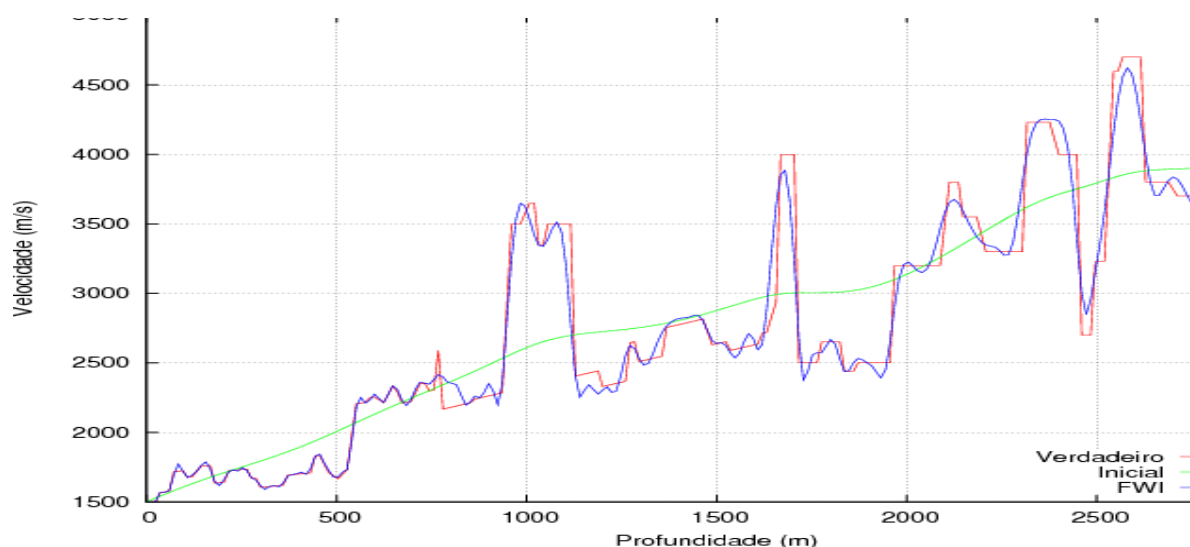


Figura 5.18: Perfis de velocidades na posição $x = 6,6km$: verdadeiro, inicial e invertido.

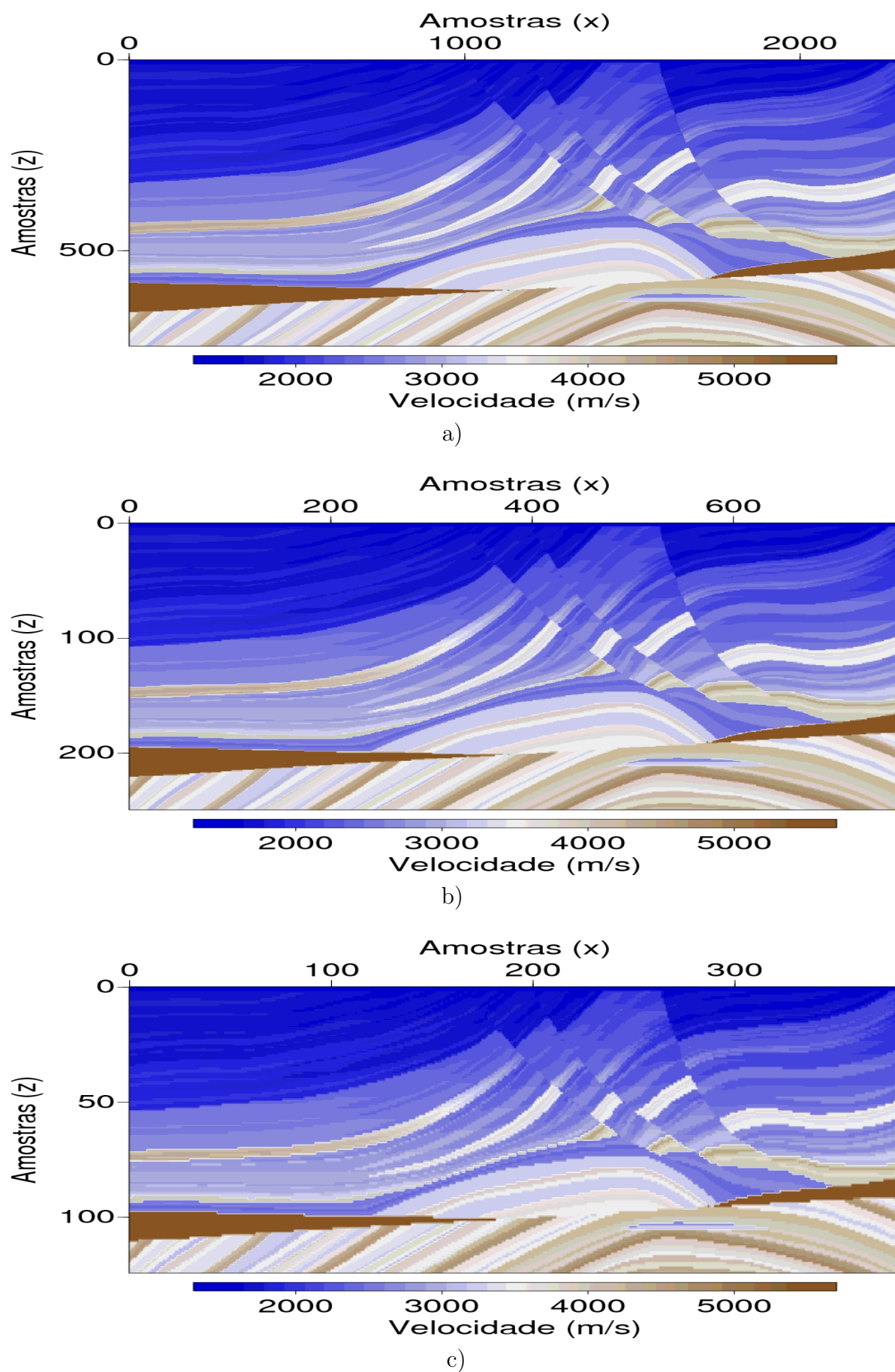


Figura 5.19: Modelos Marmousi. a) 751×2301 amostras. a) 250×767 amostras.
a) 125×383 amostras.

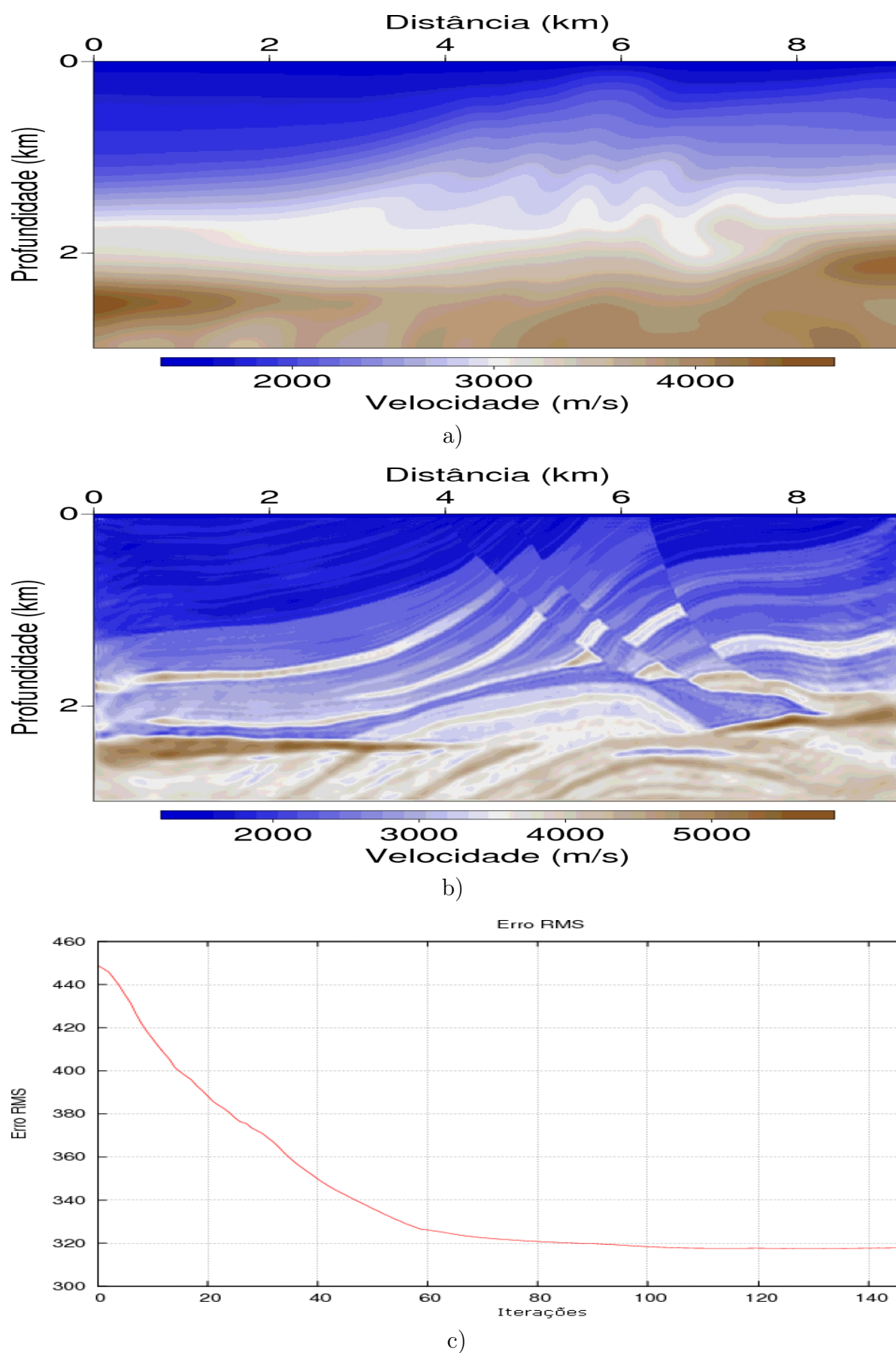


Figura 5.20: a) Modelo Marmousi 250×767 inicial (entrada da FWI). b) Resultado da FWI utilizando operador REM-DF16. c) Erro RMS total ao longo das iterações.

5.6.2 Modelo *Overthrust*

Nesta seção são mostrados resultados para o modelo da Figura 5.21(a), conhecido como SEG-EAGE *Overthrust* 2-D. Esse modelo tem dimensões $nz = 187$ e $nx = 801$ amostras, espaçamentos $\Delta z = 25$ m, $\Delta x = 25$ m, totalizando 4,675 km de profundidade e 20,025 km de extensão. O ponto de partida para a FWI foi esse mesmo modelo suavizado duas vezes pela função *smooth2* do pacote *Seismic Unix*, com suavização máxima, mostrado na Figura 5.21(b).

Os dados observado e calculado tem uma geometria *split-spread* fixa de 101 geofones, cuja fonte é uma *wavelet* Ricker de frequência de pico 15 Hz (totalizando uma frequência máxima de aproximadamente 34,7 Hz). O operador de modelagem utilizado foi DF de ordem 10, com $\Delta t = 0,002$ s, $nt = 2000$, em cada tiro. Na inversão foi utilizado método L-BFGS com passo variável dado pela equação 5.3, e 4 rodadas multiescala: de 0–8,6 Hz, de 0–17,3 Hz, de 0–26,0 Hz, de 0–34,7 Hz.

Influência da cobertura de tiros na inversão

Nessa seção se aproveitou para analisar a influência da cobertura de tiros na inversão. A primeira inversão foi feita com 70 tiros, espaçados de 10 amostras (250 m, já que $\Delta x = 25$ m); a segunda com 140 tiros espaçados de 5 amostras (125 m); e a terceira com 350 tiros, espaçados de 2 amostras (50 m). O espaçamento entre os 101 geofones de cada tiro é fixo em 1 amostra (25 m). Os resultados são mostrados na Figura 5.23 a, b, e c. Percebe-se a diferença visual nas inversões entre os resultados para 70 e 140, e 70 e 350 tiros. Entre as inversões para 140 e 350 tiros há pouca diferença.

Essa similaridade é quantificada pela Figura 5.22(c), que mostra o Erro RMS para cada uma das 3 inversões; observa-se que entre a de 140 e a de 350 tiros, o erro RMS é praticamente o mesmo. Isso é um indicativo de que uma cobertura de tiros maior que 140, para esse arranjo de geofones e para esse modelo, é redundante. Os perfis mostrados nas Figuras 5.22(a,b) (posição 14 km do modelo) também confirmam esse argumento.

O tempo computacional total de cada inversão é mostrado na Tabela 5.11, comparado ao tempo total estimado em série, com base na equação 5.4. Para a implementação em GPU foi utilizada a placa Tesla C0275, e na implementação em série utilizou-se um processador Intel Xeon E5620 (ambos do Cluster Águia CPGG-UFBA).

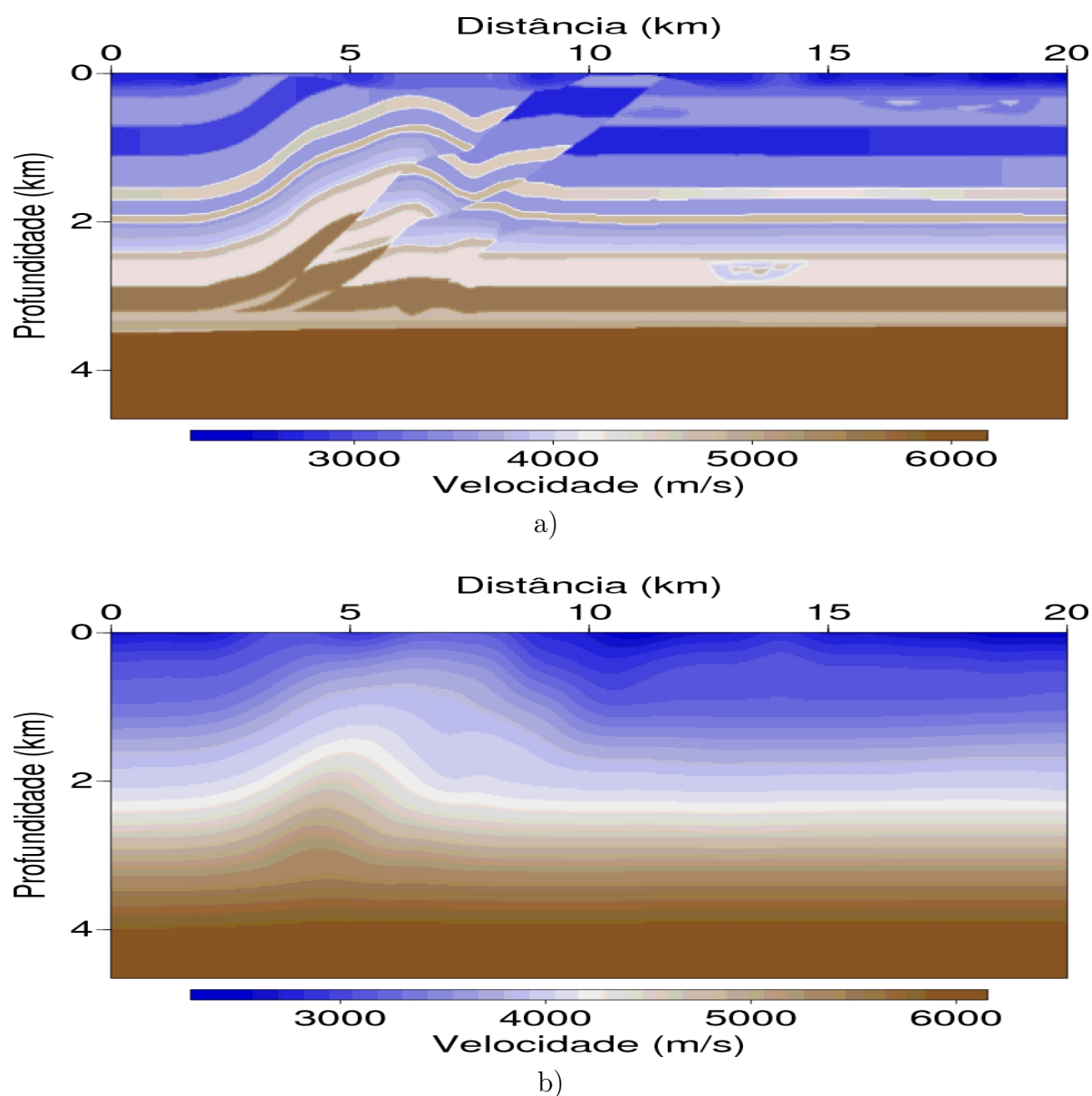
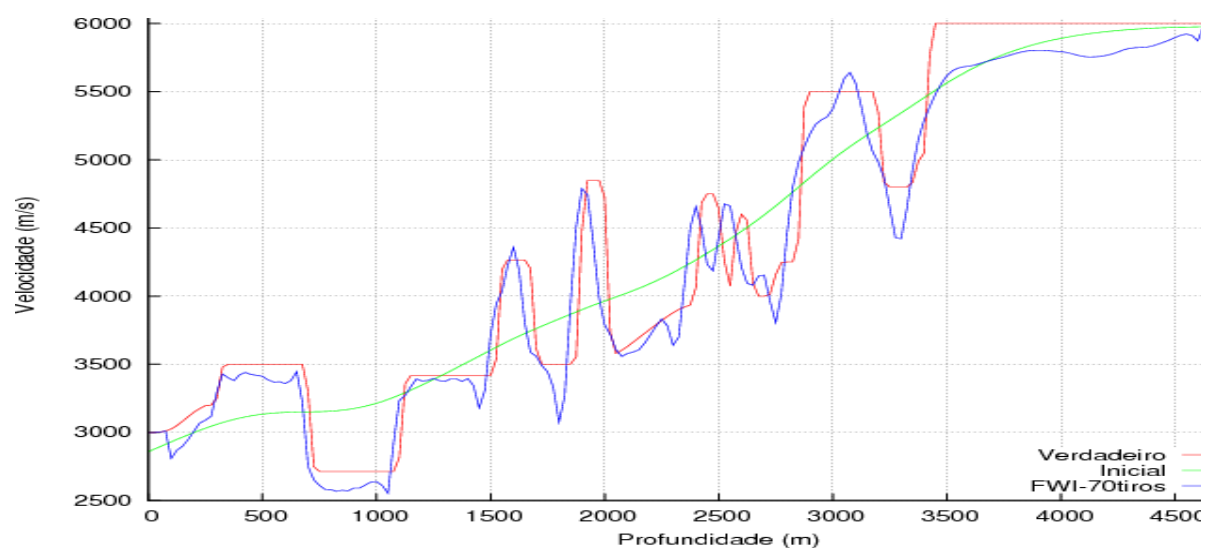


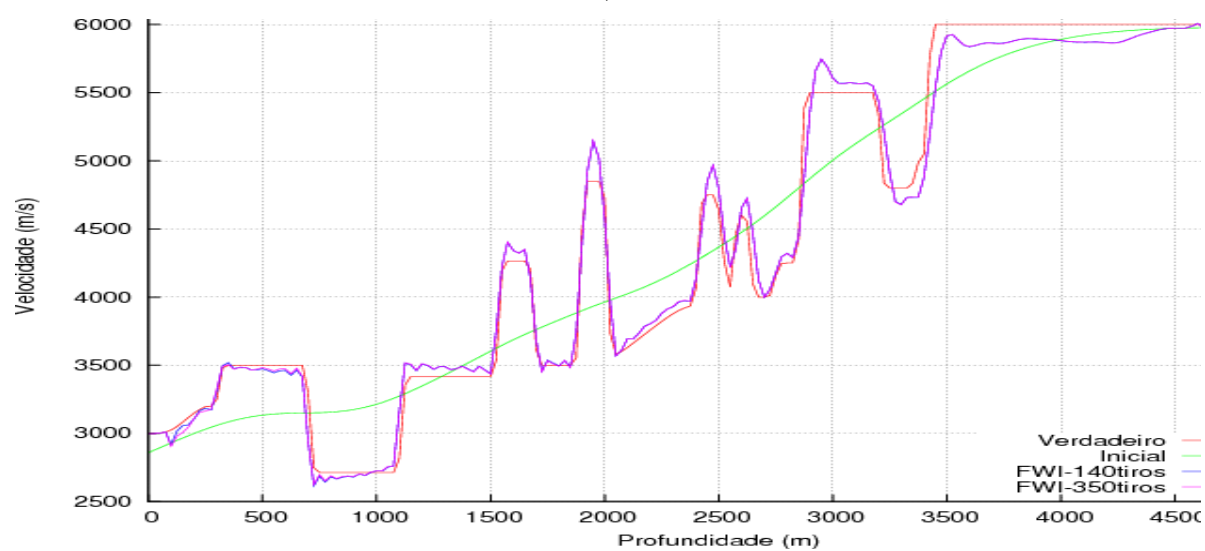
Figura 5.21: Modelo *Overthrust* 2-D. a) Modelo original. b) Modelo suavizado, campo inicial da FWI.

Tabela 5.11: Tempos de processamento, FWI do modelo *Overthrust*, para diferentes quantidades de tiros. Os tempos em série foram calculados por 5.4, a partir do tempo de 1 tiro.

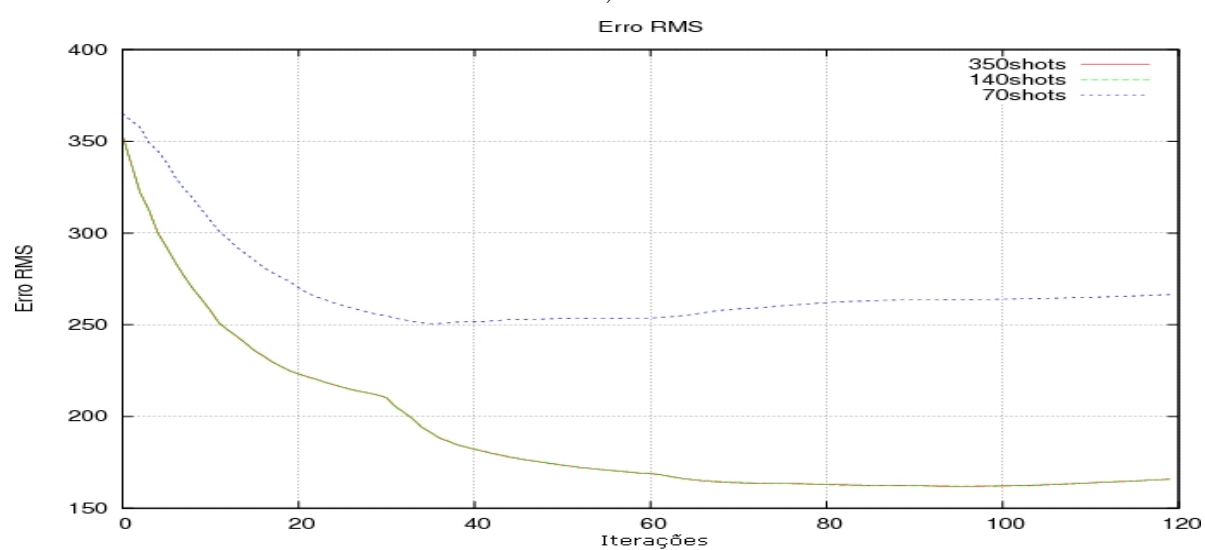
	t_{iter} (GPU, estimado)	t_{total} (GPU, medido)	t_{total} (série, estimado)	Memória (FWI)
<i>Overthrust</i> 70 tiros	2 min 08 s	4 h 17 min 03 s	14 d 1 h 10 min	340,58 MB
<i>Overthrust</i> 140 tiros	4 min 15 s	8 h 30 min 24 s	28 d 2 h 20 min	394,58 MB
<i>Overthrust</i> 350 tiros	10 min 37 s	21 h 13 min 30 s	70 d 5 h 50 min	557,34 MB



a)



b)



c)

Figura 5.22: a) Perfil na posição 14 km do modelo, para FWI de 70 tiros. b) Perfil para FWI de 140 e 350 tiros. c) Erro RMS em cada iteração da FWI, para 70, 140 e 350 tiros.

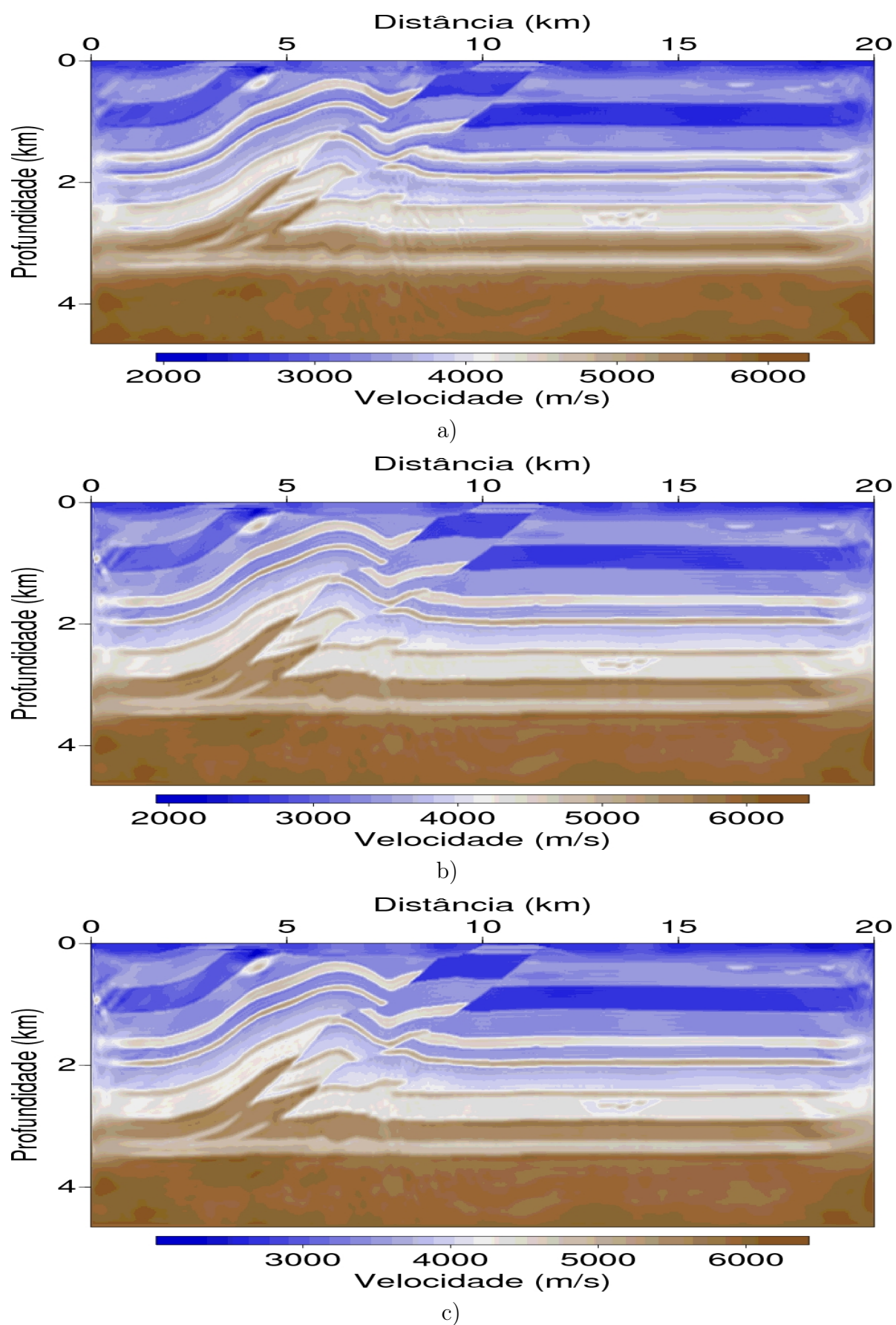


Figura 5.23: FWIs do modelo *Overthrust*, para: a) 70 tiros, b) 140 tiros e c) 350 tiros.

5.6.3 Modelo Falhas

Por fim é apresentada a FWI num modelo geológico bastante complexo, aqui denominado "Modelo Falhas". O modelo possui falhas em diferentes direções, uma base bastante irregular e algumas intrusões. O campo de velocidades tem dimensões $n_z = 265$ e $n_x = 600$ amostras, espaçamentos $\Delta z = 20$ m, $\Delta x = 20$ m, ou seja, 5,3 km de profundidade e 12 km de extensão. O campo de velocidades é mostrado na Figura 5.24(a), e sua versão suavizada duas vezes com o *smooth2* do SU, campo de velocidades inicial da FWI, é mostrada na Figura 5.24(b).

De forma a obter a melhor resolução possível, foi usado o operador pseudo-espectral, que permite maiores frequências não dispersivas, ao gerar $\mathbf{d}_{\text{obs}}$ e $\mathbf{d}_{\text{cal}}$. Para o método PS, os espaçamentos de 20 m dão uma condição de estabilidade $\Delta t < 0,001819$ s e de dispersão $f_{\text{max}} < 52,5$ Hz (equações 1.22 e 1.33). Desta forma, as modelagens foram feitas com uma *wavelet* Ricker de $f_{\text{peak}} = 22$ Hz ($f_{\text{max}} \approx 50,82$ Hz), $\Delta t = 0,0018$ s, $nt = 2222$ amostras. Foram usados 99 tiros de 101 geofones *split-spread*, espaçados de 5 amostras (100 m), cobrindo todo o modelo. A condição de borda usada foi o *taper*.

Para o método de inversão foi utilizado o L-BFGS com passo variável em todas as iterações. Foram realizadas 5 rodadas do processo multiescala: de 0 – 10 Hz, de 0 – 20 Hz, de 0 – 30 Hz, de 0 – 40 Hz, de 0 – 51 Hz, cada rodada com 30 iterações. Na última rodada a função objetivo divergiu após 14 iterações, totalizando 134 iterações.

O resultado final da inversão é apresentado na Figura 5.24(b). A Figura 5.25(a) mostra um perfil de velocidades na posição 3 km comparando modelos verdadeiro, inicial e invertido. O erro RMS total do modelo ao longo das iterações é mostrado na Figura 5.25(b).

Uma análise visual do modelo final e do erro RMS mostra que a inversão foi bem sucedida. Todavia, o erro RMS cresceu nas duas últimas rodadas (iteraões 90 a 134), mesmo com a convergência da função objetivo na rodada 4 (iteraões 90-119, Figura 5.26(d)); e houve divergência apenas nas 4 últimas iteraões da última rodada (Figura 5.26(e)), o que não é motivo suficiente para justificar o crescimento do erro RMS (visto que foram apenas 4 iteraões de divergência, e o erro RMS apresentou crescimento nas 44 últimas iteraões). Um justificativa plausível é que pode ter ocorrido *cycle-skipping* nas altas frequências, mesmo com o processo multiescala.

Na Tabela 5.12 são mostrados os tempos de processamento e o ganho em relação a uma implementação em série, baseado na equação 5.4. Para a implementação em GPU foi utilizada a placa Tesla C0275, e na implementação em série utilizou-se um processador Intel Xeon E5620 (ambos do Cluster Águia CPGG-UFBA).

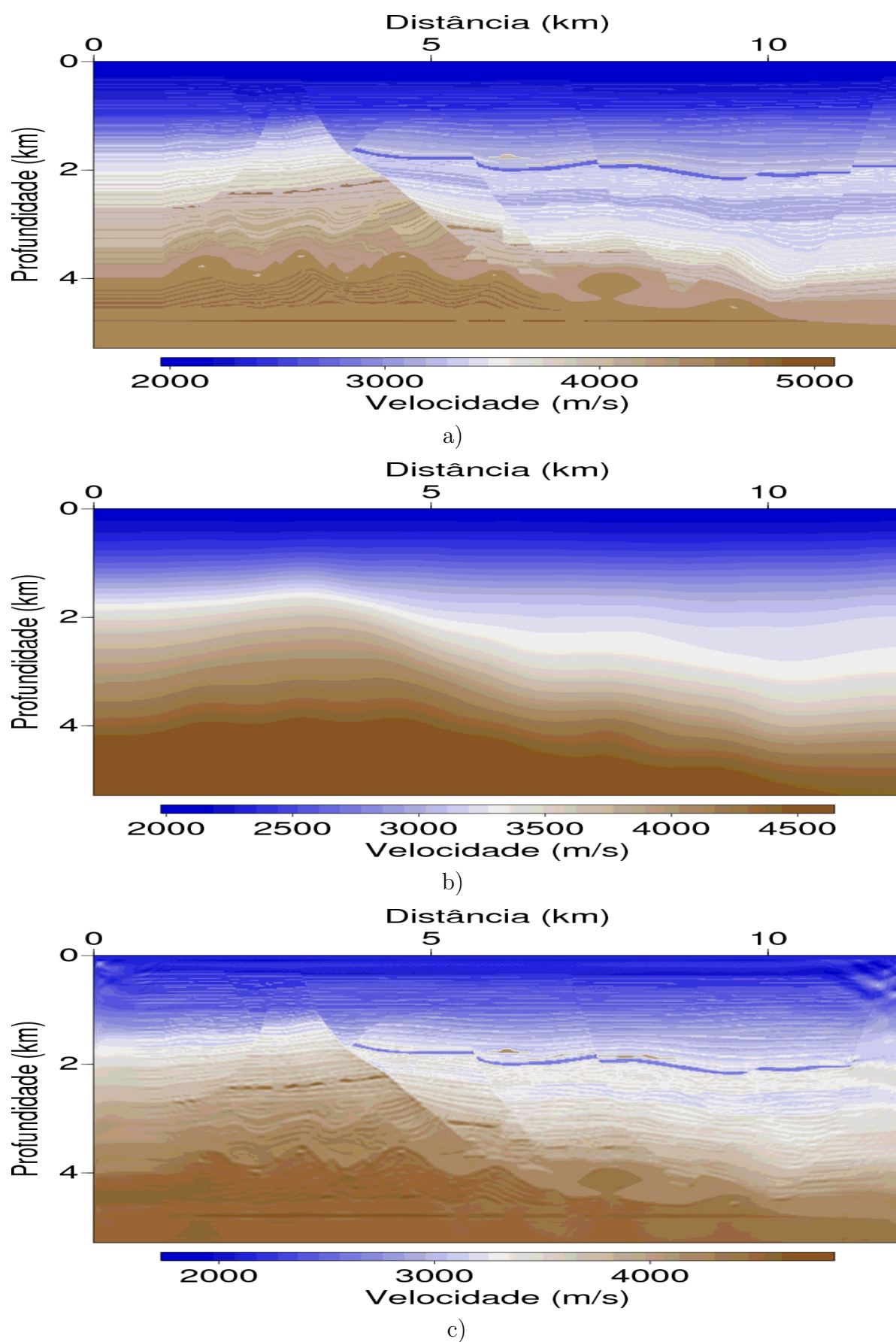
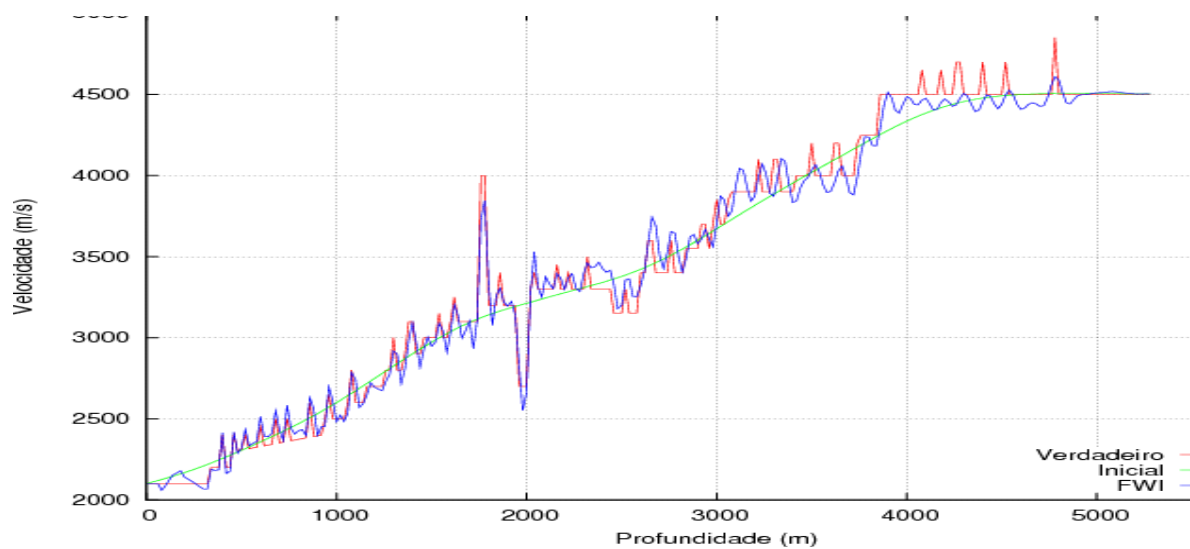


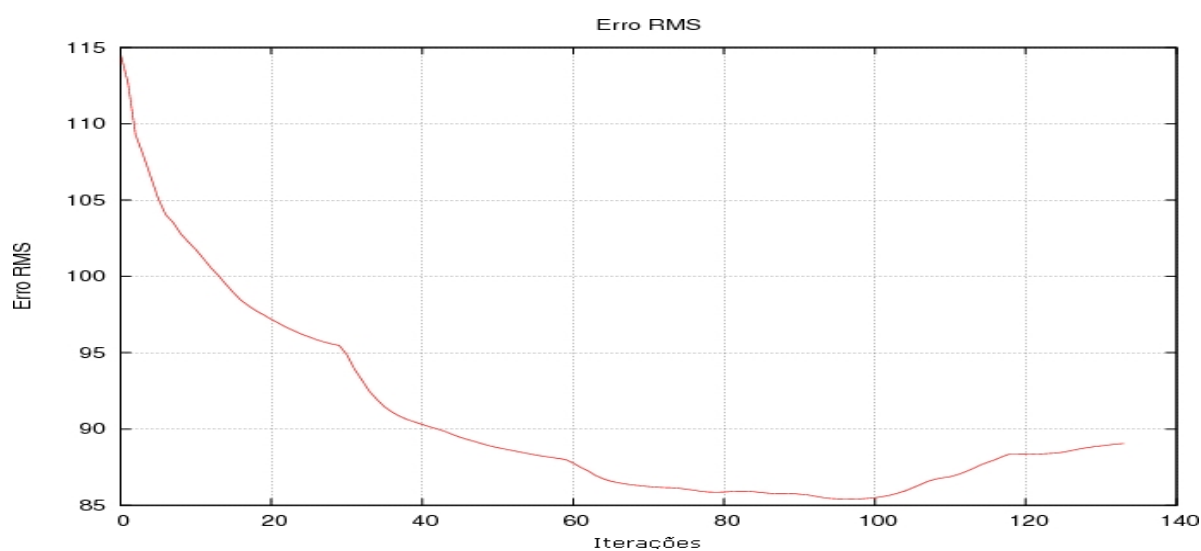
Figura 5.24: Modelo falhas. a) Modelo original. b) Modelo suavizado, campo inicial da FWI. c) Resultado final da FWI.

Tabela 5.12: Custos computacionais para a FWI do modelo Falhas.

	t_{1shot} (GPU, medido)	t_{iter} (GPU, estimado)	t_{total} (GPU, medido)	t_{total} (série, estimado)	Memória (FWI)
FWI - modelo Falhas	1,11s	7min 20s	16h 38min 58s	3d 17h 10min	372,08 MB



a)



b)

Figura 5.25: a) Perfil de velocidades na posição 3 km do modelo. b) Erro RMS ao longo das iterações.

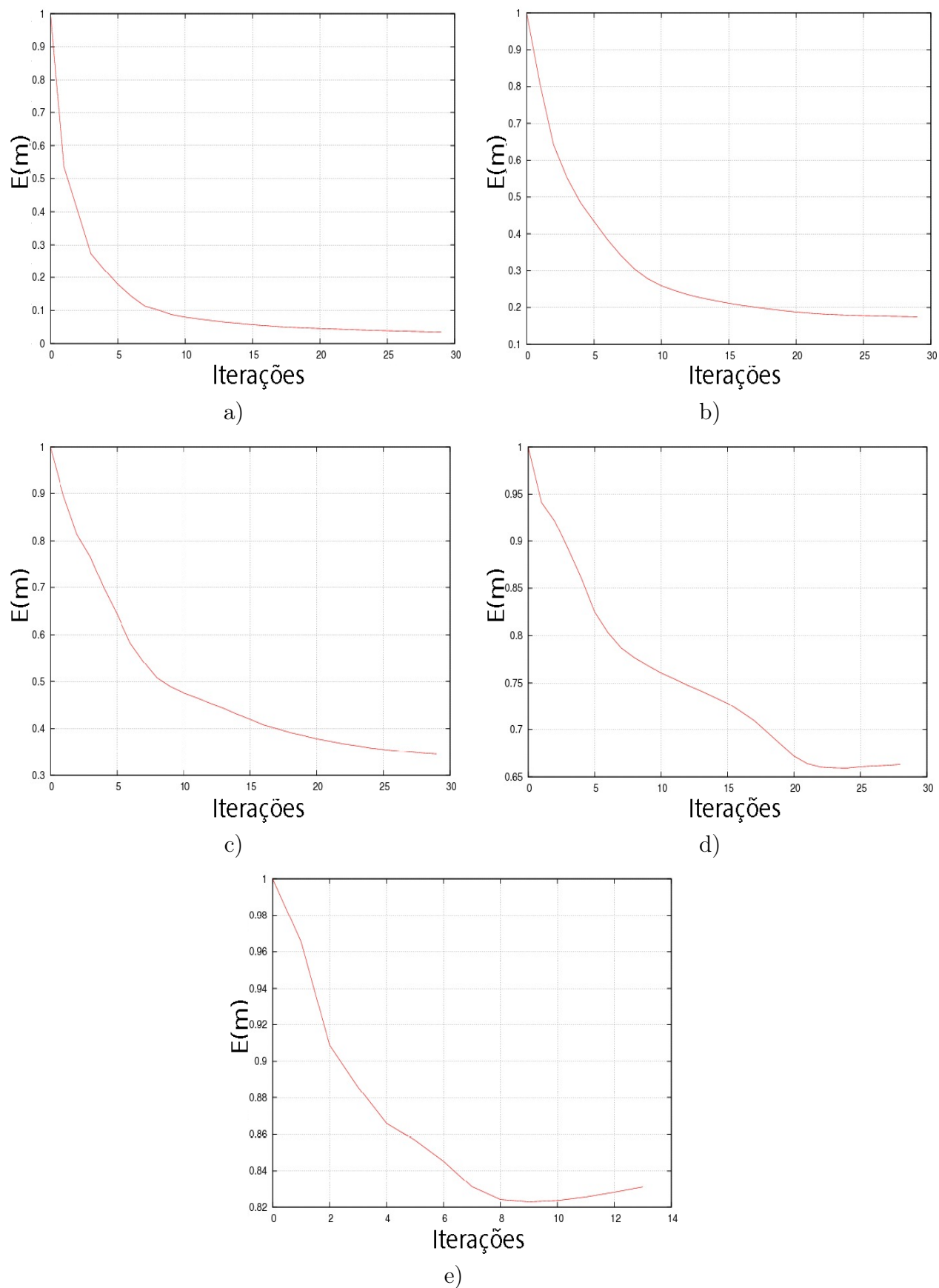


Figura 5.26: Funções objetivo normalizadas para cada uma das 5 rodadas da FWI (modelo Falhas). a) De 0 – 10, 12 Hz. b) De 0 – 20, 24 Hz. c) De 0 – 30, 36 Hz. d) De 0 – 40, 48 Hz. e) De 0 – 50, 82 Hz.

6

Conclusões

A implementação da FWI no domínio do tempo, nesta dissertação, seguiu a sequência modelagem sísmica - RTM - FWI. Como discutido anteriormente, do ponto de vista computacional, a RTM equivale a duas modelagens sísmicas, e com a técnica de borda efetiva (utilizada para economizar memória), a três modelagens sísmicas. O maior custo de cada iteração da FWI é o cálculo do gradiente, que pelo método adjunto, equivale à RTM do resíduo na iteração corrente. Desta forma a FWI custa por iteração três modelagens sísmicas, e mais uma modelagem necessária para calcular o passo α , totalizando 4 modelagens por iteração. Como discutido, a modelagem é muito intensiva computacionalmente, e a obtenção de resultados em série em tempo hábil é inviável, podendo demorar meses (vide Tabelas 5.9, 5.10, 5.11 e 5.12 da seção Resultados), de forma que é necessário algum método de paralelização dos cálculos. Nesta dissertação, se escolheu a GPU como método de paralelização.

Como mostrado no capítulo 3, a GPU tem memórias típicas de $2GB$ até $5GB$ (caso da placa Tesla C0275 do Cluster água CPGG-UFBA). Nesse sentido, nas implementações, se esteve sempre atento à memória utilizada; por este motivo se utilizou a borda efetiva na RTM, que economiza memória dispensando o armazenamento do do campo da fonte, de dimensões $(nz \times nx \times nt)$, pelo armazenamento da borda efetiva de tamanho $[2 \times (nz + nx) \times (nj - 1) \times nt]$ (sendo nj a ordem do operador), ao custo de uma modelagem a mais. Também por esse motivo se fez uso do operador REM, que permite uma extrapolação estável a passos temporais maiores, reduzindo nt , e conseqüentemente a memória da borda efetiva e do resíduo na FWI (de tamanho $ng \times ns \times nt$), gerando resultados de qualidade idêntica, a um custo de memória inferior (como mostram os resultados para o modelo Marmousi 250×767 , na Tabela 5.10 e Figura 5.20).

A implementação em GPU permitiu então testar em tempo hábil vários fatores de influência na FWI, em especial os operadores de modelagem, a condição de borda absorvedora e o método de inversão.

Quanto aos operadores de modelagem, concluiu-se que os operadores PS são os que permitem a maior frequência não dispersiva, mas são também mais lentos que os operadores DF. Os operadores REM tem maior custo computacional que seus "operadores base", mas menor custo de memória, e são boas soluções para problemas que exijam modelagem a grandes Δt 's.

Quanto às condições de borda absorvedora se confirmou que para dados observados sintéticos sem eventos de borda (mais próximos do real) a PML gerou resultados de inversão melhores que o *taper*. Para dados observados e calculados sintéticos com borda, os resultados de inversão são bons para quaisquer condições utilizadas, mas melhores se for usada a mesma condição nos dados observados e calculados. Devido ao fato do *taper* demandar menos processamento que a PML, ele foi utilizado na grande maioria dos testes da FWI, gerando modelos finais de alta resolução.

Quanto ao método de inversão, foram testados o conjugado gradiente não-linear (CGNL) e L-BFGS, com passo unitário e variável. Se confirmou a superioridade de resultados do L-BFGS em relação ao CGNL (Tabela 5.8 e Figura 5.11), e que o L-BFGS com passo unitário gera resultados similares (erro RMS ligeiramente superior, Figura 5.9) a um tempo computacional menor (uma modelagem a menos, Tabela 5.7).

A abordagem multiescala foi fundamental para evitar *cycle-skipping* e convergência para mínimos locais. Com base na metodologia multiescala e os operadores de modelagem, método de inversão, e condições de borda mais adequados, foram testados 3 diferentes e complexos modelos de velocidades, o Marmousi 250×767 , o SEG-EAGE *Overthrust* 2-D e o Modelo Falhas, que resultou em modelos finais de alta resolução e coerentes com os resultados da literatura (Yang et al., 2015; dos Santos, 2013).

A economia de memória na GPU, discutidas e implementadas ao longo desta dissertação, utilizando a borda efetiva e o operador REM, dão abertura a trabalhos futuros que testem modelos de velocidades 2-D maiores, ou mesmo 3-D.

Agradecimentos

Agradeço à CAPES pelo financiamento do projeto mediante bolsa de mestrado CAPES-DS.

Obrigado ao prof. Dr. Reynam Pestana pela orientação e oportunidade de trabalhar com GPU aplicado à FWI, pelo suporte teórico, de recursos computacionais e dúvidas ao longo do trabalho.

Agradeço a Adriano Wagner, pelos códigos base de modelagem, REM e DF, e por todas as dúvidas no decorrer do processo. Sem sua ajuda as implementações não seriam possíveis.

Agradeço a Pengliang Yang, por disponibilizar códigos base para a RTM e FWI, pontos de partida dessa tese.

A Edvaldo Araújo pelas dúvidas teóricas, em especial na seção de bordas absorvedoras.

Obrigado à minha mãe, Maria Helena Gomes Koehne, por incutir desde cedo a prática dos estudos. Essa dissertação é uma consequência direta disso. Agradeço a minha irmã, Maria Cecília Koehne, e a minhas tias, Clotildes e Alice Koehne, por todo o apoio durante o processo.

Aos colegas Peterson, Rodrigo, Daniel, Cristian e Protásio por todas as discussões e sugestões. A Wilson Figueiró e Hedison Sato pelas dúvidas eventuais. A Leonardo Moreira pela elaboração das figuras dos métodos de inversão.

Apêndice A

Dedução das equações matriciais do método inverso

A perturbação $\Delta \mathbf{m}$ que atualiza o modelo corrente $\mathbf{m}$, pela formulação mínimos quadrados do problema inverso, envolve derivar a função objetivo em relação a cada um dos N parâmetros do modelo. Como a função objetivo é função do dado calculado e observado, que tem M amostras cada, explicitar $\Delta \mathbf{m}$ envolve derivar cada uma das M amostras em relação a cada um dos N parâmetros. Esse conjunto de equações lineares pode ser então compactado na formulação matricial vista no capítulo 3. O objetivo desse apêndice é mostrar como se chega à formulação matricial a partir dessas equações individuais. A formulação matricial discreta para inversão da forma de onda completa foi inicialmente apresentada por Pratt, Shin e Hicks (1998), no domínio da frequência. Aqui apresentamos o mesmo processo para o domínio do tempo.

A.1 Formulação matricial de $\Delta \mathbf{m}$ em função de $\mathbf{d}_{\text{cal}}$

A equação 3.9 do capítulo 3 dá a expressão de $\Delta \mathbf{m}$ em termos da função objetivo E . No caso da inversão da forma de onda, é preciso ir um passo adiante e expressar $\Delta \mathbf{m}$ em termos do dado calculado $\mathbf{d}_{\text{cal}}$, através da regra da cadeia, visto que $E(\mathbf{m}) = \frac{1}{2}(\mathbf{d}_{\text{obs}} - \mathbf{d}_{\text{cal}})^T(\mathbf{d}_{\text{obs}} - \mathbf{d}_{\text{cal}})$. O dado observado $\mathbf{d}_{\text{obs}}$ é uma constante, mas o dado calculado varia em função do modelo, durante o processo iterativo. Relembrando a expressão para $\Delta \mathbf{m}$:

$$\Delta \mathbf{m} = (H_E)^{-1} \nabla E. \quad (\text{A.1})$$

Explicitando a função objetivo,

$$\begin{aligned} E(\mathbf{m}) &= \frac{1}{2}(\mathbf{d}_{\text{obs}} - \mathbf{d}_{\text{cal}})^T(\mathbf{d}_{\text{obs}} - \mathbf{d}_{\text{cal}}), \\ E(\mathbf{m}) &= \frac{1}{2}(\mathbf{d}_{\text{obs}}^T \mathbf{d}_{\text{obs}} - 2\mathbf{d}_{\text{obs}}^T \mathbf{d}_{\text{cal}} + \mathbf{d}_{\text{cal}}^T \mathbf{d}_{\text{cal}}). \end{aligned} \quad (\text{A.2})$$

Derivando no i -ésimo parâmetro m_i ,

$$\begin{aligned} \frac{\partial E(\mathbf{m})}{\partial m_i} &= \frac{1}{2} \left(\mathbf{0} - 2 \mathbf{0}^T \mathbf{d}_{\text{cal}} - 2\mathbf{d}_{\text{obs}}^T \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_i} + \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_i}^T \mathbf{d}_{\text{cal}} + \mathbf{d}_{\text{cal}}^T \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_i} \right), \\ \frac{\partial E(\mathbf{m})}{\partial m_i} &= \frac{1}{2} \left(-2\mathbf{d}_{\text{obs}}^T \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_i} + 2\mathbf{d}_{\text{cal}}^T \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_i} \right), \\ \frac{\partial E(\mathbf{m})}{\partial m_i} &= \frac{1}{2} \left(-2\Delta \mathbf{d}^T \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_i} \right), \\ \frac{\partial E(\mathbf{m})}{\partial m_i} &= -\frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_i}^T \Delta \mathbf{d}. \end{aligned} \quad (\text{A.3})$$

Derivando em cada um dos parâmetros $m_1, \dots, m_N$, temos as equações:

$$\frac{\partial E(\mathbf{m})}{\partial m_1} = -\frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_1}^T \Delta \mathbf{d}, \quad (\text{A.4})$$

$$\frac{\partial E(\mathbf{m})}{\partial m_2} = -\frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_2}^T \Delta \mathbf{d}, \quad (\text{A.5})$$

$$\vdots \quad (\text{A.6})$$

$$\frac{\partial E(\mathbf{m})}{\partial m_i} = -\frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_i}^T \Delta \mathbf{d}, \quad (\text{A.7})$$

$$\vdots \quad (\text{A.8})$$

$$\frac{\partial E(\mathbf{m})}{\partial m_N} = -\frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_N}^T \Delta \mathbf{d}. \quad (\text{A.9})$$

Agrupando em forma matricial,

$$\begin{bmatrix} \frac{\partial E(\mathbf{m})}{\partial m_1} \\ \vdots \\ \frac{\partial E(\mathbf{m})}{\partial m_N} \end{bmatrix} = - \begin{bmatrix} \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_1}^T \\ \vdots \\ \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_N}^T \end{bmatrix} \begin{bmatrix} \Delta d_1 \\ \vdots \\ \Delta d_M \end{bmatrix} \quad (\text{A.10})$$

onde a primeira matriz é $N \times 1$, a segunda $N \times M$ (jacobiana) e a terceira $M \times 1$ (resíduo dos dados). A equação matricial então fica

$$\frac{\partial E(\mathbf{m})}{\partial \mathbf{m}} = -J^T \Delta \mathbf{d}, \quad (\text{A.11})$$

de onde conclui-se que o termo $\frac{1}{2}$ da função objetivo serve somente para que a expressão acima do gradiente em função da jacobiana não esteja multiplicada por 2.

A matriz Hessiana contém as segundas derivadas da função objetivo em relação a cada parâmetro. Cada coluna da Hessiana é uma derivada do gradiente em um parâmetro de $\mathbf{m}$:

$$H_E = \begin{bmatrix} \frac{\partial}{\partial m_1} (\nabla E(\mathbf{m})) & \frac{\partial}{\partial m_2} (\nabla E(\mathbf{m})) & \dots & \frac{\partial}{\partial m_N} (\nabla E(\mathbf{m})) \end{bmatrix}, \quad (\text{A.12})$$

que é uma matriz NxN. Da dedução anterior, o gradiente é dado por

$$\nabla E(\mathbf{m}) = - \begin{bmatrix} \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_1}^T \\ \vdots \\ \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_N}^T \end{bmatrix} \begin{bmatrix} \Delta d_1 \\ \vdots \\ \Delta d_M \end{bmatrix} = - \begin{bmatrix} \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_1}^T \Delta \mathbf{d} \\ \vdots \\ \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_N}^T \Delta \mathbf{d} \end{bmatrix} \quad (\text{A.13})$$

Vamos iniciar explicitando a forma da primeira coluna da Hessiana, e o preenchimento das demais colunas se dará de modo análogo.

$$\frac{\partial E(\mathbf{m})}{\partial m_1} = \begin{bmatrix} \frac{\partial}{\partial m_1} \left(\frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_1} \right)^T \Delta \mathbf{d} \\ \frac{\partial}{\partial m_1} \left(\frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_2} \right)^T \Delta \mathbf{d} \\ \vdots \\ \frac{\partial}{\partial m_1} \left(\frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_N} \right)^T \Delta \mathbf{d} \end{bmatrix} \quad (\text{A.14})$$

O primeiro termo de A.14 fica:

$$\frac{\partial}{\partial m_1} \left(\frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_1} \right)^T \Delta \mathbf{d} = \frac{\partial^2 \mathbf{d}_{\text{cal}}}{\partial m_1^2}^T \Delta \mathbf{d} + \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_1}^T \frac{\partial \Delta \mathbf{d}}{\partial m_1}; \quad (\text{A.15})$$

lembrando que

$$\frac{\partial \Delta \mathbf{d}}{\partial m_1} = - \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_1}, \quad (\text{A.16})$$

as equações explícitas para a primeira coluna da Hessiana ficam

$$\frac{\partial}{\partial m_1} \left(\frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_1} \right)^T \Delta \mathbf{d} = \frac{\partial^2 \mathbf{d}_{\text{cal}}}{\partial m_1^2}^T \Delta \mathbf{d} + \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_1}^T \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_1}, \quad (\text{A.17})$$

$$\frac{\partial}{\partial m_1} \left(\frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_2} \right)^T \Delta \mathbf{d} = \frac{\partial^2 \mathbf{d}_{\text{cal}}}{\partial m_1 \partial m_2}^T \Delta \mathbf{d} + \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_2}^T \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_1}, \quad (\text{A.18})$$

$$\vdots \quad (\text{A.19})$$

$$\frac{\partial}{\partial m_1} \left(\frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_N} \right)^T \Delta \mathbf{d} = \frac{\partial^2 \mathbf{d}_{\text{cal}}}{\partial m_1 \partial m_N}^T \Delta \mathbf{d} + \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_N}^T \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_1}. \quad (\text{A.20})$$

Organizando essas N equações em forma matricial,

$$\frac{\partial E(\mathbf{m})}{\partial m_1} = \begin{bmatrix} \frac{\partial^2 \mathbf{d}_{\text{cal}}}{\partial m_1^2}^T \Delta \mathbf{d} + \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_1}^T \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_1} \\ \frac{\partial^2 \mathbf{d}_{\text{cal}}}{\partial m_1 \partial m_2}^T \Delta \mathbf{d} + \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_2}^T \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_1} \\ \vdots \\ \frac{\partial^2 \mathbf{d}_{\text{cal}}}{\partial m_1 \partial m_N}^T \Delta \mathbf{d} + \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_N}^T \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_1} \end{bmatrix}, \quad (\text{A.21})$$

ou

$$\frac{\partial E(\mathbf{m})}{\partial m_1} = \left[\frac{\partial J}{\partial m_1}^T \Delta \mathbf{d} + J^T \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_1} \right]. \quad (\text{A.22})$$

Analogamente, incluindo as demais colunas, a Hessiana completa fica:

$$H_E = \left[\left[\frac{\partial J}{\partial m_1}^T \Delta \mathbf{d} + J^T \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_1} \right] \quad \left[\frac{\partial J}{\partial m_2}^T \Delta \mathbf{d} + J^T \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_2} \right] \quad \dots \quad \left[\frac{\partial J}{\partial m_N}^T \Delta \mathbf{d} + J^T \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_N} \right] \right], \quad (\text{A.23})$$

onde são N colunas Nx1, e a Hessiana é NxN. Separando em duas matrizes,

$$H_E = \left[\left[\frac{\partial J}{\partial m_1}^T \Delta \mathbf{d} \right] \quad \left[\frac{\partial J}{\partial m_2}^T \Delta \mathbf{d} \right] \quad \dots \quad \left[\frac{\partial J}{\partial m_N}^T \Delta \mathbf{d} \right] \right] + \left[\left[J^T \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_1} \right] \quad \left[J^T \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_2} \right] \quad \dots \quad \left[J^T \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_N} \right] \right], \quad (\text{A.24})$$

$$H_E = \left[\frac{\partial J}{\partial m_1}^T \Delta \mathbf{d} \quad \frac{\partial J}{\partial m_2}^T \Delta \mathbf{d} \quad \dots \quad \frac{\partial J}{\partial m_N}^T \Delta \mathbf{d} \right] + J^T \left[\frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_1} \quad \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_2} \quad \dots \quad \frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_N} \right], \quad (\text{A.25})$$

$$H_E = \left[\frac{\partial J}{\partial m_1}^T \Delta \mathbf{d} \quad \frac{\partial J}{\partial m_2}^T \Delta \mathbf{d} \quad \dots \quad \frac{\partial J}{\partial m_N}^T \Delta \mathbf{d} \right] + J^T J, \quad (\text{A.26})$$

ou:

$$H_E = R + H_a, \quad (\text{A.27})$$

onde $H_a = J^T J$ é denominada Hessiana aproximada, que tem a contribuição apenas das derivadas primeiras de $\mathbf{d}_{\text{cal}}$ e o termo R , que contém informações das segundas derivadas, pode ser compactado pela notação:

$$R = \frac{\partial J^T}{\partial \mathbf{m}^T} [\Delta \mathbf{d}, \Delta \mathbf{d}, \dots, \Delta \mathbf{d}], \quad (\text{A.28})$$

de forma que:

$$H_E = \frac{\partial J}{\partial \mathbf{m}^T}^T [\Delta \mathbf{d}, \dots, \Delta \mathbf{d}] + J^T J. \quad (\text{A.29})$$

que é a forma da Hessiana apresentada na equação 3.13 capítulo 3. É válido enfatizar que a notação $\frac{\partial J^T}{\partial \mathbf{m}}$ representa uma matriz com as dimensões $N \times M$ de J^T , onde cada coluna i foi derivada no parâmetro m_i . Já a notação $\frac{\partial J^T}{\partial \mathbf{m}^T}$, que aparece na equação da Hessiana, representa uma matriz de matrizes $N \times (N \times M)$, cada matriz derivada no seu respectivo m_i , e a multiplicação pela matriz dos $\Delta \mathbf{d}$ **não é uma multiplicação matricial usual**, é uma operação bloco matricial que deve ter como resultado a forma expressa na equação A.26.

A.2 Matriz sensibilidade: forma usual

A matriz sensibilidade, ou matriz de Fréchet, ou matriz Jacobiana, representa, em suas colunas, como o dado calculado responde às perturbações em cada um dos parâmetros do modelo. Ou seja, é a sensibilidade dos dados calculados a variações no modelo. Encontrar a matriz sensibilidade envolve então encontrar cada uma das N derivadas $\frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_j}$, com $j = \{1, \dots, N\}$. Seguindo o desenvolvimento de Porsani (2008), apresentamos os passos para encontrar a matriz sensibilidade, aproximando as derivadas por um método de diferenças finitas de primeira ordem:

- Dado o modelo $\mathbf{m} = (m_1, m_2, \dots, m_N)^T$, perturbar o 1º parâmetro m_1 em uma pequena porcentagem (por exemplo, 1%), obtendo $\mathbf{m}_1 = (m_{1p}, m_2, \dots, m_N)^T$;
- Calcula $\mathbf{u}_1 = G(\mathbf{m}_1)$ (campo em (z, x, t)). Restringe à geometria dos dados: $\mathbf{d}_{\text{cal1}} = S_{s,r}(\mathbf{u}_1)$ (campo em (x, t));

- Acha a derivada de cada um dos M parâmetros de $\mathbf{d}_{\text{cal}}$. Sendo $m_{1p} - m_1 = \Delta m$ ($=1\%$),

$$\frac{\partial \mathbf{d}_{\text{cal}}}{\partial m_1} = \frac{\mathbf{d}_{\text{cal1}} - \mathbf{d}_{\text{cal}}}{\Delta m},$$

que é a primeira coluna da matriz Jacobiana (tamanho M).

- Repete os mesmos passos, perturbando o segundo parâmetro, encontrando a segunda coluna, e assim sucessivamente, N vezes, até encontrar as N colunas. Ao final, tem-se a matriz Jacobiana MxN.

Ou seja, mesmo a formulação mais simples possível para as derivadas (diferenças finitas de primeira ordem) é muito custosa, pois envolve calcular N resoluções do problema direto.

Usualmente, encontrar a matriz Hessiana envolve realizar o mesmo processo, perturbando cada um dos MxN termos da matriz sensibilidade. Ou seja, no total seriam necessárias N^2 resoluções do problema direto. Por isso, se diz que, na maioria dos casos, o cálculo da matriz Hessiana é proibitivo, se fazendo necessárias aproximações ou métodos alternativos.

Apêndice **B**

Cálculo do gradiente pelo método adjunto

O cálculo do gradiente da função objetivo 3.2 por meio da matriz de Fréchet é extremamente custoso. Neste apêndice apresentamos a dedução da fórmula para o gradiente pelo método adjunto, baseada em Plessix (2006), pela formulação do funcional estendido (ou Lagrangiano). Leitores interessados na dedução do método pela teoria da perturbação (Lailly, 1983; Tarantola, 1984) podem encontrá-la integralmente também em Plessix (2006). Uma excelente dedução, também pelo método da perturbação, e baseado em funções de Green, pode ser encontrada em Yang et al. (2015).

Partimos da equação clássica para o problema direto em geofísica (Menke, 1984):

$$\mathbf{u} = G(\mathbf{m}), \quad (\text{B.1})$$

onde $\mathbf{u}$ são os dados calculados (no caso, em z, x, t), $\mathbf{m}$ (em z, x) os parâmetros do modelo, e G o operador que representa as relações físico-matemáticas do problema. A equação pode ser reformulada por:

$$F(\mathbf{u}, \mathbf{m}) = 0, \quad (\text{B.2})$$

que generaliza e engloba B.1.

A solução do problema inverso (encontrar $\mathbf{m}$ a partir de $\mathbf{u}$) como um problema de otimização envolve minimizar uma função objetivo $E(\mathbf{m})$. Definindo-a em termos de um funcional h , temos:

$$E(\mathbf{m}) = h(\mathbf{u}, \mathbf{m}); \quad (\text{B.3})$$

um exemplo de funcional a ser minimizado é a soma dos resíduos quadrados (ver seção seguinte).

Nos métodos gradiente a busca do mínimo depende do gradiente $\frac{\partial E}{\partial \mathbf{m}}$. Pode-se definir um funcional aumentado, ou Lagrangiano, em termos dos parâmetros do modelo $\mathbf{m}$, a variável de estado $\tilde{\mathbf{u}}$, a variável de estado adjunto $\tilde{\lambda}$, e os funcionais h e F :

$$\mathcal{L}(\tilde{\mathbf{u}}, \tilde{\lambda}, \mathbf{m}) = h(\tilde{\mathbf{u}}, \mathbf{m}) - \langle \tilde{\lambda}, F(\tilde{\mathbf{u}}, \mathbf{m}) \rangle, \quad (\text{B.4})$$

onde $\langle, \rangle$ denota produto interno, e a notação $\tilde{\mathbf{u}}$ e $\tilde{\lambda}$ representa qualquer valor de $\mathbf{u}$ e λ , respectivamente. Como veremos, o Lagrangiano $\mathcal{L}$ é construído de forma que $\frac{\partial \mathcal{L}}{\partial \mathbf{m}} = \frac{\partial E}{\partial \mathbf{m}}$ no ponto de sela, e o custoso cálculo da matriz de Fréchet $\frac{\partial \tilde{\mathbf{u}}}{\partial \mathbf{m}}$, associada a $\frac{\partial E}{\partial \mathbf{m}}$, pode ser contornado ao se utilizar a equação de estado adjunta, relacionada à variável adjunta $\tilde{\lambda}$ do Lagrangiano.

As equações de estado e estado adjunto são encontradas no ponto de sela ($\tilde{\mathbf{u}} = \mathbf{u}, \tilde{\lambda} = \lambda$) onde por definição $\tilde{\mathbf{u}} = \mathbf{u}$ é um ponto de mínimo, $\tilde{\lambda} = \lambda$ é um ponto de máximo, e as duas derivadas parciais são nulas (Ciarlet, 1989). No ponto de sela:

$$\frac{\partial \mathcal{L}(\mathbf{u}, \lambda, \mathbf{m})}{\partial \tilde{\lambda}} = -F(\mathbf{u}, \mathbf{m}) = 0, \quad (\text{B.5})$$

$$\frac{\partial \mathcal{L}(\mathbf{u}, \lambda, \mathbf{m})}{\partial \tilde{\mathbf{u}}} = \frac{\partial h(\mathbf{u}, \mathbf{m})}{\partial \tilde{\mathbf{u}}} - \left(\frac{\partial F(\mathbf{u}, \mathbf{m})}{\partial \tilde{\mathbf{u}}} \right)^* \lambda = 0. \quad (\text{B.6})$$

A equação B.5 é a equação de estado, e equivale ao problema direto. A equação B.6 é a equação de estado adjunto, onde $*$ é o operador adjunto (generalização do operador transposto conjugado).

Quanto ao cálculo do gradiente $\frac{\partial E}{\partial \mathbf{m}}$, observamos que no ponto de sela $F(\mathbf{u}, \mathbf{m}) = 0$, de forma que:

$$\mathcal{L}(\mathbf{u}, \lambda, \mathbf{m}) = h(\mathbf{u}, \mathbf{m}) = E(\mathbf{m}). \quad (\text{B.7})$$

Derivando em relação a $\mathbf{m}$,

$$\frac{\partial \mathcal{L}(\mathbf{u}, \lambda, \mathbf{m})}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial \mathbf{m}} + \frac{\partial \mathcal{L}(\mathbf{u}, \lambda, \mathbf{m})}{\partial \mathbf{m}} = \frac{\partial E}{\partial \mathbf{m}}; \quad (\text{B.8})$$

e por B.5 a derivada em $\mathbf{u}$ é nula no ponto de sela, daí:

$$\frac{\partial \mathcal{L}(\mathbf{u}, \lambda, \mathbf{m})}{\partial \mathbf{m}} = \frac{\partial E}{\partial \mathbf{m}}. \quad (\text{B.9})$$

Derivando B.4 em relação a $\mathbf{m}$ obtemos a expressão para o gradiente envolvendo a variável adjunta λ :

$$\frac{\partial E}{\partial \mathbf{m}} = \frac{\partial h(\tilde{\mathbf{u}}, \mathbf{m})}{\partial \mathbf{m}} - \left\langle \lambda, \frac{\partial F(\mathbf{u}, \mathbf{m})}{\partial \mathbf{m}} \right\rangle, \quad (\text{B.10})$$

Na seção seguinte será explicitado o gradiente para o caso da onda acústica, baseado nas equações B.5, B.6 e B.10.

B.1 Aplicação à FWI

No caso da FWI no domínio do tempo, o funcional $h(\mathbf{u}, \mathbf{m})$ é dado por:

$$h(\mathbf{u}, \mathbf{m}) = \frac{1}{2} \sum_{s,r} \int_0^T (S_{s,r} \mathbf{u} - \mathbf{d}_{\text{obs}})^2, \quad (\text{B.11})$$

onde $\mathbf{d}_{\text{obs}}$ é o dado observado (depende de x, t) e $S_{s,r} \mathbf{u}$ é o dado calculado, restrito por um operador $S_{s,r}$ (subíndice s indica fonte, r receptores) que limita $\mathbf{u}$ (*snapshots* em z, x, t) à geometria espacial de $\mathbf{d}_{\text{obs}}$ (sismograma em x, t). $\mathbf{m}$ são as velocidades acústicas do modelo.

A restrição $F(\mathbf{u}, \mathbf{m}) = 0$, ou problema direto, é dada pela equação da onda acústica:

$$F(\mathbf{u}, \mathbf{m}) = \frac{1}{v^2} \frac{\partial^2 \mathbf{u}}{\partial t^2} - \nabla^2 \mathbf{u} - f_s = 0, \quad (\text{B.12})$$

que está sujeita às condições de contorno:

$$\begin{aligned} \mathbf{u}(t=0) &= 0, \\ \frac{\partial \mathbf{u}(t=0)}{\partial t} &= 0, \end{aligned} \quad (\text{B.13})$$

que podem ser interpretadas da seguinte maneira: no tempo 0 a fonte ainda não explodiu, então todo o campo de pressão é zero. O mesmo vale para a derivada do campo de pressão no instante 0; apesar da iminência de explosão, em $t = 0$ a derivada da Ricker é praticamente nula (ver Figura 1.2 -a).

O Lagrangiano, definido em B.4, aqui é dado por:

$$\mathcal{L}(\mathbf{u}, \lambda, \mathbf{m}) = \frac{1}{2} \sum_{s,r} \int_0^T (S_{s,r} \mathbf{u} - \mathbf{d}_{\text{obs}})^2 d\mathbf{x} dt - \sum_{s,r} \int_0^T \int_{\mathbf{x}} \lambda \left[\frac{1}{v^2} \frac{\partial^2 \mathbf{u}}{\partial t^2} - \nabla^2 \mathbf{u} - f_s \right] d\mathbf{x} dt, \quad (\text{B.14})$$

onde λ , a variável de estado adjunto, tem as mesmas dimensões (z, x, t) do campo propagado $\mathbf{u}$.

Reescrevendo a integral do segundo termo em B.14 como a soma de três integrais,

$$\begin{aligned} & \int_0^T \int_{\mathbf{x}} \lambda \left[\frac{1}{v^2} \frac{\partial^2 \mathbf{u}}{\partial t^2} - \nabla^2 \mathbf{u} - f_s \right] d\mathbf{x} dt = \\ & \int_0^T \int_{\mathbf{x}} \lambda \frac{1}{v^2} \frac{\partial^2 \mathbf{u}}{\partial t^2} d\mathbf{x} dt + \int_0^T \int_{\mathbf{x}} -\lambda \nabla^2 \mathbf{u} d\mathbf{x} dt + \int_0^T \int_{\mathbf{x}} -\lambda f_s d\mathbf{x} dt. \end{aligned} \quad (\text{B.15})$$

Aplicando integração por partes na primeira integral,

$$\int_0^T \lambda \frac{1}{v^2} \frac{\partial^2 \mathbf{u}}{\partial t^2} = \int_0^T g dh = (gh) \Big|_0^T - \int_0^T h dg, \quad (\text{B.16})$$

com

$$\begin{aligned} g &= \lambda; \\ dg &= \frac{\partial \lambda}{\partial t} dt; \\ dh &= \frac{1}{v^2} \frac{\partial^2 \mathbf{u}}{\partial t^2} dt; \end{aligned} \quad (\text{B.17})$$

$$h = \frac{1}{v^2} \frac{\partial \mathbf{u}}{\partial t}. \quad (\text{B.18})$$

A integral então fica

$$\int_0^T \lambda \frac{1}{v^2} \frac{\partial^2 \mathbf{u}}{\partial t^2} = \left(\lambda \frac{1}{v^2} \frac{\partial \mathbf{u}}{\partial t} \right) \Big|_0^T - \int_0^T \frac{1}{v^2} \frac{\partial \mathbf{u}}{\partial t} \frac{\partial \lambda}{\partial t} dt. \quad (\text{B.19})$$

Aplicando novamente a integração por partes a B.20, com $g = \frac{1}{v^2} \frac{\partial \lambda}{\partial t}$ e $dh = \frac{\partial \mathbf{u}}{\partial t} dt$:

$$\int_0^T \lambda \frac{1}{v^2} \frac{\partial^2 \mathbf{u}}{\partial t^2} = \left(\lambda \frac{1}{v^2} \frac{\partial \mathbf{u}}{\partial t} \right) \Big|_0^T - \left(\mathbf{u} \frac{1}{v^2} \frac{\partial \lambda}{\partial t} \right) \Big|_0^T + \int_0^T \frac{1}{v^2} \mathbf{u} \frac{\partial^2 \lambda}{\partial t^2} dt. \quad (\text{B.20})$$

Aplicando as condições de contorno B.13, os termos em $t = 0$ das diferenciais são zerados. Aplicando as seguintes condições de contorno a λ

$$\begin{aligned} \lambda(t = T) &= 0, \\ \frac{\partial \lambda(t = T)}{\partial t} &= 0, \end{aligned} \quad (\text{B.21})$$

os termos em $t = T$ das diferenciais são zerados, resultando em:

$$\int_0^T \lambda \frac{1}{v^2} \frac{\partial^2 \mathbf{u}}{\partial t^2} = \int_0^T \mathbf{u} \frac{1}{v^2} \frac{\partial^2 \lambda}{\partial t^2} dt. \quad (\text{B.22})$$

Analogamente, aplicando o mesmo procedimento de integração por partes à segunda integral de B.15, temos:

$$\int_{\mathbf{x}} -\lambda \nabla^2 \mathbf{u} d\mathbf{x} = \int_{\mathbf{x}} -\mathbf{u} \nabla^2 \lambda d\mathbf{x}, \quad (\text{B.23})$$

de forma que o Lagrangiano pode ser enfim reescrito como:

$$\mathcal{L}(\mathbf{u}, \lambda, \mathbf{m}) = \frac{1}{2} \sum_{s,r} \int_0^T (S_{s,r} \mathbf{u} - \mathbf{d}_{\text{obs}})^2 d\mathbf{x} dt - \sum_r \int_0^T \int_{\mathbf{x}} \mathbf{u} \left[\frac{1}{v^2} \frac{\partial^2 \lambda}{\partial t^2} - \nabla^2 \lambda \right] - \lambda f_s d\mathbf{x} dt. \quad (\text{B.24})$$

Como mostrado na seção anterior, $\frac{\partial \mathcal{L}}{\partial \mathbf{u}} = 0$ define a equação adjunta. Derivando B.24 em relação a $\mathbf{u}$ e igualando a zero,

$$\frac{1}{v^2} \frac{\partial^2 \lambda}{\partial t^2} - \nabla^2 \lambda = \sum_r S_{s,r}^T (S_r \mathbf{u} - \mathbf{d}_{\text{obs}}). \quad (\text{B.25})$$

Pode-se observar que a equação B.25 é equivalente à equação da onda acústica B.12, mas nesse caso o termo fonte é dado pelo resíduo entre dado calculado e observado $\sum_{s,r} S_{s,r}^T (S_{s,r} \mathbf{u} - \mathbf{d}_{\text{obs}})$, injetado nas posições dos receptores. As condições de contorno B.21 para o tempo final T indicam que a inserção do resíduo se dá de maneira reversa no tempo, de $t = T$ a $t = 0$, como na RTM.

Por fim, o gradiente é dado por $\frac{\partial E}{\partial \mathbf{m}} = \frac{\partial \mathcal{L}}{\partial \mathbf{m}}$, como visto anteriormente. Derivando B.24 em $\mathbf{m}$ (no caso, v),

$$\frac{\partial E}{\partial \mathbf{m}} = \frac{2}{v^3} \sum_{s,r} \int_0^T \lambda \frac{\partial \mathbf{u}}{\partial t^2}. \quad (\text{B.26})$$

O gradiente $\frac{\partial E}{\partial m}$ tem as dimensões de seção migrada, e sua computação é efetivamente uma migração reversa no tempo:

- o campo dos receptores λ é obtido via injeção do resíduo na superfície, do tempo máximo ao tempo zero;
- o campo da fonte é dado pela modelagem do campo $\mathbf{u}$, e a sua derivada segunda no tempo é normalmente substituída por $v^2 \nabla^2 \mathbf{u}(t)$ (equação da onda), de forma a evitar o custo de memória inerente ao cálculo de derivadas temporais;
- a condição de imagem é a mesma da RTM (a menos de um fator $\frac{2}{v^3}$), a crosscorrelação entre campo da fonte e receptor(aqui, resíduo), para todos os tiros.

Apêndice C

Pseudocódigos

Neste apêndice são apresentados pseudocódigos com os passos principais de algumas implementações desta dissertação. Todas as variáveis iniciadas por "*d_*" representam variáveis na GPU ("*d*" expressa *device*, que corresponde à GPU).

Modelagem usando operador de DF:

Algoritmo 2: Modelagem sísmica (operador DF, borda com *taper*)

- 1: Lê parâmetros: $nz, nx, dz, dx, f_{peak}, dt, nt, ns, ng$;
- 2: Lê vetores de geometria (fontes, receptores) e salva na GPU: $d_sxz(ns), d_gxz(ng)$;
- 3: Lê campo de velocidades $vel(nz, nx)$;
- 4: Expande vel com borda: $nnz = nz + 2nb; nnx = nx + 2nb; VEL(nnz, nnx)$;
- 5: Aloca na GPU $d_VEL(nnz, nnx)$;
- 6: Copia CPU-GPU (*host to device*: h2d): $d_VEL \leftarrow VEL$;
- 7: Aloca (GPU) campos: $d_p0(nnz, nnx), d_p1(nnz, nnx), ptr$; (ponteiro nulo auxiliar)
- 8: Aloca (GPU) sismograma de 1 tiro: $d_seis(nt, ng)$;
- 9: Inicializa e preenche *wavelet* na GPU: $d_wlt(nt)$;
- 10: Inicializa coeficientes de DF;
- 11: Inicializa REM (se necessário);
- 12: Inicializa *taper*; aloca e preenche na GPU: $d_taper(nnz, nnx)$;
- 13: Define a posição de todas as fontes: d_sxz ;
- 14: **para** tiros $is = (1, \dots, ns)$ **faça**
- 15: Define geometria dos receptores no modelo: d_gxz ;
- 16: **para** tempo $it = (0, \dots, nt)$ **faça**
- 18: Insere $d_wlt(it)$ em d_p1 na posição dada por $d_sxz(is)$;

```

19:      Acha campo futuro; salva em  $d\_p0$ ; (propaga_GPU)
20:      Aplica  $d\_taper$  em  $d\_p0, d\_p1$ ;
21:       $ptr \leftarrow d\_p0$ ; ( $ptr \leftarrow$  futuro)
22:       $d\_p0 \leftarrow d\_p1$ ; ( $d\_p0 \leftarrow$  passado)
23:       $d\_p1 \leftarrow ptr$ ; ( $d\_p1 \leftarrow$  futuro)
24:      Grava fatia  $d\_p1(0, d\_gxz)$  em  $d\_seis$ ;
25:      fim para
26:      Copia d2h tiro atual  $d\_seis$ , salva em arquivo;
27: fim para
28: Desaloca todas as variáveis;

```

A modelagem usando o operador PS tem o mesmo algoritmo, apenas permutando a inicialização da DF pela inicialização do PS. A função **propaga_GPU** é que faz a extrapolação por FD, PS, REM-FD ou REM-PS, dependendo da escolha.

A seguir é apresentado o pseudocódigo para RTM na GPU, também com operador DF e borda absorvedora do tipo *taper*.

Algoritmo 3: RTM (operador DF, borda com *taper*)

```

1: Lê parâmetros:  $nz, nx, dz, dx, f_{peak}, dt, nt, ns, ng$ ;
2: Lê vetores de geometria (fontes, receptores) e salva na GPU:  $d\_sxz(ns), d\_gxz(ng)$ ;
3: Lê campo de velocidades  $vel(nz, nx)$ ;
3: Lê os tiros  $dobs(ns \times ng, nt)$ ;
4: Aloca (GPU) espaço de 1 tiro:  $d\_seis$ ;
5: Expande  $vel$  com borda:  $nnz = nz + 2nb$ ;  $nnx = nx + 2nb$ ;  $VEL(nnz, nnx)$ ;
6: Aloca na GPU  $d\_VEL(nnz, nnx)$ ;
7: Copia CPU-GPU (host to device: h2d):  $d\_VEL \leftarrow VEL$ ;
8: Aloca (GPU) campos da fonte:  $d\_sp0(nnz, nnx), d\_sp1(nnz, nnx), ptr$ ; (ponteiro nulo auxiliar)
9: Aloca (GPU) campos dos receptores:  $d\_gp0(nnz, nnx), d\_gp1(nnz, nnx)$ ;
10: Aloca (GPU) borda efetiva:  $d\_bndr(2 \times (NJ - 1) \times (nz + nx) \times nt)$ ; (NJ = ordem do operador)
11: Aloca (GPU) imagem final:  $d\_I(nnz, nnx)$ ;
12: Inicializa e preenche wavelet na GPU:  $d\_wlt(nt)$ ;
13: Inicializa coeficientes de DF;
14: Inicializa REM (se necessário);
15: Inicializa taper; aloca e preenche na GPU:  $d\_taper(nnz, nnx)$ ;
16: Define a posição de todas as fontes:  $d\_sxz$ ;
17: para tiros  $is = (1, \dots, ns)$  faça

```

```

18:   Define geometria dos receptores no modelo:  $d\_gxz(is)$ ;
19:   para tempo  $it = (0, \dots, nt)$  faça
20:       Insere  $d\_wlt(it)$  em  $d\_sp1$  na posição dada por  $d\_sxz(is)$ ;
21:       Acha campo futuro; salva em  $d\_sp0$ ; (propaga_GPU)
22:       Aplica  $d\_taper$  em  $d\_sp0, d\_sp1$ ;
23:        $ptr \leftarrow d\_sp0$ ;  $d\_sp0 \leftarrow d\_sp1$ ;  $d\_sp1 \leftarrow ptr$ ;
24:       Grava borda efetiva  $d\_bndr(it)$ ;
25:   fim para
26:    $ptr \leftarrow d\_sp0$ ;  $d\_sp0 \leftarrow d\_sp1$ ;  $d\_sp1 \leftarrow ptr$ ;
27:   Lê tiro atual e copia h2d:  $d\_seis \leftarrow dobs(is)$ ;
28:   para tempo  $it = (nt, \dots, 0)$  faça
29:       Depropagação da fonte
30:       Injeta  $d\_bndr(it)$  em  $d\_sp1$ ;
31:       Acha campo depropagado; salva em  $d\_sp0$ ; (propaga_GPU)
32:       Subtrai  $d\_wlt(it)$  em  $d\_sp1$  na posição dada por  $d\_sxz(is)$ ;
33:        $ptr \leftarrow d\_sp0$ ;  $d\_sp0 \leftarrow d\_sp1$ ;  $d\_sp1 \leftarrow ptr$ ;
34:       Depropagação dos receptores
35:       Injeta fatia  $d\_seis(it)$  em  $d\_gp1(z = 0, d\_gxz)$ ;
36:       Acha campo depropagado; salva em  $d\_gp0$ ; (propaga_GPU)
37:       Aplica  $d\_taper$  em  $d\_gp0, d\_gp1$ ;
38:        $ptr \leftarrow d\_gp0$ ;  $d\_gp0 \leftarrow d\_gp1$ ;  $d\_gp1 \leftarrow ptr$ ;
39:       Condição de imagem
40:       Acumula imagem em  $d\_I \leftarrow xcorr(d\_sp0, d\_gp0)$ ;
41:   fim para
42: fim para
43: Aplica filtro Laplaciano a  $d\_I$ ;
44: Copia d2h  $d\_I$ , salva imagem em arquivo;
45: Desaloca todas as variáveis;

```

A seguir, apresentamos os passos principais do pseudocódigo para FWI multiescala usando o método L-BFGS, operador DF e *taper*. É interessante salientar que, como mostra o código, o maior custo de memória da FWI decorre das duas maiores variáveis: a borda efetiva d_bndr , de tamanho $2 \times (nz + nx) \times (nj - 1) \times nt$ e o resíduo d_derr , de tamanho $ng \times ns \times nt$. nz e nx são as amostras verticais e horizontais do modelo, respectivamente; nj é a ordem do operador de diferenças finitas (no caso do operador PS, utilizou-se $nj = 10$ nas bordas); ng é o número de receptores por tiro, ns o número total de tiros, e nt o número

de amostras no tempo. Observa-se que ambas as variáveis dependem do número de amostras temporais nt . A redução de memória utilizada, observada nos testes para os operadores REM, decorre diretamente do fato do REM possibilitar a extrapolação dos campos a maiores Δt e menores nt , reduzindo o tamanho dessas duas variáveis principais e gerando economia de memória.

Algoritmo 4: FWI (L-BFGS, operador DF, *taper*)

- 1: Lê parâmetros: $nz, nx, dz, dx, f_{peak}, dt, nt, ns, ng$;
- 2: Lê parâmetros multiescala: $niter, maxrnds$; (iterações por rodada, n^o de rodadas)
- 3: Lê vetores de geometria (fontes, receptores) e salva na GPU: $d_sxz(ns), d_gxz(ng)$;
- 4: Lê campo de velocidades inicial $vel(nz, nx)$;
- 5: Lê os tiros $dobs_ALL(ns \times ng, nt)$;
- 6: Aloca (GPU) espaço de 1 tiro observado: $d_dobs(ng, nt)$;
- 7: Aloca (GPU) espaço de 1 tiro calculado: $d_dcal(ng, nt)$;
- 8: Aloca (GPU) espaço do resíduo total: $d_derr(ns \times ng, nt)$;
- 9: Expande vel com borda: $nnz = nz + 2nb$; $nnx = nx + 2nb$; $VEL(nnz, nnx)$;
- 10: Aloca na GPU $d_VEL(nnz, nnx)$;
- 11: Copia CPU-GPU (*host to device*: h2d): $d_VEL \leftarrow VEL$;
- 12: Aloca (GPU) campos da fonte e Laplaciano: $d_sp0(nnz, nnx), d_sp1(nnz, nnx)$, $d_slap(nnz, nnx)ptr$; (ponteiro nulo auxiliar)
- 13: Aloca (GPU) campos dos receptores e Laplaciano: $d_gp0(nnz, nnx), d_gp1(nnz, nnx)$, $d_glap(nnz, nnx)$;
- 14: Aloca (GPU) borda efetiva: $d_bndr(2 \times (NJ - 1) \times (nz + nx) \times nt)$; (NJ = ordem do operador)
- 15: Aloca (GPU) campos de iluminação da fonte: $d_illum(nnz, nnx)$; (para condição de imagem, eq. 2.6)
- 16: Aloca (GPU) campo de velocidade temporário: $d_vtmp(nnz, nnx)$; (cálculo de J_s , eq. 3.26)
- 17: Aloca (GPU) gradiente atual: $d_g(nnz, nnx)$;
- 18: Aloca (GPU) vetores de diferenças atual do L-BFGS: $d_vdiff(nz, nx), d_gdiff(nz, nx)$;
- 19: Aloca (GPU) *wavelet*: $d_wlt_fpeak(nt)$; (frequência de pico de $dobs$)
- 20: Aloca (GPU) *wavelet*: $d_wlt_frnd(nt)$; (frequência da rodada)
- 21: Aloca (GPU) parâmetros de inversão: d_obj, d_eps, d_alpha ;
- 22: Inicializa coeficientes de DF;
- 23: Inicializa REM (se necessário);
- 24: Inicializa *taper*; aloca e preenche na GPU: $d_taper(nnz, nnx)$;
- 25: Inicializa *wavelet*: d_wlt_fpeak com f_{peak} ;

26: Define a posição de todas as fontes: d_sxz ;

27: Aloca (GPU) matriz total do L-BFGS: $d_x(m, nz, nx), d_y(m, nz, nx)$; ($m = 10$, normalmente)

28: **Inicializa processo multiescala**

29: $maxiter = maxrnds \times niter$;

30: $deltaf = f_{peak}/maxrnds$;

31: **para** rodadas $irnd = (0, \dots, maxrnds)$ **faça**

32: $vfreq[irnd - 1] = irnd \times deltaf$; (vetor de frequência de cada rodada)

33: **fim para**

34: **Processo multiescala**

35: **para** rodadas $irnd = (0, \dots, maxrnds)$ **faça** (loop rodadas)

36: Gera d_wlt_frnd com $vfreq[irnd]$;

37: **para** iterações $iter = (0, \dots, niter)$ **faça** (loop iterações)

38: Zera d_g, d_illum ;

39: **para** tiros $is = (0, \dots, ns)$ **faça** (loop tiros)

40: Copia h2d 1 tiro: $d_dobs \leftarrow dobs_ALL(is)$;

41: Filtra (eq. 3.36) d_dobs com d_wlt_frnd ;

42: Define geometria dos receptores no modelo: $d_gxz(is)$;

43: **Modelagem 1: salva borda efetiva e acha resíduo**

44: **para** tempo $it = (0, \dots, nt)$ **faça** (loop tempo)

45: Insere $d_wlt_frnd(it)$ em d_sp1 na posição dada por $d_sxz(is)$;

46: Acha campo futuro; salva em d_sp0 ; (**propaga_GPU**)

47: Aplica d_taper em d_sp0, d_sp1 ;

48: $ptr \leftarrow d_sp0$; $d_sp0 \leftarrow d_sp1$; $d_sp1 \leftarrow ptr$;

49: Grava fatia em $d_dcal(it)$;

50: Grava borda efetiva $d_bndr(it)$;

51: **fim para** (fim loop tempo)

52: Elimina onda direta em d_dobs, d_dcal ;

53: Normaliza amplitudes de d_dcal aos valores de d_dobs ; (eq. 3.34);

54: Salva resíduo do tiro atual $d_derr(is) \leftarrow d_dobs - d_dcal$;

55: **Modelagens 2 e 3: gradiente (depropagação da fonte e do resíduo)**

56: **para** tempo $it = (nt, \dots, 0)$ **faça** (loop tempo)

57: **Depropagação da fonte**

58: Injeta $d_bndr(it)$ em d_sp1 ;

59: Acha campo depropagado; salva em d_sp0 ; (**propaga_GPU**)

60: Acumula iluminação d_illum ;

```

61:      Subtrai  $d\_wlt(it)$  em  $d\_sp1$  na posição dada por  $d\_sxz(is)$ ;
62:       $ptr \leftarrow d\_sp0$ ;  $d\_sp0 \leftarrow d\_sp1$ ;  $d\_sp1 \leftarrow ptr$ ;
63:      Depropagação dos receptores
64:      Injeta fatia  $d\_derr(is, it)$  em  $d\_gp1(z = 0, d\_gxz)$ ;
65:      Acha campo depropagado; salva em  $d\_gp0$ ; (propaga_GPU)
66:      Aplica  $d\_taper$  em  $d\_gp0, d\_gp1$ ;
67:       $ptr \leftarrow d\_gp0$ ;  $d\_gp0 \leftarrow d\_gp1$ ;  $d\_gp1 \leftarrow ptr$ ;
68:      Condição de imagem
69:      Acumula gradiente  $d\_g(is) \leftarrow xcorr(d\_slap, d\_gp0)$ ; (eq. 3.24)
70:      fim para (fim loop tempo)
71:      fim para; (fim loop tiros)
72:      Acumula na função objetivo  $d\_obj$ , produto interno de  $d\_derr(is)$  (eq. 3.2)
73:      Compensa  $d\_g$  com iluminação da fonte  $d\_illum$ ; (eq. 2.6)
74:      Zera fatia superficial (2 a 5 amostras) de  $d\_g$ ; (artefatos da onda direta)
75:      se ( $iter == 0$ ) Inicializa L-BFGS; fim se
76:      se ( $iter > 0$ )
77:          L-BFGS: Preenche matrizes  $d\_x, d\_y$  com  $d\_vdiff, d\_gdif$ ;
78:          L-BFGS: Calcula  $d\_r$  (Algoritmo 1); atualiza  $d\_g \leftarrow d\_r$ ;
79:      fim se
80:      Calcula  $d\_eps$ ; (eq. 3.27)
81:      Calcula  $d\_vtmp$ ; (eq. 3.26)
82:      Modelagem 4: passo  $\alpha$ 
83:      para tiros  $is = (0, \dots, ns)$  faça (loop tiros)
84:          Copia h2d 1 tiro:  $d\_dobs \leftarrow dobs\_ALL(is)$ ;
85:          Filtra (eq. 3.36)  $d\_dobs$  com  $d\_wlt\_frnd$ ;
86:          Define geometria dos receptores no modelo:  $d\_gxz(is)$ ;
87:          para tempo  $it = (0, \dots, nt)$  faça (loop tempo)
88:              Insere  $d\_wlt\_frnd(it)$  em  $d\_sp1$  na posição dada por  $d\_sxz(is)$ ;
89:              Acha campo futuro; salva em  $d\_sp0$ ; (propaga_GPU)
90:              Aplica  $d\_taper$  em  $d\_sp0, d\_sp1$ ;
91:               $ptr \leftarrow d\_sp0$ ;  $d\_sp0 \leftarrow d\_sp1$ ;  $d\_sp1 \leftarrow ptr$ ;
92:              Grava fatia em  $d\_dcal(it)$ ;
93:          fim para (fim loop tempo)
94:          Elimina onda direta em  $d\_dobs, d\_dcal$ ;
95:          Normaliza amplitudes de  $d\_dcal$  aos valores de  $d\_dobs$ ; (eq. 3.34);
96:          Acumula numerador e denominador de  $\alpha$ ; (eq. 3.22)

```

97: **fim para;** (fim *loop* tiros)
98: Calcula d_alpha ; (eq. 3.22)
99: Atualiza d_VEL , (eq. 3.23)
100: Copia d2h d_VEL atual e salva em arquivo;
101: Copia d_obj atual, salva em arquivo; (opcional)
102: Calcula erro RMS atual, salva em arquivo; (opcional)
103: **fim para** (fim *loop* iterações)
104: **fim para** (fim *loop* rodadas)
105: Desaloca todas as variáveis;

Referências Bibliográficas

- Apraez, D. R. (2015) Modelagem e migração de dados sísmicos no domínio do tempo através do operador exponencial matricial e no domínio da frequência a partir da solução da equação de Helmholtz, Dissert. de Mestrado, Universidade Federal da Bahia, Salvador, Brasil.
- Araujo, E. S. (2009) Análise dos métodos de diferenças finitas e expansão rápida na migração reversa no tempo, Dissert. de Mestrado, Universidade Federal da Bahia, Salvador, Brasil.
- Baan, M. V. D. (2008) Time varying wavelet estimation and deconvolution by kurtosis maximization, *Geophysics*, **73**(2):11–18.
- Bai, J.; Yingst, D.; Bloor, R. e Leveille, J. (2014) Viscoacoustic waveform inversion of velocity structures in the time domain, *Geophysics*, **79**:R103–R109.
- Baysal, E.; Kosloff, D. D. e Sherwood, J. W. C. (1983) Reverse time migration, *Geophysics*, **11**:1512–1524.
- Boonyasiriwat, C.; Zhan, G.; Hadwiger, M.; Srinivasan, M. e Schuster, G. (2010) Multisource reverse-time migration and full-waveform inversion on a GPGPU, 72nd EAGE Conference & Exhibition.
- Bunks, C.; Salek, F.; Zaleski, S. e Chavent, G. (1995) Multiscale seismic waveform inversion, *Geophysics*, **60**:1457–1473.
- Burden, R. L. e Faires, J. D. (2010) Numerical Analysis, CENGAGE Learning, 10th edition, Boston, MA, USA.
- Cerjan, C.; Kosloff, D.; Kosloff, R. e Reshef, M. (1985) A nonreflecting boundary condition for discrete acoustic and elastic wave equations, *Geophysics*, **50**:705–708.
- Chu, C. e Stoffa, P. L. (2011) Derivation and numerical analysis of implicit time stepping schemes, SEG Technical Program Expanded Abstracts, pp. 2987–2991.
- Chu, C. e Stoffa, P. L. (2012) Determination of finite-difference weights using scaled binomial windows, *Geophysics*, **77**:W17–W26.

- Ciarlet, P. (1989) Introduction to numerical linear algebra and optimization, Cambridge University Press, New York.
- Claerbout, J. (1971) Toward a unified theory of reflector mapping, *Geophysics*, **36**:467–481.
- Collino, F. e Tsogka, C. (2001) Application of the perfectly matched absorbing layer model to the linear elastodynamic problem in anisotropic heterogeneous media, *Geophysics*, **66**:294–307.
- Cook, S. (2012) CUDA programming: A developer's guide to parallel computing with GPUs, Applications of GPU computing, Morgan Kaufmann, 1^o edic., ISBN 0124159338,9780124159334.
- Dablain, M. A. (1986) The application of higher differencing to the scalar wave equation, *Geophysics*, **51**:54–66.
- Dussaud, E.; Symes, W.; Williamson, P.; Lemaistre, L. e Singer, P. (2008) Computational strategies for reverse-time migration, 78th Annual International Meeting, SEG, Expanded Abstracts, pp. 2267–2271.
- Gauthier, O.; Virieux, J. e Tarantola, A. (1986) Two-dimensional nonlinear inversion of seismic waveforms: Numerical results, *Geophysics*, **51**:1387–1403.
- Irons, T. (1988) Marmousi Model, 52nd eage meeting, Institut Français du Pétrole, Rueil-Malmaison, Paris.
- Kaelin, B. e Guitton, A. (2006) Imaging condition of reverse time migration, SEG Technical Program Expanded Abstracts, pp. 2594–2598.
- Kaelin, B.; Guitton, A. e Ortigosa, F. (2007) Illuminating effects in reverse time migration, 69th EAGE Conference and Exhibition incorporating.
- Kirkpatrick, S.; Gelatt, C. D. e Vecchi, M. (1983) Optimization by simulated annealing, *Science, New Series*, **220**:671–680.
- Klimm, B. (2013) Time Domain Full Waveform Inversion Using ADI Modeling, Tese de Doutorado, Kaiserslautern University.
- Kosloff, D.; Filho, A.; Tessmer, E. e Behle, A. (1989) Numerical solution of the acoustic and elastic wave equation by a new rapid expansion method, *Geophysical Prospecting*, **37**:383–394.
- Kosloff, D. D. e Baysal, E. (1982) Forward modeling by a fourier method, *Geophysics*, **47**:1402–1412.
- Lailly, P. (1983) The seismic inverse problem as a sequence of before-stack migrations, *Conference on inverse scattering: Theory and Applications*, pp. 206–220.

- Liang, W. Q.; Wang, Y. F. e Yang, C. C. (2015) Comparison of numerical dispersion in acoustic finite-difference algorithms, *Exploration Geophysics*, **46**:206–212.
- Lines, L.; Bancroft, J. e Jiang, Z. (2010) Reverse-time migration imaging with/without multiples, CREWES research report, **22**.
- Lines, L. R.; Slawinski, R. e Bording, R. P. (1999) A recipe for stability of finite-difference wave-equation computations, *Geophysics*, **64**:967–969.
- Liu, F.; Zhang, G.; Morton, S. e Leveille, P. (2008) An anti-dispersion wave equation for modeling and reverse-time migration, SEG Technical Program Expanded Abstracts, pp. 2277–2281.
- Liu, F.; Zhang, G.; Morton, S. A. e Leveille, J. (2011) An effective imaging condition for reverse-time migration using wavefield decomposition, *Geophysics*, **76**:S29–S39.
- Liu, F.; Guasch, L.; Morton, S.; Warner, M.; Umpleby, A.; Meng, Z.; Fairhead, S. e Checkles, S. (2012) 3-D time-domain full waveform inversion of a Valhall OBC dataset, SEG Las Vegas 2012 Annual Meeting - expanded abstracts.
- Loewenthal, D. e Mufti, I. (1983) Reversed time migration in spatial frequency domain, *Geophysics*, **48**:627–635.
- Manning, P. M. e Margrave, G. F. (2004) Finite-difference modelling corrections: application in a variable velocity medium, CREWES research report, **16**.
- Mao, J.; Wu, R.-S. e Wang, B. (2012) Multiscale full waveform inversion using GPU, SEG Technical Program Expanded Abstracts, pp. 1–7.
- McMechan, G. A. (1983) Migration by extrapolation of time-dependent boundary values, *Geophysical Prospecting*, **31**:413–420.
- Menke, W. (1984) *Geophysical data analysis: Discrete inverse theory*, Academic Press, Inc., San Diego, CA.
- Micikevicius, P. (2009) 3d finite difference computation on gpus using cuda, In: *Proceedings of 2nd workshop on general purpose processing on graphics processing units*, pp. 79–84, ACM.
- Moreira, R.; Bulcao, A.; Pessolani, R.; Kassuga, T.; Velandia, E. e Monteiro, F. (2011) Perfect matched layer method applied to acoustic wave modeling, *Proceedings of COBEM 2011*.
- Nocedal, J. e Wright, S. (2006) *Numerical Optimization*, Springer series in operations research and financial engineering, New York, NY.
- NVIDIA (2015) cuBLAS Library, www.nvidia.com.

- NVIDIA (2017a) cuFFT Library User's Guide, www.nvidia.com.
- NVIDIA (2017b) NVIDIA CUDA C Programming Guide, Version 9.1, www.nvidia.com.
- Oliveira, A. E. L. (2014) Migração Reversa no Tempo (RTM) com Compensação de Iluminação no Domínio de Ondas Planas e por Mínimos Quadrados (LSM) no Domínio do Tiro, Dissert. de Mestrado, Universidade Federal da Bahia, Salvador, Brasil.
- Oppenheim, A. V. e Schaffer, R. W. (2009) Discrete-Time Signal Processing, Prentice Hall Press, Upper Saddle River, NJ, USA.
- Park, Y.; Pyun, S. e Ha, W. (2014) An efficient wavenumber-space-time domain finite-difference modeling of acoustic wave equation for synthesizing cmp gathers, *Geosystem Engineering*, **17**(5):287–293.
- Pestana, R. C. e Stoffa, P. (2010) Time evolution of the wave equation using rapid expansion method, *Geophysics*, **75**(4):T121–T131.
- Pica, A.; Diet, J. e Tarantola, A. (1990) Nonlinear inversion of seismic reflection data in a laterally invariant medium, *Geophysics*, **55**:284–292.
- Plessix, R. (2006) A review of the adjoint-state method for computing the gradient of a functional with geophysical applications, *Geophysical Journal International*, **167**:495–503.
- Porsani, M. J. (2008) Uma formulação de otimização para os algoritmos gradiente e gradiente conjugado pré condicionados, Notas de aula, Universidade Federal da Bahia, Salvador, Brasil.
- Pratt, R.; Shin, C. e Hicks, G. (1998) Gauss-newton and full newton methods in frequency-space waveform inversion, *Geophysical Journal International*, **133**:341–362.
- Rubio, F.; Hanzich, M.; Farrés, A.; de la Puente, J.; Ferrer, M.; Cela, J. M. et al. (2012) Generalized elastic staggered grids on multi-gpu platforms, In: *2012 SEG Annual Meeting*, Society of Exploration Geophysicists.
- Sadahiro, M. (2014) Analysis of GPU-based convolution for acoustic wave propagation modeling with finite differences: Fortran to CUDA-C step-by-step, Master of Science in Geological Sciences, The University of Texas at Austin, Austin, TX, USA.
- Sanders, J. e Kandrot, E. (2010) CUDA by example, Addison-Wesley, Upper Saddle River, NJ.
- dos Santos, A. W. G. (2013) Inversão de forma de onda aplicada à análise de velocidades sísmicas utilizando uma abordagem multiescala, Dissert. de Mestrado, Universidade Federal da Bahia, Salvador, Brasil.

- dos Santos, A. W. G. e Pestana, R. C. (2012) A pseudo-analytical method to solve the acoustic wave equation for modeling and RTM, SEG Technical Program Expanded Abstracts, pp. 1284–1289.
- dos Santos, A. W. G. e Pestana, R. C. (2015) Time-domain multiscale full-waveform inversion using the rapid expansion method and efficient step-length estimation, *Geophysics*, **80**(4):R203–R216.
- Santos, P. N. (2017) Inversão de forma de onda com abordagem multiescala: formulação clássica e com deconvolução do resíduo, Dissert. de Mestrado, Universidade Federal da Bahia, Salvador, Brasil.
- Shin, C.; Pyun, S. e Bednar, J. (2007) Comparison of waveform inversion, part 1: conventional wavefield vs logarithmic wavefield, *Geophysical Prospecting*, **55**(4):449–464.
- Stockwell, J. e Cohen, J. K. (2008) The New SU User's Manual, version 4.0, Colorado School of Mines, Golden, CO 80401, USA.
- Symes, W. (2007) Reverse time migration with optimal checkpointing, *Geophysics*, **72**:213–221.
- Tal-Ezer, H.; Kosloff, D. e Koren, Z. (1987) Time evolution of the wave equation using rapid expansion method, *Geophysical Prospecting*, **35**:479–490.
- Tarantola, A. (1984) Inversion of seismic reflection data in the acoustic approximation, *Geophysics*, **49**:1259–1266.
- Tarantola, A. (1987) Inverse problem theory and methods for model parameter estimation, Society for Industrial and Applied Mathematics, Philadelphia, PA.
- Telford, W. M.; Geldart, L. P.; Sheriff, R. E. e Keys, D. A. (1976) *Applied Geophysics*, Cambridge Un. Press, Cambridge.
- Verschuur, E. e Gisolf, D. (2010) The principles of quantitative acoustical imaging, EAGE Publications bv, 3990 DB HOUTEN, The Netherlands.
- Virieux, J. e Operto, S. (2009) An overview of full-waveform inversion in exploration geophysics, *Geophysics*, **74**:WCC127–WCC152.
- Wang, B.; Gao, J.; Zhang, H. e Zhao, W. (2011) Cuda based acceleration of full waveform inversion on GPU, SEG Technical Program Expanded Abstracts, pp. 2528–2533.
- Wang, Y. (2015) Frequencies of the Ricker wavelet, *Geophysics*, **80**(2):A31–A37.
- Whitmore, D. W. (1983) Iterative depth migration by backward time propagation, 53rd Annual International Meeting, SEG, Expanded Abstracts, pp. 382–385.

- Wu, W. J.; Lines, L. R. e Lu, H. X. (1996) Analysis of higher-order, finite-difference schemes in 3-d reverse-time migration, *Geophysics*, **61**:845–856.
- Yang, P.; Gao, J. e Wang, B. (2007) RTM using effective boundary saving: A staggered grid GPU implementation, *Geophysics*, **72**:213–221.
- Yang, P.; Gao, J. e Wang, B. (2015) A graphics processing unit implementation of time-domain full-waveform inversion, *Geophysics*, **80(3)**:F31–F39.