Comparativo entre um algoritmo paralelo e sequencial para modelagem da equação da onda acústica em duas dimensões

Igor Morgado

7 de julho de 2018

Resumo

Implementar a modelagem de diferenças finitas para a equação da onda acústica em um meio isotrópico uniforme em duas dimensãos e comparar uma implementação sequencial e paralela.

1 Introdução

O método das diferenças finitas[10] é um dos métodos mais utilizados para a solução numérica de EDPs, principalmente pela simplicidade e custo computacional. Partiremos de um ponto bem estabelecido que é a modelagem da equação da onda acústica 2D em meios homogêneos com condições inciais e de contorno, como as de *Dirchlet*, *Neumman*[18], *Reynolds*[14] e *Cerjan*[5] utilizado o método das diferenças finitas central de quarta ordem no espaço e para frente de segunda ordem no tempo. Apresentaremos brevemente o modelo físico-matemático e sua discretização pela diferença finita, nos focaremos especialmente no algoritmo e a implementação computacional sequencial e paralela.

2 Desenvolvimento

2.1 Equação da onda

A equação da onda inicialmente modelada por *d'Alambert* no século XVII (CITACAO) com uma força impulsiva externa é representada como

$$\frac{\partial^2 u}{\partial t^2} - c^2 \Delta u = f(\vec{x}, t)\delta(\vec{x}, t) \tag{1}$$

com condições iniciais

$$\begin{cases} u(\vec{x}, t) = 0 & t = 0 \\ \frac{\partial u}{\partial t} = 0 & t = 0 \end{cases} \tag{2}$$

onde u é o campo de pressão, $\vec{x}$ é o vetor posição n-dimensional, t a variável temporal, c a velocidade da onda sobre o meio físico de propagação, f uma força externa, Δ o operador laplaciano representando a segunda derivada espacial e δ a função impulso de *Dirac*. Estas condições iniciais representam a ausência de qualquer energia inicial, esta será adicionada ao problema através da função f.

Como pretendemos modelar um domínio semi-infinito, ou seja a onda propagada pela fonte encontra uma região praticamente ilimitada utilizando um método numérico que é intrinsicamente finito, devemos criar condições de contorno artificiais que representem esta característica. Ao utilizar estas condições de contorno artificiais, desejamos que a onda não gere qualquer reflexão ao atingir as extremidades do modelo discretizado. As condições de contorno usuais como de Dirichlet e Neumann dadas por

$$\begin{cases} u(\partial\Omega, t) = 0 & \forall t \\ \nabla u(\partial\Omega, t) = 0 & \forall t \end{cases} \tag{3}$$

geram uma reflexão com fase deslocada em π rad, o que seria um efeito indesejado, para evitar tal comportamento utilizaremos as condições de contorno de Reynolds[14] que cria uma condição de contorno não reflexiva dada por

$$\Delta u - \frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} = \left(\frac{d}{dx} + \frac{1}{c}\frac{\partial}{\partial t}\right)\left(\frac{d}{dx} - \frac{1}{c}\frac{\partial}{\partial t}\right) u = 0 \tag{4}$$

e de Cerjan [5] que cria uma condição de contorno de absorção aplicando um coeficiente de atenuação $w(d)$ aos pontos da malha próximos suficiente da borda dado por

$$w(d) = e^{-(\beta(n-d))^2} \tag{5}$$

com d sendo a distância máxima de atenuação da borda, n a distância sendo calculada e β o fator de atenuação.

2.2 Discretização do Domínio

A discretização do domínio continuo é feita tomando-se pontos regularmente espaçados sobre uma malha discreta com $x_N \times z_N$ pontos distantes entre si por Δx e Δz (veja figura 1. Onde x_0 e x_N são as coordenadas iniciais e finais no eixo x e z_0 e z_N os respectivos para o eixo z. $\partial\Omega$ é a fronteira do espaço Ω.

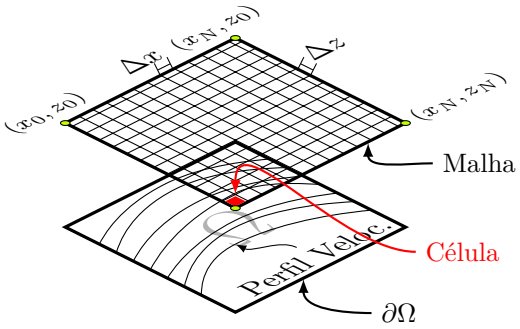


Figura 1: Domínio contínuo e malha discreta

2.3 Termo fonte

A equação da onda descreve o fenomedo da propagação da energia sobre um meio físico com uma velocidade c, normalmente esta onda é dada como o valor inicial para um determinado campo de pressão.

Ao invés disso intejaremos um pulso artificial em um ponto da malha qualquer, este pulso então será propagado conforme a simulação prossegue a este pulso chamaremos de *fonte*.

Escolheremos uma função chamada ondaleta Ricker [15] (veja figura 2), que é obtida da segunda derivada da função gaussiana vista na equação (6).

$$f(t) = \left[1 - 2\pi\left(\pi f t\right)^2\right] e^{-\pi(\pi f t)^2} \tag{6}$$

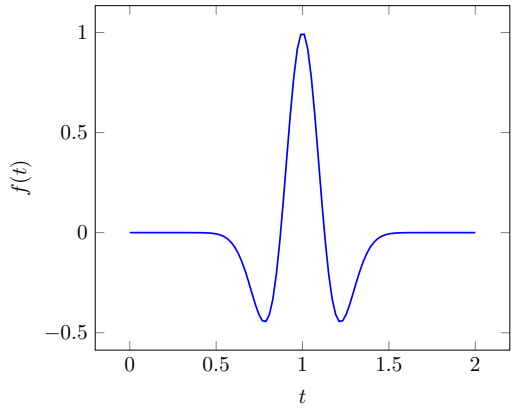


Figura 2: Ondaleta ricker

Onde f é a frequência central da onda, e t é um passo do tempo. A função Ricker é simetrica com relação ao eixo vertical (função par), que gera um problema em $t = 0$, pois $f(0) = 1$, gerando uma descontinuidade, para mitigar este efeito é necessário que a onda seja deslocada no tempo e gradualmente atinja o seu valor de pico, este deslocamento é calculado a partir da equação

$$t_d = t - \frac{2\sqrt{\pi}}{f_{corte}} \tag{7}$$

e a frequência de corte é obtida por

$$f_{corte} = 3\sqrt{\pi} f \tag{8}$$

2.4 Operador de diferenças finitas

A discretização do operador de derivada é feita tomando-se a série de Taylor em um ponto e no grupo de pontos vizinhos, quanto maior o número de pontos melhor a aproximação númerica do operador, para maiores detalhes veja [16]. Aqui utilizaremos diretamente os operadores da segunda derivada de segunda ordem

$$\frac{\partial^2 u}{\partial x^2} = \frac{+u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{\Delta x^2} + O(\Delta x^2) \tag{9}$$

e de quarta ordem

$$\frac{\partial^2 u}{\partial x^2} = \frac{-u_{i-2,j} + 16u_{i-1,j} - 30u_{i,j} + 16u_{i+1,j} - u_{i+2,j}}{12\Delta x^2} + O(\Delta x^4) \quad (10)$$

Aplicando a diferenca finita de quarta ordem central no laplaciano e a de segunda ordem na derivada temporal na equação da onda temos:

$$\frac{u_{i,j}^{n-1} - 2u_{i,j}^{n} + u_{i,j}^{n+1}}{\Delta t^2} - c^2 \left(\frac{-u_{i-2,j} + 16u_{i-1,j} - 30u_{i,j} + 16u_{i+1,j} - u_{i+2,j}}{12\Delta x^2} + \frac{-u_{i,j-2} + 16u_{i,j-1} - 30u_{i,j} + 16u_{i,j+1} - u_{i,j+2}}{12\Delta z^2} \right) = f(\vec{x}, t) \quad (11)$$

Fazendo $h = \Delta x = \Delta z$ e isolando o termo em $n+1$. Temos:

$$u_{i,j}^{n+1} = 2u_{i,j}^{n} - u_{i,j}^{n-1} + \frac{\Delta t^2}{12h^2} c^2 (u_{i-2,j} - 16u_{i-1,j} + 30u_{i,j} - 16u_{i+1,j} + u_{i+2,j} + u_{i,j-2} - 16u_{i,j-1} + 30u_{i,j} - 16u_{i,j+1} + u_{i,j+2}) + f(\vec{x}, t) \quad (12)$$

Que representa o passo temporal da próxima iteração $n+1$. Ou seja para obtermos um valor na malha em um ponto qualquer (i, j) no passo seguinte precisamos de quatro vizinhos espacialmente afastados do ponto (i, j) sobre a malha discreta e os dois pontos anteriores no passo do tempo sobre o mesmo ponto. Chamamos esta representação gráfica de esténcil do operador de diferenças finitas (que pode ser visto na figura 3).

2.5 Critério de convergência

Utilizamos então a aproximação por diferenças finitas no lugar das derivadas contínuas, entretando esta substituição deve ser realizada com critérios bem definidos.

Desejamos que o resultado da diferença finita esteja sempre próximo do resultado analítico isto é

$$|u - \tilde{u}| = 0 \text{ quando } \Delta t \to 0 \text{ e } h \to 0 \quad (13)$$

onde $\tilde{u}$ é a solução exata. Também é desejado que

$$|u - \bar{u}| < k \text{ para } \Delta t \to \infty \text{ e } k \in \mathbb{R} \quad (14)$$

onde $\bar{u}$ é a solução numérica computada. Isto significa que dado um número qualquer de iterações a

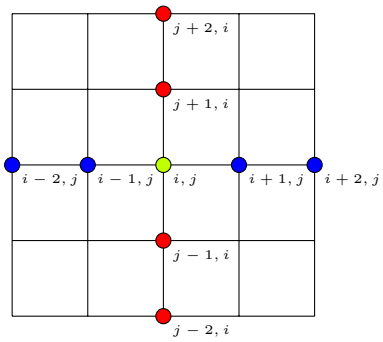


Figura 3: Estencil do laplaciano de quarta ordem. O ponto verde é o ponto sendo calculado, os azuis a derivada no eixo x e em vermelho os pontos da derivada parcial no eixo z.

diferença (erro) entre o resultado numérico esperado e o resultado numérico computado (erros de arredondamento, artefatos numéricos, etc) esteja sempre limitado superiormente.

O primeiro critério é chamado de criério de consistência e o segundo de critério de estabilidade. O *Teorema de equivalência de Lax-Richtmyer* prova que se um sistema atende estes dois critérios ele é automaticamente convergente (veja em [16]), isto é a solução numérica se aproxima da solução analítica, como desejamos.

Foi demonstrado em [1] e [17] que para manter o controle sobre a dispersão numérica basta que a condição

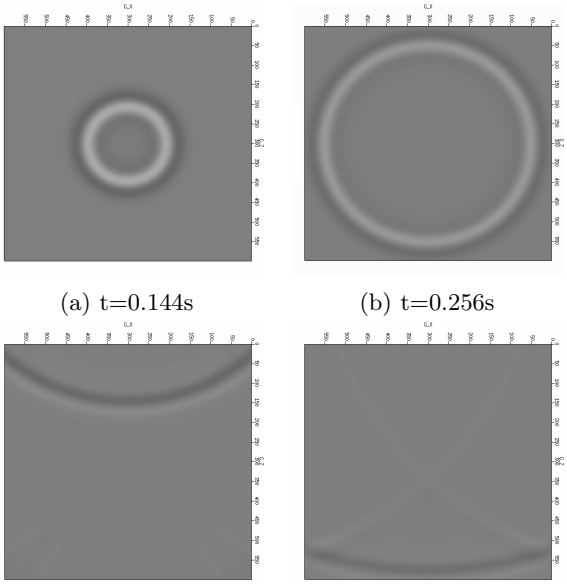
$$h \leq \frac{v_{min}}{\alpha f_{corte}} \quad (15)$$

seja satisfeita, onde v_{min} é a velocidade mínima no modelo de velocidade e f_{corte} é a frequência de corte do termo fonte e α é um paramêtro que varia com a ordem utilizada nos operadores diferenciais, para termos de quarta ordem $\alpha = 5$ é suficiente.

Continuando, precisamos também que o modelo seja estável, isto é os erros entre cada passo esteja limitado, CFL[10] estabelece a relação dada por

$$\Delta t \leq \frac{h}{\beta v_{max}} \quad (16)$$

onde v_{max} é a velocidade máxima do modelo de velocidade e β é uma constante de estabilidade, esta desigualdade é chamada de *critério CFL*. Bulcão [4] em sua tese de doutorado mostra que se a condição seja satisfeita, basta tomarmos $\beta = 4$ para um elevado grau de precisão.



(a) t=0.144s (b) t=0.256s

(c) t=0.376s (d) t=0.664s

Figura 4: Campo de pressão discretizado com 600 pontos nos eixos X e Z. O espaçamento da malha é de 1 metro, propagado sobre água (velocidade de 1500m/s).

2.6 Algoritmo Sequencial

A computação da malha de diferenças finitas possui um algoritmo bem simples que envolve alguns passos

Algorithm 1 Laço principal

```
1: Le modelo de velocidade
2: Verifica estabilidade
3: Calcula coeficientes do Laplaciano
4: for all PASSOS_SIMULACAO do
5:     Injeta fonte
6:     Aplica LAPLACIANO
7:     Aplica Condicoes de contorno
8:     Escreve Campo de Pressão
9: end for
```

As operações definidas em no algoritimo 1 estão consolidadas mas podem ser visualizadas integralmente no repositório online [12], por outro lado procedimento mais custoso computacionalmente foi adicionado neste artigo para referência e pode ser visto na listagem 2.

2.7 Algoritmo Paralelo

Com o objetivo de ser capaz de tratar um volume de dados maior do que a capacidade de memória e executar a computação do Laplaciano em múltiplos processadores simultaneamente baseando-se nos trabalhos de [6] implementamos um algorítimo que decompõe o domínio em múltiplos subdomínios e distribui as tarefas em um cluster de computadores em rede através do modelo de memória distribuída utilizando para comunicação o padrão *MPI*[7].

Em uma análise de perfilagem de software com a ferramenta Valgrind [13] foi identificado que o procedimento `LAPLACIANO()`(veja o algoritmo 2) toma $99,6\%$ do tempo do processador.

Algorithm 2 Procedimento de cálculo do Laplaciano. Aqui X e Z são os índices dos eixos, F é o campo de pressão discretizado, V é o modelo de velocidades discretizado e C os coeficientes do operador de diferenças finitas pré calculado. Este procedimento é aplicado somente aos pontos interiores, mas o primeiro laço FOR não explicita isso para simplificar a notação

1: **procedure** LAPLACIANO(F, V, C)
2: **for all** z, x in Z, X **do**
3: $lap_x \leftarrow \frac{C_0}{dx^2} * F_{x,z}$
4: $lap_z \leftarrow \frac{C_0}{dz^2} * F_{x,z}$
5: **for all** c in C **do**
6: $lap_x \leftarrow lap_x + \frac{C_c}{dx^2} (F_{x-c,z} + F_{x+c,z})$
7: $lap_z \leftarrow lap_z + \frac{C_c}{dz^2} (F_{x,z-c} + F_{x,z+c})$
8: **end for**
9: **end for**
10: $F^{n+1}_{x,z} = 2F^{n-1}_{x,z} - F^n_{x,z} + V_{x,z} * (lap_x + lap_z)$
11: **end procedure**

O objetivo é paralelizar o algoritmo 2, pois pela lei de Amdahl[2], o benefício obtido pela otimização é proporcional ao custo de computação do procedimento sendo otimizado.

2.8 Decomposição de domínio

Dado ao forte acoplamento das operações envolvidas no cálculo do Laplaciano, optamos por realizar a divisão do problema em volumes menores sendo G_X, G_Z as dimensões da malha total (G de grid, malha em inglês), através de uma decomposição do domínio, as posições destes sub domínios é então distribuída entre os computadores envolvidos. A figura 5 mostra o domínio original dividido em 9 sub-domínios a serem designadas para 9 computadores diferentes.

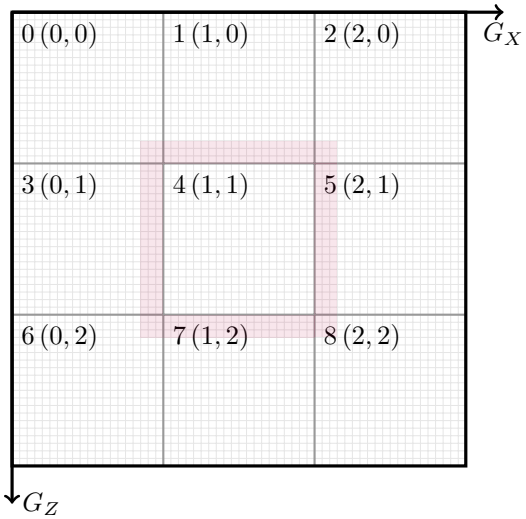


Figura 5: Dominio decomposto em sub-domínios (9 no total). No canto superior esquerdo de cada domínios esta representado o identificador de domínio ($\#D$), bem como as coordenadas do sub domínio ($D_{x,z}$). A área delimitada em vermelho mostra a borda fantasma sobre os subdomínios vizinhos que devem ser trocados em cada passo de iteração, o tamanho é referente a diferença central no espaço de ordem 4.

Cada computador de posse de seu domínio local realiza então o cálculo do laplaciano localmente e os valores calculados das zonas fantasma devem ser repassados para os vizinhos de interesse de forma que possam usalos para a computação dos passos seguintes. O tamanho da zona fantasma é proporcional a ordem da discretização, pois deve ser capaz de calcular com precisão total até a borda do domínio.

De forma simplificada o procedimento para decomposição do domínio é dada da seguinte forma:

1. Número de nós de processamento (np) a serem utilizados é dado arbitrariamente, normalmente este parametro é passado pelo usuário em tempo de execução;
2. A divisão sub-domínios em cada dimensão (neste trabalho utilizamos duas dimensões) é dado pelo princípio da menor interface (definido em [6]). Uma vez estabelecido temos duas novas constantes D_X e D_Z (repare na letra maiúscula no índice), que é o numero de subdomínios na direção X e Z respectivamente. Alem disso a desigualdade

$$D_X \cdot D_Z \leq np \tag{17}$$

é sempre satisfeita. Pois cada núcleo de processamento devera ficar somente com um subdomínio, os nós excedentes não sao utilizados neste algorítimo.

3. Cada subdomínio então possui um *identificador de Domínio* notado como $\#D$ e uma coordenada de domínio que pode ser obtida de forma biunívoca como

$$\#D = D_x + D_z * D_X, \tag{18}$$

onde D_x e D_z é a posição do subdomínio (repare na letra minúscula no íncide) na malha de subdomínios mostrada na figura 5. O *worker id* do MPI é usado como o identificador de Domínio e baseado nele é possivel obter as coordenadas de domínio $D_{x,z}$ individuais (veja algoritmo 3).

4. Cada subdominio então define localmente e de forma deterministica em que ponto da malha original(G) sua região local (g) estará limitada, isto é encontra as coordenadas de G que serão referenciadas dentro da malha de g, que será tratada localmente. O algoritimo 3 trata da limitação desta região.
5. Verifica se o subdomínio possui fontes, se possuir converte as coordenadas globais $G_{x,z}$ da fonta para as coordenadas locais $g_{x,z}$.

A partir deste ponto o mesmo algoritmo sequencial é executado, a diferença é que cada subdomínio trata seu domínio local como sendo todo o espaço.

No final de cada passo da simulação os valores que das bordas são passados para os domínios vizinhos para que o próximo passo da simulação possa ser realizado.

De uma forma mais geral o algoritmo paralelo pode ser visto na listagem 4.

3 Metodologia

A implementação do código tanto sequencial quanto paralelo foi baseada no padrão ISO C 2011[9], utilizando o compilador GCC versão 4.7.2.0 e a biblioteca C GNU/libc6 versão 2.27[11] no sistema operacional Debian GNU/Linux Buster[3]. Para as partes paralelas foi utilizado o padrão MPI[7], especificamente a implementação *OpenMPI* versão 3.1.0[8].

O computador que executou os testes é um $SurfaceBook$ com $8GB$ de RAM, e 1 Processador *Intel(R) Core(TM) i5-6300U CPU @ 3GHz* com 4 cores e 3MB de cache, disco SSD *Samsung Electronics Co Ltd NVMe SSD Controller SM951/PM951*, interface rede USB *Marvell Gigabit Ethernet*.

Para compilação foram utilizados o parametro de otimização `-O3` do GCC.

Não foi possível executar testes de rede e com distribuição de memória por limitações de recursos.

Algorithm 3 Encontra os limites do domínio local. G são os parametros da malha original, D os parametros da malha de subdominio, L os parametros do operador laplaciano, MOD e DIV são operadores de módulo e divisão inteira respectivamente. D_x, D_z são as coordenadas do subdomínio. S_x, S_z número de celulas em cada subdomínio nas respectivas direções, $S_{x0}, S_{z0}, S_{xn}, S_{zn}$ é a posição inicial e final da malha do subdominio $g(\#D)$ referente a malha original G e L_o é o tamanho das bordas do estencil.

procedure DecompoeDominio(G, D, L)
  $D_x \leftarrow \text{MOD } (\#D, D_X)$
  $D_z \leftarrow \text{DIV } (\#D, D_X)$
  $S_x \leftarrow \text{DIV } (G_X, D_X)$
  $S_z \leftarrow \text{DIV } (G_Z, D_Z)$
  $R_x \leftarrow \text{MOD } (G_X, D_X)$
  $R_z \leftarrow \text{MOD } (G_Z, D_Z)$
  $S_{x0} \leftarrow D_x * S_x$
  $S_{z0} \leftarrow D_z * S_z$
  if $D_x < R_x$ **then**
    $S_x \leftarrow S_x + 1$
    $S_{x0} \leftarrow S_{x0} + D_x$
  else
    $S_{x0} \leftarrow S_{x0} + R_x$
  end if
  if $D_z < R_z$ **then**
    $S_z \leftarrow S_z + 1$
    $S_{z0} \leftarrow S_{z0} + D_z$
  else
    $S_{z0} \leftarrow S_{z0} + R_z$
  end if
  $S_{xn} \leftarrow S_{x0} + S_x$
  $S_{zn} \leftarrow S_{z0} + S_z$
  if $D_x > 0$ **then**
    $S_{x0} \leftarrow S_{x0} - L_o$
  end if
  if $D_x < D_X - 1$ **then**
    $S_{xn} \leftarrow S_{xn} + L_o$
  end if
  if $D_z > 0$ **then**
    $S_{z0} \leftarrow S_{z0} - L_o$
    mais topo
  end if
  if $D_z < D_Z - 1$ **then**
    $S_{zn} \leftarrow S_{zn} + L_o$
  end if
end procedure

Algorithm 4 Laço principal paralelo

Carrega Parametros de Simulacao
Calcula coeficientes do Laplaciano
INICIA Paralelismo
DecompoeDominio()
if $\#D$ tem Fonte **then**
  Carrega fonte
end if
Carrega Modelo de Velocidade Local
Verifica Estabilidade
for all $PASSOS\_SIMULACAO$ **do**
  Injeta fonte
  Aplica LAPLACIANO
  Aplica Condicoes de contorno
  Escreve Campo de Pressão
end for
FIM Paralelismo
Concatena malhas locais e salva em disco

Run	Tempo(s)	Desvio
1	16.440	-0.0446
2	16.394	-0.0906
3	16.232	2.065
4	16.437	-3.156
5	16.920	-0.634
M	16.484	0

Tabela 1: Tempo de execução do algoritmo sequencial e média

O modelo de velocidades e campo de pressão foram discretizados em uma malha de $600x600$ com espaçamento de $1metro$ e foi executado 1 segundo de simulação. O intervalo de tempo foi o intervalo ótimo calculado pela condição de estabilidade dada pela equação (16), sendo salvo em disco 1 amostra a cada 0.008 segundos, totalizando 125 amostras.

4 Resultados

Realizados testes os resultados obtidos podem ser visualizados na tabela 1 para o algoritmo sequencial e na tabela 2 para a paralela.

O *Speedup* médio obtido foi 0.030.

Por outro lado não foi encontrada diferença no campo de pressão entre a algorítmo sequencial e paralelo.

Run	Tempo(s)	Desvio
1	549.208	6.181
2	543.748	0.721
3	551.288	8.261
4	530.400	-12.626
5	540.488	-2.538
M	543.026	0

Tabela 2: Tempo de execução total do algoritmo paralelo com 4 processos e média

5 Conclusão

O tempo de processamento paralelo foi inaceitável, existe algum problema que especula-se ser no tamanho muito pequeno do modelo não compensando o tempo gasto em rede para troca de mensagens pequenas ou na ineficiencia do atual algoritimo implementado ficando muito abaixo dos resultados apresentados em [6].

Por outro lado o algorítmo é numericamente coerente apresentando resultados equivalentes tanto no modelo sequencial como no paralelo.

6 Trabalhos futuros

Revisitar este trabalho é essencial, visto que não foi alcançada a performance já encontrada no artigo de referência para tal iremos proceder com os seguintes pontos:

- Reescrever algorítimo e o código em C para que fique mais fácil para ser inspecionado;
- Realizar mudancas na estrutura de dados para minimizar a busca em memória;
- Realizar testes com volumes de tamanhos diferentes;
- Realizar testes com estruturas de redes diferentes;
- Escrever um conjunto de ferramentas para coletar dados durante os testes como memória, CPU, IO, Rede para que seja possível fazer gráficos descrevendo melhor o comportamento do algoritmo;
- Implementar concomitantemente uma solução local para memória compartilhada, de forma que o MPI seja usado em máquinas remotas e o OMP localmente, ganhando o benefício do Cache Local.

Referências

[1] R. M. Alford, K. R. Kelly, and D. M. Boore. Accuracy of finite-difference modeling of the acoutic wave equation. *Geophysics*, 39(6):834–842, 1974.

[2] G. M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), pages 483–485, New York, NY, USA, 1967. ACM.

[3] O. Aoki. Debian reference, 2018.

[4] A. Bulcao. *Modelagem e migração reversa no tempo empregando operadores elásticos e acústicos*. PhD thesis, 2004.

[5] C. Cerjan, D. Kosloff, R. Kosloff, and M. Reshef. A nonreflecting boundary condition for discrete acoustic and elastic wave equations. *Geophysics*, 50(4):705–708, 1985.

[6] M. A. de Medeiros Lara. *Paralelização de um algoritimo de propagação da onda acústica 2d usando MPI*. PhD thesis, COPPE/UFRJ, 2012.

[7] M. P. Forum. Mpi: A message-passing interface standard. Technical report, Knoxville, TN, USA, 1994.

[8] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall. Open MPI: Goals, concept, and design of a next generation MPI implementation. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, pages 97–104, Budapest, Hungary, September 2004.

[9] ISO. *ISO/IEC 9899:2011 Information technology — Programming languages — C*. International Organization for Standardization, Geneva, Switzerland, December 2011.

[10] H. Lewy, K. Friedrichs, and R. Courant. Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen*, 100:32–74, 1928.

[11] S. Loosemore, R. M. Stallman, and A. Oram. The gnu c library reference manual for version 2.27, 2018.

[12] I. Morgado. Biblioteca de diferenças finitas para equação da onda, 2018.

[13] N. Nethercote and J. Seward. Valgrind: A framework for heavyweight dynamic binary instrumentation. In *Proceedings of the 28th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '07, pages 89–100, New York, NY, USA, 2007. ACM.

[14] A. C. Reynolds. Boundary conditions for the numerical solution of wave propagation problems. *Geophysics*, 43:1099–1110, 1978.

[15] N. Ricker. The form and nature of seismic waves and the structure of seismograms. *Geophysics*, 4:348–366, 1940.

[16] J. C. Strikwerda. *Finite Difference Schemes and Partial Differnetial Equations*. SIAM, 2 edition, 2004.

[17] J. Virieux. P-SV wave propagation in heterogeneous media velocity stress finite difference method. *Geophysics*, 51(4):886–901, 1986.

[18] K. S. Yee. Numerical solution of initial boundary value problems involving maxwell equations in isotropic media. *IEEE Transactions on Antennas and propagation*, 14(3):302–307, 1966.