Sobre a Modelagem Fletcher

Jairo Panetta

Versão 0 de 1 de fevereiro de 2018

1 Finalidade

Descrever o primeiro programa alvo do projeto UFRGS/ITA/Petrobras, a modelagem da propagação de ondas em meio anisotrópico, baseada no trabalho de Fletcher (*Robin P. Fletcher, Xiang Du and Paul J. Fowler, "Reverse time migration in tilted transversely isotropic (TTI) media", Geophysics Vol. 74, nº 6, Nov-Dec 2009*).

Descrevo o programa em C11 portátil que desenvolvi e que implementa essa modelagem.

2 Modelagem Fletcher em meio anisotrópico

No jargão da geofísica, modelagem é a simulação da propagação de ondas em um domínio ao longo do tempo. As ondas são emitidas por uma fonte, tipicamente no interior ou na borda do domínio. Também tipicamente, o domínio da simulação é um paralelepípedo tridimensional.

A modelagem simula a coleta de dados em um levantamento sísmico, como na Figura 1 abaixo. De tempos em tempos, equipamentos acoplados ao navio emitem ondas que refletem e refratam em mudanças de meio no subsolo. Eventualmente essas ondas voltam à superfície do mar, sendo coletadas por microfones específicos (geofones) acoplados a cabos rebocados pelo navio. O conjunto de sinais recebidos por cada geofone ao longo do tempo constitui um *traço* sísmico. Para cada emissão de ondas, gravam-se os traços sísmicos de todos os geofones do cabo. O navio continua trafegando e emitindo sinais ao longo do tempo.

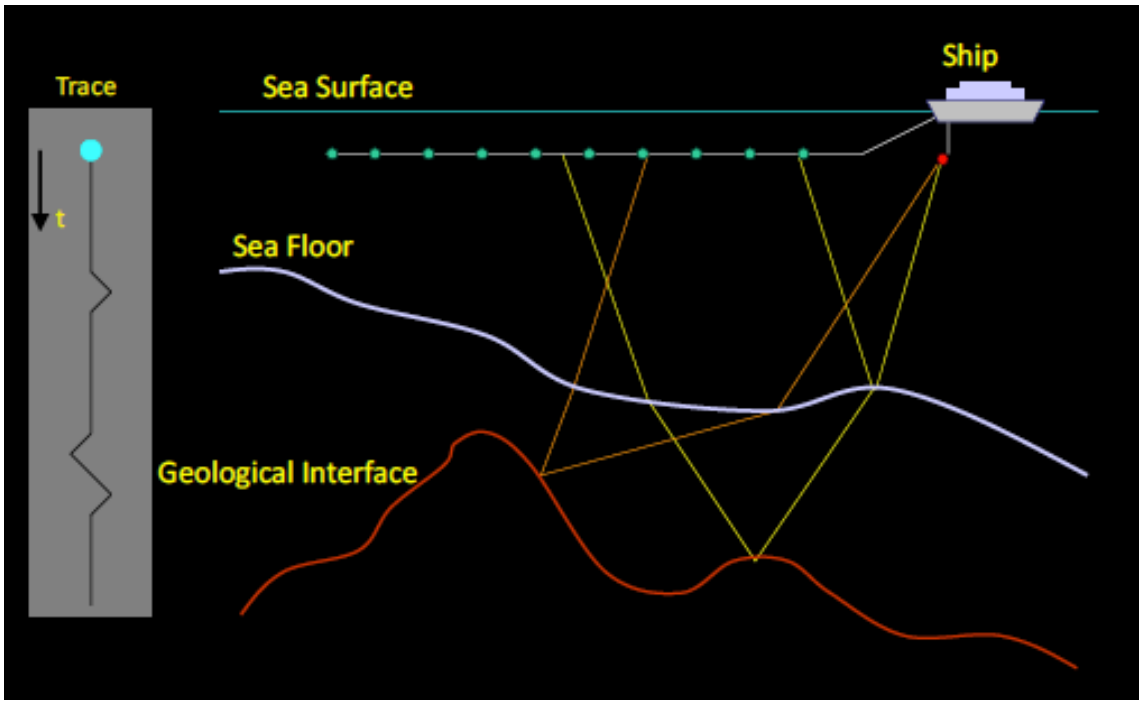

Figura 1: Coleta de dados (traços) em levantamento sísmico marítimo

Em um meio isotrópico, a propagação de ondas é governada pela equação acústica

$$\frac{\partial^2 p}{\partial t^2} = v^2 \nabla^2 p$$

onde $p(x, y, z, t)$ é a pressão em cada ponto do domínio ao longo do tempo e $v(x, y, z)$ é a velocidade de propagação da onda em cada ponto do domínio. Camadas geológicas distintas possuem velocidades distintas.

Um meio é denominado anisotrópico quando suas propriedades (como a velocidade de propagação de ondas) variam com a direção. Por exemplo, é mais fácil cortar madeira em uma direção do que em outra, pois a resistência ao corte varia com a direção.

Fletcher modelou a propagação de ondas em meio anisotrópico por equação "pseudo-acustica" *TTI* ("Tilted Transversal Isotropy"), ou seja, que trata a anisotropia como um conjunto de planos isotrópicos. Os planos são perpendiculares a um eixo de simetria, com azimute ϕ e mergulho θ conforme Figura 2 abaixo:

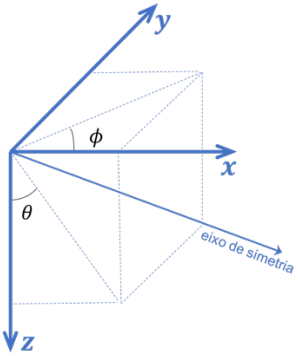

Figura 2: Eixos e ângulos na formulação de Fletcher

Caso os ângulos ϕ e θ sejam nulos, o eixo de simetria é vertical e a anisotropia é denominada *VTI* ("Vertical Transversal Isotropy").

Há dois tipos de ondas que propagam em meios anisotrópicos, denominadas ondas P e S, retratadas na Figura 3 abaixo, extraída da Wikepidia. Tais ondas propagam com velocidades distintas, representadas por v_p e v_s.

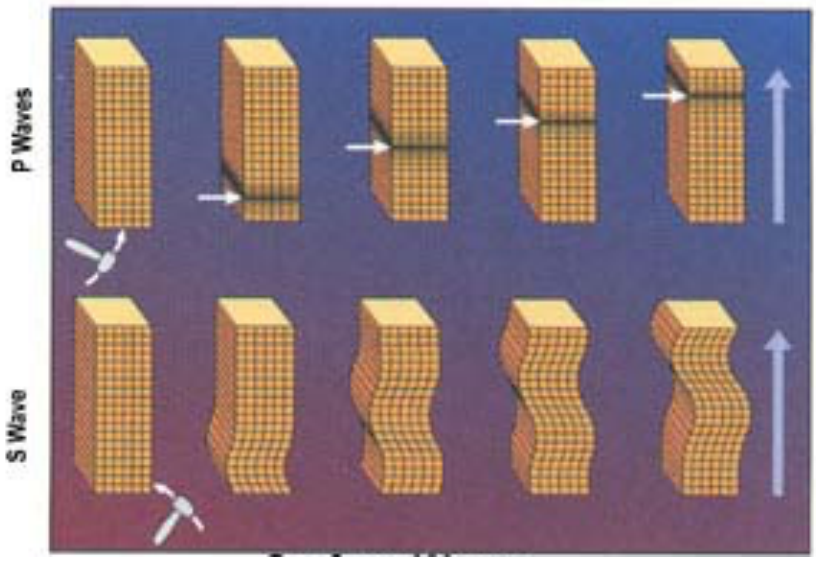

Figura 3: Propagação das ondas P e S

A anisotropia de cada material é definida pelos parâmetros $\varepsilon(x,y,z)$ e $\delta(x,y,z)$ definidos por Thomsen (*L. Thomsen, "Weak elastic anisotropy", Geophysics Vol. 51 nº 10, Oct 1986*) pela velocidade da onda P na direção normal ao plano de simetria $v_{pz}(x,y,z)$, pela velocidade da onda S na direção normal ao plano de simetria $v_{sz}(x,y,z)$ e utilizando os ângulos de azimute $\phi(x,y,z)$ e de mergulho $\theta(x,y,z)$ da Figura 2.

São dados de entrada da modelagem TTI:

$$v_{pz}(x,y,z)$$
$$v_{sz}(x,y,z)$$
$$\varepsilon(x,y,z)$$
$$\delta(x,y,z)$$
$$\phi(x,y,z)$$
$$\theta(x,y,z)$$

Fletcher introduziu as seguintes variáveis intermediárias:

$$v_{pn}(x,y,z) = v_{pz}(x,y,z)\sqrt{1+2\delta(x,y,z)}$$
$$v_{px}(x,y,z) = v_{pz}(x,y,z)\sqrt{1+2\varepsilon(x,y,z)}$$

Seja $p(x,y,z,t)$ o campo de pressão e seja $q(x,y,z,t)$ uma variável auxiliar. A modelagem TTI de Fletcher propaga pelo tempo o campo de pressão $p(x,y,z,t)$ utilizando o campo auxiliar $q(x,y,z,t)$ pelo sistema de equações

$$\frac{\partial^2 p}{\partial t^2} = v_{px}^2 H_2 p + \propto v_{pz}^2 H_1 q + v_{sz}^2 H_1 (p - \alpha q)$$
$$\frac{\partial^2 q}{\partial t^2} = \frac{v_{pn}^2}{\alpha} H_2 p + v_{pz}^2 H_1 q - v_{sz}^2 H_2 \left(\frac{1}{\alpha} p - q\right)$$

onde α é um parâmetro de acoplamento (utilizamos $\propto = 1$, como sugerido por Fletcher) e os operadores diferenciais H_1 e H_2 são definidos por:

$$H_1 = \sin^2\theta\cos^2\phi \frac{\partial^2}{\partial x^2} + \sin^2\theta\sin^2\phi \frac{\partial^2}{\partial y^2} + \cos^2\theta \frac{\partial^2}{\partial z^2} + \sin^2\theta \sin 2\phi \frac{\partial^2}{\partial x \partial y} + \sin 2\theta \sin\phi \frac{\partial^2}{\partial y \partial z} + \sin 2\theta \cos\phi \frac{\partial^2}{\partial x \partial z}$$

$$H_2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} - H_1$$

O equacionamento possui propriedades interessantes, relatadas a seguir.

No caso VTI (ou seja, $\theta = \phi = 0$), H_1 atua apenas na direção vertical pois todos os coeficientes das derivadas de H_1 são nulos exceto o coeficiente da derivada em z. Ainda mais, H_2 atua apenas na horizontal, pois a derivada em z de H_2 se anula com a derivada em z de H_1.

No caso isotrópico (ou seja, $v_{sz} = \varepsilon = \delta = \theta = \phi = 0$) as duas equações são idênticas, o que implica $p = q$ e que a equação restante é reduzida à equação acústica.

3 Discretização por diferenças finitas

Para a discretização, considere domínio 3D retangular (paralelepípedo) representado por (n_x, n_y, n_z) pontos igualmente espaçados por $(\Delta_x, \Delta_y, \Delta_z)$.

Represente a pressão $p(x, y, z, t)$ no instante t pelo array p dimensionado $[n_x, n_y, n_z]$ e por arrays de mesma dimensão p^+ e p^- nos instantes $t + \Delta t$ e $t - \Delta t$ respectivamente. Indique o elemento de p com índices i, j, k por $p_{i,j,k}$.

Proceda da mesma forma para representar o campo auxiliar q pelo array q e as velocidades $v_{px}, v_{pn}, v_{pz}, v_{sz}$ pelos arrays vpx, vpn, vpz, vsz indexados da mesma forma.

Utilizando o método das diferenças finitas, substitua as derivadas por aproximações de segunda ordem no tempo e de oitava ordem no espaço:

$$\left.\frac{\partial^2 p}{\partial t^2}\right|_{i,j,k} = \frac{1}{\Delta t^2}\left[p^+_{i,j,k} - 2p_{i,j,k} + p^-_{i,j,k}\right]$$

$$\left.\frac{\partial^2 p}{\partial x^2}\right|_{i,j,k} = \frac{1}{\Delta x^2}\left[k_0 p_{i,j,k} + \sum_{m=1}^{4} k_m\left(p_{i+m,j,k} + p_{i-m,j,k}\right)\right]$$

$$\left.\frac{\partial^2 p}{\partial y^2}\right|_{i,j,k} = \frac{1}{\Delta y^2}\left[k_0 p_{i,j,k} + \sum_{m=1}^{4} k_m\left(p_{i,j+m,k} + p_{i,j-m,k}\right)\right]$$

$$\left.\frac{\partial^2 p}{\partial z^2}\right|_{i,j,k} = \frac{1}{\Delta z^2}\left[k_0 p_{i,j,k} + \sum_{m=1}^{4} k_m\left(p_{i,j,k+m} + p_{i,j,k+m}\right)\right]$$

$$\left.\frac{\partial^2 p}{\partial x \partial y}\right|_{i,j,k} = \frac{1}{\Delta x \Delta y}\left[\sum_{m=1}^{4}\sum_{n=1}^{4} l_m l_n\left(p_{i+m,j+n,k} - p_{i+m,j-n,k} + p_{i-m,j-n,k} - p_{i-m,j+n,k}\right)\right]$$

onde $k_{0,..,4} = \left\{-\frac{205}{72}, \frac{8}{5}, -\frac{1}{5}, \frac{8}{315}, -\frac{1}{560}\right\}$ e $l_{1,..,4} = \left\{\frac{4}{5}, -\frac{1}{5}, \frac{4}{105}, -\frac{1}{280}\right\}$. As fórmulas para q são idênticas.

A Figura 4, extraída de *Raphael Fernandes Vilela, "Perfilagem do problema de resolução da equação da onda por diferenças finitas em coprocessador Xeon Phi", Dissertação de Mestrado do Programa de Pós-graduação em Engenharia Elétrica, COPPE, UFRJ, março de 2017*, mostra os estênceis das aproximações de oitava ordem por diferenças finitas das derivadas espaciais.

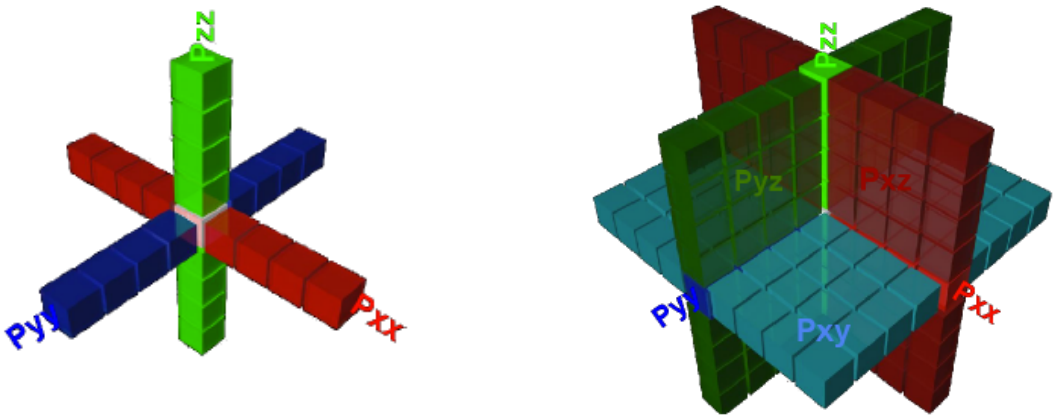

Figura 4: Estênceis das derivadas espaciais

4 Bordas da grade

Como o domínio é finito, é necessário expandir o domínio em todas as direções para que as derivadas possam ser computadas na superfície do domínio. Para tanto, basta acrescentar quatro posições (a largura do estêncil) em cada borda do domínio.

Denotando por $bord$ ($bord = 4$) a expansão do domínio em cada sentido, o domínio original (n_x, n_y, n_z) passa a ser $(bord + n_x + bord, bord + n_y + bord, bord + n_z + bord)$. Entretanto, as equações são resolvidas apenas no domínio (n_x, n_y, n_z). O campo de pressão p e o campo auxiliar q são mantidos nulos na borda.

Domínios dessa forma causam reflexão indesejável das ondas na borda, pois as condições de fronteira não são implementadas. Para evitar a reflexão, implementamos a prática de borda absortiva. Expande-se o domínio ainda mais, incluindo $absorb$ posições em cada sentido. Aplicam-se os estênceis em cada um desses pontos, resolvendo as equações no domínio $(absorb + n_x + absorb, absorb + n_y + absorb, absorb + n_z + absorb)$. Em seguida, atenuam-se os campos de pressão p e o campo auxiliar q na borda absortiva (nas $absorb$ posições em cada sentido do domínio), multiplicando os valores obtidos na borda por um fator entre 0 e 1, com valores decrescentes a medida que nos afastamos do domínio original (n_x, n_y, n_z).

Implementamos a função de absorção $f_{absor}(x, y, z) = e^{-kd^2}$ onde d é a distância normalizada a partir da borda original do domínio e k é constante arbitrária. No interior do domínio original, $f_{absor}(x, y, z) = 1.0$ e seu valor aproxima-se de zero à medida que nos afastamos do domínio original.

Implementamos a distância de duas formas. Na primeira forma, orlamos o paralelepípedo original com paralelepípedos de tamanho crescente. Na segunda forma, circunscrevemos o paralelepípedo original com esferas de raio crescente. A Figura 5 abaixo mostra os valores de f_{absor} em uma fatia x-y do domínio (ponto central em z), nessas duas implementações, utilizando escala de cinza onde preto representa o valor 1.0 e branco representa o valor 0.0. Nessas figuras, $n_x = n_y = n_z = 51$, $absor = 64$ e $bord = 4$, configurando domínio com (187,187,187) pontos.

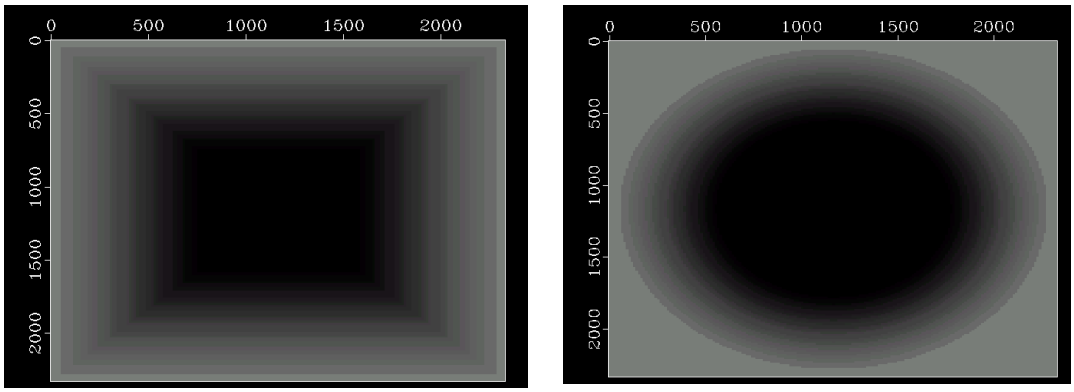

Figura 5: Bordas Absortivas

As duas formas de borda absortiva não evitaram a reflexão de ondas.

Passei a implementar outra forma, que não evita a reflexão, mas que propaga ondas aleatoriamente. Segui a proposta de *Robert G. Clapp, "Reverse time migration*

with random boundaries", 79th Annual International Meeting, SEG, Expanded Abstracts. Nessa proposta, a borda absortiva serve para inserir velocidade de propagação aleatória, produzindo sinais incoerentes. A aleatoriedade é introduzida de forma incremental, aumentando do interior do domínio para a borda.

A Figura 6 mostra o campo de velocidade da onda P na mesma posição e no mesmo domínio da Figura 5, utilizando escala de cinza onde preto representa o maior valor de velocidades e branco representa velocidade nula.

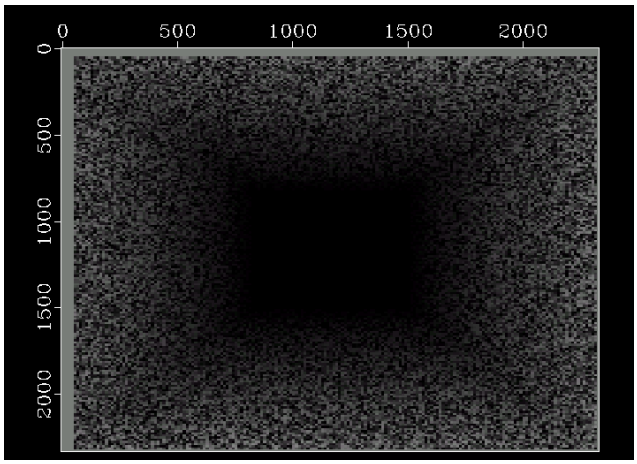

Figura 6: Velocidades aleatórias na borda absortiva

Considerei que as velocidades aleatórias apresentaram o melhor resultado.

Qualquer que seja a forma adotada para eliminar reflexões na borda, o domínio total tem dimensões $(s_x, s_y, s_z) = (2 * bord + 2 * absor + n_x, 2 * bord + 2 * absor + n_y, 2 * bord + 2 * absor + n_z)$. As equações são resolvidas no domínio $(2 * absor + n_x, 2 * absor + n_y, 2 * absor + n_z)$.

5 Condição de estabilidade

O tempo de execução do programa é proporcional à quantidade de passos no tempo, para domínio de tamanho fixo. Logo, quanto maior o passo no tempo Δt, menor o tempo de execução. Entretanto, Δt está limitado à estabilidade numérica da discretização. Fletcher define a condição de estabilidade

$$\Delta t \leq \frac{\mu \, max\{\Delta x, \Delta y, \Delta z\}}{max\{v_{px}\}}$$

onde μ é constante de estabilidade a determinar.

Embora tenha implementado esse cálculo para Δt, não trabalhei para determinar o valor apropriado de μ. O programa não utiliza o Δt calculado pela condição de estabilidade, mas utiliza o valor de Δt fornecido pelo usuário. Utilizei $\Delta t = 0{,}001s$ em todas as execuções.

6 Fonte sísmica

Utilizei a onda de Ricker, definida por $f(t) = (1 - 2g(t))e^{-g(t)}$, onde $g(t) = \pi^3 \left(\frac{w}{3\sqrt{\pi}} \left(t - \frac{2\sqrt{\pi}}{w} \right) \right)^2$ e $w = 40$hz é a frequência de corte da onda.

A Figura 7 contém a assinatura da fonte ao longo do tempo.

A fonte é localizada no ponto de coordenadas (*ixSource, iySource, izSource*), que na versão atual é posicionada no ponto central do domínio total.

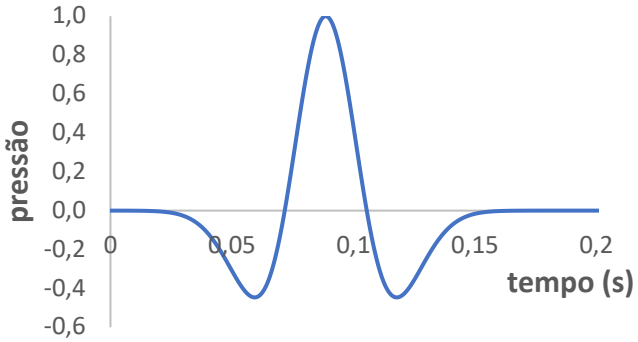

Figura 7: Assinatura da Fonte

7 Instalação do programa

O programa encontra-se no diretório */home/jairo.panetta/Modelagem/V0* da *blaise.inf.ufrgs.br*. Esse diretório possui subdiretórios *doc*, *src* e *run*. O diretório *doc* contém arquivo pdf com este documento. Fontes e Makefile em *src* e scripts de execução e visualização em *run*.

Para instalar o programa, inicie copiando os diretórios para sua área de trabalho. Compile o fonte com *make*, que utiliza o compilador *gcc* com otimização *O3* e flag *openmp*. Produz o executável *ModelagemFletcher.exe*. Copie (faça link simbólico) desse arquivo para o diretório *run*.

8 Execução

O programa espera os seguintes argumentos na linha de comando:

Argumento	**Semântica**
fNameSec (*string*)	Tipo de Anisotropia. Utilize *"ISO"* para meios isotrópicos, *"VTI"* para isotropia transversal vertical ou *"TTI"* para isotropia transversal inclinada.
nx (*int*)	Número de pontos da grade na direção *x*
ny (*int*)	Número de pontos da grade na direção *y*
nz (*int*)	Número de pontos da grade na direção *z*
absorb (*int*)	Número de pontos na borda absortiva
dx (*float*)	Distância entre pontos consecutivos na direção *x*, em metros
dy (*float*)	Distância entre pontos consecutivos na direção *y*, em metros
dz (*float*)	Distância entre pontos consecutivos na direção *z*, em metros
dt (*float*)	Passo no tempo, em segundos
tmax (*float*)	Tempo de integração, em segundos.

Usualmente os demais argumentos de entrada $(\varepsilon, \delta, v_{pz}, v_{sz}, \phi, \theta)$ são arrays tridimensionais lidos de arquivos externos. Para manter a portabilidade e evitar anexar arquivos de dados, esses valores são "hardwired" na versão atual.

O script *EXEC.sh* no diretório *run* invoca o programa com os argumentos escolhidos, imprimindo informações sobre a execução no arquivo de saída padrão. Costumo editar o script *EXEC.sh*, substituir argumentos e executá-lo. Por exemplo, o script *EXEC.sh* original emite a seguinte linha de comando:

```
time ./ModelagemFletcher.exe "VTI" 241 241 241 16 12.5 12.5 12.5 0.0010 2.0
```

que executa a modelagem Fletcher em meio VTI, com $n_x = n_y = n_z = 241$, absorção com 16 pontos, $d_x = d_y = d_z = 12.5$, $d_t = 0.001$s de $t = 0$s até $t = 2.0$s.

O programa foi levemente paralelizado utilizando diretivas OpenMP. Escolha o número de threads atribuindo valor à variável de ambiente OMP_NUM_THREADS.

Além da impressão no arquivo de saída padrão, o programa produz dois arquivos no formato RFS no diretório de execução: <fNameSec>.rfs e <fNameSec>.rfs@, contendo a evolução do campo de pressão ao longo do tempo. O primeiro é um arquivo texto contendo a localização do arquivo binário, as dimensões da grade, o número de passos no tempo e outras informações. O segundo é um arquivo binário com o campo de pressão (dimensões $s_x \times s_y \times s_z$) a cada Δ_{out} instantes de tempo, com Δ_{out} estabelecido por constante simbólica (#define) no programa principal com valor 0.01segundos.

O formato RFS é o padrão do *Madagascar* (vide http://www.ahay.org), pacote de software muito popular entre os geofísicos. Utilizei apenas os pacotes gráficos inclusos no Madagascar, embora existam outros recursos bastante úteis que possamos utilizar ao longo do tempo.

9 Visualização dos Resultados

Para visualizar os resultados é necessário utilizar os utilitários gráficos do Madagascar. No momento, este software foi instalado pelo Pedro Pais Lopes em /home/pedro.lopes/madagascar. Para colocar os executáveis do no seu $PATH, use "source /home/pedro.lopes/madagascar/share/madagascar/etc/env.sh".

O arquivo de saída (<fNameSec>.rfs@) é tetradimensional, de difícil visualização. Construí programa que acompanha a evolução temporal de uma fatia bidimensional do campo de pressão, produzindo arquivo tridimensional para ser visto como um filme. O programa, denominado *Window.exe* (fonte em /home/jairo.panetta/Window) requer os seguintes argumentos na linha de comando:

Argumento	**Semântica**
fNameInput (*string*)	Prefixo (antes do ponto) do nome do arquivo de entrada (tetradimensional). Deve ser um dentre *"ISO", "VTI"* ou *"TTI".*
ixStart (*int*)	Indice do primeiro ponto da janela na direção *x*
ixEnd (*int*)	Índice do último ponto da janela na direção *x*
iyStart (*int*)	Indice do primeiro ponto da janela na direção *y*
iyEnd (*int*)	Índice do último ponto da janela na direção *y*

izStart (*int*)	Indice do primeiro ponto da janela na direção *z*
izEnd (*int*)	Índice do último ponto da janela na direção *z*
itStart (*int*)	Indice do primeiro ponto da janela no tempo
itEnd (*int*)	Índice do último ponto da janela no tempo
fNameOutput (*string*)	Prefixo (antes do ponto) do nome do arquivo de saída.

O script original *WINDOW.sh*, no diretório *run*, facilita a execução do *Window.exe*, pois pode ser facilmente modificado. O script original produz a seguinte linha de comando:

```
Window.exe ./VTI 0 280 0 280 140 140 0 200 ./VTI_ZSLICE
```

Os scripts *EXEC.sh* e *WINDOW.sh* originais são coerentes. Como *EXEC.sh* produz arquivos VTI.rfs e VTI.rfs@ contendo 200 passos de tempo do campo de pressão na grade $(281, 281, 281)$, o script *WINDOW.sh* acompanha a fatia de domínio $(0{:}\,280,\ 0{:}\,280,\ 141{:}\,141)$ ao longo dos 200 instantes de tempo, produzindo arquivo tridimensional. Os arquivos resultantes (VTI_ZSLICE.rfs e VTI_ZSLICE.rfs@) são depositados no diretório de execução.

Fatias com X constante e com Y constante são facilmente construídas alterando o script *WINDOW.sh*.

Para visualizar o arquivo produzido por *WINDOW.sh*, utilize o script *SHOW.sh*, que requer como único argumento o nome completo do arquivo rfs a visualizar. Por exemplo, *SHOW.sh VTI_ZSLICE.rfs* produz filme que mostra a evolução temporal do plano central (em z) do campo de pressão.

10 Estrutura do programa fonte

O programa fonte é composto pelos seguintes arquivos:

Arquivo	Funcionalidade
boundary.c	Funções de absorção e de velocidade aleatória
derivatives.c	Derivadas de segunda ordem, pura e cruzada
main.c	Programa principal
map.c	Mapeamento dos índices de array 3D no índice de array 1D
source.c	Fonte sísmica
timestep.c	Aplica a formulação de Fletcher e avança no tempo
utils.c	Utilidades para output

Descrevo cada um deles brevemente.

10.1 Main

Inicia com conjunto de #define e #undef que liga ou desliga opções de execução (impressão no arquivo de saída padrão, função de absorção por paralelepípedos ou por círculos, velocidade aleatória na fronteira, etc...).

Segue lendo os dados de entrada, sem qualquer crítica. Define o tipo de problema (ISO, VTI, TTI) e calcula o tamanho do domínio total (variáveis *sx*, *sy*, *sz*) e o número de passos no tempo. Posiciona a fonte no ponto central da grade. Aloca os

arrays dos dados de entrada $(\varepsilon, \delta, v_{pz}, v_{sz}, \phi, \theta)$. Todos os arrays 3D são alocados como 1D e mapeados de 3D para 1D pelo macro *ind(ix,iy,iz)* que descrevo em *Map* (10.2). Inicializa os dados de entrada conforme o tipo de problema.

Calcula a condição de estabilidade, sem utilizá-la. Implementa velocidade aleatória na borda absortiva.

Aloca seis arrays com os coeficientes (seno e cosseno) das derivadas em H1. Opção discutível, que utiliza muita memória para reduzir a quantidade de cálculos. Adotei essa opção pois esses coeficientes independem do tempo. Pré-computando, evito recomputá-los a cada passo de tempo. Segue calculando os quatro coeficientes de H1 e H2 nas duas equações diferenciais parciais da formulação de Fletcher. Outra opção discutível. Ao adotar essas opções, utilizamos 10 arrays de *floats* com tamanho $s_x s_y s_z$ cada.

Segue alocando dois arrays (instante corrente e passado) para o campo de pressão *p* e dois arrays (mesma função) para o campo auxiliar *q*. Como veremos em Timestep (10.6), não é necessário alocar arrays para o campo futuro, pois podemos reutilizar o campo passado. Em seguida, inicializa esses arrays para zero e insire a fonte sísmica no instante inicial nos dois arrays de cada um dos dois campos.

Calcula as funções de absorção, se escolhidas. Prepara os arquivos de saída.

Segue o laço no tempo, que propaga a onda, avança o tempo trocando os arrays do instante atual para o instante futuro, insere a fonte, aplica a absorção (se selecionada) e, se for instante de saída, escreve o campo de pressão para o arquivo de saída.

No total, essa versão utiliza 20 arrays com tamanho do domínio total.

10.2 Map

O macro *ind(ix,iy,iz)* em *map.h* mapeia índices 3D em índice 1D, mas exige que as variáveis *sx*, *sy* e *sz* tenham os valores corretos (as dimensões da grade total) no ponto de uso.

10.3 Boundary

Os métodos *CreateSquareAbsorb* e *CreateSphereAbsorb* implementam bordas absortivas (retangulares e circulares, respectivamente), atribuindo valores ao array *fatAbsorb*. Esse array, com dimensão do domínio total, possui valor 1.0 dentro do domínio original e valores que tendem a zero a medida em que se afastam do domínio original. O método *AbsorbingBoundary* multiplica o campo de pressão e o campo auxiliar por esse array.

O método *RandomVelocityBoundary* gera velocidades crescentemente aleatórias na faixa de absorção do domínio total.

10.4 Derivatives

O arquivo *derivatives.h* contém constantes simbólicas dos coeficientes das aproximações de oitava ordem das derivadas segundas no espaço de uma variável e das derivadas cruzadas da variável.

Os métodos *Der2* e *DerCross* em *derivatives.c* retornam as derivadas espaciais segunda e cruzada de uma variável em um ponto do domínio total. As direções das derivadas são definidas pelo espaçamento (*stride*) entre pontos consecutivos na mesma direção no array 1D. O espaçamento é o valor da variável *s* na invocação a *Der2*. No caso da derivada cruzada, os espaçamentos são os valores das variáveis *s11* e *s21*.

10.5 Source

O arquivo *source.c* contém os métodos *Source*, que retorna o valor da fonte no instante solicitado e *InsertSource*, que insere o valor retornado por *Source* no ponto selecionado dos campos de pressão e auxiliar.

10.6 Timestep

O método *Propagate* em *timestep.c* propaga ondas um Δt no domínio total, aplicando a formulação de Fletcher. A situação inicial do domínio é definida pelos arrays *pp*, *pc*, *qp*, *qc*. Na entrada desse método, a primeira letra desses identificadores representa o campo de pressão (se *p*) ou a variável auxiliar (se *q*) e a segunda letra representa a situação anterior (se *p*) ou corrente (se *c*). Os arrays *pc* e *qc* mantem-se inalterados na saída, enquanto os arrays *pp* e *qp* contém a situação final do domínio, após a propagação.

O método TimeForward avança os campos no tempo, trocando ponteiros (entre *pp* e *pc* e entre *qp* e *qc*).

É interessante observar que é desnecessário armazenar o estado do campo em três instantes do tempo (por exemplo, p^-, p, p^+). Basta armazenar em dois instantes no tempo. Isso ocorre pois as equações em diferenças finitas podem ser representadas por

$$p^+_{i,j,k} - 2p_{i,j,k} + p^-_{i,j,k} = \Delta t\ rhs(p,q)$$

onde $rhs(p,q)$ representa o lado direito da equação, que atua apenas nos valores do instante corrente de *p* e *q* (ou seja, $p_{i,j,k}$ e $q_{i,j,k}$, para diversos valores de *i,j,k*), enquanto os valores dos instantes anterior e futuro no ponto (*i,j,k*) são referenciados uma única vez, exatamente no cálculo desse ponto. Logo, p^+ e p^- podem coincidir na memória, pois usa-se p^- unicamente para calcular p^+.

10.7 Utils

O arquivo *utils.c* contém funções para salvar em arquivos arrays tridimensionais no formato RFS. O método *DumpFieldToFile* salva o trecho 3D de índices *(ixStart:ixEnd, iyStart:iyEnd, izStart:izEnd)* do array de entrada de dimensões *(sx, sy, sz)* no arquivo *fname.rfs@* e as informações sobre as dimensões do trecho do array salvo no arquivo *fname.rfs*.

Esse método não permite concatenar valores do mesmo trecho de um array ao longo do tempo em um único arquivo. Para isso, existe a estrutura de dados *Slice* definida em *utils.h* e os métodos que implementam essa funcionalidade.

O método *DumpSlicePtr* imprime informações sobre a estrutura *Slice* no arquivo de saída padrão, para depuração.

O método *OpenSliceFile* abre os arquivos binário e texto, cria uma estrutura *Slice*, armazenando nessa estrutura o trecho do array que se deseja salvar e o prefixo do nome dos arquivos de saída. Retorna ponteiro para a estrutura *Slice* criada.

O método *DumpSliceFile* apenda o estado atual do array no arquivo binário e contabiliza quantos instantes de tempo foram apendados.

O método *CloseSliceFile* fecha o arquivo binário e produz o arquivo texto.

A estrutura de dados e os métodos foram preparadas para trabalhar em fatias de X constante, Y constante, Z constante ou em subdomínios retangulares do domínio total. Produz arquivos no diretório corrente.