CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA
DE MINAS GERAIS

ENGENHARIA DE COMPUTAÇÃO

Paralelização de Métodos Numéricos para Resolver Equações Diferenciais Parciais

Orientando:
Marcelo Lopes de Macedo
FERREIRA CÂNDIDO

Orientador:
Prof. Dr. Luis Alberto
D'AFONSECA

BELO HORIZONTE
19 de fevereiro de 2019

Sumário

Capítulo 1

Introdução

Capítulo 2

Aquisições Sísmicas

introduzir

2.1 O Que São Aquisições Sísmicas e Como Modelá-las

No ramo da mineração não se pode tentar a esmo a descoberta de recursos minerais no subterrâneo de um local em que se já se suspeita sua existência. É necessário que, de alguma forma, se obtenha a forma dessa estrutura oculta para se saber os pontos onde se encontram as jazidas/poços desse recurso. A obtenção dos dados de como é essa estrutura se chama **aquisição sísmica**.

A forma de se realizar essa aquisição pode variar com o ambiente e os métodos adotados para coleta de dados e processamento dos mesmos. Os recursos minerais desejados podem se encontrar tanto em meios terrestres e/ou subaquáticos. Contudo, o meio em que a aquisição será realizada pouco importa nesse trabalho, o que será melhor explicado mais a frente.

Para a coleta dos dados a serem processados podemos citar dois exemplos de aquisição

buscar referências

1. **marítima**: um navio equipado com um canhão sonoro emite ondas sonoras cujas reflexões e refrações nas camadas terrestres submarinas são captadas por filas de hidrofones puxadas pelo mesmo navio.

2. **terrestre**: um explosivo é colocado (preferencialmente enterrado) em um terreno. Sua explosão gera uma onda sonora cujas reflexões são captadas por geofones distribuídos relativamente próximos, na superfície.

Costuma-se alterar a posição da fonte sonora na realização da aquisição para facilitar a modelagem do meio de propagação das ondas.

conferir se procede

Esse colhimento dos dados consistirá em traços, que são o gráfico das amplitudes das ondas sonoras captadas pelos hidrofones/geofones,como se pode ver na Figura **??**. A partir desses traços, detecta-se, por análise técnica, onde se encontram as interfaces entre as camadas do domínio analisado e do que são feitas essas. Nisso consiste o processamento dos dados colhidos.

Colocar a imagem dos traços

Vale lembrar que, quanto aos métodos de processamento utilizados, nesse trabalho simularemos um problema direto, ou seja, estipulando o meio (nesse caso, não-homogêneo) da aquisição, veremos como as ondas se propagam nele. Para tal, usaremos um método matemático específico. Nesse trabalho, como já foi dito no Capítulo 1, é o de Diferenças Finitas.

colocar isso

2.2 A Equação da Onda

Para que seja possível avaliar matematicamente um fenômeno ondulatório produzido por uma fonte em um domínio é necessário se ter uma fórmula matemática para o que se entende por onda. Tal fórmula, que nos servirá durante todo esse trabalho, principalmente na parte do código é a **equação da onda**, dada por

$$\frac{\partial^2 u}{\partial t^2} = \frac{1}{v^2}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + f(x,y,t) \tag{2.1}$$

onde x e y são variáveis espaciais e t, temporal. A constante (no caso desse trabalho) v representa a velocidade da frente de onda. Trata-se de uma **equação diferencial parcial hiperbólica** com solução analítica, mas que pode ser resolvida de forma fácil numericamente, utilizando, por exemplo, o **método de diferenças finitas**, a ser mais explicado na próxima seção.

Caso o leitor se interesse por estudar ou revisar mais sobre Ondulatória, pode conferir em Cândido [Mac18].

2.3 O Método de Diferenças Finitas (MDF)

Uma equação diferencial parcial, que geralmente é considerada em um domínio contínuo, pode ser discretizada. Isso é feito para que a equação possa ser representada e resolvida computacionalmente.

No caso do método de diferenças finitas para esse trabalho, basta transcrever cada termo da equação para o equivalente na fórmula de diferenças finitas para derivações de segundo grau, sobre a qual podemos ver um exemplo na Tabela 2.1 para o caso da parte espacial em x da equação.

Tabela 2.1: Fórmula de Diferenças Finitas para cálculo de derivada de segunda ordem, aqui representando a derivada na dimensão espacial x

$\frac{\partial^2 u}{\partial x^2}$	$\frac{u_{i-1,j,k} - 2u_{i,j,k} + u_{i+1,j,k}}{\Delta x^2}$

Para entender o que significa o índice i visto na tabela acima é bom se seguir uma analogia: suponhamos um *array* de tamanho maior que três. A posição i seria qualquer posição intermediária, $i-1$ a antecessora e a $i+1$ sucessora. Tal estrutura é chamada de *stencil*. O mesmo vale para os índices j e k. Podemos ver uma alegoria dessa analogia na Figura **??**.

Inserir uma imagem para o stencil 1D

No caso desse trabalho, para a simulação da propagação de ondas em um meio bidimensional ao longo do tempo, teremos que usar três *stencils* (os dois restantes podem ser vistos na Tabela 2.2), um para cada dimensão espacial ou temporal. Para tal, podemos utilizar outra analogia: um *array* tridimensional, onde cada plano de posições no espaço é um instante no tempo. Tal analogia é representada na Figura **??**.

Tabela 2.2: Demais fórmulas de Diferenças Finitas para as dimensões espacial y e temporal t

$\frac{\partial^2 u}{\partial y^2}$	$\frac{u_{i,j-1,k} - 2u_{i,j,k} + u_{i,j+1,k}}{\Delta y^2}$
$\frac{\partial^2 u}{\partial t^2}$	$\frac{u_{i,j,k-1} - 2u_{i,j,k} + u_{i,j,k+1}}{\Delta t^2}$

Contudo, até então, não falamos sobre o método de Diferenças Finitas em si. Trata-se de uma sequência de iterações que marcham em função de alguma variável. No caso desse trabalho, o avanço se dá no tempo. Essa marcha no tempo quer dizer que o valor para um ponto no espaço no próximo instante de tempo será calculado com base nos valores para pontos no espaço em instantes anteriores. Vamos ser explícitos. Temos que a equação 2.1 traduzida nas fórmulas vistas nas tabelas 2.1 e 2.2 se dá por

$$\frac{u_{i,j,k-1} - 2u_{i,j,k} + u_{i,j,k+1}}{\Delta t^2} = \frac{1}{v^2}\left(\frac{u_{i-1,j,k} - 2u_{i,j,k} + u_{i+1,j,k}}{\Delta x^2} + \frac{u_{i,j-1,k} - 2u_{i,j,k} + u_{i,j+1,k}}{\Delta y^2}\right) + f(x,y,t) \tag{2.2}$$

onde u_{k+1} é o ponto com valor a ser calculado para o próximo instante. Logo, ele precisa ser isolado, o que é mostrado na equação 2.3

$$u_{i,j,k+1} = \frac{\Delta t^2}{v^2}\left(\frac{u_{i-1,j,k} - 2u_{i,j,k} + u_{i+1,j,k}}{\Delta x^2} + \frac{u_{i,j-1k} - 2u_{i,j,k} + u_{i,j+1,k}}{\Delta y^2}\right) + \Delta t^2 f(x,y,t) - u_{i,j,k-1} + 2u_{i,j,k} \tag{2.3}$$

buscar referências sobre o método na PIC01

Capítulo 3

A arquitetura computacional atual e a necessidade de paralelização

Para compreender a questão da paralelização envolvida nesse trabalho é necessário entender de onde e porque ela veio. Para tal, é necessário se apresentar as principais peças que constituem um computador moderno, os problemas que surgiram no processo de evolução da *arquitetura computacional* atual e como isso culminou no paralelismo [Bar18].

conferir se deve tratar como atual

Antes de iniciar esse processo, é também necessário se explicar o que se entende por arquitetura computacional. Trata-se da área do conhecimento que estuda a interface entre *software* e *hardware*, desde o mais baixo nível, no qual o processador manipula as informações (instruções e dados) entregues a ele, para toda operação realizada no computador; passando pelas políticas de manipulação de dados nas memórias cache (e seus níveis), de acesso aleatório (RAM - *random access memory*) e de armazenamento não-volátil (discos rígidos, por exemplo); chegando também à interação dos computadores com os demais periféricos que por ventura estão nele conectados, realizando *inputs* (entradas) e/ou *outputs* (saídas), também conhecidas pela abreviação "I/O", como teclado, *mouse*, monitor, etc [Cat09].

Ainda no âmbito da arquitetura de computadores, são estabelecidas análises e metas de performance. Por exemplo, tenta-se determinar se um processador A é mais rápido que um B (sendo B diferente de A) comparando-se o número de instruções que cada um processa, ou quantos ciclos (que serão explicados mais a frente) podem ser dados em um segundo [Wik19]. Questão semelhante encontra-se na computação de alta performance[1]: quão mais rápido um trecho de código executa ao ser paralelizado.

Essa organização focada em um processador, memórias cache, RAM e de armazenamento não-volátil vieram da **arquitetura de von Neumann**, que possui esse nome por conta de seu idealizador, John von Neumann.

3.1 A arquitetura de von Neumann

- von Neumann e o conceito de programa armazenado;
- a forma da arquitetura (componentes e ligações);

[1] **computação de alta performance** (ou **supercomputação**) é um ramo da computação que usa supercomputadores e estuda técnicas de paralelismo visando alcançar velocidades de processamento maiores (do que as de computadores normais, como os pessoais) para a resolução de problemas computacionais complexos, como simulações, modelagens e análises computacionais. Esse ramo também pode, no lugar de focar em processamento mais rápido, resolver problemas de dimensões maiores, que não poderiam ser resolvidos em tempo hábil por computadores comuns [Tec]

- o contraste da arquitetura idealizada com as anteriores.

3.1.1 O gargalo de von Neumann

- o gargalo existente entre o processador e a memória, que não consegue acompanhá-lo.

3.2 O processador

- introdução ao que é um processador;
- apresentação de um processador MIPS simples;
  - entradas;
  - módulos internos;
  - saídas;
- incrementos que os processadores receberam ao longo da história (ILP).

3.3 A memória principal

- introdução do que é uma memória;
- apresentação básica dos tipos de memória (enfase na RAM);
- apresentação básica das propriedades de uma memória RAM.

3.4 A cache

- O princípio da localidade no tempo e no espaço;
- A necessidade da existência de uma cache, visto o princípio da localidade;
- Explicação do que é uma cache;

3.5 Disco rígido

- O que são memórias voláteis e exemplos delas;
- A necessidade de memórias não-voláteis e o HD;
- Outras diferenças entre o HD e as outras memórias envolvidas no trabalho

3.6 Conseguir mais em menos tempo

- A necessidade de se aumentar a velocidade de processamento;
- introdução aos problemas encontrados na computação serial

3.6.1 A barreira da memória - *Memory wall*

Conferir se existe mesmo a memory wall

Revisar o que é a memory wall e completar o itemize abaixo

-

3.6.2 A barreira do paralelismo a nível de instruções - *ILP wall*

- Explicar do que se trata o paralelismo a nível de instruções;
- Explicar os conceitos de superpipeline e superescalar;
- Explicar que, ao se estender muito um pipeline, temos problemas;
- Explicar os problemas da superescalaridade

3.6.3 A barreira no gasto de energia dos processadores - *Power wall*

- Introduzir a lei de Moore;
- Explicar que houve evolução na taxa de clock ao longo do tempo;
- Explicar que essa evolução foi amortecida nos últimos anos devido ao gasto energético e às altas temperaturas que os processadores alcançaram

3.7 Paralelismo - a alternativa para se contornar as barreiras

- Explicar no que consiste o paralelismo;
- em seguida, explicar porquê ele é uma saída possível

3.7.1 Arquiteturas de memória na computação paralela

- Explicar as arquiteturas:
  - de memória compartilhada;
  - memória distribuída;
  - híbrida

3.7.2 Modelos da computação paralela

- Shared Memory Model;
- Threads Model;
- Distributed Memory / Message Passing Model;
- Data Parallel Model;
- Hybrid Model

3.7.3 Desenhando programas paralelos

- explicar a diferença entre a paralelização automática e a manual;
- explicar a necessidade de se entender o problema e o programa;
- explicar
  - particionamento;
  - sincronização.

Capítulo 4

Paralelismo em prática

Introduzir as apis e porque elas existem

4.1 OpenMP

- Do que se trata a OpenMP e onde ela está incluída;
- um programa hello world com OpenMP.

4.2 Pthreads

- Do que se trata a Pthreads e onde ela está incluída;
- um programa hello world com Pthreads.

4.3 MPI

- Do que se trata a MPI e onde ela está incluída;
- um programa hello world com MPI.

Capítulo 5

Do serial ao paralelo

Introduzir

5.1 A construção do código serial

5.2 Primeiro contato do código com as *threads* - OpenMP

5.2.1 Construindo o caminho para a OpenMP

- Explicar o código fake da avaliação de função;
- explicar o código fake das diferenças finitas 1D.

5.3 Um contato mais profundo do código com as *threads* - Pthreads

5.3.1 Construindo o caminho para a Pthreads

- explicar o código fake das diferenças finitas 1D.

5.4 Realizando a mescla de Pthreads e MPI

5.4.1 Contruindo o caminho para a mescla

- explicar o código fake híbrido;
- explicar como o código fake foi rodado no cluster (?).

Capítulo 6

Comparando o paralelo com o serial

Capítulo 7

Considerações Finais

Bibliografia

[Cat09] John Catsoulis. *Designing Embedded Hardware*. O'REILLY, 2009.

[Bar18] Blaise Barney. *Introduction to Parallel Computing*. Jun. de 2018. URL: https://computing.llnl.gov/tutorials/parallel_comp/#Neumann.

[Mac18] Marcelo Lopes de Macedo Ferreira Cândido. *Modelagem Matemática da Propagação de Ondas em Meios Não Homogêneos*. Iniciação Científica. Centro Federal de Educação Tecnológica de Minas Gerais, 2018.

[Wik19] Wikipedia contributors. *Computer architecture — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Computer_architecture&oldid=876516273. [Online; accessed 12-January-2019]. 2019.

[Tec] Techopedia. *High-Performance Computing (HPC)*. URL: https://www.techopedia.com/definition/4595/high-performance-computing-hpc.