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 coises 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

Lista de Tabelas

Códigos

Abreviaturas e Siglas

AoA	Ângulo de Chegada (*Angle of Arrival*)
ATT	Atenuação
AWGN	Ruído Gaussiano Branco Aditivo (*Additive White Gaussian Noise*)
FFT	Transformada Rápida de Fourier (*Fast Fourier Transform*)
GNSS	Sistema Global de Navegação por Satélite (*Global Navigation Satellite System*)
GPS	Sistema de Posicionamento Global (*Global Positioning System*)
IoT	Internet das Coisas (*Internet of Things*)
LoRa	*Longe Range*
LoS	Linha de visão (*Line of Sight*)
NLoS	Sem Linha de visão (*Non Line of Sight*)
RF	Radiofrequência (*Radio Frequency*)
SDR	Rádio Definido por *Software* (*Software-Defined Radio*)
SNR	Relação Sinal-Ruído (*Signal-Noise Ratio*)
UCA	Matriz Circular Uniforme (*Uniform Circular Array*)
ULA	Matriz Linear Uniforme (*Uniform Linear Array*)

Símbolos e Operadores

$\mathbf{A_g}$	Coordenada geográfica A
A_k	k-ésima antena da malha
$\mathbf{B_g}$	Coordenada geográfica B
c	Velocidad da luz no ar
$D_{\mathbf{ant}}$	Maior dimensão da antena emissora para cálculo da distância de Fraunhofer
d	Distância entre antenas
$d_{\mathbf{F}}$	Distância de Fraunhofer
$d_{\mathbf{AB}}$	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
$\imath$	Unidade imaginária, $\sqrt{-1}$
k	Índice das antenas da malha
$N_{\mathbf{ant}}$	Número de antenas da malha
Q_k	Componente em quadratura do valor complexo do sinal recebido
$R_{\mathbf{Terra}}$	Raio do planeta
T	Período do sinal w de interesse
w	Sinal de interesse para análise, incidente na malha 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 *bearing* relativo
$\Delta\Phi_k$	Diferença de fase em par de antenas A_k e A_{k+1}
Δ_θ	Diferença entre os ângulos $\theta_{\mathbf{A}_\mathrm{g}}$ e $\theta_{\mathbf{B}_\mathrm{g}}$
Δ_ϕ	Diferença entre os ângulos $\phi_{\mathbf{A}_\mathrm{g}}$ e $\phi_{\mathbf{B}_\mathrm{g}}$
δ	Intervalo de quantização e de filtro para valores de Θ

Θ	Conjunto de todos os valores $\theta_{\pm k}$ aferidos
$\Theta_{\lfloor\bullet\rceil}$	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 θ_{AoA} referentes ao par de antenas A_k e A_{k+1}
$\theta_{\mathcal{M}_o}$	Moda estatística do conjunto $\Theta_{\lfloor\bullet\rceil}$
$\theta_{\mathbf{A_g}}$	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 da malha 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
$\lfloor h\rceil$	Operação arredondar, arredonda o valor de h para o inteiro mais próximo

Sumário

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

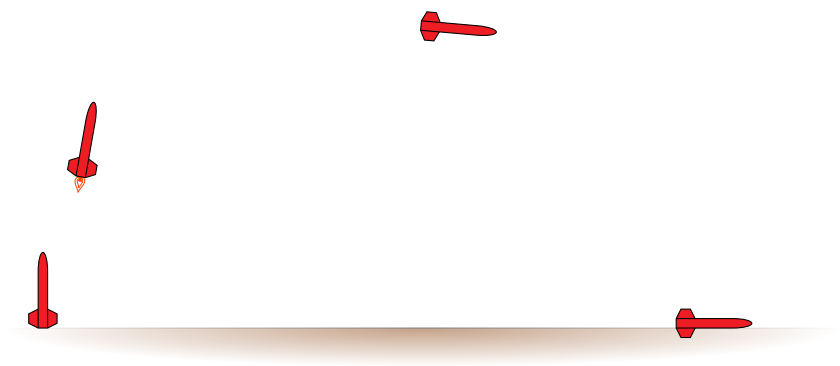
Foguetes de sondagem atmosférica são veículos aeroespaciais sub-orbitais utilizados para levar sensores e experimentos científicos a altos níveis atmosféricos, com o intuito de realizar estudos e análises relacionados às diversas condições ali presentes [1]. Estes veículos geralmente utilizam motor de 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 rastreio, 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 fundamentação teórica e revisão de trabalhos relacionados. O Capítulo 3 apresenta o detalhe da metodologia adotada na construção do trabalho. No Capítulo 4 são apresentados detalhes sobre a performance da 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 a fundamentação teórica utilizada ao longo do trabalho, bem como um breve levantamento de trabalhos relacionados, que mostram a relevância do assunto abordado.

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 $R_{\text{Terra}} = 6371 \cdot 10^3$ 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 $\mathbf{A}_\text{g}$ e $\mathbf{B}_\text{g}$ duas coordenadas geográficas, $\phi_{\mathbf{A}_\text{g}}$ e $\phi_{\mathbf{B}_\text{g}}$ suas respectivas latitudes, e $\theta_{\mathbf{A}_\text{g}}$ e $\theta_{\mathbf{B}_\text{g}}$ suas respectivas longitudes, conforme ilustrado na Figura 2.

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

$$\Delta_\phi = \phi_{\mathbf{B}_\text{g}} - \phi_{\mathbf{A}_\text{g}} \tag{1}$$

$$\Delta_\theta = \theta_{\mathbf{B}_\text{g}} - \theta_{\mathbf{A}_\text{g}} \tag{2}$$

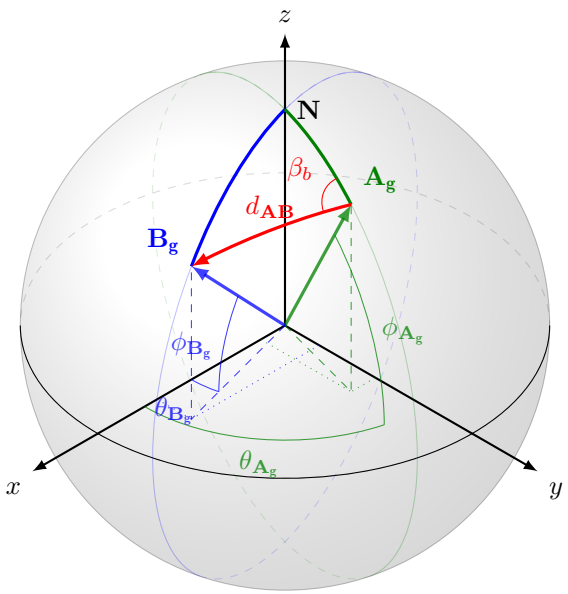
Através da lei dos haversines é possível obter a distância mínima d entre as coordenadas, sobre a superfície, e também o ângulo de *Bearing* β_b formado no vértice $\mathbf{A}_\text{g}$ do triângulo esférico $\mathbf{N}\mathbf{A}_\text{g}\mathbf{B}_\text{g}$ [8]. Para o cálculo de distância, os ângulos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

$$X = \cos\left(\theta_{\mathbf{B}_\text{g}}\right) \cdot \sin\left(\Delta_\phi\right) \tag{3}$$

$$Y = \cos\left(\theta_{\mathbf{A}_\text{g}}\right) \cdot \sin\left(\theta_{\mathbf{B}_\text{g}}\right) - \sin\left(\theta_{\mathbf{A}_\text{g}}\right) \cdot \cos\left(\theta_{\mathbf{B}_\text{g}}\right) \cdot \cos\left(\Delta_\phi\right) \tag{4}$$

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

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



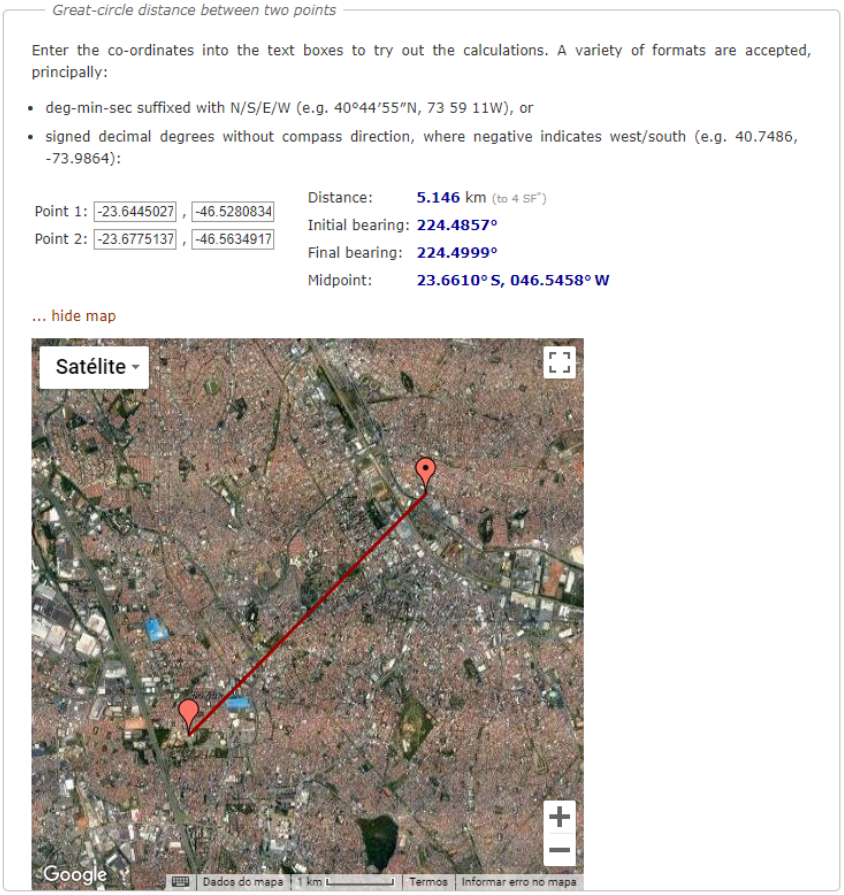
Fonte: Autor.

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

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

O ângulo β_b calculado aqui é referente à direção cardeal 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].

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]

2.1.2 Estimar AoA utilizando malha de antenas

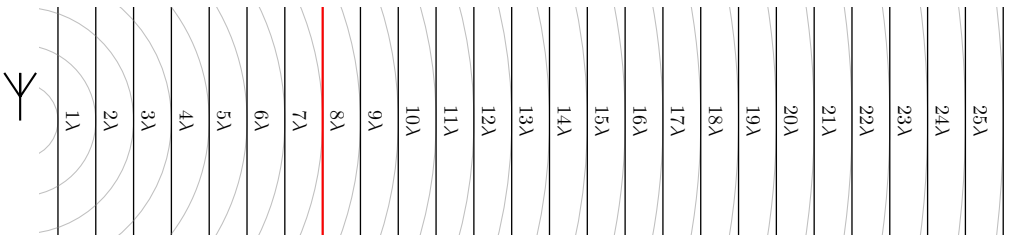
Analisando a defasagem de um sinal de RF incidindo em uma malha de antenas, é possível estimar seu Ângulo de Chegada (*Angle of Arrival*, AoA), ou seja, determinar a direção do emissor do sinal em relação ao sistema. Este valor é calculado utilizando dados como a distância entre as antenas, o comprimento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

$$\lambda = \frac{c}{f} = \frac{2\pi \cdot c}{\omega} \tag{8}$$

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 $D_\mathrm{ant} = 2\lambda$, para uma antena de dipolo, obtém-se $d_\mathrm{F} = 8\lambda$. A Figura 4 ilustra o comportamento planar de uma frente de onda, com destaque na distância d_F.

$$d_\mathrm{F} = \frac{2 \cdot {D_\mathrm{ant}}^2}{\lambda} \quad \Rightarrow \quad d_\mathrm{F} = \frac{2 \cdot (2 \cdot \lambda)^2}{\lambda} = 8\lambda \tag{9}$$

Figura 4: Característica de frente de onda a cada λ a partir da antena emissora, destaque para $d_\mathrm{F} = 8\lambda$.



Fonte: Autor.

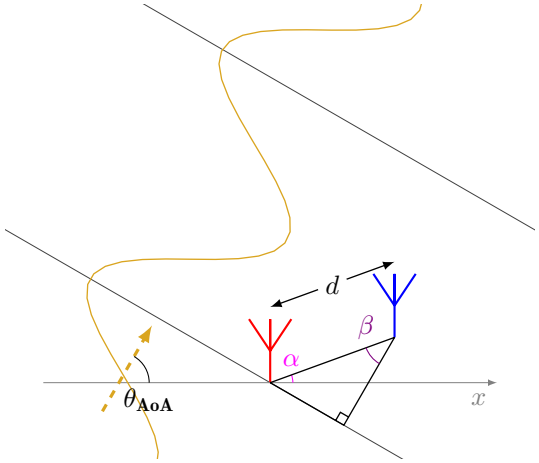
Tomando agora um par de antenas separadas por uma distância fixa d, torna-se viável fazer a análise trigonométrica entre as antenas e a frente de onda incidente, onde essa distância d será a hipotenusa do triângulo retângulo formado. Para realizar esta análise, ainda é necessário conhecer uma segunda 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.

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

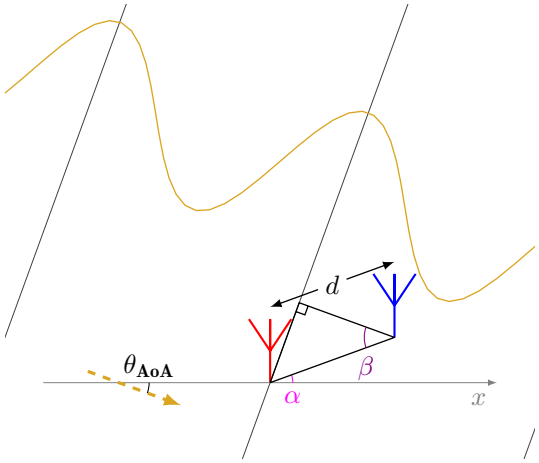
É importante ressaltar que um sistema com um único par de antenas não é suficiente

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

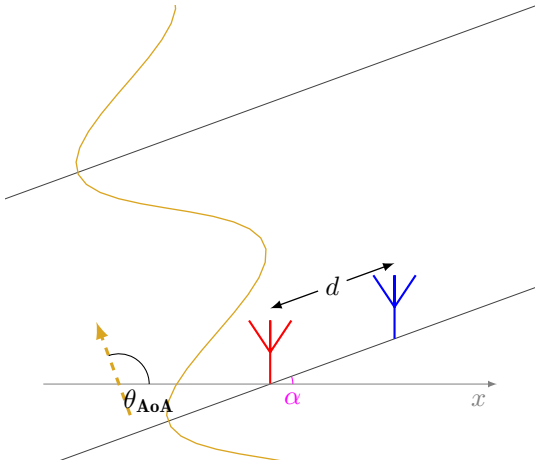
(a) $\theta_{\text{AoA}} = 60°$, $\beta_{\pm k} = 40°$



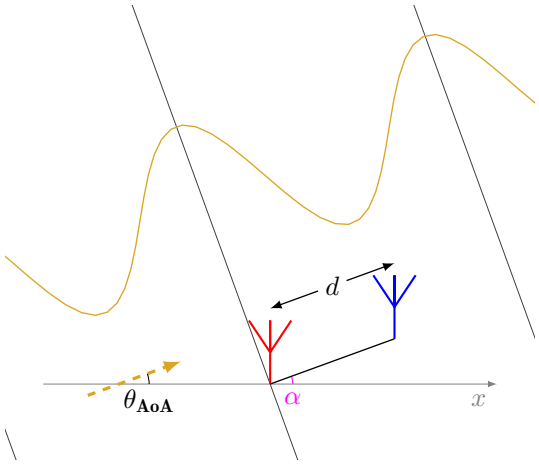
(b) $\theta_{\text{AoA}} = -20°$, $\beta_{\pm k} = 40°$



(c) $\theta_{\text{AoA}} = 110°$, $\beta_{\pm k} = 90°$



(d) $\theta_{\text{AoA}} = 20°$, $\beta_{\pm k} = 0°$



Fonte: Autor.

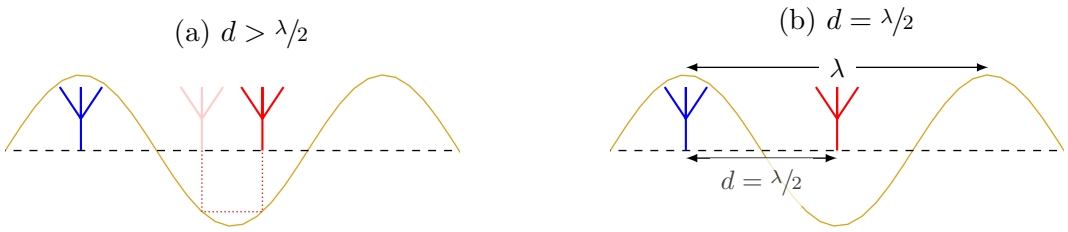
para determinar completamente o θ_{AoA}, já que o valor calculado de $\beta_{\pm k}$ é igual para casos simétricos em relação ao par de antenas, criando um caso de ambiguidade. As Figuras 5a e 5b apresentam exemplos de diferentes valores de θ_{AoA} para o mesmo valor de $\beta_{\pm k}$. Existem ainda 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].

$$d = \frac{\lambda}{2} \tag{11}$$

Para contornar a ambiguidade de simetria, é possível adicionar mais antenas à malha.

Figura 6: Diferentes valores para d.



Fonte: Autor.

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.

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

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.

$$k = \{1, 2, \ldots, N_{\text{ant}}\} \tag{13}$$

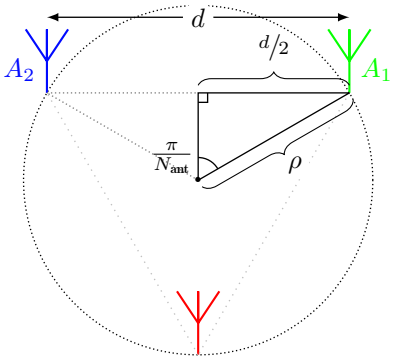
$$A_k = \rho \cdot \exp\left(\imath \cdot k \cdot \frac{2\pi}{N_{\text{ant}}}\right) = (\mathcal{R}e\,(A_k)\,,\ \mathcal{I}m\,(A_k)) = (x_{A_k},\ y_{A_k}) \tag{14}$$

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

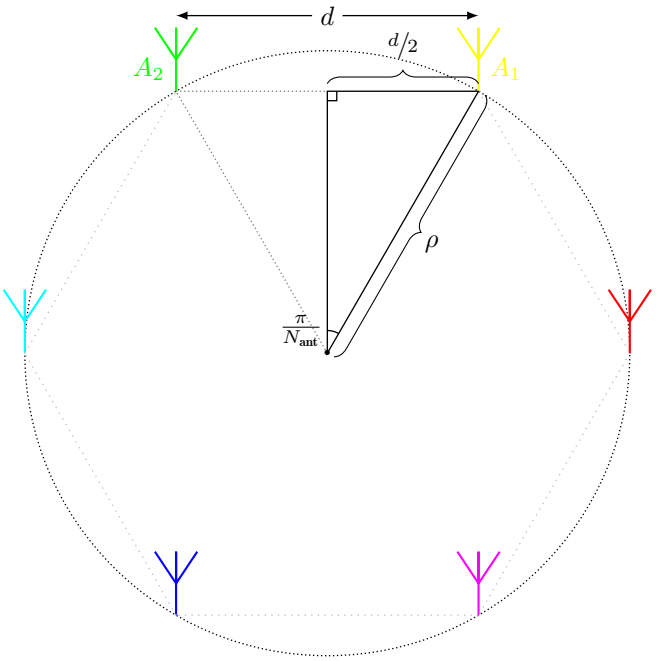
Para calcular a fase em uma antena, é interessante representar o sinal recebido como um valor complexo. Uma forma de obter o complexo de fase consiste em analisar a correlação do sinal incidente w com sinais de referência de mesma frequência que o sinal de interesse, em um período completo, Equação 16. 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.

Figura 7: Diferentes distribuições de antenas.

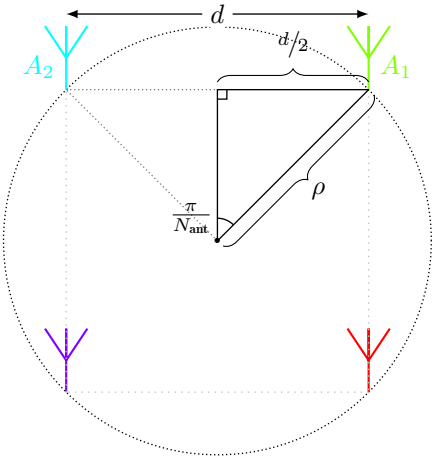
(a) $N_{\text{ant}} = 3$.



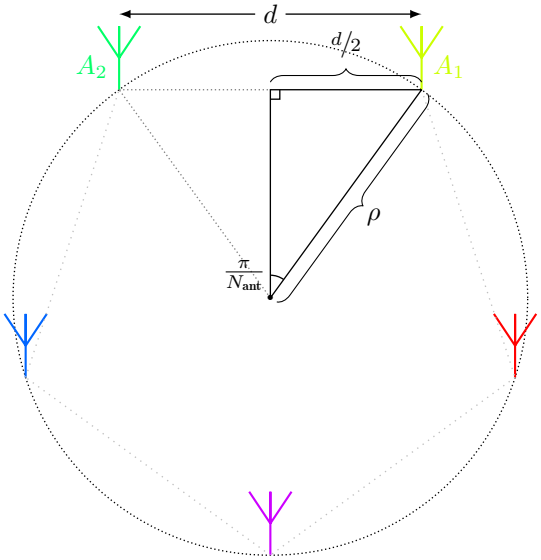
(b) $N_{\text{ant}} = 6$.



(c) $N_{\text{ant}} = 4$.



(d) $N_{\text{ant}} = 5$.



Fonte: Autor.

$$T = \frac{2\pi}{\omega} = \frac{1}{f} \tag{16}$$

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

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

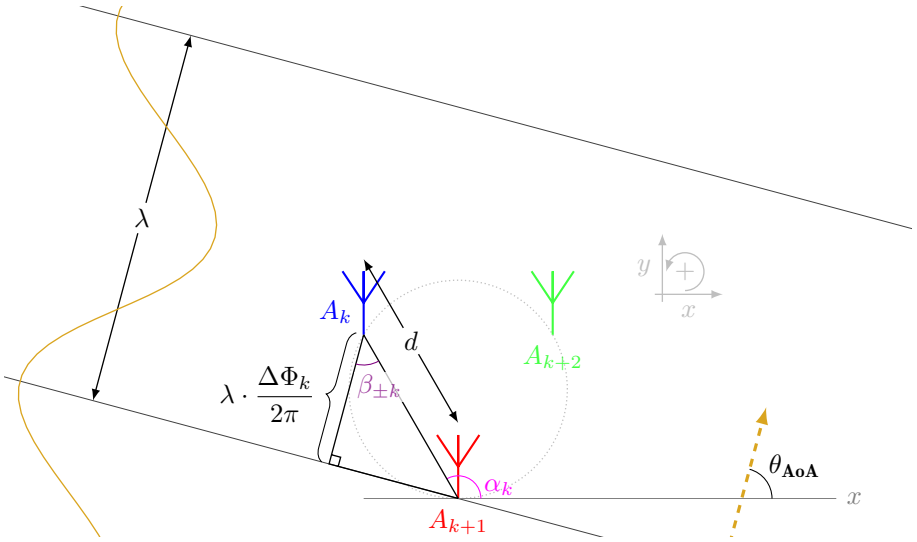
$$Z_k = \frac{\omega}{\pi} \cdot \left(I_k + \imath Q_k\right) \tag{19}$$

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 $N_{\text{ant}} = 3$.

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

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

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.

$$\theta_{\pm k} = \alpha_k \pm \beta_{\pm k} \tag{22}$$

$$\Theta = \{\theta_{\pm k} \mid \forall k\} \tag{23}$$

Com os possíveis valores de θ_{AoA} obtidos, é necessário estimar qual o valor correto. Para isso, é criada uma lista auxiliar $\Theta_{\lfloor\bullet\rceil}$, 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.

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

$$\Theta_{\lfloor\bullet\rceil} = \left\{ \left\lfloor \frac{\theta}{\delta} \right\rceil \cdot \delta \;\middle|\; \forall \theta \in \Theta \right\} \tag{25}$$

Salvo casos com muito ruído, espera-se que alguns valores em $\Theta_{\lfloor\bullet\rceil}$ 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.

$$\theta_{\mathcal{M}_o} = \mathcal{M}_o \left(\Theta_{\lfloor\bullet\rceil} \right) \tag{26}$$

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.

$$\Theta_{\text{F}} = \{\theta \in \Theta \mid \theta_{\mathcal{M}_o} - \delta \leq \theta \leq \theta_{\mathcal{M}_o} + \delta\} \tag{27}$$

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

$$\theta_{\text{AoA}} = \widetilde{\Theta_{\text{F}}} \tag{28}$$

2.2 Trabalhos relacionados

Em seu trabalho, Horst [12] analisa dois algoritmos de detecção de AoA, realizando as análises em ambientes internos e utilizando matrizes 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 Matriz 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 da malha;
- `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 da malha;
- `omega_w`, frequência angular ω;
- `phase_w`, fase ϕ do sinal no emissor;
- `Rho`, raio ρ do polígono que dispõe as antenas na malha;
- `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:

- `ant_array`, coordenadas das antenas da malha;
- `delta_A_x_B_array`, contendo o ângulo $\theta_{\pm k}$ calculado por $\alpha_k + \beta_{\pm k}$ aferido para cada par de antenas da malha;
- `delta_B_x_A_array`, contendo o ângulo $\theta_{\pm k}$ calculado por $\alpha_k - \beta_{\pm k}$ aferido para cada par de antenas da malha;
- `Z_phase_array`, matriz de valores numéricos complexos, contendo o sinal complexo aferido para cada antena da malha;
- `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 na malha;

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`, `lambda_w` e `omega_w`. O Código 1 apresenta uma versão simplificada da função `argument_r` desenvolvida.

Código 1: Função `argument_r`, simplificada.

```
function res = argument_r(...)
  r = r_w * lambda_w;

  x_0 = r * cos(ang_w);
  y_0 = r * sin(ang_w);

  res = (2*pi/lambda_w) * ( sqrt((y_w-y_0).^2 + ...
    (x_w-x_0).^2) ) + omega_w*t_w + phase_w;
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.

Código 2: Função `ref_cos`, simplificada.

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

Fonte: Autor.

Código 3: Função `ref_sin`, simplificada.

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

Fonte: Autor.

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. 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.

Código 4: Função `signal_r`, simplificada.

```
function res = signal_r(...)
  res = 0;
  if S
    res = res + sin( argument_r(...) );
  end %if
  if C
    res = res + cos( argument_r(...) );
  end %if
  if S && C
    res = res / sqrt(2);
  end %if
  if ATT
    %%% Lei de Friis
    G_t = 1; % Ganho Antena Tx
    G_r = 1; % Ganho Antena Rx
    R_t = 1; % Distância do emissor
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    P_t = (amp_w^2)/R_t;
    P_r = P_t * G_t * G_r * (lambda_w / (4 * pi * r_w));
    amp_r = sqrt(P_r * R_t);
    res = res * amp_r;
  else
    res = res * amp_w;
  end %if
  if NOISE
    res = awgn(res, SNR_dB, 'measured');
  end %if
end %function
```

Fonte: Autor.

A última função auxiliar desenvolvida foi `isoctave`, que confere se a corrente simulação está sendo executada no GNU Octave, retornando um valor binário e não recebe qualquer parâmetro. O Código 5 apresenta uma versão simplificada da função `isoctave` desenvolvida.

Código 5: Função `isoctave`, simplificada.

```
function r = isoctave ()
  persistent x;
  if (isempty (x))
    x = exist ('OCTAVE_VERSION', 'builtin');
  end
  r = x;
end
```

Fonte: Autor.

3.1.3 Função de cálculo para AoA

A primeira grande função desenvolvida foi `calc_AoA`, 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 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 6 apresenta uma versão simplificada da função `calc_AoA` desenvolvida.

A subfunção `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 7 apresenta uma versão simplificada da função `phase_z` desenvolvida.

A subfunção `dephase_A_to_B` calcula o valor complexo de defasagem $\Delta\Phi_k$, o ângulo relativo $\beta_{\pm k}$ e o ângulo α_k entre um par de antenas. Seus argumentos são, respectivamente, `Z_phase_A` e `Z_phase_B`. O Código 8 apresenta uma versão simplificada da função `dephase_A_to_B` desenvolvida.

E a subfunção `deltas_A_B` calcula os ângulos $\theta_{\pm k}$ para um par de antenas. Seus argumentos são, respectivamente, `angle_Z_A_x_B`, `Z_antenna_A` e `Z_antenna_B`. O Código 9 apresenta uma versão simplificada da função `deltas_A_B` desenvolvida.

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`. 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

Código 6: Função `calc_AoA`, simplificada.

```
function return_struct = calc_AoA(...)
  Rho = d/(2*sin(pi / N_antenas));
  ant_angles = % ...
  ant_array = Rho * exp(i * deg2rad(ant_angles));

  Z_phase_array = arrayfun(@(a) phase_z(...), ant_array);
  [Z_x_array angle_Z_A_x_B_array] = arrayfun( ...
    @(a, b) dephase_A_to_B(a, b), ... );

  [delta_A_x_B delta_B_x_A] = arrayfun( ...
    @(ang, a, b) deltas_A_B(ang, a, b), ... );

  range_angle = pi/(2*(N_antenas+1));

  angle_vector = [delta_A_x_B delta_B_x_A];
  angle_vector = [...]; % Normalização

  angle_vector_round = ...
    round(angle_vector./range_angle).*range_angle;

  target_angle = mode(angle_vector_round);

  angle_vector = angle_vector(abs(target_angle ...
   - angle_vector) <= range_angle );

  choose_angle = median(angle_vector);

  return_struct = { ...
    choose_angle ...
    Rho ...
    ant_array ...
    Z_phase_array ...
    Z_x_array ...
    delta_A_x_B ...
    delta_B_x_A ...
  };

end %function
```

Fonte: Autor.

Código 7: Função `phase_z`, simplificada.

```
function Z_phase = phase_z(...)
  I_medido = trapz(t, ref_cos(...) .* signal_r(...) );
  Q_medido = trapz(t, ref_sin(...) .* signal_r(...) );

  Z_phase = (omega_w/pi)*(I_medido + i*Q_medido);
end % function
```

Fonte: Autor.

Código 8: Função `dephase_A_to_B`, simplificada.

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

Fonte: Autor.

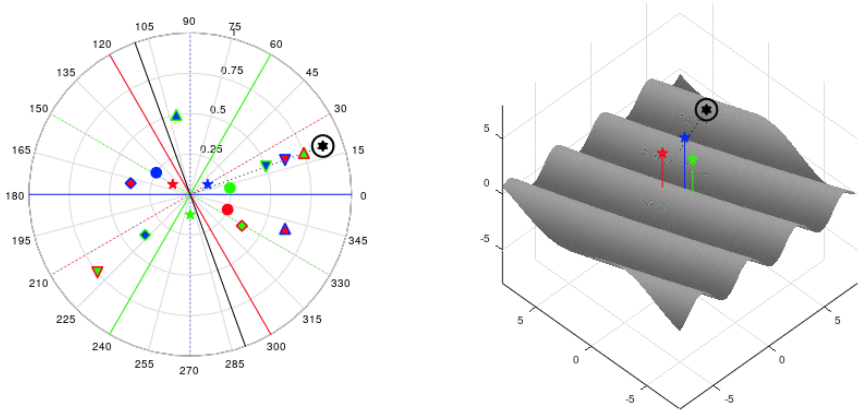
Código 9: Função `deltas_A_B`, simplificada.

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

Fonte: Autor.

Figura 9 ilustra os gráficos gerados pela função `generate_fig`.

Figura 9: Exemplo de quadro da animação de saída da função `generate_fig`.



Fonte: Autor, saída gráfica disponível em GitHub.

3.1.5 Função geral da simulção

Finalmente a função responsável por juntar todas as partes é `w_xyt`, a base para a simulação, ela invoca as funções `calc_AoA` e `generate_fig` com os devidos 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, SNR $\to \infty$ dB) ao caso de potência de ruído igual à potência de sinal (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 %.

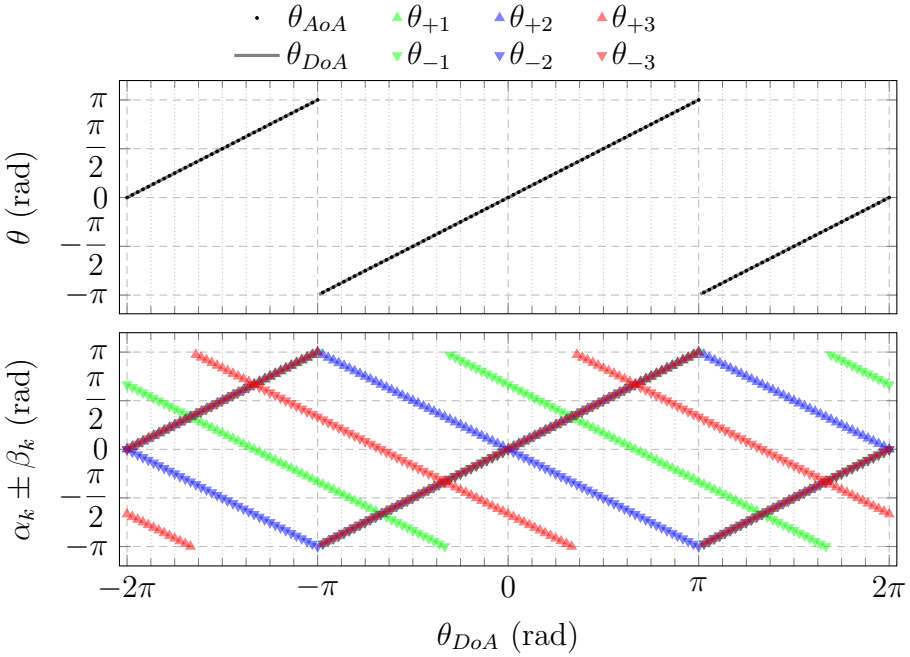
Simulações realizadas na configurações de três antenas têm os valores de R^2 apresentados na Tabela 1, e os resultados dos valores destacados são apresentados nas Figuras 10, 11 e 12.

Tabela 1: Valores de R^2 para simulações notáveis com três antenas.

SNR (dB)	R^2 sem ATT (%)	R^2 com ATT (%)
∞	**100,00**	100,00
20	88,06	90,45
17	88,18	87,98
14	99,99	84,49
7	90,12	83,50
0	**76,32**	**78,73**

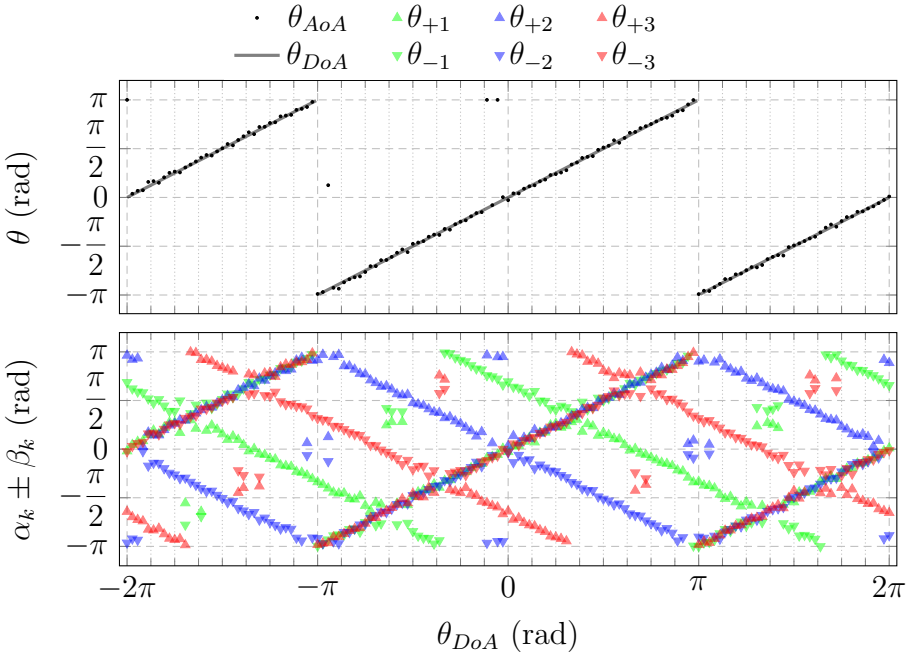
Fonte: Autor, saídas das simulações disponíveis em GitHub.

Figura 10: Simulação para três antenas, caso ideal (SNR $\to \infty$ dB).



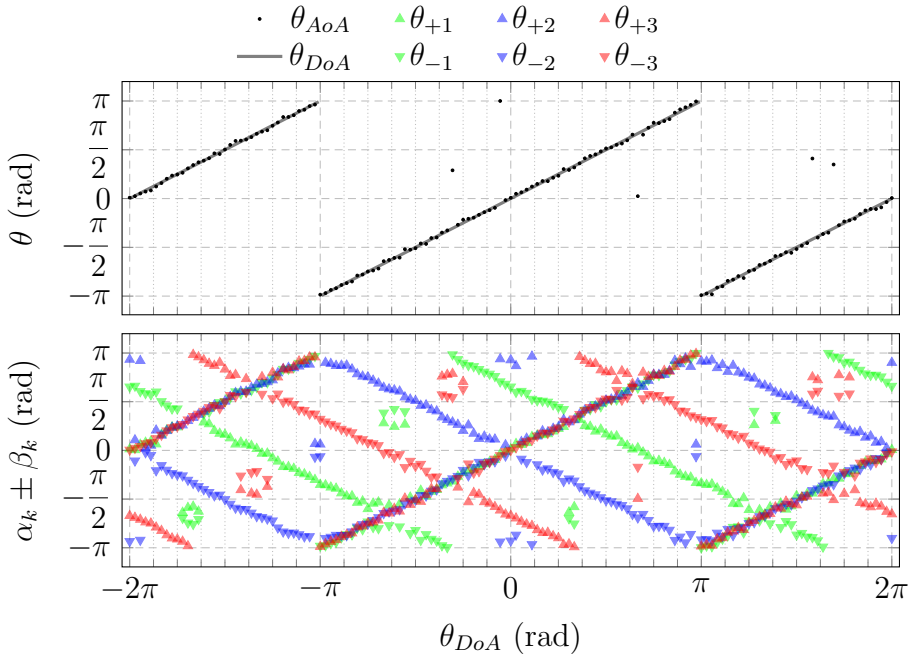
Fonte: Autor, saída gráfica disponível em GitHub.

Figura 11: Simulação para três antenas, caso SNR = 0 dB, sem atenuação.



Fonte: Autor, saída gráfica disponível em GitHub.

Figura 12: 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 80 %.

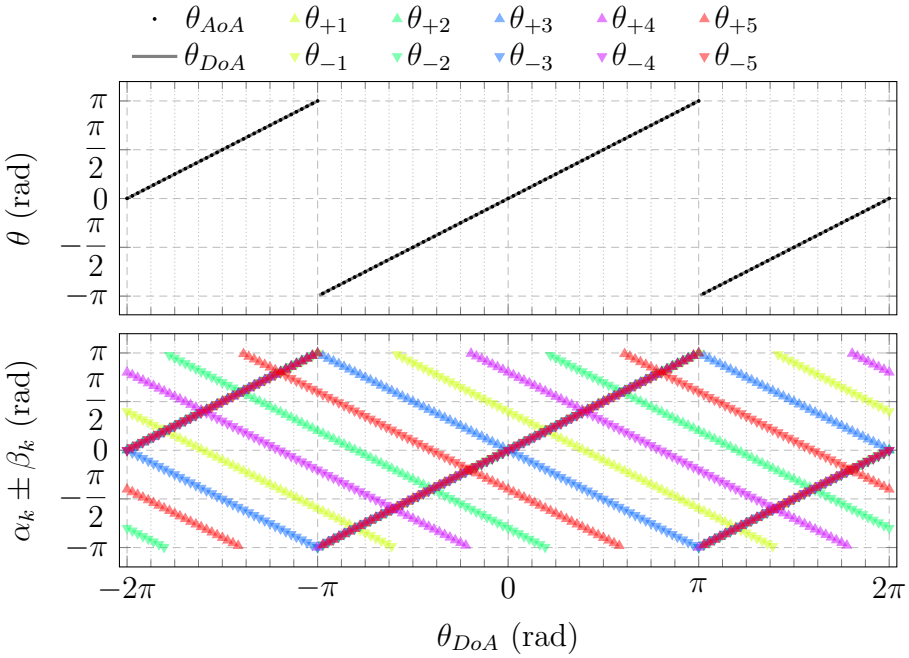
Simulações realizadas na configurações de cinco antenas têm os valores de R^2 apresentados na Tabela 2, e os resultados dos valores destacados são apresentados nas Figuras 13, 14 e 15.

Tabela 2: Valores de R^2 para simulações notáveis com cinco antenas.

SNR (dB)	R^2 sem ATT (%)	R^2 com ATT (%)
∞	**100,00**	100,00
20	100,00	100,00
17	91,90	91,89
14	91,91	91,90
7	84,27	91,93
0	**80,73**	**96,28**

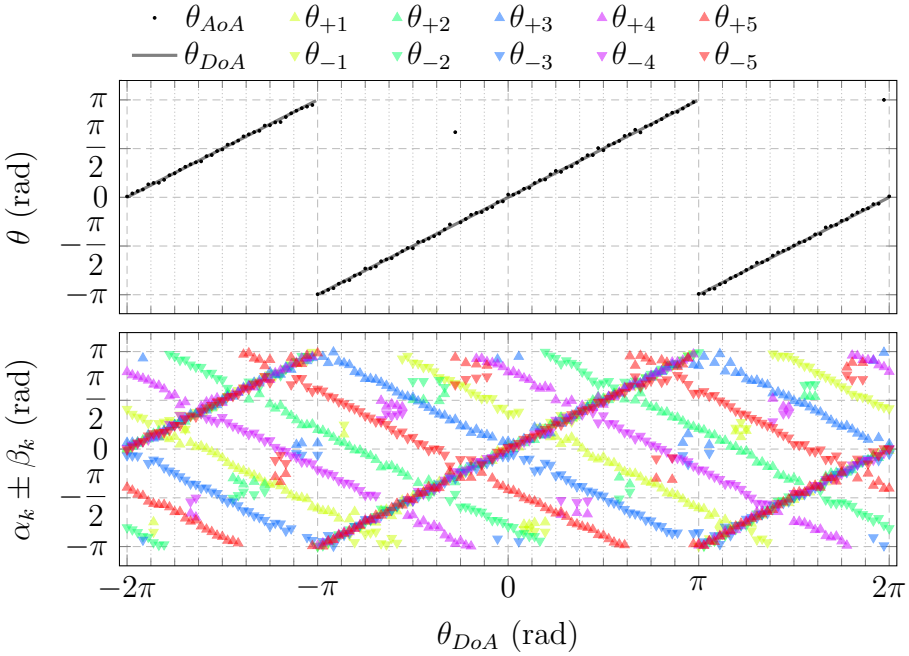
Fonte: Autor, saídas das simulações disponíveis em GitHub.

Figura 13: Simulação para cinco antenas, caso ideal (SNR $\to \infty$ dB).



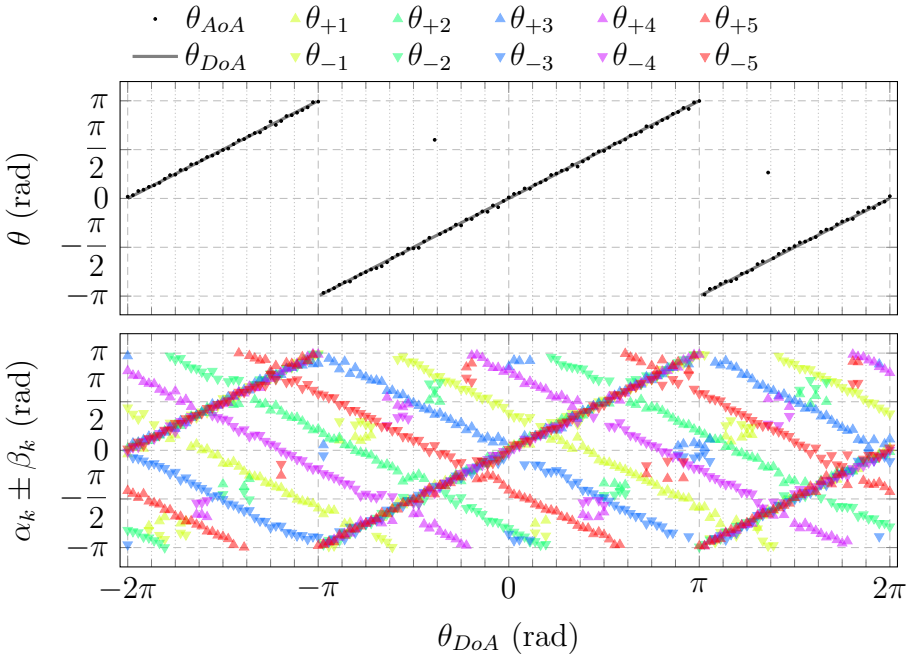
Fonte: Autor, saída gráfica disponível em GitHub.

Figura 14: Simulação para cinco antenas, caso SNR = 0 dB, sem atenuação.



Fonte: Autor, saída gráfica disponível em GitHub.

Figura 15: Simulação para cinco antenas, caso 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 %.

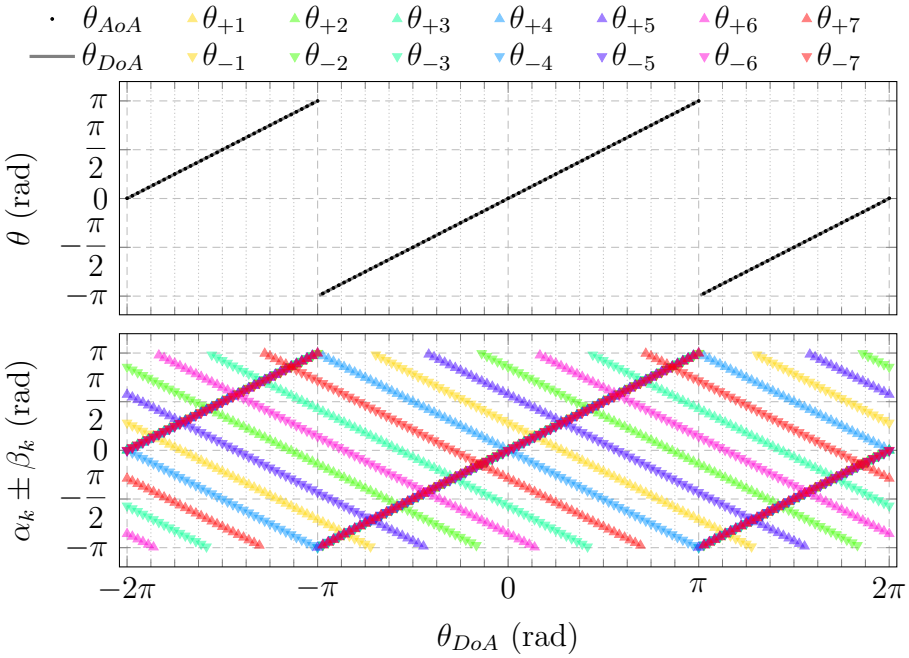
Simulações realizadas na configurações de sete antenas têm os valores de R^2 apresentados na Tabela 3, e os resultados dos valores destacados são apresentados nas Figuras 16, 17 e 18.

Tabela 3: Valores de R^2 para simulações notáveis com sete antenas.

SNR (dB)	R^2 sem ATT (%)	R^2 com ATT (%)
∞	**100,00**	100,00
20	84,25	100,00
17	100,00	84,24
14	91,90	100,00
7	99,99	84,28
0	**80,15**	**99,98**

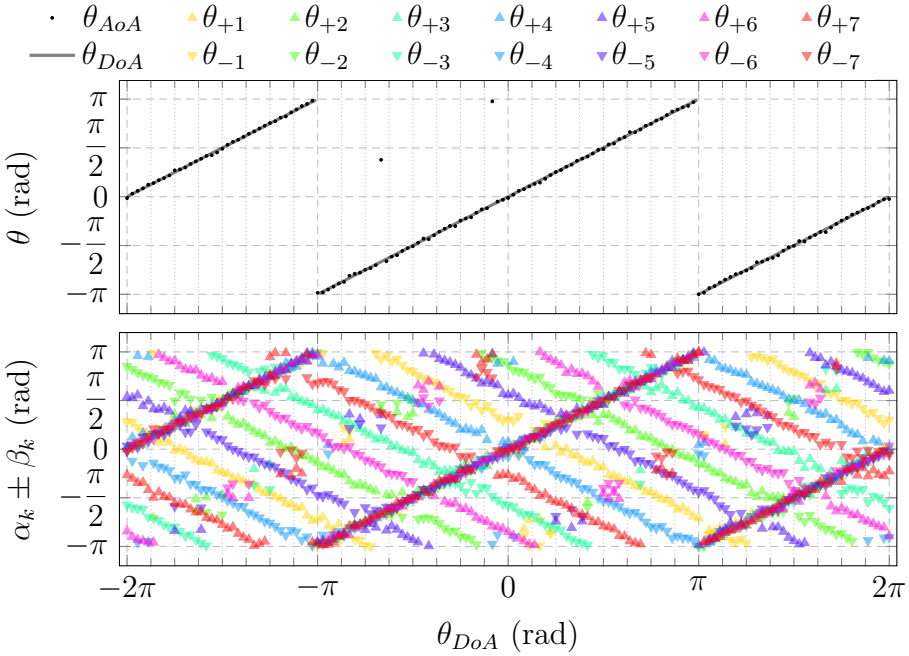
Fonte: Autor, saídas das simulações disponíveis em GitHub.

Figura 16: Simulação para sete antenas, caso ideal (SNR $\to \infty$ dB).



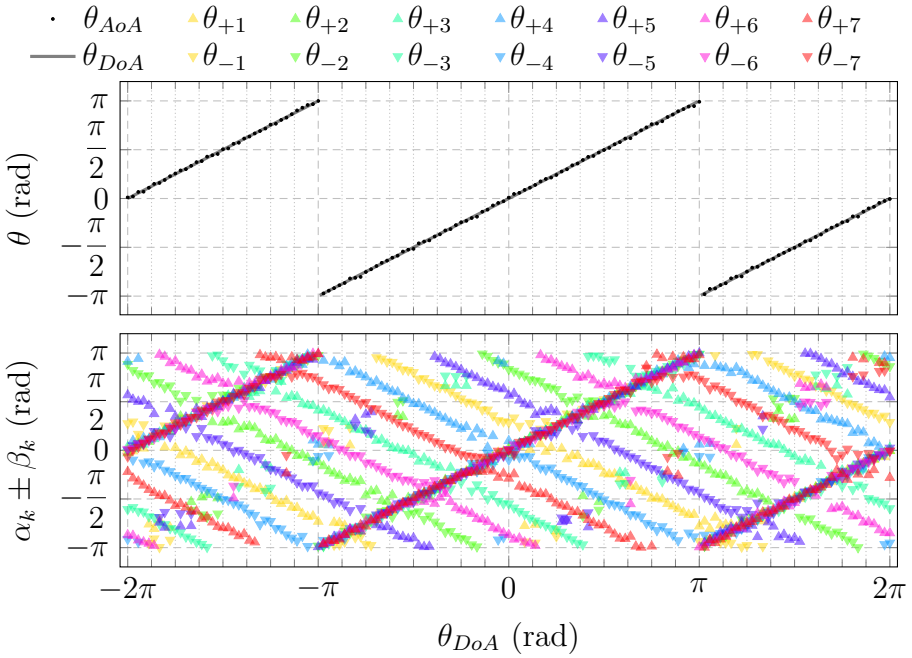
Fonte: Autor, saída gráfica disponível em GitHub.

Figura 17: Simulação para sete antenas, caso SNR = 0 dB, sem atenuação.



Fonte: Autor, saída gráfica disponível em GitHub.

Figura 18: 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^2textsuperscript2 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 ^{2} valor médio de R^2 acima de 92 %.

Para trabalhos futuros, é possível analisar outras disposições de antenas na malha.

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 Positionning", 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`.

```
EARTH_RADIUS = 6371E3;

                  % Lat                 Lon
coord_ufabc_SA  = [-23.64450271006082 -46.52808340455618];
coord_ufabc_SBC = [-23.67751377362596
   -46.563491717644425];

A = coord_ufabc_SA;
B = coord_ufabc_SBC;

theta_A = deg2rad(A(1));
theta_B = deg2rad(B(1));

phi_A = deg2rad(A(2));
phi_B = deg2rad(B(2));

delta_theta = theta_B - theta_A;
delta_phi = phi_B - phi_A;

X = cos(theta_B)*sin(delta_phi);
Y = cos(theta_A)*sin(theta_B) - sin(theta_A)*cos(theta_B)*
   cos(delta_phi);
Z = (sin(delta_theta/2))^2 + cos(theta_B) * cos(theta_A) *
    (sin(delta_phi/2))^2;

beta = atan2(X,Y) - pi/2;

if beta < -pi
  beta = beta + 2*pi
end

d = EARTH_RADIUS * 2 * atan2(sqrt(Z), sqrt(1 - Z))

d_deg = rad2deg(beta)
```

A.2 Simulação de AoA

A.2.1 Funções auxiliares

Código 11: Arquivo de código argument_r.m.

```
function res = argument_r(x_w, y_w, t_w, ang_w, r_w,
   lambda_w, omega_w, phase_w)
  % Argumento da funcao senoidal
  % x_w = coordenada x associada ao ponto da antena
  % y_w = coordenada y associada ao ponto da antena
  % t_w = tempo t associado ao instante de afericao do
     sinal
  % ang_w = angulo theta de chegada do sinal em relacao ao
      sistema de antenas
  % r_w = distância que o emissor de sinal está da
     coordenada (0,0) do sistema
  % lambda_w = comprimento de onda
  % omega_w = frequencia angular
  % phase_w = defasagem do sinal no emissor

  r = r_w * lambda_w;

  x_0 = r * cos(ang_w);
  y_0 = r * sin(ang_w);
  % y_0 = r * 10;

  res = (2*pi/lambda_w) * ( sqrt((y_w-y_0).^2 + (x_w-x_0)
     .^2) ) + omega_w*t_w + phase_w;
end %function
```

Código 12: Arquivo de código ref_cos.m.

```
function c = ref_cos(t_w, omega_w)
  % funcao auxiliar de sinal cosseno
  c = cos(argument_r(0, 0, t_w, 0, 0, 1, omega_w, 0));
end %function
```

Código 13: Arquivo de código ref_sin.m.

```
function s = ref_sin(t_w, omega_w)
  % funcao auxiliar de sinal seno
  s = sin(argument_r(0, 0, t_w, 0, 0, 1, omega_w, 0));
end %function
```

Código 14: Arquivo de código signal_r.m.

```
function res = signal_r(x_w, y_w, t_w, amp_w, ang_w, r_w,
   phase_w, lambda_w, omega_w, S, C, NOISE, SNR_dB, ATT)
% Funcao de sinal senoidal
```

```
% x_w = coordenada x associada ao ponto da antena
% y_w = coordenada y associada ao ponto da antena
% t_w = tempo t associado ao instante de afericao do sinal
% amp_w = amplitude desejada para o sinal
% ang_w = angulo theta de chegada do sinal em relacao ao
    sistema de antenas
% r_w = distancia que o emissor de sinal está da
    coordenada (0,0) do sistema
% phase_w = defasagem phi do sinal
% lambda_w = comprimento de onda
% omega_w = frequencia angular
% S = utilizacao de funcao Seno
% C = utilizacao de funcao Cosseno
% NOISE = se o sinal contara com ruido
% SNR_dB = relacao sinal-ruido
% ATT = se o sinal contara com atenuacao por distancia

  res = 0;
  if S
    res = res + sin( argument_r(x_w, y_w, t_w, ang_w, r_w,
        lambda_w, omega_w, phase_w) );
  end %if
  if C
    res = res + cos( argument_r(x_w, y_w, t_w, ang_w, r_w,
        lambda_w, omega_w, phase_w) );
  end %if
  if S && C
    res = res / sqrt(2);
  end %if

  if ATT
    %%% Lei de Friis
    G_t = 1; % Ganho Antena Tx
    G_r = 1; % Ganho Antena Rx
    R_t = 1; % Distância do emissor
    %%%%%%%%%%%%%%%%%%%%%%%%%

    P_t = (amp_w^2)/R_t;
    P_r = P_t * G_t * G_r * (lambda_w / (4 * pi * r_w));

    amp_r = sqrt(P_r * R_t);
    res = res * amp_r;
  else
    res = res * amp_w;
  end %if

  if NOISE
    res = awgn(res, SNR_dB, 'measured');
```

```
47    end %if
48
49  end %function
```

Código 15: Arquivo de código isoctave.m.

```
1  function r = isoctave() % 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
```

A.2.2 Função de cálculo para AoA

Código 16: Arquivo de código calc_AoA.m.

```
1  function return_struct = calc_AoA( ...
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   function Z_phase = phase_z(t, Z_antenna, amp_w, ang_w,
       r_w, phase_w, lambda_w, omega_w, S, C, NOISE, SNR_dB,
       ATT)
19       % Calculos de amostra I/Q para antena em (0,0)
20     I_medido = trapz(t,ref_cos(t, omega_w) ...
21       .* signal_r(real(Z_antenna), imag(Z_antenna), t,
           amp_w, ang_w, ...
22       r_w, phase_w, lambda_w, omega_w, S, C, NOISE, SNR_dB
           , ATT));
23     Q_medido = trapz(t,ref_sin(t, omega_w) ...
24       .* signal_r(real(Z_antenna), imag(Z_antenna), t,
           amp_w, ang_w, ...
```

```
25      r_w, phase_w, lambda_w, omega_w, S, C, NOISE, SNR_dB
            , ATT));
26
27    Z_phase = (omega_w/pi)*(I_medido + i*Q_medido);
28  end % function
29
30  function [Z_phase_A_x_B angle_Z_A_x_B] = dephase_A_to_B(
        Z_phase_A, Z_phase_B)
31    Z_phase_A_x_B = Z_phase_A * conj(Z_phase_B);
32    deltaPhi_A_x_B = angle(Z_phase_A_x_B);
33    angle_Z_A_x_B = acos(deltaPhi_A_x_B/(pi));
34  end % function
35
36  function [delta_A_x_B delta_B_x_A] = deltas_A_B(
        angle_Z_A_x_B, Z_antenna_A, Z_antenna_B)
37    ang_A_x_B = deg2rad(mod(rad2deg(angle(Z_antenna_A -
          Z_antenna_B)),360));
38    delta_A_x_B = ang_A_x_B + angle_Z_A_x_B;
39    delta_B_x_A = ang_A_x_B - angle_Z_A_x_B;
40  end % function
41
42  % Raio do circulo de circunscreve o poligono com N lados
        de tamanho d
43  Rho = d/(2*sin(pi / N_antenas));
44  ant_angles_shift = -90;
45  ant_angles = (0 + ant_angles_shift):(360/N_antenas):(359
        + ant_angles_shift);
46
47  % Coordenadas das antenas
48  ant_array = Rho * exp(i * deg2rad(ant_angles));
49
50  t = linspace(0,(2 * pi / omega_w), resolution); %
        Intervalo de integração
51
52  % Calculos de fase
53  Z_phase_array = arrayfun(@(a) phase_z(t, a, amp_w, ang_w
        , r_w, phase_w, ...
54    lambda_w, omega_w, S, C, NOISE, SNR_dB, ATT),
          ant_array);
55
56  % Calculos de simetria
57  [Z_x_array angle_Z_A_x_B_array] = arrayfun(@(a, b)
        dephase_A_to_B(a, b), ...
58    Z_phase_array, circshift(Z_phase_array,-1));
59
60  [delta_A_x_B delta_B_x_A] = arrayfun(@(ang, a, b)
        deltas_A_B(ang, a, b), ...
61    angle_Z_A_x_B_array, ant_array, circshift(ant_array
```

```
        ,-1));
62
63    % Fazer "votacao" de angulo escolhido
64    range_angle = pi/(2*(N_antenas+1));
65
66    angle_vector = [delta_A_x_B delta_B_x_A];
67    angle_vector = [...
68      (angle_vector(angle_vector<0)+(2*pi)) ...
69      (angle_vector(0<=angle_vector & angle_vector<=(2*pi)))
          ...
70      (angle_vector((2*pi)<angle_vector)-(2*pi)) ...
71    ]; % Normalizar vetor de angulos entre 0 e 2 pi
72
73    angle_vector_round = round(angle_vector./range_angle).*
        range_angle;
74    target_angle = mode(angle_vector_round);
75
76    angle_vector = angle_vector(abs(target_angle -
        angle_vector) <= range_angle ); % Descartar
        improvavei
77
78    choose_angle = median(angle_vector); % Calcular angulo
        provavel
79
80    return_struct = { ...
81      choose_angle ...
82      Rho ...
83      ant_array ...
84      Z_phase_array ...
85      Z_x_array ...
86      delta_A_x_B ...
87      delta_B_x_A ...
88    };
89
90  end %function
```

A.2.3 Função de geração saída visual

Código 17: 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,...
```

```
    ant_array,...
    Z_phase_array,...
    Z_x_array,...
    delta_A_x_B_array,...
    delta_B_x_A_array...
  ) % Graficos

  index_list = 1:length(ant_array);

  % Utilizar conjunto de cores diversas para antenas
  color_list = hsv(length(hsv))(:,:);
  % if isoctave()
  %   color_list = rainbow(length(rainbow))(:,:);
  % else
  %   color_list = gist_rainbow(length(gist_rainbow))(:,:)
     ;
  % end % if

  % color_list = [...
  %   [1 0 0];
  %   [0 1 0];
  %   [0 0 1]];

  % if length(color_list) < length(ant_array)
  %   color_list = [...
  %     [1 0 0];
  %     [1 1 0];
  %     [0 1 0];
  %     [0 1 1];
  %     [0 0 1];
  %     [1 0 1]];
  % end % if

  while length(color_list) < length(ant_array)
    aux_color_list = normalize(color_list+circshift(
       color_list,1),"range");
    color_list = sortrows(cat(1,color_list, aux_color_list
       .*0.75));
  end %while

  % Garantir maior contraste entre cores de antenas
  % color_index =  floor(linspace(1,length(color_list),
     length(ant_array)));
  color_index = floor(index_list * (length(color_list)/
     length(ant_array)));

  if isoctave()
    AoA_x = cos(choose_angle);
```

```
53      AoA_y = sin(choose_angle);
54
55      DoA_x = cos(ang_w);
56      DoA_y = sin(ang_w);
57    end % if
58
59    set(0, 'defaultlinelinewidth', 0.5);
60    set(0, 'defaultlinemarkersize', 5);
61    set(0, 'defaultlinemarkerfacecolor', 'auto');
62
63    clf;
64
65    % Calcular a figura de fundo para visualizacao em imagem
66
67    subplot(1,2,2, 'align');
68    surf(x_w,y_w,z_plot); % Desenhar sinal eletromagnetico
69    view(-45, 45)
70
71    colormap('gray');
72    hold on;
73      % Desenhar antenas
74      for idx = index_list
75        ant = ant_array(idx);
76        color = color_list(color_index(idx),:);
77        % color = color_list(1 + mod(idx-1, length(
              color_list)),:);
78        plot3(real(ant)*[1 1], imag(ant)*[1 1], [-lambda_w
              lambda_w], '-p', 'color', color);
79      end % for
80
81      plot3([0 lambda_w]*AoA_x, [0 lambda_w]*AoA_y, [
            lambda_w lambda_w],':k');
82      plot3(lambda_w*AoA_x, lambda_w*AoA_y, lambda_w,'hk');
83
84      % Angulo de chegada real
85      plot3(DoA_x*lambda_w, DoA_y*lambda_w, lambda_w, 'ok',
            'markerfacecolor', 'none', 'markersize', 10, '
            linewidth', 1);
86
87      shading interp;
88      axis([-interval interval -interval interval -interval
            interval],'square')
89      caxis([-lambda_w lambda_w]);
90    hold off;
91
92    % set(0, 'CurrentFigure', fig2d)
93    % clf;
94    subplot(1,2,1, 'align');
```

```
95
96  interval_2d = 1.25*lambda_w;
97  axis([-interval_2d interval_2d -interval_2d interval_2d
      ],'square')
98
99  if isoctave()
100   polar(0,0,':w')
101 else
102   polarplot(0,0,':w')
103 end % if
104
105 hold on;
106 if isoctave()
107   set( gca, 'rtick', [ 0 : 0.25 : 1 ] );
108   set( gca, 'ttick', [ 0  : 15 : 359 ] );
109 else % MATLAB
110   rticks([ 0 : 0.25 : 1 ]);
111   thetaticks([ 0  : 15 : 359 ]);
112 end %if
113
114 grid on;
115 grid minor;
116
117 for idx = index_list
118   idx_next = idx+1;
119   if idx_next > length(ant_array)
120     idx_next = 1;
121   end %if
122   % color = color_list(idx);
123   % color_next = color_list(idx_next);
124   % color = color_list(1 + mod(idx-1, length(color_list)
        ),:);
125   % color_next = color_list(1 + mod(idx_next-1, length(
        color_list)),:);
126   % color = color_list(floor((idx/length(ant_array)) *
        length(color_list)),:);
127   % color_next = color_list(floor((idx_next/length(
        ant_array)) * length(color_list)),:);
128   color = color_list(color_index(idx),:);
129   color_next = color_list(color_index(idx_next),:);
130   part_ang_r = 8 + (12-8)*(idx-1)/(length(ant_array)-1);
131
132   % Antenas
133   ant = ant_array(idx);
134   ant_plot_aux = ant/(8 * Rho);
135   if isoctave()
136     plot(real(ant_plot_aux), imag(ant_plot_aux), 'p', '
        color', color);
```

```
137     else % MATLAB
138       polarplot(angle(ant_plot_aux), abs(ant_plot_aux), 'p
          ', 'color', color);
139     end % if
140
141     % Fase por antena
142     Z_phase = Z_phase_array(idx);
143     Z_phase_plot_aux = Z_phase/abs(Z_phase)*(4/16);
144     if isoctave()
145       plot(real(Z_phase_plot_aux), imag(Z_phase_plot_aux),
           'o', 'color', color);
146     else % MATLAB
147       polarplot(angle(Z_phase_plot_aux), abs(
          Z_phase_plot_aux), 'o', 'color', color);
148     end % if
149
150     % Eixos auxiliares
151     ant_next = ant_array(idx_next);
152     ant_axis = ant_next - ant;
153     if isoctave()
154       plot(real(ant_axis)*[1 -1]/abs(ant_axis), imag(
          ant_axis)*[1 -1]/abs(ant_axis), '-', 'color',
          color);
155       plot(real(ant_axis*i)*[1 -1]/abs(ant_axis), imag(
          ant_axis*i)*[1 -1]/abs(ant_axis), ':', 'color',
          color);
156     else % MATLAB
157       polarplot(angle(ant_axis), [1 -1], '-', 'color',
          color);
158       polarplot(angle(ant_axis*i), [1 -1], ':', 'color',
          color);
159     end % if
160
161     % Defasagem entre antenas
162     Z_A_x_B = Z_x_array(idx);
163     aux_A_x_B = (Z_A_x_B/abs(Z_A_x_B))*(6/16);
164     if isoctave()
165       plot(real(aux_A_x_B), imag(aux_A_x_B), 'd', 'color',
           color, 'markerfacecolor', color_next, 'linewidth
          ', 0.75);
166     else % MATLAB
167       polarplot(angle(aux_A_x_B), abs(aux_A_x_B), 'd', '
          color', color, 'markerfacecolor', color_next, '
          linewidth', 0.75);
168     end % if
169
170     % Angulos de chegada parciais
171     delta_A_x_B = delta_A_x_B_array(idx);
```

```
172     delta_B_x_A = delta_B_x_A_array(idx);
173     if isoctave()
174       plot(cos(delta_A_x_B)*(part_ang_r/16), sin(
              delta_A_x_B)*(part_ang_r/16), 'v', 'color', color
              , 'markerfacecolor', color_next, 'linewidth',
              0.75);
175       plot(cos(delta_B_x_A)*(part_ang_r/16), sin(
              delta_B_x_A)*(part_ang_r/16), '^', 'color', color
              , 'markerfacecolor', color_next, 'linewidth',
              0.75);
176     else % MATLAB
177       polarplot(delta_A_x_B, part_ang_r/16, 'v', 'color',
              color, 'markerfacecolor', color_next, 'linewidth'
              , 0.75);
178       polarplot(delta_B_x_A, part_ang_r/16, '^', 'color',
              color, 'markerfacecolor', color_next, 'linewidth'
              , 0.75);
179     end % if
180   end % for
181
182   if isoctave()
183     % Angulo de chegada calculado
184     plot([0 7/8]*AoA_x, [0 7/8]*AoA_y,':k');
185     plot(7/8*AoA_x, 7/8*AoA_y,'hk');
186     cplx_aux_AoA = i*exp(i*choose_angle);
187     plot(real(cplx_aux_AoA)*[1 -1], imag(cplx_aux_AoA)*[1
          -1],'-k');
188
189     % Angulo de chegada real
190     plot(DoA_x*7/8, DoA_y*7/8,'ok', 'markerfacecolor', '
          none', 'markersize', 10, 'linewidth', 1);
191   else % MATLAB
192     % Angulo de chegada calculado
193     plot(choose_angle, [0 7/8],':k');
194     plot(choose_angle, 7/8,'hk');
195     cplx_aux_AoA = i*exp(i*choose_angle);
196     plot(real(cplx_aux_AoA)*[1 -1], imag(cplx_aux_AoA)*[1
          -1],'-k');
197
198     % Angulo de chegada real
199     plot(ang_w, 7/8,'ok', 'markerfacecolor', 'none', '
          markersize', 10, 'linewidth', 1);
200   end % if
201
202   hold off;
203
204 end % function
```

A.2.4 Função geral da simulção

Código 18: Arquivo de código `w_xyt.m`.

```
function w_xyt( ...
  NOISE, ...
  ATT, ...
  CHG_PHI, ...
  CHG_R, ...
  CHG_THETA, ...
  S_GIF, ...
  S_DAT, ...
  SNR, ...
  range_step, ...
  N_antenas ...
  )

  function ang_norm = normalize_angle(ang)
    ang_norm = ang;
    if ang > pi
      ang_norm = ang_norm - (2*pi);
    end % if
    if ang < -pi
      ang_norm = ang_norm + (2*pi);
    end % if
  end % function

  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  DEBUG = false;

  SNR_dB = 10*log10(SNR);

  phase = 0;

  R_upper = 50;
  R_lower = 1;

  range_shift = 0;

  DoA_range = (0+range_shift):range_step:(360+range_shift
    -1);
  DoA = deg2rad(DoA_range);

  limits = 2; % -+ * Lambda

  c = 1; % Velocidade da luz, simplificacao
  lambda = 4;
```

```
44  omega = 2*pi*c/lambda;
45  d = lambda / 2;
46  T = 2 * pi / omega;
47
48  amp_0 = 1;
49
50  C = true;
51  S = true;
52
53  interval = limits*lambda;
54  resolution = 100;
55
56  space = linspace(-interval,interval,resolution+1); % 1-
        by-100
57  [x, y] = meshgrid(space);
58  t_0 = 0;
59
60  ant_idx_list = 1:1:N_antenas;
61  ant_idx_list_shift = circshift(ant_idx_list, 1);
62
63  name = 'simul';
64  folder = '';
65
66  name = [name '_POLY_' num2str(N_antenas)];
67  folder = [folder 'POLY_' num2str(N_antenas)];
68
69  if CHG_R
70    name = [name '_R_' num2str(R_upper) '~' num2str(
        R_lower) ];
71  else
72    name = [name '_R_' num2str(R_upper)];
73  end %if
74  if ~CHG_THETA
75    name = [name '_FIXED_W'];
76  end %if
77  if NOISE
78    name = [name '_SNR_' num2str(SNR)];
79  end %if
80  if ATT
81    name = [name '_ATT'];
82  end %if
83
84  if (S_GIF || S_DAT)
85    foldername = fullfile('Output', folder);
86    if not(isfolder(foldername))
87      mkdir(foldername);
88    end %if
89    if S_GIF
```

```
90        gif_filename = fullfile(foldername, [name '.gif']);
91      end %if
92      if S_DAT
93        dat_filename = fullfile(foldername, [name '.dat']);
94      end %if
95    end %if
96
97    if S_GIF
98      if isoctave()
99        f = figure(1, 'name', name, 'visible', 'off', '
            Position', [1 1 1000 500]);
100     else % MATLAB
101       f = figure('name', name, 'visible', 'off', 'Position
            ', [1 1 1000 500]);
102     end % if
103
104   else
105     f = figure('name', name);
106   end %if
107
108   if S_DAT
109     dat_file = fopen(dat_filename, 'w');
110     fprintf(dat_file, "%s", "percent");
111     fprintf(dat_file, "\t%s", "ang_W");
112     fprintf(dat_file, "\t%s", "r");
113     fprintf(dat_file, "\t%s", "phase");
114     fprintf(dat_file, "\t%s", "choose_angle");
115     for i = ant_idx_list
116       fprintf(dat_file, "\t%s%d_x_%d", "delta_", i,
            ant_idx_list_shift(i));
117     end % for
118     for i = ant_idx_list
119       fprintf(dat_file, "\t%s%d_x_%d", "delta_",
            ant_idx_list_shift(i), i);
120     end % for
121     fprintf(dat_file, "\n");
122     if isoctave()
123       fflush(dat_file);
124     end %if
125   end % if
126
127   DoA_loop_range = [DoA_range DoA_range 360]; % Duas
        voltas
128
129   percent = 0;
130   ref_iteration = 1/(length(DoA_loop_range)-1);
131   it = 0;
132   DelayTime = 15*ref_iteration;
```

```
133   r = R_upper+R_lower;
134
135   for DoA = DoA_loop_range
136     it = it + 1;
137
138     fprintf('%.2f%% -> %s\n', percent*100, name);
139
140     if CHG_PHI
141       phase = phase + 2*pi*ref_iteration;
142     end %if
143
144     if CHG_R
145       r = r - R_upper/(2*length(DoA_range));
146     end %if
147
148     if CHG_THETA
149       ang_W = deg2rad(DoA);
150     else
151       ang_W = pi*5/12;
152     end %if
153
154     return_struct = calc_AoA(amp_0, ang_W, r, phase,
          lambda, ...
155       omega, S, C, NOISE, SNR_dB, ATT, resolution, d ,
            N_antenas);
156
157     [ ...
158       choose_angle, ...
159       Rho, ...
160       ant_array, ...
161       Z_phase_array, ...
162       Z_x_array, ...
163       delta_A_x_B, ...
164       delta_B_x_A, ...
165     ] = return_struct{:};
166
167     % Calcular a figura de fundo para visualizacao em
          imagem
168     z_plot = signal_r(x, y, t_0, amp_0, ang_W, r, phase,
          ...
169       lambda, omega, S, C, NOISE, SNR_dB, ATT);
170
171     generate_fig( ...
172       z_plot, ...
173       x, ...
174       y, ...
175       ang_W, ...
176       lambda, ...
```

```
177         interval, ...
178         Rho, ...
179         choose_angle, ...
180         ant_array, ...
181         Z_phase_array, ...
182         Z_x_array, ...
183         delta_A_x_B, ...
184         delta_B_x_A ...
185       )
186
187       drawnow;
188
189       if S_GIF
190         frame = getframe(f);
191         im = frame2im(frame);
192
193         [imind, cm] = rgb2ind(im);
194         if it == 1
195           % Create GIF file
196           imwrite(imind, cm, gif_filename,'gif','DelayTime',
                DelayTime , 'Compression' , 'lzw');
197         else
198           % Add each new plot to GIF
199           imwrite(imind, cm, gif_filename,'gif','WriteMode',
              'append','DelayTime', DelayTime , 'Compression'
               , 'lzw');
200         end %if
201       else
202         % pause(1/30)
203         pause(0.0001)
204       end %if
205
206       if S_DAT
207         fprintf(dat_file, "%.2f", percent*100);
208         fprintf(dat_file, "\t%.3f", normalize_angle(ang_W));
209         fprintf(dat_file, "\t%.3f", r);
210         fprintf(dat_file, "\t%.3f", normalize_angle(phase));
211         fprintf(dat_file, "\t%.3f", normalize_angle(
              choose_angle));
212
213         for i = ant_idx_list
214           fprintf(dat_file, "\t%.3f", normalize_angle(
                delta_A_x_B(i)));
215         end % for
216         for i = ant_idx_list
217           fprintf(dat_file, "\t%.3f", normalize_angle(
                delta_B_x_A(i)));
218         end % for
```

```
219
220         fprintf(dat_file, "\n");
221         if isoctave()
222           fflush(dat_file);
223         end %if
224       end % if
225
226       percent = percent + ref_iteration;
227
228     end %for
229
230     if S_GIF
231       printf('Check: %s\a\n', gif_filename);
232     end %if
233
234     if S_DAT
235       fclose(dat_file);
236       printf('Check: %s\a\n', dat_filename);
237     end %if
238
239 end %function
```

A.2.5 Arquivos de simulação em sequência

Código 19: 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   true, ... % 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   5, ... % range_step
20   5 ... % N_antenas
21   );
```

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

```
clear all;
close all;
clc;

if isoctave()
  pkg load communications;
  pkg load statistics;
end %if

S_GIF = true;
S_DAT = true;

range_step = 5;

N_antenas = 3;

% w_xyt(NOISE, ATT, CHG_PHI, CHG_R, CHG_THETA, S_GIF,
   S_DAT, SNR, range_step, N_antenas);

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

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

w_xyt(false, false, false, false, true,  S_GIF, S_DAT,
   1/1,   range_step, N_antenas);
% % w_xyt(false, false, false, true,  true,  S_GIF, S_DAT,
```

```
    1/1,   range_step, N_antenas);

w_xyt(false, true,  false, false, true,  S_GIF, S_DAT,
   1/1,   range_step, N_antenas);
% % w_xyt(false, true,  false, true,  true,  S_GIF, S_DAT,
    1/1,   range_step, N_antenas);

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

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

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

w_xyt(true,  true,  false, false, true,  S_GIF, S_DAT,
   1/1,   range_step, N_antenas);
w_xyt(true,  true,  false, false, true,  S_GIF, S_DAT,
   5/1,   range_step, N_antenas);
w_xyt(true,  true,  false, false, true,  S_GIF, S_DAT,
   25/1,  range_step, N_antenas);
w_xyt(true,  true,  false, false, true,  S_GIF, S_DAT,
   50/1,  range_step, N_antenas);
```

```
61 w_xyt(true,  true,  false, false, true,  S_GIF, S_DAT,
       100/1, range_step, N_antenas);
```

Código 21: Arquivo 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 range_step = 5;
14
15 N_antenas = 7;
16
17 % w_xyt(NOISE, ATT, CHG_PHI, CHG_R, CHG_THETA, S_GIF,
       S_DAT, SNR, range_step, N_antenas);
18
19 w_xyt(false, false, true,  false, false, S_GIF, S_DAT,
       1/1,   range_step, N_antenas);
20 w_xyt(true,  false, true,  false, false, S_GIF, S_DAT,
       1/1,   range_step, N_antenas);
21 w_xyt(true,  false, true,  false, false, S_GIF, S_DAT,
       5/1,   range_step, N_antenas);
22 w_xyt(true,  false, true,  false, false, S_GIF, S_DAT,
       25/1,  range_step, N_antenas);
23 w_xyt(true,  false, true,  false, false, S_GIF, S_DAT,
       50/1,  range_step, N_antenas);
24 w_xyt(true,  false, true,  false, false, S_GIF, S_DAT,
       100/1, range_step, N_antenas);
25
26 w_xyt(false, true,  true,  false, false, S_GIF, S_DAT,
       1/1,   range_step, N_antenas);
27 w_xyt(true,  true,  true,  false, false, S_GIF, S_DAT,
       1/1,   range_step, N_antenas);
28 w_xyt(true,  true,  true,  false, false, S_GIF, S_DAT,
       5/1,   range_step, N_antenas);
29 w_xyt(true,  true,  true,  false, false, S_GIF, S_DAT,
       25/1,  range_step, N_antenas);
30 w_xyt(true,  true,  true,  false, false, S_GIF, S_DAT,
       50/1,  range_step, N_antenas);
31 w_xyt(true,  true,  true,  false, false, S_GIF, S_DAT,
       100/1, range_step, N_antenas);
```

```
32
33 % w_xyt(false, false, false, false, true,  S_GIF, S_DAT,
      1/1,   range_step, N_antenas);
34 w_xyt(false, false, false, true,  true,  S_GIF, S_DAT,
      1/1,   range_step, N_antenas);
35
36 % w_xyt(false, true,  false, false, true,  S_GIF, S_DAT,
      1/1,   range_step, N_antenas);
37 w_xyt(false, true,  false, true,  true,  S_GIF, S_DAT,
      1/1,   range_step, N_antenas);
38
39 w_xyt(true,  false, false, true,  true,  S_GIF, S_DAT,
      1/1,   range_step, N_antenas);
40 w_xyt(true,  false, false, true,  true,  S_GIF, S_DAT,
      5/1,   range_step, N_antenas);
41 w_xyt(true,  false, false, true,  true,  S_GIF, S_DAT,
      25/1,  range_step, N_antenas);
42 w_xyt(true,  false, false, true,  true,  S_GIF, S_DAT,
      50/1,  range_step, N_antenas);
43 w_xyt(true,  false, false, true,  true,  S_GIF, S_DAT,
      100/1, range_step, N_antenas);
44
45 w_xyt(true,  true,  false, true,  true,  S_GIF, S_DAT,
      1/1,   range_step, N_antenas);
46 w_xyt(true,  true,  false, true,  true,  S_GIF, S_DAT,
      5/1,   range_step, N_antenas);
47 w_xyt(true,  true,  false, true,  true,  S_GIF, S_DAT,
      25/1,  range_step, N_antenas);
48 w_xyt(true,  true,  false, true,  true,  S_GIF, S_DAT,
      50/1,  range_step, N_antenas);
49 w_xyt(true,  true,  false, true,  true,  S_GIF, S_DAT,
      100/1, range_step, N_antenas);
50
51 % w_xyt(true,  false, false, false, true,  S_GIF, S_DAT,
      1/1,   range_step, N_antenas);
52 % w_xyt(true,  false, false, false, true,  S_GIF, S_DAT,
      5/1,   range_step, N_antenas);
53 % w_xyt(true,  false, false, false, true,  S_GIF, S_DAT,
      25/1,  range_step, N_antenas);
54 % w_xyt(true,  false, false, false, true,  S_GIF, S_DAT,
      50/1,  range_step, N_antenas);
55 % w_xyt(true,  false, false, false, true,  S_GIF, S_DAT,
      100/1, range_step, N_antenas);
56
57 % w_xyt(true,  true,  false, false, true,  S_GIF, S_DAT,
      1/1,   range_step, N_antenas);
58 % w_xyt(true,  true,  false, false, true,  S_GIF, S_DAT,
      5/1,   range_step, N_antenas);
```

```
59 % w_xyt(true,  true,  false, false, true,  S_GIF, S_DAT,
       25/1,  range_step, N_antenas);
60 % w_xyt(true,  true,  false, false, true,  S_GIF, S_DAT,
       50/1,  range_step, N_antenas);
61 % w_xyt(true,  true,  false, false, true,  S_GIF, S_DAT,
       100/1, range_step, N_antenas);
```