UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELETRÔNICA E DE TELECOMUNICAÇÕES

HENRIQUE SANTOS DE LIMA

UTILIZAÇÃO DE PLACAS DE VÍDEO NVIDIA PARA OTIMIZAR O TEMPO DE CÁLCULO DA PREDIÇÃO DE PERDA DE PERCURSO UTILIZANDO O MÉTODO DE EQUAÇÕES PARABÓLICAS E O MÉTODO DAS DIFERENÇAS FINITAS

UBERLÂNDIA – MINAS GERAIS
2023

HENRIQUE SANTOS DE LIMA

UTILIZAÇÃO DE PLACAS DE VÍDEO NVIDIA PARA OTIMIZAR O TEMPO DE CÁLCULO DA PREDIÇÃO DE PERDA DE PERCURSO UTILIZANDO O MÉTODO DE EQUAÇÕES PARABÓLICAS E O MÉTODO DAS DIFERENÇAS FINITAS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Eletrônica e de Telecomunicações do Faculdade de Engenharia Elétrica da Universidade Federal de Uberlândia, como requisito parcial à obtenção do grau de bacharel em Engenharia Eletrônica e de Telecomunicações.

Orientador: Lorenço Santos Vasconcelos

UBERLÂNDIA – MINAS GERAIS

2023

AGRADECIMENTOS

Aos meus pais e aos meus irmãos, pelo amor, incentivo e apoio incondicional que me deram desde o início. Agradeço as minhas irmãs por me ensinarem a ler e escrever quando eu era apenas uma criança sonhadora, por sua dedicação e amor. Agradeço a minha mãe por me incentivar a ir além do conhecimento disponível e por ter estudado para poder me ensinar. Ao meu pai pelas noites não dormidas de trabalho dedicadas para que esse sonho se tornasse realidade. Aos ensinamentos de como ser uma boa pessoa e principalmente a conquistar as coisas com honra e mérito. O apoio dado foi fundamental para o meu sucesso acadêmico e eu sou profundamente grato por tudo o que vocês fizeram por mim. Obrigado por serem minha família, minha base, minha fonte de amor e apoio.

Agradeço à Universidade Federal de Uberlândia, ao seu corpo docente, direção e administração, por proporcionar todo o apoio e recursos necessários para que eu pudesse realizar meu sonho de infância, me tornar um engenheiro. Agradeço a todos os professores que não trataram suas funções como apenas um trabalho, mas sim como uma missão de preparar e capacitar estudantes para se tornarem profissionais respeitáveis e competentes. A minha formação acadêmica e profissional é resultado do esforço e dedicação de todos os envolvidos na instituição e estou imensamente grato por tudo o que eles fizeram por mim.

Um agradecimento especial à Professora Dr. Maria das Graças Pereira, que me ensinou muito mais do que apenas Cálculo. Ela me proporcionou uma base matemática sólida, que havia faltado em meu ensino médio e, em momentos de dificuldade, me ofereceu apoio e orientação para continuar meus estudos. O carinho e dedicação utilizados em sua metodologia de ensino me permitiram abrir minha mente e apreciar o aprendizado nas matérias de cálculo, fundamentais para meu desenvolvimento matemático. Agradeço imensamente a ela por seu empenho e dedicação em me ensinar e me proporcionar uma base sólida para minha carreira acadêmica.

Gostaria de agradecer ao Professor Dr. Lorenço Santos Vasconcelos, por ter tido a paciência e habilidade para ensinar Eletromagnetismo, matéria considerada por mim e outros como uma das mais complexas do curso. Sua orientação em minha iniciação científica e neste projeto foi fundamental para o meu sucesso e desenvolvimento acadêmico. Agradeço pela visão investigativa e conhecimento teórico que adquiri ao ser seu aluno em três matérias fundamentais para um engenheiro eletrônico e de telecomunicações: Sinais e Sistemas, Eletromagnetismo e Princípios de Comunicações. A sua orientação em dois projetos de pesquisa desafiadores me permitiu crescer acadêmica e profissionalmente, e eu sou profundamente grato por tudo o que você fez.

Gostaria de expressar minha sincera gratidão à minha melhor amiga e amada namorada, Lesly Viviane Montufar Berrios, por ter aceitado o desafio de aprender tudo desde o início e contribuir com suas habilidades para a realização deste projeto. Seu tempo dedicado, suporte nas dúvidas, soluções criativas para os desafios encontrados, foram fundamentais para o sucesso deste projeto. Sua colaboração e apoio foram inestimáveis e eu sou profundamente grato por tudo o que você fez.

Agradeço ao Guilherme Ferreira de Jesus, por ser um parceiro incansável desde o início até o fim de minha jornada acadêmica. Estudar e crescer juntos foi uma experiência enriquecedora e uma fonte constante de apoio e motivação. Sua dedicação, amizade e colaboração foram fundamentais para o sucesso deste trabalho, e eu sou profundamente grato por tudo o que você fez.

Agradeço a todos os amigos e colegas que, mesmo que não nos víssemos todos os dias, me incentivaram e se alegraram com meus progressos e conquistas. Especialmente durante a pandemia, quando o isolamento social foi implantado, vocês foram minha companhia remota que me apoiaram em todos os momentos.

"É melhor lançar-se à luta em busca do triunfo mesmo expondo-se ao insucesso, que formar fila com os pobres de espírito, que nem gozam muito nem sofrem muito; E vivem nessa penumbra cinzenta sem conhecer nem vitoria nem derrota."

(Franklin Roosevelt)

RESUMO

LIMA, H. S. de. **Utilização de placas de vídeo NVIDIA para otimizar o tempo de cálculo da predição de perda de percurso utilizando o método de equações parabólicas e o método das diferenças finitas.** Monografia (Graduação) — Faculdade de Engenharia Elétrica, Universidade Federal de Uberlândia, 2023.

Este trabalho é resultado de um projeto realizado em grupo para otimizar o tempo de cálculo nas predições de perda de percurso, utilizando técnicas de computação paralela no método de equações parabólicas e diferenças finitas para aproximar a solução da equação de onda eletromagnética.

Com a crescente expansão das redes de telecomunicações, a agilidade de implementação dos projetos é um diferencial para qualquer empresa, por exemplo, para comunicação celular. Para isso, é importante otimizar, também, o tempo de projeto desses sistemas de comunicação sem fio. Uma das etapas de projeto é a análise de cobertura ou predição de perda de percurso do sistema. Essa análise pode ser feita de duas maneiras: uma forma mais rápida (encontrada em alguns softwares comerciais), mas que assume uma modelagem simplificada e uma forma mais detalhada que requer métodos complexos e com tempos computacionais elevados.

Um desses métodos que oferece resultados muito bons é o das equações parabólicas, que pode ser resolvido por meio de diferenças finitas. Entretanto, a sua execução pode ser dispendiosa. Por isso, esse método não costuma ser utilizado em projetos práticos. Assim, é interessante procurar maneiras de otimizar sua execução, possibilitando seu uso em aplicações práticas.

Para solucionar o problema do longo tempo gasto fazendo cálculos durante a simulação, foi utilizado a *Graphics Processing Units* (GPU) para realizar os cálculos de modo paralelo, utilizando melhor os recursos disponíveis no computador e entregando os resultados com erro insignificante ao erro da simulação sequencial.

Atualmente, o projeto está em andamento e apenas a primeira etapa foi concluída, que consiste em gerar simulações com uma precisão aceitável. Os resultados obtidos até agora são promissores, mas ainda é necessário corrigir alguns detalhes para que o erro não ultrapasse o erro de ponto flutuante 10^{-14}. A próxima etapa é validar e corrigir todas as funcionalidades implementadas, antes de adicionar novas funcionalidades.

Palavras-chave: Computação-paralela. CUDA.Equações-parabólicas. GPU. Redução cíclica paralela.PCR.

ABSTRACT

This work is the result of a group project to optimize the calculation time in path loss predictions, using parallel computing techniques in the method of parabolic equations and finite differences to approximate the solution of the electromagnetic wave equation.

With the growing expansion of telecommunications networks, the agility of project implementation is a differential for any company, for example, for cellular communication. To do this, it is also important to optimize the project time of these wireless communication systems. One of the project stages is the coverage analysis or prediction of path loss of the system. This analysis can be done in two ways: a faster way (found in some commercial software), but that assumes a simplified modeling and a more detailed way that requires complex methods and with high computational times.

One of the methods that offers very good results is the parabolic equations, which can be solved by finite differences. However, its execution can be costly. Therefore, this method is not usually used in practical projects. Thus, it is interesting to look for ways to optimize its execution, making it possible to use it in practical applications.

To solve the problem of the long time spent doing calculations during the simulation, the GPU was used to perform the calculations in parallel mode, making better use of the resources available on the computer and delivering the results with insignificant error to the error of the sequential simulation.

Currently, the project is ongoing and only the first stage has been completed, which consists of generating simulations with an acceptable precision. The results obtained so far are promising, but it is still necessary to correct some details so that the error does not exceed the floating point error 10^{-14}. The next step is to validate and correct all the implemented features, before adding new features.

Keywords: Parallel computing. CUDA. Parabolic equations. GPU. Parallel cyclic reduction. PCR.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE CÓDIGOS-FONTE

LISTA DE ABREVIATURAS E SIGLAS

5G	5^a Geração
API	*Application Programming Interface*
CPU	*Central Processing Unit*
CR	*Cyclic Reduction*
CUDA	*Compute Unified Device Architecture*
DMA	*Direct Memory Access*
GPU	*Graphics Processing Units*
GW	*Global Works*
LTE	Long Term Evolution
LW	*Local Works*
MWG	*Max Work Group*
nvcc	*NVIDIA CUDA Compiler*
PCR	Parallel Cyclic Reduction
PE	*Parabolic Equation*
PML	*Perfectly Matched Layer*
PVI	Problema de Valor Inicial
RAM	*Random Access Memory*
SI	*Sistema Internacional de Unidades*
VRAM	*Video Random Access Memory*

LISTA DE SÍMBOLOS

ε	Permissividade Elétrica
σ	Condutividade Elétrica
ζ	Variável relacionada a altura na mudança de variável
ξ	Variável relacionada a distância para-axial na mudança de variável
Δ	Indica a variação de uma varável
∇	Operador Nabla
δ	Impedância superficial do solo
Ψ	Campo Eletromagnético generalizado
μ	permissividade magnética
θ	Distância
λ	Comprimento de onda
$\in$	Pertence
ω	Frequência de oscilação da onda
ω_0	Fase da onda
H	Campo magnético
E	Campo elétrico
B	Densidade de campo magnético
a_e	Raio médio da Terra

SUMÁRIO

1 INTRODUÇÃO

Ao realizar um projeto de telecomunicações, por exemplo, o de uma estação radio base para telefonia celular, a estimativa da sua área de cobertura é indispensável. Sem essa informação não é possível determinar onde será instalado a nova célula, até onde esta célula causará interferências, até onde as antenas receptoras obterão o sinal transmitidos, quais os parâmetros da antena receptora, etc. Métodos analíticos precisos envolvem a solução de equações extremamente complexas, devido a todos os fatores que interagem com o campo eletromagnético e, em sua maioria, são computacionalmente inviáveis. Entretanto, existem métodos aproximados de obtenção do campo eletromagnético que adicionam pequenos erros ao resultado, simplificando algumas equações e possibilitando a solução computacional, como exemplo o Método de elementos finitos, o Método de Monte Carlo, o Método de espectros modificados, entre outros.

O método utilizado é o de equações parabólicas que, por meio de condições estabelecidas para adotar algumas aproximações, a equação da onda é simplificada e assim se torna computacionalmente possível, utilizando a discretização e uso do método de diferenças finitas. Por outro lado, o método tem os seus inconvenientes e, em geral, envolve matrizes muito grandes com alto custo computacional envolvido. Dentre suas operações, a operação de solução de sistema matricial é a que mais demanda processamento e consequentemente eleva o tempo de predição.

1.1 Justificativas

Mesmo com a solução sendo viável computacionalmente, a espera de dias e horas para obtenção dos resultados continua sendo um problema para os projetos de propagação. Com a chegada e expansão das redes 5^a Geração (5G) e novos enlaces Long Term Evolution (LTE), é importante usufruir de uma ferramenta que entregue os resultados o mais rápido possível, permitindo que o projetista, com as informações do terreno na direção da propagação, manipule os parâmetros da antena e obtenha as predições das métricas antes mesmo de instalar uma antena sem esperar longos períodos de simulações.

1.2 Objetivos

O objetivo deste trabalho é dissertar sobre a implementação de técnicas de otimização computacional e técnicas de computação paralela para reduzir o tempo de simulação gasto nos

cálculos para obter a perda de percurso sem perder a precisão.

Para isso utiliza-se a *Application Programming Interface* (API) *Compute Unified Device Architecture* (CUDA) para implementar os algoritmos paralelos e efetuar as operações da simulação mais rápido que uma solução sequencial.

O *software* gerado é capaz de ler os parâmetros de um arquivo JSON e realizar a simulação, desde a geração do campo inicial até a propagação.

A geração dos gráficos ainda requer o uso de outros softwares auxiliares, que lêem o arquivo gerado pelo projeto e exibem os resultados para o usuário. Em melhorias futuras será implementado uma interface gráfica, para que o usuário possa interagir com os resultados e extrair as informações desejadas.

1.3 Organização do Trabalho

O trabalho inicia-se na obtenção das equações parabólica partindo das equações de Maxwell. Aplicando as devidas aproximações até obter as equações matriciais.

Para solução de sistemas lineares tridiagonais, é estudado diferentes algorítimos para definir qual a melhor escolha para uma solução em arquitetura paralela.

Para uma compreensão solida, é estudado as características de *software* e *hardware* para implementação dos algorítimos paralelos, uma vez que, com uma implementação descuidada, a solução paralela pode ser mais ineficiente que uma solução sequencial.

No último capitulo são apresentados os resultados obtidos e trabalhos futuros para continuação desse trabalho.

2 PROPAGAÇÃO

2.1 Introdução

O propósito deste trabalho é aplicar técnicas e algoritmos de computação paralela para soluções aproximadas da Equação de onda eletromagnética. Para entendimento da solução pelo método de Equações parabólicas é preciso conhecer os conceitos de eletromagnetismo e propagação.

O estudo sobre o eletromagnetismo se iniciou na eletrostática, onde as cargas são invariantes no tempo. A maioria das aplicações, como a transmissão de sinais, só são possíveis devido às interações que os campos elétricos e magnéticos fazem quando são variáveis no tempo.

2.2 Eletrostática

Cargas são definidas como positivas(+) ou negativas(-) e interagem entre si. Cargas de mesmo sinal se repelem e cargas opostas se atraem. A intensidade da força que uma carga aplica sobre a outra foi formulada por Charles Augustin de Coulomb (1736–1806), e ficou conhecida como lei de Coulomb (NUSSENZVEIG, 2015). Coulomb verificou experimentalmente que a intensidade da força que uma carga aplicava a outra era proporcional ao produto das cargas e inversamente proporcional ao quadrado da distância, como mostra a Equação 2.1.

$$F \propto \frac{q_1 q_2}{r^2} \tag{2.1}$$

Em que:

- F é a força que a carga q_2 percebe devido à presença da carga q_1 em Newton [N].
- q_1 e q_2 são cargas em Coulomb [C].
- r é a distância entre as cargas.

Atualmente a forma mais moderna e adequada ao SI (Sistema internacional) é mostrada na Equação (2.2), em que a constante de proporcionalidade está relacionada com o meio, representado pela permissividade elétrica ε.

$$\vec{F} = \frac{q_1 q_2}{4\pi\varepsilon r^3}\vec{r} \quad [N] \tag{2.2}$$

2.2.1 Campo elétrico estático

Michael Faraday propôs o conceito de campo elétrico, uma grandeza que não existe fisicamente de fato, mas é uma ferramenta matemática importante no estudo do eletromagnetismo.

Dado uma carga puntual, isto é, de tamanho desprezível, sem a presença de outra carga não existe interação de forças, porém se existisse uma outra carga, essa faria a interação com a carga puntual e seria aplicado uma força elétrica (2.2). Assim, por meio de uma carga de prova define-se o campo elétrico como a força elétrica normalizada pela carga de prova q_p, Equações (2.3) e (2.4).

$$\vec{E} = \frac{\vec{F}}{q_p} \tag{2.3}$$

$$\vec{E} = \frac{q}{4\pi\varepsilon r^3}\vec{r} \quad \left[\frac{N}{C}\right] \text{ ou } \left[\frac{V}{m}\right] \tag{2.4}$$

2.2.2 Campo magnético estacionário

Um campo magnético é uma região onde existe uma força magnética. Ele pode ser produzido por correntes elétricas, como as encontradas em ímãs naturais e artificiais, e também pode afetar o comportamento de partículas carregadas, como elétrons e prótons.

Os campos magnéticos podem ser representados usando linhas de campo, que mostram a direção e a intensidade do campo magnético em diferentes pontos no espaço. Essas linhas de campo sempre formam um caminho fechado chamado de "fluxo de campo".

A lei de Biot-Savart é uma equação matemática usada para calcular o campo magnético produzido por uma corrente elétrica, desenvolvida pelos físicos franceses Jean-Baptiste Biot e Félix Savart (NUSSENZVEIG, 2015) no início do século XIX. A Equação (2.5) mostra a expressão:

$$d\vec{B} = \frac{\mu}{4\pi}\frac{Id\vec{l}\times\vec{r}}{r^3}[T] \tag{2.5}$$

Em que:

- $d\vec{B}$ é a densidade de fluxo magnético diferencial em T ou $\frac{Wb}{m}$
- μ é a permeabilidade magnética do meio (Tm/A)
- I é a corrente elétrica.
- $d\vec{l}$ é um elemento infinitesimal de comprimento da corrente.
- $\vec{r}$ é o vetor deslocamento entre o elemento diferencial de corrente e o ponto em que o campo magnético é medido.

A Equação (2.5) mostra que o campo magnético produzido por uma corrente elétrica depende da intensidade da corrente, da distância entre a corrente e o ponto onde o campo magnético é medido, e da orientação da corrente em relação ao ponto de medida. A equação é válida somente para correntes estacionárias.

O campo magnético é definido pela Equação (2.6).

$$\vec{H} = \frac{\vec{B}}{\mu}\left[\frac{A}{m}\right] \tag{2.6}$$

2.3 Campos Eletromagnéticos

As Equações de Maxwell são um conjunto de Equações matemáticas que descrevem como os campos elétricos e magnéticos se relacionam e como eles se propagam através do espaço. Elas foram desenvolvidas pelo físico James Clerk Maxwell no século XIX e são fundamentais para a compreensão do eletromagnetismo.

As quatro equações de Maxwell em forma diferencial para o espaço livre são:

$$\vec{\nabla} \cdot \vec{E} = \frac{\rho}{\varepsilon_0} \tag{2.7}$$

$$\vec{\nabla} \cdot \vec{B} = 0 \tag{2.8}$$

$$\vec{\nabla} \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \tag{2.9}$$

$$\vec{\nabla} \times \vec{B} = \mu_0 \vec{J} + \mu_0 \varepsilon_0 \frac{\partial \vec{E}}{\partial t} \tag{2.10}$$

Em que:

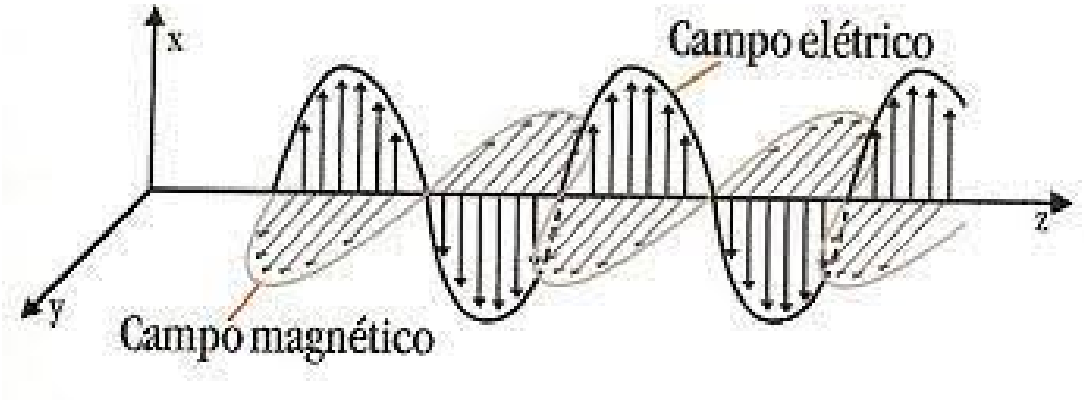
- $\vec{E}$ é o campo elétrico,
- $\vec{B}$ é a densidade de fluxo magnético,
- ρ é a densidade de carga elétrica,
- ε_0 é a permissividade elétrica do vácuo ($8.854 \cdot 10^{-12}$ F/m),
- μ_0 é a permeabilidade magnética do vácuo ($4\pi 10^{-7}$ $\left[\frac{H}{m}\right]$),
- $\vec{J}$ é a densidade de corrente elétrica.

As quatro equações de Maxwell descrevem a relação entre os campos elétricos e magnéticos e a geração e propagação desses campos. As equações (2.9) e (2.10) mostram que a existência de um campo elétrico com variação no tempo produzirá um campo magnético. Quando um campo magnético é variável no tempo, por sua vez, também produzira um campo elétrico. Dessa forma, o campo eletromagnético é definido como a presença simultânea do campo elétrico variante no tempo e do campo magnético variante no tempo. As quatro Equações de Maxwell, (2.7), (2.8), (2.9), (2.10), resumem todas as interações que o campo eletromagnético faz. São elas lei de Gauss, lei de Gauss para o magnetismo, Lei de Faraday-Lenz e lei de Ampère-Maxwell respectivamente.

Quando a variação temporal de um campo é sinuosidal, é dito que esse campo é harmônico no tempo. Além disso, campos com variação não sinuosidal podem ser expressos, por meio de análise de Fourier, pela combinação linear de campos harmônicos (Equação 2.14). Um campo elétrico harmônico gera um campo magnético também harmônico, e vice-versa. O campo elétrico é perpendicular ao campo magnético e à direção de propagação da onda eletromagnética. A figura 2.1 ilustra os campos perpendiculares entre si e à direção de propagação

$$\vec{E}(x,y,z,t) = \sum \vec{E}_i(x,y,z)e^{-j\omega_i t} \tag{2.11}$$

Figura 2.1 – Propagação do campo eletromagnético .



Fonte: VELAME (2019).

2.4 Ondas eletromagnéticas

A criação e radiação de ondas eletromagnéticas ocorre através do movimento acelerado das cargas elétricas, criando uma onda eletromagnética que se propaga através do espaço, carregando energia e informação. Ao se propagar, a onda eletromagnética pode sofrer alguns fenômenos devido às características do meio de propagação como a refração, difração, reflexão, absorção, interferência e polarização.

- Refração: Fenômeno pelo qual a direção de propagação de uma onda eletromagnética é alterada ao passar por um meio com densidade diferente. Isso ocorre pois a velocidade de propagação é diferente para cada meio de propagação.
- Difração: As ondas eletromagnéticas se espalham ao passar por uma abertura ou obstáculo. Isso resulta em padrões de interferência e difração nos quais a intensidade da onda eletromagnética varia de acordo com a posição.

- Reflexão: Ocorre quando a onda eletromagnética volta para o meio de origem quando encontra uma superfície reflexiva. A reflexão é responsável por efeitos como o espelhamento e a reflexão de raios de luz
- Absorção: É o fenômeno pelo qual a onda eletromagnética perde energia ao entrar em contato com um meio absorvente.
- Interferência: Duas ou mais ondas eletromagnéticas se sobrepõem e se combinam para formar uma onda resultante.
- Polarização: A direção do campo elétrico de uma onda eletromagnética é restrita a uma determinada direção.

Para obter a Equação de onda, parte-se das Equações de Maxwell. Considerando o vácuo como meio de propagação, não existirá densidade de cargas e nem densidade de corrente.

$$\vec{\nabla} \cdot \vec{E} = 0 \tag{2.12}$$

$$\vec{\nabla} \cdot \vec{B} = 0 \tag{2.13}$$

$$\vec{\nabla} \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \tag{2.14}$$

$$\vec{\nabla} \times \vec{B} = \mu_0 \varepsilon_0 \frac{\partial \vec{E}}{\partial t} \tag{2.15}$$

Aplicando o operador rotacional na Equação (2.14).

$$\vec{\nabla} \times \left(\vec{\nabla} \times \vec{E} \right) = \vec{\nabla} \times \left(-\frac{\partial \vec{B}}{\partial t} \right) \tag{2.16}$$

Como o rotacional não depende da variável t, a Equação (2.16) pode ser escrita como:

$$\vec{\nabla} \times \left(\vec{\nabla} \times \vec{E} \right) = -\frac{\partial \left(\vec{\nabla} \times \vec{B} \right)}{\partial t} \tag{2.17}$$

Utilizando a propriedade do triplo produto vetorial (SAUTER; AZEVEDO; KONZEN, 2022) Equação (2.18).

$$\vec{A} \times \vec{B} \times \vec{C} = \left(\vec{A} \cdot \vec{C} \right) \vec{B} - \left(\vec{A} \cdot \vec{B} \right) \vec{C} \tag{2.18}$$

A Equação (2.17) fica:

$$\vec{\nabla} \left(\vec{\nabla} \cdot \vec{E} \right) - \left(\vec{\nabla} \cdot \vec{\nabla} \right) \vec{E} = -\frac{\partial \left(\vec{\nabla} \times \vec{B} \right)}{\partial t} \tag{2.19}$$

Substituindo as Equações (2.12) e (2.15) na Equação (2.19), chega-se à Equação (2.20).

$$\vec{\nabla}(0) - \left(\vec{\nabla}^2\right)\vec{E} = -\frac{\partial\left(\mu_0\varepsilon_0\frac{\partial\vec{E}}{\partial t}\right)}{\partial t}$$

$$\vec{\nabla}^2\vec{E} = \mu_0\varepsilon_0\frac{\partial^2\vec{E}}{\partial t^2} \tag{2.20}$$

De modo análogo, aplicando o operador rotacional na Equação (2.15) pode-se escrever a Equação 2.21.

$$\vec{\nabla}\times\left(\vec{\nabla}\times\vec{B}\right) = \vec{\nabla}\times\left(-\mu_0\varepsilon_0\frac{\partial\vec{E}}{\partial t}\right) \tag{2.21}$$

Como o rotacional não depende da variável t a Equação (2.21) pode ser escrita como:

$$\vec{\nabla}\times\left(\vec{\nabla}\times\vec{B}\right) = \mu_0\varepsilon_0 - \frac{\partial\left(\vec{\nabla}\times\vec{E}\right)}{\partial t} \tag{2.22}$$

Utilizando a propriedade do triplo produto vetorial Equação (2.18). A Equação (2.22) fica:

$$\vec{\nabla}\left(\vec{\nabla}\cdot\vec{B}\right) - \left(\vec{\nabla}\cdot\vec{\nabla}\right)\vec{B} = -\mu_0\varepsilon_0\frac{\partial\left(\vec{\nabla}\times\vec{E}\right)}{\partial t} \tag{2.23}$$

Substituindo as Equações (2.13) e (2.14) na Equação (2.23), chega-se na Equação (2.24).

$$\vec{\nabla}(0) - \left(\vec{\nabla}^2\right)\vec{B} = -\mu_0\varepsilon_0\frac{\partial\left(\frac{\partial\vec{B}}{\partial t}\right)}{\partial t}$$

$$\vec{\nabla}^2\vec{B} = \mu_0\varepsilon_0\frac{\partial^2\vec{B}}{\partial t^2} \tag{2.24}$$

As Equações (2.20) e (2.24) são Equações de onda genéricas e descrevem a propagação da onda eletromagnética no vácuo.

A Equação de onda eletromagnética é uma EDP(Equação diferencial parcial) e a sua solução para longas distâncias é computacionalmente inviável. Com algumas condições, o método de Equações parabólicas e diferenças finitas (VASCONCELOS, 2017) possibilita um cálculo aproximado e viável computacionalmente.

3 EQUAÇÕES PARABÓLICAS

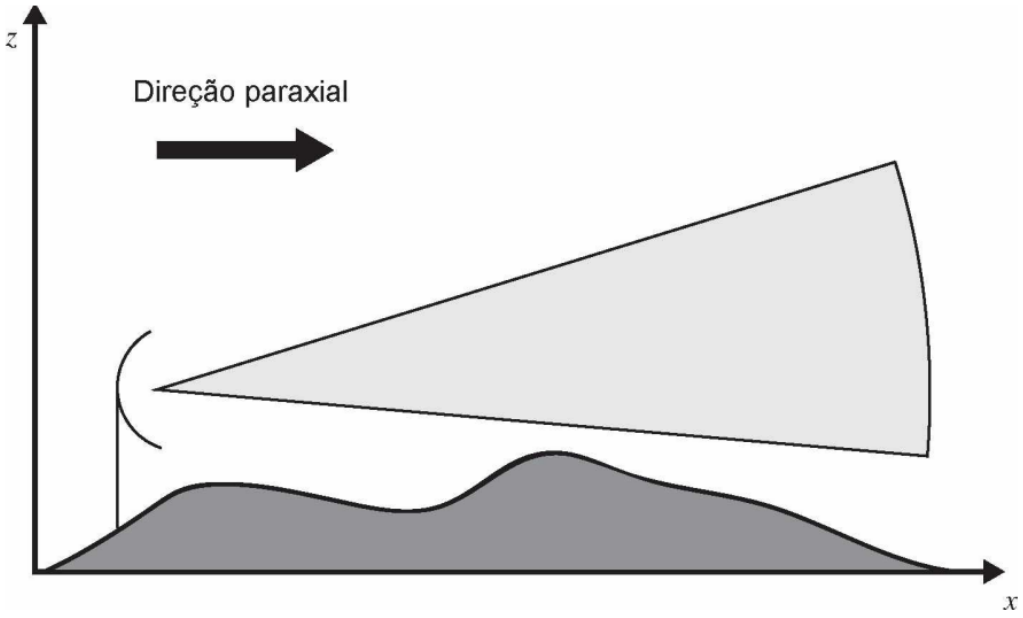
3.1 Introdução

O método *Parabolic Equation* (PE) ainda é muito estudado, pois considera vários fenômenos que acontecem na propagação de ondas eletromagnéticas, como a reflexão, refração, difração e ondas de superfície.

Este método simplifica a equação da onda de Helmholtz (VASCONCELOS, 2017) para uma equação parabólica e obtém a solução numericamente. O cerne do método PE é considerar que a maior parcela da energia eletromagnética propaga em um sentido, isto é, a componente de campo retroespalhado é desprezada. Assim, trabalha-se com uma aproximação parabólica.

Em grande parte dos problemas, a fronteira inferior é um interface de atmosfera/solo e a fronteira superior deveria se estender ao infinito (VASCONCELOS, 2017), o que computacionalmente é inviável, mas com os devidos tratamentos se torna possível. A Figura 3.1 ilustra o cenário típico de propagação para-axial, direção na qual a maior parte da energia da onda está concentrada. O cone sombreado é a região onde o método PE possui maior precisão.

Figura 3.1 – Cenário de propagação Para-axial



Fonte: Levy (2000)

Para a formulação do método PE (VASCONCELOS, 2017) deve-se considerar:

- É assumido que os campos possuem variação harmônica com frequência angular ω, $\vec{E} = \vec{E}(x,y,z)e^{-j(\omega t+\omega_0)}, \vec{H} = \vec{H}(x,y,z)e^{-j(\omega t+\omega_0)}$.

- O método PE apresentado neste trabalho se limita a problemas bidimensionais.
- É utilizado o sistema cartesiano de coordenadas espaciais.
- É assumido a polarização horizontal como $\vec{E} = E_y(x,z)\hat{y}$ possuindo apenas a componente em y.
- É assumido a polarização vertical como $\vec{H} = H_y(x,z)\hat{y}$ possuindo apenas a componente em y.
- Para generalização das equações, utiliza-se a componente de campo $\psi(x,z)$ definida na Equação (3.1). Em que o campo ψ representa o campo elétrico $E_y(x,z)$ quando a polarização é horizontal e, quando a polarização é vertical, representa o campo magnético $H_y(x,z)$.

$$\psi(x,z) = \begin{cases} E_y(x,z) & \text{se polarização horizontal} \\ H_y(x,z) & \text{se polarização vertical} \end{cases} \tag{3.1}$$

Considerando campos com variação harmônica no tempo (OPPENHEIM, 2010), utilizando notação fasorial e considerando os quatro tipos principais de meios para propagação de ondas eletromagnéticas:

- espaço livre ($\sigma = 0$, $\varepsilon = \varepsilon_0$, $\mu = \mu_0$);
- dielétricos sem perdas ($\sigma = 0$, $\varepsilon = \varepsilon_r\varepsilon_0$, $\mu = \mu_r\mu_0$);
- dielétricos com perdas ($\sigma \neq 0$, $\varepsilon = \varepsilon_r\varepsilon_0$, $\mu = \mu_r\mu_0$);
- bons condutores ($\sigma = \infty$, $\varepsilon = \varepsilon_0$, $\mu = \mu_r\mu_0$ ou $\sigma >> \omega\varepsilon$);

Em que:

- σ é a condutividade Elétrica.
- ω é a frequência angular das ondas.

Fazendo procedimento análogo para dedução das equações (2.4) e (2.6), obtêm-se as Equações (3.2) e (3.3):

$$\nabla^2 E - \gamma^2 E = 0 \tag{3.2}$$

$$\nabla^2 H - \gamma^2 H = 0 \tag{3.3}$$

Em que:

$$\gamma^2 = -\omega^2\mu\varepsilon + j\omega\mu\sigma \tag{3.4}$$

As equações (3.2) e (3.3) são conhecidas como equações homogêneas vetoriais de Helmholtz .

A equação homogênea vetorial de Helmholtz em um meio livre de fontes para o campo vetorial ψ fica:

$$\nabla^2\psi - \gamma^2\psi = 0 \tag{3.5}$$

Como o campo vetorial ψ só possui a componente em y, a Equação (3.5) se reduz a Equação (3.6).

$$\frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial z^2} - \gamma^2\psi = 0 \tag{3.6}$$

em que a constante de propagação γ é definida pela Equação (3.4).

Considerando um meio sem perdas e não magnético ($\varepsilon = \varepsilon_r\varepsilon_0$, $\mu = \mu_0$, $\sigma = 0$) a Equação (3.4) pode ser rescrita como a Equação (3.7).

$$\gamma^2 = -\omega^2\mu_0\varepsilon_0\varepsilon_r \tag{3.7}$$

Assim, a Equação (3.6) se torna a Equação (3.9).

$$\frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial z^2} - \omega^2\mu_0\varepsilon_0\varepsilon_r\psi = 0 \tag{3.8}$$

$$\frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial z^2} - k_0^2\varepsilon_r\psi = 0 \tag{3.9}$$

Em que k_0 é a constante de fase do vácuo definido na Equação (3.10):

$$k_0 = \omega\sqrt{\mu_0\varepsilon_0} \tag{3.10}$$

Em um meio sem perdas e não magnético, o índice de refração pode ser dado por $n = \sqrt{\varepsilon_r}$. Desse modo, chega-se à Equação (3.11)

$$\frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial z^2} - k_0^2n^2\psi = 0 \tag{3.11}$$

Em que n é o índice de refração do meio.

Se o índice de refração variar espacialmente, a Equação (3.11) não é exatamente válida, porém se a variação for suave na escala do comprimento de onda, então a Equação (3.12) se torna uma boa aproximação.

$$\frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial z^2} - k_0^2n^2(x,z)\psi = 0 \tag{3.12}$$

O ponto-chave da aproximação parabólica é considerar que a componente de campo $\psi(x,z)$ pode ser fatoradas em dois termos (VASCONCELOS, 2017), conforme Equação (3.13).

$$\psi(x,z) = u(x,z)e^{jk_0x} \tag{3.13}$$

Em que o termo e^{jk_0x} é uma variação rápida de fase e $u(x,z)$ é uma variação lenta de amplitude. Como o interesse está nas variações de escalas grandes em relação ao comprimento de onda, essa condição pode ser tomada. Essa decomposição simplifica a equação parabólica frente à equação elíptica, contudo diminui a precisão para a região fora do cone mostrado na figura 3.1, não sendo um problema, pois a região de interesse está em longas distâncias.

Substituindo a Equação (3.13) em (3.12), desenvolve-se as equações (3.14) e (3.15).

$$\frac{\partial^2\left(ue^{jk_0x}\right)}{\partial x^2} + \frac{\partial^2\left(ue^{jk_0x}\right)}{\partial z^2} - k_0^2 n^2(x,z)ue^{jk_0x} = 0 \tag{3.14}$$

$$\frac{\partial^2 u}{\partial x^2} + 2jk_0\frac{\partial u}{\partial x} + \frac{\partial^2 u}{\partial z^2} + k_0^2\left(n^2(x,z)-1\right)u = 0 \tag{3.15}$$

A Equação (3.15) pode ser fatorada utilizando o operador pseudodiferencial (LEVY, 2000; VASCONCELOS, 2017).

$$\left[\frac{\partial}{\partial x} + jk_0(1-Q)\right]\left[\frac{\partial}{\partial x} + jk_0(1+Q)\right]u = 0 \tag{3.16}$$

Em que:

$$Q = \sqrt{\frac{1}{k_0^2}\frac{\partial^2}{\partial z^2} + n^2(x,z)} \tag{3.17}$$

Operadores pseudodiferenciais são formados por derivadas parcias e funções ordinárias das variáveis em questão (VASCONCELOS, 2017).

Fatorar a Equação (3.15) nos termos da Equação (3.16) atribui erros devido ao índice de refração variar conforme a distância para-axial x varia. O operador Q não considera as variações em x, portanto a fatoração é incorreta (VASCONCELOS, 2017). Contudo, com os cuidados necessários, é possível considerar que as variações de n em relação a x são pequenas o suficiente para os erros resultantes permanecerem também pequenos.

Tomando a Equação fatorada (3.16) como correta, desde que atenda as condições impostas, a próxima etapa é dividi-la em duas equações e encontrar as suas soluções separadamente.

Separando em duas equações obtém-se as equações (3.18) e (3.19).

$$\frac{\partial u}{\partial x} = -jk_0(1-Q)u \tag{3.18}$$

$$\frac{\partial u}{\partial x} = -jk_0(1+Q)u \tag{3.19}$$

Em que a Equação (3.18) corresponde à propagação no sentido positivo da direção para-axial e a Equação (3.19) à propagação no sentido negativo da direção para-axial. Em meios independentes da distância para-axial ou onde pode-se fazer tal aproximação sem conflitar com as condições impostas para fatoração da Equação (3.15), a solução das equações (3.18) e (3.19) satisfazem a Equação (3.15) (VASCONCELOS, 2017). Contudo essas soluções não representam o campo eletromagnético real (LEVY, 2000).

A solução completa do campo eletromagnético deve considerar o retroespalhamento do campo e, para isso, é necessário a solução simultânea do sistema de equações (3.20). O campo eletromagnético real é constituído pela soma da componente de propagação (3.18) com a componente de retroespalhamento (3.19).

$$\begin{cases} u = u^+ + u^- \\ \frac{\partial u}{\partial x}^+ = -jk_0(1-Q)u^+ \\ \frac{\partial u}{\partial x}^- = -jk_0(1+Q)u^- \end{cases} \tag{3.20}$$

Em que u^+ e u^- representam as componentes de propagação na direção para-axial positiva de propagação e de retroespalhamento respectivamente.

Supondo que as superfícies de propagação são inclinadas suavemente, pode-se considerar que a maior parte da energia eletromagnética propagada estará no sentido do transmissor para o receptor, e que a componente de campo retroespalhado é relativamente pequena, assim, podendo ser ignorada.

Nessas condições, a Equação (3.19) é desconsiderada, e resolve-se somente a Equação (3.18). Desse modo a Equação de Helmontz(elíptica) pode ser simplificada por uma Equação parabólica mais simples, esse procedimento é conhecido como aproximação para-axial (VASCONCELOS, 2017).

A solução formal da Equação (3.18) é dada pela Equação (3.21)

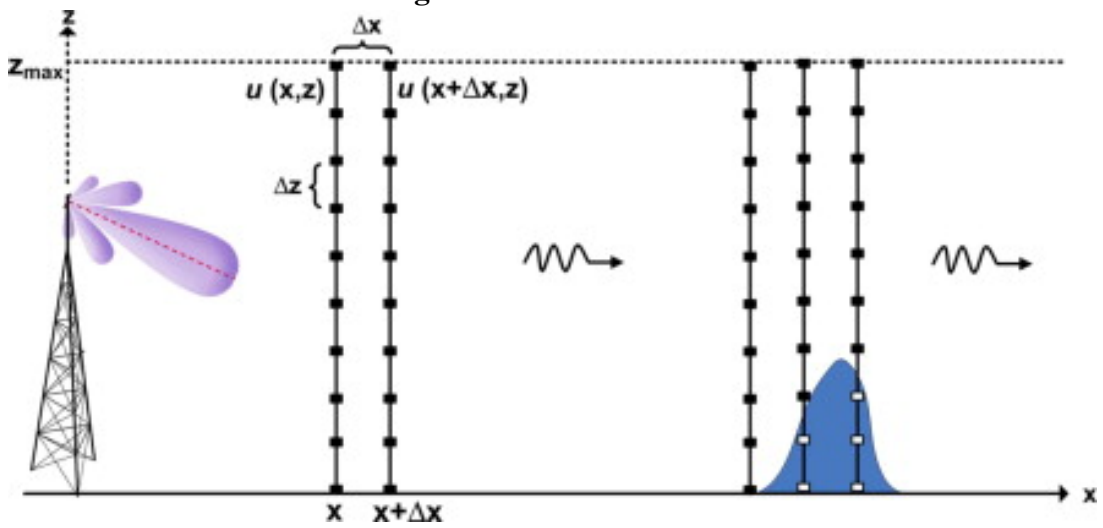
$$u(x+\Delta x, z) = e^{jk_0\Delta x(-1+Q)}u(x,z) \tag{3.21}$$

A componente de campo reduzida u é obtida na distância para-axial $x+\Delta x$, a partir dos valores do campo na distância anterior x, ressaltando a utilização de condições de contorno

apropriadas. A partir da solução inicial, é feito a iteração de Δx em Δx e são obtidos todos os valores do campo u na direção para-axial, portanto a solução $u(0,z)$ deve ser conhecida.

A Figura 3.2 ilustra como o método é solucionado numericamente, tendo conhecimento da função $u(x,z)$ encontra-se o valor em $u(x+\Delta x,z)$ e assim a solução vai "caminhando" na direção para-axial x.

Figura 3.2 – Metódo PE



Fonte: Ozgun *et al.* (2011)

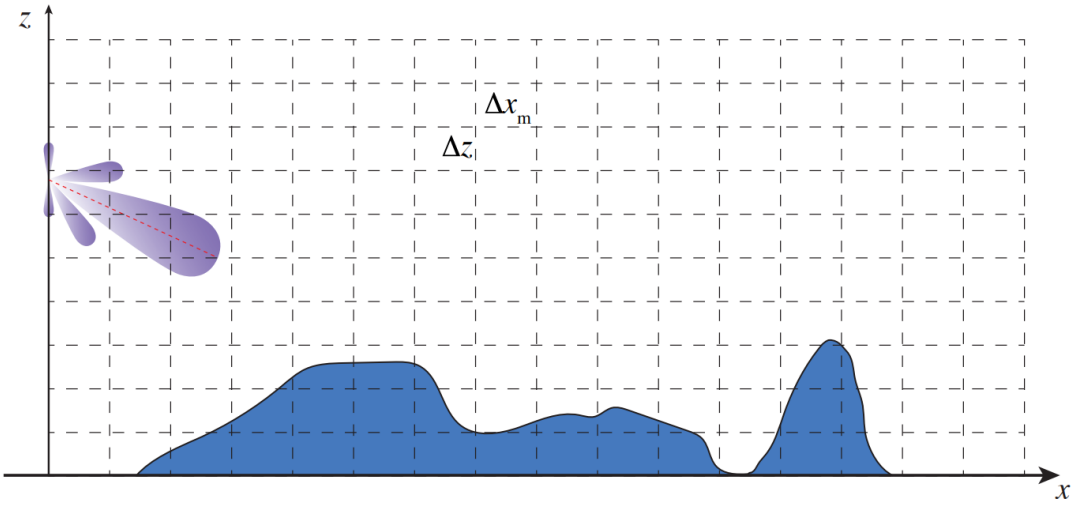
Esse método é computacionalmente mais fácil e mais rápido que a solução da Equação elíptica. Devido à fatoração da Equação (3.15), o método se limita a campos que estão dentro do cone mostrado na figura 3.1.

3.2 Diferenças Finitas e o Esquema de Crank-Nicolson

O esquema de Crank-Nicolson é um método numérico usado para resolver equações diferenciais parciais do tipo parabólico, como a Equação de calor. O método se baseia em uma combinação dos métodos de Euler explícito e trapezoidal, e é considerado um método de ordem temporal intermediária. Ele é estável e conservativo, o que significa que o valor total de uma grandeza física é mantido constante ao longo do tempo (CRANK; NICOLSON, 1947).

O primeiro passo para solução da Equação parabólica é discretizar o domínio, definindo uma malha de integração. A malha deve ser igualmente espaçada na direção de z mas na direção para-axial não precisa ser uniforme. A Figura 3.3 ilustra a malha de integração com Δz constante e Δx_m, onde cada índice m possui um Δx.

Figura 3.3 – Malha de integração



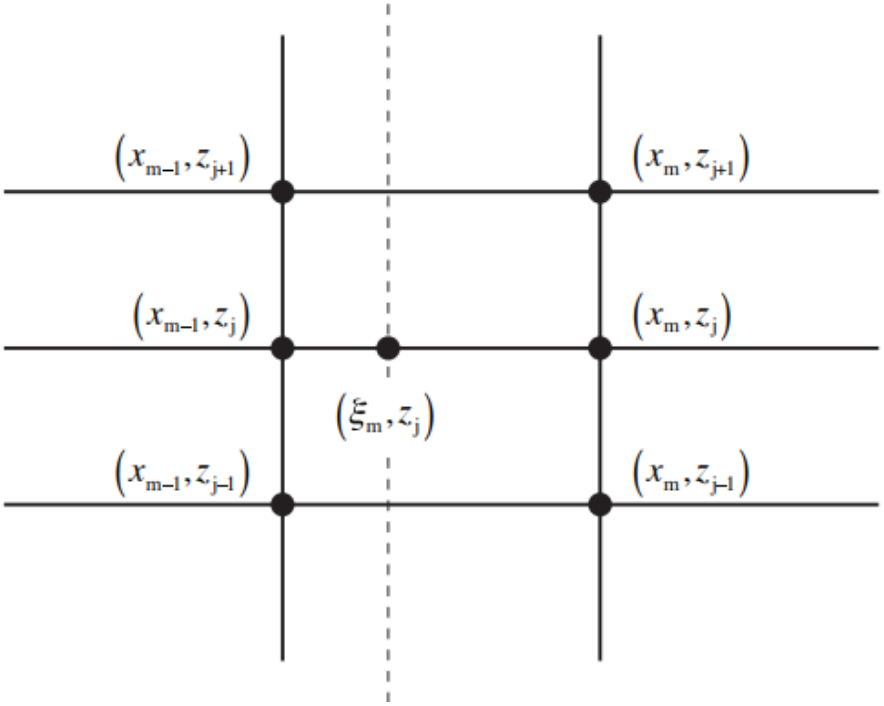
Fonte: Vasconcelos (2017)

Assim as distâncias discretas passam a ser representadas pela Equação (3.22).

$$\begin{cases} x_0 = 0 & m = 0 \\ x_m = x_{m-1} + \Delta x_m & m = 1,2,\ldots,M-1 \\ z_j = j\Delta z & j = 0,1,2,\ldots,N-1 \end{cases} \tag{3.22}$$

Considerando um ponto virtual entre cada passo de integração na direção para-axial como mostra a Figura 3.4

Figura 3.4 – Ponto virtual na Malha de integração



Fonte: Vasconcelos (2017)

O ponto intermediário (ξ_m, z_j) está em qualquer ponto entre (x_{m-1}, z_j) e (x_m, z_j), e é definido na Equação (3.23).

$$(\xi_m, z_j) = (x_{m-1} + \theta\Delta x_m, z_j) = (x_m - (1-\theta)\Delta x_m, z_j) \tag{3.23}$$

Em que θ é uma constante no intervalo [0, 1] e indica onde o ponto (ξ_m, z_j) está localizado em relação a (x_{m-1}, z_j) e (x_m, z_j).

Assim as derivadas podem ser avaliadas nesse ponto virtual, a derivada de primeira ordem na direção para-axial no ponto (ξ_m, z_j) pode ser definida por uma diferença finita central (DOE, 2010), como mostra a Equação (3.24).

$$\frac{\partial u}{\partial x}(\xi_m, z_j) = \frac{u(x_m, z_j) - u(x_{m-1}, z_j)}{\theta\Delta x_m + (1-\theta)\Delta x_m} = \frac{u(x_m, z_j) - u(x_{m-1}, z_j)}{\Delta x_m} \tag{3.24}$$

A derivada de segunda ordem em relação a z no ponto (ξ_m, z_j) é definida de modo semelhante, como mostra a Equação (3.25).

$$\frac{\partial^2 u}{\partial z^2}(\xi_m, z_j) = (1-\theta)\frac{\partial^2 u}{\partial z^2}(x_{m-1}, z_j) + \theta\frac{\partial^2 u}{\partial z^2}(x_m, z_j) \tag{3.25}$$

Em que as derivadas de segunda ordem nos pontos (x_{m-1}, z_j) e (x_m, z_j) são calculadas como diferenças finitas centrais pelas equações (3.26) e (3.27).

$$\frac{\partial^2 u}{\partial z^2}(x_{m-1}, z_j) = \frac{u(x_{m-1}, z_{j+1}) + u(x_{m-1}, z_{j-1}) - 2u(x_{m-1}, z_j)}{\Delta z^2} \tag{3.26}$$

$$\frac{\partial^2 u}{\partial z^2}(x_m, z_j) = \frac{u(x_m, z_{j+1}) + u(x_m, z_{j-1}) - 2u(x_m, z_j)}{\Delta z^2} \tag{3.27}$$

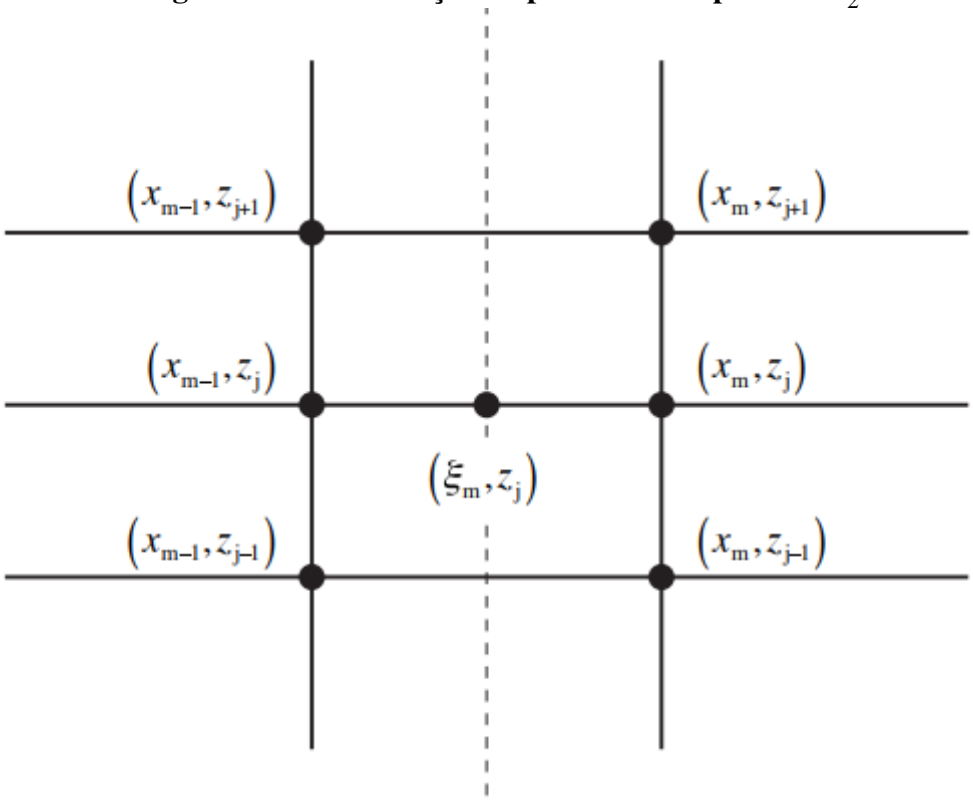
Substituindo as equações (3.26) e (3.27) na Equação (3.25) chega-se na Equação (3.28).

$$\frac{\partial^2 u}{\partial z^2}(\xi_m, z_j) = \frac{1}{\Delta z^2}\left[u(\xi_m, z_{j+1}) + u(\xi_m, z_{j-1}) - 2u(\xi_m, z_j)\right] \tag{3.28}$$

No esquema de Crank-Nicolson, o ponto virtual (ξ, z_j) está situado no ponto médio entre (x_{m-1}, z_j) e (x_m, z_j), ou seja, $\theta = \frac{1}{2}$. A Figura 3.5 mostra a localização do ponto virtual (ξ, z_j) .

A Equação (3.28) só pode ser usada no domínio $m \geq 1$ e $1 \leq j \leq N-2$. Para $m = 0$ não há problema, pois na coluna inicial, o campo é obtido pela distribuição inicial através de um Problema de Valor Inicial (PVI). Para $j = 0$ e $j = N-1$, condições de solo e fronteira superior respectivamente, deve-se utilizar outras expressões incorporando as condições de contorno adequadas.

Figura 3.5 – Localização do ponto virtual para $\theta = \frac{1}{2}$



Fonte: Vasconcelos (2017)

3.3 Método 1 - ângulo estreito

Considerando a equação parabólica 3.18 para propagação troposférica em Terra plana (VASCONCELOS, 2017). Primeiro adiciona-se e subtrai 1 no operador Q, para então expandir a raiz em série de Taylor de primeira ordem do tipo $\sqrt{1+A} \cong 1 + \frac{A}{2}$. Assim obtém-se a Equação (3.29).

$$\frac{\partial^2 u}{\partial z^2}(x,z) + 2jk_0 \frac{\partial u}{\partial x}(x,z) + k_0^2 \left[\tilde{m}^2(x,z) - 1\right] u(x,z) = 0 \tag{3.29}$$

Em que $\tilde{m}$ é o índice de refração, mas pode ser modificado para aderir outras condições, como Terra esférica e relevo (VASCONCELOS, 2017).

Nesta etapa é utilizado a discretização no ponto ξ_m, z_j, a fim de obter clareza na escrita do texto é feito a troca da notação, onde o índice da coordenada x é representado como expoente e o índice da coordenada z é representado como índice, por exemplo $u(x_m, z_j) = u_j^m$.

O campo no ponto virtual (ξ_m, z_j) é calculado como ponto médio do campo u_j^m e u_j^{m-1} como mostra a Equação (3.30).

$$u(\xi_m, z_j) = \frac{u_j^m + u_j^{m-1}}{2} \tag{3.30}$$

Com a troca de notação a Equação (3.28) é reescrita como 3.31.

$$\frac{\partial^2 u}{\partial z^2}(\xi_m, z_j) = \frac{1}{2\Delta z^2}\left[u_{j+1}^m + u_{j+1}^{m-1} + u_{j-1}^m + u_{j-1}^{m-1} - 2\left(u_j^m + u_j^{m-1}\right)\right] \tag{3.31}$$

A derivada de primeira ordem calculada na Equação (3.24) é reescrita como 3.32.

$$\frac{\partial u}{\partial x}(\xi_m, z_j) = \frac{u_j^m - u_j^{m-1}}{\Delta x_m} \tag{3.32}$$

Substituindo as equações 3.30, 3.32 e 3.31 na Equação (3.29) e adotando as definições 3.33 e 3.34 para simplificação de escrita, obtém-se a Equação (3.35).

$$a_j^m = k_0^2 \Delta z^2 \left[\tilde{m}^2(\xi_m, z_j) - 1\right] \tag{3.33}$$

$$b^m = 4jk_0 \frac{\Delta z^2}{\Delta x_m} \tag{3.34}$$

$$u_{j+1}^m + u_{j-1}^m + u_j^m\left(-2 + b^m + a_j^m\right) = -u_{j+1}^{m-1} - u_{j-1}^{m-1} + u_j^{m-1}\left(2 + b^m - a_j^m\right) \tag{3.35}$$

Para facilitar a escrita da Equação (3.35) defini-se $\alpha_j^m = -2 + b^m + a_j^m$ e $\beta_j^m = 2 + b^m - a_j^m$, e a Equação (3.35) é rescrita como a Equação (3.36)

$$u_{j+1}^m + u_{j-1}^m + u_j^m \alpha_j^m = -u_{j+1}^{m-1} - u_{j-1}^{m-1} + u_j^{m-1}\beta_j^m \tag{3.36}$$

Para as regiões fora do domínio ($j = 0$ e $j = N - 1$) da Equação (3.35) devem ser utilizadas as condições de contorno nas fronteiras superior e inferior, e para $m = 0$ deve ser tratado como um problema de valor inicial (PVI).

3.3.1 Condições de fronteira inferior

O esquema de Crank-Nicolson precisa de condições de contorno para fazer o uso da Equação (3.35). Na fronteira ar/solo ($j = 0$) utiliza-se as condições de contorno de Leontovich apropriadas. Se a modelagem do solo for PEC(Permeabilidade Eletrocinética), ou seja, um condutor perfeito, utiliza-se as condições de contorno de Dirichlet ou Neumann (LEVY, 2000), conforme as equações 3.37 e 3.38.

$$u(x, 0) = 0 \text{ Dirichlet (pol. horizontal)} \tag{3.37}$$

$$\frac{\partial u}{\partial z}(x, 0) = 0 \text{ Neumann (pol. vertical)} \tag{3.38}$$

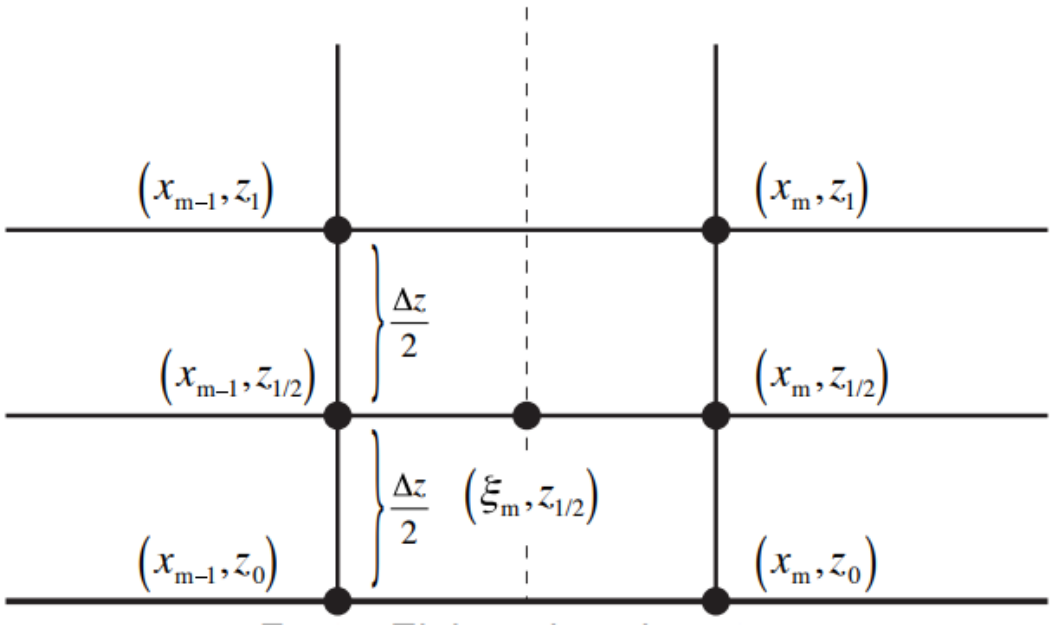
Para o caso de polarização horizontal, a condição de contorno de Dirichlet é imposta diretamente de modo que $u_0^m = 0$. Para a polarização vertical, a solução é mais elaborada.

A Equação (3.29) deve ser discretizada no ponto $(\xi_m, 0)$ e a expressão utilizada para representar a derivada de segunda ordem depende do ponto (ξ_m, z_{j-1}), conforme a Equação (3.39) mostra.

$$\frac{\partial^2 u}{\partial z^2}(\xi_m, 0) = \frac{1}{2\Delta z^2}\left[u_1^m + u_1^{m-1} + u_{-1}^m + u_{-1}^{m-1} - 2\left(u_0^m + u_0^{m-1}\right)\right] \tag{3.39}$$

Para contornar esse problema, é feito uma aproximação unilateral para a derivada de segunda ordem em vez de utilizar a Equação (3.39). Considerando o ponto médio $(\xi_m, z_{\frac{1}{2}})$ como ilustra a Figura 3.6.

Figura 3.6 – Esquema para aproximação unilateral na fronteira inferior



Fonte: Vasconcelos (2017)

Aproxima-se a derivada de segunda ordem no ponto $(\xi_m, 0)$ pela derivada de segunda ordem no ponto $(\xi_m, \frac{\Delta z}{2})$, como mostra a Equação (3.40).

$$\frac{\partial^2 u}{\partial z^2}(\xi_m, 0) \cong \frac{\partial^2 u}{\partial z^2}\left(\xi_m, \frac{\Delta z}{2}\right) \tag{3.40}$$

Portanto, é desenvolvida uma expressão para a segunda derivada no ponto $(\xi_m, \frac{\Delta z}{2})$. Como mostra as equações 3.41, e 3.42.

$$\frac{\partial^2 u}{\partial z^2}\left(\xi_m, \frac{\Delta z}{2}\right) = \frac{1}{2\left(\frac{\Delta z}{2}\right)^2}\left[u_1^m + u_1^{m-1} + u_0^m + u_0^{m-1} - \left(u_{\frac{1}{2}}^m + u_{\frac{1}{2}}^{m-1}\right)\right] \tag{3.41}$$

$$\frac{\partial^2 u}{\partial z^2}\left(\xi_m, \frac{\Delta z}{2}\right) = \frac{1}{\Delta z}\left[\frac{\left(u_1^m - u_{\frac{1}{2}}^m\right)}{\frac{\Delta z}{2}} + \frac{\left(u_1^{m-1} - u_{\frac{1}{2}}^{m-1}\right)}{\frac{\Delta z}{2}} - \frac{\left(u_{\frac{1}{2}}^m - u_0^m\right)}{\frac{\Delta z}{2}} - \frac{\left(u_{\frac{1}{2}}^{m-1} - u_0^{m-1}\right)}{\frac{\Delta z}{2}}\right] \tag{3.42}$$

Identificando expressões de diferenças finitas progressivas na Equação (3.42), então pode ser reescrita como a Equação (3.43).

$$\frac{\partial^2 u}{\partial z^2}\left(\xi_m, \frac{\Delta z}{2}\right) = \frac{1}{\Delta z}\left[\frac{\partial u}{\partial z}(x_m, \frac{\Delta z}{2}) + \frac{\partial u}{\partial z}(x_{m-1}, \frac{\Delta z}{2}) - \frac{\partial u}{\partial z}(x_m, 0) - \frac{\partial u}{\partial z}(x_m, 0)\right] \tag{3.43}$$

Utilizando a condição de Neumann, Equação (3.38), a expressão 3.43 se torna 3.44.

$$\frac{\partial^2 u}{\partial z^2}\left(\xi_m, \frac{\Delta z}{2}\right) = \frac{1}{\Delta z}\left[\frac{\partial u}{\partial z}(x_m, \frac{\Delta z}{2}) + \frac{\partial u}{\partial z}(x_{m-1}, \frac{\Delta z}{2})\right] \tag{3.44}$$

Calculando a expressão no ponto médio, e considerando a aproximação, obtém-se a Equação (3.45).

$$\frac{\partial^2 u}{\partial z^2}(\xi_m, 0) \cong \frac{\partial^2 u}{\partial z^2}\left(\xi_m, \frac{\Delta z}{2}\right) = \frac{1}{\Delta z^2}\left[u_1^m - u_0^m + u_1^{m-1} - u_0^{m-1}\right] \tag{3.45}$$

Desenvolvendo a Equação (3.39) utilizando a aproximação 3.45 obtém-se a Equação (3.46).

$$u_1^m + u_0^m\left(-1 + \frac{b^m + a_0^m}{2}\right) = -u_1^{m-1} + u_j^{m-1}\left(1 + \frac{b^m - a_0^m}{2}\right) \tag{3.46}$$

Em que a_0^m e b^m são definidos nas equações 3.33 e 3.34. Em termos de α_j^m e β_j^m a Equação (3.46) é escrita como 3.47.

$$u_1^m + u_0^m\frac{\alpha_0^m}{2} = -u_1^{m-1} + u_0^{m-1}\frac{\beta_0^m}{2} \tag{3.47}$$

Para considerar as condições de solos que não são condutores perfeitos($\sigma = \infty$), utiliza-se as condições de Cauchy (LEVY, 2000), deve-se substituir a condição na derivada unilateral da Equação (3.43). E assim obtém-se a Equação (3.48).

$$\frac{\partial^2 u}{\partial z^2}\left(\xi_m, \frac{\Delta z}{2}\right) = \frac{1}{\Delta z^2}\left[u_1^m + u_1^{m-1} - u_0^m(1 - \alpha_{BC}^m\Delta z) - u_0^{m-1}(1 - \alpha_{BC}^{m-1}\Delta z)\right] \tag{3.48}$$

Em que α_{BC}^{m} é definido na Equação (3.49) e δ^{m} é a impedância superficial do solo, mostrada na Equação (3.50).

$$\alpha_{BC}^{m} = jk_0\delta^{m} \tag{3.49}$$

$$\delta^{m}\begin{cases} \sqrt{\varepsilon_r'^{m} - j\frac{\sigma^m Z_0}{k_0} - 1} & \text{se polarização horizontal} \\ \frac{\sqrt{\varepsilon_r'^{m} - j\frac{\sigma^m Z_0}{k_0} - 1}}{\varepsilon_r'^{m} - j\frac{\sigma^m Z_0}{k_0}} & \text{se polarização vertical} \end{cases} \tag{3.50}$$

Assim obtém-se a solução da equação parabólica 3.29 no ponto (ξ_m, z_0), como mostra a Equação (3.51).

$$u_1^m + u_0^m\left[\frac{\alpha_0^m}{2} + \alpha_{BC}^m\Delta z\right] = u_1^{m-1} + u_0^{m-1}\left[\frac{\alpha_0^{m-1}}{2} + \alpha_{BC}^{m-1}\Delta z\right] \tag{3.51}$$

3.3.2 Condições da fronteira superior

Para condição da fronteira superior, o domínio será estendido adicionando uma camada absorvedora para representar o sinais se propagando até o infinito, como mostrado no capítulo 7 da tese Vasconcelos (2017). Portanto, pode-se modelar a fonteira superior como um espelho de reflexão total, pois a camada absorvedora ficará com o encargo de não permitir nenhuma reflexão. Assim utiliza-se as condições de contorno 3.52 e 3.53.

$$u(x, z_{N-1}) = 0 \text{ Dirichlet (pol. horizontal)} \tag{3.52}$$

$$\frac{\partial u}{\partial z}(x, z_{N-1}) = 0 \text{ Neumann (pol. vertical)} \tag{3.53}$$

Para a polarização horizontal basta apenas impor $u_{N-1}^m = 0$. Para a polarização vertical, de modo similar a obtenção da Equação (3.47), encontra-se a Equação (3.54).

$$u_{N-2}^m + u_{N-1}^m\frac{\alpha_{N-1}^m}{2} = -u_{N-2}^{m-1} + u_{N-1}^{m-1}\frac{\beta_0^m}{2} \tag{3.54}$$

3.4 Método 2 - ângulo amplo

Para o ângulo amplo na equação parabólica, considera-se a solução formal da equação parabólica, equações 3.55 e 3.56.

$$u(x+\Delta x_m,z) = e^{jk_0\Delta x_m(-1+Q)}u(x,z) \tag{3.55}$$

$$u(x+\Delta x_m,z) = e^{jk_0\Delta x_m(-1+\sqrt{1+Z})}u(x,z) \tag{3.56}$$

Em que o operador Z é definido pela Equação (3.57).

$$Z = \frac{1}{j_0^2}\frac{\partial^2}{\partial z^2} + \tilde{m}^2(x,z) - 1 \tag{3.57}$$

Expandindo o termo exponencial $e^{jk_0\Delta x_m(-1+\sqrt{1+Z})}$ como a Equação (3.58) mostra.

$$e^{jk_0\Delta x_m(-1+\sqrt{1+Z})} \cong \frac{P_2(Z)}{P_1(Z)} \tag{3.58}$$

A Equação (3.56) pode ser reescrita como as Equações (3.59) e (3.60).

$$u(x+\Delta x_m,z) = \frac{P_2(Z)}{P_1(Z)}u(x,z) \tag{3.59}$$

$$P_1(Z)u(x+\Delta x_m,z) = P_2(Z)u(x,z) \tag{3.60}$$

Utilizando o aproximador de Padé-(1,1) para a exponencial e equivalente à expansão de Taylor de segunda ordem é dado na Equação (3.61).

$$e^{jk_0\Delta x_m\left(-1+\sqrt{1+Z}\right)} \cong \frac{1+\frac{1}{4}(1-jk_0\Delta x_m)Z}{1+\frac{1}{4}(1+jk_0\Delta x_m)Z} \tag{3.61}$$

Fazendo a substituição da Equação (3.61) na Equação (3.60) chega-se na Equação (3.62).

$$\left[1+\frac{1}{4}(1+jk_0\Delta x_m)Z\right]u(x+\Delta x_m,z) = \left[1+\frac{1}{4}(1-jk_0\Delta x_m)Z\right]u(x,z) \tag{3.62}$$

Substituindo a expressão da Equação (3.57) na Equação (3.62) e aplicando a discretização em z por meio da diferença central, uma vez que em x já foi discretizado , obtém-se a Equação (3.63).

$$u_{j+1}^m + u_{j-1}^m + u_j^m\left(-2+a_j^m+b'^m\right) = c^m\left[u_{j+1}^{m-1} + u_{j-1}^{m-1} + u_j^{m-1}\left(-2+a_j^m+b'^{m*}\right)\right] \tag{3.63}$$

Em que:

$$a_j^m = k_0^2\Delta z^2[\tilde{m}^2(x,z)-1] \tag{3.64}$$

$$b'^m = \frac{4k_0\Delta z^2}{1+jk_0\Delta x_m} \tag{3.65}$$

$$c^m = \frac{1-jk_0\Delta x_m}{1+jk_0\Delta x_m} \tag{3.66}$$

A Equação (3.63) pode ser escrita como a Equação (3.67).

$$u_{j+1}^{m}+u_{j-1}^{m}+u_{j}^{m}\alpha_{j}^{'m}=c^{m}\left[u_{j+1}^{m-1}+u_{j-1}^{m-1}+u_{j}^{m-1}\alpha_{j}^{'m*}\right] \tag{3.67}$$

Em que:

$$\alpha_{j}^{'m}=-2+a_{j}^{m}+b^{'m} \tag{3.68}$$

Do mesmo modo que o ângulo estreito, a Equação (3.67) só é valida para os domínios $m \geq 1$ e $1 \leq j \leq N-2$, sendo necessário também condições de contorno para a solução.

Fazendo as considerações semelhante ao método de ângulo estreito, chega-se na Equação (3.69) para $j=0$.

$$u_{1}^{m}+\frac{\left(\alpha^{'m}+2\alpha_{BC}^{m}\Delta z\right)}{2}u_{0}^{m}=c^{m}\left[u_{1}^{m-1}+\frac{\left(\alpha^{'m*}+2\alpha_{BC}^{m}\Delta z\right)}{2}u_{0}^{m-1}\right] \tag{3.69}$$

Em que $\alpha_{BC}^{m}=jk_0\delta^{m}$ e δ^{m} é a impedância superficial do solo no ponto $(x_m,0)$ e é dado pela Equação (3.70).

$$\delta^{m}=\begin{cases}\sqrt{\varepsilon_{r}^{'m}-j\frac{\sigma^{m}Z_0}{k_0}-1} & \text{Polarização horizontal}\\ \\ \frac{\sqrt{\varepsilon_{r}^{'m}-j\frac{\sigma^{m}Z_0}{k_0}-1}}{\varepsilon_{r}^{'m}-j\frac{\sigma^{m}Z_0}{k_0}} & \text{Polarização vertical}\end{cases} \tag{3.70}$$

Para a fronteira superior, de forma análoga ao método de ângulo estreito, encontra-se a Equação (3.71).

$$u_{N-2}^{m}+\frac{\alpha_{N-1}^{'m}}{2}u_{N-1}^{m}=c^{m}\left[u_{N-2}^{m-1}+\frac{\alpha_{N-1}^{'m*}}{2}u_{N-1}^{m-1}\right] \tag{3.71}$$

Assim o sistema completo é mostrado na equação abaixo.

$$\begin{cases}u_{N-1}^{m}=0 & \text{Se } j=N-1 \text{ e Dirichlet}\\ u_{N-2}^{m}+\frac{\alpha_{N-1}^{'m}}{2}u_{N-2}^{m}=c^{m}\left[u_{N-2}^{m-1}+\frac{\alpha_{N-1}^{'m}2u_{N-2}^{m-1}}{2}\right] & \text{Se } j=N-1 \text{ e Dirichlet}\\ u_{j+1}^{m}+u_{j-1}^{m}+u_{j}^{m}\alpha_{j}^{'m}=c^{m}\left[u_{j+1}^{m-1}+u_{j-1}^{m-1}+u_{j}^{m-1}\alpha_{j}^{'m}\right] & \text{Se } 1\leq j\leq N-2\\ u_{0}^{m}=0 & \text{Se } j=0 \text{ e Dirichlet}\\ u_{1}^{m}+\frac{\alpha^{'m}}{2}u_{0}^{m}=c^{m}\left[u_{1}^{m-1}+\frac{\alpha^{'m*}}{2}u_{0}^{m-1}\right] & \text{Se } j=0 \text{ e Neumann}\\ u_{1}^{m}+\frac{\left(\alpha^{'m}+2\alpha_{BC}^{m}\Delta z\right)}{2}u_{0}^{m}=c^{m}\left[u_{1}^{m-1}+\frac{\left(\alpha^{'m}+2\alpha_{BC}^{m}\Delta z\right)}{2}u_{0}^{m-1}\right] & \text{Se } j=0 \text{ e Cauchy}\end{cases} \tag{3.72}$$

Em sua forma matricial o sistema pode ser escrito como:

$$AU^m = BU^{m-1} \tag{3.73}$$

Em que:

- A - Matriz tridiagonal em que $A_{j-1,j}$, $A_{j,j}$ e $A_{j+1,j}$ representa os fatores que multiplicam u^m_{j-1}, u^m_j e u^m_{j+1} respectivamente.
- U^m - M-ésima coluna do campo U.
- B - Matriz tridiagonal em que $B_{j-1,j}$, $B_{j,j}$ e $B_{j+1,j}$ representa os fatores que multiplicam u^{m-1}_{j-1}, u^{m-1}_j e u^{m-1}_{j+1} respectivamente.

3.5 Método 3 Mapa de Deslocamento de Ângulo para um Terreno definido por Partes

Este método (HOLM, 2007) aplica algumas aproximações para calcular as derivadas e condições de contorno a fim de resolver a Equação (3.18). Considerando a Equação parabólica (3.18) para propagação troposférica em Terra lisa (VASCONCELOS, 2017). Primeiro adiciona-se e subtrai 1 no operador Q, para então expandir a raiz em série de Taylor de primeira ordem do tipo $\sqrt{1+A} \cong 1+\frac{A}{2}$. Assim obtém-se a Equação (3.74).

$$\frac{\partial^2 u}{\partial z^2}(x,z) + 2jk_0\frac{\partial u}{\partial x}(x,z) + k_0^2\left[n^2(x,z)-1\right]u(x,z) = 0 \tag{3.74}$$

A expansão de primeira ordem que a série de Taylor fornece, limita a precisão dos resultados em uma região de apenas 15°, e a expansão de $\sqrt{1+Z}$ em ordem maiores apresenta instabilidades numéricas. Uma melhor escolha é a aproximação de Padé, Equação (3.75).

$$\sqrt{1+Z} = \frac{1+\frac{3}{4}Z}{1+\frac{1}{4}Z} \tag{3.75}$$

Utilizando essa aproximação chega-se na Equação (3.76), conhecida como Equação de Claerbout (CLAERBOUT, 1976).

$$\left(1+\frac{Z}{4}\right)\frac{\partial u}{\partial x} - ik\frac{Z}{2}u = 0 \tag{3.76}$$

Expandindo o termo Z na Equação (3.76), obtém-se a Equação (3.77).

$$\frac{\partial^3 u}{\partial x \partial z^2} - 2ik\frac{\partial^2 u}{\partial z^2} + k^2\left(n^2+3\right)\frac{\partial u}{\partial x} - 2ik^3\left(n^2-1\right)u = 0. \tag{3.77}$$

A Equação de Claerbout produz erros aceitáveis para uma região de um ângulo de até 45° na direção para-axial.

Aplicando as diferenças finitas e realizando as manipulações semelhantes aos métodos 1 e 2, chega-se no sistema de Equação matricial (3.78).

$$A_m U_m = B_m U_{m-1} \tag{3.78}$$

Em que a matriz B é definida como o conjugado da matriz A, como mostra a Equação (3.79).

$$B_m = A^*{}_m \tag{3.79}$$

Em que a matriz A é escrita pela Equação (3.80).

$$A_m = \begin{bmatrix} \alpha_0^m & c & 0 & 0 & \cdots & 0 & 0 & 0 \\ c & \alpha_1^m & c & 0 & \cdots & 0 & 0 & 0 \\ 0 & c & \alpha_2^m & c & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & c & \alpha_{N-2}^m & c \\ 0 & 0 & 0 & 0 & \cdots & 0 & c & \alpha_{N-1}^m \end{bmatrix} \tag{3.80}$$

Em que os coeficientes da matriz A são definidos nas Equações (3.81), (3.82), (3.83), (3.84) e (3.85).

$$\alpha_j^m = \begin{cases} D_j^m/2 + ikc\tilde{m}_j^m \Delta z & \text{Se } j = 0 \\ D_j^m & \text{Se } j = 1,2,3,\ldots \end{cases} \tag{3.81}$$

$$D_j^m = c\left(a_j^m + b - 2\right) \tag{3.82}$$

$$c = 1 - ik\Delta x \tag{3.83}$$

$$a_j^m = k^2\Delta z^2 \left[\left(\frac{\tilde{m}_j^m + \tilde{m}_j^{m-1}}{2} \right)^2 - 1 \right] \tag{3.84}$$

$$b = \frac{4k^2\Delta z^2}{c} \tag{3.85}$$

Em que $\tilde{m}^m$ representa o índice de refração modificado que incorpora a superfície da terra.

Porém, esse projeto utiliza uma solução diferente, onde a primeira solução u_0^m é calculada a partir de u_1^m, conforme a Equação (3.86) mostra.

$$U_0^m = \frac{U_1^m}{1 - k_m + 0.5 \cdot k_m^2} \tag{3.86}$$

Em que k_m é definido pela Equação (3.87).

$$k_m = \Delta z \cdot \left(\alpha_{BC}^m + j\frac{k_0}{2} \cdot \frac{dH(x_i)}{dx}^3 \right) \tag{3.87}$$

A matriz A é então definida como a Equação (3.88).

$$\mathbf{A}_m = \begin{pmatrix} \alpha_1^m & c & 0 & 0 & \cdots & 0 & 0 & 0 \\ c & \alpha_2^m & c & 0 & \cdots & 0 & 0 & 0 \\ 0 & c & \alpha_3^m & c & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & c & \alpha_{N-2}^m & c \\ 0 & 0 & 0 & 0 & \cdots & 0 & c & \alpha_{N-1}^m \end{pmatrix}. \tag{3.88}$$

3.6 Considerações finais

Neste capitulo foram apresentados as aproximações utilizadas para obtenção de um modelo discreto que permita a implementação computacional. As equações na forma matricial permite que a solução computacional seja escalável e confiável, desde que atenda as condições impostas pelas aproximações.

Para implementação das equações matriciais obtidas, é necessário um modelo de representação computacional para a matriz tridiagonal, também definir as operações de solução de sistema tridiagonal e multiplicação de sistema tridiagonal sem desperdiçar processamento e memória.

4 SISTEMA LINEAR TRIDIAGONAL

4.1 Introdução

Para que um sistema seja considerado tridiagonal, o número de equações (N) deve ser maior que três (3), e cada variável x_i depender apenas das variáveis x_{i+1} e x_{i-1}, como mostra a Equação 4.1.

$$a_i \cdot x_{i-1} + b_i \cdot x_i + c_i \cdot x_{i+1} = d_i \tag{4.1}$$

Escrito em sua forma matricial como mostra a Equação 4.2.

$$A \cdot X = D \tag{4.2}$$

$$\begin{bmatrix} b_0 & c_0 & 0 & 0 & \cdots & 0 \\ a_1 & b_1 & c_1 & 0 & \cdots & 0 \\ 0 & a_2 & b_2 & c_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{N-1} & b_{N-1} \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{bmatrix} = \cdot \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ \vdots \\ d_{N-1} \end{bmatrix} \tag{4.3}$$

Como todos os elementos da matriz A são zeros, com exceção da diagonal principal, diagonal superior e diagonal inferior, não é necessário utilizar memória e nem computação com esses elementos.

4.2 Algoritmo de Thomas

Llewellyn Hilleth Thomas propôs um algoritmo, que ficou conhecido como Algoritmo de Thomas, para resolver o sistema com menos operações do que a eliminação de Gauss. A solução é obtida a partir da eliminação gaussiana genérica, substituindo as equações até que se obtenha uma Equação que dependa apenas da variável x_{N-1}, e assim encontrando as soluções das equações anteriores.

O algoritmo se inicia alterando os coeficientes da matriz A, em que os coeficientes da diagonal superior devem ser alterados para c'_i como mostra a Equação (4.4).

$$c'_i = \begin{cases} \frac{c_i}{b_i} & ; i = 0 \\ \frac{c_i}{b_i - c'_{i-1} \cdot a_i} & ; i = 1,2,3,4,5,...,N-1 \end{cases} \tag{4.4}$$

Os coeficientes do vetor D também devem ser alterados para d', conforme a Equação (4.5).

$$d'_i = \begin{cases} \frac{d_i}{b_i} & ; i = 0 \\ \frac{d_i - d'_{i-1} \cdot a_i}{b_i - c'_{i-1} \cdot a_i} & ; i = 1,2,3,4,5,...,N-1 \end{cases} \tag{4.5}$$

Ao final, d'_{N-1} é a solução de x_{N-1} e assim encontra os valores do vetor X.

$$x_i = \begin{cases} d'_i & ; i = N-1 \\ d'_i - c'_i \cdot xi + 1 & ; i = N-2, N-3, N-4, ..., 0 \end{cases} \tag{4.6}$$

Embora a solução apresente maior eficiência que a eliminação de gauss, para sistemas muito grandes a solução demora um tempo considerável de processamento. As Equações 4.4 e (4.5) podem ser realizadas em conjunto em uma mesma iteração, mas o método não permite que as equações sejam realizadas de modo paralelo. Como menciona o capítulo 5, essas equações não são independentes entre si, a variável c'_i depende da variável c'_{i-1}, assim como d'_i depende de d'_{i-1} e x_i depende de x_{i+1}. Portanto, o algoritmo de Thomas não pode ser paralelizado.

4.3 Algoritmo Parallel Cyclic Reduction

O Algoritmo de redução cíclica *Cyclic Reduction* (CR) inventado por G. H. Golub e R. W. Hockney nos anos de 1960, é uma solução linear para sistemas de diferenças finitas aplicados a Equação de Poisson (KULKARNI, 2021), introduzido como uma particularidade da fatoração LU sem pivoteamento para solução de sistemas tridiagonais lineares. Em (AMODIO; MAZZIA, 1976) é mostrado que o algoritmo Parallel Cyclic Reduction (PCR), o algoritmo CR executado de modo paralelo, possui as mesmas propriedades que a fatoração LU sem pivoteamento, com a adição de algumas condições quanto ao número de equações. Para convergência, o número de equações deve ser $N = 2^k - 1$, onde $k = 2,3,4,...$ (AMODIO, 1992).

Partindo da Equação 4.1, em que $i = 0,1,2,...,2^k - 2$ e $a_0 = 0, c_{N-1} = 0$. Quando N não satisfizer a condição, adicionam-se as equações "fictícias" 4.7, apenas para satisfazer a condição de $N = 2^k - 1$.

$$x_i = 1 \tag{4.7}$$

Em que os termos de a_i, b_i, c_i e d_i são descritos pela Equação (4.8).

$$\begin{aligned} a_i &= 0 \\ b_i &= 1 \\ c_i &= 0 \\ d_i &= 1 \end{aligned} \tag{4.8}$$

Em cada iteração, eliminam-se as equações imediatamente subjacentes e adjacentes. Para a primeira iteração elimina-se as variáveis com índices pares $mod(i,2) = 0$, restando as equações $i = 1,3,5,7,...,2^k - 3$. Para cada 3 equações a eliminação é feita multiplicando as três equações por α_i, β_i e γ_i respectivamente;

$$\begin{aligned} \alpha_i \cdot a_{i-2} \cdot x_{i-2} + \alpha_i \cdot b_{i-1} \cdot x_{i-1} + \alpha_i \cdot c_{i-1} \cdot x_i &= \alpha_i \cdot d_{i-1} \\ \beta_i \cdot a_i \cdot x_{i-1} + \beta_i \cdot b_i \cdot x_i + \beta_i \cdot c_i \cdot x_{i+1} &= \beta_i \cdot d_i \\ \gamma_i \cdot a_{i+1} \cdot x_i + \gamma_i \cdot b_{i+1} \cdot x_{i+1} + \gamma_i \cdot c_{i+1} \cdot x_{i+2} &= \gamma_i \cdot d_{i+1} \end{aligned} \tag{4.9}$$

Combinando as três equações e tomando $\beta_i = 1$, $\alpha_i \cdot b_i + \beta_i \cdot a_i = 0$ e $\beta_i \cdot c_i + \gamma_i \cdot b_{i+1} = 0$, chega-se a Equação (4.10).

$$\begin{aligned} & x_{i-2} \cdot \alpha_i \cdot a_{i-2} \\ & + x_{i-1} \cdot (\alpha_i \cdot b_{i-1} + \beta_i \cdot a_i) \\ & + x_i \cdot (\alpha_i \cdot c_{i-1} + \beta_i \cdot b_i + \gamma_i \cdot a_{i+1}) \\ & + x_{i+1} \cdot (\beta_i \cdot c_1 + \gamma_i \cdot b_{i+1}) \\ & + x_{i+2} \cdot \gamma_i \cdot c_{i+1} \\ & = \alpha_i \cdot d_{i-1} + \beta_i \cdot d_i + \gamma_i \cdot d_{i+1} \end{aligned} \tag{4.10}$$

Removendo os termos nulos, a Equação (4.10) é escrita como na Equação (4.11).

$$\begin{aligned} x_{i-2} \cdot \alpha_i \cdot a_{i-2} + x_i \cdot (\alpha_i \cdot c_{i-1} + \beta_i \cdot b_i + \gamma_i \cdot a_{i+1}) + x_{i+2} \cdot \gamma_i \cdot c_{i+1} \\ = \alpha_i \cdot d_{i-1} + \beta_i \cdot d_i + \gamma_i \cdot d_{i+1} \end{aligned} \tag{4.11}$$

Substituindo as novas constantes por $\hat{a}_i$, $\hat{b}_i$, $\hat{c}_i$ e $\hat{d}_i$, obtém-se a Equação (4.12).

$$\hat{a}_i \cdot x_{i-2} + \hat{b}_i \cdot x_i + \hat{c}_i \cdot x_{i+2} = \hat{d}_i \tag{4.12}$$

Em que:

$$\hat{a}_i = \alpha_i \cdot a_{i-2} \tag{4.13}$$

$$\hat{b}_i = (\alpha_i \cdot c_{i-1} + \beta_i \cdot b_i + \gamma_i \cdot a_{i+1}) \tag{4.14}$$

$$\hat{c}_i = \gamma_i \cdot c_{i+1} \tag{4.15}$$

$$\hat{d_i} = \alpha_i \cdot d_{i-1} + \beta_i \cdot d_i + \gamma_i \cdot d_{i+1} \tag{4.16}$$

Para os casos particulares $i = 1$ e $i = 2^k - 3$, $\hat{a}_1 = 0$ e $\hat{c}_{2^k-3} = 0$. Pois as equações $i = 1$ e $i = 2^k - 3$ dependem somente de outra variável remanescente x_3 e x_{2^k-5} respectivamente. Portanto $\hat{a}_1 = 0$ e $\hat{c}_{2^k-3} = 0$.

O novo sistema obtido contém $2^{k-1} - 1$ equações, então é feito novamente a eliminação nesse novo sistema até obter um sistema de tamanho $2^1 - 1 = 1$. A Equação do sistema totalmente reduzido será do tipo da Equação (4.17).

$$\hat{b}_{2^{k-1}} \cdot x_{2^{k-1}} = \hat{d}_{2^{k-1}} \tag{4.17}$$

Para cada variável encontrada, obtém-se as duas variáveis eliminadas no passo de redução anterior, e assim é encontrado a solução do sistema. A vantagem desse método em relação ao algoritmo de Thomas é a não dependência dos coeficientes $\hat{a}_i$, $\hat{b}_i$, $\hat{c}_i$ e $\hat{d_i}$, ou seja, $\hat{a}_i$ não depende de $\hat{a}_l$, onde $i \neq l$, em uma mesma redução.

O número de reduções de equações, $N_{\text{reduções}}$, é calculado pela Equação (4.18).

$$N_{\text{reduções}} = \log_2 (N+1) - 1 = k - 1 \tag{4.18}$$

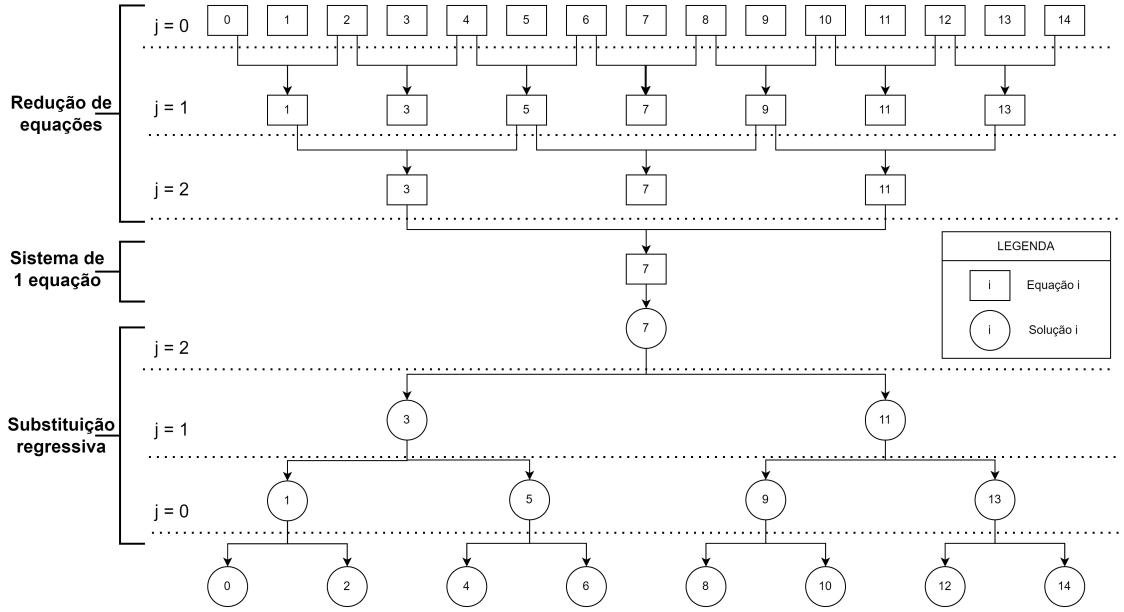
As equações combinadas forma o algorítimo 1, que pode ser implementado utilizando vários processadores em conjunto.

A Figura 4.1 ilustra a solução de um sistema de 15 equações, utilizando o algoritmo PCR. São feitas 3 reduções $j = 0, 1, 2$, em seguida a solução do sistema de uma única equação e a partir das 3 substituições regressivas $j = 0, 1, 2$, determina as demais variáveis. As 15 equações foram reduzidas para 7 equações, com os índices 1, 3, 5, 7, 9, 11 e 13. O novo sistema de 7 equações é reduzido para 3 equações com os índices 3, 7 e 11. As equações 3 e 11 são eliminadas nesse último passo da redução e, assim, o sistema foi reduzido para uma única equação conforme a Equação (4.19) mostra.

$$\hat{b}_7 \cdot x_7 = \hat{d}_7 \tag{4.19}$$

Finalizada a etapa de redução de equações, é então encontrada a solução para a variável x_7. Nesse último sistema reduzido a solução é bem simples, bastando dividir $\hat{d}_7$ por $\hat{b}_7$

Figura 4.1 – Diagrama de redução cíclica para um sistema de 15 equações.



Fonte: Elaborado pelo autor.

conforme a Equação (4.20).

$$x_7 = \frac{\hat{d}_7}{\hat{b}_7} \tag{4.20}$$

Por fim, é iniciado a etapa de substituição regressiva, encontrando as variáveis x_3 e x_{11}. Substituindo a variável x_3, são encontrados as variáveis x_1 e x_5, substituindo x_{11}, encontra-se x_9 e x_{13}. Essa etapa é feita até que todas as variáveis tenham sido determinadas.

A eficiência do algoritmo dependerá da implementação e do número de threads utilizadas. O algoritmo de redução cíclica é de simples entendimento quando tratado de forma sequencial. Em uma implementação paralela, é preciso dar especial atenção a alguns detalhes. Apenas o laço de repetição *i* pode ser resolvido de forma paralela e uma implementação ingênua poderia criar uma rotina paralela para resolver esse laço e a cada iteração de j chamar essa rotina. Esse tipo de implementação pode ser ainda mais lento do que uma implementação do algoritmo de Thomas. O ideal é que seja criado apenas uma rotina paralela e essa termine quando a execução completa terminar, extraindo todo potencial do algoritmo.

Algoritmo 1: Redução cíclica paralelizável

Requisito: Tamanho do sistema $N = 2^k - 1$
Entrada: Vetores a[], b[], c[] e d[] representando diagonal inferior, diagonal principal e diagonal superior da matriz A e o vetor D[], i= 0, 1, ... N - 1
Saída: Tamanho da matriz: $N = 2^k - 1$, vetores a[], b[], c[], d[]
início
  %Redução de equações;
  para $j = 0$; $j < log_2(N+1) - 1$; $j = j + 1$ **faça**
    $offset \leftarrow 2^j$;
    $step \leftarrow 2^{j+1}$;
    para $i = step - 1$; $i < N$; $i = i + step$ **faça**
      $id1 \leftarrow i - offset$;
      $id2 \leftarrow i + offset$;
      $alpha \leftarrow a[i]/b[id1]$;
      $gamma = c[i]/b[id2]$;
      $a[i] \leftarrow -a[id1] * alpha$;
      $b[i] \leftarrow b[i] - c[id1] * alpha - a[id2] * gamma$;
      $c[i] \leftarrow -c[id2] * gamma$;
      $d[i] \leftarrow d[i] - d[id1] * alpha - d[id2] * gamma$;
    fim
  fim
  % Resolver sistema de uma Equação;
  $id \leftarrow (N-1)/2$;
  $x[id] \leftarrow d[id]/b[id]$;
  % Substituição regressiva;
  para $j = log_2(N+1) - 2$; $j >= 0$; $j = j - 1$ **faça**
    $offset \leftarrow 2^{i+j}$ **para** $i = step - 1$; $i < N$; $i = i + step$ **faça**
      $id1 \leftarrow i - offset$;
      $id2 \leftarrow i + offset$;
      se $id1 - offset < 0$ **então**
        $x[id1] \leftarrow (d[id1] - c[id1] * x[id1 + offset])/b[id1]$;
      fim
      senão
        $x[id1] \leftarrow$
        $d[id1] - a[id1] * x[id1 - offset] - c[id1] * x[id1 + offset])/b[id1]$;
      fim
      se $id2 + offset \geq N$ **então**
        $x[id2] \leftarrow (d[id2] - a[id1] * x[id2 - offset])/b[id2]$;
      fim
      senão
        $x[id2] \leftarrow$
        $(d[id2] - a[id2] * x[id2 - offset] - c[id2] * x[id2 + offset])/b[id2]$;
      fim
    fim
  fim
fim

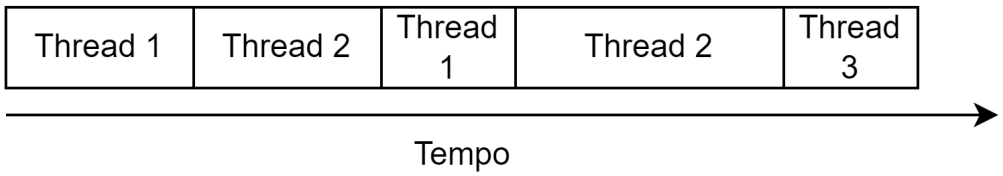
5 ARQUITETURA PARALELA

A arquitetura de execução de uma rotina computacional pode ser sequencial, paralela ou pseudo-paralela. Uma rotina é um conjunto de atividades que um processo de uma aplicação possui, as rotinas são mais conhecidas como *threads* , em tradução literal uma tarefa a ser executada. A arquitetura sequencial é a mais simples para os processadores com poucos recursos. O uso de *threads* tornou-se indispensável quando houve a necessidade de uma máquina executar diversas atividades simultaneamente ou pseudo-simultaneamente. Nessa arquitetura os processos são executados de forma paralela, todas as rotinas vão ser executadas sem a necessidade de que uma rotina finalize por completo. As *threads* são responsáveis por permitir a execução de diversos programas em uma mesma maquina, e sua execução pode ser feita por um processador exclusivo, onde somente essa *thread* é executada, ou por meio de concorrência.

5.1 Concorrência

As *threads* executadas em concorrência, disputam entre si para executar suas instruções, cada uma sendo executada por um intervalo de tempo variável conforme a prioridade e disponibilidade do processador. A Figura 5.1 ilustra três *threads* sendo executadas ao longo do tempo em um processador.

Figura 5.1 – threads executadas por um intervalo de tempo.



Fonte: Elaborado pelo autor.

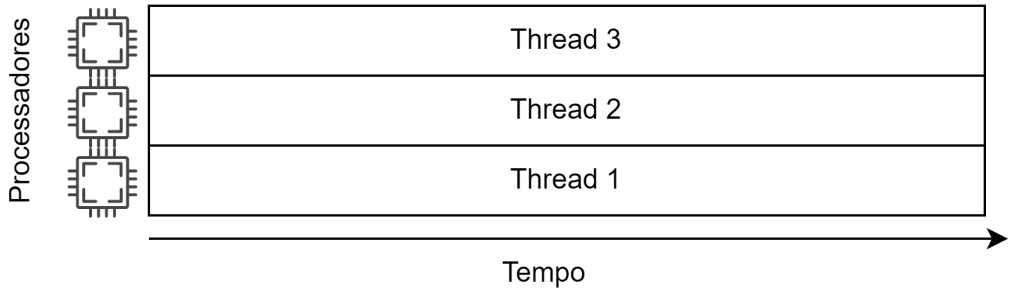
Para aplicações que não demandam alta prioridade, essa solução é extremamente satisfatória. O usuário não percebe as pausas que cada *thread* realiza e obtém a sensação de que as aplicações estão funcionando de modo paralelo, portanto o nome pseudo-paralelo.

5.2 Threads simultâneas

Um dos problemas gerados pela concorrência entre *threads* é a perda de desempenho para uma aplicação que precisa ser executada de modo paralelo. As *threads* simultâneas são

independentes entre si e ocupam recursos do hardware para execução de cada uma (KIRK; HWU, 2010).

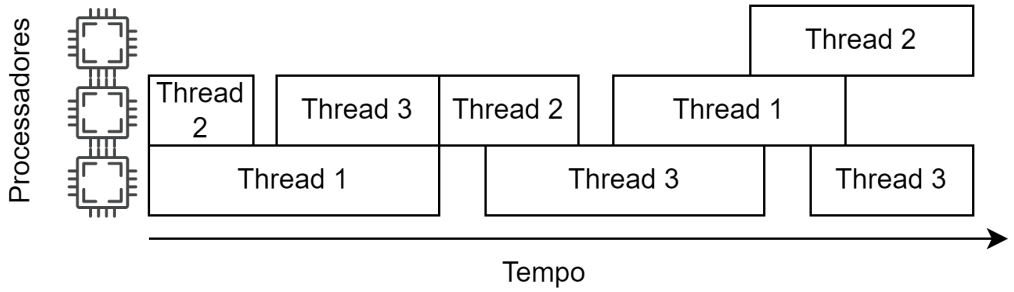
Figura 5.2 – threads executadas simultaneamente



Fonte: Autor.

A Figura 5.2 mostra a execução das *threads* simultaneamente, essas não possuem interrupções e executam continuamente no processador. Aplicações com *threads* com essa característica consomem mais energia, pois utilizam mais recursos do hardware. Os sistemas operacionais atualmente otimizam a utilização de recursos para manter um desempenho razoável na execução de suas aplicações e, ao mesmo tempo, não utilizar desnecessariamente os recursos que o hardware dispõe. A Figura 5.3 mostra um exemplo de como o sistema operacional com a disposição de 3 processadores, pode lidar em uma situação didática com apenas 3 *threads* que não executam com alta prioridade e nem continuamente, por exemplo, a *thread* 2 só é executada em certos eventos.

Figura 5.3 – Otimização do uso de recursos sem diminuir significadamente a sensação de paralelismo.



Fonte: Elaborado pelo autor.

5.3 Implementações matemáticas em computação paralela

Algumas operações com vetores e matrizes podem ser resolvidas simultaneamente, diminuindo o tempo de cálculo da solução. Vale lembrar que a utilização de *threads* concorrentes para soluções de equações matriciais obterá um desempenho inferior à solução sequencial.

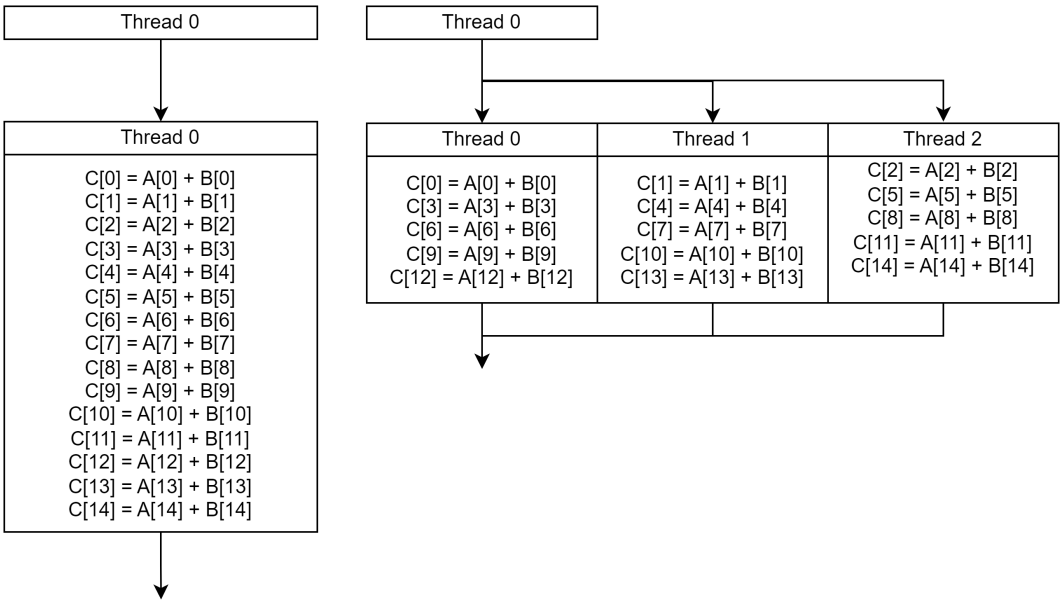
Uma operação matemática matricial só pode ser paralelizada se atender a seguinte condição.

- O elemento A_i deve ser independente de qualquer outro elemento A_j onde $i \neq j$.

O algoritmo de Thomas não pode ser paralelizado, pois, na Equação (4.4) a variável c'_i depende da variável c'_{i-1}, impossibilitando o cálculo paralelo. O mesmo acontece para as Equações (4.5) e (4.6). Já no algoritmo PCR, a obtenção dos novos coeficientes (4.16) independe dos demais coeficientes que estão sendo calculados na mesma redução, permitindo o cálculo simultâneo.

A Figura 5.4 mostra como uma operação de soma vetorial $C = A + B$ é realizado de modo sequencial e de modo paralelo.

Figura 5.4 – Distribuição de operações em execução sequencial à esquerda e execução paralela com três *threads* à direita



Fonte: Elaborado pelo autor.

Cada processador executa 1 *thread* e, assim, são executadas 3 *threads* simultâneas. Cada *thread* se responsabiliza por 5 sub*threads* que assumem um único índice do vetor, e de modo eficiente cada sub*thread* espera a conclusão da sub*thread* anterior para iniciar, não concorrendo

entre si. Ao final é feito o sincronismo entre os processadores até que todos finalizem suas *threads* e a operação é encerrada em menor tempo do que se fosse realizada por uma única thread. Esses conceitos são amplamente utilizados nas arquiteturas de processadores gráficos (GPU), que possuem diversos processadores para realizar rapidamente os processamentos que uma imagem, video ou áudio precisam.

Para solução da predição da perda de percurso, diversas equações podem ser paralelizadas, tais como o cálculo de coeficientes das matrizes A e B, a multiplicação de matriz tridiagonal, a solução do sistema tridiagonal, a multiplicação por escalar, a interpolação linear, etc.

5.3.1 Multiplicação de matriz tridiagonal

A multiplicação de duas matrizes, A e G, resultam em uma terceira matriz, C. Considerando a matriz A uma matriz quadrada de dimensão nxn, e a matriz G uma matriz coluna de n linhas. O elemento C_i é definido pela Equação (5.1).

$$C_i = \sum_{k=0}^{n} \left(A_{i,k} \cdot G_k \right) \tag{5.1}$$

Como em uma matriz tridiagonal a maioria dos elementos são nulos, com exceção das 3 diagonais principais, a Equação (5.1) pode ser simplificada para a Equação (5.2).

$$C_i = \begin{cases} b_i \cdot G_i + c_i \cdot G_{i+1} & \text{Se } i = 0 \\ a_i \cdot G_{i-1} + b_i \cdot G_i & \text{Se } i = n-1 \\ a_i \cdot G_{i-1} + b_i \cdot G_i + c_i \cdot G_{i+1} & \text{Caso contrário} \end{cases} \tag{5.2}$$

Em que os termos a_i, b_i e c_i são os coeficientes da diagonal inferior, diagonal principal e diagonal superior respectivamente.

Essa simplificação remove o somatório que seria implementado por um laço de repetição e também facilita a implementação utilizando os vetores a, b e c. Os casos particulares $i = 0$ e $i = n-1$ podem ser tratados separados, e os demais índices podem ser resolvidos paralelamente.

5.3.2 Interpolação linear

O terreno carregado, geralmente, possui uma taxa de amostragem muito baixa e o passo na direção para-axial é do tamanho de frações do comprimento de onda λ, portanto é

necessário a re-amostragem do terreno. Na teoria de sinais, um sinal pode ser reamostrado através da soma ponderada das amostras por um filtro (COUCH, 2012), como mostra a Equação (5.3).

$$y(t) = \sum_{i=0}^{N} y_a[i] \cdot h\left(t - \frac{i}{F_a}\right) \tag{5.3}$$

Em que :

- $y(t)$ é o sinal no tempo t.
- y_a é o sinal amostrado.
- N é o número de amostras do sinal y_a.
- F_a é a taxa de amostragem do sinal y_a.
- $h(t)$ é o filtro de reconstrução para o sinal.

Para a recuperação do sinal original, seria necessário infinitas amostras e a função sinc, Equação (5.4), como filtro.

$$sinc\,(t) = \lim_{x \to t} \frac{sen\,(\pi x)}{\pi x} \tag{5.4}$$

Porém, não é possível computacionalmente ter as infinitas amostras e nem representar a função sinc com precisão em todo domínio real. Uma boa aproximação é utilizar a função sinc e assumir o erro de ponto flutuante 10^{-14}. Assim, obtém-se um sinal muito próximo ao sinal original, com maior erro apresentado nas extremidades.

Uma outra abordagem é utilizar a função triângulo, Equação (5.5), que, na prática, equivale a ligar os pontos amostrados como uma interpolação linear.

$$\Delta(t) = \begin{cases} 1 - |t| & \text{Se } |t| \leq 1 \\ 0 & \text{Caso contrário} \end{cases} \tag{5.5}$$

Utilizando a função triângulo, chega-se em um caso particular da Equação (5.3), em que não é necessário realizar a soma de todos as amostras ponderadas.

Considerando que o sinal y_a na amostra de índice 0 corresponda ao tempo $t = 0$, e que a taxa de reamostragem do sinal é F_r, é válido a Equação (5.6).

$$y[i] = \begin{cases} y_a[j] \cdot (1 - t_j) + y_a[j+1] \cdot t_j & \text{Se } j+1 < N \\ y_a[j] \cdot (1 - t_j) & \text{Caso contrário} \end{cases} \tag{5.6}$$

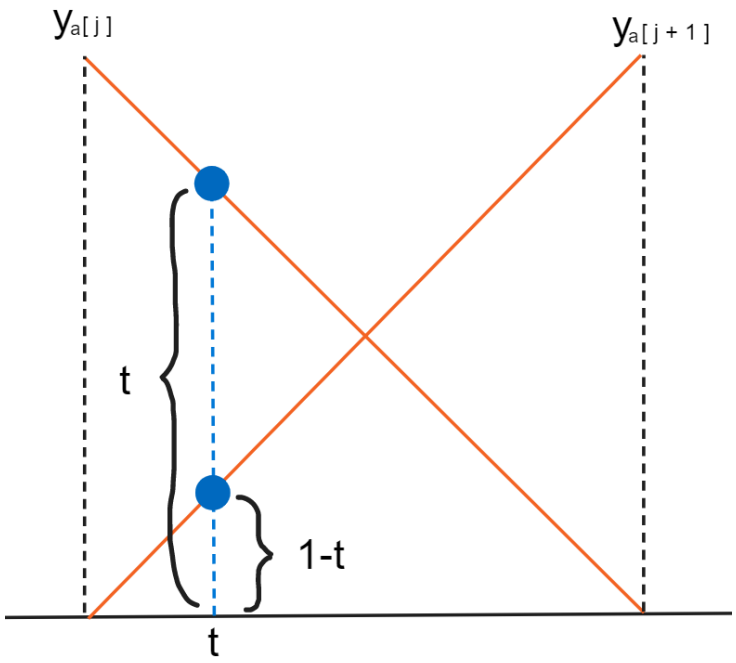
Em que:

$$t = \frac{i}{F_r} \tag{5.7}$$

$$j = floor(t \cdot F_a) \tag{5.8}$$

$$t_j = t - floor(t) \tag{5.9}$$

Figura 5.5 – Interpolação linear graficamente



Fonte: Elaborado pelo autor.

A Figura 5.5 ilustra o ponto t onde será feito a soma ponderada de $y_a[j]$ e $y_a[j+1]$.

Esse caso particular é muito útil para quando deseja-se realizar a interpolação linear dentro do domínio do sinal amostrado y_a.

5.3.3 Derivada por diferença central

A derivada de uma função no ponto é a taxa de variação no ponto em questão, geometricamente é a inclinação da reta que tangencia o ponto (GUIDORIZZI, 1994) , sua definição é dada na Equação (5.10).

$$\frac{df(t)}{dt} = \lim_{t_h->t} \frac{f(t_h) - f(t)}{t_h - t} = \lim_{h->0} \frac{f(t+h) - f(t)}{h} \tag{5.10}$$

Realizar aproximações para a Equação (5.10) podem gerar muitos erros conforme o ponto th se distância do ponto t, portanto é utilizado a derivada por diferença central (CHAPRA,

2008), que é definida pela Equação (5.11).

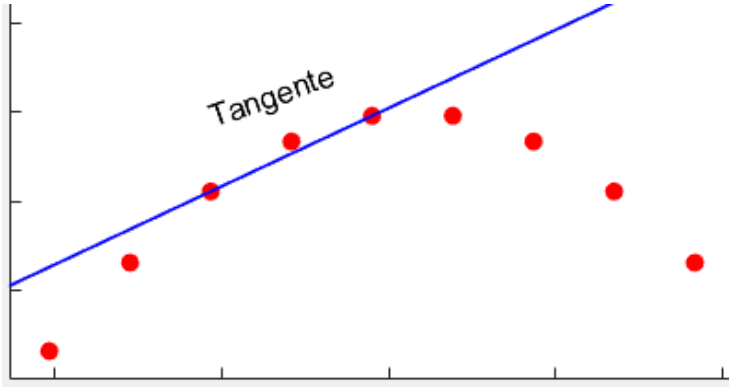
$$\frac{df(t)}{dt} = \lim_{h->0} \frac{f(t+h) - f(t-h)}{2h} \tag{5.11}$$

No limite as Equações (5.10) e (5.11) são equivalentes, porém para aproximações a derivada por diferença central apresenta menor erro se comparada a derivada lateral.

Considerando um sinal amostrado y_a com N amostras e taxa de amostragem F_s, a derivada central por diferença central pode ser aproximada pela Equação (5.12).

$$dy_a[i] = \frac{df(\frac{i}{F_s})}{dt} \approx (y_a[i+1] - y_a[i-1]) \cdot 2F_s \tag{5.12}$$

Figura 5.6 – Derivada por diferença central



Fonte: Autor.

A Figura 5.6 ilustra geometricamente como a derivada por diferença central é calculada. A partir da vizinhança do ponto i, encontra-se a tangente aproximada que passa no ponto i.

A Equação (5.12) não é válida para as extremidades, então é feito uma artimanha para aproximar a derivada. As Equações (5.13) e (5.14) calculam a derivada nas extremidades $i = 0$ e $i = N-1$, respectivamente.

$$dy_a[0] = [(y_a[0] - y_a[2]) \cdot 0.5 + (y_a[1] - y_a[0]) \cdot 2] \cdot F_s \tag{5.13}$$

$$dy_a[N-1] = [(y_a[N-1] - y_a[N-2]) \cdot 2 + (y_a[N-3] - y_a[N-1]) * 0.5] \cdot F_s \tag{5.14}$$

5.4 Considerações Finais

Neste capitulo foram abordados os fundamentos da arquitetura paralela, diferenciando *threads* paralelas e pseudo-paralelas. Definindo algumas operações matemáticas que

podem ser calculadas paralelamente.

Para configurar e utilizar as *threads* do *hardware*, pode ser utilizada uma API para facilitar e obter uma solução estável e confiável.

6 NVIDIA-CUDA

Em 2006, a NVIDIA, fabricante de placas de vídeo, lançou o CUDA como a primeira solução para computação geral nas GPUs. Antes disso, entre os anos de 1999 e 2000, cientistas da computação e pesquisadores de áreas como processamento de imagens médicas e eletromagnetismo, começaram a usar GPUs da NVIDIA para processar as aplicações cientificas. Utilizavam de técnicas para que suas aplicações se parecessem aplicações gráficas e, assim, conseguindo utilizar as linguagens de programação gráfica como OpenGL e Cg (FRANCO, 2012). Por se tratar de uma adaptação, não era possível extrair todo potencial das placas de vídeo. Observando as possibilidades de mercado, a NVIDIA investiu na modificação da GPU para torna-la totalmente programável e acrescentou suporte para as linguagens C, C++ e Fortran. Assim, criando a plataforma de computação paralela CUDA para GPUs. A partir disso, os desenvolvedores poderiam escrever códigos diretamente para a GPU e obter um desempenho muito maior.

6.1 Arquitetura de placas de vídeo compatíveis com CUDA

A arquitetura CUDA se baseia em arquitetura de memória compartilhada, na qual cada núcleo CUDA (*CUDA core*) tem acesso a uma memória global compartilhada, permitindo que os núcleos trabalhem em conjunto para processar os dados e compartilhar informações entre si (PILLA, 2009). A quantidade massiva de núcleos de uma GPU gera uma vantagem em cima da *Central Processing Unit* (CPU) em aplicações com grande volume de dados, como exemplo no processamento de imagens, áudios e vídeos.

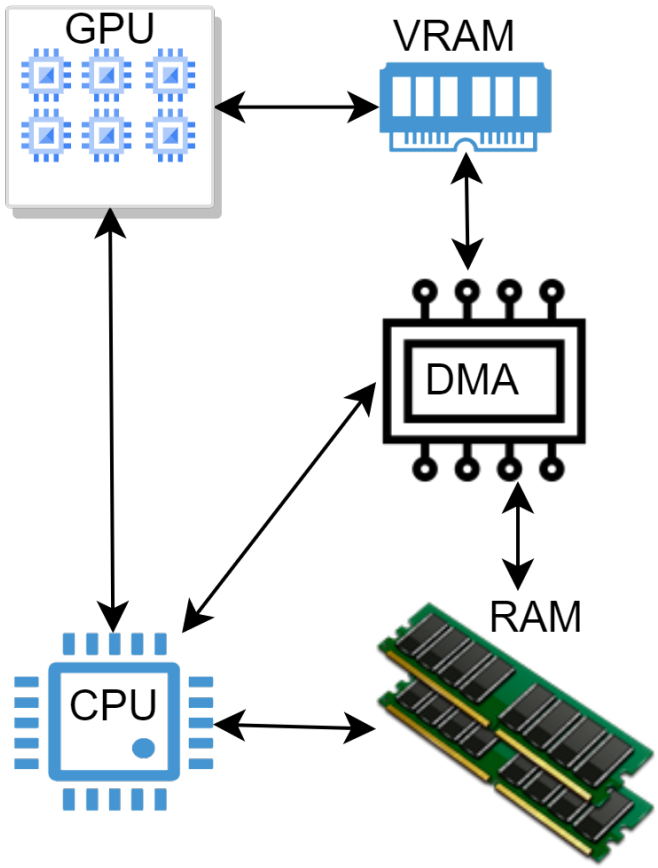
Os núcleos CUDA são organizados em conjuntos de blocos, e cada bloco possui vários núcleos que realizam diferentes operações de ponto flutuante. As rotinas destinadas aos núcleos CUDA são chamadas de kernels.

Os kernels em CUDA e suas chamadas, são compilados. Atribuindo um desempenho melhor do que se fosse uma chamada em tempo de execução, isto é, a montagem de cabeçalhos de cada kernel é compilada, não sendo necessário uso de funções auxiliares para realizar a passagem dos parâmetros para o kernel.

6.2 Acesso à memória

A GPU, para aplicações CUDA referenciado como “device”, possui sua própria memória *Video Random Access Memory* (VRAM), isto é, uma memória volátil de acesso rápido dedicada a placa gráfica. E em sua maioria, as placas de video não possuem acesso direto a memória *Random Access Memory* (RAM), como a CPU (referenciada como “host”) possui. Como a GPU não possui acesso direto, então deve ser realizado uma cópia dos dados para a memória VRAM, na maior parte das soluções é a etapa que consome maior tempo. Com o uso de *Direct Memory Access* (DMA) a cópia pode ser feita de modo assíncrono, e o processador se encarrega de sincronizar os dados. A Figura 6.1 ilustra a topologia de acesso de memória.

Figura 6.1 – Topologia de acesso à memória



Fonte: Elaborado pelo autor.

Antes de tudo deve-se ter em mente que se tratam de dois conjuntos de processadores isolados, com clocks, memórias e arquiteturas diferentes. A CPU agenda eventos na GPU, e por consultas verifica se os eventos agendados finalizaram, assim é realizado a sincronização.

Para a cópia de dados da memória RAM para VRAM, é utilizado o barramento (representado pelo circuito DMA) para transmitir as informações para a VRAM. Permitindo a cópia de modo assíncrono, enquanto as cópias dos dados estão sendo realizadas, o processador continua realizando outras atividades.

6.3 Kernels

Os kernels podem ser executados em vários núcleos simultaneamente, aumentando a eficiência do processamento. Além de que, CUDA permite que os kernels sejam executados em um grupo de núcleos e, assim, não ocupam todo o hardware.

Antes de inciar as implementações dos kernels, deve-se definir alguns conceitos para programação em CUDA. Uma placa de vídeo possui milhares de núcleos para realizar as tarefas, por objetivo de didática, é definido o número máximo de grupo de trabalho *Max Work Group* (MWG), grupo de trabalho global ou *Global Works* (GW), grupo de trabalho local *Local Works* (LW) e Thread, utilizados para realizar N operações em paralelas.

1. Número máximo de grupo de trabalho MWG, corresponde ao número máximo de processadores que permitirão a execução de $N_{threads}$ simultaneamente.
2. Grupo de trabalho global GW, representa o número máximo de *threads* a serem executadas. Pode ser chamado também como **itens de trabalho**. Deve ser múltiplo de LW.
3. Grupo de trabalho local LW, indica o número de processadores que serão utilizados na execução do kernel, sendo este maior ou igual a 1 e menor ou igual a MWG.
4. Thread, identifica qual índice dos GW. Pode ser chamado de **item de trabalho**.

A declaração de um kernel é muito semelhante a declaração de uma função em C/C++, diferenciando-se pela macro "__global__" antes do tipo de retorno. Essa macro indica ao compilador *NVIDIA CUDA Compiler* (nvcc) que a função em questão é um kernel e será executado pela GPU. Dentro do kernel deve-se tomar alguns cuidados, um único item de trabalho não pode passar de 5 segundos de execução, variáveis globais não são vistas pelo escopo do kernel, a sincronização de *threads* por barramento deve ser feita com cautela, etc (NVIDIA, 2007).

O código 1 mostra a criação de um kernel para realizar a derivada por diferença central, linhas 1 à 10, e a chamada para execução, linhas 11 à 18.

Código-fonte 1 – Kernel para calcular derivada

```
1  __global__ void cuda_kernel_derivada(double *dy, double *y,
       double fs, int64_t n) {
2    int64_t i = blockIdx.x*blockDim.x+threadIdx.x;
3    if (i >= n)return;
4    if (i == 0) {
5      dy[0] = ((y[0]-y[2])*0.5+(y[1]-y[0])*2)*fs;
6      dy[n-1] = ((y[n-1]-y[n-2])*2+(y[n-3]-y[n-1])*0.5)*fs;
7      return;
8    }
9    dy[i] = ((-y[i-1])+(y[i+1]))*fs*0.5;
10 }
11 int derivada(double *dy, double *y, double delx, int64_t n)
       {
12   double fs = 1. / delx;
13   size_t MWG = 1024;
14   size_t LW = n - 1 > MWG ? MWG : n - 1;
15   size_t GW = ceil((double) (n - 1) / LW);
16   cuda_kernel_derivada<<<GW, LW>>>(dy, y, fs, n);
17   return cudaDeviceSynchronize();
18 }
```

A chamada do kernel é feita na linha 16, diferenciando de uma chamada de função de C++ pela presença de "<<<GW , LW>>>", em que serão executados GW itens de trabalho, sendo LW itens executados simultaneamente. MWG é uma característica de cada *hardware*, para o *hardware* do exemplo, considera-se MWG= 1024. A verificação feita na linha 14, tem como intuito não utilizar núcleos desnecessariamente, que seus itens seriam capturados e descartados na linha 3 do kernel.

6.4 Biblioteca para matrizes Esparsas- cuSPARSE

Matrizes esparsas possuem grande parte de seus elementos nulos, não necessitando de memória para representa-los. A matriz tridiagonal é uma matriz esparsa, como mostrado no Capitulo 4, pois possui a maior parte de seus elementos nulos. A representação de seus elementos é feita pelos vetores a, b e c.

CUDA fornece a API cuSPARSE (NVIDIA, ; VALERO-LARA *et al.*, 2017), com um conjunto de funções para manipulação de matrizes esparsas. Dentre elas, algumas funções que implementam o algoritmo PCR e CR. A implementação fornecida pela API é extremamente otimizada e estável, portanto nesse projeto não foi implementado em CUDA a própria PCR.

6.5 Considerações Finais

Neste capitulo foi abordado as implicações de utilização de dois *hardwares* separados e com arquiteturas diferentes, deixando claro o problema de desempenho que as transferências de dados entre eles podem gerar.

Apresentado os conceitos de grupo local e grupo global, fundamental para utilização dos kernels em CUDA. Abordado, também, como são criados e executados os kernels na GPU.

7 DESENVOLVIMENTO

Todo desenvolvimento foi baseado na Tese de Vasconcelos (2017), em que o objetivo era apresentar os mesmos resultados, porém com um tempo menor.

O escopo do projeto é definido por:

- Entrada de dados: Por se tratar de um projeto envolvendo três plataformas diferentes OPENCL, OPENMP e CUDA, as entradas e saídas foram padronizadas para que um mesmo arquivo de entrada gerasse a mesma saída. Para isso, foi utilizado um arquivo no formato JSON (CROCKFORD, 2006) contendo os parâmetros necessários para a propagação. A escolha desse formato de arquivo foi pelo fato, de ser um formato robusto e bem definido, tirando a possibilidade ambiguidades durante a leitura.
- Cálculo de parâmetros iniciais: A entrada de dados possui alguns parâmetros que estão fora do *Sistema Internacional de Unidades* (SI), portanto é feito a conversão de unidades.
- Criação da camada absorvedora: A camada absorvedora foi utilizada para absorver as reflexões geradas pela condição de contorno utilizada no esquema de Crank-Neumann.
- Interpolação do relevo do Terreno: O terreno carregado não possui a taxa de amostragem necessária para implementação dos métodos, portanto é feito a interpolação linear aplicando a taxa de amostragem que a solução exige.
- Cálculo valores das impedâncias do solo: através dos parâmetros de entrada, são calculados os valores de α_{BC}.
- Geração de campo inicial: O programa gera até três tipos de campo inicial, são eles gerados por Fonte Gaussiana, Fonte retangular e fonte Omnidirecional (VASCONCELOS, 2017).
- Inicialização e manutenção das Matrizes A e B: Os coeficientes das Matrizes A e B são preenchidos, e somente os coeficientes que se alteram a cada iteração são reescritos novamente.
- Aplicação do PHASE-SHIFT: É aplicado a transformação do campo $u(x,z)$ para o campo $v(\xi,\zeta)$.
- Solução do sistema Tridiagonal: É utilizado o algoritmo de PCR para solucionar o sistema.
- Aplicação do PHASE-SHIFT-REVERSO: É aplicado a transformação do campo $v(\xi,\zeta)$ para o campo $u(x,z)$.
- Exportar modulo do campo $u(x,z)$: O valor de interesse é o fator de perda e a perda de percurso (VASCONCELOS, 2017), esses são obtidos tendo conhecimento do módulo do

campo $u(x,z)$. Ao final, o programa salva um arquivo contendo o modulo de $u(x,z)$ em todas a posições calculadas.

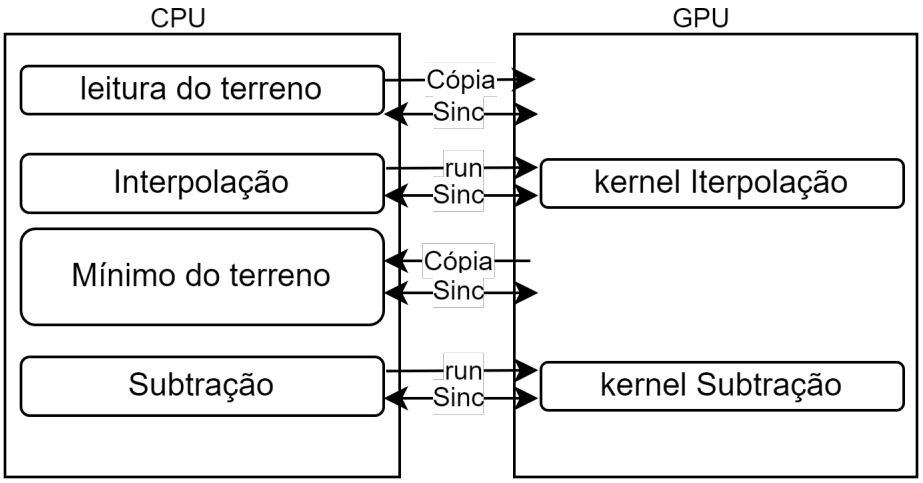
7.1 Implementação

Muitas implementações que utilizam CUDA para acelerar o processamento, acabam não obtendo um resultado satisfatório, podendo piorar o desempenho, como as matrizes GPU disponibilizadas pela ferramenta Matlab (SHURE, 2012). Isso ocorre devido às cópias realizadas entre memórias do host e do device, portanto, uma implementação com o mínimo de cópias realizadas garante o desempenho máximo.

Para esse projeto, os vetores e matrizes foram todos armazenados na memória de video, somente a GPU manipulando os valores por meio dos kernels. Atividades estritamente sequenciais, como a obtenção do valor máximo de um vetor, foram realizadas pelo processador. Desse modo, as cópias entre memórias foram reduzidas para somente quando necessário. Por outro lado, essa abordagem retirou toda a carga da CPU para a GPU, uma solução mais robusta seria utilizar tanto GPU e CPU para calcularem simultaneamente. A utilização de ambos hardwares aumentaria a complexidade de implementação, portanto foi utilizado somente a GPU. Outra limitação por utilizar somente a GPU é a memória disponível, pois as GPUs, em sua maioria, possui uma memória VRAM menor que a RAM, portanto, representando menos elementos de uma matriz.

A Figura 7.1 ilustra a ordem de execução de funções para preparação do perfil terreno.

Figura 7.1 – Diagrama de execução preparação do perfil de alturas do terreno.



Fonte: Elaborado pelo autor.

A execução é realizada de cima para baixo, iniciando na leitura do terreno. A leitura é efetuada totalmente pelo processador. Após a leitura dos dados, através da função auxiliar *cudaAllocMem*, a CPU se comunica com a GPU para reservar um espaço de memória na VRAM. Com a memória alocada no device, o processador utiliza outra função auxiliar, *cudaMemcpy*, para realizar a cópia dos valores carregados e armazenas no host para o device. A função *cudaMemcpy* faz o sincronismo internamente, para cópias assíncronas pode ser utilizado da função *cudaMemcpyAsync*. Nessa etapa ambos estão sincronizados e armazenam a mesma informação do terreno lido, e assim, iniciando a próxima etapa, a interpolação do terreno.

Para execução do kernel, a CPU enfileira na fila de execuções o kernel e seus parâmetros, informando também GW e LW. A GPU executa os kernels que estão na fila de execução, nesse momento a CPU está livre e realizando outras operações. Como a operação de obtenção do mínimo do terreno depende do resultado da interpolação, a CPU sincroniza com a GPU, ou seja, aguarda até que a fila de kernels da GPU esteja vazia.

Para finalizar a etapa de preparação do perfil terreno, a CPU faz a cópia device para host do perfil interpolado. Encontra o valor mínimo e enfileira o kernel para fazer a subtração do perfil do terreno pelo valor mínimo.

As classes do C++ foram utilizadas para abstração e facilitar a implementação, deixando claro se uma tarefa será executada na CPU ou na GPU. O algoritmo 2 mostra um trecho da implementação equivalente ao fluxograma da Figura 7.1.

Código-fonte 2 – Codigo Preparação do terreno

```
1  CPU::carregarTerreno(path_terreno, p_Terr);
2  p_Terr.sincD();
3  M = (lli) floor(max_range / delx) + 1;
4  p_Inte = VetorDH<double>(M);
5  slopes = VetorDH<double>(M);
6  GPU::interp1(p_Terr.dx, p_Terr.device, delx, p_Inte.device,
       p_Terr.z, M);
7  GPU::derivada(slopes.device, p_Inte.device, delx, M);
8  p_Inte.sincH();
9  zbase = CPU::minimo(p_Inte.Host(), M);
10 GPU::subtrair(p_Inte.device, zbase, M);
11 p_Inte.sincH();
```

As chamadas de funções precedidas por *"CPU"*, indicam que a execução será feita pela CPU, assim como as que são precedidas por *"GPU"* serão executadas pela GPU. As cópias são realizadas pelas funções *sincH* e *sincD*, cópia da VRAM para RAM e cópia da RAM para VRAM respectivamente.

8 RESULTADOS

Foram realizados alguns testes para validar o *software* criado, considerando como referência as implementações em Matlab. Os testes focaram em avaliar o desempenho e estabilidade na simulação, variando os parâmetros principais que mais afetam o tempo de simulação, como a frequência por estar relacionada com o passo na distância para-axial e o passo das alturas, e também a condição de salvar os dados discretos. No salvamento discreto a simulação é feita com o mesmo passo, porém os dados são salvos com um passo maior. O salvamento discreto reduz significadamente o uso de memória, permitindo simulações em grandes distâncias.

Para os testes foi utilizado um notebook PREDATOR PH315-53, com as seguintes especificações:

- GPU: NVIDIA RTX 2060 mobile, 1920 cuda cores, 1024 *threads*, VRAM de 6 GB.
- CPU: Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz
- RAM: 16 GB

8.1 Teste 1: Tempo de processamento sem relevo

Tabela 1 – Simulação sem relevo

Salvamento	Frequência (MHz)	Polarização	Distancia (m)	Tempo Matlab (s)	Tempo CUDA (s)	Eficiência Cuda x Matlab	Erro Máximo
Discreto	300	H	1000	10,288	0,555	18,54	6,99E-16
Completo	300	H	1000	9,323	0,538	17,34	6,99E-16
Discreto	300	V	1000	8,907	0,559	15,94	6,55E-16
Completo	300	V	1000	9,199	0,566	16,24	6,64E-16
Discreto	1875	V	10000	1161,236	19,614	59,20	3,25E-14
Discreto	700	H	65000	1403,327	34,086	41,17	1,41E-14
Discreto	700	V	65000	1409,967	34,595	40,76	1,38E-14

Fonte**:** Elaborado pelo autor

Para os testes mostrados na Tabela 1, foram utilizados os parâmetros:

- Modelo: Terra esférica sem relevo.
- Altura da fonte: 125 metros.
- Inclinação do lóbulo principal: 0°.
- Parâmetros do solo: solo seco.

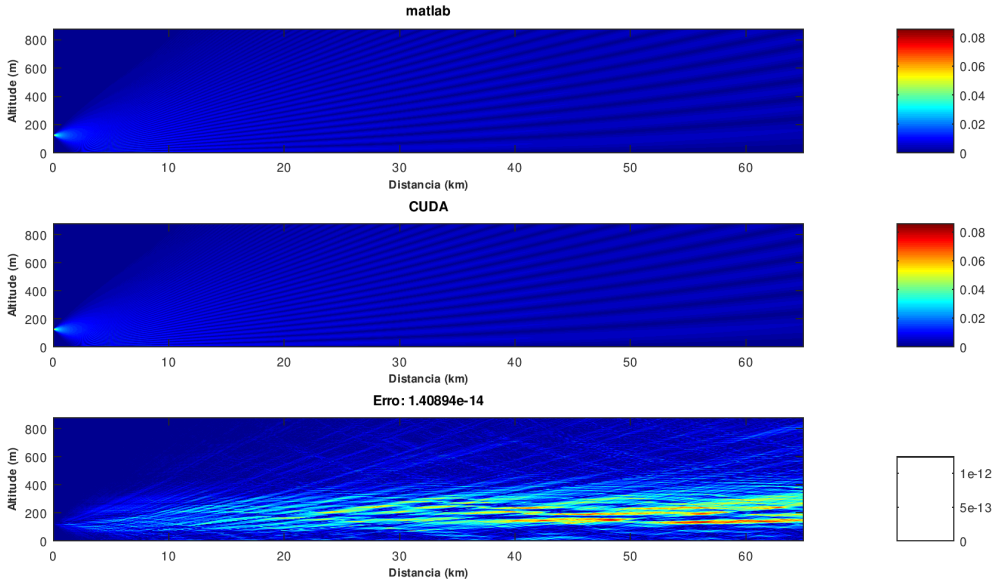
A Tabela 1 mostra uma comparação entre os tempos obtidos em CUDA e MATLAB. Para as frequências baixas, o software desenvolvido foi em torno de 18 vezes mais rápido. Já para as frequências mais altas, o desempenho foi ainda maior, por volta de 40 vezes para frequência

de 700 MHz.

Para frequências maiores, o comprimento de onda é menor e assim o passo de integração é menor, existindo mais equações para serem resolvidas. Nessa situação a GPU leva vantagem, por resolver mais elementos por unidade de tempo.

O erro obtido entre o software e o Matlab foi da ordem de 10^{-14}, como esperado para as operações de ponto flutuante.

Figura 8.1 – Simulação de 65 km, frequência de 700 MHz, Pol. horizontal



Fonte: Elaborado pelo autor.

A Figura 8.1 apresenta os resultados obtidos paralelamente por CUDA e sequencialmente pelo Matlab. Ambas as figuram apresentam o mesmo resultado, com a diferença do erro de ponto flutuante.

8.2 Teste 2 - Tempo de processamento Terra com relevo

Tabela 2 – Simulação Com relevo

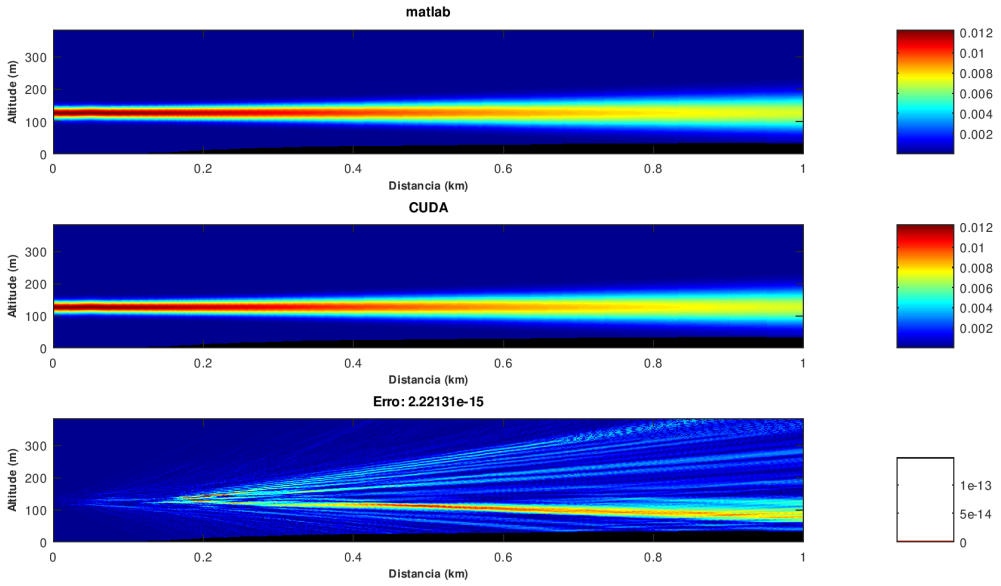
Salvamento	Frequência (MHz)	Polarização	Distancia (m)	Tempo Matlab (s)	Tempo CUDA (s)	Eficiencia Cuda x Matlab	Erro Máximo
Discreto	100	V	1000	9,82	0,95	10,37	2,00e-15
Completo	100	V	1000	9,98	0,97	10,25	2,00E-15
Discreto	100	H	65000	2815,01	126,95	22,17	2,00E-06

Fonte**:** Elaborado pelo autor

A adição do terreno aumenta o tempo da simulação, se comparado ao cenário de Terra

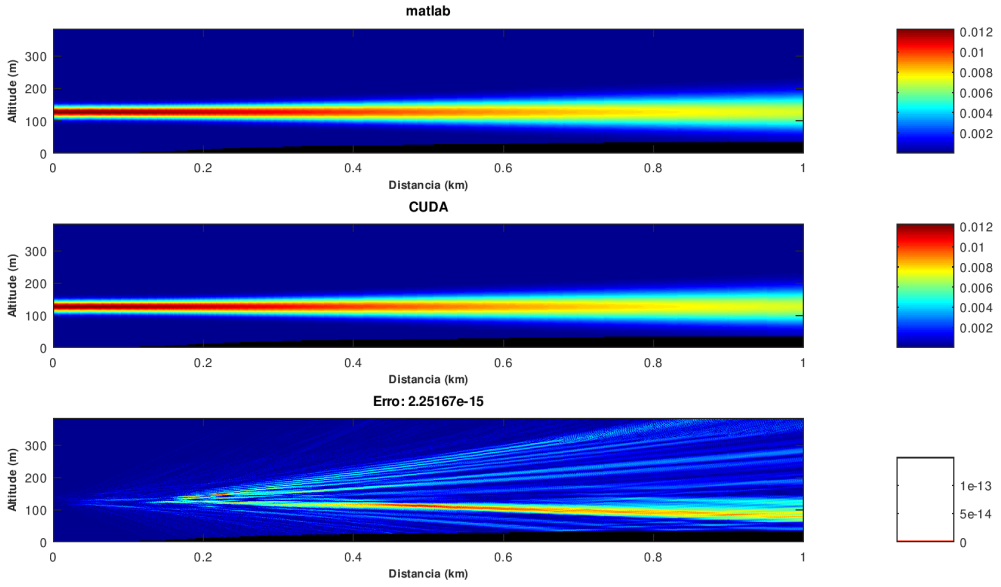
sem relevo. O modelo que implementa o relevo, aumenta o número de cálculos realizados. Para cada passo deve-se calcular os valores das matrizes A e B. Além disso, deve ser feito o PHASE-SHIFT e PHASE-SHIFT-REVERSO (HOLM, 2007), aumentando o tempo de processamento, e por essas duas ultimas operações utilizarem as funções seno e cosseno, aumentam o erro máximo para a ordem de 10^{-13}. A implementação dessas funções na GPU, não são tão precisas quanto as implementações das bibliotecas disponibilizadas para C++.

Figura 8.2 – Simulação de 1 km, Pol. V. Discreto.



Fonte: Elaborado pelo autor.

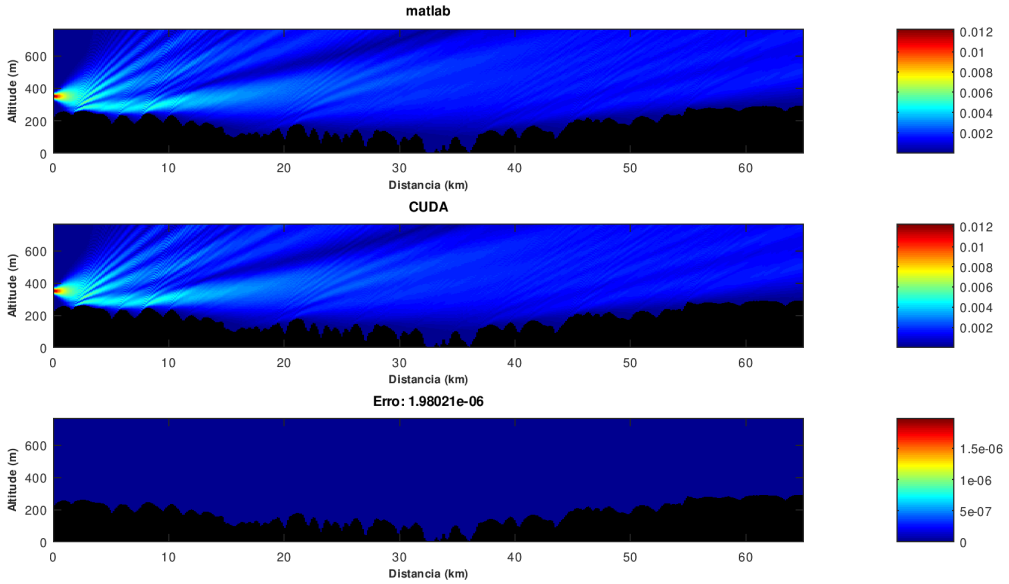
Figura 8.3 – Simulação de 1 km, Pol. V. Completo.



Fonte: Elaborado pelo autor.

As Figuras 8.2 e 8.3 comparam a utilização do salvamento discreto e completo, para os mesmos parâmetros. Analisando pela métrica das cores, os máximos e mínimos são mantidos, indicando que a utilização do salvamento com um passo maior pode ser utilizado.

Figura 8.4 – Simulação de 65 km, Pol. H. Discreto.



Fonte: Elaborado pelo autor.

A Figura 8.4 corresponde a terceira simulação da Tabela 2, da ideia de como é a propagação do sinal nesse cenário. O erro apresentado não é fidedigno. Analisando gráfico do erro, não é perceptível nenhum ponto em tom mais forte. Indicando que o método de avaliação pode ter considerado um ponto imperceptível entre as variações do relevo.

O desempenho de CUDA só não foi maior devido à frequência utilizada de 100 MHz, a escolha da frequência foi baseada no tempo de simulação do Matlab.

8.3 Estimativa do Custo Computacional

Considerando todas as operações feitas pelos kernels, e as operações realizadas em uma solução sequencial.

- O algoritmo da PCR possui custo computacional estimado de $2log_2(N)$.
- O algoritmo de Thomas pode ser executado com custo de $2N$.
- A multiplicação de uma matriz tridiagonal possui custo estimado de $\frac{N}{T}$ para solução paralela e N para solução sequencial.
- As outras operações com custo linear, quando sequencialmente, apresentam custo divido

pelas *threads*. Portanto, se o custo para execução sequencial for N, para a solução paralela pode ser estimado por $\frac{N}{T}$.

A partir dessas informações a Tabela 3 mostra o levantamento do custo operacional para cada operação, em que.

- N corresponde ao número de pontos discretizados da altura z.
- N_u corresponde ao número de pontos discretizados da altura z desconsiderando a camada absorvedora.
- M corresponde ao número de pontos discretizados da distância para-axial.
- T corresponde ao número de threads utilizada para realizar os cálculos.

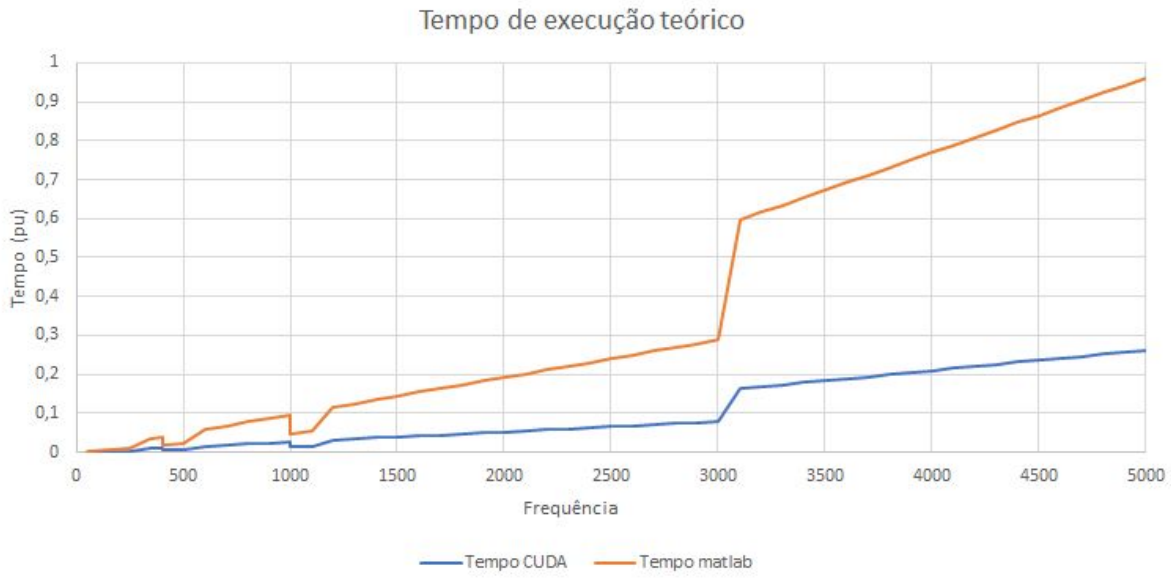
Tabela 3 – Levantamento teórico do custo computacional para as operações

Operação	Matlab	CUDA
Calcular Parâmetros	$6 \times M \times N$	$\frac{M \times N}{T}$
Multiplicar Matriz	N	$\frac{N}{T}$
Solução do Sistema	$3N$	$\frac{2Log_2(N)}{T}$
Salvar dados na matriz	$M \times N_u$	$\frac{MNu}{T}$

Vale lembrar que se trata de dois dispositivos diferentes, GPU e CPU, ambos possuem características que irão influenciar no tempo de cálculo. Por exemplo, a GPU possui um *clock* inferior em relação ao da CPU portanto, em soluções pequenas o processador pode ser mais rápido do que a GPU. Outros fatores que não está sendo considerado, são as operações com pontos flutuantes de dupla precisão. A GPU utilizada é voltada para jogos, portanto não é otimizada para operações com variáveis de tamanho de 64 bits.

A Figura 8.5 mostra o levantamento do tempo considerando os custos computacionais da Tabela 3. Os valores de tempo foram normalizados, pois o interesse está no formato da curva e não nos valores especificamente.

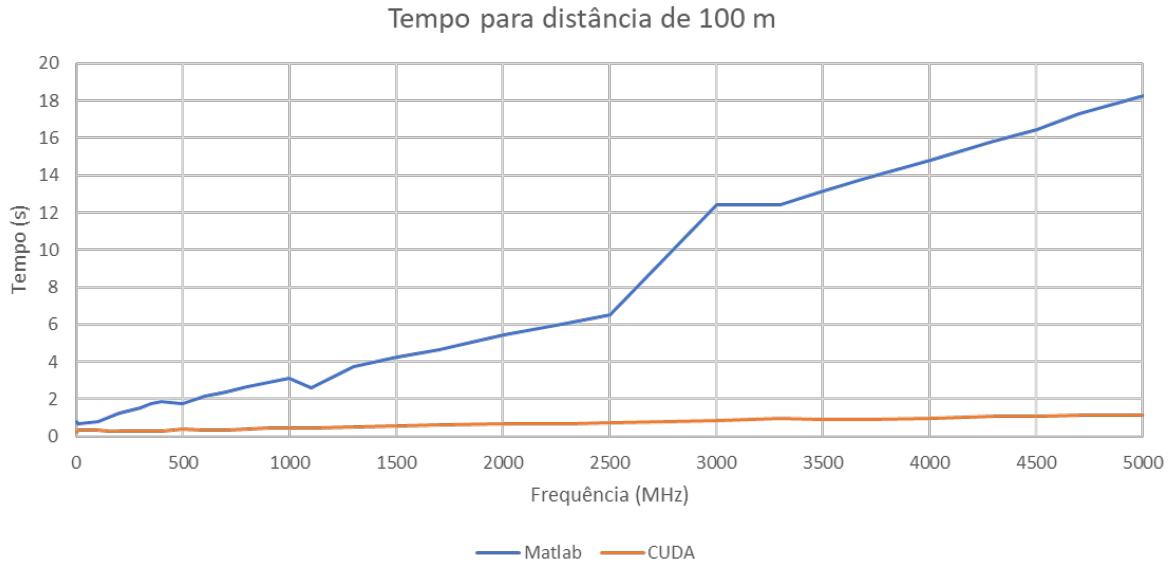
Figura 8.5 – Comparação teórica dos tempos obtidos sequencialmente e utilizando a GPU



Fonte: Elaborado pelo autor.

A Figura 8.6 mostra a comparação real, em um mesmo cenário, variando a frequência, e assim, consequentemente, variando os valores de N e M.

Figura 8.6 – Simulação variando a frequência para levantamento dos tempos obtidos sequencialmente e utilizando a GPU



Fonte: Elaborado pelo autor.

As curva teórica e empírica são parecidas, os degraus ocorrem nas mesmas frequências, pois nessas frequências o tamanho da janela absorvedora é modificado e assim podendo aumentar ou diminuir o número de pontos N, porém o número de pontos M cresce linearmente

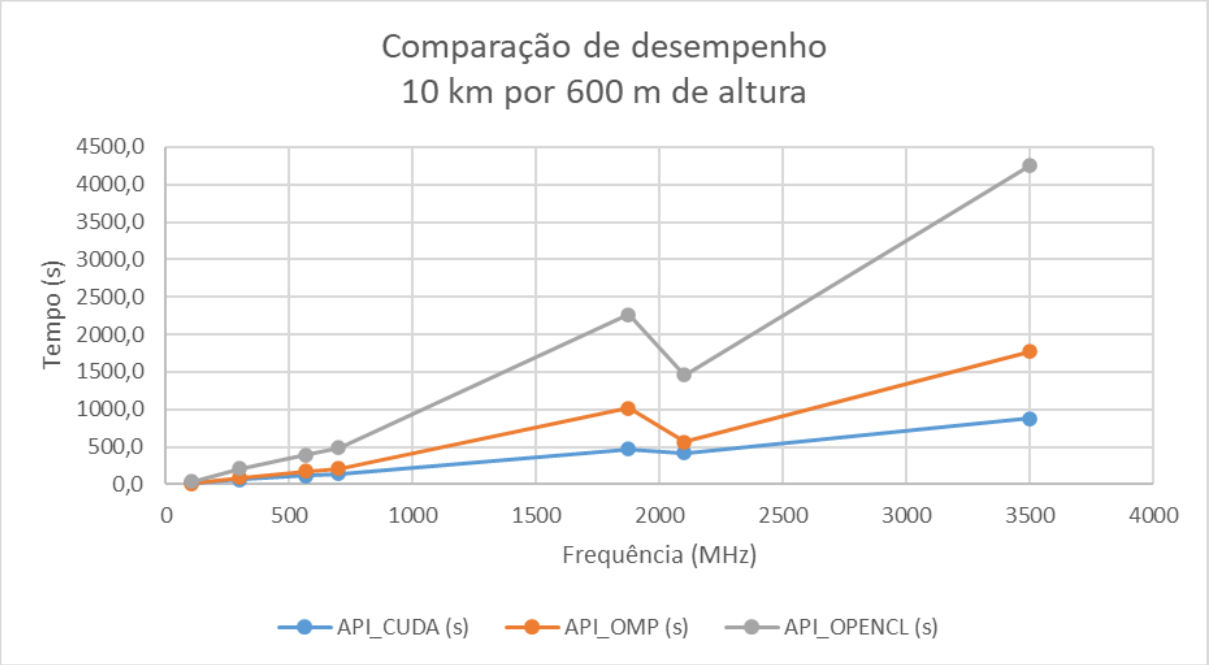
com a frequência.

8.4 Comparação com soluções em outras API

Este mesmo trabalho foi solucionado com outras APIs, elas são: OPENCL e OPENMP. Semelhante ao cuda, a OPENCL utilizou a GPU para realizar os cálculos, com o diferencial de não estar presa a placas gráficas da NVIDIA e ser compilada em tempo de execução. Enquanto o OPENMP utilizou apenas a CPU, distribuindo as tarefas entre todas as *threads* do processador.

Na Figura 8.7 são mostrados os tempos de execução de um mesmo problema para frequências diferentes em uma mesma maquina, os dados do gráfico são mostrados na Tabela 4.

Figura 8.7 – Comparativo entre APIs CUDA, OPENCL e OpenMP em uma mesma maquina



Fonte: Elaborado pelo autor.

Tabela 4 – Tempos em segundos para simulação de 600 metros de altura por 10 km com frequências variadas.

Frequência	API_CUDA (s)	API_OMP (s)	API_OPENCL (s)
105,7	18,8	9,2	34,1
300	59,8	87,9	207,2
569	112,9	171,5	394,9
700	138,7	212,6	485,5
1875	469,7	1013,6	2266,0
2100	416,1	562,7	1462,0
3500	874,1	1773,8	4251,7

Fonte: Elaborado pelo autor.

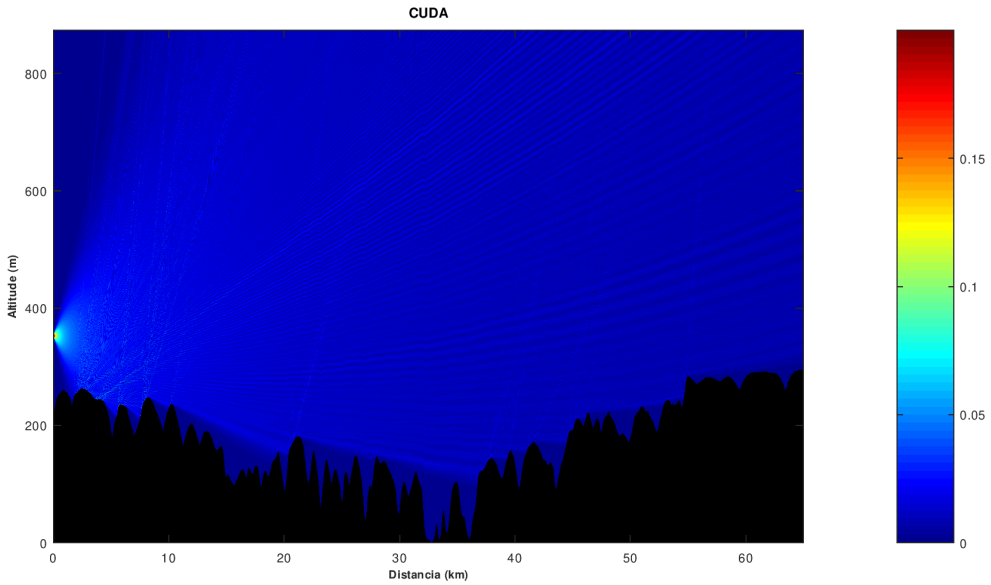
CUDA e OPENMP obtiveram resultados bem próximos, a CPU durante a simulação manteve seu *clock* estável em 4.5 GHz enquanto a GPU, devido ao aumento de temperatura durante as simulações, teve seu *clock* variado entre 1920 MHz a 1800 MHz. O *clock* mais baixo somado ao fato das operações de pontos flutuantes não serem otimizadas levaram aos resultados semelhantes.

O baixo desempenho da OPENCL em comparação as demais soluções pode possuir diversos fatores, as operações de ponto flutuante de precisão dupla serem menos eficientes que as implementadas por CUDA, o agendamento dos kernels realizado em tempo de execução por funções auxiliares impactaram significativamente nos resultados.

8.5 Simulação de Enlace e Perda de Percurso

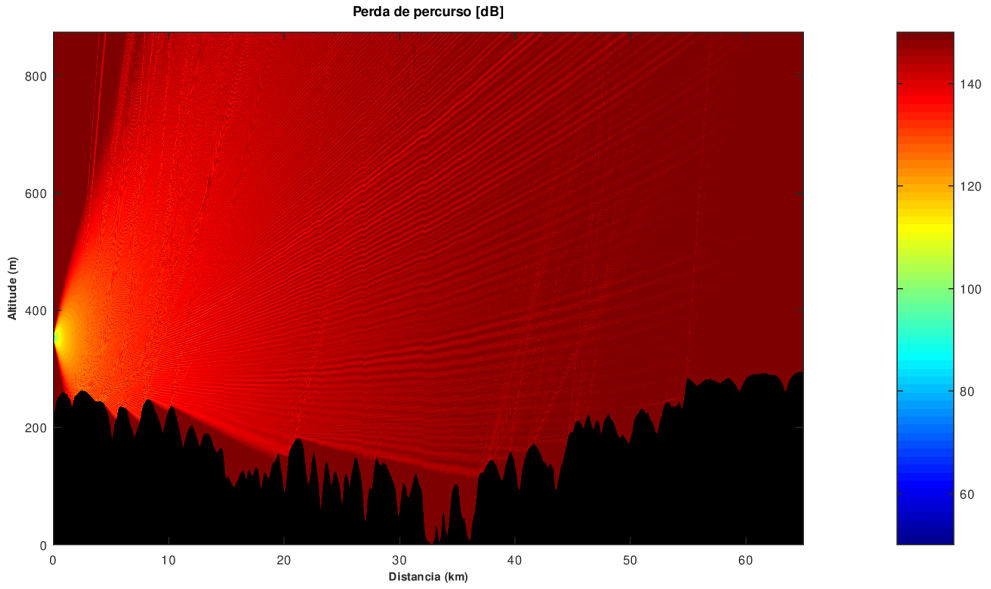
As simulações realizadas utilizaram frequências baixas apenas por efeito comparativo. Como para esses cenários o programa se comportou corretamente, é então realizada a simulação na frequência de 6.125 GHz apenas no programa que utiliza CUDA.

Figura 8.8 – Campo $u(x,y)$



Fonte: Elaborado pelo autor.

Figura 8.9 – Perda de percurso



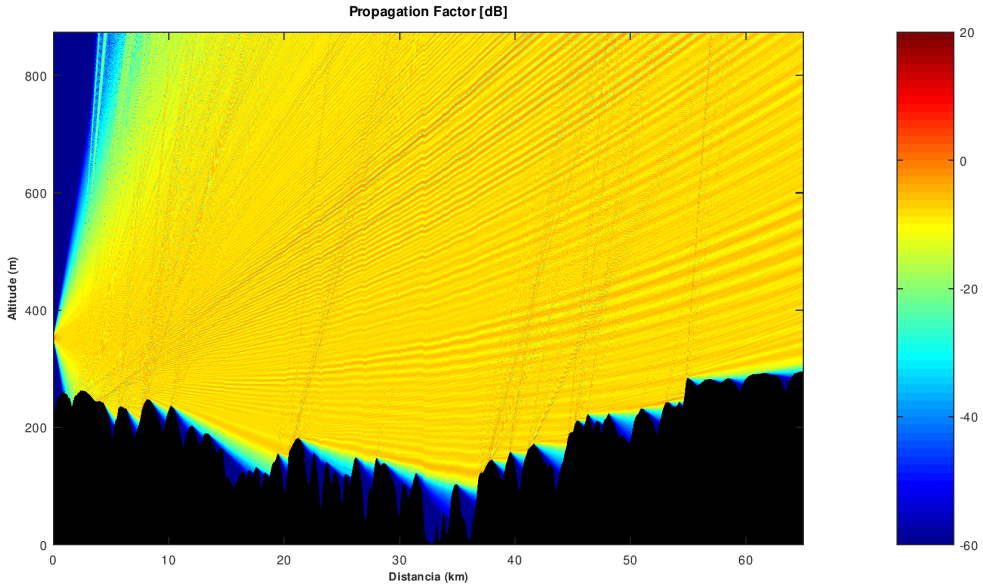
Fonte: Elaborado pelo autor.

A simulação levou 28 minutos e 56 segundos para se realizada, elevando a temperatura do dispositivo ao máximo (66°C), extraindo todo potencial da placa gráfica.

A Figura 8.8 mostra a simulação de um sistema real. A partir do módulo do campo $u(x,y)$ calcula-se a perda de percurso e o fator de propagação, figuras 8.9 e 8.10.

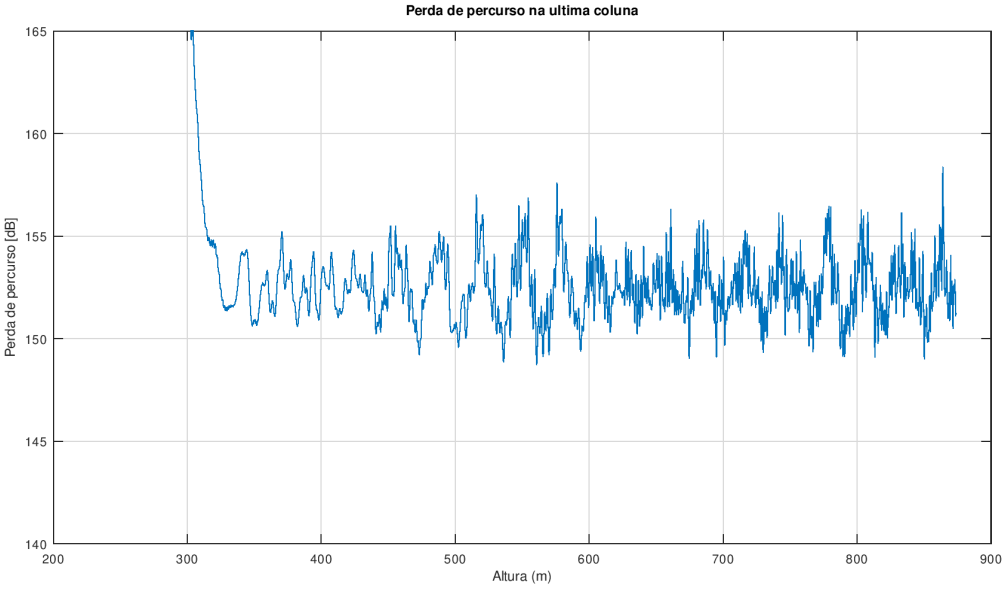
O interesse mesmo é na última posição, onde está localizado a antena receptora. A

Figura 8.10 – Fator de perda



Fonte: Elaborado pelo autor.

Figura 8.11 – Perda de percurso na última coluna



Fonte: Elaborado pelo autor.

Figura 8.11 mostra a perda de percurso para a última coluna em função das alturas. Para uma altura de 45 metros do solo, a perda de percurso é de aproximadamente 152 dB. Em medições reais no local a perda de percurso é de 145 dB. Para realizar a simulação em um tempo viável, foi utilizado um passo de 0.02 em vez de $\frac{\lambda}{20} = 0.0024$, apresentando essa discrepância entre o valor real aferido e o simulado.

9 CONCLUSÕES E TRABALHOS FUTUROS

Após a realização do projeto, foi possível observar que os resultados obtidos foram excelentes, apresentando um alto desempenho e com erros insignificantes. Contudo, ainda é cedo para iniciar o seu uso em aplicações reais, primeiramente é necessária uma etapa de validação, certificação e homologação, a fim de garantir que todas as funcionalidades implementadas estejam funcionando corretamente.

As simulações apresentadas nas Tabelas 1 e 2 mostraram que a utilização da GPU permitiu uma maior velocidade de processamento em comparação com a execução sequencial na CPU. Isso foi possível devido ao uso exclusivo da memória VRAM, evitando, assim, as cópias de dados entre o *host* e o *device*.

9.1 Estudos futuros

Durante o desenvolvimento desse projeto, foram encontradas diversas maneiras de otimizar e tornar os cálculos mais rápidos, como reduzir o número de cálculos com constantes, evitar operações de divisão, evitar o uso de vetores auxiliares que consomem armazenamento desnecessário. Uma possível melhoria, é a implementação da multiplicação de matriz conjugada, $V = (B*)U$, reduzindo o uso de memória e também removendo os cálculos dos parâmetros da matriz B. Nessa melhoria a matriz A seria utilizada para representar os elementos da matriz B, utilizando apenas uma variável para efetuar a condição de contorno do solo.

Para aproveitar todo o poder do *hardware* e sua arquitetura otimizada para pontos flutuantes de precisão simples, modificar as equações para que os cálculos não tenham valores de grandezas muito pequena e nem grandezas muito grandes para permitir o uso do ponto flutuante de precisão simples. Um estudo deve ser feito para averiguar se o erro de precisão simples impactará significadamente no erro final da simulação.

A adição da camada absorvedora ao sistema resulta em um aumento significativo no número de equações, deixando a execução lenta e consumindo muita memória. Uma possível solução para esse problema é implementar a técnica chamada *Perfectly Matched Layer* (PML) (VASCONCELOS, 2017). O estudo consistiria em avaliar se os cálculos para encontrar a PML seriam mais eficientes do que utilizar a camada absorvedora.

As fontes utilizadas para o campo inicial são teóricas e pré-definidas. Realizar um estudo para implementar fontes de campo inicial a partir de sinais reais ampliaria o escopo do projeto e o tornaria mais aplicável na prática.

Outra abordagem, mais complexa, seria aproveitar todo o *hardware* disponível na máquina, utilizando múltiplos processadores e memórias separadas, como em um *cluster*, para realizar a simulação. Nesse projeto, toda a carga foi delegada para a GPU, enquanto o processador aguarda o término dos cálculos para continuar o processo.

REFERÊNCIAS

AMODIO, P. Optimized cyclic reduction for the solution of linear tridiagonal systems on parallel computers*. **Dipartimento di Matematica,Universitità di Bari. Via Orabona 4, 70125 Bari, Italy**, 09 1992.

AMODIO, P.; MAZZIA, F. Stability of the cyclic reduction for the solution of tridiagonal linear systems. 1976.

CHAPRA, S. C. **Métodos numéricos para engenharia**. 5. ed. São Paulo, SP: McGraw-Hill, 2008.

CLAERBOUT, J. F. **Fundamentals of Geophysical Data Processing With Application to Petroleum Prospect**. New York: McGraw-Hill, 1976.

COUCH, L. W. **Digital & Analog Communication Systems**. 8. ed. Upper Saddle River, NJ: Pearson, 2012.

CRANK, J.; NICOLSON, P. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. **Proceedings of the Cambridge Philosophical Society**, v. 43, n. 5, p. 50–67, 1947.

CROCKFORD, D. **The application/json Media Type for JavaScript Object Notation (JSON)**. 2006. <https://tools.ietf.org/html/rfc4627>.

DOE, J. A brief introduction to central finite differences. **Journal of Numerical Analysis**, v. 25, n. 4, p. 567–578, 2010.

FRANCO, C. **Entenda como funciona a plataforma NVIDIA CUDA | Adrenaline**. 2012. <https://adrenaline.com.br/artigos/v/19445/entenda-como-funciona-a-plataforma-nvidia-cuda>. (Accessed on 01/23/2023).

GUIDORIZZI, J. F. **Curso de Cálculo, Volume 1**. [S.l.]: Editora LTC, 1994. v. 1.

HOLM, P. D. Wide-angle shift-map pe for a piecewise linear terrain—a finite-difference approach. **IEEE Transactions on Antennas and Propagation**, v. 55, n. 10, p. 2773–2789, 2007.

KIRK, D. B.; HWU, W.-m. W. **Introduction to GPU computing**. [S.l.]: CRC press, 2010.

KULKARNI, S. Implementation of a parallel tridiagonal solver for linear system of equations arising in physicell-biofvm. **Departamento Ingeniería Civil y Ambiental**, 06 2021.

LEVY, M. **Parabolic equation methods for electromagnetic wave propagation**. [S.l.]: IET, 2000.

NUSSENZVEIG, H. **Curso De Fisica Basica, V.3 - Eletromagnetismo**. [S.l.]: EDGARD BLUCHER, 2015. ISBN 9788521208013.

NVIDIA, C. In: **cuSPARSE The API reference guide for cuSPARSE, the CUDA sparse matrix library.** [s.n.]. Disponível em: <https://docs.nvidia.com/cuda/cusparse/>.

NVIDIA, C. Cuda: A platform for general-purpose computation on graphics processing units. In: IEEE PRESS. **Proceedings of the ACM/IEEE Conference on Supercomputing**. [S.l.], 2007.

OPPENHEIM, A. V. **Sinais e Sistemas**. [S.l.]: Prentice-Hall, 2010.

OZGUN, O.; APAYDIN, G.; KUZUOGLU, M.; SEVGI, L. PETOOL: MATLAB-based one-way and two-way split-step parabolic equation tool for radiowave propagation over variable terrain. **Computer Physics Communications**, Elsevier BV, v. 182, n. 12, p. 2638–2654, dez. 2011. Disponível em: <https://doi.org/10.1016/j.cpc.2011.07.017>.

PILLA, L. L. Análise de desempenho da arquitetura cuda utilizando os nas parallel benchmarks. Porto Alegre, Brasil, 2009. Disponível em: <http://hdl.handle.net/10183/18536>.

SAUTER, E.; AZEVEDO, F. S. de; KONZEN, P. H. de A. **Os triplos produtos e outras identidades vetoriais**. [S.l.]: UFRGS - IME - Recursos Educacionais Abertos de Matemática, 2022. <https://www.ufrgs.br/reamat/Calculo/livro-cfvv/xv-os_triplos_produtos_e_outras_identidades_vetoriais.html>. (Accessed on 01/12/2023).

SHURE, L. **Using GPUs in MATLAB » Loren on the Art of MATLAB - MATLAB & Simulink**. 2012. <https://blogs.mathworks.com/loren/2012/02/06/using-gpus-in-matlab/>. (Accessed on 01/25/2023).

VALERO-LARA, P.; MARTÍNEZ-PÉREZ, I.; SIRVENT, R.; MARTORELL, X.; PEÑA, A. J. Nvidia gpus scalability to solve multiple (batch) tridiagonal systems. implementation of cuthomasbatch. In: **Parallel Processing and Applied Mathematics-12th International Conference (PPAM)**. [S.l.: s.n.], 2017.

VASCONCELOS, L. S. **Análise da propagação troposférica sobre terrenos irregulares em VHF e UHF utilizando equações parabólicas e o desenvolvimento de um novo modelo híbrido para predição de perda de percurso.** 530 p. Monografia (Tese (Doutorado em Engenharia Elétrica)) — Universidade Federal do Uberlândia, Uberlândia, Minas Gerais, 2017.

VELAME, M. R. **CÁLCULO DOS CAMPOS ELETROMAGNÉTICOS DA LINHA DE TRANSMISSÃO GOVERNADOR MANGABEIRA-SAPEAÇU C1 VIA OS MÉTODOS DE SIMULAÇÃO DAS CARGAS E DAS IMAGENS**. 58 p. Monografia (TCC) — Universidade Federal do Recôncavo da Bahia, Cruz das almas, Bahia, 2019.

ANEXOS

ANEXO A – Parametros.json

```
{
  "Diretorio de trabalho": "../terrenos",
  "Arquivo do perfil do terreno": "perfil1.ghl",
  "Modelo": 0,
  "__modelo__": "TERRA_ESFERICA_LISA_2 = 0,
     TERRA_ESFERICA_LISA_1 = 1 /*nao implementado*/,
     TERRA_ESFERICA = 2",
  "Modelo de Otimizacao PCR": 1,
  "__modelo_de_otimizacao_pcr__": "OTM_MANTER_O_MAIS_PROXIMO
      = 0, OTM_AUMENTAR_DADOS = 1, OTM_DIMINUIR_DADOS = 2",
  "Distancia (geodesica) maxima [m]": 3000,
  "Altura maxima acima do terreno [m]": 50,
  "PASSO DE DISTANCIA PARAXIAL [m]": 0,
  "PASSO DE ALTITUDE [m]": 0,
  "Fonte": {
  "Frequencia [MHz]": 1000,
  "Tipo da fonte": 0,
  "__tipo_da_fonte__": "FONTE_GAUSSIANA = 0, FONTE_RECT = 1,
      FONTE_OMNI = 2",
  "Largura de Feixe de 3dB das fontes [graus]": [2],
  "Beam tilt das fontes [graus]": [1],
  "Altura das fontes [m] acima do solo": [25],
  "Polarizacao da fonte [H/V]": "V"
  },
  "Refracao": {
  "Perfil de refracao": 4,
  "__perfil_de_refracao__": "ATMOSFERA_PADRAO = 0, LINEAR =
     1, BILINEAR = 2, TRILINEAR = 3, EXPONENCIAL = 4",
  "Indice de refracao inicial": 1.0003,
  "Refractividade a nivel do mar": 315,
  "Altura da escala da refractividade": -0.136,
  "Alturas (em ordem de baixo pra cima) de limite na
     atmosfera": [50, 100],
  "Inclinacoes do perfil (para caso bilinear ou trilinear)":
      [-1e-5, 1e-5, 1e-3]
  },

  "Saida de dados": {
  "Diretorio de saida": "OUTPUT",
  "Tipo de salvamento": 0,
  "__tipo_de_salvamento__": "SALVAR_COMPLETO = 0,
     SALVAR_DISCRETIZADO = 1/*nao implementado*/,
     SALVAR_MEDIA = 2/*nao implementado*/, SALVAR_MEDIAMOVEL
      = 3/*nao implementado*/",
  "PASSO DE ALTURA Salva [m]": -8000,
  "PASSO DE DISTANCIA Salva [m]": -8000,
  "Imagem de Saida": [1000,1000]
  },
```

```
39   "__tipos_de_solo__": "DEFINIDO_PELO_USER = 0, MAR = 1,
       AGUA_DOCE = 2, SOLO_UMIDO = 3, SOLO_MEIO_UMIDO = 4,
       SOLO_SECO = 5, AGUA_PADRAO = 6",
40   "Solo": [
41   {"tipo": 6, "posicao": 0},
42   {"tipo": 5, "posicao": 25},
43   {"tipo": 4, "posicao": 50},
44   {"tipo": 3, "posicao": 100},
45   {"tipo": 2, "posicao": 150},
46   {"tipo": 1, "posicao": 200},
47   {"tipo": 0, "posicao": 200,"epsilon relativo": 1,"sigma":
       1e-6}
48   ]
49 }
```