UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FACULDADE DE ENGENHARIA ELÉTRICA

CURSO DE GRADUAÇÃO EM ENGENHARIA ELETRÔNICA E DE TELECOMUNICAÇÕES

GUILHERME FERREIRA DE JESUS

COMPUTAÇÃO PARALELA COM OPENCL APLICADA À PREDIÇÃO DE PERDA DE PERCURSO UTILIZANDO EQUAÇÕES PARABÓLICAS E O MÉTODO DAS DIFERENÇAS FINITAS

UBERLÂNDIA - MG

2023

GUILHERME FERREIRA DE JESUS

COMPUTAÇÃO PARALELA COM OPENCL APLICADA À PREDIÇÃO DE PERDA DE PERCURSO UTILIZANDO 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: Prof. Dr. Lorenço Santos Vasconcelos

UBERLÂNDIA - MG

2023

RESUMO

JESUS, G. F. **Computação paralela com OpenCL aplicada à predição de perda de percurso utilizando equações parabólicas e o método das diferenças finitas**. 2023, 73 p. Monografia (Graduação) - Faculdade de Engenharia Elétrica, Universidade Federal de Uberlândia, 2023.

O trabalho de conclusão de curso em questão trata da aplicação da computação paralela com OpenCL para a predição de perda de percurso utilizando equações parabólicas e o método das diferenças finitas. OpenCL é uma linguagem de programação de alto nível que permite a criação de aplicações que rodam em paralelo em diferentes tipos de dispositivos de computação, como CPUs, GPUs e outros processadores especializados.

O objetivo deste trabalho é aumentar a velocidade de processamento da predição de perda de percurso utilizando a computação paralela, para que ela possa ser realizada de forma mais eficiente e rápida. Para isso, foram implementadas equações parabólicas e o método das diferenças finitas em OpenCL, permitindo que esses cálculos fossem realizados de forma paralela em diversos núcleos de processamento.

Os resultados mostram que a utilização da computação paralela com OpenCL permitiu um aumento significativo na velocidade de processamento da predição de perda de percurso, tornando essa tarefa mais eficiente e rápida. Além disso, o uso da OpenCL permitiu a utilização de diversos tipos de dispositivos de computação, expandindo as possibilidades de implementação desta solução em diferentes contextos.

Palavras-chaves: Computação Paralela; Equações parabólicas; OpenCL; Propagação.

ABSTRACT

The final project in question deals with the application of parallel computing with OpenCL for the prediction of path loss using parabolic equations and the finite differences method. OpenCL is a high-level programming language that allows the creation of applications that run in parallel on different types of computing devices, such as CPUs, GPUs, and other specialized processors. The goal of this final project is to increase the processing speed of path loss prediction using parallel computing, so that it can be performed more efficiently and quickly. To achieve this, parabolic equations and the finite differences method were implemented in OpenCL, allowing these calculations to be performed in parallel on multiple processing cores.

The results show that the use of parallel computing with OpenCL allowed a significant increase in the processing speed of path loss prediction, making this task more efficient and faster. Additionally, the use of OpenCL allowed the use of different types of computing devices, expanding the possibilities of implementation of this solution in different contexts.

Key-words: OpenCl; Parallel Computing; Parabolic equation; Propagation.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LIST OF ALGORITHMS

LISTA DE ABREVIATURAS E SIGLAS

API *Application Programming Interface*

CPU *Central Processing Unit*

DPS *Digital Signal Processor*

GPU *Graphics Processing Units*

GUI *Interface Gráfica do Usuário*

PCR Parallel cyclic reduction

PML *Perfectly Matched Layer*

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

Muitos dos sistemas de telecomunicações atuais envolvem enlaces de rádio. Alguns exemplos são: sistemas de difusão rádio e televisão, a interface aérea de sistemas celulares, enlaces de micro-ondas, entre outros. Em todas essas situações, é necessário realizar estudos sobre a área de cobertura dos sistemas irradiantes e isso requer o cálculo de perda de percurso. Existem diferentes métodos para cálculo de perda de percurso, cada um considerando diferentes fatores e com complexidades distintas. Existem alguns softwares disponíveis hoje no mercado que oferecem esse serviço, porém utilizam métodos que fornecem resultados aproximados. Um dos métodos mais generalistas e robustos, principalmente para aplicações de longa distância, é o método de equações parabólicas, porém sua implementação comum exige alto tempo de processamento e uso de memória, podendo levar cerca 40 min para fornecer resultados de cenários relativamente pequenos. Sob essa perspectiva, esse trabalho justifica-se por se tratar de uma otimização do tempo de processamento por meio da utilização de computação paralela. Assim, torna-se possível obter resultados muito satisfatórios com o método das equações parabólicas e de maneira rápida, viabilizando a utilização desse método de forma comercial.

Para esse propósito, será utilizada a *Application Programming Interface* (API) (application programming interfaces) OpenCL para a implementação do método de equações parabólicas por diferenças finitas, utilizando algoritmos paralelos e utilizando máxima performance do computador.

1.1 Objetivos

O objetivo deste trabalho é implementar o cálculo de perda de percurso pelo método de equações parabólicas por diferenças finitas utilizando computação paralela para reduzir o tempo de processamento. Assim, obtendo predições de perda de percurso em menos tempo do que as implementações convencionais, com a possibilidade de alterações dos parâmetros e visualização em tempo real. Isso é muito desejado no projeto de enlaces de rádio, em que é de suma importância a predição de área de cobertura. Com este trabalho, os projetos de sistemas de comunicação sem fio ponto a ponto e ponto-área poderão vislumbrar ganhos em termos de agilidade e precisão.

2 PROPAGAÇÃO DE ONDAS

2.1 Introdução

A propagação de ondas é o processo pelo qual as ondas se espalham ao longo do espaço, transportando energia e informação. Há vários tipos diferentes de ondas, incluindo ondas mecânicas e eletromagnéticas. As ondas mecânicas, como as ondas sonoras e as ondas na água, requerem um meio material para se propagarem, enquanto as ondas eletromagnéticas, como as ondas de rádio e as ondas de luz, podem se propagar no vácuo. A propagação de ondas pode ser descrita por sua equação de onda, que governa como a onda se espalha ao longo do tempo e do espaço.

2.2 Ondas Eletromagnéticas

As ondas eletromagnéticas são ondas que se propagam através do espaço como uma combinação de campos elétrico e magnético oscilantes. Elas não precisam de um meio material para se propagarem e podem viajar através do vácuo com a velocidade da luz.

As ondas eletromagnéticas têm uma variedade de aplicações, incluindo a comunicação, a iluminação, a medicina e a observação científica. Alguns exemplos de ondas eletromagnéticas incluem:

- Rádio ondas: usadas para a transmissão de sinais de rádio, televisão e telefonia sem fio
- Microondas: usadas em sistemas de comunicação de dados e em equipamentos de cozinha
- Luz visível: usada para a visão e em aplicações de iluminação
- Infravermelho: usado em sistemas de vigilância, tecnologia remota e medicina
- Raios-X e raios gama: usados em medicina e na investigação científica.

Todas as ondas eletromagnéticas podem ser descritas por sua frequência, comprimento de onda e polarização, e seu comportamento é determinado pelas equações de Maxwell.

2.2.1 Campo Eletromagnético

2.2.1.1 Equações de Maxwell forma diferencial

Um campo elétrico e um campo magnético são dois aspectos diferentes da mesma coisa: o campo eletromagnético. As equações de Maxwell para o campo elétrico incluem a equação de Gauss, que descreve como a carga influencia o campo elétrico em uma dada região, e

as equações de Maxwell-Ampère e Faraday, que descrevem como um campo elétrico varia no tempo e no espaço.

As equações de Maxwell para o campo magnético incluem a equação de Gauss para o campo magnético, que descreve como a corrente influencia o campo magnético em uma dada região, as equações de Maxwell-Faraday, que descrevem como o campo magnético varia no tempo e no espaço.

As equações de Maxwell também incluem a equação de Maxwell-Lorentz, que descreve como os campos elétrico e magnético estão relacionados e como eles se afetam mutuamente. Essas equações são importantes para entender como a luz e outras formas de radiação eletromagnética se comportam.

As equações de Maxwell são uma conjunto de quatro equações matemáticas que descrevem a relação entre o campo elétrico e o campo magnético. Elas foram desenvolvidas pelo físico James Clerk Maxwell no século XIX e têm sido fundamentais para a compreensão da eletromagnetismo. As quatro equações de Maxwell na sua forma diferencial são mostradas nas equações (2.1) a (2.4) (BUCK JOHN A.; HAYT JR, 2013):

Lei de Gauss para o campo elétrico

$$\vec{\nabla} \cdot \vec{E} = \frac{\rho}{\varepsilon_0} \tag{2.1}$$

Equação de Faraday:

$$\vec{\nabla} \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \tag{2.2}$$

Equação de conservação do fluxo magnético:

$$\vec{\nabla} \cdot \vec{B} = 0 \tag{2.3}$$

Equação de Maxwell-Ampère:

$$\vec{\nabla} \times \vec{B} = \mu_0 \vec{J} + \mu_0 \varepsilon_0 \frac{\partial \vec{E}}{\partial t} \tag{2.4}$$

em que,

$\vec{E}$ é o campo elétrico (em V/m)

ρ é a densidade de carga elétrica (em C/m³)

ε_0 é a permissividade elétrica do vácuo (em F/m)

μ_0 é a permeabilidade magnética do vácuo (em H/m)

$\vec{B}$ a densidade de fluxo magnético (em T)

$\vec{J}$ é a densidade de corrente elétrica (em A/m²)

∇ é o operador nabla (ou gradiente)

$\frac{\partial}{\partial t}$ é o operador de diferenciação parcial com relação ao tempo

∂ é o operador de diferenciação parcial

$\nabla\times$ é operador rotacional

A Equação (2.1) diz que a densidade de carga elétrica em um ponto específico no espaço é proporcional à divergência do campo elétrico nesse ponto.

A Equação (2.2) diz que a fonte do rotacional do campo elétrico é a variação temporal do campo magnético nesse ponto.

A Equação (2.3) mostra que B é um campo solenoidal e que as linhas de fluxo magnético formam caminhos fechados

A quarta Equação (2.4) determina as correntes elétricas e a variação temporal do campo elétrico como fontes da circulação de B.

2.2.1.2 Equações de Maxwell forma integral

As equações de Maxwell também podem ser representadas na sua forma integral e são utilizadas para descrever e entender vários fenômenos elétricos e magnéticos que ocorrem no espaço, tais como ondas eletromagnéticas, radiação, geração de corrente elétrica, entre outros. Elas também são usadas para fazer cálculos precisos em sistemas elétricos e eletromagnéticos, tais como antenas, geradores de energia, transmissores de rádio, entre outros.

As quatro equações de Maxwell na forma integral são mostradas nas Equações (2.5) a (2.8).

- A equação de Gauss para o campo elétrico:

$$\oint_S \vec{E} \cdot d\vec{S} = \frac{1}{\varepsilon_0} \int_V \rho dV \tag{2.5}$$

  A equação de Gauss para o campo elétrico relaciona o fluxo do campo elétrico através de qualquer superfície fechada com a carga total dentro dessa superfície.

- A equação de Gauss para o campo magnético:

$$\oint_S \vec{B} \cdot d\vec{S} = 0 \tag{2.6}$$

  A equação de Gauss para o campo magnético diz que o fluxo do campo magnético através de qualquer superfície fechada é zero.

- A equação de Maxwell-Faraday para o campo elétrico:

$$\oint_C \vec{E} \cdot d\vec{l} = -\frac{\partial}{\partial t} \int_S \vec{B} \cdot dS \tag{2.7}$$

A equação de Maxwell-Faraday para o campo elétrico relaciona a variação temporal do fluxo do campo magnético através de qualquer superfície com a circulação do campo elétrico ao longo do contorno da superfície fechada.

- A equação de Maxwell-Ampère para o campo magnético:

$$\oint \vec{B} \cdot d\vec{l} = \mu_0 \oint \vec{J} \cdot d\vec{l} + \mu_0 \varepsilon_0 \oint \frac{\partial \vec{E}}{\partial t} \cdot d\vec{l} \tag{2.8}$$

A equação de Maxwell-Ampère para o campo magnético relaciona a circulação do campo magnético por qualquer caminho fechado com a corrente elétrica que atravessa a superfície, mais a variação temporal do fluxo do campo elétrico polarizado através da região.

2.2.2 Equação da Onda

As equações de Helmholtz, também conhecidas como equações vetoriais da onda, são uma classe de equações matemáticas que descrevem a propagação de ondas em meios contínuos e isotrópicos. Elas são usadas para descrever ondas harmônicas, como as ondas sonoras e as ondas eletromagnéticas.

As equações de Helmholtz são baseadas nas equações de Maxwell mostradas na seção 2.2, na forma fasorial em um meio dielétrico sem fontes e com perdas, linear, isotrópico e homogêneo. E portanto as equações de Maxwell para este caso podem ser reduzidas às Equações (2.9) a (2.12).

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

$$\vec{\nabla} \times \vec{H} = \varepsilon \frac{\partial \vec{E}}{\partial t} \tag{2.10}$$

$$\vec{\nabla} \cdot \vec{E} = 0 \tag{2.11}$$

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

em que, $\vec{H} = \frac{\vec{B}}{\mu}$.

Tomando o rotacional em ambos os lados da Equação (2.10), obtém-se a Equação (2.13):

$$\vec{\nabla} \times (\vec{\nabla} \times \vec{H}) = \vec{\nabla} \times (-\mu\varepsilon \frac{\vec{\partial E}}{\partial t}) \tag{2.13}$$

Aplicando a identidade vetorial $\vec{\nabla} \times (\vec{\nabla} \times H) = \vec{\nabla}(\vec{\nabla} \cdot \vec{H}) - \nabla^2 \vec{H}$, e substituindo 2.12,então tem-se a Equação (2.14):

$$\vec{\nabla}^2 \vec{H} = (\mu\varepsilon \frac{\partial^2 \vec{H}}{\partial t^2}) \tag{2.14}$$

É importante notar que a equação de Helmholtz para o campo magnético é baseada na hipótese de meio isotrópico e linear, e é válida somente para ondas eletromagnéticas planas e harmônicas.

De forma análoga para o campo elétrico tem-se a Equação (2.15).

$$\vec{\nabla}^2 \vec{E} = (\mu\varepsilon \frac{\partial^2 \vec{E}}{\partial t^2}) \tag{2.15}$$

Essas equações são uma forma simplificada das equações de Maxwell para ondas estacionárias e são geralmente utilizadas para resolver problemas de ondas em meios homogêneos e isotrópicos, como ondas em uma corda tensa ou ondas em uma placa fina. Elas também são usadas na teoria da difração para descrever a propagação de ondas através de aberturas e obstáculos.

2.3 Equações Parabólicas

2.3.1 Introdução

O método das equações parabólicas é uma técnica utilizada para descrever a propagação de ondas em meios contínuos. Ele é baseado na equação de onda parabólica, que é uma equação diferencial parcial que descreve como uma onda se propaga através de um meio. O método das equação parabólicas simplifica a equação da onda de Helmholtz para uma equação parabólica e, então, obtém a sua solução por meio de métodos de resolução numérica (VASCONCELOS, 2017).

Para resolver a equação de onda parabólica, é necessário especificar as condições iniciais e de contorno. As condições iniciais especificam o estado inicial da onda, enquanto as condições de contorno especificam como a onda se comporta nas bordas do meio.

Existem vários métodos diferentes para resolver a equação de onda parabólica, como o método de diferenças finitas, o método de elementos finitos e o método *Fourier split-step* . Cada um desses métodos tem suas próprias vantagens e desvantagens, e é importante escolher o método mais adequado para o problema em questão.

Uma vez resolvida a equação de onda parabólica, é possível calcular a amplitude da onda em diferentes pontos no tempo e no espaço. Isso permite simular a propagação da onda em diferentes meios

A relação entre as equações parabólicas e o custo computacional depende do método utilizado para resolver essas equações. Métodos como o método de diferenças finitas e o método de elementos finitos envolvem a discretização do espaço e do tempo, o que geralmente resulta em um grande número de equações a serem resolvidas, e portanto, soluções mais precisas. Isso pode resultar em altos custos computacionais, especialmente quando simulações de alta precisão são necessárias ou quando o domínio do problema é muito grande.

Por outro lado, métodos como o método *Fourier split-step* podem ser computacionalmente mais eficientes, pois permitem a utilização de intervalos de discretização maiores. No entanto, esses métodos possuem aspectos desfavoráveis no tratamento de condições de contorno e modelagem de terreno.

Em geral, o custo computacional de resolver as equações parabólicas pode variar amplamente dependendo do método utilizado, da precisão da simulação e do tamanho do problema.

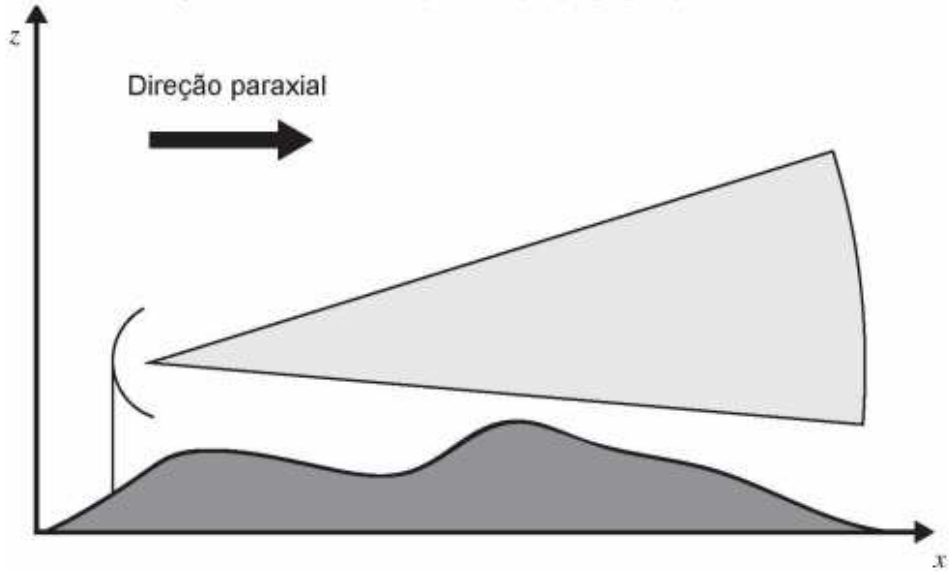
Além disso, avanços na computação paralela e na computação de alto desempenho têm permitido que simulações baseadas em equações parabólicas sejam executadas em cluster de computadores e computadores de alto desempenho, o que pode reduzir significativamente o custo computacional.

2.3.2 Formulação do Método

O método de equações parabólicas é uma abordagem utilizada para modelar a propagação de energia em sistemas onde há uma direção principal de interesse, chamada de direção para-axial. Ele utiliza uma aproximação da equação da onda, e assume que a propagação

da energia ocorre principalmente dentro de um cone centrado na direção para-axial, como indica a Figura 1. Isso permite simplificar o problema e resolvê-lo de forma mais eficiente.

Figura 1 – Propagação Paraxial na Troposfera



Fonte: Parabolic equation methods for electromagnetic wave propagation (LEVY, 2000)

Para o desenvolvimento genérico serão feitas as seguintes considerações (VASCONCELOS, 2017):

- Os campos variam de forma harmônica com uma frequência angular ω
- O método descrito é voltado para resolver problemas que possuem duas dimensões.
- Será utilizado o sistema cartesiano de coordenadas espaciais
- De acordo com a convenção adotada, o eixo x é considerado como a direção principal de interesse, chamada de direção para-axial e a variação de altitude é representada no eixo z.
- Esse método permite dividir o problema em dois casos distintos: problema horizontalmente polarizado ou problema verticalmente polarizado.
- Para polarização horizontal o campo elétrico terá somente a componente $E_y(x,z)$
- Para polarização vertical o campo magnético terá somente a componente $H_y(x,z)$
- Para fins de generalidade nos cálculos, é utilizada a componente de campo $\psi(x,z)$, definida na Equação (2.16). Esta componente é usada para representar tanto o campo elétrico quanto o campo magnético, dependendo do caso em questão.

$$\begin{aligned} \psi(x,z) &= E_y(x,z) \quad \text{Polarização Horizontal} \\ \psi(x,z) &= H_y(x,z) \quad \text{Polarização Vertical} \end{aligned} \tag{2.16}$$

De forma análoga à seção 2.2.2 é possível obter as equações homogêneas de Helmholtz em função de ψ (VASCONCELOS, 2017).

$$\nabla^2\psi - \gamma^2\psi = 0 \tag{2.17}$$

em que $\gamma^2 = -\omega^2\mu\varepsilon + j\omega\mu\sigma$

Como mostrado na Equação (2.16) o campo em questão tem apenas a componente em y, com isso a Equação (2.17) se reduz à Equação (2.18)

$$\frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial z^2} - \gamma^2\psi = 0 \tag{2.18}$$

Considerando um meio sem perdas e não magnético, ou seja $\varepsilon = \varepsilon_0\varepsilon_r, \mu = \mu_0, \sigma = 0$, tem-se que, $\gamma^2 = -\omega^2\mu_0\varepsilon_0\varepsilon_r$. Aplicando essas considerações na Equação (2.18), chega-se à Equação (2.19)

$$\frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial z^2} + \omega^2\mu_0\varepsilon_0\varepsilon_r\psi = 0 \tag{2.19}$$

Sendo, $k_0 = \omega\sqrt{\mu_0\varepsilon_0}$ a constante de fase do vácuo. E para o meio em questão, o índice de refração é $n = \sqrt{\varepsilon_r}$ (CHENG, 2007), com isso a Equação (2.19) se torna :

$$\frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial z^2} + k_0^2 n^2\psi = 0 \tag{2.20}$$

A Equação (2.20) não é precisa quando o índice de refração varia com a direção paraxial, mas pode ser uma boa aproximação se essas variações forem suaves em relação à escala do comprimento de onda, com isso pode-se considerar um $n(x,z)$.

A essência da aproximação parabólica é dividir a componente de campo $\psi(x,z)$ em dois termos, de acordo com a Equação (2.21) (VASCONCELOS, 2017).

$$\psi(x,z) = u(x,z)e^{jk_0x} \tag{2.21}$$

em que, $u(x,z)$ é uma variação lenta de amplitude e e^{jk_0x} é uma variação rápida de fase.

Substituindo a aproximação de 2.21 na Equação (2.20), obtém-se as Equações (2.22) e (2.23)

$$\frac{\partial^2(ue^{jk_0x})}{\partial x^2}+\frac{\partial^2(ue^{jk_0x})}{\partial z^2}+k_0^2n^2(x,z)ue^{jk_0x}=0 \tag{2.22}$$

$$\frac{\partial^2 u}{\partial x^2}+2jk_0\frac{\partial u}{\partial x}+\frac{\partial^2 u}{\partial z^2}+k_0^2[n^2(x,z)-1]u=0 \tag{2.23}$$

Fatorando a Equação (2.23) utilizando o operador pseudo diferencial (VASCONCELOS, 2017) (LEVY, 2000), tem-se a Equação (2.24):

$$\left[\frac{\partial}{\partial x}+jk_0(1-Q)\right]\left[\frac{\partial}{\partial x}+jk_0(1+Q)\right]u=0 \tag{2.24}$$

em que o operador pseudodiferencial é dado pela Equação (2.25)

$$Q=\sqrt{\frac{1}{k_0^2}\frac{\partial^2}{\partial z^2}+n^2(x,z)} \tag{2.25}$$

É necessário tomar alguns cuidados para considerar que as variações de n com x seja pequenas, para que erros permaneçam pequenos, pois o operador Q não comuta com a derivada em relação a x (não leva em conta variações em x), visto que o índice de refração varia com a distância para-axial x (VASCONCELOS, 2017).

O próximo passo é separar a equação da onda em dois termos definidos pela Equação (2.24) e encontrar suas soluções. Isso é mostrado nas Equações (2.26) e (2.27).

$$\frac{\partial u}{\partial x}=-jk_0(1-Q)u \tag{2.26}$$

$$\frac{\partial u}{\partial x}=-jk_0(1+Q)u \tag{2.27}$$

As Equações (2.26) e (2.27) correspondem, respectivamente, a ondas que propagam no sentido positivo e negativo da direção para-axial x e são chamadas de *forward* e *backward propagation* (LEVY, 2000).

Quando a distância entre o transmissor e o receptor é grande comparada às alturas das antenas, é possível considerar que a energia eletromagnética propaga entre as duas antenas em ângulos rasantes em relação ao terreno. Nestas condições, despreza-se a Equação (2.27) e resolve-se apenas a Equação (2.26) e a equação de Helmholtz é substituída por uma única

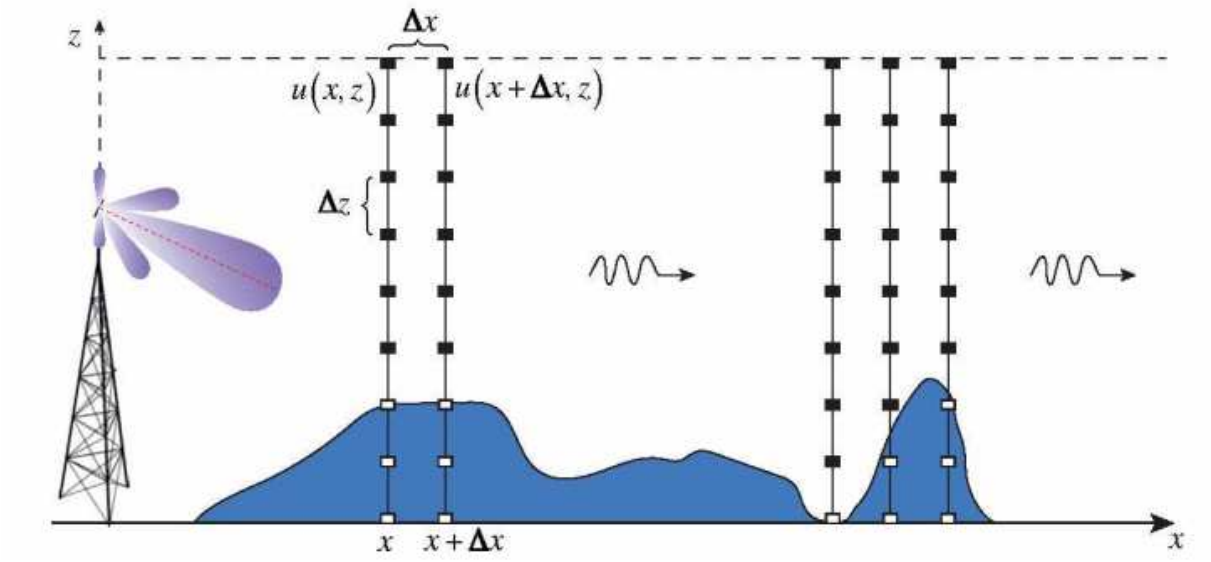
equação parabólica mais simples. Esse procedimento é chamado de aproximação para-axial (VASCONCELOS, 2017).

As Equações (2.26) e (2.27) são equações pseudodiferenciais de primeira ordem em x (por isso a nomenclatura parabólica). A solução formal para Equação (2.26) de modo geral é dada pela Equação (2.28)(LEVY, 2000)

$$u(x+\Delta x,.) = e^{ik\Delta x(-1+Q)}u(x,.) \tag{2.28}$$

A componente de campo reduzido u é obtida em uma distância $x+\Delta x$, usando valores de campo na distância anterior x e condições de contorno apropriadas. Conhecendo-se a distribuição inicial $u(0,z)$, a partir dela é possível encontrar iterativamente o valor de u para cada incremento de Δx, como mostra a Figura 2

Figura 2 – Processo iterativo para resolução das equações parabólicas



Fonte: (OZGUN *et al.*, 2011)

2.3.3 Método das diferenças finitas

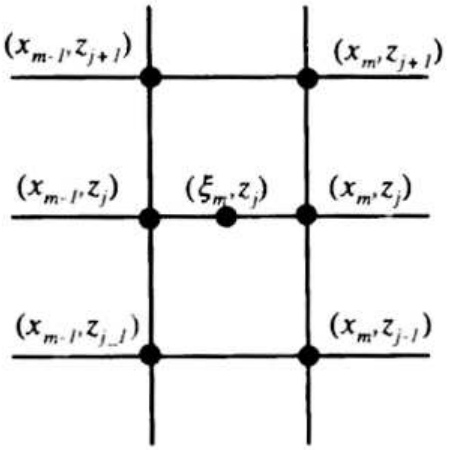
Nesta seção será apresentado o método das diferenças finitas baseado no esquema implícito de diferenças finitas do tipo Crank-Nicolson.

O método Crank-Nicolson é um método de diferenças finitas usado para resolver equações diferenciais parciais (EDPs) substituindo por diferenças finitas. Desse modo, as equações diferenciais passam a ser equações de diferenças.

O método começa discretizando o domínio, dividindo-os em uma grade de pontos, como indica a Figura 3. A solução é aproximada nestes pontos da grade, e as derivadas são aproximadas usando diferenças finitas. A partir disto, a distâncias discretas podem ser escritas por:

$$\begin{aligned} x_0 &= 0 & m &= 0 \\ x_m &= x_{m-1} + \Delta x_m & m &= 1, M \\ z_j &= j\Delta_z & j &= 0, N \end{aligned} \tag{2.29}$$

Figura 3 – Grade de Pontos esquema Crank-Nicolson



Fonte: (LEVY, 2000)

O ponto intermediário do passo x_{m-1} para x_m para o esquema de Crank-Nicolson é o ponto médio e é dado por:

$$\xi_m = \frac{x_{m-1} + x_m}{2} \tag{2.30}$$

Em resumo, o método Crank-Nicolson é um método de diferenças finitas que resolve EDPs discretizando os domínios espacial e temporal, e então aproximando a solução e suas derivadas em cada ponto da grade. O método é implícito, o que significa que ele usa os valores da solução do próximo passo nas equações de diferenciação, e é considerado mais preciso e estável, mas computacionalmente mais caro do que o método explícito.

A aproximação de diferença finita central da derivada no intervalo é dada pela Equação (2.31)

$$\frac{\partial u}{\partial x}(\xi_m, z_j) \sim \frac{u(x_m, z_j) - u(x_{m-1}, z_j)}{\Delta x_m} \tag{2.31}$$

em que, $\Delta x_m = x_m - x_{m-1}$

De modo semelhante, a derivada de segunda ordem com relação a z é definida no ponto (ξ_m, z_j) como uma média ponderada das derivadas nos pontos (x_{m-1}, z_j) e (x_m, z_j), e a aproximação de diferença finita central da derivada no intervalo é dada por (LEVY, 2000)

$$\frac{\partial^2 u}{\partial z^2}(\xi_m, z_j) \sim \frac{u(\xi_m, z_{j+1}) + u(\xi_m, z_{j-1}) - 2u(\xi_m, z_j)}{\Delta z^2} \tag{2.32}$$

Sabendo que $u(\xi_m, z_j)$ pode ser escrito como a Equação (2.33)

$$u(\xi_m, z_j) = \frac{u_j^m + u_j^{m-1}}{2} \tag{2.33}$$

Pode-se reescrever as Equações (2.31) e (2.32), respectivamente, como as Equações (2.34) e (2.35)

$$\frac{\partial u}{\partial x}(\xi_m, z_j) \sim \frac{u_j^m - u_j^{m-1}}{\Delta x_m} \tag{2.34}$$

$$\frac{\partial^2 u}{\partial z^2}(\xi_m, z_j) \sim \frac{1}{2\Delta z^2}[u_{j+1}^m + u_{j-1}^m + u_{j-1}^{m-1} + u_{j-1}^{m-1} - 2(u_j^m + u_j^{m-1})] \tag{2.35}$$

Vale ressaltar que a expressão 2.32 só faz sentido se todos os pontos envolvidos estiverem dentro do domínio, o que significa que j não pode ser 0(solo) ou N(Fronteira superior), expressões especificas devem ser utilizadas para incluir as condições de contorno adequadas.

2.3.4 Resolução para ângulo amplo

Para resolução de ângulo amplo será necessário utilizar a aproximação de Padé-(1,1), de acordo com a Equação (2.36):

$$\sqrt{1+Z} = \frac{1 + \frac{3}{4}Z}{1 + \frac{1}{4}Z} \tag{2.36}$$

em que,

$$Z = \frac{1}{k_0^2}\frac{\partial^2}{\partial z^2} + n(x,z) - 1. \tag{2.37}$$

e, n é o indice de refração.

Desse modo a equação parabólica de (2.26) pode ser desenvolvida como a Equação (2.38)

$$(4+Z)\frac{\partial u}{\partial x}(x,z) = 2jk_0 Zu(x,z) \tag{2.38}$$

Substituindo o valor da variável Z, a Equação (2.38) pode ser reescrita como a Equação (2.39).

$$\frac{\partial^3 u}{\partial x \partial z^2} - 2jk_0\frac{\partial^2 u}{\partial z^2} + k_0^2\left(n^2+3\right)\frac{\partial u}{\partial x} - 2jk_0^3\left(n^2-1\right)u = 0 \tag{2.39}$$

A Equação (2.39) é chamada de equação de Claerbout e apresenta resultado com erros aceitáveis para ângulos de propagação até cerca de 45° da direção paraxial (LEVY, 2000).

2.3.4.1 Equações de Fronteira

Para o caso j = 0 , que representa a fronteira inferior (ar/solo), devem ser utilizadas as condições de contorno de Leontovich apropriadas que, para um terreno plano, pode ser escrito como a Equação (2.40)

$$\frac{\partial \psi(x,0)}{\partial z} + jk_0\eta(x)\psi(x,0) \tag{2.40}$$

em que, $\eta(x)$ é dado pela Equação (2.41)

$$\eta(x) = \begin{cases} \sqrt{\varepsilon(x)-1} & \text{Pol. Horizontal} \\ \frac{\sqrt{\varepsilon(x)-1}}{\varepsilon(x)} & \text{Pol. Vertical} \end{cases} \tag{2.41}$$

e $\varepsilon(x)$ é a permissividade relativa complexa do solo.

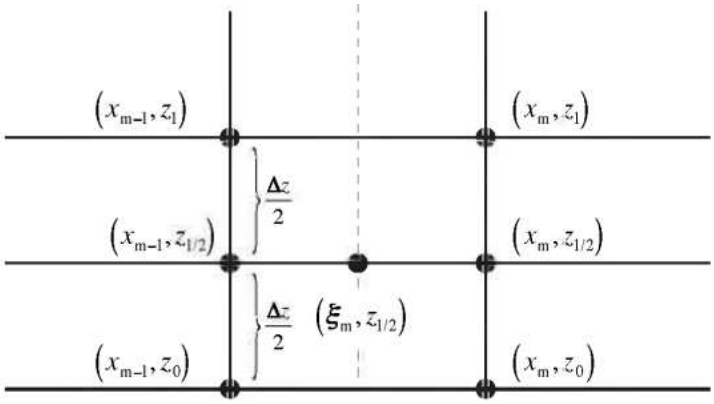
Como mostra a Equação (2.21), a condição de contorno para o campo $u(x,z)$ é a mesma que para $\psi(x,z)$, com isso tem-se a Equação (2.42)

$$\frac{\partial u(x,0)}{\partial z} = -jk_0\eta(x)u(x,0) \tag{2.42}$$

Agora, deve-se discretizar a equação parabólica no ponto $(\xi_m, 0)$ e, para isso, é necessário fazer uma aproximação unilateral para a derivada de segunda ordem (VASCONCELOS, 2017).

Considerando o ponto médio $(\xi_m, z_{1/2})$, como indica a Figura 4.

Figura 4 – Aproximação unilateral na fronteira inferior



Fonte: (VASCONCELOS, 2017)

Tem-se a aproximação mostrada na Equação (2.43)

$$\frac{\partial^2 u}{\partial z^2}(x,0) \cong \frac{\partial^2 u}{\partial z^2}(x, \Delta z/2) \tag{2.43}$$

Desenvolvendo a expressão (2.43), chega-se na Equação (2.44).

$$\frac{\partial^2 u}{\partial z^2}(x, \Delta z/2) = \frac{2}{\Delta z}\left[\frac{\partial u}{\partial z}(x_{\mathrm{m}}, \Delta z/2) - \frac{\partial u}{\partial z}(x,0)\right] \tag{2.44}$$

Substituindo a condição de contorno obtida pela Equação (2.42) na Equação (2.44), desenvolve-se a Equação (2.45)

$$\frac{\partial^2 u}{\partial z^2}(x, \Delta z/2) = \frac{2}{\Delta z}\left[\frac{\partial u}{\partial z}(x_{\mathrm{m}}, \Delta z/2) + jk_0\eta(x)u(x,0)\right] \tag{2.45}$$

Agora, para incluir a condição de contorno no esquema de diferenças finitas, basta substituir a Equação (2.45) na Equação (2.39), a partir disto desenvolve-se a Equação 2.46

$$\begin{aligned} &\frac{2}{\Delta z}\left[\frac{\partial^2 u(x,\Delta z/2)}{\partial x \partial z} + jk\eta(x)\frac{\partial u(x,0)}{\partial x} + jk\frac{\partial \eta(x)}{\partial x}u(x,0)\right]\cdots \\ &\cdots \quad -2jk\frac{2}{\Delta z}\left[\frac{\partial u(x,\Delta z/2)}{\partial z} + jk\eta(x)u(x,0)\right]\cdots \\ &\cdots \quad +k^2\left(n^2+3\right)\frac{\partial u(x,0)}{\partial x} - 2jk^3\left(n^2-1\right)u(x,0) = 0 \end{aligned} \tag{2.46}$$

em que, se η for constante em cada segmento linear, o termo contendo $\frac{\partial \eta(x)}{\partial x}$ pode ser removido. No entanto, é possível manter este termo, já que não oferece grandes dificuldades ao elaborar o esquema de diferenças finitas. Além disso, se x é o ponto médio dos intervalos x_m e $xm-1$, $\frac{\partial \eta(x)}{\partial x}$ é simplificado por $(\eta(x_m)-\eta(x_{m-1})/\Delta x$, em que $\Delta_x = x_m - x_{m-1}$ é o passo no intervalo. No método apresentado nas seções seguintes será considerado que $\eta(x)$ é constante sobre cada segmento linear com o valor do primeiro ponto da grade no segmento, ou seja, $\eta(x) = \eta(x_{m-1})$ para $x_{m-1} \leq x < x_m$ (HOLM, 2007).

Para melhor visualização e entendimento do método para implementação computacional, podemos representar o sistema de equações através de equações matriciais da forma como mostra a Equação (2.47)

$$\mathbf{A}^m \mathbf{U}^m = \mathbf{B}^m \mathbf{U}^{m-1} \tag{2.47}$$

em que,

$$\mathrm{U}^m = \begin{bmatrix} u_0^m \\ u_1^m \\ \vdots \\ u_j^m \\ \vdots \\ u_N^m \end{bmatrix} \quad \mathrm{U}^{m-1} \begin{bmatrix} u_0^{m-1} \\ u_1^{m-1} \\ \vdots \\ u_j^{m-1} \\ \vdots \\ u_N^{m-1} \end{bmatrix} \tag{2.48}$$

$$\mathbf{A}^m = \begin{bmatrix} \alpha_0^m & c & 0 & \cdots & & & & 0 \\ c & \alpha_1^m & c & 0 & & & & \\ 0 & c & \ddots & c & 0 & & & \vdots \\ 0 & 0 & c & \ddots & c & 0 & & \\ \vdots & & & c & \ddots & c & 0 & \\ & & & & c & \alpha_j^m & c & 0 \\ & & & & & c & \ddots & c \\ & & & & & 0 & c & \alpha_N^m \end{bmatrix} \tag{2.49}$$

Na representação da Equação (2.50) a notação "˜" representa que quando uma função depender de Δ_x, ou seja, $F(\Delta_x)$. Então $\tilde{F} = F(-\Delta_x)$.

$$\mathbf{B}^m = \begin{bmatrix} \tilde{\alpha}_0^m & \tilde{c} & 0 & \cdots & & & & 0 \\ \tilde{c} & \tilde{\alpha}_1^m & \tilde{c} & 0 & & & & \\ 0 & \tilde{c} & \ddots & \tilde{c} & 0 & & & \vdots \\ 0 & 0 & \tilde{c} & \ddots & \tilde{c} & 0 & & \\ \vdots & & & \tilde{c} & \ddots & \tilde{c} & 0 & \\ & & & & \tilde{c} & \tilde{\alpha}_j^m & \tilde{c} & 0 \\ & & & & & \tilde{c} & \ddots & \tilde{c} \\ & & & & & 0 & \tilde{c} & \tilde{\alpha}_N^m \end{bmatrix} \tag{2.50}$$

As variáveis das matrizes (2.49) e (2.50) são definidas pelo conjunto de Equações (2.51) a (2.55)

$$\alpha_j^m = \begin{cases} D_j^m/2 + ik_0 c\eta_m \Delta z & \text{for } j = 0 \\ D_j^m & \text{for } j = 1,2,3,\ldots \end{cases} \tag{2.51}$$

$$D_j^m = c\left(a_j^m + b - 2\right) \tag{2.52}$$

$$c = 1 - ik_0\Delta x \tag{2.53}$$

$$a_j^m = k^2\Delta z^2 \left[\left(\frac{n_j^m + n_j^{m-1}}{2} \right)^2 - 1 \right] \tag{2.54}$$

$$b = \frac{4k^2\Delta z^2}{c} \tag{2.55}$$

2.3.5 Fronteira Superior

Para condição de fronteira superior é utilizada uma camada absorvedora que estende o domínio de cálculo e aplica um filtro à equação parabólica para absorver a energia incidente na fronteira superior e evitar sua reflexão de volta. O filtro mais simples e eficaz utilizado é a janela de Hanning, mostrado na Equação (2.56)

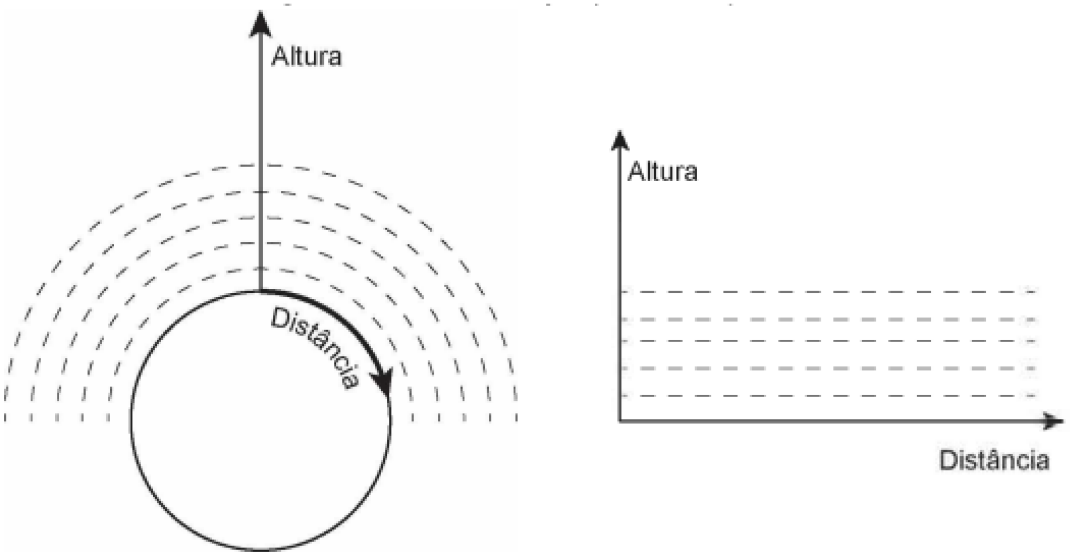
$$w(n) = 0.5 - 0.5\cos\left(\frac{2\pi n}{N-1}\right) \tag{2.56}$$

em que, n é o índice do elemento da janela e N é o tamanho da janela.

2.4 Conversão para geometria de Terra plana

A transformação da Terra esférica para a Terra plana indicada na figura (5) é uma solução mais comum e econômica para se determinar as condições de onda. Esta ideia foi introduzida por Pekeris em 1946 e ainda é amplamente utilizada. Embora a equação da onda seja modificada, ela ainda é válida em altitudes baixas se o índice de refração original é substituído pelo índice de refração modificado. Essa transformação permite uma análise mais fácil e precisa das condições de onda, especialmente em baixas altitudes.

Figura 5 – Transformação para Terra plana



Fonte: (LEVY, 2000).

A Equação (2.57) apresenta a modificação ideal proposta por Pekeris, mas ela não pode ser usada sem causar distorções na equação da onda devido a sua natureza não conformal. Em vez de utilizar a equação (2.57), uma transformação diferente, definida pela equação (2.58), é empregada (VASCONCELOS, 2017).

$$m(x,h) = n(x,h) + \frac{h}{a} \tag{2.57}$$

em que, a é o raio médio da Terra e h é a altitude acima da superfície da Terra.

$$\nabla^2(x,z) = e^{\frac{2z}{a}} \nabla^2(X,Z) \tag{2.58}$$

Substituindo o Laplaciano da Equação (2.58) na Equação da aproximação parabólica

(2.59) (VASCONCELOS, 2017), obtém-se a equação da onda no novo sistema de coordenadas na Equação (2.60).

$$\frac{\nabla^2 \psi(x,z)}{\partial z^2} + k_0^2 n(x,z)^2 \psi = 0 \tag{2.59}$$

$$\nabla^2 \psi(x,z) + k_0^2 \tilde{n}^2(x,z) e^{\frac{2z}{a}} \psi(X,Z) = 0 \tag{2.60}$$

Sendo,

$$\tilde{m}(x,z) = \tilde{n}(x,z) e^{\frac{z}{a}} \tag{2.61}$$

tem-se a Equação (2.62).

$$\nabla^2 \psi(x,z) + k_0^2 \tilde{m}^2(x,z) \psi(X,Z) = 0 \tag{2.62}$$

2.5 Perda de Percurso e Fator de Propagação

A perda de percurso é um fator extremamente importante em aplicações de comunicação sem fio pois ela afeta diretamente a qualidade da comunicação. A perda de percurso ocorre quando a energia de uma onda de rádio é dissipada ao longo do caminho entre o transmissor e o receptor, causando uma diminuição no sinal recebido. Isso pode levar a problemas como baixa velocidade de transmissão de dados, interferência e até mesmo perda de comunicação. Portanto, é importante considerar a perda de percurso ao projetar e implementar sistemas de comunicação sem fio para garantir uma boa qualidade de comunicação.

Ao avaliar a performance de um enlace, é importante considerar tanto os efeitos da propagação quanto os efeitos das antenas, nesse cenário separar os efeitos da propagação dos efeitos das antenas pode ajudar a identificar as principais fontes de perda de desempenho em um enlace de comunicação sem fio.

A perda de transmissão em um enlace de comunicação sem fio é dada pela soma da perda básica de propagação e o ganho das antenas. A perda básica de propagação é a perda de sinal devido aos efeitos da propagação, como atenuação devido a obstáculos físicos, absorção e reflexão. Por outro lado, o ganho das antenas é a capacidade das antenas de concentrar e direcionar o sinal transmitido e recebido.

A perda de transmissão é geralmente medida em dB (decibéis) e pode ser expressa como:

$$L_{total} = L_B - G_T - G_R \tag{2.63}$$

em que G_T e G_R são os ganhos das antenas transmissora e receptora, respectivamente. E L_B é a perda básica que pode ser calculada como a Equações (2.64)

$$L_B = 20log_{10}\left(\frac{4\pi R}{\lambda}\right) \tag{2.64}$$

Em ambientes com fortes efeitos refrativos da atmosfera, é necessário considerar a perda de percurso, em dB é a diferença entre a potência efetivamente radiada pelo transmissor (AEIRP) e a potência recebida pelo receptor, ou seja, a razão entre eles. A perda de percurso é influenciada pelos efeitos da propagação, incluindo reflexão, refração, difração e absorção. Além disso, a perda de percurso também é afetada pela orientação e polarização das antenas.

Em uma distância r da fonte, a densidade de potência isotropicamente radiada pela antena isotrópica equivalente é dada pela Equação (2.65)

$$W_0 = \frac{AEIRP}{4\pi r^2} \tag{2.65}$$

A intensidade de radiação também pode ser relacionada com o campo distante, assim como na Equação (2.66)

$$U(\theta,\phi) = \frac{|E^{\circ}(\theta,\phi)|^2}{2Z} \tag{2.66}$$

A densidade de potência também pode ser escrita como:

$$W(\theta,\phi) = \frac{U(\theta,\phi)}{r^2} \cong \frac{|E^{\circ}(\theta,\phi)|^2}{2Zr^2} \tag{2.67}$$

Considerando simetria azimutal, tem-se:

$$W(\theta) = \frac{U(\theta)}{r^2} \cong \frac{|E^{\circ}(\theta)|^2}{2Zr^2} \tag{2.68}$$

A intensidade de radiação de campo distante é a grandeza utilizada para a construção dos diagramas de radiação das antenas. Então, pode-se substituir o termo $|E^{\circ}(\theta)|^2$ por $|B(\theta)|^2$,

em que $B(\theta)$ é a função que descreve o diagrama de radiação da antena (VASCONCELOS, 2017).

Com isso tem-se:

$$W_{max} = \frac{|B_{max}|^2}{2Zr^2} \tag{2.69}$$

Igualando 2.69 com 2.65, temos:

$$AEIRP = \frac{2\pi|B_{max}|^2}{Z} \tag{2.70}$$

Como perda de percurso não é um valor absoluto, mas sim uma razão de potências, pode-se definir $B_{max} = \frac{1}{\sqrt{2\pi}}$, com isso tem-se a Equação (2.71)

$$AEIRP = \frac{1}{Z} \tag{2.71}$$

em que Z é a impedância intrínseca do meio, que geralmente é denotada por η.

2.5.0.1 Polarização Horizontal

A parte imaginária do vetor de Poynting representa a troca de energia entre a onda eletromagnética e a fonte que a gerou, e essa troca é negligenciável na região de campo distante. Portanto, a densidade de potência total é dada pela magnitude do vetor de Poynting, que é proporcional à intensidade da onda eletromagnética. Portanto, tem-se a Equação (2.72)

$$\vec{W}(X,Z) = p_{med} = \frac{1}{2}\mathrm{Re}\{\mathbf{E}(X,Z) \times \mathbf{H}^*(X,Z)\} \tag{2.72}$$

No campo distante, a Equação (2.72) se torna, aproximadamente, a Equação (2.73)

$$\vec{W}(X,Z) = \frac{1}{2}[\mathbf{E}(X,Z) \times \mathbf{H}^*(X,Z)] \tag{2.73}$$

A magnitude da densidade de potência é dada por:

$$W(X,Z) = \frac{1}{2}||\mathbf{E}(X,Z)||\mathbf{H}^*(X,Z)|\sin\phi| \tag{2.74}$$

Considerando que, no campo distante, a onda é aproximadamente plana, os vetores de campo elétrico e magnético são ortogonais entre si e o ângulo entre eles é de 90º.

$$W(X,Z) = \frac{1}{2}|\mathbf{E}(X,Z)||\mathbf{H}^*(X,Z)| \tag{2.75}$$

Sabendo que a relação do campo elétrico com o campo magnético em uma onda plana é dado pela Equação (2.76)

$$|\mathbf{H}^*(X,Z)| = \frac{1}{\eta}|\mathbf{E}(X,Z)| \tag{2.76}$$

em que η é a impedância intrínseca do meio. Substituindo 2.76 em 2.75, tem-se o mostrado na Equação (2.77)

$$W(X,Z) = \frac{1}{2\eta}|\mathbf{E}(X,Z)|^2 \tag{2.77}$$

A potência recebida por uma antena receptora isotrópica no ponto (X,Z) da atmosfera é dada pela Equação (2.78).

$$P_r(X,Z) = W(X,Z)A_{eff} = W(X,Z)\frac{\lambda_0^2}{4\pi} \tag{2.78}$$

Substituindo 2.77 em 2.78, obtém-se a Equação (2.79)

$$P_r(X,Z) = \frac{\lambda_0^2|E(X,Z)|^2}{8\pi\eta} \tag{2.79}$$

Então, a razão entre a potência transmitida e a potência recebida pode ser calculada pela Equação (2.80).

$$\frac{AEIRP}{P_r(X,Z)} = \frac{8\pi}{\lambda_0^2|E(X,Z)|^2} \tag{2.80}$$

Sabendo que, $E(X,Z) = E_\phi(X,Z)$, desenvolve-se a Equação (2.81).

$$\begin{aligned} E_\phi(r,\theta) &= \frac{\psi_\mathrm{h}(r,\theta)}{\sqrt{k_0 r\sin\theta}} \\ E_\phi(X,Z) &= \frac{\psi_\mathrm{h}(X,Z)}{\sqrt{k_0 X}} \\ \frac{AEIRP}{P_\mathrm{r}(X,Z)} &= \frac{8\pi k_0 X}{\lambda_0^2\,|\psi_\mathrm{h}(X,Z)|^2} \\ \frac{AEIRP}{P_\mathrm{r}(X,Z)} &= \frac{(4\pi)^2 X}{\lambda_0^3\,|\psi_\mathrm{h}(X,Z)|^2} \end{aligned} \tag{2.81}$$

Agora será necessário transformar a componente $\psi_h(X,Z)$ em coordenadas planas da Terra (x,z), que é dada pelas Equações (2.82) a (2.84)

$$\begin{cases} X = (a+h)sin(\theta) \\ Z = (a+h)cos(\theta) \end{cases} \tag{2.82}$$

$$\begin{cases} x = a\theta \\ z = a\ln\left(1+\frac{h}{a}\right) \end{cases} \tag{2.83}$$

$$\begin{gathered} X = \left(a + ae^{\frac{z}{a}} - a\right)\sin\left(\frac{x}{a}\right) \\ X = ae^{\frac{z}{a}}\sin\left(\frac{x}{a}\right) \end{gathered} \tag{2.84}$$

Com isso, podemos reescrever a Equação (2.81) como a Equação (2.85)

$$\frac{AEIRP}{P_{\mathrm{r}}(X,Z)} = \frac{(4\pi)^2 ae^{\frac{z}{a}}\sin\left(\frac{x}{a}\right)}{\lambda_0^3 \mid \psi_{\mathrm{h}}(x,z)^2} \tag{2.85}$$

A partir da Equação (2.85) podemos, finalmente, encontrar a perda de percurso em dB mostrada nas Equações (2.86) a (2.87).

$$L_{\mathrm{P}}(x,z) = 10\log_{10}\left[\frac{(4\pi)^2 ae^{\frac{z}{a}}\sin\left(\frac{x}{a}\right)}{\lambda_0^3\left|\psi_{\mathrm{h}}(x,z)\right|^2}\right] \tag{2.86}$$

$$L_{\mathrm{P}}(x,z) = -20\log_{10}\left|\psi_{\mathrm{h}}(x,z)\right| + 20\log_{10}(4\pi) + 10\log_{10}\left[ae^{\frac{z}{a}}\sin\left(\frac{x}{a}\right)\right] - 30\log_{10}(\lambda_0) \tag{2.87}$$

Como $\psi(x,z) = u(x,z)e^{jk_0x}$, $|\psi(x,z)| = |u(x,z)|$, tem-se a Equação (2.88)

$$L_{\mathrm{P}}(x,z) = -20\log_{10}|u(x,z)| + 20\log_{10}(4\pi) + 10\log_{10}\left[ae^{\frac{z}{a}}\sin\left(\frac{x}{a}\right)\right] - 30\log_{10}(\lambda_0) \tag{2.88}$$

E o fator de propagação também pode ser calculado pelas Equações (2.89) e (2.90)

$$PF(x,z) = L_{\mathrm{B}}(x,z) - L_{\mathrm{P}}(x,z)$$

$$\begin{gathered} PF(x,z) = 20\log_{10}(4\pi) + 20\log_{10}(R) - 20\log_{10}\lambda_{-\ldots} \\ \cdots - \left[-20\log_{10}|u(x,z)| + 20\log_{10}(4\pi) + 10\log_{10}\left[ae^{\frac{z}{a}}\sin\left(\frac{x}{a}\right)\right] - 30\log_{10}(\lambda_0)\right] \end{gathered} \tag{2.89}$$

$$PF(x,z) = 20\log_{10}|u(x,z)| - 10\log_{10}\left[ae^{\frac{z}{a}}\sin\left(\frac{x}{a}\right)\right] + 10\log_{10}(\lambda_0) + 10\log_{10}\left(x^2+z^2\right) \tag{2.90}$$

2.5.0.2 Polarização Vertical

Assim como para polarização horizontal as Equações (2.88) e (2.90) também são válidas para polarização vertical, para isso basta utilizar o magnético como ponto de partida.

2.6 Modelagem de Terreno Irregular

2.6.1 Aproximação de primeira ordem da equação de ângulo amplo para terreno irregular

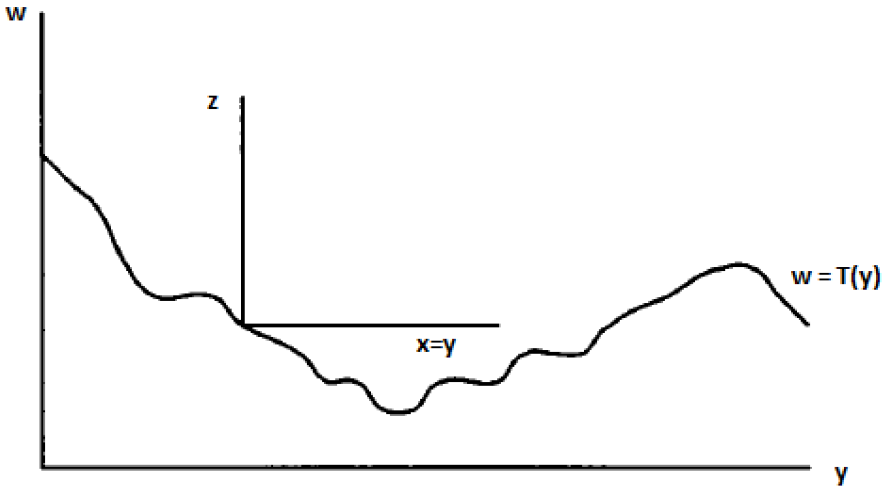
Para esta aproximação será utilizada a técnica apresentada por Donohue e Kuttler (DONOHUE; KUTTLER, 2000) que é um método para incorporar variações do terreno na equação de ângulo amplo para terreno. Esse método pode fornecer uma previsão mais precisa do comportamento de ondas eletromagnéticas sobre terrenos acidentados.

Assumindo que a altura do terreno é dada por uma equação da forma como mostra a Equação (2.91).

$$w = T(y) \tag{2.91}$$

em que, w e y são as coordenadas fixas (não mapeadas) como mostra a Figura 6

Figura 6 – Ilustração dos sistemas de coordenadas mapeados (x,z) e não mapeados (u,v) com um perfil de terreno



Fonte: (DONOHUE; KUTTLER, 2000)

Aplicando uma transformação da coordenada de altura mostrada na Equação (2.91), obtém-se uma equação mais geral da Equação (2.23), como mostra a Equação (2.92).

$$\left[\frac{\partial}{\partial x}+j\frac{\partial\theta}{\partial x}-T'\left(\frac{\partial}{\partial z}+j\frac{\partial\theta}{\partial z}\right)-j\sqrt{\left(\frac{\partial}{\partial z}+j\frac{\partial\theta}{\partial z}\right)^2+k_0^2n^2}\right]u=0 \tag{2.92}$$

em que, $T'=dT/dy=dT/dx$ é a inclinação do terreno, $x=y$, $z=w-T(y)$. Como mostrado anteriormente, $T(y)$ é o perfil do terreno.

A função $u(x,z)$ se relaciona com $\psi(x,z)$ pela expressão $\psi(x,z)=u(x,z)e^{j\theta}$, em que o fator de fase é dado pela Equação (2.93)

$$\theta(x,z)=k_0zT'(x)+f(x) \tag{2.93}$$

e $f(x)$ é uma função arbitrária.

Para um terreno linear por partes, $\theta(x,z)$ pode ser escrito pela Equação (2.94)

$$\theta(x,z)=k_0Sz+f(x) \tag{2.94}$$

em que, a inclinação do terreno $S=T'(x)$ é constante em cada segmento linear.

Usando as Equações (2.37) e (2.94), pode-se reescrever a Equação 2.92 como a Equação 2.95

$$\left[\frac{\partial}{\partial x}+jf'-S\left(\frac{\partial}{\partial z}+jk_0S\right)-jk\sqrt{1+Z+\zeta}\right]u=0 \tag{2.95}$$

em que,

$$\zeta=\frac{2jS}{k_0}\frac{\partial}{\partial z}-S^2 \tag{2.96}$$

A raiz da Equação (2.95) pode ser aproximada pela Equação (2.97) (HOLM, 2007).

$$\sqrt{1+Z+\zeta}\cong\sqrt{1+Z}\left(1+\frac{\zeta}{2(1+Z)}\right)\cong\sqrt{1+Z}+\frac{\zeta}{2} \tag{2.97}$$

Substituindo a Equação (2.97) na Equação (2.95), tem-se a Equação 2.98

$$\left[\frac{\partial}{\partial x}+jk_0\left(\frac{f'}{k_0}-\frac{S^2}{2}-\sqrt{1-Z}\right)\right]u=0 \tag{2.98}$$

Para obter a Equação (2.98), foi considerado apenas uma expansão de primeira ordem em relação ao operador ζ, ou seja, apenas os termos de primeira ordem são considerados

em relação à inclinação do terreno. Contudo, para o termo $\sqrt{1-Z}$ será feita a aproximação de padé mostrada na Equação (2.36), que permite ângulos de propagação de 45° a partir da direção paraxial (LEVY, 2000), para pequenos valores na inclinação do terreno.

O termo negativo da Equação (2.24) pode ser escrito em função da Equação (2.37), com isso tem-se a Equação (2.99)

$$\left(\frac{\partial}{\partial x}+ik(1-Q)\right)u=\left[\frac{\partial}{\partial x}+ik(1-\sqrt{1+Z})\right]u=0 \tag{2.99}$$

Comparando a Equação (2.99) com a Equação(2.98), podemos encontrar o valor arbitrário da função $f'(x)$, como mostra a Equação (2.100)

$$\frac{f'}{k_0}-\frac{S^2}{2}=1 \tag{2.100}$$

Desenvolvendo a Equação (2.100), pode-se encontrar o valor da função $f(x)$, como mostra a Equação (2.101)

$$f(x)=k_0x+0.5k_0S^2x \tag{2.101}$$

Substituindo a Equação (2.101) na Equação (2.94), tem-se a Equação (2.102)

$$\theta(x,z)=k_0(Sz+x+0.5S^2x) \tag{2.102}$$

2.6.1.1 Equações de Fronteira

Para uma fronteira com inclinação fixa S, a condição de Leontovich pode ser escrita como a Equação (2.103) (HOLM, 2007)

$$\frac{\partial\psi(x,0)}{\partial n}+\frac{jk_0\eta(x)\psi(x,0)}{\sqrt{1+S^2}}=0 \tag{2.103}$$

em que, $\frac{\partial}{\partial n}$ é a derivada normal externa a superfície e o fator $\frac{1}{\sqrt{1+S^2}}$ é adicionado para diminuir os componentes de retroespalhamento.

Usando $x=u$ e $z=v-T(u)$ a derivada normal pode ser escrita como a Equação (2.104)

$$\frac{\partial}{\partial n} = \frac{1}{\sqrt{1+S^2}}\left(\frac{\partial}{\partial v} - S\frac{\partial}{\partial u}\right) = \frac{1}{\sqrt{1+S^2}}\left((1+S^2)\frac{\partial}{\partial z} - S\frac{\partial}{\partial x}\right) \tag{2.104}$$

Substituindo a expressão $\psi(x,z) = u(x,z)e^{j\theta}$ na Equação (2.103),e sabendo que θ é dado pela Equação (2.102), tem-se a Equação (2.105)

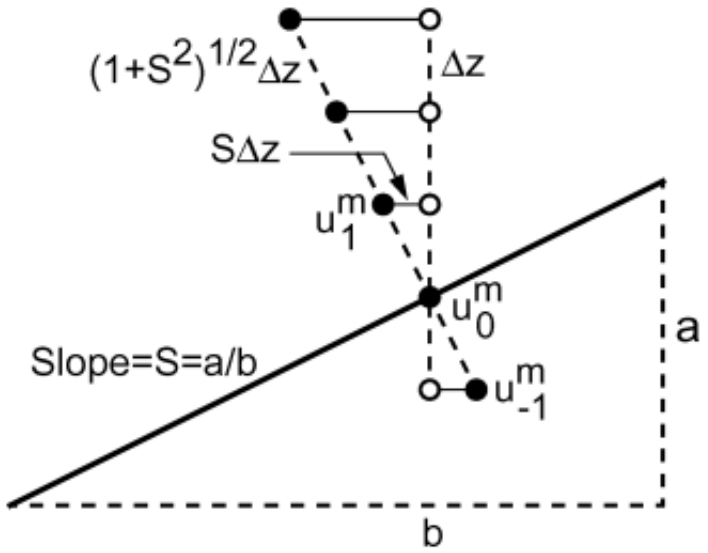
$$\frac{\partial u(x,0)}{\partial n} = -\left(j\frac{\partial \theta}{\partial n} + \frac{jk_0\eta(x)}{\sqrt{1+S^2}}\right)u(x,0) = -jk_0\left(\eta(x) + \frac{S^3}{2}\right)\frac{u(x,0)}{\sqrt{1+S^2}} \tag{2.105}$$

A derivada de segunda ordem da Equação (2.105) é dada pela Equação (2.106)

$$\frac{\partial^2 u(x,0)}{\partial n^2} = -k_0^2\left(\eta(x) + \frac{S^3}{2}\right)^2\frac{u(x,0)}{1+S^2} \tag{2.106}$$

Matematicamente, seria possível reescrever a derivada $\frac{\partial u(x,0)}{\partial n}$ usando a Equação (2.104). No entanto, isso resultará em uma derivada problemática em relação a x. Felizmente, é possível fazer algumas aproximações. Fisicamente, antes que o esquema de diferenças seja resolvido, a fase do campo é deslocada. Essa mudança corresponde a uma inclinação do campo, pelo menos no que diz respeito aos pontos de campo mais próximos da fronteira. Ademais, para que isso funcione, não pode haver raios com ângulos grandes, pois a mudança de fase corresponde apenas a inclinação do campo para raios horizontais e apenas uma aproximação para raios com ângulos maiores (HOLM, 2007).

Figura 7 – Ilustração da mudança de fase da onda, em que Δ_z é o passo de altura.



Fonte: (HOLM, 2007)

Na Figura 7, a inclinação do campo próximo à fronteira é ilustrada. De acordo com a Figura 7, e fazendo $u_i^m = u(m\Delta_x, i\Delta_z)$ em que Δ_x é o passo em alcance e Δ_z é o degrau em altura, a derivada normal de primeira e segunda ordem de $u(x_m, 0)$ pode ser aproximada, respectivamente, pelas Equações (2.107) e (2.108).

$$\frac{\partial u(x_m,0)}{\partial n} \approx \frac{u_1^m - u_{-1}^m}{2\Delta z\sqrt{1+S^2}} \approx -\left(\eta + \frac{S^3}{2}\right)\frac{jku_0^m}{\sqrt{1+S^2}} \tag{2.107}$$

$$\frac{\partial^2 u(x_m,0)}{\partial n^2} \approx \frac{u_1^m - 2u_0^m + u_{-1}^m}{\Delta z^2(1+S^2)} \approx -\left(\eta + \frac{S^3}{2}\right)^2\frac{k^2u_0^m}{(1+S^2)} \tag{2.108}$$

em que, u_1^m, u_0^m, u_{-1}^m, representa respectivamente, $u(x_m, \Delta_z), u(x_m, 0)u(x_m, -\Delta_z)$

Para a iézima linha das matrizes mostradas nas Equações (2.49) e (2.50), tem-se a Equação (2.109)

$$cu_{i-1}^m + D_i^m u_i^m + cu_{i+1}^m = \tilde{c}u_{i-1}^{m-1} + \tilde{D}_i^m u_i^{m-1} + \tilde{c}u_{i+1}^{m-1} \tag{2.109}$$

Para a primeira linha, ou seja, $z = z_0 = 0$, tem-se a Equação 2.110

$$cu_{-1}^m + D_0^m u_0^m + cu_1^m = \tilde{c}u_{-1}^{m-1} + \tilde{D}_0^m u_0^{m-1} + \tilde{c}u_1^{m-1} \tag{2.110}$$

agora, basta substituir a Equação (2.107) na Equação (2.110) para encontrar sua solução. Para $S = 0$, o resultado será exatamente o mesmo da Equação (2.51).

A abordagem acima para a condição de contorno funcionará. Porém, ao exigir que a derivada de segunda ordem da Equação (2.108) seja atendida, o esquema de diferença resultante produzirá um resultado um pouco melhor. Assim, das Equações (2.107) e (2.108), tem-se a Equação (2.111).

$$\begin{cases} u_{-1}^m - 2jk_0\Delta z\left(\eta + \frac{1}{2}S^3\right)u_0^m \approx u_1^m \\ u_{-1}^m - 2u_0^m + k_0^2\Delta z^2\left(\eta + \frac{1}{2}S^3\right)^2 u_0^m \approx -u_1^m \end{cases} \tag{2.111}$$

Conhecendo-se o valor de u_1^m, a Equação (2.111) pode ser escrita em função dele para determinar os valores de u_0^m e u_{-1}^m, como mostra a Equação (2.112)

$$\begin{cases} u_0^m = \frac{u_1^m}{1-\kappa+\kappa^2/2} \\ u_{-1}^m = \frac{1+\kappa+\kappa^2/2}{1-\kappa+\kappa^2/2} u_1^m \end{cases} \tag{2.112}$$

em que, $\kappa = jk_0\Delta_z(\eta + S^3/2)$.

Nessa solução o esquema de diferenças começa na segunda linha e não na primeira, o que significa que a matriz $\mathbf{A}^m$ é lida como a Equação (2.113)

$$\mathbf{A}^m = \begin{bmatrix} \alpha_1^m & c & 0 & \cdots & & & & 0 \\ c & \alpha_2^m & c & 0 & & & & \\ 0 & c & \ddots & c & 0 & & & \vdots \\ 0 & 0 & c & \ddots & c & 0 & & \\ \vdots & & & c & \ddots & c & 0 & \\ & & & & c & \alpha_j^m & c & 0 \\ & & & & & c & \ddots & c \\ & & & & & 0 & c & \alpha_N^m \end{bmatrix} \tag{2.113}$$

Com isso, os elementos α_j^m da matriz $\mathbf{A}^m$ da Equação (2.113) pode ser escrito a partir da Equação (2.114).

$$\alpha_j^m = \begin{cases} D_j^m + \frac{c}{1-\kappa_m+\kappa_m^2/2} & \text{for } j = 1 \\ D_j^m & \text{for } j = 2,3,4\ldots \end{cases} \tag{2.114}$$

A matriz $\mathbf{B}^m$, assim como apresentado anteriormente, é escrita como $\mathbf{B}^m = \tilde{\mathbf{A}}^m$

Quando o esquema de diferenças é resolvido, o campo na fronteira é finalmente encontrado a partir da Equação (2.112).

2.7 Considerações finais

Em resumo, este capítulo abordou conceitos fundamentais da propagação de ondas eletromagnéticas, incluindo as equações matemáticas que as governam, técnicas numéricas para resolvê-las e ferramentas para modelar as condições de contorno.

Foi mostrado que o método das diferenças finitas é uma técnica utilizada para resolver essas equações numericamente, permitindo simulações precisas da propagação das ondas eletromagnéticas em diferentes condições. Além disso, foi demonstrado as equações de

fronteira utilizadas para modelar as condições de contorno em sistemas de propagação de ondas eletromagnéticas que fugia do domínio discreto apresentado pelo método das diferenças finitas.

Por fim, a representação matricial é outra ferramenta valiosa na análise de sistemas de propagação de ondas eletromagnéticas, permitindo a representação eficiente e compacta das equações e condições de contorno, e será utilizada na implementação dos códigos que serão abordados no capítulo 5.

3 MÉTODOS DE SOLUÇÃO DO SISTEMA TRIDIAGONAL

3.1 Sistema Linear Tridiagonal

Um sistema linear tridiagonal é um sistema de equações lineares que pode ser escrito na forma A.X = D, em que A é uma matriz tridiagonal, ou seja, uma matriz cujos elementos são não-nulos somente na diagonal principal, diagonal superior e diagonal inferior. A sua representação matricial é mostrada na Equação (3.1).

$$\begin{bmatrix} b_0 & c_0 & 0 & 0 & \cdots & 0 \\ a_1 & b_1 & c_1 & 0 & \cdots & 0 \\ 0 & a_2 & b_2 & c_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{N-1} & b_{N-1} \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ \vdots \\ d_{N-1} \end{bmatrix} \tag{3.1}$$

O sistema tridiagonal apresenta a seguinte relação de recorrência indicada na Equação (3.2).

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = d_i, \tag{3.2}$$

em que, $a_1 = 0$ e $c_n = 0$.

Sistemas tridiagonais são geralmente encontrados em problemas de diferenciação finita, onde uma equação diferencial é discretizada para formar um sistema de equações lineares. A solução deste tipo de sistemas geralmente é feita usando algoritmos específicos, como por exemplo, o algoritmo de Thomas e o Parallel cyclic reduction (PCR).

3.1.1 Algoritmo de Thomas

O algoritmo de Thomas é um método numérico para resolver sistemas de equações lineares tridiagonais. O algoritmo de Thomas é uma versão específica do método de eliminação de Gauss para sistemas tridiagonais. Este algoritmo é particularmente útil para sistemas tridiagonais grandes e esparsos, pois é mais rápido do que os métodos de eliminação de Gauss convencionais e também requer menos memória.

A implementação do algoritmo de Thomas consiste em três etapas:

- Fatoração: O primeiro passo é fatorar a matriz tridiagonal em três matrizes diagonais: uma diagonal, uma diagonal principal e uma superdiagonal. Isso é feito através de uma série de operações elementares de linha que transformam a matriz tridiagonal em uma matriz escalonada superior.
- Substituição progressiva: A segunda etapa é a substituição progressiva, onde usa-se a matriz escalonada superior e o vetor de termos independentes para calcular os valores intermediários de X (vetor de solução). Nessa etapa, começa-se com a primeira equação e utiliza os valores já calculados para resolver a equação seguinte.
- Substituição regressiva: Por fim, a terceira etapa é a substituição regressiva, onde utiliza-se os valores intermediários de x e a matriz escalonada superior para calcular o vetor final de solução. Nessa etapa, começa-se com a última equação e utiliza os valores já calculados para resolver as equações anteriores.

O algoritmo de Thomas pode ser dado pelas as Equações (3.3) e (3.4)

$$c'_i = \begin{cases} \frac{c_i}{b_i} & ; i = 0 \\ \frac{c_i}{b_i - a_i c'_{i-1}} & ; i = 1,2,3,\cdots,N-1 \end{cases} \tag{3.3}$$

$$d'_i = \begin{cases} \frac{d_i}{b_i} & ; i = 0 \\ \frac{d_i - a_i d'_{i-1}}{b_i - a_i c'_{i-1}} & ; i = 1,2,3,\cdots,N-1 \end{cases} \tag{3.4}$$

Finalmente a solução final é obtida pela substituição regressiva como mostra as Equações (3.5) e (3.6)

$$x_i = d'_n \; ; \; i = N-1 \tag{3.5}$$

$$x_i = d'_i - c'_i \cdot x_{i+1} \; ; \; i = N-2, N-3, ..., 0. \tag{3.6}$$

3.1.2 Algoritmo Parallel Cyclic Reduction (PCR)

Parallel Cyclic Reduction é uma técnica utilizada para resolver sistemas de equações lineares em computadores paralelos. Ela é baseada no método conhecido de eliminação de Gauss, mas com algumas modificações que a tornam mais eficiente para a execução paralela.

A ideia básica por trás do PCR é dividir a matriz de coeficientes (que define o sistema de equações lineares) em submatrizes menores, e então realizar a eliminação de Gauss nessas submatrizes em paralelo. A vantagem desse abordagem é permitir um uso mais eficiente dos processadores paralelos, pois cada processador pode trabalhar em uma submatriz separada ao mesmo tempo.

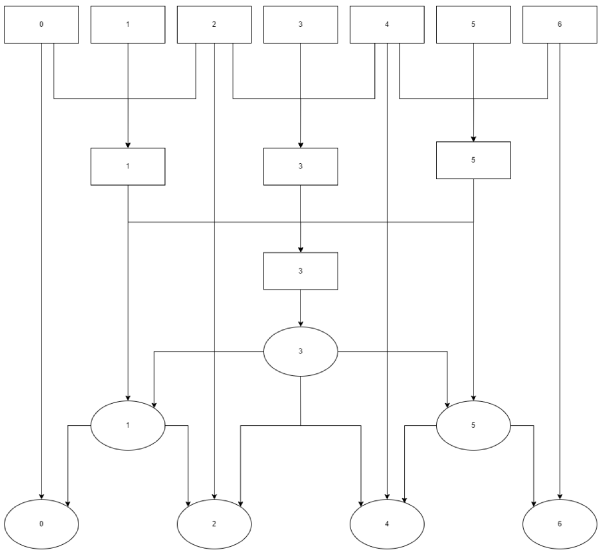
O algoritmo PCR (KULKARNI, 2021), pode ser dividido em duas fases principais: fase de redução e fase de retro-substituição.

Durante a fase de redução, a matriz é dividida em submatrizes e a eliminação de Gauss é realizada em cada submatriz em paralelo. O elemento pivô (ou seja, o elemento usado para eliminar outros elementos na mesma coluna) é escolhido como o elemento com o maior valor absoluto na submatriz. Uma vez que as submatrizes foram reduzidas, elas são combinadas para formar a matriz reduzida.

Na fase de retro-substituição, a matriz reduzida é usada para encontrar a solução do sistema de equações. Isso é feito resolvendo o sistema de equações representado pela matriz reduzida, em um processo semelhante ao método de eliminação de Gauss tradicional.

A Figura 8 ilustra o algoritmo PCR para N = 7.

Figura 8 – Diagrama de redução cíclica para N=7



Fonte: Autor

Vale ressaltar, que para a convergência do algoritmo o número de equações do sistema deve ser igual a $N = 2^k - 1$, sendo $k = 2, 3, 4, ...$ (AMODIO, 1992) (KULKARNI, 2021). Caso não satisfaça esta condição será necessário adicionar equações para completar o sistema da forma como indica a Equação (3.7).

$$x_i = 1; \tag{3.7}$$

sendo $a_i = 0$; $c_i = 0$ e $b_i = 1$.

O Algoritmo (1) mostra o pseudo-código da implementação do PCR.

Algorithm 1 Redução cíclica

Entrada: Matriz de tamanho $N = 2^k - 1$, vetores a[], b[], c[] representando as diagonais da matriz A e o vetor D[], i= 0, 1, ... N - 1
Saída: Vetor coluna x[], i= 0, 1, ... N - 1
Redução

para i $\leftarrow 0$ **até** $log_2(N+1)-1$ **faça**
  $offset \leftarrow 2^j$
  $step \leftarrow 2^{j+1}$
  para j $\leftarrow$ step-1 **até** $N, j = j + step$ **faça**
    $id1 \leftarrow j - offset$
    $id2 \leftarrow j + offset$
    $alpha \leftarrow a[j]/b[id1]$;
    $gamma = c[j]/b[id2]$;
    $a[j] \leftarrow -a[jd1]alpha$;
    $b[j] \leftarrow b[j] - c[id1]alpha - a[id2] * gamma$
    $c[j] \leftarrow -c[jd2]gamma$;
    $d[j] \leftarrow d[j] - d[id1]alpha - d[id2] * gamma$;
  fim para
fim para

$id \leftarrow (N-1)/2$;
$x[id] \leftarrow d[id]/b[id]$;

Retro-substituição

para i $\leftarrow log_2(N+1)-2$ **até** $0, i = i-1$ **faça**
  $offset \leftarrow 2^{i+1}$
  para j $\leftarrow step-1$ **até** $N, j = j + step$ **faça**
    $id1 \leftarrow i - offset$
    $id2 \leftarrow i + offset$
    se $id1 - offset < 0$ **então**
      $x[id1] \leftarrow (d[id1] - c[id1]x[id1 + offset])/b[id1]$
    senão
      $x[id1] \leftarrow d[id1] - a[id1]x[id1 - offset] - c[id1]x[id1 + offset])/b[id1]$;
    fim se
    se $id2 + offset \geq N$ **então**
      $x[id2] \leftarrow (d[id2] - a[id1]x[id2 - offset])/b[id2]$;
    senão
      $x[id2] \leftarrow (d[id2] - a[id2]x[id2 - offset] - c[id2]x[id2 + offset])/b[id2]$;
    fim se
  fim para
fim para

3.2 Considerações Finais

Este capítulo discutiu as soluções para sistemas tridiagonais, especificamente, os algoritmos de Thomas e Parallel Cyclic Reduction . Ambas as técnicas foram mostradas como eficientes para resolver esse tipo de sistemas, com complexidades $O(n)$ e $O(nlogn)$ respectivamente.

O algoritmo de Thomas é uma técnica direta e fácil de implementar, enquanto o Algoritmo Parallel Cyclic Reduction é uma técnica iterativa que pode ser paralelizada. Ambos os algoritmos têm aplicações em vários campos, incluindo a análise numérica, a física computacional e a engenharia.

É importante lembrar que a escolha do algoritmo dependerá das especificidades do problema e das necessidades do usuário. Além disso, a implementação correta e otimizada é fundamental para obter resultados precisos e eficientes.

Em resumo, este capítulo apresentou as soluções para sistemas tridiagonais, mostrando as vantagens e desvantagens dos algoritmos de Thomas e Parallel Cyclic Reduction. A compreensão destes algoritmos é fundamental para a resolução de problemas reais e aplicações práticas.

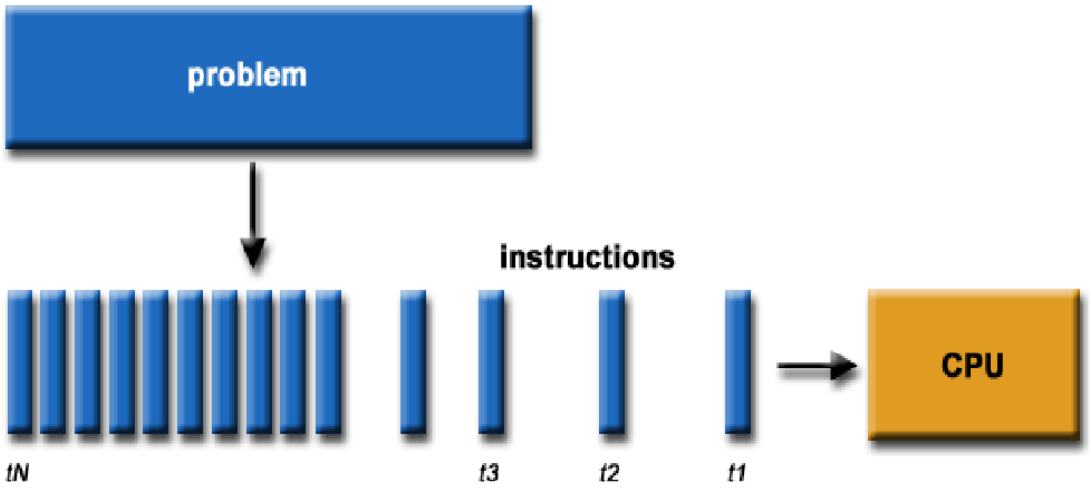
4 COMPUTAÇÃO PARELELA COM OPENCL

4.1 Introdução

Antes de atacar a Computação Paralela, primeiro, é interessante observar o histórico de cálculos de *software* de computador e por que ele falhou na era moderna.

O *software* de computador foi escrito convencionalmente para computação serial. Isso significava que, para resolver um problema, um algoritmo divide o problema em instruções menores. E, essas instruções discretas são então executadas na Unidade Central de Processamento de um computador, uma a uma. Somente depois que uma instrução é concluída, a próxima é iniciada, como indica a Figura 9.

Figura 9 – Arquitetura de uma computação serial



Fonte: (RATHOD; KHADSE; BAGWAN, 2014)

Em resumo, uma declaração de problema é dividida em instruções discretas. Em seguida, as instruções são executadas uma a uma. Apenas uma instrução é executada a qualquer momento, o que causava um grande problema na indústria de computação. Isso representa um grande desperdício de recursos de *hardware*, pois apenas uma parte do *hardware* estará em execução para instruções específicas.

4.2 Simultaneidade

Concorrência e programação paralela são conceitos relacionados, mas possuem algumas diferenças importantes.

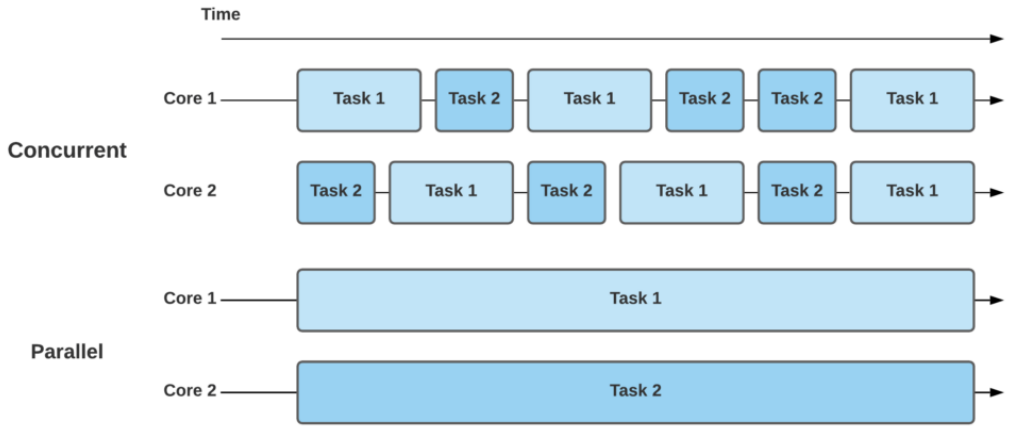
A concorrência é o design e a implementação de programas de computador que podem ser executados simultaneamente em vários processadores ou núcleos. O objetivo da

programação concorrente é fazer o uso mais eficiente dos recursos, e melhorar o desempenho e a capacidade de resposta de um programa. Isso pode ser feito dividindo o programa em tarefas menores e independentes, executando essas tarefas simultaneamente em processadores ou núcleos separados.

A programação paralela, por outro lado, refere-se ao uso de vários processadores ou núcleos para executar uma única tarefa mais rapidamente. O objetivo da programação paralela é acelerar a execução de um programa, dividindo-o em partes menores que podem ser executadas simultaneamente em vários processadores ou núcleos. Isso é obtido usando algoritmos e bibliotecas paralelos especializados e projetando o programa especificamente para aproveitar os recursos de computação paralela.

A Figura 10 ilustra bem a diferença entre o modo de execução da computação com concorrência e a paralela, nesse exemplo, podemos observar que existem dois núcleos e duas tarefas. Na abordagem concorrente, cada núcleo está executando ambas as tarefas alternando entre elas ao longo do tempo. Em contraste, a abordagem paralela não alterna entre as tarefas, mas as executa em paralelo ao longo do tempo.

Figura 10 – Computação concorrente vs Paralela



Fonte: (CONCURRENCY. . . ,)

Resumindo, a programação com concorrência se concentra em tornar os programas mais eficientes e responsivos ao executar tarefas simultaneamente, enquanto a programação paralela se concentra em tornar os programas mais rápidos, dividindo-os em partes menores que podem ser executadas simultaneamente em vários processadores ou núcleos.

4.3 Computação Paralela

Agora que entendemos o processo por trás da computação paralela e sua distinção do processo de concorrência, será apresentado o método de implementação de resolução de problemas matemáticos utilizando o paralelismo, ou seja, o método para otimizar o cálculo de equação, no nosso o cálculo da predição de perda de percurso utilizando equações parabólicas e o método das diferenças finitas.

Recapitulando, existem várias formas de implementar a computação paralela para resolver problemas matemáticos, incluindo:

- Divisão de tarefas: O problema é dividido em subproblemas que podem ser resolvidos independentemente, cada um em um processador ou núcleo diferente. A solução final é então combinada a partir das soluções dos subproblemas.
- Tarefas paralelas: Várias tarefas matemáticas são executadas simultaneamente em diferentes processadores ou núcleos, aumentando a velocidade de processamento.
- Comunicação entre processadores: Vários processadores trabalham juntos para resolver um único problema, comunicando-se entre si para compartilhar dados e informações de progresso.
- Programação de memória compartilhada: Vários processadores acessam e modificam a mesma memória, permitindo que eles trabalhem juntos para resolver um problema.
- Programação de memória distribuída: Cada processador tem sua própria memória e se comunica com outros processadores para compartilhar dados e informações de progresso.

Esses métodos podem ser combinados de várias maneiras para resolver problemas matemáticos específicos, e as escolhas dependem do tamanho e complexidade do problema, bem como das limitações de *hardware* e *software*.

Porém antes de paralelizar qualquer equação deve-se atentar em alguns pontos para o correto e otimizado funcionamento do código. É importante considerar as seguintes condições:

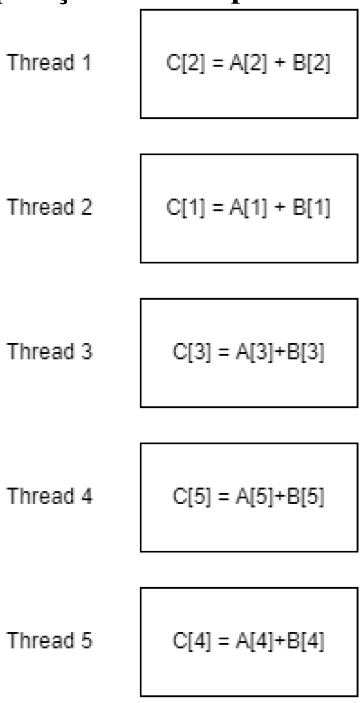
- Paralelismo: O problema deve ser altamente paralelizável, ou seja, deve ser possível dividi-lo em subproblemas que podem ser resolvidos independentemente.
- Escalabilidade: O problema deve ser escalável, ou seja, a velocidade de resolução deve aumentar proporcionalmente à quantidade de processadores ou núcleos disponíveis.
- Comunicação: As tarefas matemáticas devem ser capazes de se comunicar entre si para compartilhar dados e informações de progresso.
- Sincronização: As tarefas matemáticas devem ser sincronizadas para garantir que elas

trabalhem juntas de forma eficiente.

- Balanceamento de carga: É importante garantir que todos os processadores ou núcleos estejam ocupados e trabalhando de forma equilibrada, sem sobrecarregar nenhum deles.
- Eficiência: O algoritmo deve ser eficiente e evitar desperdícios de recursos, como acessos desnecessários à memória.
- *Hardware*: O *hardware* deve ser capaz de suportar a computação paralela, como processadores multinúcleo ou clusters de computadores.
- *Software*: O *software* deve ser capaz de suportar a programação paralela e fornecer ferramentas para dividir, distribuir e sincronizar tarefas matemáticas.

Um exemplo simples de computação paralela é soma de dois vetores $C_n = A_n + B_n$, onde cada *thread* ficaria responsável pela soma de uma posição dos vetores de forma aleatória, para $n = 5$ e supondo que temos 5 *threads*, a Figura 11 mostra a distribuição das operações.

Figura 11 – Exemplo Computação Paralela para soma de vetores com 5 posições



Fonte: Autor.

4.4 OpenCL

OpenCL é uma API de computação de alto desempenho que permite a utilização de diferentes dispositivos de computação, como *Central Processing Unit* (CPU), *Graphics Processing Units* (GPU) e *Digital Signal Processor* (DPS), de forma transparente e unificada. Algumas vantagens de utilizar OpenCL incluem:

- Portabilidade: OpenCL permite escrever códigos que podem ser executados em diferentes plataformas e dispositivos, sem necessidade de modificações significativas.
- Aproveitamento de recursos: OpenCL permite aproveitar todos os recursos disponíveis em uma plataforma, incluindo diferentes tipos de dispositivos de computação. Isso permite otimizar o desempenho do código e aumentar a escalabilidade.
- Flexibilidade: OpenCL permite escrever códigos de forma flexível, permitindo que o programador escolha a melhor forma de utilizar os recursos disponíveis.
- Suporte: OpenCL é uma especificação aberta e amplamente suportada, com implementações disponíveis para diversas plataformas e sistemas operacionais. Isso permite aproveitar as vantagens de OpenCL em uma variedade de aplicações e sistemas.

OpenCL permite a compilação de programas em tempo de execução, o que significa que o código fonte do programa pode ser compilado em um dispositivo de paralelismo no momento em que o programa está sendo executado, em vez de ter que ser compilado previamente. Isso pode ser útil, por exemplo, quando se deseja otimizar um programa para uma determinada arquitetura de dispositivo ou quando se deseja fornecer flexibilidade para usuários finais para compilar seus próprios programas.

4.4.1 Kernel

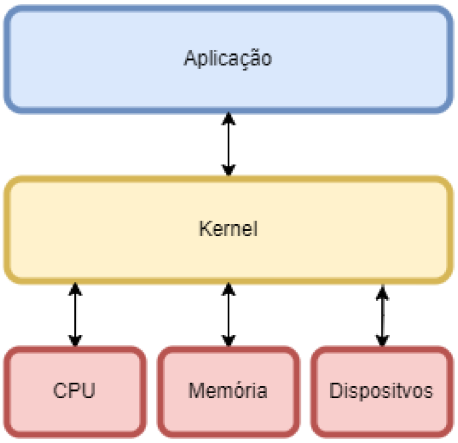
Um *kernel* é o núcleo central de um sistema operacional. Ele gerencia as operações de baixo nível, como acesso à memória e ao *hardware*, e fornece uma interface para os programas de aplicativo. De modo geral, as funcionalidades de um *kernel* podem ser listadas como:

- Gerenciamento de memória: o *kernel* gerencia a alocação de memória para os programas e os dados. Ele também pode fornecer mecanismos de proteção de memória para evitar que os programas acessem áreas de memória não autorizadas.
- Gerenciamento de processos: o *kernel* gerencia a criação, execução e finalização de processos. Ele também pode fornecer mecanismos para garantir que os processos não interrompam uns aos outros e para garantir que os processos tenham acesso equitativo à CPU.
- Gerenciamento de dispositivos: o *kernel* gerencia o acesso a dispositivos de entrada/saída, como discos rígidos, unidades de CD/DVD, impressoras e dispositivos de rede. Ele também pode fornecer mecanismos para garantir que os dispositivos não sejam acessados simultaneamente por vários processos.

- Gerenciamento de arquivos: o *kernel* gerencia o acesso a arquivos e pastas no sistema de arquivos. Ele também pode fornecer mecanismos para garantir que os arquivos sejam acessados de forma segura e para garantir que os arquivos não sejam corrompidos.
- Gerenciamento de rede: o *kernel* gerencia o acesso à rede e pode fornecer mecanismos para garantir que as comunicações de rede sejam seguras e confiáveis.

Em resumo, é a camada intermediaria que conecta os programas de usuário com o *hardware* do computador, como indica a Figura 12.

Figura 12 – Funcionalidade do *kernel*



Fonte: Autor.

Especificamente para o nosso caso, o *kernel* no contexto da OpenCL é uma função que é executada de forma paralela em vários núcleos de dispositivos, como CPUs e GPUs. Ele é usado para descrever as operações de computação que serão executadas em paralelo e é o principal mecanismo para aproveitar a capacidade de computação paralela de dispositivos.

Quando um programa OpenCL é executado, o *kernel* é enviado para os dispositivos específicos onde é dividido em várias tarefas menores, chamadas de *work-items*, que são executadas em paralelo em diferentes núcleos.

Os *kernel*s são utilizados para aproveitar a capacidade de processamento paralelo dos dispositivos, geralmente GPUs, para acelerar aplicações que são intensivas em processamento, como cálculos científicos, processamento de imagens, simulações e outros.

Em resumo, o *kernel* OpenCL é a forma como os desenvolvedores descrevem as operações de computação que serão executadas em paralelo em dispositivos compatíveis, permitindo aproveitar a capacidade de processamento desses dispositivos para acelerar aplicações.

4.4.2 Programando com OpenCL

Antes de apresentar o processo para a implementação de um script com OpenCL, é deve-se atentar a ter instaldo o SDK OpenCL para o seu sistema operacional e plataforma. Isso lhe dará as bibliotecas e cabeçalhos necessários para desenvolver programas OpenCL. Com SDK OpenCL, o modelo de programação com OpenCL é listado abaixo:

- **Preparar e inicializar dados no *host*:** é o processo de preparar os dados que serão processados pelo dispositivo OpenCL antes de serem enviados para o dispositivo. Isso pode incluir alocar memória para esses dados, preencher os dados com valores iniciais e configurar quaisquer parâmetros que serão usados pelo dispositivo.
- **Descobrir e inicializar os dispositivos:** é o processo de detectar e se conectar a dispositivos OpenCL que estão disponíveis no sistema. Isso pode incluir dispositivos de CPU, GPU, FPGA (Arranjo de porta programável em campo) ou outros dispositivos compatíveis com OpenCL.
- **Criar um contexto:** é o processo de criar um ambiente onde os recursos OpenCL serão compartilhados entre os dispositivos. Isso inclui a alocação de memória compartilhada e outros recursos, bem como a configuração de quais dispositivos serão usados no contexto.
- **Criar uma fila de comando:** é o processo de criar uma fila de comandos que será usada para enviar comandos para o dispositivo. Isso pode incluir configurar quais dispositivos serão usados na fila e especificar quais comandos serão enviados para o dispositivo.
- **Criar o objeto programa no contexto:** é o processo de criar um objeto programa OpenCL que contém o código que será executado pelo dispositivo. Isso pode incluir carregar o código de um arquivo ou especificando o código como uma cadeia de caracteres.
- **Construir o programa OpenCL:** é o processo de transformar o código do programa em um formato que possa ser executado pelo dispositivo. Isso pode incluir compilar o código, otimizá-lo e gerar código de máquina.
- **Criar os *buffers* no dispositivo:** é o processo de criar áreas de memória no dispositivo para armazenar dados. Isso pode incluir configurar o tamanho dos *buffers*, definir se eles serão usados para leitura ou escrita e configurar quais dados serão armazenados nos *buffers*.
- **Escrever dados nos *buffers* do dispositivo:** é o processo de enviar dados para os *buffers* no dispositivo. Isso pode incluir copiar dados de memória do *host* para os *buffers* no dispositivo, usando funções como clEnqueueWriteBuffer.

- **Criar e compilar o *kernel*:** é o processo de criar uma função chamada *kernel* que será executada pelo dispositivo. Isso pode incluir escrever o código do *kernel* em OpenCL C e compilá-lo usando funções como clCreate*kernel* e *clBuildProgram*.
- **configurar os argumentos do *kernel***: é o processo de especificar quais *buffers* e variáveis o *kernel* terá acesso e como esses dados serão usados durante a execução. Isso pode ser feito usando funções como clSet*kernel*Arg.
- **Definir o modelo de execução e enfileirar o *kernel* para execução:** é o processo de especificar como o *kernel* será executado no dispositivo e colocá-lo na fila de comandos para execução. Isso pode incluir definir o número de trabalhos e grupos de trabalhos, e enfileirando o *kernel* para execução usando funções como clEnqueueNDRange*kernel*.
- **Recuperar o resultado da memória do dispositivo para o *host*:** é o processo de copiar os dados do dispositivo de volta para a memória do *host* após a execução do *kernel*. Isso pode ser feito usando funções como *clEnqueueReadBuffer*. Após essa etapa, os dados processados podem ser usados pelo aplicativo *host*.

Vale ressaltar que nem todos os itens listados acima são obrigatórios, alguns podem variar de acordo com a aplicação.

5 METODOLOGIA

5.1 Etapas do Desenvolvimento do Trabalho

Inicialmente, as equações modeladas foram implementadas usando a linguagem MATLAB com base na tese de (VASCONCELOS, 2017), e esse código foi usado como um modelo para testar a validade e desempenho das equações. O *script* é descrito como "protótipo de referência", o que significa que é uma versão inicial do código e que ele foi criado com o objetivo de ser usado como uma referência para futuros desenvolvimentos e melhorias. Esse *script* é o primeiro passo para o desenvolvimento de um projeto que envolve modelagem matemática e programação.

Em seguida, o *script* de referência, que antes havia sido escrito na linguagem MATLAB, foi reescrito em C++. A migração do *script* de uma linguagem para outra foi realizada pela facilidade de integração com outros sistemas, associada com o aumento de desempenho, viso que o C++ é uma linguagem de programação de alto desempenho, o que significa que o *script* migrado terá um desempenho melhor em comparação com o *script* escrito em MATLAB. Ademais, C++ é uma linguagem de programação amplamente utilizada, então essa migração tornou o *script* mais fácil de ser integrado com outros sistemas e programas, como a OpenCL.

Por fim, a computação paralela foi adicionada ao *script* migrado para C++, utilizando a API OpenCL. A computação paralela é uma técnica que permite que várias tarefas sejam executadas simultaneamente, aproveitando múltiplos núcleos de processadores ou dispositivos de computação distribuídos, para aumentar a velocidade de processamento. A adição da computação paralela no *script* permite que ele processe dados mais rapidamente, aumentando sua eficiência.

Para auxiliar a integração da OpenCl com C++ foram implementados algoritmos para auxiliar a solução de métodos de equações parabólicas. Esses algoritmos foram projetados para melhorar a velocidade e precisão na solução deste tipo de equação. Um desses algoritmos mencionado é o PCR (Parallel cyclic reduction), que é utilizado para resolver sistemas de equações lineares tridiagonais como apresentado no capítulo 3.

5.2 Funcionamento Geral do Programa

O propósito desta seção é fornecer uma descrição detalhada do funcionamento geral do programa criado. De modo geral, as tarefas fundamentais realizadas pelo programa implantado são:

- Os dados que são fornecidos ao programa para processamento: Nesta etapa, foi utilizando um arquivo no formato JSON (JSON,) contendo todos parâmetros necessários para o cálculo da propagação no percurso, o arquivo JSON é encontrado no Anexo A com a indicação de todos os parâmetros de entrada.
- Conversões de Unidades do parâmetros iniciais: Nesta etapa, todos os parâmetros de entrada são convertidos para o Sistema Internacional de Unidades (SI).
- Criação do Perfil Terreno considerando a curvatura da terra: Caso haja um terreno como parâmetro de entrada, faz-se necessário a interpolação linear deste terreno, visto que, não há uma taxa de amostragem necessária para a implementação, para o MATLAB utiliza-se a função "interp1", para as demais linguagens foi necessário a implementação desta função.
- Determinação da Janela Absorvedora: técnica utilizada para reduzir ou eliminar as reflexões indesejadas geradas por uma condição de contorno em um sistema. Isso é feito colocando uma camada absorvente na superfície de contorno, que é projetada para absorver as ondas incidentes com eficiência.
- Cálculo da impedância do solo: com os parâmetros de entrada encontra-se os valores de α_{BC}
- Criação do Campo inicial: Uma vez modelado como a fonte injeta energia eletromagnética que propagará no domínio, atribui-se valor ao campo inicial, neste programa há opção de três fontes: Fonte Gaussiana, Fonte Retangular e Fonte Omnidirecional.
- Inicialização das matrizes: Inicializa-se as diagonais inferiores, superiores e principais das matrizes **A** e **B**, e para fins de otimização, a cada iteração, apenas os coeficientes das matrizes que são modificados pelas condições de contorno são alterados, os demais termos são mantidos.
- Realização do processo de *Phase Shift* mostrado na seção 2.6.1.
- Solução do sistema tridiagonal utilizando, para implementação paralela, o algoritmo PCR mostrado na seção 3.1.2.
- Realização do processo de *Phase Shift* Reverso mostrado na seção 2.6.1.
- Salvar o módulo do campo $u(x,z)$: O módulo do campo é necessário para obter os valores da perda de percurso e fator de propagação apresentados na seção 2.5, que são plotados a partir do *software* MATLAB.

5.3 Funções Auxiliares

5.3.1 Interpolação Linear

A interpolação linear é um método utilizado para calcular valores intermediários entre dois pontos conhecidos. No contexto de reamostragem de sinais, a interpolação linear é usada para aumentar ou diminuir a taxa de amostragem de um sinal sem alterar a sua frequência fundamental.

A reamostragem é necessária quando a taxa de amostragem de um sinal é diferente da taxa de amostragem desejada para processamento ou reprodução, como no caso do perfil do terreno. Para mudar a taxa de amostragem de um sinal sem alterar a sua frequência fundamental, é necessário adicionar ou remover amostras do sinal. A interpolação linear é usada para calcular os valores das amostras adicionais ou removidas.

A interpolação linear é baseada na equação de uma reta. Dado dois pontos $(x1, y1)$ e $(x2, y2)$, a equação da reta que passa por esses pontos é dada pela Equaçã (5.1):

$$y = y1 + (y2 - y1) * \frac{(x - x1)}{(x2 - x1)} \tag{5.1}$$

Para reamostrar um sinal, é necessário calcular os valores das amostras adicionais ou removidas. Isso é feito tomando-se um conjunto de amostras consecutivas do sinal original e calculando-se os valores intermediários usando a equação da reta.

Por exemplo, para aumentar a taxa de amostragem de um sinal de 10 amostras por segundo para 20 amostras por segundo, é necessário adicionar amostras intermediárias. Isso é feito tomando-se dois amostras consecutivas do sinal original, $(x1, y1)$ e $(x2, y2)$, e calculando-se o valor da amostra intermediária $(x1 + 0.5, y)$ usando a equação da reta.

A interpolação linear é uma técnica simples e eficaz para reamostrar sinais, mas pode ser insuficiente para sinais com variações abruptas ou que contenham componentes de alta frequência. Nesses casos, pode ser necessário usar técnicas de interpolação mais avançadas, como a interpolação polinomial ou a interpolação cúbica.

5.3.2 Derivada por Diferença Central

A derivada por diferença central é uma técnica utilizada para aproximar a derivada de uma função em um ponto específico. A ideia básica é calcular a taxa de variação da função

em um ponto, dividindo a diferença entre o valor da função em um ponto e o valor da função em outro ponto próximo, pelo tamanho do intervalo entre esses dois pontos.

Matematicamente, a derivada por diferença central pode ser escrita pela Equação (5.2)

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \tag{5.2}$$

em que, $f(x)$ é a função, $f'(x)$ é a derivada da função em x, h é o tamanho do intervalo, e x é o ponto em que a derivada é aproximada.

Quanto menor for o valor de h, mais precisa será a aproximação. No entanto, h não pode ser muito pequeno, pois isso pode causar erros de arredondamento devido à representação finita dos números em computadores.

A derivada por diferença central é uma técnica comum utilizada na computação numérica para aproximar derivadas, especialmente quando a expressão analítica da derivada é desconhecida ou muito complexa para ser calculada diretamente.

5.3.3 Multiplicação de Matrizes Tridiagonais

Uma forma eficiente de multiplicar uma matriz tridiagonal (**A**) *nxn* por uma matriz coluna (**B**) de n linhas é utilizando vetores que representam cada uma das três diagonais não nulas da matriz tridiagonal. Esses vetores são chamados de vetor da diagonal principal (D), vetor da diagonal superior (U) e vetor da diagonal inferior (L).

Para multiplicar a matriz tridiagonal por uma matriz coluna, é preciso percorrer cada linha da matriz tridiagonal e calcular o produto dos elementos não nulos com os elementos correspondentes na matriz coluna.

A Equação (5.3) mostra a equação para esta multiplicação otimizada.

$$X_i = \begin{cases} D_i \cdot B_i + U_i \cdot B_{i+1} & \text{Se } i = 0 \\ L_i \cdot B_{i-1} + D_i \cdot B_i & \text{Se } i = n-1 \\ L_i \cdot B_{i-1} + D_i \cdot B_i + U_i \cdot B_{i+1} & \text{Caso contrário} \end{cases} \tag{5.3}$$

A multiplicação pode ser feita de forma eficiente utilizando esses vetores, pois não é preciso percorrer toda a matriz tridiagonal para calcular os produtos. Em vez disso, é preciso

somente percorrer os vetores D, L e U e calcular os produtos com os elementos correspondentes na matriz coluna.

Esse método é eficiente porque ele é capaz de aproveitar a estrutura específica da matriz tridiagonal e evitar multiplicações desnecessárias. Além disso, ele permite acesso direto aos elementos da diagonal principal, superior e inferior, tornando a operação mais rápida.

6 RESULTADOS E DISCUSSÕES

6.1 Introdução

Com a fundamentação teórica e a metodologia estabelecidas, prossegue-se com a implementação deste trabalho. Neste capítulo, será apresentado os resultados obtidos a partir da aplicação do método proposto, como parte do fechamento deste estudo. Além de apresentar os resultados, este capítulo também visa levantar discussões sobre a validade dos mesmos e apresentar possíveis maneiras de melhorá-los.

6.2 Validação do Método

Para validar o método proposto, foram realizados testes de comparação entre o *script* de referência em MATLAB e a implementação com a utilização da computação paralela com OpenCL, os testes foram realizados para diferentes parâmetros de entrada, como frequência, polarização da fonte, tipo de fonte, variação dos índices de refração, distâncias de propagação, tipos de solo, entre outros. Nestes testes, foram avaliados o tempo de execução e o erro máximo absoluto do resultado obtido entre o *script* em Matlab e o *script* paralelizado com OpenCL.

Para os testes foi utilizado um notebook com as seguintes configurações:

- GPU: NVIDIA GeForce GTX 1650 (4GB) Mobile
- CPU: Intel Core i5-10500H
- RAM: 8GB

6.2.1 Cálculo para Terreno sem Relevo

Os parâmetros fixos utilizados para o primeiro teste com a terra sem relevo é mostrado na Tabela (1), neste teste variou-se a frequência e o método de salvamento.

Tabela 1 – Parâmetros Usados para o teste 1

Parâmetros	Valores
Altitude máx	500 m
Altura Fonte	125
θ_{BW}	1°
θ_{tilt}	0°

Fonte**:** Elaborado pelo autor

Para o teste com os parâmetros indicados na Tabela (1), obteve-se os seguintes

resultados mostrado na Tabela (2).

Tabela 2 – Parâmetros Usados para o teste com terra lisa

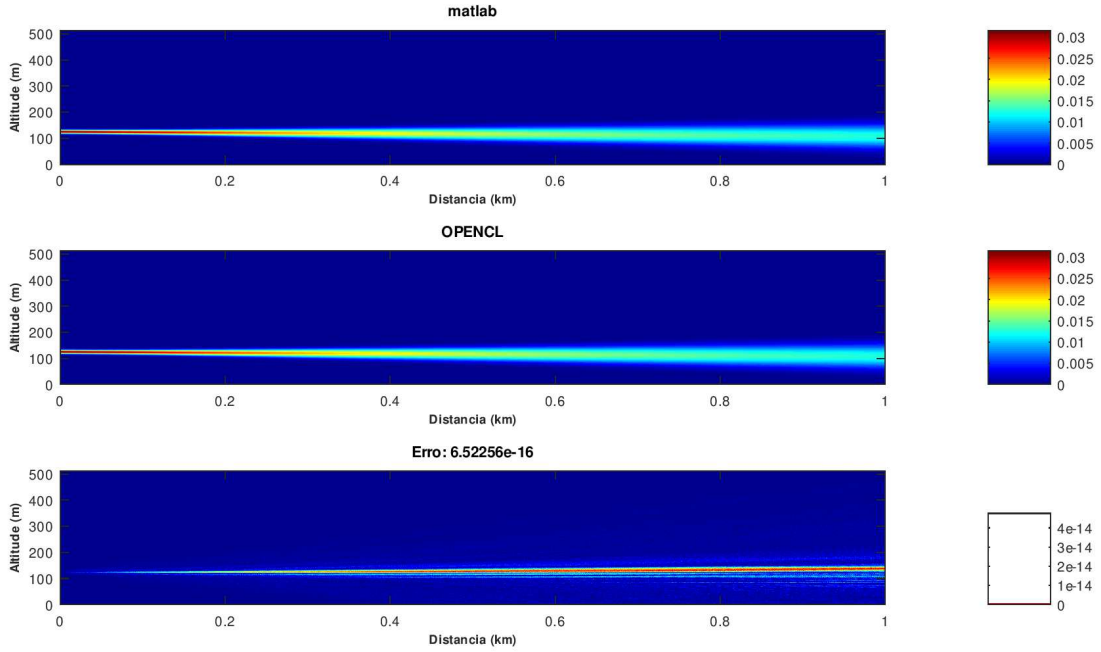
Frequência (MHz)	Distância	Polarização	Salvamento	Tempo Matlab (s)	Tempo OpenCL (s)	Eficiencia OpenCL x Matlab	Erro Máximo
300	1000	H	Completo	5.9	1.14	5.2	10^{-16}
300	1000	H	Discreto	6.3	1.16	5.4	10^{-16}
300	1000	V	Discreto	5.7	1.13	5.1	10^{-16}
300	1000	V	Discreto	5.8	1.12	5.2	10^{-16}
700	65000	H	Discreto	708.1	162.49	4.3	10^{-16}
700	65000	V	Discreto	709.1	162.49	4.3	10^{-16}

Fonte**:** Elaborado pelo autor

A análise da Tabela (2) revela que o uso de OpenCL para paralelizar o programa resultou em uma melhoria de, em média, 500% em comparação ao MATLAB. É importante mencionar que as configurações da máquina utilizada para os cálculos é um fator crucial na avaliação dos resultados e pode afetar diretamente no resultado obtido. Além disso, o erro máximo absoluto é da ordem de 10^{-16}, o que é esperado para operações de ponto flutuante. A partir da Tabela (2), também se observa que, para uma terra sem relevo, a relação entre o tempo de processamento em MATLAB e OpenCL tende a permanecer constante, independentemente do tamanho do problema. Isso se deve à necessidade de copiar a matriz A a cada iteração no cálculo da propagação em OpenCL devido ao algoritmo PCR, enquanto essa operação não é necessária no MATLAB, mantendo a diferença de tempo entre os dois programas consistente para diferentes parâmetros.

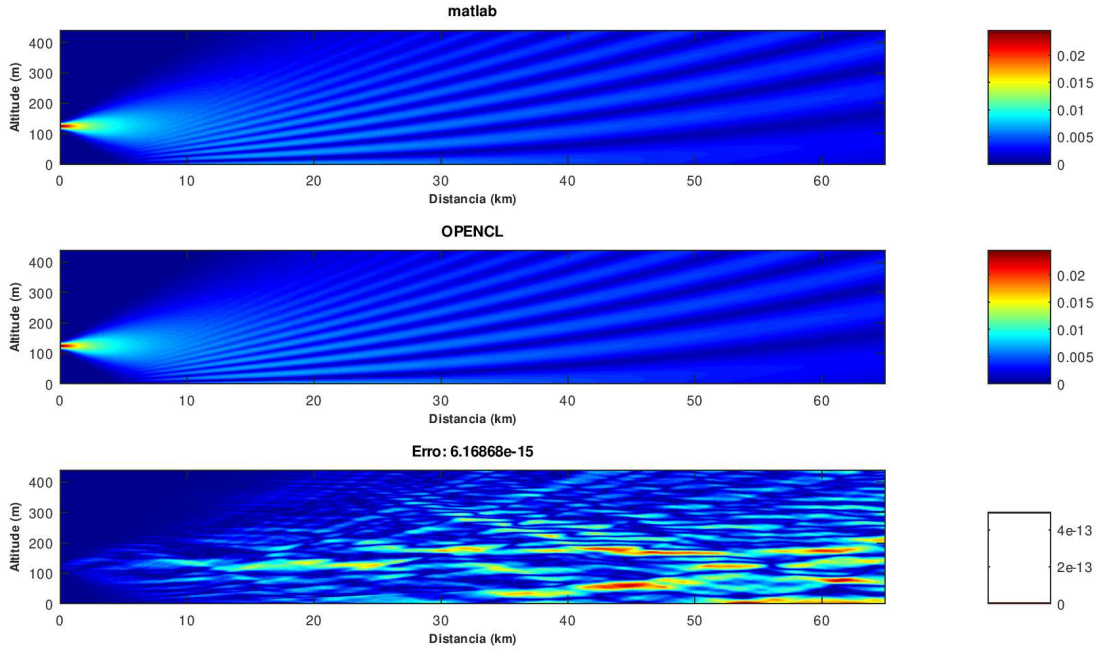
As Figuras (13) e (14) apresentam os resultados obtidos para os caso das simulações com frequências de $300MHz$ $700MHz$ e distâncias de $1000m$ e $65km$, respectivamente, tanto pela utilização da OpenCL quanto pelo processamento sequencial no Matlab, como é possível observar ambos os métodos apresentam o mesmo resultado, com a diferença sendo o erro de ponto flutuante.

Figura 13 – Comparativo OpenCL vs MATLAB, 300MHz-1Km-Pol. V



Fonte: Autor.

Figura 14 – Comparativo OpenCL vs MATLAB, 700MHz-65Km-Pol. V



Fonte: Autor.

6.2.2 Cálculo para Terreno com Relevo

Os parâmetros fixos utilizados para o teste com a terra com relevo é mostrado na Tabela (3), neste teste variou-se as distâncias e o método de salvamento.

Tabela 3 – Parâmetros Usados para o teste 1

Parâmetros	Valores
Altitude máx	500 m
Altura Fonte	125
θ_{BW}	1°
θ_{tilt}	0°
Freq	100MHz

Fonte**:** Elaborado pelo autor

Para o teste com os parâmetros indicados na Tabela (3), obteve-se os seguintes resultados mostrado na Tabela (4).

Tabela 4 – Parâmetros Usados para o teste com terra lisa

Salvamento	Frequência (MHz)	Polarização	Distancia (m)	Tempo Matlab (s)	Tempo OpenCL (s)	Eficiencia OpenCL x Matlab	Erro Máximo
Completo	100	V	1000	11,96166	1,573526	7,601817	1,00E-15
Discreto	100	V	1000	11,51604	1,55978	7,383118	1,00E-15
Completo	100	H	1000	11,53367	1,472168	7,834478	9,00E-16
Discreto	100	H	1000	11,58801	1,454521	7,966888	9,00E-16
Discreto	100	H	65000	3172,35	369,8596	8,577172	1,00E-06
Discreto	100	V	65000	3227,387	394,7242	8,176309	1,00E-06

Fonte**:** Elaborado pelo autor

O aumento do tempo de simulação indicado na Tabela (4) em relação à terra sem relevo, se deve ao fato de que as equações que implementa o perfil do terreno no método faz-se necessário o cálculo das matrizes A e B a cada iteração de m. Ademais, o método de solução para ângulo amplo utilizado (HOLM, 2007), aplica o PHASE-SHIFT e o PHASE-SHIFT-REVERSO no campo, gerando mais cálculos para o processo, para realizar essas operações é necessário utilizar as funções seno e cosseno, que não são precisas quando implementadas na GPU, gerando assim o erro indicado na tabela (4).

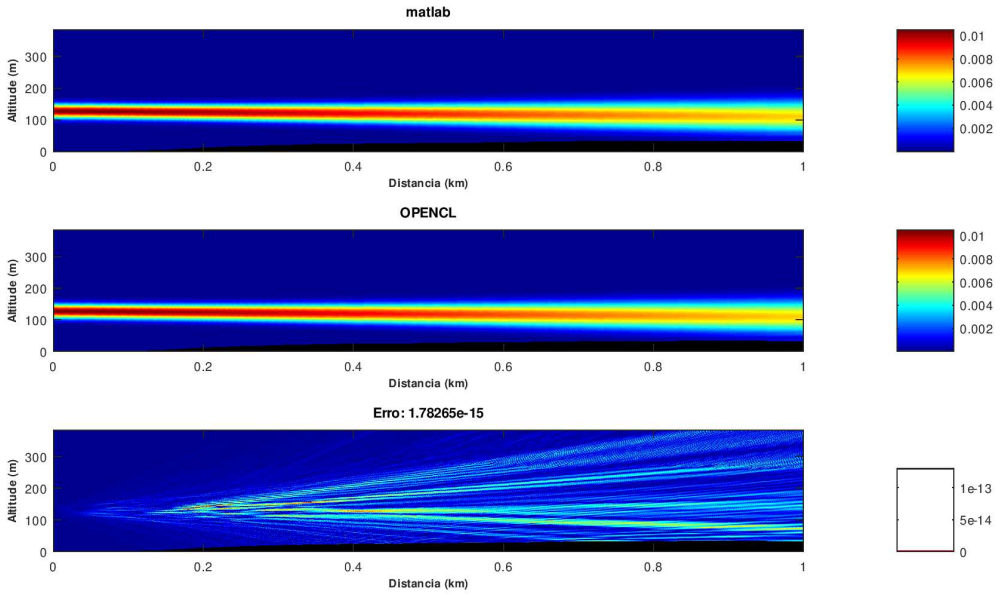
Também, a partir da Tabela (4) nota-se a eficiência no tempo de simulação da OpenCL em relação ao MATLAB que para a terra com relevo foi de aproximadamente 800% melhor, este valor manteve-se constante para diferentes frequências.

As Figuras (15), (16), mostram os resultados das simulações para distâncias de 1000

metros e 65000 metros, respectivamente.

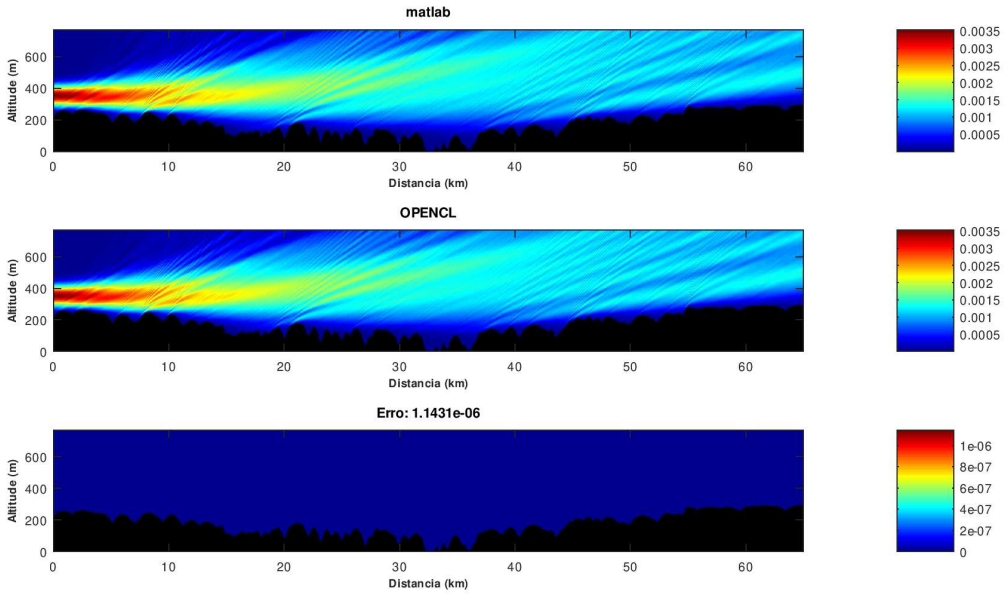
O erro apresentado na Figura (16) é de 10^{-6}, mas esse valor não reflete precisamente a análise realizada com a métrica das cores no gráfico do erro. Isso porque o erro mostrado representa apenas o erro máximo que pode ter ocorrido em um ponto imperceptível do terreno.

Figura 15 – Comparativo OpenCL vs MATLAB, 1Km - Pol. H



Fonte: Autor.

Figura 16 – Comparativo OpenCL vs MATLAB, 65Km - Pol. H - Discreto



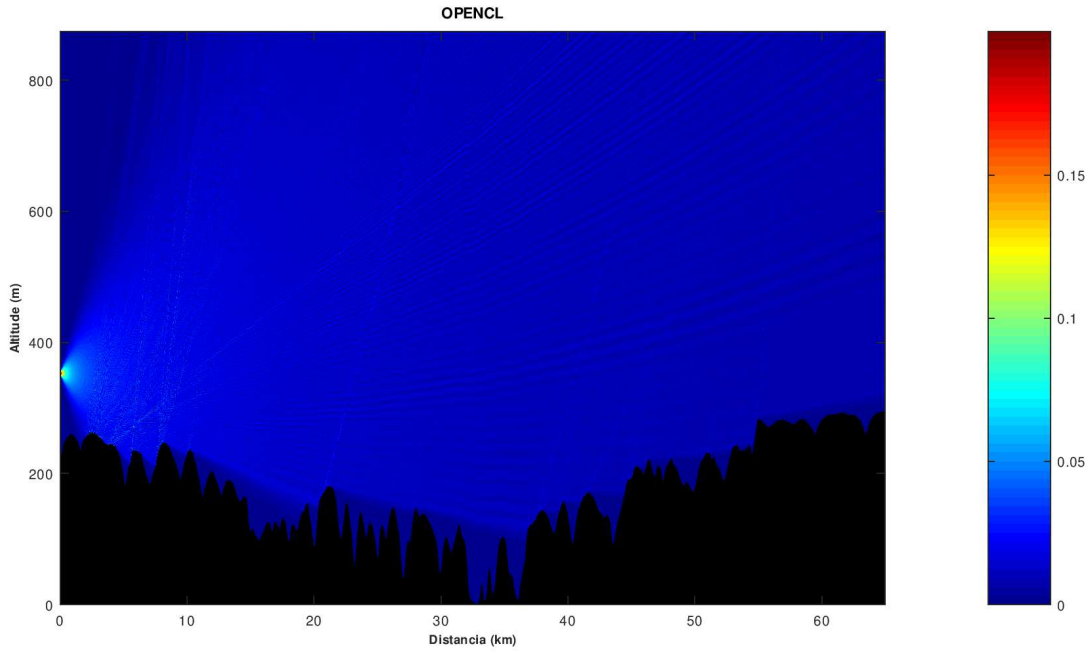
Fonte: Autor.

6.2.3 Simulação de um Enlace Real e sua Perda de Percurso

Validado o método proposto para casos de menor complexidade, realizou-se uma simulação de um enlace de micro-ondas real localizado na cidade de Uberlândia para fins de validação do programa implementado numa situação prática, para isso, foi utilizado os seguintes parâmetros: frequência de 6.125GHz, distância de aproximadamente 65.5Km, passo de 0.02 para x e z, e salvamento discreto com 2500×2500 pontos. Para esses parâmetros obteve-se um tempo de simulação de 62 minutos e 46 segundos, com uso de 100% da placa de video. Está simulação foi realizada somente com a utilização do programa com OpenCL, visto que o tempo de simulação para o MATLAB seria muito grande.

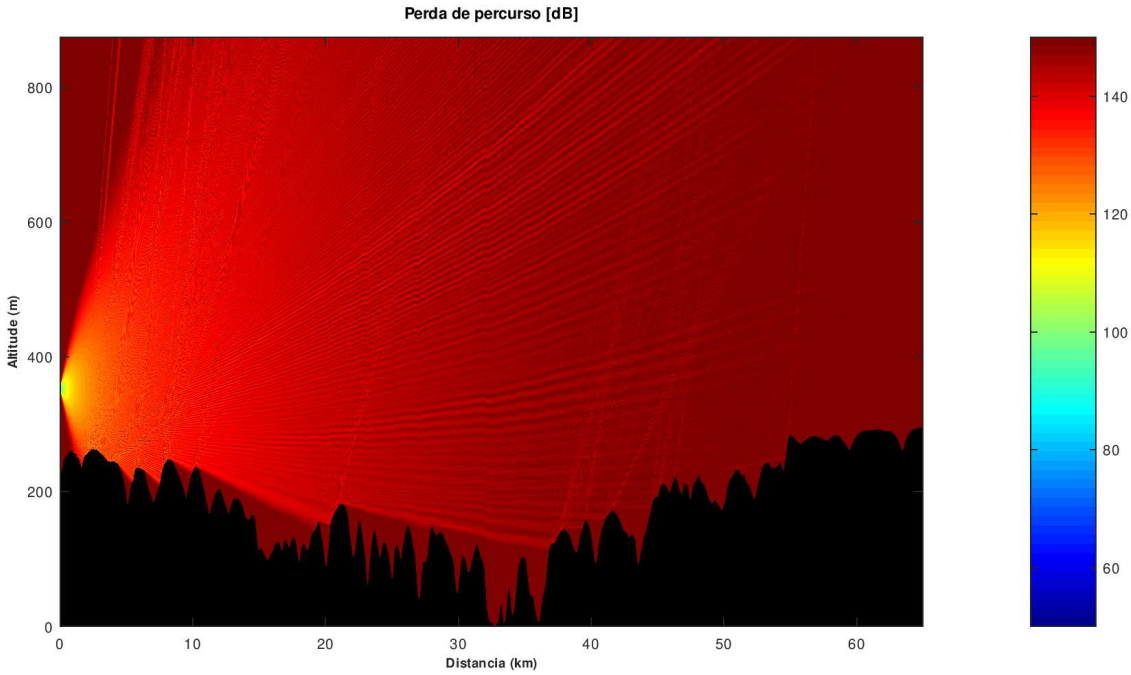
Tendo o módulo do campo $u(x,z)$ é possível calcular as perdas de percurso e o fator de propagação, como mostrado na seção 2.5. As figuras (17), (18) e (19) mostram, respectivamente, o campo $u(x,z)$, as perdas de percurso e o fator de propagação.

Figura 17 – Campo $u(x,z)$ para simulação do Enlace real com as condições especificadas pelo usuário



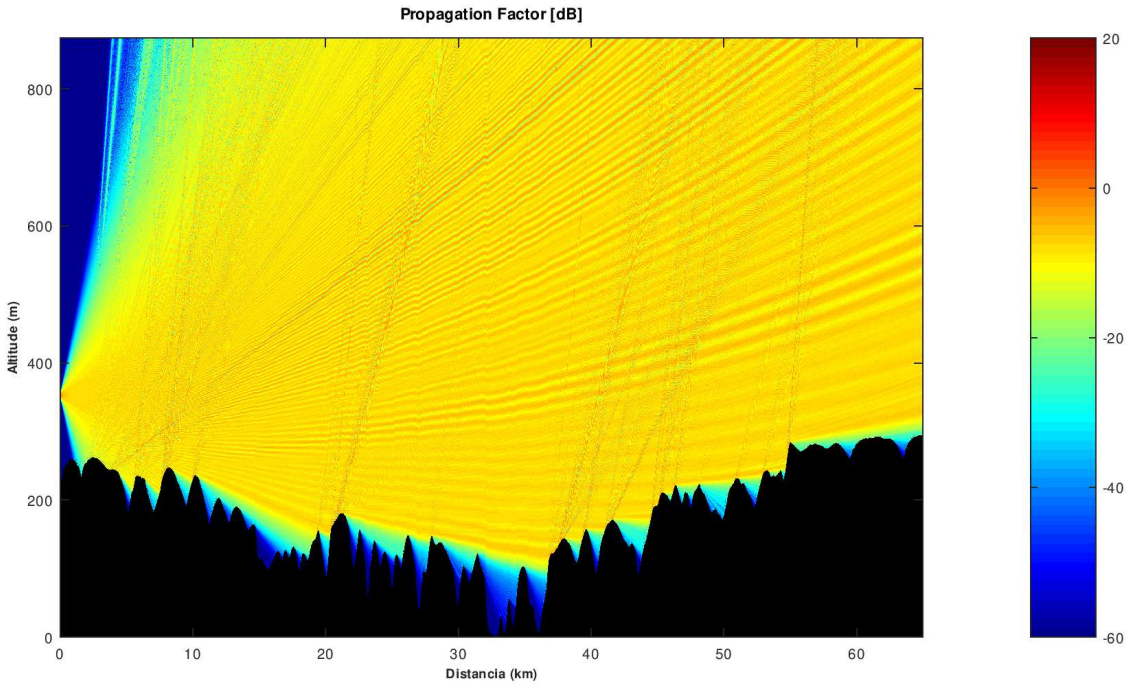
Fonte: Autor.

Figura 18 – Perdas de Percurso L_P



Fonte: Autor.

Figura 19 – Fator de propagação FP

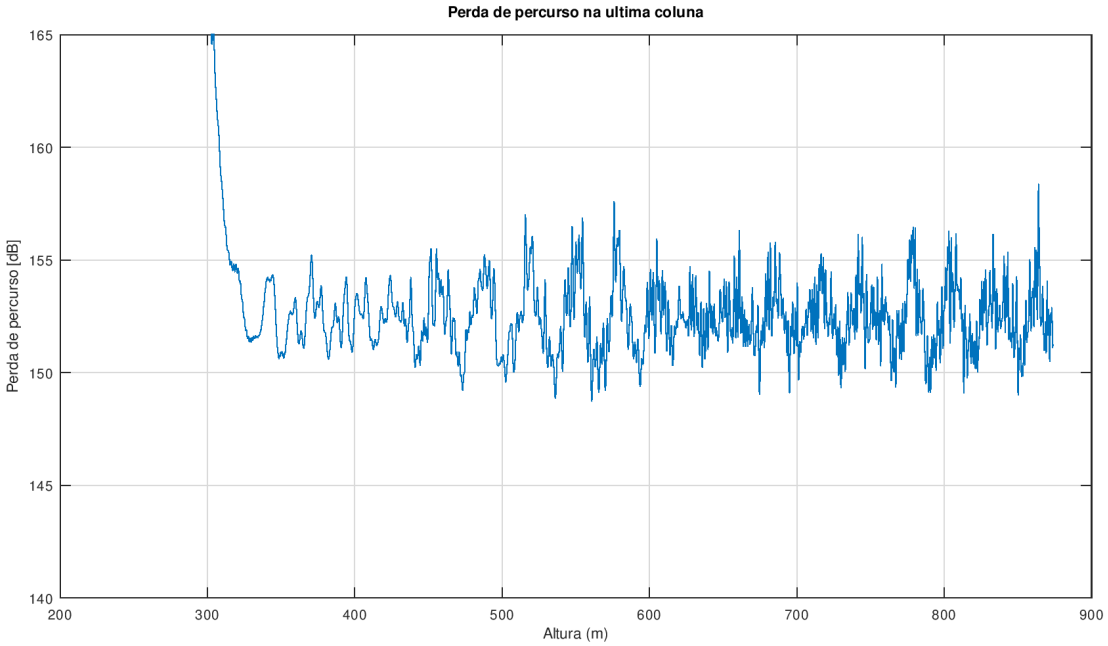


Fonte: Autor.

A Figura (20) apresenta a perda de percurso na última posição, onde está localizada a antena receptora, em função das alturas. Para uma altura de 45 metros acima do solo, é possível

observar que a perda é de aproximadamente 152 dB, que em medições práticas, o valor é de 145dB.

Figura 20 – Perda de Percurso na distância máxima do terreno simulado



Fonte: Autor.

7 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresenta a fundamentação teórica de propagação de ondas e dos métodos utilizados para implementação e validação do método proposto, comparando o desempenho do Matlab com a paralelização em OpenCL. Os resultados demonstram que a paralelização em OpenCL ofereceu uma melhoria significativa na performance em comparação com o Matlab, com ganhos de velocidade até 9 vezes maiores.

Em geral, os resultados sugerem que a paralelização em OpenCL é uma abordagem eficaz para acelerar aplicações que possuem uma elevada complexidade computacional. A pesquisa também mostrou que é possível obter ganhos significativos de velocidade ao comparar o Matlab com a paralelização em OpenCL.

No entanto, é importante lembrar que, apesar dos resultados promissores, ainda é necessário conduzir mais pesquisas para avaliar a escalabilidade e a robustez do método proposto em situações reais. Esta avaliação é crítica para garantir que o método seja aplicável e confiável em diferentes contextos de utilização.

7.1 Trabalhos Futuros

A partir do desenvolvimento do trabalho apresentado, nota-se as seguintes margens para melhoria:

- Otimização do armazenamento das matrizes: Durante o desenvolvimento do projeto, notou-se que a quantidade de memória utilizada para armazenar as matrizes era relativamente alta. Isso pode ser um problema em casos de grandes modelos ou problemas de alta dimensionalidade. Um trabalho futuro seria investigar técnicas de otimização de armazenamento para reduzir o uso de memória sem comprometer a precisão dos resultados.
- Utilização da camada absorvedora: A utilização da camada absorvedora aumentou significativamente o tamanho do problema, o que pode ser um problema em casos de grandes modelos ou problemas de alta dimensionalidade. Uma possível solução seria utilizar a técnica de *Perfectly Matched Layer* (PML) para modelar a camada absorvedora, que é mais eficiente computacionalmente.
- Interface amigável para o usuário final: atualmente, o programa é utilizado para fins de teste e validação do método proposto. Um trabalho futuro seria desenvolver uma interface amigável para o usuário final, permitindo que ele escolha os pontos do terreno e visualize os resultados de forma intuitiva e fácil de entender. Isso pode incluir a construção de

uma *Interface Gráfica do Usuário* (GUI) ou a integração com outras ferramentas de visualização de dados.

- Neste trabalho, as fontes utilizadas para o campo inicial são teóricas e pré-definidas, o que pode limitar a aplicabilidade do projeto na prática. Ao utilizar fontes reais, será possível aprimorar a representação do campo inicial e, consequentemente, a precisão dos resultados obtidos. E, portanto, a implementação de fontes de campo inicial a partir de sinais reais representa uma oportunidade para ampliar o escopo do projeto, torná-lo mais aplicável na prática e aprimorar a precisão dos resultados obtidos.
- Neste trabalho, foi identificada a necessidade de aprimorar o método da PCR. Atualmente, apesar de já ser eficiente, há margem de melhoria para torná-lo ainda mais otimizado, evitando copias desnecessárias.

REFERÊNCIAS

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

BUCK JOHN A.; HAYT JR, W. H. **Eletromagnetismo**. 8ª. ed. [S.l.]: LTC, 2013.

CHENG, D. k. **Field and Wave Electromagnetics**. [S.l.]: Tsinghua University Press, 2007.

CONCURRENCY vs Parallelism | Baeldung on Computer Science. <https://www.baeldung.com/cs/concurrency-vs-parallelism>. (Accessed on 01/15/2023).

DONOHUE, D.; KUTTLER, J. Propagation modeling over terrain using the parabolic wave equation. **IEEE Transactions on Antennas and Propagation**, v. 48, n. 2, p. 260–277, 2000.

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.

JSON. <https://www.json.org/json-pt.html>. (Accessed on 01/23/2023).

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. F. **Parabolic equation methods for electromagnetic wave propagation.** [S.l.]: London: IEE, 2000.

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**, v. 182, n. 12, p. 2638–2654, 2011. ISSN 0010-4655. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0010465511002669>.

RATHOD, A. B.; KHADSE, R.; BAGWAN, M. F. Serial computing vs. parallel computing: A comparative study using matlab. In: . [S.l.: s.n.], 2014.

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. **Tese (Doutorado em Engenharia Elétrica) - Universidade Federal de Uberlândia, Uberlândia**, 2017.

ANEXOS

ANEXO A – Parâmetros de Entrada 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,
```

```
26   "Altura da escala da refractividade": -0.136,
27   "Alturas (em ordem de baixo pra cima) de limite na
       atmosfera": [50, 100],
28   "Inclinacoes do perfil (para caso bilinear ou trilinear)
       ": [-1e-5, 1e-5, 1e-3]
29   },
30
31   "Saida de dados": {
32   "Diretorio de saida": "OUTPUT",
33   "Tipo de salvamento": 0,
34   "__tipo_de_salvamento__": "SALVAR_COMPLETO = 0,
       SALVAR_DISCRETIZADO = 1/*nao implementado*/,
       SALVAR_MEDIA = 2/*nao implementado*/,
       SALVAR_MEDIAMOVEL = 3/*nao implementado*/",
35   "PASSO DE ALTURA Salva [m]": -8000,
36   "PASSO DE DISTANCIA Salva [m]": -8000,
37   "Imagem de Saida": [1000,1000]
38   },
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 }
```