

UNIVERSIDADE FEDERAL DO ABC
CENTRO DE ENGENHARIA, MODELAGEM E CIÊNCIAS SOCIAIS APLICADAS - CECS

Heitor Rodrigues Savegnago

**ANÁLISE DE SISTEMAS DE GEOLOCALIZAÇÃO BASEADOS EM
GPS E AOA PARA FOGUETES DE SONDAGEM ATMOSFÉRICA**

Santo André, SP
2025

Heitor Rodrigues Savegnago

ANÁLISE DE SISTEMAS DE GEOLOCALIZAÇÃO BASEADOS EM GPS E AOA PARA FOGUETES DE SONDAGEM ATMOSFÉRICA

Trabalho de Conclusão de Curso apresentado ao Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas da Universidade Federal do ABC como requisito parcial à obtenção do título de Bacharel em Engenharia de Informação.

Orientador: Prof. Dr. Ivan Roberto de Santana Casella.

Santo André, SP
2025

Agradecimentos

Agradeço aos meus pais, que sempre me incentivaram e deram suporte pra seguir atrás dos meus sonhos;

Ao meu orientador do presente trabalho, Dr. Ivan Roberto de Santana Casella, por aceitar me orientar e acompanhar pacientemente ao longo do desenvolvimento deste projeto;

Ao meu orientador do Projeto de Graduação em Computação, Dr. Francisco de Assis Zampirolli, que pacientemente seguiu me orientando mesmo eu levando mais tempo que o esperado;

Ao meu irmão, por me ajudar a entender os problemas que tive programando, mesmo que só estivesse lá pra me ouvir falar a respeito.

Aos amigos que tantas vezes pedi opiniões sobre como escrever e descrever tantas das coisas neste trabalho;

À Universidade Federal do ABC, seus docentes, técnicos e terceirizados, sem o qual eu não poderia chegar aqui;

E a todos que contribuíram, direta ou indiretamente, no meu caminho até aqui, meu muito obrigado!

Resumo

Ao ser lançado, um foguete de sondagem atmosférica pode pousar em qualquer lugar dentro do seu raio de alcance, e realizar a busca pode se tornar um grande desafio sem uma estratégia de localização eficaz. Muitos desses veículos contam com localização por GNSS, como o GPS, porém ainda dependem de um sistema de telemetria que garanta a correta transmissão das coordenadas geográficas à equipe de busca. O presente trabalho considera o cenário onde a informação de localização não pôde ser decodificada, porém o sinal de RF ainda é detectável, propondo a aferição de Ângulo de Chegada (*Angle of Arrival*, AoA) para determinar a direção de busca. Foi construída a simulação de um sistema que, baseado na diferença de defasagens em uma malha circular de antenas, é capaz de determinar o AoA do sinal RF incidente. Durante o desenvolvimento, foi almejada a compatibilidade com diferentes *softwares* de resolução numérica, particularmente o GNU Octave e o MATLAB, o que se mostrou um desafio, considerando as limitações no uso de *software* livre. Foram consideradas malhas de antenas com três, cinco e sete antenas. A geometria com menos antenas apresentou uma acurácia geral mais baixa comparada às demais. As simulações contemplaram casos com diferentes níveis de interferência por ruído do tipo AWGN, e consideraram cenários com e sem atenuação no sinal. Em todas as simulações, o valor de R^2 foi acima de 75 % e, em média, acima de 92 %. Esses resultados indicam que o método proposto se mostra eficaz em diferentes contextos.

Palavras-chave: *Angle of Arrival*; Radiofrequência; Foguetes de Sondagem Atmosférica; Resolução Numérica; Telemetria; Localização.

Abstract

Upon launch, an atmospheric sounding rocket can land anywhere within its range, and conducting the search can be a major challenge without an effective location strategy. Many of these vehicles rely on GNSS location, such as GPS, but they still rely on a telemetry system to ensure the correct transmission of geographic coordinates to the search team. This work considers a scenario where the location information could not be decoded, but the RF signal is still detectable, proposing the Angle of Arrival (AoA) measurement to determine the search direction. A simulation was built for a system that, based on the phase shift difference in a circular antenna array, is capable of determining the AoA of the incident RF signal. During development, compatibility with various numerical resolution software programs was sought, particularly GNU Octave and MATLAB, which proved challenging given the limitations of using free software. Antenna arrays with three, five, and seven antennas were considered. The geometry with fewer antennas showed lower overall accuracy compared to the others. The simulations considered cases with different levels of AWGN noise interference and considered scenarios with and without signal attenuation. In all simulations, the R^2 value was above 75 % and, on average, above 92 %. These results indicate that the proposed method is effective in different contexts.

Keywords: *Angle of Arrival; Radio Frequency; Atmospheric Sounding Rockets; Numerical Resolution; Telemetry; Location.*

Lista de Figuras

1	Sequência operações simplificada para foguete de sondagem.....	2
2	Representação geométrica de distância e ângulo em relação ao norte entre coordenadas geográficas \mathbf{A}_g e \mathbf{B}_g	4
3	Cálculo do ângulo de <i>Bearing</i> β_b entre as coordenadas dos Campi Santo André e São Bernardo do Campo da UFABC.....	5
4	Característica de frente de onda a cada λ a partir da antena emissora, destaque para $d_F = 8\lambda$	6
5	Diferentes valores de θ_{AoA} para sinal incidente em par de antenas, sistema com ângulo $\alpha_k = 20^\circ$ em relação à referência.	7
6	Diferentes valores para d	7
7	Diferentes distribuições de antenas.	9
8	Geometria geral do sistema com $N_{ant} = 3$	10
9	Fluxograma de operações da função <code>calc_AoA_geometric</code>	20
10	Exemplo de quadro da animação de saída da função <code>generate_fig</code>	22
11	Simulação para três antenas, caso ideal ($SNR \rightarrow \infty$ dB).	24
12	Simulação para três antenas, caso $SNR = 0$ dB, sem atenuação.....	25
13	Simulação para três antenas, caso $SNR = 0$ dB, com atenuação.	25
14	Simulação para cinco antenas, caso ideal ($SNR \rightarrow \infty$ dB).	26
15	Simulação para cinco antenas, caso $SNR = 0$ dB, sem atenuação.	27
16	Simulação para cinco antenas, caso $SNR = 0$ dB, com atenuação.....	27
17	Simulação para sete antenas, caso ideal ($SNR \rightarrow \infty$ dB).	28
18	Simulação para sete antenas, caso $SNR = 0$ dB, sem atenuação.	29
19	Simulação para sete antenas, caso $SNR = 0$ dB, com atenuação.	29

Lista de Tabelas

- 1 Coeficiente de determinação $R^2(\%)$ para simulações notáveis com três antenas. . 24
- 2 Coeficiente de determinação $R^2(\%)$ para simulações notáveis com cinco antenas. 26
- 3 Coeficiente de determinação $R^2(\%)$ para simulações notáveis com sete antenas. . 28

Códigos

1	Função <code>argument_r</code> , simplificada.....	16
2	Função <code>ref_cos</code> , simplificada.	17
3	Função <code>ref_sin</code> , simplificada.	17
4	Função <code>signal_r</code> , simplificada.	18
5	Função <code>phase_z</code> , simplificada.	18
6	Função <code>dephase_A_to_B</code> , simplificada.	19
7	Função <code>deltas_A_B</code> , simplificada.....	19
8	Função <code>is octave</code> , simplificada.	19
9	Função <code>calc_AoA_geometric</code> , simplificada.	21
10	Arquivo de código <code>bearing.m</code>	36
11	Arquivo de código <code>argument_r.m</code>	37
12	Arquivo de código <code>ref_cos.m</code>	37
13	Arquivo de código <code>ref_sin.m</code>	37
14	Arquivo de código <code>signal_r.m</code>	38
15	Arquivo de código <code>phase_z.m</code>	39
16	Arquivo de código <code>dephase_A_to_B.m</code>	39
17	Arquivo de código <code>deltas_A_B.m</code>	40
18	Arquivo de código <code>is octave.m</code>	40
19	Arquivo de código <code>copy_to_documentation.m</code>	40
20	Arquivo de código <code>calc_AoA_geometric.m</code>	41
21	Arquivo de código <code>generate_fig.m</code>	43
22	Arquivo de código <code>w_xyt.m</code>	47
23	Arquivo de código <code>calc_AoA_music_ULA.m</code>	57
24	Arquivo de código <code>calc_AoA_music_UCA.m</code>	58
25	Arquivo de código <code>music_algorithm.m</code>	59
26	Arquivo de código <code>music_spectrum.m</code>	60
27	Arquivo de código <code>steering_linear.m</code>	61
28	Arquivo de código <code>steering_circular.m</code>	61
29	Arquivo de código <code>settings.m</code>	62
30	Arquivo de código <code>w_xyt_single.m</code>	62
31	Arquivo de código <code>w_xyt_dat.m</code>	63

32 Archivo de código `w_xyt_auto.m`..... 65

33 Archivo de código `w_xyt_full_auto.m`..... 67

Abreviaturas e Siglas

AoA	Ângulo de Chegada (<i>Angle of Arrival</i>)
ATT	Atenuação
AWGN	Ruído Gaussiano Branco Aditivo (<i>Additive White Gaussian Noise</i>)
BLE	<i>Bluetooth</i> de Baixo Consumo de Energia (<i>Bluetooth Low Energy</i>)
FFT	Transformada Rápida de Fourier (<i>Fast Fourier Transform</i>)
GEO	Geométrico
GNSS	Sistema Global de Navegação por Satélite (<i>Global Navigation Satellite System</i>)
GPS	Sistema de Posicionamento Global (<i>Global Positioning System</i>)
IoT	Internet das Coisas (<i>Internet of Things</i>)
LoRa	<i>Longe Range</i>
LoS	Linha de visão (<i>Line of Sight</i>)
MUSIC	Classificação de Múltiplos Sinais (<i>MUltiple SIgnal Classification</i>)
NaN	Número Indefinido ou Inválido (<i>Not a Number</i>)
NLoS	Sem Linha de visão (<i>Non Line of Sight</i>)
RF	Radiofrequência (<i>Radio Frequency</i>)
SDR	Rádio Definido por <i>Software</i> (<i>Software-Defined Radio</i>)
SNR	Relação Sinal-Ruído (<i>Signal-Noise Ratio</i>)
UCA	Arranjo Circular Uniforme (<i>Uniform Circular Array</i>)
ULA	Arranjo Linear Uniforme (<i>Uniform Linear Array</i>)

Símbolos e Operadores

\mathbf{A}_g	Coordenada geográfica A
A_k	k -ésima antena do arranjo
\mathbf{B}_g	Coordenada geográfica B
c	Velocidad da luz no ar
D_{ant}	Maior dimensão da antena emissora para cálculo da distância de Fraunhofer
d	Distância entre antenas
d_{F}	Distância de Fraunhofer
d_{AB}	Distância entre coordenadas geográficas A e B
f	Frequência do sinal w de interesse
I_k	Componente em fase do valor complexo do sinal recebido
i	Unidade imaginária, $\sqrt{-1}$
k	Índice das antenas do arranjo
N_{ant}	Número de antenas do arranjo
Q_k	Componente em quadratura do valor complexo do sinal recebido
R_{Terra}	Raio do planeta
T	Período do sinal w de interesse
w	Sinal de interesse para análise, incidente no arranjo de antenas
x_{A_k}	Componente x de coordenada para a antena A_k
y_{A_k}	Componente y de coordenada para a antena A_k
Z_k	Valor complexo do sinal recebido na antena A_k
α_k	Ângulo formado pelo par de antenas A_k e A_{k+1} em relação à geometria do sistema
$\beta_{\pm k}$	Par de ângulos simétricos calculados a partir do sinal incidente no par de antenas A_k e A_{k+1}
β_b	Ângulo de <i>bearing</i> relativo
$\Delta\Phi_k$	Diferença de fase em par de antenas A_k e A_{k+1}
$\Delta\theta$	Diferença entre os ângulos $\theta_{\mathbf{A}_g}$ e $\theta_{\mathbf{B}_g}$
$\Delta\phi$	Diferença entre os ângulos $\phi_{\mathbf{A}_g}$ e $\phi_{\mathbf{B}_g}$
δ	Intervalo de quantização e de filtro para valores de Θ

Θ	Conjunto de todos os valores $\theta_{\pm k}$ aferidos
$\Theta_{\lfloor \bullet \rfloor}$	Valores de Θ quantizados por δ
$\Theta_{\mathbf{F}}$	Valores de Θ filtrados
$\theta_{\mathbf{AoA}}$	Valor do Ângulo de Chegada AoA
$\theta_{\pm k}$	Par de possíveis ângulos de $\theta_{\mathbf{AoA}}$ referentes ao par de antenas A_k e A_{k+1}
$\theta_{\mathcal{M}_o}$	Moda estatística do conjunto $\Theta_{\lfloor \bullet \rfloor}$
$\theta_{\mathbf{A}_g}$	Longitude da coordenada geográfica A
$\theta_{\mathbf{B}_g}$	Longitude da coordenada geográfica B
λ	Comprimento de onda do sinal w de interesse
ρ	Raio do polígono regular formador do arranjo de antenas
Φ_k	Fase do sinal na antena k
$\phi_{\mathbf{A}_g}$	Latitude da coordenada geográfica A
$\phi_{\mathbf{B}_g}$	Latitude da coordenada geográfica B
ω	Frequência angular do sinal w de interesse
$\widetilde{\mathcal{H}}$	Operação estatística mediana para o conjunto \mathcal{H}
$\mathcal{M}_o(\mathcal{H})$	Operação estatística moda para o conjunto \mathcal{H}
$\mathcal{I}m(h)$	Parte imaginária do valor complexo h
$\mathcal{R}e(h)$	Parte real do valor complexo h
$\lfloor h \rfloor$	Operação arredondar, arredonda o valor de h para o inteiro mais próximo

Sumário

1	INTRODUÇÃO	1
1.1	Motivação	1
1.2	Objetivos	2
1.3	Estrutura do documento	2
2	REVISÃO BIBLIOGRÁFICA	3
2.1	Fundamentação teórica	3
2.1.1	Direcionamento por coordenadas geográficas	3
2.1.2	Estimar AoA utilizando malha de antenas	4
2.2	Trabalhos relacionados	11
3	METODOLOGIA	14
3.1	Simulação	14
3.1.1	Parâmetros envolvidos	14
3.1.2	Funções auxiliares	16
3.1.3	Função de cálculo para AoA	17
3.1.4	Função de geração saída visual	20
3.1.5	Função geral da simulação	20
4	RESULTADOS	23
4.1	Performance da simulação	23
4.1.1	Três antenas	23
4.1.2	Cinco antenas	26
4.1.3	Sete antenas	28
4.2	Problemas encontrados	30
4.2.1	Compatibilidade de código	30
4.2.2	Limitações de <i>software</i> livre	30
5	CONCLUSÃO	31
	REFERÊNCIAS	33
	APÊNDICES	36
A	CÓDIGOS DESENVOLVIDOS PARA SIMULAÇÃO	36
A.1	Simulação de direcionamento GNSS	36
A.2	Simulação de AoA	37
A.2.1	Funções auxiliares	37

A.2.2	Função de cálculo para AoA.....	41
A.2.3	Função de geração saída visual	43
A.2.4	Função geral da simulação.....	47
A.2.5	Função do MUSIC	57
A.2.6	Arquivos de simulação em sequência.....	62

1 Introdução

Este capítulo apresenta a motivação para o trabalho proposto, os objetivos gerais e secundários, e uma apresentação da estrutura geral do trabalho.

1.1 Motivação

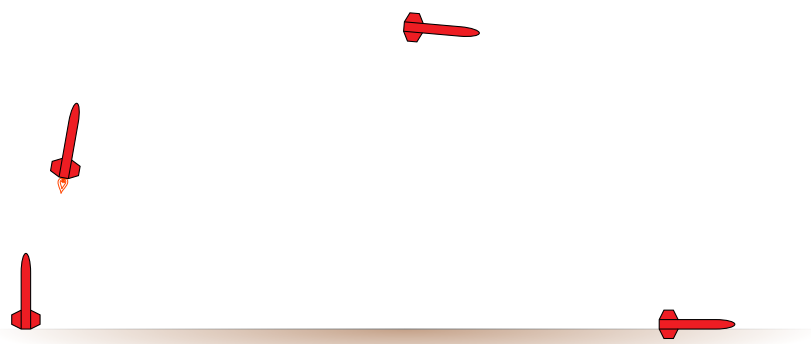
Foguetes de sondagem atmosférica são veículos aeroespaciais sub-orbitais utilizados para levar sensores e experimentos científicos a altos níveis atmosféricos, com o intuito de realizar estudos e análises relacionados às diversas condições ali presentes [1]. Estes veículos geralmente utilizam motor de propelente sólido, de um ou dois estágios, e são equipados com sistemas de controle, telemetria e recuperação, além de transportarem o experimento científico, denominado carga-paga [2, 3]. Algumas das vantagens desses veículos são o baixo custo e a menor necessidade de alcance para sistemas de telemetria e rastreo, tendo em vista que não entram em órbita [4].

No contexto de foguetes de sondagem, existem competições que fomentam o desenvolvimento e a competitividade em equipes universitárias de foguetemodelismo [5]. Algumas dessas competições tem grande parte de suas categorias definidas nas bases de foguetes de sondagem, com apogeu de voo entre 1 km e 10 km de altura acima do nível do solo. Nestes casos, a sequência de operações normal do foguete, simplificada na Figura 1, consiste em: ignição do primeiro estágio do motor, decolagem, período propulsionado, término de queima do primeiro estágio, desacoplamento do primeiro estágio, ignição do segundo estágio, segundo período propulsionado, término de queima do segundo estágio, início do período inercial balístico, apogeu, detecção do apogeu pelos sistemas embarcados, liberação do paraquedas piloto, fase de desaceleração do paraquedas piloto, liberação do paraquedas principal a certa altitude, fase de desaceleração do paraquedas principal e finalmente o pouso [2, 3]. Desacoplamento do primeiro estágio, ignição e fase propulsionada do segundo somente se aplicam a foguetes de dois estágios.

A partir do momento do pouso, o próximo objetivo nessas competições consiste em localizar o foguete, vários métodos podem ser empregados nessa situação, desde cores chamativas no veículo e paraquedas, até sinais sonoros. Essas competições geralmente recomendam, e até exigem, a presença de um GNSS, capaz de transmitir as coordenadas do veículo após o pouso para localização, como um GPS [6].

O processo de localização baseada em dados simples de GNSS, latitude e longitude, pode se tornar mais complicado se o grupo de busca não tem certeza de como encontrar essas coordenadas. Existem dispositivos de GNSS portáteis, porém estes podem criar dificuldades na interface com os dados recebidos da telemetria do foguete. Neste caso, seria possível desenvolver um dispositivo capaz de lidar diretamente com as informações de localização fornecidas pela telemetria e guiar o grupo de busca na direção correta.

Figura 1: Sequência operações simplificada para foguete de sondagem.



Fonte: Autor

Os dados recebidos da telemetria ainda precisam de certo grau de confiabilidade para que sejam devidamente processados e tratados, o que pode ser um problema se o veículo estiver longe do grupo de busca ou o dispositivo de GNSS a bordo não esteja apto a fornecer dados corretamente. Nesse caso, ainda é possível buscar o foguete utilizando o próprio sinal da telemetria, independente dos dados transmitidos.

1.2 Objetivos

O principal objetivo deste trabalho é desenvolver e projetar um dispositivo portátil capaz de indicar a direção da origem de um sinal de RF baseado em métodos de detecção de AoA.

Como objetivo secundário, a análise comparativa com um sistema de utilidade semelhante, porém baseado inteiramente em coordenadas de GNSS.

1.3 Estrutura do documento

O trabalho proposto está organizado em cinco capítulos, apresentando, após este introdutório, mais quatro capítulos. O Capítulo 2 traz um levantamento bibliográfico, contendo revisão de trabalhos relacionados e fundamentação teórica. O Capítulo 3 apresenta o detalhe da metodologia adotada na construção do trabalho. No Capítulo 4 são apresentados detalhes sobre a performance das simulações realizadas e problemas encontrados. Por fim, o Capítulo 5 traz as considerações finais, conclusões e sugestões para trabalhos futuros que podem resultar em melhorias na presente proposta.

O documento também conta com o Apêndice A, que apresenta o conjunto de códigos construídos ao longo do desenvolvimento do trabalho.

2 Revisão Bibliográfica

Este capítulo apresenta um breve levantamento de trabalhos relacionados, que mostram a relevância do assunto abordado, bem como a fundamentação teórica utilizada ao longo do trabalho.

2.1 Fundamentação teórica

A construção deste trabalho fundamentou-se em princípios teóricos, utilizando as bases de direcionamento por coordenadas geográficas, apresentada na Subseção 2.1.1, e princípios de eletromagnetismo para estimar o AoA, apresentados na Subseção 2.1.2.

2.1.1 Direcionamento por coordenadas geográficas

Coordenadas geográficas são definidas por dois valores, latitude e longitude, associadas a coordenadas esféricas referenciadas a partir do centro da terra, assumindo o raio da coordenada como o raio médio da superfície do planeta, cerca de $R_{\text{Terra}} = 6371 \cdot 10^3 \text{ m}$ [7, 8]. A latitude equivale à componente polar ϕ centralizada na linha do equador, enquanto a longitude equivale à componente θ centralizada no meridiano de Greenwich [7, 9].

Conhecendo as coordenadas de dois pontos distintos A e B, é possível determinar seu ângulo de *bearing* β_b relativo, referente ao norte, ou seja, o ângulo da direção a se seguir partindo do ponto A para chegar ao ponto B, a partir da direção norte no ponto de origem A [9].

Sendo \mathbf{A}_g e \mathbf{B}_g duas coordenadas geográficas, $\phi_{\mathbf{A}_g}$ e $\phi_{\mathbf{B}_g}$ suas respectivas latitudes, e $\theta_{\mathbf{A}_g}$ e $\theta_{\mathbf{B}_g}$ suas respectivas longitudes, conforme ilustrado na Figura 2.

Calculam-se Δ_ϕ e Δ_θ conforme Equações 1 e 2, respectivamente.

$$\Delta_\phi = \phi_{\mathbf{B}_g} - \phi_{\mathbf{A}_g} \quad (1)$$

$$\Delta_\theta = \theta_{\mathbf{B}_g} - \theta_{\mathbf{A}_g} \quad (2)$$

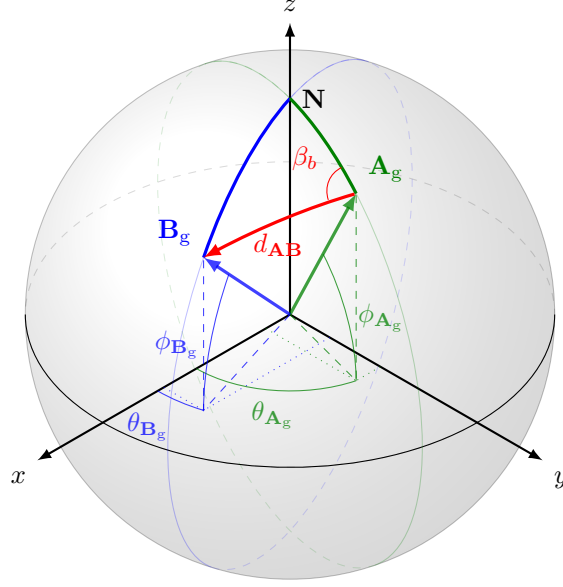
Através da lei dos haversines é possível obter a distância mínima d entre as coordenadas, sobre a superfície, e também o ângulo de *Bearing* β_b formado no vértice \mathbf{A}_g do triângulo esférico $\mathbf{N}\mathbf{A}_g\mathbf{B}_g$ [8]. Para o cálculo de distância, os ângulos devem ser tratados em radianos.

$$X = \cos(\theta_{\mathbf{B}_g}) \cdot \sin(\Delta_\phi) \quad (3)$$

$$Y = \cos(\theta_{\mathbf{A}_g}) \cdot \sin(\theta_{\mathbf{B}_g}) - \sin(\theta_{\mathbf{A}_g}) \cdot \cos(\theta_{\mathbf{B}_g}) \cdot \cos(\Delta_\phi) \quad (4)$$

$$Z = \sin^2\left(\frac{\Delta_\theta}{2}\right) + \cos(\theta_{\mathbf{B}_g}) \cdot \cos(\theta_{\mathbf{A}_g}) \cdot \sin^2\left(\frac{\Delta_\phi}{2}\right) \quad (5)$$

Figura 2: Representação geométrica de distância e ângulo em relação ao norte entre coordenadas geográficas \mathbf{A}_g e \mathbf{B}_g .



Fonte: Autor.

$$\beta_b = \arctan\left(\frac{X}{Y}\right) - \frac{\pi}{2} \quad (6)$$

$$d_{AB} = R_{\text{Terra}} \cdot 2 \cdot \arctan\left(\frac{\sqrt{Z}}{\sqrt{1-Z}}\right) \quad (7)$$

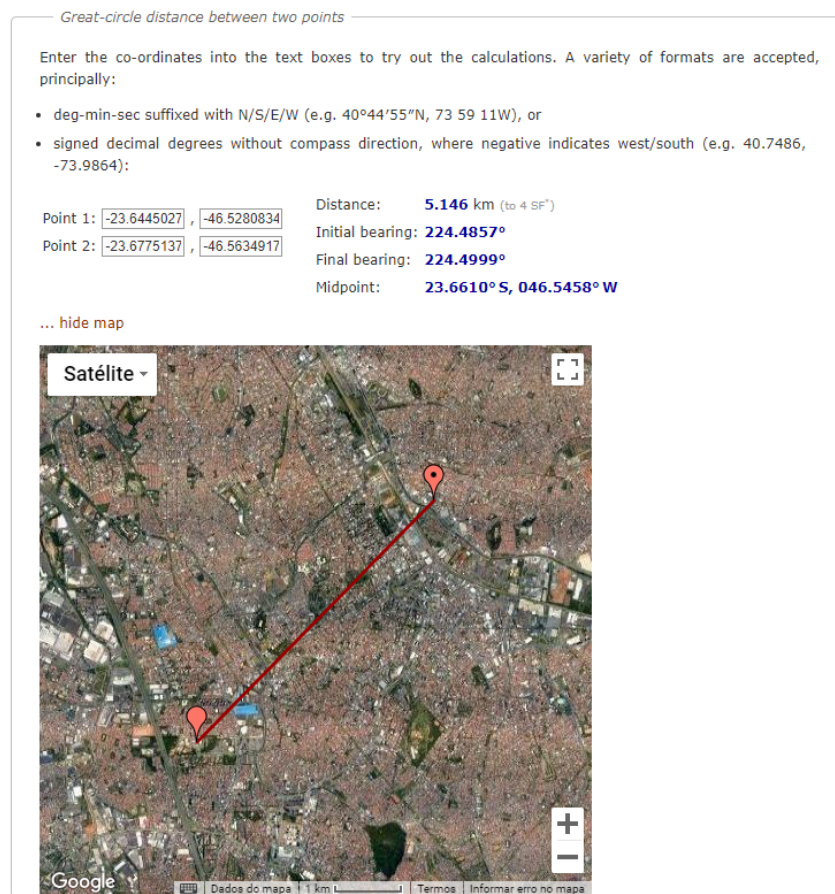
O ângulo β_b calculado aqui é referente à direção cardinal Norte, assim, uma equipe de busca equipada com uma bússola simples seria capaz de seguir a direção correta. A Figura 3 apresenta a aplicação desenvolvida por Veness, capaz de calcular o ângulo de *Bearing* entre duas coordenadas, note que, neste caso, o ângulo referido é relacionado à direção cardinal Leste [8].

2.1.2 Estimar AoA utilizando malha de antenas

Analisando a defasagem de um sinal de RF incidindo em uma malha de antenas, é possível estimar seu Ângulo de Chegada (*Angle of Arrival*, AoA), ou seja, determinar a direção do emissor do sinal em relação ao sistema. Este valor é calculado utilizando dados como a distância entre as antenas, o comprimento de onda λ do sinal e a velocidade da luz no meio, usualmente tomada como $c = 299792458,6 \pm 0,3 \text{ m s}^{-1}$ no ar [10, 11, 12, 13]. A Equação 8 apresenta a relação do comprimento de onda λ com a frequência f , a frequência angular ω e a velocidade da luz c .

$$\lambda = \frac{c}{f} = \frac{2\pi \cdot c}{\omega} \quad (8)$$

Figura 3: Cálculo do ângulo de *Bearing* β_b entre as coordenadas dos Campi Santo André e São Bernardo do Campo da UFABC.

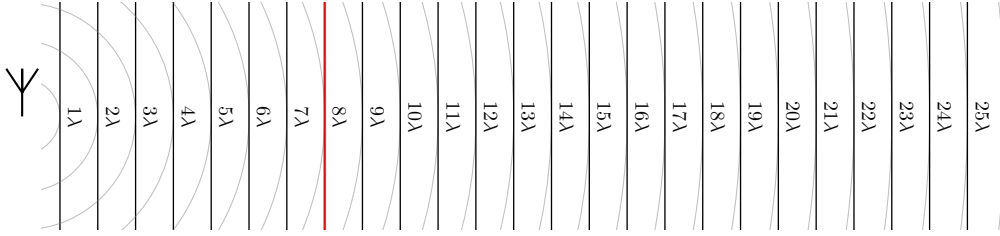


Fonte: Veness 2019 [8]

Se um emissor de sinal estiver suficientemente distante, é possível considerar que a frente de onda tem um comportamento planar, essa característica simplifica as operações envolvidas. A distância de Fraunhofer (d_F) é a mínima para essa condição, ela define o início da região de *far-field*, conforme apresentado na Equação 9, onde D_{ant} é a maior dimensão da antena emissora [14]. Tomando $D_{\text{ant}} = 2\lambda$, para uma antena de dipolo, obtém-se $d_F = 8\lambda$. A Figura 4 ilustra o comportamento planar de uma frente de onda, com destaque na distância d_F .

$$d_F = \frac{2 \cdot D_{\text{ant}}^2}{\lambda} \Rightarrow d_F = \frac{2 \cdot (2 \cdot \lambda)^2}{\lambda} = 8\lambda \quad (9)$$

Figura 4: Característica de frente de onda a cada λ a partir da antena emissora, destaque para $d_F = 8\lambda$.



Fonte: Autor.

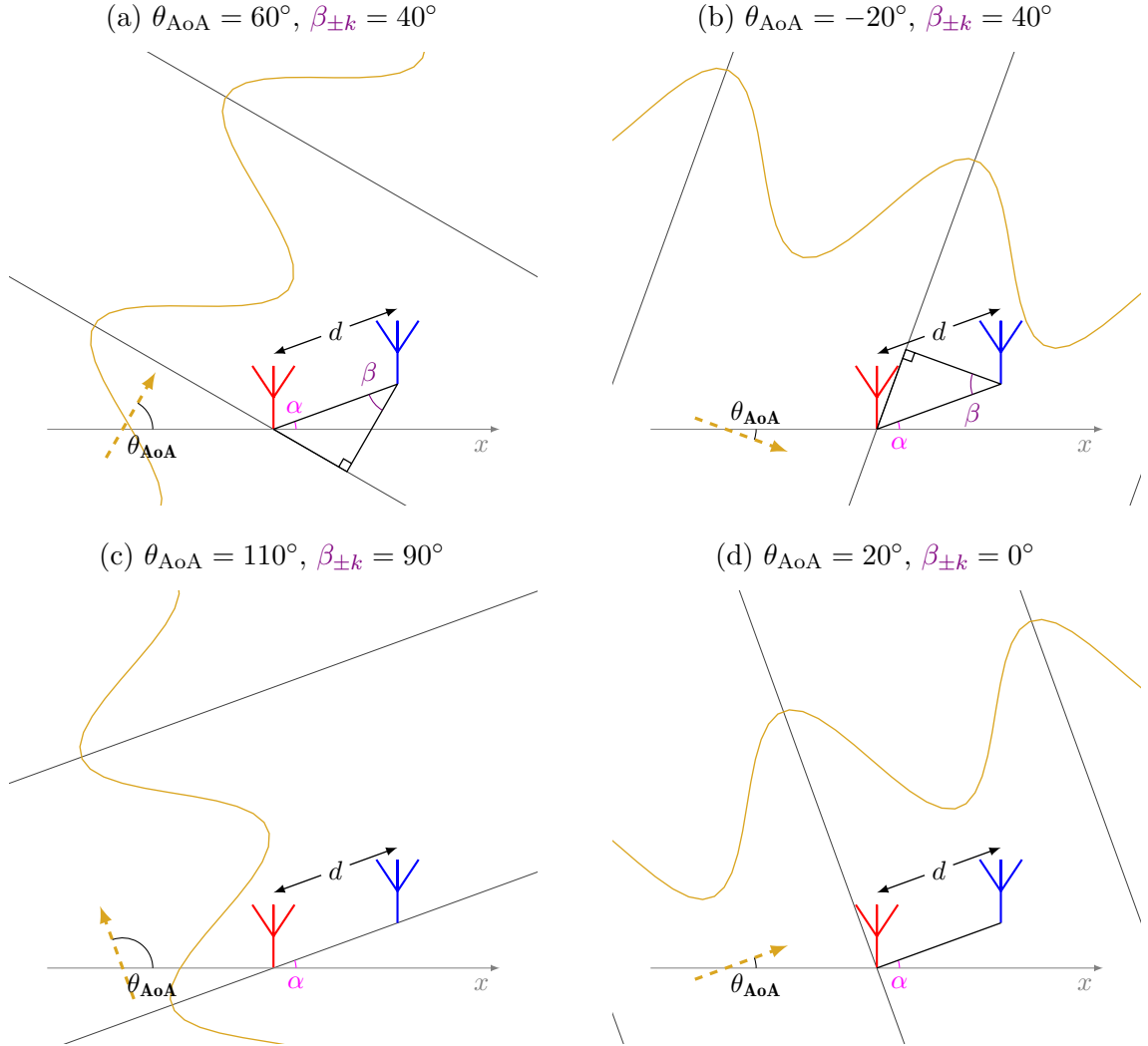
Tomando agora um par de antenas separadas por uma distância fixa d , torna-se viável fazer a análise trigonométrica entre as antenas e a frente de onda incidente, onde essa distância d será a hipotenusa do triângulo retângulo formado. Para realizar esta análise, ainda é necessário conhecer uma segunda dimensão do triângulo retângulo envolvido, esta é obtida da defasagem $\Delta\Phi_k$ dos sinais incidentes nas antenas, conforme apresentado na Equação 10. A Figura 5 apresenta quatro casos de chegada do sinal de RF em um par de antenas.

$$d \cdot \cos(\beta_{\pm k}) = \lambda \cdot \frac{\Delta\Phi_k}{2\pi} \quad (10)$$

É importante ressaltar que um sistema com um único par de antenas não é suficiente para determinar completamente o θ_{AoA} , já que o valor calculado de $\beta_{\pm k}$ é igual para casos simétricos em relação ao par de antenas, criando um caso de ambiguidade. As Figuras 5a e 5b apresentam exemplos de diferentes valores de θ_{AoA} para o mesmo valor de $\beta_{\pm k}$. Existem ainda dois casos notáveis, onde o sinal chega alinhado ao par de antenas ou perpendicular a elas, apresentados respectivamente nas Figuras 5c e 5d.

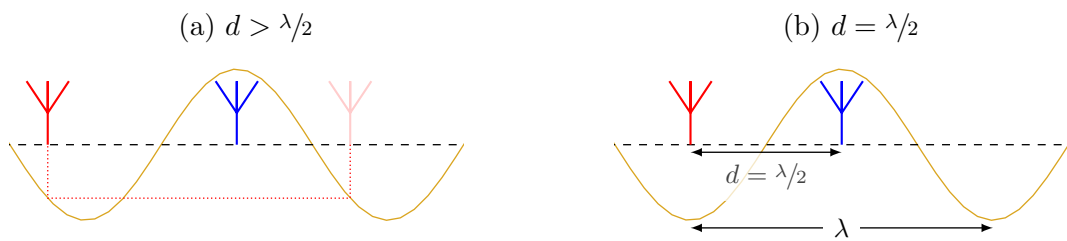
A escolha da distância d entre as antenas deve ser feita de forma a otimizar a resolução da medida de defasagem, com a maior distância possível. Porém é necessário evitar ambiguidades na análise, por se tratar de um sinal periódico, o valor se repetirá a cada λ , e terá valores simétricos quando $d > \lambda/2$, ilustrado na Figura 6a. Adota-se então $d = \lambda/2$, conforme apresentado na Figura 6b e Equação 11 [11, 12, 13].

Figura 5: Diferentes valores de θ_{AoA} para sinal incidente em par de antenas, sistema com ângulo $\alpha_k = 20^\circ$ em relação à referência.



Fonte: Autor.

Figura 6: Diferentes valores para d .



Fonte: Autor.

$$d = \frac{\lambda}{2} \quad (11)$$

Para contornar a ambiguidade de simetria, é possível adicionar mais antenas à malha. O conjunto de N_{ant} antenas deve respeitar a distância d entre as antenas de um par, e pode ser disposto como um polígono regular com N_{ant} lados de tamanho d , onde cada antena está em um vértice. A Figura 7 apresenta exemplos dessa disposição de antenas, note que valores pares de N_{ant} implicam que existirão pares de antenas paralelos, que resultam em leituras redundantes.

A Equação 12 descreve o raio ρ do círculo que circunscreve o polígono regular de N_{ant} antenas. Este raio equivale à distância das antenas em relação ao ponto central do polígono.

$$\rho = \frac{d}{2 \cdot \sin\left(\frac{\pi}{N_{\text{ant}}}\right)} \quad (12)$$

Cada antena é identificada por um índice k , conforme a Equação 13, e tem sua coordenada espacial definida como um valor complexo descrito na Equação 14. Essas coordenadas são definidas como números complexos para simplificar a análise do ângulo α_k em que um par de antenas A_k e A_{k+1} se dispõe em relação à geometria do sistema, conforme Equação 15.

$$k = \{1, 2, \dots, N_{\text{ant}}\} \quad (13)$$

$$A_k = \rho \cdot \exp\left(i \cdot k \cdot \frac{2\pi}{N_{\text{ant}}}\right) = (\mathcal{R}e(A_k), \mathcal{I}m(A_k)) = (x_{A_k}, y_{A_k}) \quad (14)$$

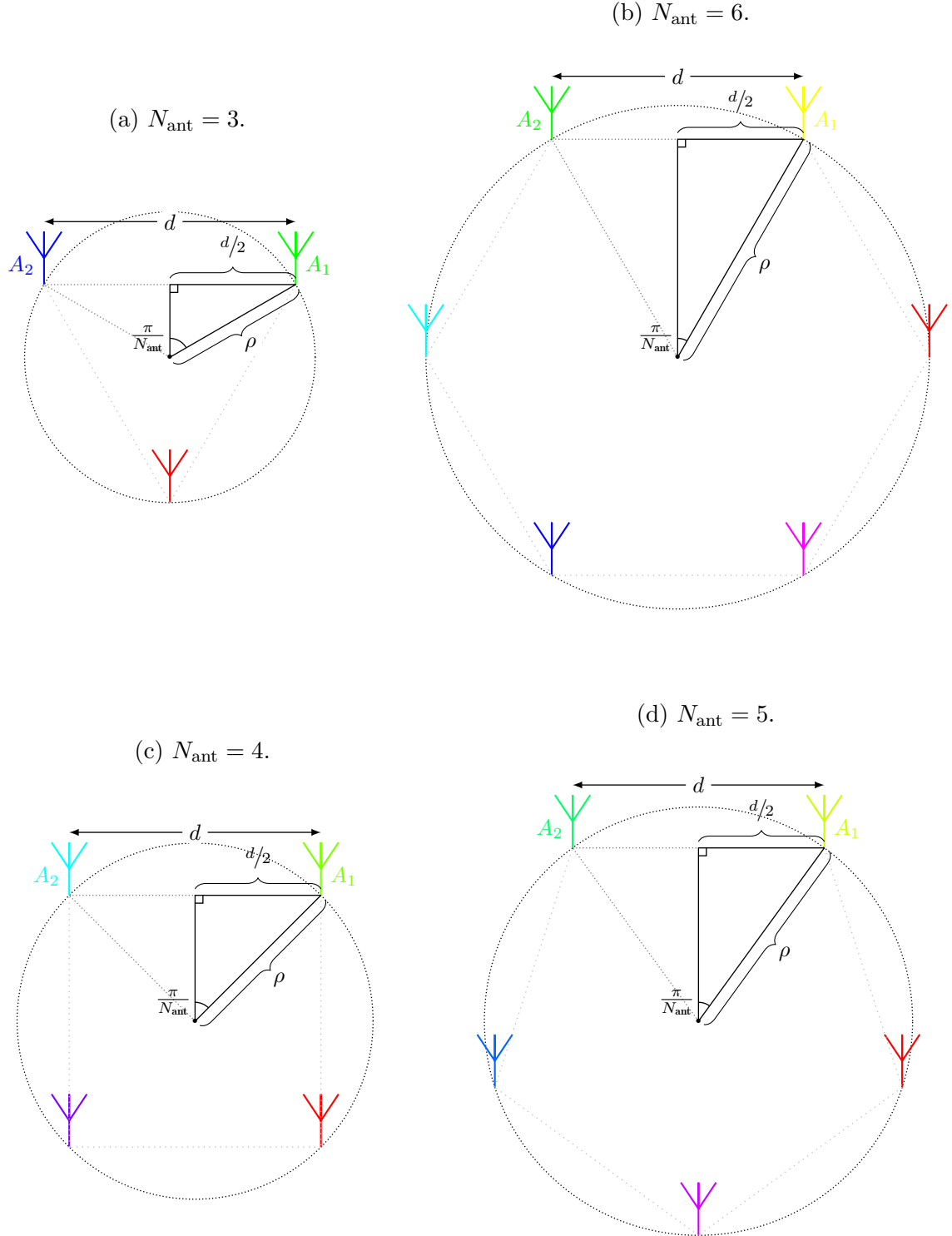
$$\alpha_k = \arg(A_k - A_{k+1}) \quad (15)$$

Para calcular a fase em uma antena, é interessante representar o sinal recebido como um valor complexo. Uma forma de obter o complexo de fase consiste em analisar a correlação do sinal incidente w com sinais de referência de mesma frequência que o sinal de interesse, em um período completo, Equação 16. Os valores I_k (em fase) e Q_k (em quadratura) são calculados respectivamente pela correlação com um cosseno, conforme Equação 17, e com um seno, conforme Equação 18. A Equação 19 apresenta o valor complexo Z_k de fase para a antena A_k .

$$T = \frac{2\pi}{\omega} = \frac{1}{f} \quad (16)$$

$$I_k = \int_0^T \cos(\omega \cdot \tau) \cdot w(x_{A_k}, y_{A_k}, \tau) \partial\tau \quad (17)$$

Figura 7: Diferentes distribuições de antenas.



Fonte: Autor.

$$Q_k = \int_0^T \sin(\omega \cdot \tau) \cdot w(x_{A_k}, y_{A_k}, \tau) \partial \tau \quad (18)$$

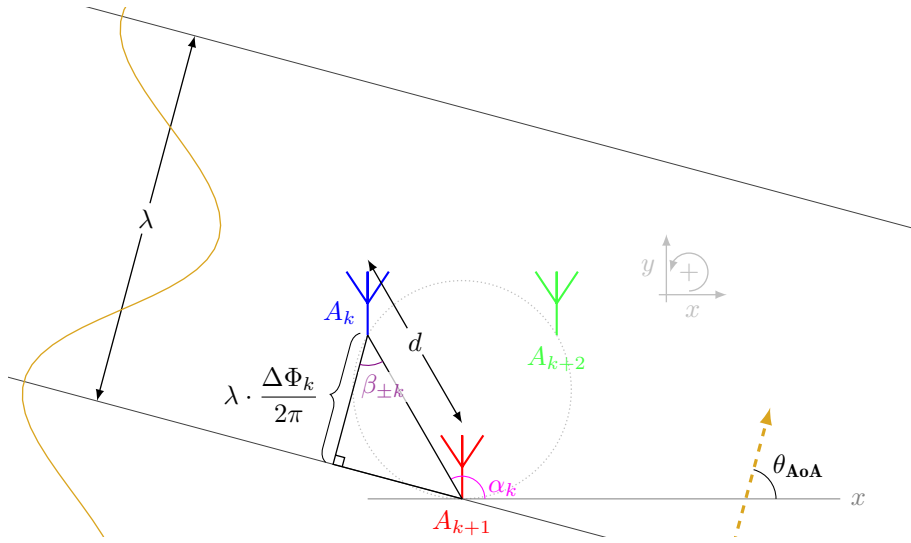
$$Z_k = \frac{\omega}{\pi} \cdot (I_k + \imath Q_k) \quad (19)$$

O cálculo de defasagem de sinal em um par de antenas consiste na análise de diferença de fase dos valores Z_k e Z_{k+1} do par de antenas A_k e A_{k+1} , conforme apresentado na Equação 20. Obtido o valor de defasagem $\Delta\Phi_k$ entre o par de antenas, finalmente é possível calcular o ângulo $\beta_{\pm k}$ através da Equação 21, note que a simplificação somente é possível com valor de $d = \lambda/2$. A Figura 8 apresenta a geometria do sistema destacando os valores de interesse na análise de um dos pares de antenas, tomando $N_{\text{ant}} = 3$.

$$\Delta\Phi_k = \Phi_k - \Phi_{k+1} = \arg(Z_k) - \arg(Z_{k+1}) = \arg(Z_k \cdot \overline{Z_{k+1}}) \quad (20)$$

$$\beta_{\pm k} = \arccos\left(\frac{\lambda}{d} \cdot \frac{\Delta\Phi_k}{2\pi}\right) \quad (21)$$

Figura 8: Geometria geral do sistema com $N_{\text{ant}} = 3$.



Fonte: Autor.

Para cada par de antenas, são calculados dois valores $\theta_{\pm k}$ conforme a Equação 22, equivalentes a dois valores possíveis para o θ_{AoA} . A Equação 23 define o conjunto Θ dos valores aferidos de $\theta_{\pm k}$ para todos os pares de antenas do sistema, este conjunto sempre terá $2 \cdot N_{\text{ant}}$ elementos, dos quais, metade estão próximos do real valor de θ_{AoA} e os demais são valores distintos do objetivo e entre si.

$$\theta_{\pm k} = \alpha_k \pm \beta_{\pm k} \quad (22)$$

$$\Theta = \{\theta_{\pm k} \mid \forall k\} \quad (23)$$

Com os possíveis valores de θ_{AoA} obtidos, é necessário estimar qual o valor correto. Para isso, é criada uma lista auxiliar $\Theta_{[\bullet]}$, quantizando os valores de Θ em intervalos de tamanho δ , descrito na Equação 24. A Equação 25 descreve a operação de quantização dos valores de Θ , que, por se tratar de um cálculo auxiliar, utiliza-se o arredondamento para o inteiro mais próximo. A quantização implica que os valores de Θ serão agrupados por faixas de largura δ .

$$\delta = \frac{\pi}{2 \cdot (1 + N_{\text{ant}})} \quad (24)$$

$$\Theta_{[\bullet]} = \left\{ \left\lfloor \frac{\theta}{\delta} \right\rfloor \cdot \delta \mid \forall \theta \in \Theta \right\} \quad (25)$$

Salvo casos com muito ruído, espera-se que alguns valores em $\Theta_{[\bullet]}$ se repitam, partindo disso, calcula-se $\theta_{\mathcal{M}_o}$, a moda estatística destes valores, conforme Equação 26. Esse valor deverá estar próximo ao θ_{AoA} , e será utilizado na filtragem dos valores aferidos em Θ .

$$\theta_{\mathcal{M}_o} = \mathcal{M}_o(\Theta_{[\bullet]}) \quad (26)$$

O conjunto Θ_F contém itens de Θ que estejam ao redor do valor $\theta_{\mathcal{M}_o}$ calculado, num intervalo de δ para mais ou para menos, conforme Equação 27.

$$\Theta_F = \{\theta \in \Theta \mid \theta_{\mathcal{M}_o} - \delta \leq \theta \leq \theta_{\mathcal{M}_o} + \delta\} \quad (27)$$

Finalmente obtém-se o valor de θ_{AoA} pela mediana dos valores em Θ_F , conforme Equação 28.

$$\theta_{\text{AoA}} = \widetilde{\Theta_F} \quad (28)$$

2.2 Trabalhos relacionados

Em seu trabalho, Horst [12] analisa dois algoritmos de detecção de AoA, realizando as análises em ambientes internos e utilizando arranjos de antenas. O primeiro método analisado consiste em uma aproximação do ângulo, feita utilizando um *software* fornecido pela Texas Instruments, fabricante do *hardware* utilizado. Já o segundo método, baseia-se na construção matemática do AoA calculado pela diferença de fase instantânea do sinal entre as antenas do sistema, uma abordagem semelhante à proposta neste trabalho. Os resultados obtidos indicam que o método de aproximação teve melhor acurácia nos valores de ângulo.

A proposta de Zeaiter *et al.* [15] busca validar a performance da detecção de AoA em ambiente fechado, realizando a análise em diferentes modulações, larguras de canal e fatores

de espalhamento. Também propõe que, ao combinar de seu algoritmo de localização de AoA com a função de autocorrelação, é possível analisar os dados de dois sinais recebidos simultaneamente.

Outro trabalho de Zeaiter *et al.* [16] consiste em uma aproximação do AoA utilizando um método de autocorrelação em um sinal *Longe Range* (LoRa) de baixa potência. Seu objetivo consiste em detectar o sinal LoRa operando em transmissão de baixa potência, caso onde a vida útil da bateria do sistema transmissor é estendida. O algoritmo apresentado busca picos de autocorrelação no sinal recebido, além de utilizar Transformada Rápida de Fourier (*Fast Fourier Transform*, FFT) para denotá-los e melhorar a Relação Sinal-Ruído (*Signal-Noise Ratio*, SNR). Quando um pico é detectado, o algoritmo é capaz de encontrar o AoA.

BniLam *et al.* [17] propõe uma técnica que, sem qualquer informação prévia de largura de banda, consegue estimar AoA do sinal recebido. O sistema proposto consiste em uma Arranjo Circular Uniforme (*Uniform Circular Array*, UCA) seguida de um filtro transversal, também utiliza de vetores especiais de largura de banda variável junto com um estimador de relação sinal-ruído térmico para determinar simultaneamente AoA e largura de banda do sinal recebido.

Em outro trabalho, BniLam *et al.* [18] estudam a possibilidade de estimar AoA para transceptores de Internet das Coisas (*Internet of Things*, IoT) em ambiente interno. Também propõe um modelo probabilístico adaptativo que opera no modelo de estimativa de AoA, incrementando sua performance. Seus resultados indicam que estes métodos superam a performance de modelos probabilísticos estáticos tradicionais, tanto em acurácia de localização quanto em estabilidade no valor obtido.

Neste trabalho, BniLam *et al.* [19] propõe um dispositivo de baixo custo capaz de estimar o AoA, de forma que seja viável sua utilização em dispositivos de IoT. O dispositivo consiste em uma conversão de vários Rádio Definido por *Software* (*Software-Defined Radio*, SDR) individuais de baixo custo num único SDR com múltiplos canais de RF. Seus resultados experimentais indicam que o dispositivo é capaz de estimar valores de AoA de forma estável e acurada.

A proposta de BniLam *et al.* [20] neste trabalho consiste em um novo algoritmo para determinação de AoA chamado ANGLE (*ANGular Location Estimation*), baseado em modelos probabilísticos para a resposta do sinal recebido. Sua proposta ainda sugere duas versões do método, para o caso de amostragem única e de decomposição de subespaço, como utilizado no algoritmo MUSIC (*MUltiple SIgnal Classification*).

BniLam *et al.* [21] apresenta, neste trabalho, uma abordagem mais amigável para estimativa de AoA em redes LoRa. O sistema proposto, denominado LoRay (LoRa *array*) é composto por *hardware* e *software* preparados para fazer a estimativa de AoA em ambiente urbano, onde o sistema foi validado. O hardware utilizado foi descrito em um trabalho anterior [19]. Este sistema apresentou resultados estáveis e acurados para estimativa de AoA tanto nos casos Linha de visão (*Line of Sight*, LoS) quanto nos Sem Linha de visão (*Non Line of Sight*, NLoS).

Em seu trabalho, Niculescu e Nath [22] propõe métodos para detecção de posição

e orientação em cada nó de uma rede *ad hoc*. A proposta parte de possíveis problemas relacionados à utilização de GPS em ambiente fechado

3 Metodologia

Neste capítulo, são explorados os métodos utilizados para a construção do trabalho proposto.

3.1 Simulação

A construção da simulação partiu de uma abordagem físico-matemática, definindo o sinal w como uma função de onda relativa ao tempo e ao espaço, analisando seus valores incidindo em cada antena A_k e comparando as defasagens $\Delta\Phi_k$ entre os diferentes pares de antenas. Para simplificar a construção da simulação, foram utilizadas funções paramétricas, descritas na presente seção.

3.1.1 Parâmetros envolvidos

Com o objetivo de garantir a coerência entre as partes da simulação, vários parâmetros foram utilizados, definindo detalhes em relação às operações matemáticas e às formas de registro dos valores calculados. Estes parâmetros são divididos entre os que recebem valores numéricos, booleanos ou matrizes numéricas.

Os parâmetros numéricos são:

- `amp_w`, amplitude desejada para o sinal;
- `ang_w`, direção do emissor do sinal, equivalente ao ângulo θ_{AoA} de chegada do sinal em relação à malha de antenas;
- `angle_Z_A_x_B`, ângulo relativo $\beta_{\pm k}$ para par de antenas;
- `d`, distância d entre par de antenas do arranjo;
- `D_signals`, ?? ;
- `choose_angle`, ângulo θ_{AoA} final calculado pelo sistema;
- `interval`, indica os limites para a geração de imagem da simulação;
- `lambda_w`, comprimento de onda λ ;
- `N_antenas`, quantidade N_{ant} de antenas do arranjo;
- `omega_w`, frequência angular ω ;
- `phase_w`, fase ϕ do sinal no emissor;
- `Rho`, raio ρ do polígono que dispõe as antenas no arranjo;

- `r_w`, distância que o emissor de sinal está da coordenada (0, 0) do sistema;
- `range_step`, largura em graus do passo na simulação.
- `resolution`, relativo à quantidade de pontos utilizados na aproximação numérica do cálculo de correlação;
- `SNR`, valor da SNR linear;
- `SNR_dB`, valor da SNR em dB;
- `t_w`, tempo t associado ao instante de aferição do sinal;
- `x_w` ou `y_w`, coordenada x ou y no espaço para aferição do sinal w ;
- `Z_antenna`, `Z_antenna_A` ou `Z_antenna_B`, valor complexo, coordenada de antena;
- `Z_phase_A` ou `Z_phase_B`, valor complexo, fase Φ_k de antena;

Os parâmetros booleanos são:

- `ATT`, indica se o sinal contará com atenuação por distância;
- `C`, indica a utilização de componente cossenoidal na construção do sinal;
- `CHG_PHI`, indica se a fase geral do sinal deve mudar ao longo da simulação;
- `CHG_R`, indica se a distância do emissor do sinal deverá mudar ao longo da simulação;
- `CHG_THETA`, indica se o ângulo de origem do sinal deverá mudar ao longo da simulação;
- `NOISE`, indica se o sinal contará com ruído;
- `S`, indica a utilização de componente senoidal na construção do sinal;
- `S_DAT`, indica se os pontos gerados pela simulação deverão ser salvos;
- `S_GIF`, indica se a imagem gerada pela simulação deverá ser salva;

Os parâmetros de matrizes numéricas são:

- `AoA_range`, ?? ;
- `ant_array`, coordenadas das antenas do arranjo;
- `delta_A_x_B_array`, contendo o ângulo $\theta_{\pm k}$ calculado por $\alpha_k + \beta_{\pm k}$ aferido para cada par de antenas do arranjo;
- `delta_B_x_A_array`, contendo o ângulo $\theta_{\pm k}$ calculado por $\alpha_k - \beta_{\pm k}$ aferido para cada par de antenas do arranjo;

- `Z_phase_array`, matriz de valores numéricos complexos, contendo o sinal complexo aferido para cada antena do arranjo;
- `Z_phase_matrix`, ?? ;
- `z_plot`, estado corrente do sinal no espaço, utilizado na geração de imagem da simulação;
- `Z_x_array`, valores complexos, contendo a defasagem $\Delta\Phi_k$ aferido para cada par de antenas no arranjo;

Parâmetros de outros tipos:

- `steering_function`, ?? ;
- `steering_arguments`, ?? ;

3.1.2 Funções auxiliares

A primeira função a ser definida é `argument_r`, que opera como auxiliar para normalização de argumento para as funções trigonométricas utilizadas nas análises, garantindo coerência em frequência angular e coordenadas espaciais. Seus argumentos são, respectivamente, `x_w`, `y_w`, `t_w`, `ang_w`, `r_w`, `phase_w`, `lambda_w` e `omega_w`. O Código 1 apresenta uma versão simplificada da função `argument_r` desenvolvida.

Código 1: Função `argument_r`, simplificada.

```

1 function res = argument_r(...)
2     r_0 = r_w * lambda_w;
3
4     x_0 = r_0 * cos(ang_w);
5     y_0 = r_0 * sin(ang_w);
6
7     res = (2*pi/lambda_w) * ( sqrt((y_w-y_0).^2 + ...
8         (x_w-x_0).^2) ) + omega_w*t_w + phase_w;
9 end %function

```

Fonte: Autor.

Para determinar a fase do sinal w , incidente em cada antena A_k , calcula-se a correlação deste sinal com sinais de referência seno e cossenos, fornecidos respectivamente pelas funções `ref_sin` e `ref_cos`. As duas funções recebem os mesmos argumentos, e estes são, respectivamente, `t_w` e `omega_w`. Ambos os casos utilizam a função `argument_r` para garantir coerência de frequência com o sinal incidente. Os Códigos 2 e 3 apresentam, respectivamente, versões simplificadas das funções `ref_cos` e `ref_sin` desenvolvidas.

A próxima função construída foi `signal_r`, que calcula o valor do sinal w numa coordenada (x, y) e um instante t . Considera-se que o sinal é composto pela soma de seno e cosseno, e que são determinadas a distância e a direção de sua fonte emissora.

Código 2: Função `ref_cos`, simplificada.

```
1 function c = ref_cos(...)
2     c = cos(argument_r(0, 0, t_w, 0, 0, 0, 1, omega_w));
3 end %function
```

Fonte: Autor.

Código 3: Função `ref_sin`, simplificada.

```
1 function s = ref_sin(...)
2     s = sin(argument_r(0, 0, t_w, 0, 0, 0, 1, omega_w));
3 end %function
```

Fonte: Autor.

Também é possível definir amplitude e fase na origem, além da presença de atenuação e ruído do tipo AWGN. Seus argumentos são, respectivamente, `x_w`, `y_w`, `t_w`, `amp_w`, `ang_w`, `r_w`, `phase_w`, `lambda_w`, `omega_w`, `S`, `C`, `NOISE`, `SNR_dB` e `ATT`. É utilizada a função `argument_r` para garantir coerência de frequência entre as componentes e com os sinais de referência utilizados no cálculo de correlação. Para implementação do ruído, foi utilizada a função `awgn`, no GNU Octave, é necessária a biblioteca `communications`, porém para o MATLAB, não é necessário carregar bibliotecas [23, 24]. O Código 4 apresenta uma versão simplificada da função `signal_r` desenvolvida.

A função `phase_z` calcula o valor complexo de fase Z_k para a antena A_k através da correlação pelos sinais de seno e cosseno. Seus argumentos são, respectivamente, `t`, `Z_antenna`, `amp_w`, `ang_w`, `r_w`, `phase_w`, `lambda_w`, `omega_w`, `S`, `C`, `NOISE`, `SNR_dB` e `ATT`. O Código 5 apresenta uma versão simplificada da função `phase_z` desenvolvida.

O cálculo do valor complexo de defasagem $\Delta\Phi_k$, o ângulo relativo $\beta_{\pm k}$ e o ângulo α_k entre um par de antenas é realizado pela função `dephase_A_to_B`. Seus argumentos são, respectivamente, `Z_phase_A` e `Z_phase_B`. O Código 6 apresenta uma versão simplificada da função `dephase_A_to_B` desenvolvida.

Os ângulos $\theta_{\pm k}$ para um par de antenas são calculados pela função `deltas_A_B`. Seus argumentos são, respectivamente, `angle_Z_A_x_B`, `Z_antenna_A` e `Z_antenna_B`. O Código 7 apresenta uma versão simplificada da função `deltas_A_B` desenvolvida.

A última função auxiliar desenvolvida foi `is octave`, que confere se a corrente simulação está sendo executada no GNU Octave, retornando um valor binário e não recebe qualquer parâmetro. O Código 8 apresenta uma versão simplificada da função `is octave` desenvolvida.

3.1.3 Função de cálculo para AoA

A primeira grande função desenvolvida foi `calc_AoA_geometric`, que é responsável pelo cálculo geral da simulação. Inicialmente são calculadas as coordenadas das N_{ant} antenas e, em sequência, os valores de fase do sinal incidente w em cada antena A_k , então as defasagens entre os pares de antenas e finalmente a seleção do valor mais provável

Código 4: Função `signal_r`, simplificada.

```

1 function res = signal_r(...)
2     res = 0;
3     if S
4         res = res + sin( argument_r(...) );
5     end %if
6     if C
7         res = res + cos( argument_r(...) );
8     end %if
9     if S && C
10        res = res / sqrt(2);
11    end %if
12    if ATT
13        %%% Lei de Friis
14        G_t = 1; % Ganho Antena Tx
15        G_r = 1; % Ganho Antena Rx
16        %%% Potencia eletrica
17        R_t = 1; % Resistencia Antena Tx (Reatância)
18        R_r = 1; % Resistencia Antena Rx (Reatância)
19        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
20        P_t = (amp_w^2)/R_t;
21        P_r = P_t * G_t * G_r * (lambda_w / (4 * pi * r_w));
22        amp_r = sqrt(P_r * R_r);
23        res = res * amp_r;
24    else
25        res = res * amp_w;
26    end %if
27    if NOISE
28        res = awgn(res, SNR_dB, 'measured');
29    end %if
30 end %function

```

Fonte: Autor.

Código 5: Função `phase_z`, simplificada.

```

1 function Z_phase = phase_z(...)
2     I_medido = trapz(t, ref_cos(...) .* signal_r(...));
3     Q_medido = trapz(t, ref_sin(...) .* signal_r(...));
4
5     Z_phase = (omega_w/pi)*(I_medido + i*Q_medido);
6 end % function

```

Fonte: Autor.

Código 6: Função `dephase_A_to_B`, simplificada.

```
1 function [Z_phase_A_x_B angle_Z_A_x_B] = dephase_A_to_B(...)  
2     Z_phase_A_x_B = Z_phase_A * conj(Z_phase_B);  
3     deltaPhi_A_x_B = angle(Z_phase_A_x_B);  
4     angle_Z_A_x_B = acos(deltaPhi_A_x_B/(pi));  
5 end % function
```

Fonte: Autor.

Código 7: Função `deltas_A_B`, simplificada.

```
1 function [delta_A_x_B delta_B_x_A] = deltas_A_B(...)  
2     ang_A_x_B = deg2rad(mod(rad2deg(...  
3         angle(Z_antenna_A - Z_antenna_B)),360));  
4     delta_A_x_B = ang_A_x_B + angle_Z_A_x_B;  
5     delta_B_x_A = ang_A_x_B - angle_Z_A_x_B;  
6 end % function
```

Fonte: Autor.

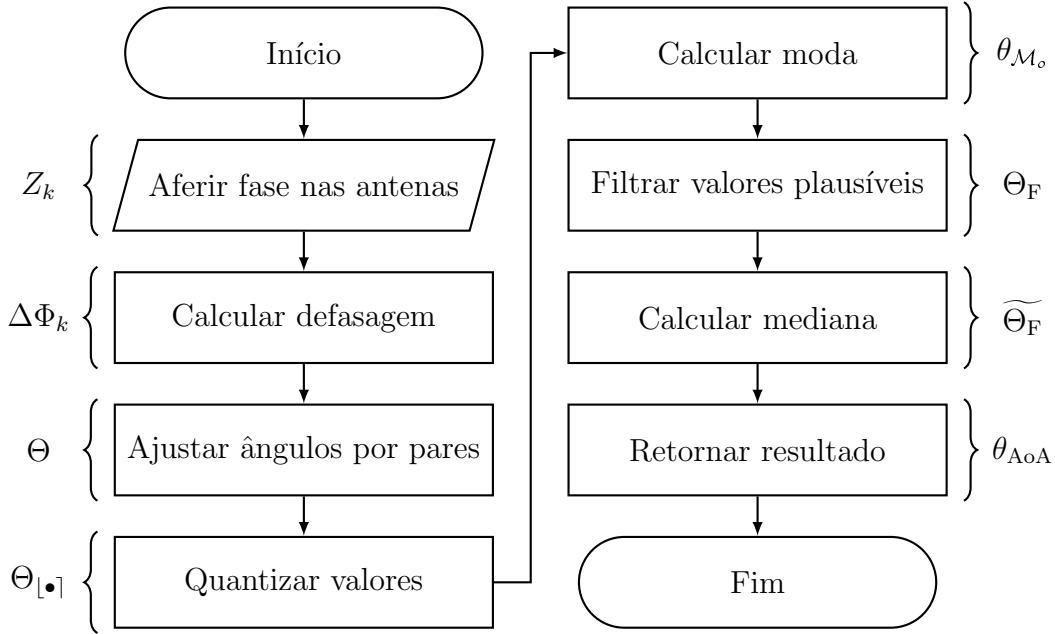
Código 8: Função `is octave`, simplificada.

```
1 function r = is octave ()  
2     persistent x;  
3     if (isempty (x))  
4         x = exist ('OCTAVE_VERSION', 'builtin');  
5     end  
6     r = x;  
7 end
```

Fonte: Autor.

para θ_{AoA} . Seus argumentos são, respectivamente, `amp_w`, `ang_w`, `r_w`, `phase_w`, `lambda_w`, `omega_w`, `S`, `C`, `NOISE`, `SNR_dB`, `ATT`, `resolution`, `d` e `N_antenas`. Nessa função também são definidas três subfunções auxiliares `phase_z`, `dephase_A_to_B` e `deltas_A_B`. O Código 9 apresenta uma versão simplificada da função `calc_AoA_geometric` desenvolvida. A Figura 9 apresenta a sequências de operações realizadas pela função.

Figura 9: Fluxograma de operações da função `calc_AoA_geometric`.



Fonte: Autor.

3.1.4 Função de geração saída visual

A segunda grande função desenvolvida foi `generate_fig`, que constrói a animação de saída da simulação, formada por dois gráficos. O primeiro gráfico, à esquerda nas animações geradas, apresenta a disposição das antenas, os valores de fase para cada uma delas, os valores de defasagem entre os pares de antenas, todos os possíveis valores de $\theta_{\pm k}$, e finalmente o valor real e o escolhido para θ_{AoA} . O segundo gráfico, à direita nas animações geradas, apresenta a disposição das antenas e uma representação do sinal w no espaço exibido. Os valores exibidos são calculados pela função `calc_AoA_geometric`. Seus argumentos são, respectivamente, `z_plot`, `x_w`, `y_w`, `ang_w`, `lambda_w`, `interval`, `Rho`, `choose_angle`, `ant_array`, `Z_phase_array`, `Z_x_array`, `delta_A_x_B_array` e `delta_B_x_A_array`. A Figura 10 ilustra os gráficos gerados pela função `generate_fig`.

3.1.5 Função geral da simulação

Finalmente a função responsável por juntar todas as partes é `w_xyt`, a base para a simulação, ela invoca as funções `calc_AoA_geometric` e `generate_fig` com os devidos

Código 9: Função `calc_AoA_geometric`, simplificada.

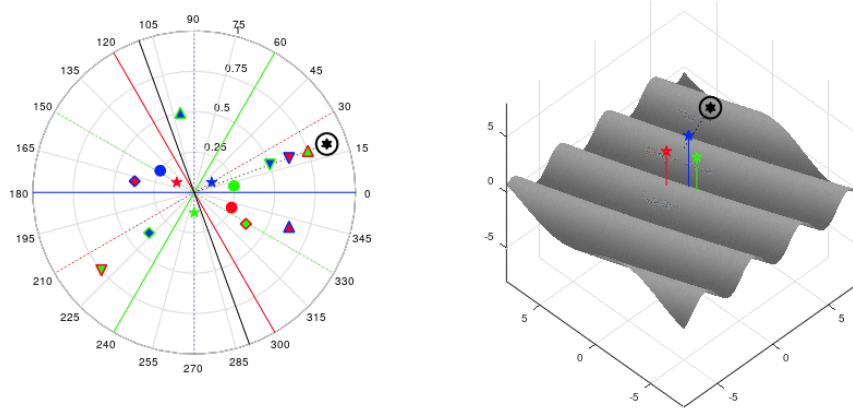
```

1 function return_struct = calc_AoA_geometric(...)
2     Rho = d/(2*sin(pi / N_antenas));
3     ant_angles = % ...
4     ant_array = Rho * exp(i * deg2rad(ant_angles));
5
6     Z_phase_array = arrayfun(@(a) phase_z(...), ant_array);
7     [Z_x_array angle_Z_A_x_B_array] = arrayfun( ...
8         @(a, b) dephase_A_to_B(a, b), ... );
9
10    [delta_A_x_B delta_B_x_A] = arrayfun( ...
11        @(ang, a, b) deltas_A_B(ang, a, b), ... );
12
13    range_angle = pi/(2*(N_antenas+1));
14
15    angle_vector = [delta_A_x_B delta_B_x_A];
16    angle_vector = [...]; % Normalização
17
18    angle_vector_round = ...
19        round(angle_vector./range_angle).*range_angle;
20
21    target_angle = mode(angle_vector_round);
22
23    angle_vector = angle_vector(abs(target_angle ...
24        - angle_vector) <= range_angle );
25
26    choose_angle = median(angle_vector);
27
28    return_struct = { ...
29        choose_angle ...
30        Rho ...
31        ant_array ...
32        Z_phase_array ...
33        Z_x_array ...
34        delta_A_x_B ...
35        delta_B_x_A ...
36    };
37
38 end %function

```

Fonte: Autor.

Figura 10: Exemplo de quadro da animação de saída da função `generate_fig`.



Fonte: Autor, saída gráfica disponível em [GitHub](#).

parâmetros, além de garantir que os arquivos gerados sejam salvos corretamente. Seus argumentos são, respectivamente, `NOISE`, `ATT`, `CHG_PHI`, `CHG_R`, `CHG_THETA`, `S_GIF`, `S_DAT`, `SNR`, `range_step` e `N_antenas`.

4 Resultados

Neste capítulo são apresentados resultados e detalhes sobre a performance do sistema proposto, comparando a acurácia para diferentes configurações. Também são apontados problemas encontrados ao longo do desenvolvimento do trabalho.

4.1 Performance da simulação

Foram analisadas diferentes quantidades de antenas, contando ou não com ruído e atenuação no sinal. Para todas as quantidades de antenas analisadas, são sumarizadas simulações com e sem Atenuação (ATT), diferentes valores de SNR, partindo do caso de ruído ideal (sem qualquer ruído, $\text{SNR} \rightarrow \infty$ dB) ao caso de potência de ruído igual à potência de sinal ($\text{SNR} = 0$ dB), com o emissor do sinal circular orbitando a uma distância fixa de 50λ do centro do sistema de antenas.

Para garantir a robustez do sistema, foram realizadas outras simulações, considerando casos onde o emissor está se aproximando do sistema de antenas e também casos onde o emissor está estático no espaço. Estas simulações não foram sumarizadas.

4.1.1 Três antenas

As simulações com três antenas foram as que apresentaram menores valores de R^2 dentre as analisadas. Apesar disso, todos os valores foram acima de 75 %.

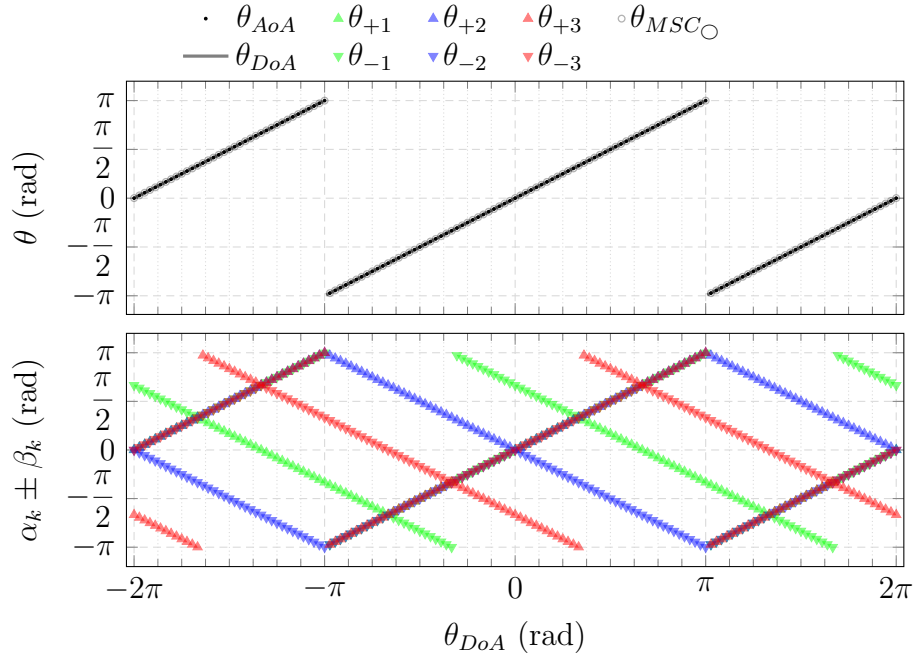
Simulações realizadas na configurações de três antenas têm os valores de R^2 apresentados na Tabela 1, e os resultados dos valores destacados são apresentados nas Figuras 11, 12 e 13.

Tabela 1: Coeficiente de determinação $R^2(\%)$ para simulações notáveis com três antenas.

SNR (dB)	R^2 (%)			
	Sem ATT		Com ATT	
	Geométrico	MUSIC _{UCA}	Geométrico	MUSIC _{UCA}
∞	92,06	92,06	92,06	92,06
20	88,05	84,35	93,89	92,05
17	94,25	92,06	95,98	92,06
14	92,07	92,06	90,00	84,35
7	88,33	92,06	79,35	92,06
0	80,79	77,16	65,83	92,06

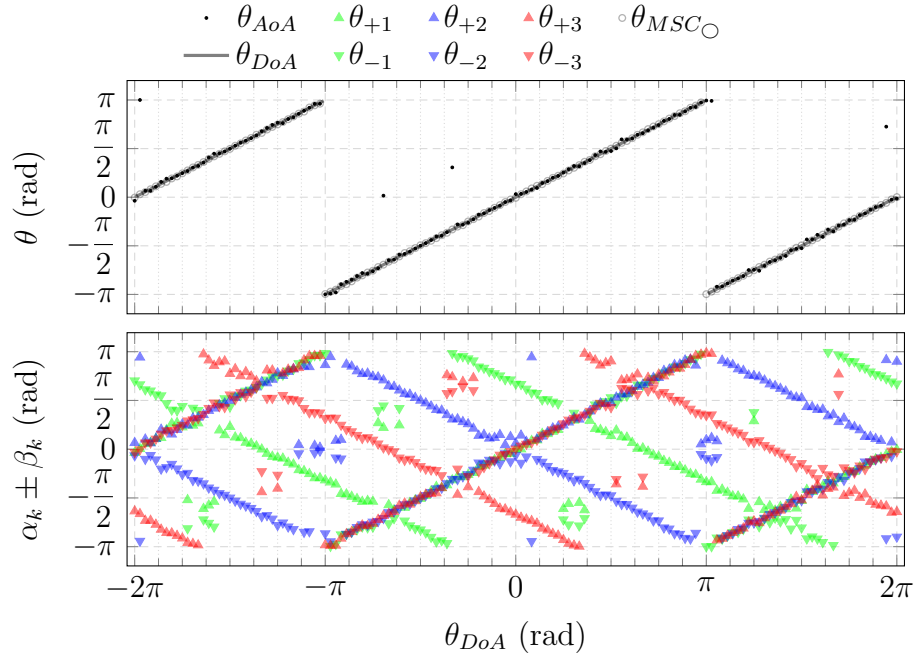
Fonte: Autor, saídas das simulações disponíveis em [GitHub](#).

Figura 11: Simulação para três antenas, caso ideal ($\text{SNR} \rightarrow \infty$ dB).



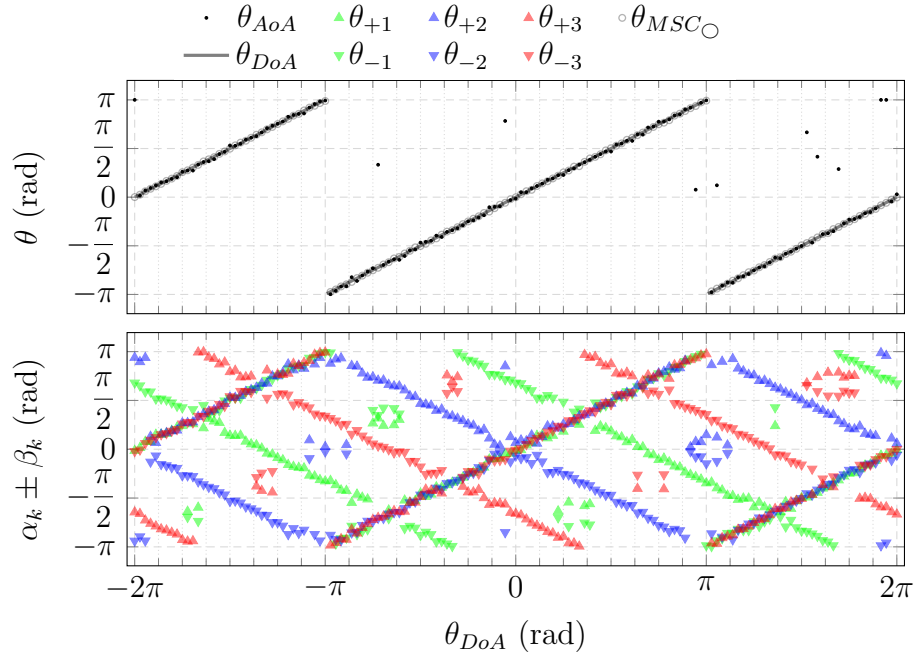
Fonte: Autor, saída gráfica disponível em [GitHub](#).

Figura 12: Simulação para três antenas, caso SNR = 0 dB, sem atenuação.



Fonte: Autor, saída gráfica disponível em [GitHub](#).

Figura 13: Simulação para três antenas, caso SNR = 0 dB, com atenuação.



Fonte: Autor, saída gráfica disponível em [GitHub](#).

4.1.2 Cinco antenas

As simulações com cinco antenas apresentaram valores intermediários de R^2 . Todos os valores foram acima de 75 %, e em média, acima de 80 %.

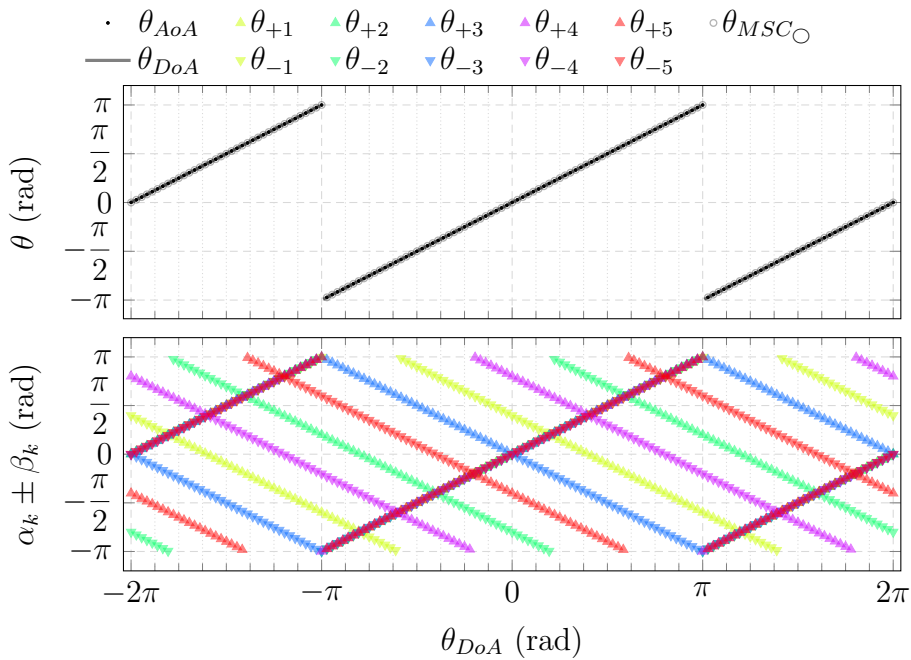
Simulações realizadas na configurações de cinco antenas têm os valores de R^2 apresentados na Tabela 2, e os resultados dos valores destacados são apresentados nas Figuras 14, 15 e 16.

Tabela 2: Coeficiente de determinação $R^2(\%)$ para simulações notáveis com cinco antenas.

SNR (dB)	R^2 (%)			
	Sem ATT		Com ATT	
	Geométrico	MUSIC _{UCA}	Geométrico	MUSIC _{UCA}
∞	92,07	92,05	92,07	92,05
20	92,07	92,06	92,06	84,34
17	100,00	100,00	84,36	84,55
14	100,00	84,35	92,08	84,35
7	84,38	84,35	84,53	77,09
0	89,83	84,37	77,37	100,00

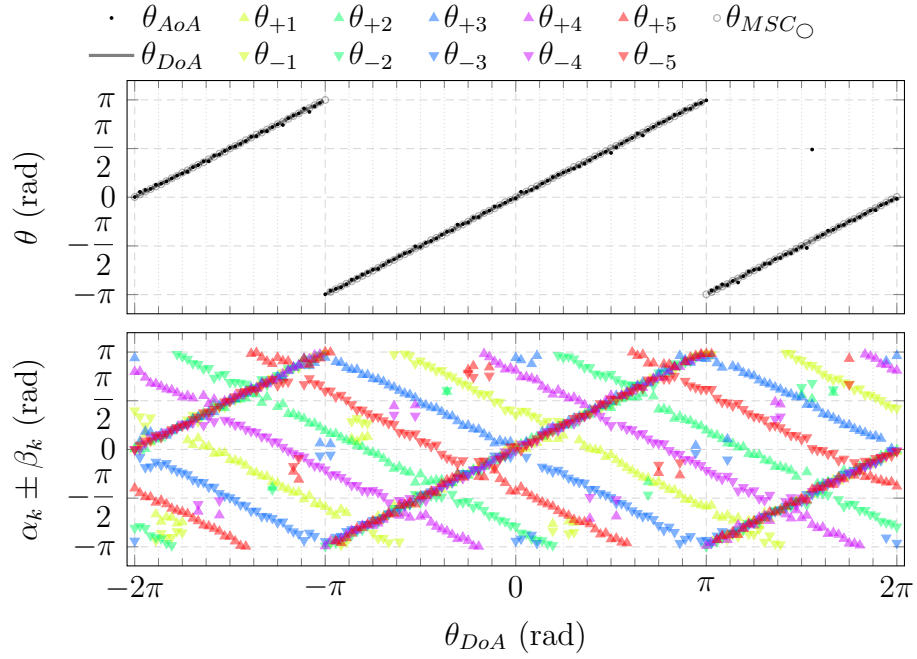
Fonte: Autor, saídas das simulações disponíveis em [GitHub](#).

Figura 14: Simulação para cinco antenas, caso ideal ($\text{SNR} \rightarrow \infty$ dB).



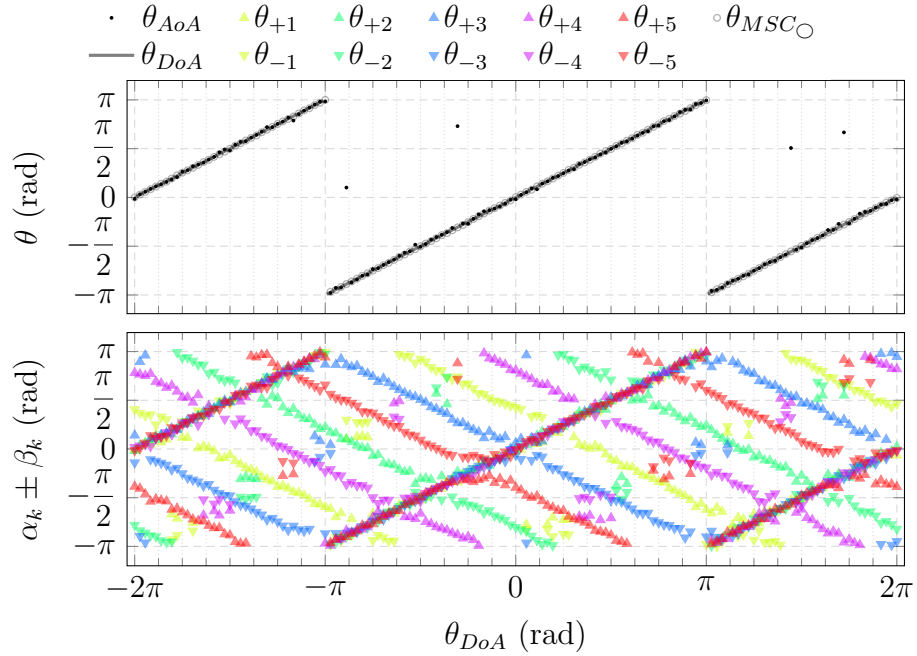
Fonte: Autor, saída gráfica disponível em [GitHub](#).

Figura 15: Simulação para cinco antenas, caso $\text{SNR} = 0$ dB, sem atenuação.



Fonte: Autor, saída gráfica disponível em [GitHub](#).

Figura 16: Simulação para cinco antenas, caso $\text{SNR} = 0$ dB, com atenuação.



Fonte: Autor, saída gráfica disponível em [GitHub](#).

4.1.3 Sete antenas

As simulações com sete antenas apresentaram os melhores valores de R^2 , com casos acima de 99 %. Todos os valores foram acima de 80 %.

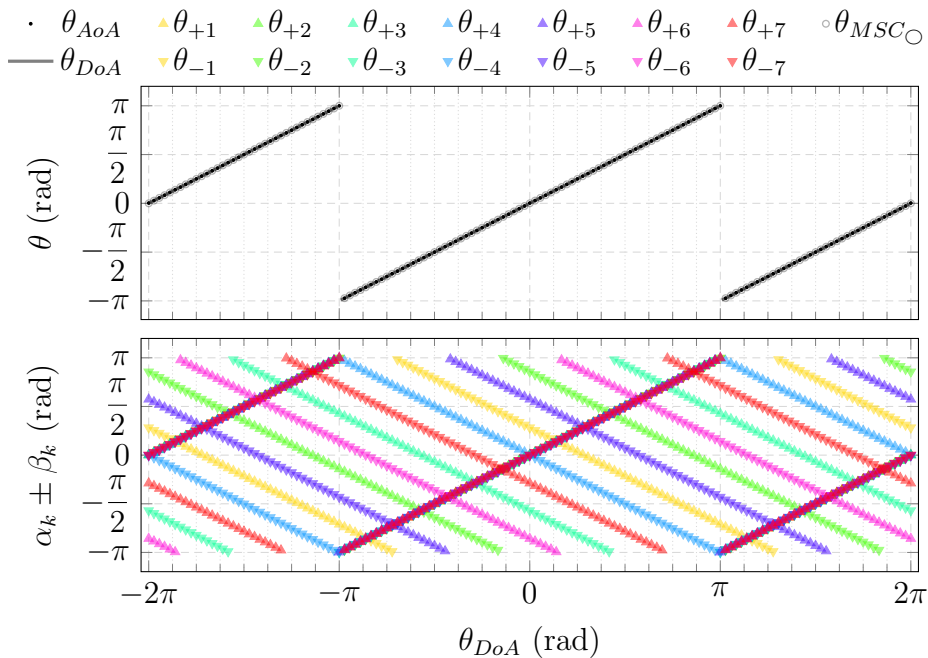
Simulações realizadas na configurações de sete antenas têm os valores de R^2 apresentados na Tabela 3, e os resultados dos valores destacados são apresentados nas Figuras 17, 18 e 19.

Tabela 3: Coeficiente de determinação $R^2(\%)$ para simulações notáveis com sete antenas.

SNR (dB)	R^2 (%)			
	Sem ATT		Com ATT	
	Geométrico	MUSIC _{UCA}	Geométrico	MUSIC _{UCA}
∞	92,07	84,54	92,07	84,54
20	100,00	92,06	92,09	92,06
17	100,00	84,54	84,37	77,07
14	84,41	84,35	92,08	92,06
7	92,14	92,06	92,09	100,00
0	83,09	92,06	84,43	92,06

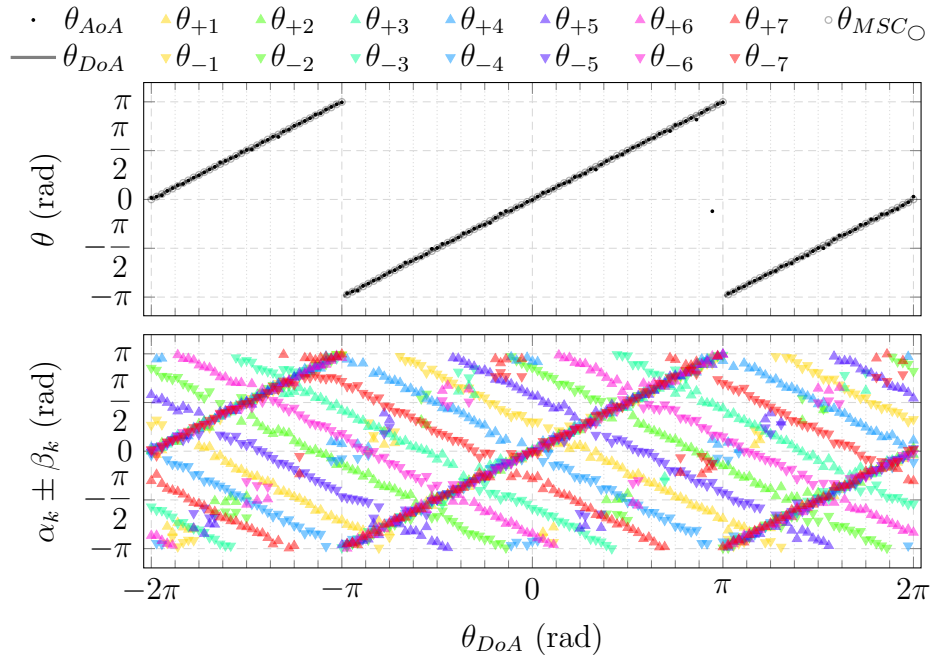
Fonte: Autor, saídas das simulações disponíveis em [GitHub](#).

Figura 17: Simulação para sete antenas, caso ideal ($\text{SNR} \rightarrow \infty$ dB).



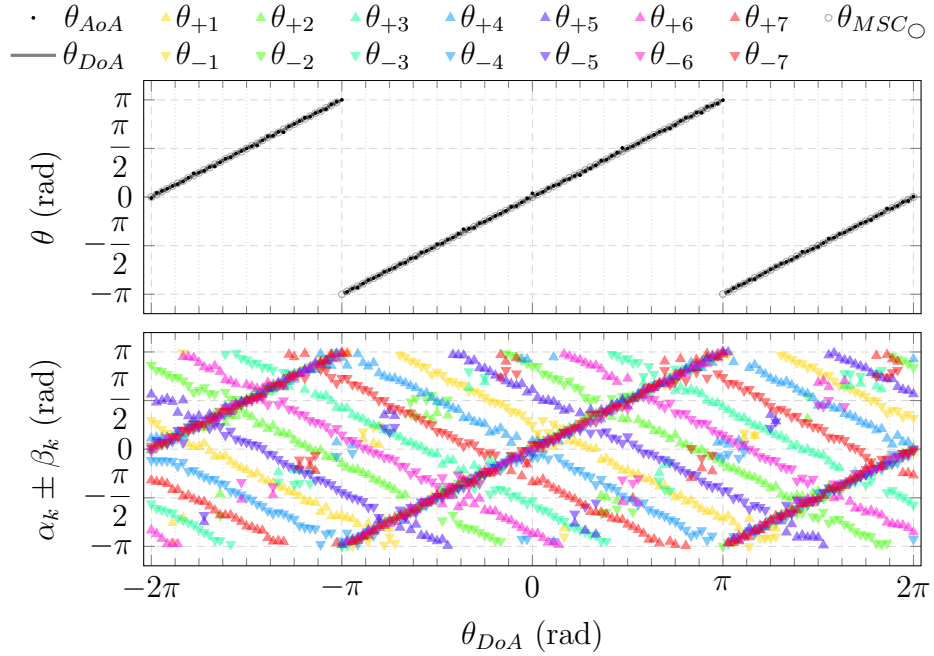
Fonte: Autor, saída gráfica disponível em [GitHub](#).

Figura 18: Simulação para sete antenas, caso SNR = 0 dB, sem atenuação.



Fonte: Autor, saída gráfica disponível em [GitHub](#).

Figura 19: Simulação para sete antenas, caso SNR = 0 dB, com atenuação.



Fonte: Autor, saída gráfica disponível em [GitHub](#).

4.2 Problemas encontrados

Ao longo do desenvolvimento do projeto, alguns problemas foram encontrados e contornados da melhor forma possível. Esta seção sumariza estes problemas e as soluções aplicadas.

4.2.1 Compatibilidade de código

Apesar de ter sido desenvolvido para o GNU Octave, houve a preocupação de manter o código compatível com o MATLAB. Partindo disso, foram necessárias várias alterações em partes do código, que não tinham o mesmo comportamento em ambos os *softwares*.

4.2.2 Limitações de *software* livre

Por se tratar de um *software* proprietário, o MATLAB não disponibiliza o código fonte de todas as suas ferramentas internas e, assim, nem todas as funcionalidades estão implementadas no GNU Octave. A falta de algumas dessas funções moldou o decorrer do desenvolvimento do projeto, optando por operações viáveis à versão de uso livre.

5 Conclusão

Foguetes de sondagem atmosférica podem pousar em qualquer lugar dentro do raio de alcance do voo, e recuperá-los pode ser inviável sem uma estratégia de localização eficaz. Uma estratégia muito utilizada é a localização por GNSS, por exemplo, o GPS, contudo, esta ainda depende que a equipe de busca tenha acesso às próprias coordenadas geográficas e comunicação efetiva com o sistema embarcado do foguete.

O presente trabalho propõe a utilização de um sistema de localização baseada no sinal RF emitido pelo veículo, e não pela informação contida nesse sinal, analisando as diferenças de defasagem do sinal incidente em uma malha de antenas, e assim calculando o AoA deste sinal.

Durante a revisão bibliográfica, fundamentou-se a teoria aplicada nessa proposta. Partindo de uma abordagem físico-matemática para analisar o sinal e a forma que a defasagem entre antenas pode ser utilizada para calcular a direção do emissor. Também foram listadas algumas propostas que atuam de forma semelhante, analisando o sinal incidente em uma malha de antenas, que demonstra a relevância do método. Além disso, foi realizado um breve levantamento sobre o método de direcionamento por coordenadas geográficas e o ângulo de *bearing*, que guia uma equipe de busca ao veículo almejado.

Com base na fundamentação físico-matemática, foi desenvolvida uma simulação com o sinal de RF incidente em uma malha de antenas. Considerando que o foguete esteja em solo, assumiu-se um espaço de duas dimensões, porém ainda mantendo a possibilidade do veículo, emissor do sinal, se mover livremente em relação ao sistema de antenas. As simulações foram construídas a partir dessas possibilidades, com o veículo circulando o sistema de antenas e se aproximando.

As simulações realizadas utilizaram geometrias de três, cinco e sete antenas. A escolha dessas quantidades deu-se por questões geométricas, pois polígonos regulares com uma contagem par de lados terão lados paralelos, enquanto polígonos de lados ímpares não apresentam essa propriedade. Os valores de R^2 para as configurações simuladas foram todos acima de 75 %, o que indica grande acurácia na modelagem proposta. A comparação entre as geometrias estudadas indicou que o sistema com três antenas teve uma acurácia média menor que as geometrias com mais antenas.

As limitações impostas pelo uso de *software* livre fizeram com que fossem utilizados métodos diferentes dos levantados na revisão bibliográfica, porém o método estatístico proposto se mostrou eficaz nos testes realizados. Outra limitação foi relacionada à compatibilidade do código escrito, já que a sintaxe e algumas funções do MATLAB têm algumas diferenças em relação às equivalentes do GNU Octave.

Em conclusão, o trabalho aqui proposto se mostrou eficaz na determinação do AoA para um sinal incidente em uma malha de antenas, garantindo um valor de R^2 acima de 75 % em todos os casos e valor médio de R^2 acima de 92 %.

Para trabalhos futuros, é possível analisar outras disposições de antenas no arranjo.

Apesar da formulação atual optar por polígonos regulares por simplicidade, a matemática utilizada calcula os ângulos de cada par de antenas individualmente, o que viabiliza outras disposições de antenas, que respeitem a distância entre antenas de um par. Outras possibilidades englobam analisar mais classes de ruídos e até problemas de propagação multicaminho. Além disso, a construção de um dispositivo eletrônico capaz de realizar a aferição de fase em uma malha de antenas poderá corroborar no levantamento de outros problemas a serem analisados e também validar a presente proposta.

Referências

- [1] ISRO, Indian Space Research Organisation, Departament of Space, *Sounding Rockets*. endereço: <https://www.isro.gov.in/soundingRockets.html>.
- [2] ESA, European Space Agency, *Sounding rockets*. endereço: https://www.esa.int/Science_Exploration/Human_and_Robotic_Exploration/Research/Sounding_rockets.
- [3] M. Sabbatini e N. Sentse, “ESA User Guide to Low Gravity Platforms”, *Directorate of Human Spaceflight and Operations*,, 2014. endereço: https://www.esa.int/Science_Exploration/Human_and_Robotic_Exploration/Research/European_user_guide_to_low_gravity_platforms.
- [4] NASA, National Aeronautics and Space Administration, *About Sounding Rockets*. endereço: <https://www.nasa.gov/soundingrockets/overview/>.
- [5] ESRA, Experimental Sounding Rocket Association, *The Intercollegiate Rocket Engineering Competition*. endereço: <https://www.soundingrocket.org/what-is-irec.html>.
- [6] ESRA, Experimental Sounding Rocket Association, *Spaceport America Cup Intercollegiate Rocket Engineering Competition Rules & Requirements Document*. endereço: https://www.soundingrocket.org/uploads/9/0/6/4/9064598/sa_cup_irec_rules_and_requirements_document-2023_v1.3_20231001.pdf.
- [7] P. Guitarrara, *Coordenadas geográficas*. endereço: <https://brasilescola.uol.com.br/geografia/coordenadas-geograficas.htm>.
- [8] C. Veness, *Calculate distance, bearing and more between Latitude/Longitude points*. endereço: <https://www.movable-type.co.uk/scripts/latlong.html>.
- [9] H. Fleming, *Coordenadas esféricas*, ago. de 2003. endereço: <http://fma.if.usp.br/~fleming/diffeo/node4.html>.
- [10] D. Jennings, R. Drullinger, K. Evenson, C. Pollock e J. Wells, “The continuity of the meter: the redefinition of the meter and the speed of visible light”, *Journal of Research of the National Bureau of Standards*, v. 92, n. 1, p. 11, 1987.
- [11] A. Bensky, *Wireless positioning technologies and applications*. Artech House, 2016. endereço: https://scholar.rose-hulman.edu/cgi/viewcontent.cgi?article=1012&context=electrical_grad_theses.
- [12] V. Horst, “Localization and Angle-of-Arrival in Bluetooth Low Energy”, 2021. endereço: <https://www.ds.informatik.uni-kiel.de/en/teaching/bachelor-and-master-theses/completed-master-and-bachelor-theses/2021%20bachelor%20thesis%20Valentin%20Horst.pdf>.

- [13] M. Schüssel, “Angle of Arrival Estimation using WiFi and Smartphones”, 2016. endereço: https://ipin2016.web.uah.es/usb/app/descargas/223_WIP.pdf.
- [14] C. A. Balanis, *Antenna Theory: Analysis and Design*. John Wiley & Sons, 2005.
- [15] H. Zeaiter, O. Baala, F. Spies e V. Thierry, “Performance Evaluation of the Angle of Arrival of LoRa Signals under Interference”, em *9th IEEE International Conference on Communications and Electronics (ICE 2022)*, IEEE, Nha Trang, Vietnam: IEEE, jul. de 2022. endereço: <https://ut3-toulouseinp.hal.science/hal-03693641>.
- [16] H. Zeaiter, F. Spies, O. Baala e T. Val, “Measuring accurate Angle of Arrival of weak LoRa signals for Indoor Positioning”, em *12th International Conference on Indoor Positioning and Indoor Navigation (IPIN 2022)*, Beijing, China, set. de 2022. DOI: 10.1109/IPIN54987.2022.9918114. endereço: <https://hal.science/hal-03932846>.
- [17] N. BniLam, J. Steckel e M. Weyn, “2D angle of arrival estimations and bandwidth recognition for broadband signals”, em *2017 11th European Conference on Antennas and Propagation (EUCAP)*, IEEE, 2017, pp. 2041–2045. endereço: https://www.researchgate.net/publication/317397561_2D_angle_of_arrival_estimations_and_bandwidth_recognition_for_broadband_signals.
- [18] N. BniLam, G. Ergeerts, D. Subotic, J. Steckel e M. Weyn, “Adaptive probabilistic model using angle of arrival estimation for IoT indoor localization”, em *2017 International conference on indoor positioning and indoor navigation (IPIN)*, IEEE, set. de 2017, pp. 1–7.
- [19] N. BniLam, D. Joosens, J. Steckel e M. Weyn, “Low cost AoA unit for IoT applications”, em *2019 13th European Conference on Antennas and Propagation (EuCAP)*, IEEE, 2019, pp. 1–5. endereço: https://www.researchgate.net/profile/Noori-Bnilam-2/publication/332141599_Low_Cost_AoA_Unit_for_IoT_Applications/links/5ca2ee8d299bf1116956bf0e/Low-Cost-AoA-Unit-for-IoT-Applications.pdf.
- [20] N. BniLam, E. Tanghe, J. Steckel, W. Joseph e M. Weyn, “ANGLE: ANGular location estimation algorithms”, *IEEE access*, v. 8, pp. 14 620–14 629, 2020. endereço: <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8959176>.
- [21] N. BniLam, T. Janssen, M. Aernouts, R. Berkvens e M. Weyn, “LoRa 2.4 GHz communication link and range”, *Sensors*, v. 20, n. 16, p. 4366, 2020.
- [22] D. Niculescu e B. Nath, “Ad hoc positioning system (APS) using AOA”, em *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, vol. 3, 2003, 1734–1743 vol.3. DOI: 10.1109/INFCOM.2003.1209196.
- [23] Nir Krakauer, *Function Reference: awgn*, 2016. endereço: <https://octave.sourceforge.io/communications/function/awgn.html>.

- [24] The MathWorks, Inc., *awgn*, 2025. endereço: <https://www.mathworks.com/help/comm/ref/awgn.html>.

A Códigos desenvolvidos para simulação

O sistema desenvolvido, os arquivos de saída das simulações citadas ao longo do documento e os arquivos-fonte deste relatório estão disponíveis [neste repositório no GitHub](#).

A.1 Simulação de direcionamento GNSS

Código 10: Arquivo de código `bearing.m`.

```
1 EARTH_RADIUS = 6371E3;
2
3           % Lat           Lon
4 coord_ufabc_SA = [-23.64450270 -46.52808340];
5 coord_ufabc_SBC = [-23.67751378 -46.56349172];
6
7 A = coord_ufabc_SA;
8 B = coord_ufabc_SBC;
9
10 theta_A = deg2rad(A(1));
11 theta_B = deg2rad(B(1));
12
13 phi_A = deg2rad(A(2));
14 phi_B = deg2rad(B(2));
15
16 delta_theta = theta_B - theta_A;
17 delta_phi = phi_B - phi_A;
18
19 X = cos(theta_B)*sin(delta_phi);
20 Y = cos(theta_A)*sin(theta_B) - sin(theta_A)*cos(theta_B)*
    cos(delta_phi);
21 Z = (sin(delta_theta/2))^2 + cos(theta_B) * cos(theta_A) *
    (sin(delta_phi/2))^2;
22
23 beta = atan2(X,Y) - pi/2;
24
25 if beta < -pi
26     beta = beta + 2*pi
27 end
28
29 d = EARTH_RADIUS * 2 * atan2(sqrt(Z), sqrt(1 - Z))
30
31 d_deg = rad2deg(beta)
```

A.2 Simulação de AoA

A.2.1 Funções auxiliares

Código 11: Arquivo de código `argument_r.m`.

```
1 function res = argument_r( ... %  
2   x_w, ... % coordenada x associada ao ponto da antena  
3   y_w, ... % coordenada y associada ao ponto da antena  
4   t_w, ... % tempo t associado ao instante de afericao do  
   sinal  
5   ang_w, ... % angulo theta de chegada do sinal em relacao  
   ao sistema de antenas  
6   r_w, ... % distancia que o emissor de sinal está da  
   coordenada (0,0) do sistema  
7   phase_w, ... % defasagem do sinal no emissor  
8   lambda_w, ... % comprimento de onda  
9   omega_w ... % frequencia angular  
10 )  
11  
12   r_0 = r_w * lambda_w;  
13  
14   x_0 = r_0 * cos(ang_w);  
15   y_0 = r_0 * sin(ang_w);  
16  
17   res = (2*pi/lambda_w) * ( sqrt((y_w-y_0).^2 + (x_w-x_0)  
   .^2) ) + omega_w*t_w + phase_w;  
18 end %function
```

Código 12: Arquivo de código `ref_cos.m`.

```
1 function c = ref_cos( ...  
2   t_w, ... % tempo t associado ao instante de afericao do  
   sinal  
3   omega_w ... % frequencia angular  
4 ) % Funcao auxiliar de sinal cosseno  
5   c = cos(argument_r(0, 0, t_w, 0, 0, 0, 1, omega_w));  
6 end %function
```

Código 13: Arquivo de código `ref_sin.m`.

```
1 function s = ref_sin( ...  
2   t_w, ... % tempo t associado ao instante de afericao do  
   sinal  
3   omega_w ... % frequencia angular  
4 ) % Funcao auxiliar de sinal seno  
5   s = sin(argument_r(0, 0, t_w, 0, 0, 0, 1, omega_w));  
6 end %function
```

Código 14: Arquivo de código `signal_r.m`.

```

1 function res = signal_r( ...
2     x_w, ... % coordenada x associada ao ponto da antena
3     y_w, ... % coordenada y associada ao ponto da antena
4     t_w, ... % tempo t associado ao instante de afericao do
        sinal
5     amp_w, ... % amplitude desejada para o sinal
6     ang_w, ... % angulo theta de chegada do sinal em relacao
        ao sistema de antenas
7     r_w, ... % distancia que o emissor de sinal esta da
        coordenada (0,0) do sistema
8     phase_w, ... % defasagem phi do sinal
9     lambda_w, ... % comprimento de onda
10    omega_w, ... % frequencia angular
11    S, ... % utilizacao de funcao Seno
12    C, ... % utilizacao de funcao Cosseno
13    NOISE, ... % se o sinal contara com ruido
14    SNR_dB, ... % relacao sinal-ruido
15    ATT ... % se o sinal contara com atenuacao por distancia
16 ) % Funcao de sinal senoidal
17
18 res = 0;
19 if S
20     res = res + sin( argument_r(x_w, y_w, t_w, ang_w, r_w,
        phase_w, lambda_w, omega_w) );
21 end %if
22 if C
23     res = res + cos( argument_r(x_w, y_w, t_w, ang_w, r_w,
        phase_w, lambda_w, omega_w) );
24 end %if
25 if S && C
26     res = res / sqrt(2);
27 end %if
28
29 if ATT
30     %%% Lei de Friis
31     G_t = 1; % Ganho Antena Tx
32     G_r = 1; % Ganho Antena Rx
33     %%% Potencia eletrica
34     R_t = 1; % Resistencia Antena Tx (Reatância)
35     R_r = 1; % Resistencia Antena Rx (Reatância)
36     %%%%%%%%%%%%%%%
37
38     P_t = (amp_w^2)/R_t;
39
40     P_r = P_t * G_t * G_r * (lambda_w / (4 * pi * r_w));
41
42     amp_r = sqrt(P_r * R_r);

```

```

43     res = res * amp_r;
44     else
45         res = res * amp_w;
46     end %if
47
48     if NOISE
49         res = awgn(res, SNR_dB, 'measured');
50     end %if
51
52 end %function

```

Código 15: Arquivo de código `phase_z.m`.

```

1  function Z_phase = phase_z( ...
2      t, ...
3      Z_antenna, ...
4      amp_w, ...
5      ang_w, ...
6      r_w, ...
7      phase_w, ...
8      lambda_w, ...
9      omega_w, ...
10     S, ...
11     C, ...
12     NOISE, ...
13     SNR_dB, ...
14     ATT ...
15 )
16     % Calculos de amostra I/Q para antena em (0,0)
17     I_medido = trapz(t, ref_cos(t, omega_w) ...
18         .* signal_r(real(Z_antenna), imag(Z_antenna), t, ...
19         amp_w, ang_w, r_w, phase_w, lambda_w, omega_w, ...
20         S, C, NOISE, SNR_dB, ATT));
21     Q_medido = trapz(t, ref_sin(t, omega_w) ...
22         .* signal_r(real(Z_antenna), imag(Z_antenna), t, ...
23         amp_w, ang_w, r_w, phase_w, lambda_w, omega_w, ...
24         S, C, NOISE, SNR_dB, ATT));
25
26     Z_phase = (omega_w/pi)*(I_medido + i*Q_medido);
27 end % function

```

Código 16: Arquivo de código `dephase_A_to_B.m`.

```

1  function [Z_phase_A_x_B angle_Z_A_x_B] = dephase_A_to_B(
2      ...
3      Z_phase_A, ...
4      Z_phase_B ...
5  )
6      Z_phase_A_x_B = Z_phase_A * conj(Z_phase_B);
7      deltaPhi_A_x_B = angle(Z_phase_A_x_B);

```

```

7   angle_Z_A_x_B = acos(deltaPhi_A_x_B/(pi));
8 end % function

```

Código 17: Arquivo de código `deltas_A_B.m`.

```

1 function [delta_A_x_B delta_B_x_A] = deltas_A_B( ...
2   angle_Z_A_x_B, ...
3   Z_antenna_A, ...
4   Z_antenna_B ...
5 )
6   ang_A_x_B = deg2rad(mod(rad2deg(angle(Z_antenna_A -
   Z_antenna_B)),360));
7   delta_A_x_B = ang_A_x_B + angle_Z_A_x_B;
8   delta_B_x_A = ang_A_x_B - angle_Z_A_x_B;
9 end % function

```

Código 18: Arquivo de código `is octave.m`.

```

1 function r = is octave() % Confere se esta executando no
   octave ou nao
2   persistent x;
3   if (isempty (x))
4     x = exist ('OCTAVE_VERSION', 'builtin');
5   end
6   r = x;
7 end

```

Código 19: Arquivo de código `copy_to_documentation.m`.

```

1 function copy_to_documentation(...
2   N_antenas, ...
3   file_adress ...
4 )
5
6   folder = '';
7   folder = [folder 'POLY_' num2str(N_antenas)];
8   doc_foldername = fullfile('..', '..', 'documentation');
9
10  [ ...
11   origin_foldername, ...
12   name, ...
13 ] = file_adress{:};
14
15  destination_foldername = fullfile(doc_foldername, '
   pictures', folder);
16  if not(isfolder(destination_foldername))
17    mkdir(foldername);
18  end %if
19
20  sorigin_gif_filename = fullfile(origin_foldername, [name
   '.gif']);

```

```

21 destination_gif_filename = fullfile(
    destination_foldername, [name '.gif']);
22 copyfile(sorigin_gif_filename, destination_gif_filename,
    'f');
23
24 destination_foldername = fullfile(doc_foldername, 'data'
    , folder);
25 if not(isfolder(destination_foldername))
26     mkdir(foldername);
27 end %if
28
29 sorigin_dat_filename = fullfile(origin_foldername, [name
    '.dat']);
30 destination_dat_filename = fullfile(
    destination_foldername, [name '.dat']);
31 copyfile(sorigin_dat_filename, destination_dat_filename,
    'f');
32
33 sorigin_dat_r2_filename = fullfile(origin_foldername, [
    name '.r2.dat']);
34 destination_dat_r2_filename = fullfile(
    destination_foldername, [name '.r2.dat']);
35 copyfile(sorigin_dat_r2_filename,
    destination_dat_r2_filename, 'f');
36
37 end % function

```

A.2.2 Função de cálculo para AoA

Código 20: Arquivo de código `calc_AoA_geometric.m`.

```

1 function return_struct = calc_AoA_geometric( ...
2     amp_w, ...
3     ang_w, ...
4     r_w, ...
5     phase_w, ...
6     lambda_w, ...
7     omega_w, ...
8     S, ...
9     C, ...
10    NOISE, ...
11    SNR_dB, ...
12    ATT, ...
13    resolution, ...
14    d, ...
15    N_antenas ...
16 )
17
18 % Raio do circulo de circunscreve o poligono com N lados

```

```

    de tamanho d
19 Rho = d/(2*sin(pi / N_antenas));
20 ant_angles_shift = -90;
21 ant_angles = deg2rad((0 + ant_angles_shift):(360/
    N_antenas):(359 + ant_angles_shift));
22
23 % Coordenadas das antenas
24 ant_array = Rho * exp(i * ant_angles);
25
26 t = linspace(0,(2 * pi / omega_w), resolution); %
    Intervalo de integração
27
28 % Calculos de fase
29 Z_phase_array = arrayfun(@(a) phase_z(t, a, amp_w, ...
30     ang_w, r_w, phase_w, lambda_w, omega_w, S, C, NOISE,
    ...
31     SNR_dB, ATT), ant_array);
32
33 % Calculos de simetria
34 [Z_x_array angle_Z_A_x_B_array] = arrayfun(@(a, b)
    dephase_A_to_B(a, b), ...
35     Z_phase_array, circshift(Z_phase_array, -1));
36
37 [delta_A_x_B delta_B_x_A] = arrayfun(@(ang, a, b)
    deltas_A_B(ang, a, b), ...
38     angle_Z_A_x_B_array, ant_array, circshift(ant_array
    , -1));
39
40 % Fazer "votacao" de angulo escolhido
41 range_angle = pi/(2*(N_antenas+1));
42
43 angle_vector = [delta_A_x_B delta_B_x_A];
44 angle_vector = [...
45     (angle_vector(angle_vector<0)+(2*pi)) ...
46     (angle_vector(0<=angle_vector & angle_vector<=(2*pi)))
    ...
47     (angle_vector((2*pi)<angle_vector)-(2*pi)) ...
48 ]; % Normalizar vetor de angulos entre 0 e 2 pi
49
50 angle_vector_round = round(angle_vector./range_angle).*
    range_angle;
51 target_angle = mode(angle_vector_round);
52
53 angle_vector = angle_vector(abs(target_angle -
    angle_vector) <= range_angle ); % Descartar
    improvavei
54
55 choose_angle = median(angle_vector); % Calcular angulo

```



```

        provavel
56
57     return_struct = { ...
58         choose_angle ...
59         Rho ...
60         ant_array ...
61         Z_phase_array ...
62         Z_x_array ...
63         delta_A_x_B ...
64         delta_B_x_A ...
65     };
66
67 end %function

```

A.2.3 Função de geração saída visual

Código 21: Arquivo de código `generate_fig.m`.

```

1  function generate_fig( ...
2      z_plot, ...
3      x_w, ...
4      y_w, ...
5      ang_w, ...
6      lambda_w, ...
7      interval, ...
8      Rho, ...
9      choose_angle, ...
10     ant_array, ...
11     Z_phase_array, ...
12     Z_x_array, ...
13     delta_A_x_B_array, ...
14     delta_B_x_A_array ...
15 ) % Graficos
16
17     index_list = 1:length(ant_array);
18
19     % Utilizar conjunto de cores diversas para antenas
20     color_list = hsv(length(hsv))(:,:);
21
22     while length(color_list) < length(ant_array)
23         aux_color_list = normalize(color_list+circshift(
24             color_list,1),'range');
25         color_list = sortrows(cat(1,color_list, aux_color_list
26             .*0.75));
27     end %while
28
29     % Garantir maior contraste entre cores de antenas
30     color_index = floor(index_list * (length(color_list)/
31         length(ant_array)));

```

```

29
30 AoA_x = cos(choose_angle);
31 AoA_y = sin(choose_angle);
32
33 DoA_x = cos(ang_w);
34 DoA_y = sin(ang_w);
35
36 set(0, 'defaultlinelinerwidth', 0.5);
37 set(0, 'defaultlinemarker size', 5);
38 set(0, 'defaultlinemarkerfacecolor', 'auto');
39
40 clf;
41
42 % Calcular a figura de fundo para visualizacao em imagem
43
44 subplot(1,2,2, 'align');
45 surf(x_w,y_w,z_plot); % Desenhar sinal eletromagnetico
46 view(-45, 45)
47
48 colormap('gray');
49 hold on;
50 % Desenhar antenas
51 for idx = index_list
52     ant = ant_array(idx);
53     color = color_list(color_index(idx),:);
54     plot3(real(ant)*[1 1], imag(ant)*[1 1], [-lambda_w
55         lambda_w], '-p', 'color', color);
56 end % for
57
58 plot3([0 lambda_w]*AoA_x, [0 lambda_w]*AoA_y, [
59     lambda_w lambda_w],':k');
60 plot3(lambda_w*AoA_x, lambda_w*AoA_y, lambda_w,'hk');
61
62 % Angulo de chegada real
63 plot3(DoA_x*lambda_w, DoA_y*lambda_w, lambda_w, 'ok',
64     'markerfacecolor', 'none', 'markersize', 10, '
65     linewidth', 1);
66
67 shading interp;
68 axis([-interval interval -interval interval -interval
69     interval],'square')
70 caxis([-lambda_w lambda_w]);
71 hold off;
72
73 subplot(1,2,1, 'align');
74
75 interval_2d = 1.25*lambda_w;
76 axis([-interval_2d interval_2d -interval_2d interval_2d

```

```

        ], 'square')
72
73     if isoctave()
74         polar(0,0, ':w')
75     else
76         polarplot(0,0, ':w')
77     end % if
78
79     hold on;
80     if isoctave()
81         set( gca, 'rtick', [ 0 : 0.25 : 1 ] );
82         set( gca, 'ttick', [ 0 : 15 : 359 ] );
83     else % MATLAB
84         rticks([ 0 : 0.25 : 1 ]);
85         thetaticks([ 0 : 15 : 359 ]);
86     end %if
87
88     grid on;
89     grid minor;
90
91     for idx = index_list
92         idx_next = idx+1;
93         if idx_next > length(ant_array)
94             idx_next = 1;
95         end %if
96         color = color_list(color_index(idx),:);
97         color_next = color_list(color_index(idx_next),:);
98         part_ang_r = 8 + (12-8)*(idx-1)/(length(ant_array)-1);
99
100        % Antenas
101        ant = ant_array(idx);
102        ant_plot_aux = ant/(8 * Rho);
103        if isoctave()
104            plot(real(ant_plot_aux), imag(ant_plot_aux), 'p', '
105                color', color);
106        else % MATLAB
107            polarplot(angle(ant_plot_aux), abs(ant_plot_aux), 'p
108                ', 'color', color);
109        end % if
110
111        % Fase por antenna
112        Z_phase = Z_phase_array(idx);
113        Z_phase_plot_aux = Z_phase/abs(Z_phase)*(4/16);
114        if isoctave()
115            plot(real(Z_phase_plot_aux), imag(Z_phase_plot_aux),
116                'o', 'color', color);
117        else % MATLAB
118            polarplot(angle(Z_phase_plot_aux), abs(

```

```

116         Z_phase_plot_aux), 'o', 'color', color);
117     end % if
118 % Eixos auxiliares
119 ant_next = ant_array(idx_next);
120 ant_axis = ant_next - ant;
121 if isoctave()
122     plot(real(ant_axis)*[1 -1]/abs(ant_axis), imag(
123         ant_axis)*[1 -1]/abs(ant_axis), '-', 'color',
124         color);
125     plot(real(ant_axis*i)*[1 -1]/abs(ant_axis), imag(
126         ant_axis*i)*[1 -1]/abs(ant_axis), ':', 'color',
127         color);
128 else % MATLAB
129     polarplot(angle(ant_axis), [1 -1], '-', 'color',
130         color);
131     polarplot(angle(ant_axis*i), [1 -1], ':', 'color',
132         color);
133 end % if
134
135 % Defasagem entre antenas
136 Z_A_x_B = Z_x_array(idx);
137 aux_A_x_B = (Z_A_x_B/abs(Z_A_x_B))*(6/16);
138 if isoctave()
139     plot(real(aux_A_x_B), imag(aux_A_x_B), 'd', 'color',
140         color, 'markerfacecolor', color_next, 'linewidth',
141         0.75);
142 else % MATLAB
143     polarplot(angle(aux_A_x_B), abs(aux_A_x_B), 'd', '
144         color', color, 'markerfacecolor', color_next, '
145         linewidth', 0.75);
146 end % if
147
148 % Angulos de chegada parciais
149 delta_A_x_B = delta_A_x_B_array(idx);
150 delta_B_x_A = delta_B_x_A_array(idx);
151 if isoctave()
152     plot(cos(delta_A_x_B)*(part_ang_r/16), sin(
153         delta_A_x_B)*(part_ang_r/16), 'v', 'color', color
154         , 'markerfacecolor', color_next, 'linewidth',
155         0.75);
156     plot(cos(delta_B_x_A)*(part_ang_r/16), sin(
157         delta_B_x_A)*(part_ang_r/16), '^', 'color', color
158         , 'markerfacecolor', color_next, 'linewidth',
159         0.75);
160 else % MATLAB
161     polarplot(delta_A_x_B, part_ang_r/16, 'v', 'color',
162         color, 'markerfacecolor', color_next, 'linewidth'

```

```

    , 0.75);
146     polarplot(delta_B_x_A, part_ang_r/16, '^', 'color',
        color, 'markerfacecolor', color_next, 'linewidth'
        , 0.75);
147     end % if
148 end % for
149
150 if isoctave()
151     % Angulo de chegada calculado
152     plot([0 7/8]*AoA_x, [0 7/8]*AoA_y, ':k');
153     plot(7/8*AoA_x, 7/8*AoA_y, 'hk', 'markersize', 10, '
        linewidth', 1);
154     cplx_aux_AoA = i*exp(i*choose_angle);
155     plot(real(cplx_aux_AoA)*[1 -1], imag(cplx_aux_AoA)*[1
        -1], '-k');
156
157     % Angulo de chegada real
158     plot(DoA_x*7/8, DoA_y*7/8, 'ok', 'markerfacecolor', '
        none', 'markersize', 15, 'linewidth', 1);
159 else % MATLAB
160     % Angulo de chegada calculado
161     polarplot([choose_angle choose_angle], [0 7/8], ':k');
162     polarplot(choose_angle, 7/8, 'hk', 'markersize', 10, '
        linewidth', 1);
163     cplx_aux_AoA = choose_angle + pi/2;
164     polarplot([cplx_aux_AoA cplx_aux_AoA], [1 -1], '-k');
165
166     % Angulo de chegada real
167     polarplot(ang_w, 7/8, 'ok', 'markerfacecolor', 'none',
        'markersize', 15, 'linewidth', 1);
168 end % if
169
170 hold off;
171
172 end % function

```

A.2.4 Função geral da simulação

Código 22: Arquivo de código `w_xyt.m`.

```

1 function file_adress = w_xyt( ...
2     NOISE, ...
3     ATT, ...
4     CHG_PHI, ...
5     CHG_R, ...
6     CHG_THETA, ...
7     S_GIF, ...
8     S_DAT, ...
9     SNR, ...

```

```

10 range_step, ...
11 N_antenas ...
12 )
13
14 function ang_norm = normalize_angle(ang)
15     ang_norm = ang;
16     if ang > pi
17         ang_norm = ang_norm - (2*pi);
18     end % if
19     if ang < -pi
20         ang_norm = ang_norm + (2*pi);
21     end % if
22 end % function
23
24 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
25
26 DEBUG = false;
27
28 USE_GEOMETRIC = true;
29 USE_MSC_UA = true;
30 USE_MSC_UCA = true;
31 USE_GN = false;
32
33 % Evitar processar dados que nao serao salvos
34 USE_MSC_UA = (S_DAT && USE_MSC_UA);
35 USE_MSC_UCA = (S_DAT && USE_MSC_UCA);
36 USE_GN = (S_DAT && USE_GN);
37
38 CALC_R2 = true;
39
40 SNR_dB = 10*log10(SNR);
41
42 phase = 0;
43
44 R_upper = 50;
45 R_lower = 1;
46
47 range_shift = 0;
48
49 DoA_range = (0+range_shift):range_step:(360+range_shift
50     -1);
51 DoA = deg2rad(DoA_range);
52
53 limits = 2; % -+ * Lambda
54
55 c = 1; % Velocidade da luz, simplificacao
56 lambda = 4;
57 omega = 2*pi*c/lambda;

```

```

57     d = lambda / 2;
58     T = 2 * pi / omega;
59
60     amp_0 = 1;
61
62     C = true;
63     S = true;
64
65     interval = limits*lambda;
66     resolution = 100;
67
68     space = linspace(-interval,interval,resolution+1); % 1-
        by-100
69     [x, y] = meshgrid(space);
70     t_0 = 0;
71
72     ant_idx_list = 1:1:N_antenas;
73     ant_idx_list_shift = circshift(ant_idx_list, 1);
74
75     name = 'simul';
76     folder = '';
77
78     name = [name '_POLY_' num2str(N_antenas)];
79     folder = [folder 'POLY_' num2str(N_antenas)];
80
81     if CHG_R
82         name = [name '_R_' num2str(R_upper) '~' num2str(
            R_lower) ];
83     else
84         name = [name '_R_' num2str(R_upper)];
85     end %if
86     if ~CHG_THETA
87         name = [name '_FIXED_W'];
88     end %if
89     if NOISE
90         name = [name '_SNR_' num2str(SNR)];
91     end %if
92     if ATT
93         name = [name '_ATT'];
94     end %if
95
96     if (S_GIF || S_DAT)
97         foldername = fullfile('Output', folder);
98         if not(isfolder(foldername))
99             mkdir(foldername);
100         end %if
101         if S_GIF
102             gif_filename = fullfile(foldername, [name '.gif']);

```

```

103     end %if
104     if S_DAT
105         dat_filename = fullfile(foldername, [name '.dat']);
106     end %if
107     if (S_DAT && CALC_R2)
108         dat_r2_filename = fullfile(foldername, [name '.r2.
109             dat']);
110     end %if
111 end %if
112 if USE_GEOMETRIC
113     if S_GIF
114         if isoctave()
115             f = figure(1, 'name', name, 'visible', 'off', '
116                 Position', [1 1 1000 500]);
117             else % MATLAB
118                 f = figure('name', name, 'visible', 'off', '
119                     Position', [1 1 1000 500]);
120             end % if
121         else
122             f = figure('name', name);
123         end %if
124     end %if
125     if S_DAT
126         dat_file = fopen(dat_filename, 'w');
127         fprintf(dat_file, '%s', 'percent');
128         fprintf(dat_file, '\t%s', 'ang_W');
129         fprintf(dat_file, '\t%s', 'r');
130         fprintf(dat_file, '\t%s', 'phase');
131         if USE_GEOMETRIC
132             fprintf(dat_file, '\t%s', 'choose_angle_GEOMETRIC');
133         end % if
134         if USE_MSC_ULA
135             fprintf(dat_file, '\t%s', 'choose_angle_MSC_ULA');
136         end % if
137         if USE_MSC_UCA
138             fprintf(dat_file, '\t%s', 'choose_angle_MSC_UCA');
139         end % if
140         if USE_GN
141             fprintf(dat_file, '\t%s', 'choose_angle_GN');
142         end % if
143         if USE_GEOMETRIC
144             for i = ant_idx_list
145                 fprintf(dat_file, '\t%s%d_x_%d', 'delta_', i,
146                     ant_idx_list_shift(i));
147             end % for

```



```

147     for i = ant_idx_list
148         fprintf(dat_file, '\t%s%d_x_%d', 'delta_',
149             ant_idx_list_shift(i), i);
150     end % for
151 end % if
152 fprintf(dat_file, '\n');
153 if isoctave()
154     fflush(dat_file);
155 end %if
156 end % if
157 if (S_DAT && CALC_R2)
158     angles_DOA = [];
159     if USE_GEOMETRIC
160         angles_GEOMETRIC = [];
161     end % if
162     if USE_MSC_ULA
163         angles_MSC_ULA = [];
164     end % if
165     if USE_MSC_UCA
166         angles_MSC_UCA = [];
167     end % if
168     if USE_GN
169         angles_GN = [];
170     end % if
171 end % if
172
173 DoA_loop_range = [DoA_range DoA_range 360]; % Duas
174     voltas
175
176 percent = 0;
177 ref_iteration = 1/(length(DoA_loop_range)-1);
178 it = 0;
179 DelayTime = 15*ref_iteration;
180 r = R_upper+R_lower;
181
182 for DoA = DoA_loop_range
183     it = it + 1;
184
185     fprintf('%0.2f%% -> %s\n', percent*100, name);
186
187     if CHG_PHI
188         phase = phase + 2*pi*ref_iteration;
189     end %if
190
191     if CHG_R
192         r = r - R_upper/(2*length(DoA_range));
193     end %if

```

```

193
194     if CHG_THETA
195         ang_W = deg2rad(DoA);
196     else
197         ang_W = pi*5/12;
198     end %if
199
200     % ang = normalize_angle(ang_W)
201     if (S_DAT && CALC_R2)
202         angles_DOA = [angles_DOA; ang_W];
203     end % if
204
205     if USE_GEOMETRIC
206         return_struct = calc_AoA_geometric(amp_0, ang_W, r,
207             phase, lambda, ...
208             omega, S, C, NOISE, SNR_dB, ATT, resolution, d ,
209             N_antenas);
210
211         [ ...
212             choose_angle_GEOMETRIC, ...
213             Rho, ...
214             ant_array, ...
215             Z_phase_array, ...
216             Z_x_array, ...
217             delta_A_x_B, ...
218             delta_B_x_A, ...
219         ] = return_struct{:};
220         if (S_DAT && CALC_R2)
221             angles_GEOMETRIC = [angles_GEOMETRIC;
222                 choose_angle_GEOMETRIC];
223         end % if
224     end % if
225
226     if USE_MSC_ULA
227         return_struct = calc_AoA_music_ULA(amp_0, ang_W, r,
228             phase, lambda, ...
229             omega, S, C, NOISE, SNR_dB, ATT, resolution, d ,
230             N_antenas);
231
232         [ ...
233             choose_angle_MSC_ULA, ...
234         ] = return_struct{:};
235         if (S_DAT && CALC_R2)
236             angles_MSC_ULA = [angles_MSC_ULA;
237                 choose_angle_MSC_ULA];
238         end % if
239     end % if
240
241

```

```

235     if USE_MSC_UCA
236         return_struct = calc_AoA_music_UCA(amp_0, ang_W, r,
            phase, lambda, ...
237         omega, S, C, NOISE, SNR_dB, ATT, resolution, d ,
            N_antenas);
238
239         [ ...
240         choose_angle_MSC_UCA, ...
241     ] = return_struct{:};
242     if (S_DAT && CALC_R2)
243         angles_MSC_UCA = [angles_MSC_UCA;
            choose_angle_MSC_UCA];
244     end % if
245 end % if
246
247     if USE_GN
248         return_struct = calc_AoA_gauss_newton(amp_0, ang_W,
            r, phase, lambda, ...
249         omega, S, C, NOISE, SNR_dB, ATT, resolution, d ,
            N_antenas);
250
251         [ ...
252         choose_angle_GN, ...
253     ] = return_struct{:};
254     if (S_DAT && CALC_R2)
255         angles_GN = [angles_GN; choose_angle_GN];
256     end % if
257 end % if
258
259     if USE_GEOMETRIC
260         % Calcular a figura de fundo para visualizacao em
            imagem
261         z_plot = signal_r(x, y, t_0, amp_0, ang_W, r, phase,
            ...
262         lambda, omega, S, C, NOISE, SNR_dB, ATT);
263
264         generate_fig( ...
265             z_plot, ...
266             x, ...
267             y, ...
268             ang_W, ...
269             lambda, ...
270             interval, ...
271             Rho, ...
272             choose_angle_GEOMETRIC, ...
273             ant_array, ...
274             Z_phase_array, ...
275             Z_x_array, ...

```

```

276     delta_A_x_B, ...
277     delta_B_x_A ...
278 );
279
280     drawnow;
281
282     if S_GIF
283         frame = getframe(f);
284         im = frame2im(frame);
285
286         [imind, cm] = rgb2ind(im);
287         % [imind, cm] = rgb2ind(RGB, im); % MATLAB
288         if it == 1
289             % Create GIF file
290             if isoctave()
291                 imwrite(imind, cm, gif_filename, 'gif', '
                    DelayTime', DelayTime, 'Compression', '
                    lzw');
292             else % MATLAB
293                 imwrite(imind, cm, gif_filename, 'gif', '
                    DelayTime', DelayTime);
294             end % if
295         else
296             % Add each new plot to GIF
297             if isoctave()
298                 imwrite(imind, cm, gif_filename, 'gif', '
                    WriteMode','append','DelayTime', DelayTime
                    , 'Compression', 'lzw');
299             else
300                 imwrite(imind, cm, gif_filename, 'gif', '
                    WriteMode','append','DelayTime', DelayTime)
                    ; % MATLAB
301             end % if
302         end %if
303     else
304         % pause(1/30)
305         pause(0.0001)
306     end %if
307
308 end % if
309
310 if S_DAT
311     fprintf(dat_file, '%.2f', percent*100);
312     fprintf(dat_file, '\t%.3f', normalize_angle(ang_W));
313     fprintf(dat_file, '\t%.3f', r);
314     fprintf(dat_file, '\t%.3f', normalize_angle(phase));
315     if USE_GEOMETRIC
316         fprintf(dat_file, '\t%.3f', normalize_angle(

```

```

        choose_angle_GEOMETRIC));
317     end % if
318     if USE_MSC_ULA
319         fprintf(dat_file, '\t%.3f', normalize_angle(
            choose_angle_MSC_ULA));
320     end % if
321     if USE_MSC_UCA
322         fprintf(dat_file, '\t%.3f', normalize_angle(
            choose_angle_MSC_UCA));
323     end % if
324     if USE_GN
325         fprintf(dat_file, '\t%.3f', normalize_angle(
            choose_angle_GN));
326     end % if
327
328     if USE_GEOMETRIC
329         for i = ant_idx_list
330             fprintf(dat_file, '\t%.3f', normalize_angle(
                delta_A_x_B(i)));
331         end % for
332         for i = ant_idx_list
333             fprintf(dat_file, '\t%.3f', normalize_angle(
                delta_B_x_A(i)));
334         end % for
335     end % if
336
337     fprintf(dat_file, '\n');
338     if isoctave()
339         fflush(dat_file);
340     end %if
341 end % if
342
343 percent = percent + ref_iteration;
344
345 end %for
346
347 if (S_DAT && CALC_R2)
348     % Coefficient of determination
349     if (USE_MSC_ULA || USE_GN)
350         abs_angles_DOA = arrayfun(@(ang) abs(normalize_angle(
            ang)), angles_DOA);
351     end % if
352     if USE_GEOMETRIC
353         r2_GEOMETRIC = (corr(angles_DOA, angles_GEOMETRIC))
            ^2;
354     end % if
355     if USE_MSC_ULA
356         r2_MSC_ULA = (corr(((abs_angles_DOA))), (

```

```

        angles_MSC_ULA)))^2;
357     end % if
358     if USE_MSC_UCA
359         r2_MSC_UCA = (corr(angles_DOA, angles_MSC_UCA))^2;
360     end % if
361     if USE_GN
362         r2_GN = (corr(abs_angles_DOA, angles_GN))^2;
363     end % if
364 end % if
365
366 fprintf('\n');
367
368 if S_GIF
369     fprintf('Check: %s\a\n', gif_filename);
370 end %if
371
372 if S_DAT
373     fclose(dat_file);
374     fprintf('Check: %s\a\n', dat_filename);
375 end %if
376
377 if (S_DAT && CALC_R2)
378     dat_r2_file = fopen(dat_r2_filename, 'w');
379     if USE_GEOMETRIC
380         fprintf(dat_r2_file, '%s\t', 'GEOMETRIC');
381     end % if
382     if USE_MSC_ULA
383         fprintf(dat_r2_file, '%s\t', 'MSC_ULA');
384     end % if
385     if USE_MSC_UCA
386         fprintf(dat_r2_file, '%s\t', 'MSC_UCA');
387     end % if
388     if USE_GN
389         fprintf(dat_r2_file, '%s\t', 'GN');
390     end % if
391     fprintf(dat_r2_file, '\n');
392
393     if USE_GEOMETRIC
394         fprintf(dat_r2_file, '%.2f\t', 100*r2_GEOMETRIC);
395     end % if
396     if USE_MSC_ULA
397         fprintf(dat_r2_file, '%.2f\t', 100*r2_MSC_ULA);
398     end % if
399     if USE_MSC_UCA
400         fprintf(dat_r2_file, '%.2f\t', 100*r2_MSC_UCA);
401     end % if
402     if USE_GN
403         fprintf(dat_r2_file, '%.2f\t', 100*r2_GN);

```

```

404     end % if
405     fprintf(dat_r2_file, '\n');
406
407     fclose(dat_r2_file);
408     fprintf('Check: %s\a\n', dat_r2_filename);
409 end %if
410
411 fprintf('\n%s\n\n', repelem('-',70));
412
413 file_address = { ...
414     foldername, ...
415     name, ...
416 };
417
418 end %function

```

A.2.5 Função do MUSIC

Código 23: Arquivo de código `calc_AoA_music_ULA.m`.

```

1  function return_struct = calc_AoA_music_ULA ( ...
2      amp_w, ...
3      ang_w, ...
4      r_w, ...
5      phase_w, ...
6      lambda_w, ...
7      omega_w, ...
8      S, ...
9      C, ...
10     NOISE, ...
11     SNR_dB, ...
12     ATT, ...
13     resolution, ...
14     d, ...
15     N_antenas ...
16 )
17
18 choose_angle = nan;
19 D_signals = 1; % Quantidade de sinais incidentes
20 N_samples = 10; % Quantidade de amostras de sinal
21
22 % Coordenadas das antenas
23 ant_array = complex(d.*(0:N_antenas-1)', 0);
24
25 t = linspace(0,(2 * pi / omega_w), resolution); %
    Intervalo de integração
26
27 Z_phase_matrix = [];
28

```

```

29 for i = 1:N_samples
30     % Calculos de fase
31     Z_phase_array = arrayfun(@(ant) phase_z(t, ant, amp_w,
32         ...
33         ang_w, r_w, phase_w, lambda_w, omega_w, S, C, NOISE,
34         ...
35         SNR_dB, ATT), ant_array);
36
37     Z_phase_matrix = [Z_phase_matrix Z_phase_array];
38 end % for
39
40 AoA_range = deg2rad(0:0.1:180);
41
42 steering_arguments = { ...
43     d, ...
44     N_antenas, ...
45     lambda_w ...
46 };
47
48 choose_angle = music_algorithm(Z_phase_matrix, D_signals
49     , AoA_range, @ steering_linear, steering_arguments);
50
51 return_struct = { ...
52     choose_angle, ...
53 };
54
55 end %function

```

Código 24: Arquivo de código `calc_AoA_music_UCA.m`.

```

1 function return_struct = calc_AoA_music_UCA ( ...
2     amp_w, ...
3     ang_w, ...
4     r_w, ...
5     phase_w, ...
6     lambda_w, ...
7     omega_w, ...
8     S, ...
9     C, ...
10    NOISE, ...
11    SNR_dB, ...
12    ATT, ...
13    resolution, ...
14    d, ...
15    N_antenas ...
16 )
17
18 choose_angle = nan;
19 D_signals = 1; % Quantidade de sinais incidentes

```



```

20 N_samples = 10; % Quantidade de amostras de sinal
21
22 % Raio do circulo de circunscribe o poligono com
    N_samples lados de tamanho d
23 Rho = d/(2*sin(pi / N_antenas));
24 ant_angles_shift = -90;
25 ant_angles = deg2rad((0 + ant_angles_shift):(360/
    N_antenas):(359 + ant_angles_shift))';
26
27 % Coordenadas das antenas
28 ant_array = Rho * exp(i * ant_angles);
29
30 t = linspace(0,(2 * pi / omega_w), resolution); %
    Intervalo de integração
31
32 Z_phase_matrix = [];
33
34 for i = 1:N_samples
35     % Calculos de fase
36     Z_phase_array = arrayfun(@(ant) phase_z(t, ant, amp_w,
        ...
37         ang_w, r_w, phase_w, lambda_w, omega_w, S, C, NOISE,
        ...
38         SNR_dB, ATT), ant_array);
39
40     Z_phase_matrix = [Z_phase_matrix Z_phase_array];
41 end % for
42
43 AoA_range = deg2rad(0:0.1:360);
44
45 steering_arguments = { ...
46     Rho, ...
47     ant_angles, ...
48     lambda_w ...
49 };
50
51 choose_angle = music_algorithm(Z_phase_matrix, D_signals
    , AoA_range, @ steering_circular, steering_arguments)
    ;
52
53 return_struct = { ...
54     choose_angle, ...
55 };
56
57 end %function

```

Código 25: Arquivo de código `music_algorithm.m`.

```

1 function doa_estimates = music_algorithm( ...

```

```

2   Z_phase_matrix, ...
3   D_signals, ...
4   AoA_range, ...
5   steering_function, ...
6   steering_arguments ...
7 )
8
9 % Compute the MUSIC spectrum
10 [Vn, Pmusic] = music_spectrum(Z_phase_matrix, D_signals,
    AoA_range, ...
11     steering_function, steering_arguments);
12
13 % Find the initial estimates of DoAs
14 [~, doa_indices] = sort(Pmusic, 'descend');
15
16 % If more than D_signals peaks are found, select the
    D_signals largest ones
17 if length(doa_indices) > D_signals
18     doa_indices = doa_indices(1:D_signals);
19 end
20
21 initial_DoAs = AoA_range(doa_indices);
22 doa_estimates = initial_DoAs;
23
24 % % Refine the DoAs using fminsearch
25 % doa_estimates = zeros(1, D_signals);
26 % for i = 1:length(initial_DoAs)
27 %     error_function = @(steering_angle) abs(1 / abs( ...
28 %         (steering_function(steering_angle,
29 %             steering_arguments)' * Vn) * ...
30 %         (Vn' * steering_function(steering_angle,
31 %             steering_arguments)) ...
32 %         ) - Pmusic(doa_indices(i))));
33 %     doa_estimates(i) = fminsearch(error_function,
34 %         initial_DoAs(i));
35 % end
36 % % If less than D_signals estimates were found, fill
    the remaining ones with NaN
37 % if length(doa_estimates) < D_signals
38 %     doa_estimates = [doa_estimates, nan(1, D_signals -
39 %         length(doa_estimates))];
40 % end
41 end

```

Código 26: Archivo de código `music_spectrum.m`.

```

1 function [Vn, Pmusic] = music_spectrum( ...
2     Z_phase_matrix, ...
3     D_signals, ...

```

```

4   AoA_range, ...
5   steering_function, ...
6   steering_arguments ...
7   )
8
9   Rxx = (Z_phase_matrix * Z_phase_matrix') / size(
10      Z_phase_matrix, 2);
11   [V, Dia] = eig(Rxx);
12   [~, idx] = sort(diag(Dia), 'descend');
13   V = V(:, idx);
14   Vn = V(:, D_signals+1:end);
15   % Calculate the MUSIC spectrum
16
17   Pmusic = arrayfun(@(steering_angle) (1 / abs( ...
18      (steering_function(steering_angle, steering_arguments)
19      ' * Vn) * ...
20      (Vn' * steering_function(steering_angle,
21      steering_arguments)) ...
22      )), AoA_range);
23   end

```

Código 27: Archivo de código `steering_linear.m`.

```

1   function steering_vector = steering_linear( ...
2       steering_angle, ...
3       steering_arguments ...
4   )
5
6   [ ...
7       d, ...
8       N_antenas, ...
9       lambda_w ...
10  ] = steering_arguments{:};
11
12  % Frank Gross - Smart Antennas for Wireless
13  % Communications, pg. 69
14  steering_vector = exp(1i*d*(0:N_antenas-1)'*cos(
15      steering_angle)*2*pi/lambda_w);
16  end

```

Código 28: Archivo de código `steering_circular.m`.

```

1   function steering_vector = steering_circular( ...
2       steering_angle, ...
3       steering_arguments ...
4   )
5
6   [ ...
7       Rho, ...

```

```

8     ant_angles, ...
9     lambda_w ...
10 ] = steering_arguments{:};
11
12 % Frank Gross - Smart Antennas for Wireless
13 % Communications, pg. 90
14 steering_vector = exp(1i*Rho*cos(steering_angle-
    ant_angles)*2*pi/lambda_w);
15 end

```

A.2.6 Arquivos de simulação em sequência

Código 29: Arquivo de código `settings.m`.

```

1 function settings_struct = settings()
2
3     range_step = 5;
4
5     N_antenas = 7;
6
7     settings_struct = { ...
8         range_step, ...
9         N_antenas, ...
10    };
11
12 end % function

```

Código 30: Arquivo de código `w_xyt_single.m`.

```

1 clear all;
2 close all;
3 clc;
4
5 if isoctave()
6     pkg load communications;
7     pkg load statistics;
8 end %if
9
10 w_xyt( ...
11     false, ... % NOISE
12     false, ... % ATT
13     false, ... % CHG_PHI
14     false, ... % CHG_R
15     true, ... % CHG_THETA
16     true, ... % S_GIF
17     true, ... % S_DAT
18     1/1, ... % SNR
19     36*2, ... % range_step
20     9 ... % N_antenas

```

21 |);

Código 31: Archivo de código `w_xyt_dat.m`.

```
1  clear all;
2  close all;
3  clc;
4
5  if isoctave()
6      pkg load communications;
7      pkg load statistics;
8  end %if
9
10 S_GIF = true;
11 S_DAT = true;
12
13 settings_struct = settings();
14 [ ...
15     range_step, ...
16     N_antenas, ...
17 ] = settings_struct{:};
18
19 % w_xyt(NOISE, ATT, CHG_PHI, CHG_R, CHG_THETA, S_GIF,
20     S_DAT, SNR, range_step, N_antenas);
21
22 % % w_xyt(false, false, true, false, false, S_GIF, S_DAT,
23     1/1, range_step, N_antenas);
24 w_xyt(false, false, false, false, true, S_GIF, S_DAT,
25     1/1, range_step, N_antenas);
26 % % w_xyt(false, false, false, true, true, S_GIF, S_DAT,
27     1/1, range_step, N_antenas);
28
29 % % w_xyt(false, true, true, false, false, S_GIF, S_DAT,
30     1/1, range_step, N_antenas);
31 w_xyt(false, true, false, false, true, S_GIF, S_DAT,
32     1/1, range_step, N_antenas);
33 % % w_xyt(false, true, false, true, true, S_GIF, S_DAT,
34     1/1, range_step, N_antenas);
35
36 % % w_xyt(true, false, true, false, false, S_GIF, S_DAT,
37     1/1, range_step, N_antenas);
38 % % w_xyt(true, false, true, false, false, S_GIF, S_DAT,
39     5/1, range_step, N_antenas);
40 % % w_xyt(true, false, true, false, false, S_GIF, S_DAT,
41     25/1, range_step, N_antenas);
42 % % w_xyt(true, false, true, false, false, S_GIF, S_DAT,
43     50/1, range_step, N_antenas);
44 % % w_xyt(true, false, true, false, false, S_GIF, S_DAT,
45     100/1, range_step, N_antenas);
```

```

34
35 % % w_xyt(true, true, true, false, false, S_GIF, S_DAT,
    1/1, range_step, N_antenas);
36 % % w_xyt(true, true, true, false, false, S_GIF, S_DAT,
    5/1, range_step, N_antenas);
37 % % w_xyt(true, true, true, false, false, S_GIF, S_DAT,
    25/1, range_step, N_antenas);
38 % % w_xyt(true, true, true, false, false, S_GIF, S_DAT,
    50/1, range_step, N_antenas);
39 % % w_xyt(true, true, true, false, false, S_GIF, S_DAT,
    100/1, range_step, N_antenas);
40
41 % % w_xyt(true, false, false, true, true, S_GIF, S_DAT,
    1/1, range_step, N_antenas);
42 % % w_xyt(true, false, false, true, true, S_GIF, S_DAT,
    5/1, range_step, N_antenas);
43 % % w_xyt(true, false, false, true, true, S_GIF, S_DAT,
    25/1, range_step, N_antenas);
44 % % w_xyt(true, false, false, true, true, S_GIF, S_DAT,
    50/1, range_step, N_antenas);
45 % % w_xyt(true, false, false, true, true, S_GIF, S_DAT,
    100/1, range_step, N_antenas);
46
47 % % w_xyt(true, true, false, true, true, S_GIF, S_DAT,
    1/1, range_step, N_antenas);
48 % % w_xyt(true, true, false, true, true, S_GIF, S_DAT,
    5/1, range_step, N_antenas);
49 % % w_xyt(true, true, false, true, true, S_GIF, S_DAT,
    25/1, range_step, N_antenas);
50 % % w_xyt(true, true, false, true, true, S_GIF, S_DAT,
    50/1, range_step, N_antenas);
51 % % w_xyt(true, true, false, true, true, S_GIF, S_DAT,
    100/1, range_step, N_antenas);
52
53 w_xyt(true, false, false, false, true, S_GIF, S_DAT,
    1/1, range_step, N_antenas);
54 w_xyt(true, false, false, false, true, S_GIF, S_DAT,
    5/1, range_step, N_antenas);
55 w_xyt(true, false, false, false, true, S_GIF, S_DAT,
    25/1, range_step, N_antenas);
56 w_xyt(true, false, false, false, true, S_GIF, S_DAT,
    50/1, range_step, N_antenas);
57 w_xyt(true, false, false, false, true, S_GIF, S_DAT,
    100/1, range_step, N_antenas);
58
59 w_xyt(true, true, false, false, true, S_GIF, S_DAT,
    1/1, range_step, N_antenas);
60 w_xyt(true, true, false, false, true, S_GIF, S_DAT,

```

```

5/1, range_step, N_antenas);
61 w_xyt(true, true, false, false, true, S_GIF, S_DAT,
    25/1, range_step, N_antenas);
62 w_xyt(true, true, false, false, true, S_GIF, S_DAT,
    50/1, range_step, N_antenas);
63 w_xyt(true, true, false, false, true, S_GIF, S_DAT,
    100/1, range_step, N_antenas);

```

Código 32: Archivo de código `w_xyt_auto.m`.

```

1 clear all;
2 close all;
3 clc;
4
5 if isoctave()
6     pkg load communications;
7     pkg load statistics;
8 end %if
9
10 S_GIF = true;
11 S_DAT = false;
12
13 settings_struct = settings();
14 [ ...
15     range_step, ...
16     N_antenas, ...
17 ] = settings_struct{:};
18
19 % w_xyt(NOISE, ATT, CHG_PHI, CHG_R, CHG_THETA, S_GIF,
    S_DAT, SNR, range_step, N_antenas);
20
21 w_xyt(false, false, true, false, false, S_GIF, S_DAT,
    1/1, range_step, N_antenas);
22 % % w_xyt(false, false, false, false, true, S_GIF, S_DAT,
    1/1, range_step, N_antenas);
23 w_xyt(false, false, false, true, true, S_GIF, S_DAT,
    1/1, range_step, N_antenas);
24
25 w_xyt(false, true, true, false, false, S_GIF, S_DAT,
    1/1, range_step, N_antenas);
26 % % w_xyt(false, true, false, false, true, S_GIF, S_DAT,
    1/1, range_step, N_antenas);
27 w_xyt(false, true, false, true, true, S_GIF, S_DAT,
    1/1, range_step, N_antenas);
28
29 w_xyt(true, false, true, false, false, S_GIF, S_DAT,
    1/1, range_step, N_antenas);
30 w_xyt(true, false, true, false, false, S_GIF, S_DAT,
    5/1, range_step, N_antenas);

```

```

31 w_xyt(true, false, true, false, false, S_GIF, S_DAT,
    25/1, range_step, N_antenas);
32 w_xyt(true, false, true, false, false, S_GIF, S_DAT,
    50/1, range_step, N_antenas);
33 w_xyt(true, false, true, false, false, S_GIF, S_DAT,
    100/1, range_step, N_antenas);
34
35 w_xyt(true, true, true, false, false, S_GIF, S_DAT,
    1/1, range_step, N_antenas);
36 w_xyt(true, true, true, false, false, S_GIF, S_DAT,
    5/1, range_step, N_antenas);
37 w_xyt(true, true, true, false, false, S_GIF, S_DAT,
    25/1, range_step, N_antenas);
38 w_xyt(true, true, true, false, false, S_GIF, S_DAT,
    50/1, range_step, N_antenas);
39 w_xyt(true, true, true, false, false, S_GIF, S_DAT,
    100/1, range_step, N_antenas);
40
41 w_xyt(true, false, false, true, true, S_GIF, S_DAT,
    1/1, range_step, N_antenas);
42 w_xyt(true, false, false, true, true, S_GIF, S_DAT,
    5/1, range_step, N_antenas);
43 w_xyt(true, false, false, true, true, S_GIF, S_DAT,
    25/1, range_step, N_antenas);
44 w_xyt(true, false, false, true, true, S_GIF, S_DAT,
    50/1, range_step, N_antenas);
45 w_xyt(true, false, false, true, true, S_GIF, S_DAT,
    100/1, range_step, N_antenas);
46
47 w_xyt(true, true, false, true, true, S_GIF, S_DAT,
    1/1, range_step, N_antenas);
48 w_xyt(true, true, false, true, true, S_GIF, S_DAT,
    5/1, range_step, N_antenas);
49 w_xyt(true, true, false, true, true, S_GIF, S_DAT,
    25/1, range_step, N_antenas);
50 w_xyt(true, true, false, true, true, S_GIF, S_DAT,
    50/1, range_step, N_antenas);
51 w_xyt(true, true, false, true, true, S_GIF, S_DAT,
    100/1, range_step, N_antenas);
52
53 % w_xyt(true, false, false, false, true, S_GIF, S_DAT,
    1/1, range_step, N_antenas);
54 % w_xyt(true, false, false, false, true, S_GIF, S_DAT,
    5/1, range_step, N_antenas);
55 % w_xyt(true, false, false, false, true, S_GIF, S_DAT,
    25/1, range_step, N_antenas);
56 % w_xyt(true, false, false, false, true, S_GIF, S_DAT,
    50/1, range_step, N_antenas);

```



```

57 % w_xyt(true, false, false, false, true, S_GIF, S_DAT,
    100/1, range_step, N_antenas);
58
59 % w_xyt(true, true, false, false, true, S_GIF, S_DAT,
    1/1, range_step, N_antenas);
60 % w_xyt(true, true, false, false, true, S_GIF, S_DAT,
    5/1, range_step, N_antenas);
61 % w_xyt(true, true, false, false, true, S_GIF, S_DAT,
    25/1, range_step, N_antenas);
62 % w_xyt(true, true, false, false, true, S_GIF, S_DAT,
    50/1, range_step, N_antenas);
63 % w_xyt(true, true, false, false, true, S_GIF, S_DAT,
    100/1, range_step, N_antenas);

```

Código 33: Archivo de código `w_xyt_full_auto.m`.

```

1  clear all;
2  close all;
3  clc;
4
5  if isoctave()
6      pkg load communications;
7      pkg load statistics;
8  end %if
9
10 S_GIF = true;
11 range_step = 5; % degrees
12 N_antenas_LIST = [3 5 7];
13
14 COPY_FILES = true;
15
16 % w_xyt(NOISE, ATT, CHG_PHI, CHG_R, CHG_THETA, S_GIF,
    S_DAT, SNR, range_step, N_antenas);
17
18 for S_DAT = [true false]
19     for N_antenas = N_antenas_LIST
20         if S_DAT
21             CHG_LIST = [true];
22         else
23             CHG_LIST = [false true];
24         end
25         for NOISE = [false true]
26             if NOISE
27                 SNR_LIST = [1/1 5/1 25/1 50/1 100/1];
28             else
29                 SNR_LIST = [0];
30             end %if
31             for ATT = [false true]
32                 for CHG = CHG_LIST

```

```

33     CHG_PHI = not(CHG);
34     CHG_R = xor(CHG, S_DAT);
35     CHG_THETA = CHG;
36     for SNR = SNR_LIST
37         file_address = w_xyt(NOISE, ATT, CHG_PHI, CHG_R
38             , CHG_THETA, S_GIF, S_DAT, SNR, range_step,
39             N_antenas);
40         % fprintf('w_xyt(%d, %d, %d, %d, %d, %d, %d, %d, %d);\n', NOISE, ATT, CHG_PHI, CHG_R,
41             CHG_THETA, S_GIF, S_DAT, SNR, range_step,
42             N_antenas)
43         if (S_DAT && COPY_FILES)
44             copy_to_documentation(N_antenas, file_address
45                 );
46         end %if
47     end % for
48 end % for
49
50     fprintf('\n%s\n', repelem('|',70));
51     fprintf('%s\n', strftime('%Y-%m-%d %H:%M:%S', localtime(
52         time()))))
53 end % for

```