UNIVERSIDADE FEDERAL DE CAMPINA GRANDE – UFCG
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA – CEEI
DEPARTAMENTO DE ENGENHARIA ELÉTRICA – DEE

José Joseilton dos Santos Souza

Processamento de Sinais Aplicados ao radiotelescópio BINGO para filtragem e classificação de eventos

Campina Grande - PB

5 de novembro de 2024

José Joseilton dos Santos Souza

Processamento de Sinais Aplicados ao radiotelescópio BINGO para filtragem e classificação de eventos

Projeto de Engenharia Elétrica submetido à Coordenadoria de Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Engenharia Elétrica.

Orientador: Prof. Edmar Candeia Gurjão, D.Sc.

Campina Grande - PB

5 de novembro de 2024

José Joseilton dos Santos Souza

Processamento de Sinais Aplicados ao radiotelescópio BINGO para filtragem e classificação de eventos

Trabalho aprovado. Campina Grande - PB, 5 de novembro de 2024:

Prof. Edmar Candeia Gurjão, D.Sc.
Orientador

Prof. Leocarlos Bezerra da Silva Lima, D.Sc.
Convidado

Campina Grande - PB
5 de novembro de 2024

Dedico este trabalho a minha querida avó, Mãe Carminha (in memoriam), por todo amor e carinho.

Agradecimentos

Expresso minha profunda gratidão aos meus pais, Cícera e Jorge, cujo apoio incondicional foi uma âncora essencial, especialmente nos momentos mais difíceis, sempre incentivando meus sonhos. Aos meus irmãos, Josenildo e Joseildo, agradeço por todo o carinho e companheirismo. Aos demais familiares, em especial à minha avó Mãe Carminha (*in memoriam*), que acreditou e torceu por mim até seus últimos dias, e à minha tia Nega e meu avô Benedito, por todo incentivo e afeto. Amo vocês profundamente e reconheço o papel essencial que cada um teve na minha trajetória.

Agradeço aos amigos(as) Ana Térsia, João Lennon, João Vitor, Raimundo, Rute, Maria Analice, Kevyn, Thiago Cruz, Lucas Farias, Isac, Sávio Mateus e Sávio Bezerra pela amizade e companheirismo que tornaram essa jornada mais leve. Suas palavras de apoio e confiança foram essenciais para manter meu ânimo e celebrar cada etapa conquistada.

Aos amigos da capacitação em microeletrônica – Ícaro, Luiz, Juliana, João Pedro, Valmir e demais colegas da Xmen, cuja menção individual não é possível, expresso minha gratidão pela amizade, colaboração e pelas valiosas trocas de ideias ao longo dessa jornada. Compartilhar conhecimento e experiências com cada um de vocês tornou essa vivência ainda mais rica e inspiradora.

Minha sincera gratidão à Professora Georgina pela orientação valiosa durante a capacitação (Xmen). Sua dedicação e conhecimento tornaram essa experiência profundamente enriquecedora. Agradeço pelo tempo e sabedoria compartilhados, que foram essenciais para meu desenvolvimento acadêmico e pessoal.

Expresso minha gratidão ao Professor Luciano Barosi pelo apoio constante e pela paciência ao longo de dessa jornada.

Agradeço ao Professor Edmar pela orientação valiosa, pelo conhecimento compartilhado e pela dedicação em me guiar no desenvolvimento deste trabalho. Sou profundamente grato pela paciência, pela confiança em meu potencial e pela oportunidade de aprendizado que me foi concedida.

Finalmente, agradeço a todos que cruzaram o meu caminho e contribuíram para quem sou hoje.

"Minhas irmãs, meus irmãos
Se assumam como realmente são
Não deixem que suas matrizes
Que suas raízes morram por falta de irrigação
Ser nortista e nordestino, meus conterrâneos
Num é ser seco nem litorâneo
É ter em nossas mãos um destino
Nunca clandestino para os desfechos metropolitanos"
RAPadura Xique-Chico. *Norte Nordeste Me Veste*, 2019.

Resumo

Este estudo, intitulado "Processamento de Sinais Aplicados ao Radiotelescópio BINGO para Filtragem e Classificação de Eventos", investiga métodos de mitigação de interferência de radiofrequência (RFI) em dados de radioastronomia, com ênfase nos métodos arPLS e *SumThreshold* para correção de linha de base e preservação de sinais astrofísicos. A pesquisa utiliza dados obtidos do radiotelescópio Uirapuru, aos quais foram intencionalmente adicionados ruídos senoidais, rosa, gaussiano, de banda larga e impulsivo, permitindo avaliar a eficácia desses métodos na melhoria da qualidade dos dados captados pelo BINGO. Os resultados demonstram que os métodos aplicados são eficazes na identificação e isolamento de interferências, possibilitando a recuperação de informações essenciais para a análise observacional e contribuindo significativamente para a robustez dos dados em ambientes densamente saturados por sinais artificiais.

Palavras-chaves: Radioastronomia, Radiotelescópio BINGO, Interferência de Radiofrequência (RFI), arPLS, *SumThreshold*, Processamento de Sinais

Abstract

This study, titled "Signal Processing Applied to the BINGO Radio Telescope for Event Filtering and Classification," investigates methods for mitigating radio frequency interference (RFI) in radio astronomy data, with an emphasis on the arPLS and SumThreshold methods for baseline correction and preservation of astrophysical signals. The research utilizes data obtained from the Uirapuru radio telescope, to which sinusoidal, pink, Gaussian, broadband, and impulsive noise were intentionally added, allowing for an assessment of the effectiveness of these methods in improving the quality of data captured by BINGO. The results demonstrate that the applied methods are effective in identifying and isolating interferences, enabling the recovery of essential information for observational analysis and significantly contributing to data robustness in environments densely saturated with artificial signals.

Key-words: Radio Astronomy, BINGO Radio Telescope, Radio Frequency Interference (RFI), arPLS, *SumThreshold*, Signal Processing;

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ALMA	Atacama Large Millimeter Array
ARPLS	Asymmetrically Reweighted Penalized Least Squares
BAO	Baryon Acoustic Oscillations
BINGO	Baryon Acoustic Oscillations in Neutral Gas Observations
FITS	Flexible Image Transport System
FRB	Fast Radio Bursts
GHz	Gigahertz
K	Kelvin
MHz	Megahertz
PDF	Probability Density Function
RFI	Radio Frequency Interference
RSD	Redshift Space Distortions
SNR	Signal-to-Noise Ratio
ST	SumThreshold
USRP	Universal Software Radio Peripheral

Sumário

1 Introdução

A radioastronomia é um ramo da astronomia que alavancou a observação e análise de fenômenos do cosmos que não podem ser percebidos a luz visível. A ciência de ver o universo nas frequências de rádio foi disponibilizada para os cientistas em 1932, quando Karl Jansky, por acidente, viu ondas de rádio vindo da Via Láctea. A primeira descoberta que ele fez se tornou histórica, marcando o início da radioastronomia como campo científico. Assim, radioastronomia complementa a ciência da óptica de modo direto. Aumentando as possibilidades de estudar as áreas escuras e inacessíveis do cosmos. Eles incluem as massas de nuvens moleculares interestelares e radiação cósmica de fundo que é importante para a compreensão da formação e evolução do universo.

À medida que a radioastronomia vem sendo aprimorada com técnicas como interferometria, consegue uma melhor resolutividade angular e, em consequência, um melhor detalhamento das imagens e observação mais precisa dos objetos. Esta evolução tecnológica possibilitou, por exemplo, o estudo de galáxias ativas, pulsares e até fenômenos de alta energia, como jatos relativísticos. As observações em rádio permitem ainda o estudo da formação estelar e galáxias, ao detectar hidrogênio neutro, que é um dos principais materiais para o estudo da repartição da matéria no universo.

No entanto, o problema com a radioastronomia é a crescente densidade de interferências de radiofrequência causadas pelo número crescente de satélites de telecomunicação e tecnologias de transmissão. Perturbações de comunicação por sistemas fornecidos por dispositivos como GPS, Galileo e GLONASS constituem uma interferência constante ao realizar observações astronômicas que amostram sinais de baixa intensidade astrofísicos. Essa situação requer respostas adequadas que possam filtrar efetivamente os sinais indesejados de tudo que não é de interesse astronômico.

Para garantir a qualidade dos dados em projetos de radioastronomia, são empregadas técnicas de correção avançadas, como o método *Asymmetric Reweighted Penalized Least Squares* (ArPLS) e o *SumThreshold*, que se consolidaram como estratégias essenciais para o processamento de sinais astrofísicos. O ArPLS ajusta a linha de base dos dados de maneira assimétrica, facilitando a remoção de interferências de fundo sem comprometer as características do sinal de interesse. Já o *SumThreshold* atua diretamente na identificação de interferências de radiofrequência (RFI), permitindo a preservação e a integridade dos sinais astrofísicos captados. Dessa forma, esses métodos asseguram a obtenção de dados de alta qualidade que contribuem para a observação e estudo do universo.

1.1 Objetivos

1.1.1 Objetivo Geral

Investigar e aplicar métodos de mitigação de interferências de radiofrequência (RFI) em dados astrofísicos captados por radiotelescópios, com foco nos métodos *ArPLS* e *SumThreshold*, para a correção de *baseline* e aprimoramento da qualidade dos dados.

1.1.2 Objetivos Específicos

- Analisar as propriedades da interferência de radiofrequência (RFI) e seu impacto no processamento de dados em radioastronomia.
- Explorar o uso do método *SumThreshold* para a identificação e mitigação de RFI em observações astrofísicas.
- Aplicar o método *ArPLS* para a correção de *baseline* em dados espectrais contaminados por interferências.
- Avaliar a eficiência das técnicas de mitigação em diferentes tipos de ruído, como ruído gaussiano, senoidal e de banda larga.

1.2 Justificativa

A utilização dos métodos ArPLS e *SumThreshold* é justificada pela necessidade de mitigar a interferência de radiofrequência (RFI), um problema crescente que compromete a qualidade dos dados em radioastronomia. A interferência de sinais de rádio emitidos por outras tecnologias afeta diretamente as observações, tornando-se um desafio que os radioastrônomos precisam enfrentar para preservar a integridade dos sinais astrofísicos captados.

Assim, esses métodos são fundamentais para melhorar a qualidade dos dados e garantir a precisão das observações em ambientes saturados por sinais artificiais, assegurando que as investigações sobre a origem e o desenvolvimento do universo possam ser realizadas de forma confiável e precisa.

1.3 Estrutura do Trabalho

Este trabalho está estruturado em cinco capítulos. O Capítulo 1 - Introdução oferece uma contextualização sobre a radioastronomia e os desafios oriundos da interferência de radiofrequência (RFI), definindo os objetivos e a justificativa do estudo. No Capítulo 2 - Fundamentação Teórica, são abordados conceitos centrais de radioastronomia, as

características da RFI e os métodos de mitigação *ArPLS* e *SumThreshold*, além de uma visão detalhada dos radiotelescópios BINGO e Uirapuru. O Capítulo 3 - Metodologia, Simulação e Análise dos Resultados descreve os procedimentos experimentais para simulação e contaminação dos dados com diversos tipos de ruído, detalhando a aplicação dos métodos de mitigação de RFI. Em seguida, o Capítulo 4 - Resultados e Discussão apresenta e examina os resultados obtidos com a mitigação das interferências, discutindo a eficácia dos métodos *ArPLS* e *SumThreshold* em diferentes cenários. Por fim, o Capítulo 5 - Conclusão resume as principais contribuições do estudo, enfatizando as técnicas de mitigação e propondo sugestões para trabalhos futuros que visem aprimorar a redução de RFI em radioastronomia.

2 Fundamentação teórica

2.1 Radioastronomia

Devido à capacidade da radioastronomia representar em frequências de rádio fora do espectro visível, torna-se possível a identificação de eventos invisíveis por outros meios. De fato, o avanço na radioastronomia vem do fato de que os cientistas começaram a explorar instrumentos espaciais a frequências não visíveis. Isso permite que os astrônomos acessem a visualização de eventos que são obscurecidos pela poeira interestelar. Portanto, a radioastronomia cobre uma ampla gama de frequências de aproximadamente alguns 10 MHz a 1 THz. Ela permite que visualizemos fontes distantes e frias, que incluem nuvens moleculares interestelares, radio compactas estelares, pulsares e radiação cósmica de fundo ou resíduo do Big Bang, (GARY, 2024; CONDON; RANSOM, 2024).

Além disso, esta área cresceu exponencialmente após Karl Jansky em 1932 acidentalmente descobrir ondas de rádio sendo emitidas pela Via Láctea. Antes de Jansky, os astrônomos interpretavam que os objetos que emitiam radiação detectável provavelmente seriam visíveis, e não pensavam que eles poderiam ser fortemente emitentes em ondas de rádio (CONDON; RANSOM, 2024).(CONDON; RANSOM, 2024).

Por essa razão, a radioastronomia é capaz de atravessar camadas de nuvens de poeira no espaço interestelar, fornecendo visões que seriam impossíveis em outros comprimentos de onda. Essa possibilidade expandiu significativamente o campo da astronomia moderna, permitindo a observação e o estudo de fenômenos de alta energia e objetos distantes e frios que emitem pouca ou nenhuma luz visível (CONDON; RANSOM, 2024).

Além do mais, radioastrônomos também usam técnicas de interferometria, que é o uso de várias antenas para sintetizar uma única imagem com alta resolução. Com a ajuda dessa técnica, os radioastrônomos podem obter detalhes da estrutura cósmica em escalas angulares muito menores do que as que obtêm nos telescópios ópticos. Essas observações são essenciais para investigar objetos tais como buracos negros supermassivos, jatos relativísticos em galáxias ativas e a estrutura do meio interestelar em nossa galáxia.(GARY, 2024; CONDON; RANSOM, 2024).

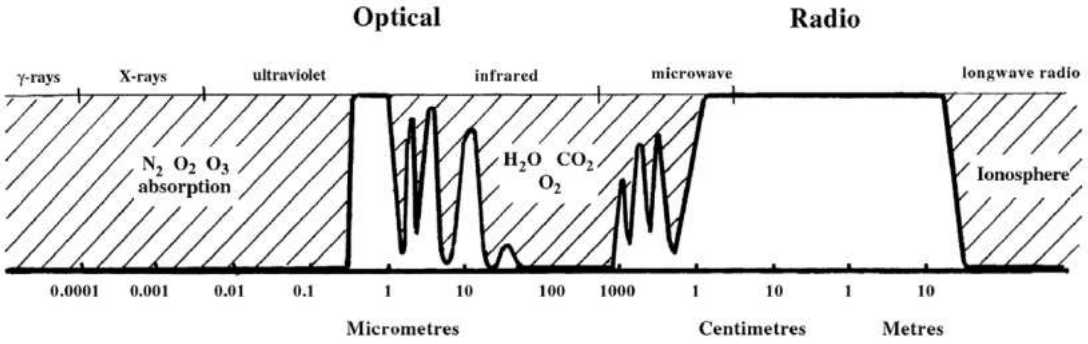
A radioastronomia não só ampliou a compreensão do universo ao permitir a descoberta de novos objetos e fenômenos, mas também continua sendo uma ferramenta crucial para investigar as condições físicas dos corpos celestes, ajudando a responder a questões fundamentais sobre a formação e evolução do cosmos (GARY, 2024; CONDON; RANSOM, 2024).

2.1.1 Janela Atmosférica

Como mencionado, a atmosfera age como um filtro de radiação eletromagnética da Terra, extinguindo a maioria das ondas produzidas em todo o cosmos nas várias faixas de frequência. Algumas frequências, no entanto, são capazes de penetrar essa barreira e atingir a superfície da Terra. Esses bandas são chamadas janelas atmosféricas. Por exemplo, desde a radioastronomia, no que diz respeito a janela de rádio "aberta", observamos o universo em uma das frequências mais amplas de rádio, variando de aproximadamente 10 MHz a 1 THz (CONDON; RANSOM, 2024).

A janela de rádio da Terra, que se estende aproximadamente de $15\,MHz$ ($\lambda \simeq 20\,m$) a $1{,}5\,THz$ ($\lambda \simeq 0{,}2\,mm$), como mostrado na Figura 1, não possui limites bem definidos devido às variações com altitude, localização geográfica e condições atmosféricas. A absorção em altas frequências é causada pela rotação de moléculas como H_2O e O_2, que possuem bandas de absorção significativas em frequências específicas, como $22{,}2\,GHz$ e $60\,GHz$, respectivamente. Para estender os limites superiores da frequência, é necessário realizar medições a partir de locais com baixo conteúdo de vapor d'água ou utilizar satélites e balões estratosféricos, já que a absorção por oxigênio é difícil de ser minimizada a partir da superfície terrestre (GARY, 2024).

Figura 1 – Espectro eletromagnético, mostrando o comprimento de onda das "janelas"atmosféricas.



Fonte: (BUKE; GRAHAM-SMITH; WILKINSON, 2019).

A janela de rádio é particularmente importante porque, ao contrário de outras formas de radiação, as ondas de rádio não são significativamente absorvidas pela atmosfera terrestre, com exceção de frequências muito baixas, que são refletidas pela ionosfera. Essa propriedade faz da radioastronomia uma das poucas áreas da astronomia que pode ser amplamente realizada a partir da superfície terrestre, sem a necessidade de telescópios espaciais (CONDON; RANSOM, 2024).

Nem todas as frequências de rádio são acessíveis. Em frequências abaixo de cerca de $10.MHz$, a ionosfera da Terra reflete as ondas de rádio de volta ao espaço, tornando sinais extraterrestres a essas frequências inacessíveis a observatórios terrestres. Na extremidade

oposta da janela de rádio, ao redor de $1.THz$, a absorção de moléculas de água e oxigênio na atmosfera começa a fazer as observações terrestres inatingíveis. A absorção por moléculas, em particular vapor de água mas também oxigênio, absorve a maior parte da radiação em comprimentos de onda mais curtos, restringindo a transparência atmosférica em locais de alta umidade (CONDON; RANSOM, 2024; GARY, 2024).

Por isso, os melhores locais para a instalação de radiotelescópios são aqueles com pouca interferência atmosférica, como desertos em grandes altitudes, na qual a atmosfera é mais rarefeita e seca. O Atacama *Large Millimeter Array* (ALMA), localizado no deserto do Atacama no Chile, é um exemplo de observatório construído em um ambiente com condições ideais para a observação em frequências mais altas da janela de rádio (CONDON; RANSOM, 2024).

Além da absorção atmosférica, a atmosfera também emite radiação de fundo que pode interferir com as observações astronômicas. Essa emissão, causada principalmente por moléculas de vapor d'água e hidrossóis (pequenas gotículas de água suspensas), pode aumentar significativamente o ruído de fundo em frequências específicas, particularmente nas bandas de $22\,GHz$ e $557\,GHz$, em que há linhas de emissão de vapor d'água (CONDON; RANSOM, 2024).

Em resumo, a janela atmosférica de rádio proporciona uma visão única do universo em comprimentos de onda que não podem ser observados através de outros meios, permitindo que astrônomos estudem fenômenos cósmicos como pulsares, buracos negros, galáxias distantes e a radiação cósmica de fundo, todos os quais emitem ondas de rádio que podem ser detectadas na Terra (GARY, 2024; CONDON; RANSOM, 2024).

2.2 Radiotelescópio BINGO

O Radiotelescópio BINGO, cujo nome é um acrônimo do inglês *Baryon Acoustic Oscillations in Neutral Gas Observations*, traduzido como 'Oscilações Acústicas de Bárions em Observações de Gás Neutro', representa uma iniciativa inovadora no campo da cosmologia, focada na investigação das oscilações acústicas de bárions por meio da observação do hidrogênio neutro. Ao medir a radiação de 21 cm, o projeto visa oferecer novas perspectivas sobre a distribuição da matéria no universo, contribuindo para o estudo da formação e evolução do universo em larga escala.

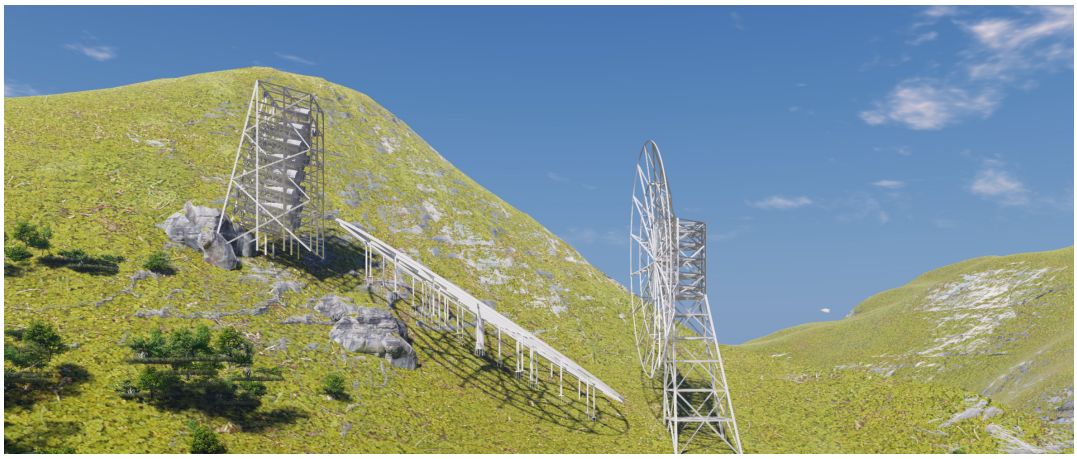
A configuração estrutural do radiotelescópio BINGO é composta por duas antenas parabólicas que focalizam a radiação em um conjunto de cornetas (Figura 2). Essa radiação é subsequentemente direcionada, por meio de um guia de onda, até o receptor. A instalação dará-se na Serra da Catarina, localizada no município de Aguiar, Paraíba, foi estrategicamente selecionada com base em níveis excepcionalmente baixos de interferência

de rádio (RFI), os quais são cruciais para assegurar a alta qualidade das observações. A topografia da região, caracterizada por colinas, contribui para a formação de uma "zona de silêncio de rádio", elemento indispensável para a precisão das medições. Além disso, outros fatores foram cuidadosamente avaliados, tais como a densidade populacional, as condições meteorológicas, o acesso à infraestrutura, e as propriedades do sinal de rádio que seria captado pelo telescópio (RAMALHO, 2023).

O BINGO, um telescópio de trânsito fixo, observa o céu na faixa de frequência de 980-1260 MHz, utilizando a rotação da Terra para revisitar diariamente o mesmo campo de visão. Como as antenas parabólicas são fixas e apontadas para o norte celeste, à medida que a Terra gira, o telescópio varre o céu, permitindo a observação de uma grande área em um curto período. Essa técnica de observação por varredura é eficiente e econômica para mapear grandes regiões do céu. Seu design ótico oferece um campo de visão instantâneo de 14,75° × 6,0°, coberto por 28 cornetas. Alinhado no sentido norte-sul, o telescópio apresenta baixos lóbulos laterais e excelente rejeição de polarização cruzada, com uma distância focal de 63 metros. A Tabela 1 resume os principais parâmetros do BINGO (ABDALLA et al., 2022)..

Os principais objetivos científicos do BINGO incluem a medição das oscilações acústicas de bárions (BAO) e da distorção do espaço de *redshift* (RSD), visando restringir parâmetros cosmológicos em diferentes modelos teóricos. Além disso, o telescópio permitirá a exploração de fenômenos astrofísicos, como explosões rápidas de rádio (FRBs) e pulsares, consolidando-se como uma ferramenta crucial na astronomia moderna.

Figura 2 – Representação da instalação do radiotelescópio BINGO na colina, destacando a disposição dos espelhos e a estrutura das cornetas.



Fonte: (ABDALLA et al., 2022).

Tabela 1 – Parâmetros Relevantes do Radiotelescópio BINGO para Mitigação de RFI

Parâmetro	**Valor**
T_{sys} (K)	70
Frequência de operação (MHz)	980-1260
Sensibilidade estimada (μK)	206
Banda de frequência - B (MHz)	280
Número de cornetas	28
Faixa de declinação	14,75°
Área total de levantamento (graus^2)	5324

Fonte: Adaptado de (ABDALLA et al., 2022).

2.2.1 Radiotelescópio Uirapuru

O Uirapuru é um radiotelescópio auxiliar ao projeto BINGO, sendo desenvolvido para apoiar pesquisas na detecção rápida de sinais de rádio. Instalado no município de Campina Grande, Paraíba, junto ao Laboratório de Metrologia (Labmet) da Universidade Federal de Campina Grande (UFCG), o Uirapuru utiliza uma corneta idêntica à do BINGO, otimizada para operar na faixa de frequência entre 960 MHz e 1280 MHz, permitindo a detecção precisa de emissão de hidrogênio neutro em galáxias distantes (RAMALHO, 2023).

A construção do Uirapuru envolveu uma série de desafios técnicos, desde o transporte e montagem da antena até o ajuste final, resultando em um equipamento que já contribui significativamente para o avanço da compreensão da evolução cósmica . A corneta do Uirapuru, com seu alinhamento norte-sul e capacidade de movimentação em elevação, faz dele um telescópio de trânsito, permitindo a varredura de áreas extensas do céu.

Além de sua função no suporte ao BINGO, o Uirapuru atua como campo de testes para o desenvolvimento de sensores remotos e softwares que serão empregados no BINGO. Ele também é utilizado para testar a resposta eletromagnética da corneta e para treinar pessoal e estudantes, preparando-os para a futura operação do radiotelescópio BINGO. Do ponto de vista científico, o Uirapuru contribui para o aprimoramento da sensibilidade da corneta, visando a monitorar pulsares brilhantes e, a médio prazo, servirá como protótipo para uma rede de *outriggers* que, em conjunto com o BINGO e outros radiotelescópios menores, buscará e localizará *Fast Radio Bursts* (FRBs).

Com uma capacidade de operação contínua de 24 horas por dia, o Uirapuru coleta dados que são processados para interpretar informações cosmológicas, oferecendo uma visão aprofundada sobre a estrutura do universo primordial e sua evolução. Atualmente, o radiotelescópio está em fase de testes e operando com sucesso, mesmo que no momento esteja coletando dados apenas em uma polarização, utilizando um radiômetro simples e um USRP Ettus como *frontend*.

2.3 Interferência de Radiofrequência (RFI)

A interferência de radiofrequência (RFI) é uma das principais ameaças ao processamento de sinais em sistemas de alta sensibilidade, como os utilizados na radioastronomia. Esse fenômeno ocorre quando sinais eletromagnéticos indesejados, originados principalmente de fontes terrestres e satélites de comunicação, sobrepõem-se aos sinais de interesse, prejudicando a qualidade dos dados. A expansão das tecnologias de comunicação, como o GPS, Galileo e GLONASS, tem aumentado a densidade de sinais de radiofrequência na atmosfera, tornando a RFI um desafio crescente, tanto para a radioastronomia quanto para outras áreas que dependem de dados precisos de fontes astrofísicas.

2.3.1 Tipos de RFI

A RFI pode ser classificada de acordo com a sua origem e características. Existem três tipos principais de interferência: banda larga, banda estreita e transitória. A interferência de **banda larga** abrange uma ampla faixa de frequências e pode ser contínua ou impulsiva, sendo frequentemente causada por emissões eletromagnéticas provenientes de dispositivos eletrônicos industriais, veículos e sistemas de transmissão de dados. Esse tipo de interferência afeta grandes porções do espectro, sendo particularmente danosa para radiotelescópios sensíveis, que precisam captar sinais fracos emitidos por fontes astrofísicas distantes (ZAINUD-DIN et al., 2021; THOMPSON; MORAN; JR, 2014).

Por outro lado, a interferência de **banda estreita** se manifesta em faixas específicas de frequências e é comumente associada a transmissores fixos, como emissoras de rádio e TV, além dos sistemas de comunicação via satélite, como GPS e Galileo. Esses sistemas, embora operem fora das bandas de frequências usadas em observações astronômicas, podem gerar harmônicos ou interferências que degradam a qualidade dos sinais captados. Finalmente, a **interferência transitória** é caracterizada por picos de intensidade de curta duração e ocorre de maneira imprevisível, muitas vezes causada por reflexões de sinais em objetos em movimento, como aviões ou satélites (OFFRINGA et al., 2010).

2.3.2 Impacto da RFI no Processamento de Sinais

A presença de RFI no processamento de sinais pode gerar efeitos significativos, sobretudo na radioastronomia, nos quais a captação de sinais fracos, como os emitidos por pulsares e quasares, exige uma relação sinal-ruído (SNR) extremamente alta. A RFI pode mascarar ou distorcer esses sinais, prejudicando a análise e a interpretação dos dados. Além disso, há o risco de falsos positivos, em que sinais interferentes podem ser erroneamente identificados como eventos astrofísicos, o que já foi observado em tentativas de detecção de tecnossignaturas extraterrestres (ZHANG; LI; LIU, 2022).

Os sistemas de comunicação via satélite, como o GPS e o Galileo, têm agravado o problema da RFI. Esses sistemas transmitem sinais poderosos em faixas de frequência específicas que, mesmo não coincidindo diretamente com as bandas astronômicas, podem interferir indiretamente nas observações. Isso se torna especialmente problemático com o crescente número de satélites orbitando a Terra, aumentando a probabilidade de interferências transitórias e contínuas, o que torna a mitigação da RFI uma prioridade crescente (OFFRINGA et al., 2010).

2.3.3 Técnicas de Mitigação de RFI

Diante desse cenário, várias técnicas de mitigação de RFI têm sido desenvolvidas para minimizar os impactos sobre o processamento de sinais. Entre elas, destaca-se o método **SumThreshold**, uma técnica amplamente utilizada em radiotelescópios modernos, como o LOFAR. Esse método pós-correlação foi projetado para detectar e eliminar interferências de maneira eficiente, utilizando um processo iterativo que ajusta a sensibilidade de detecção em diferentes escalas de tempo e frequência. Ao longo das iterações, o *SumThreshold* identifica e flagra os dados contaminados por RFI, sendo eficaz tanto para interferências de banda estreita quanto para as transitórias (OFFRINGA et al., 2010).

Outra técnica relevante é o **ArPLS (Mínimos Quadrados Penalizados Assimétricos)**, utilizado na correção de *baseline* em dados espectrais. O ArPLS corrige as distorções causadas pela interferência sem remover os sinais de interesse, o que o torna uma ferramenta ideal para observações astrofísicas em ambientes altamente interferidos. Ele penaliza as variações de intensidade de forma assimétrica, ajustando a *baseline* e preservando os sinais mais fracos. Esse método é especialmente útil para mitigar a interferência de banda larga, comum em áreas próximas a fontes de emissão contínua, como zonas urbanas e industriais (THOMPSON; MORAN; JR, 2014).

2.4 Correção de *Baseline* Usando Mínimos Quadrados Penalizados

A correção de *baseline* é uma etapa essencial na análise de espectros, especialmente em espectroscopia, na qual a presença de ruído de fundos variáveis pode comprometer a precisão das medições e a calibração dos instrumentos. Um dos principais desafios nesses dados é a presença de um fundo curvo, frequentemente causado por fluorescência. A eliminação desse fundo e a correção da *baseline* têm sido amplamente estudadas, com diversas metodologias propostas para lidar com essas variações.

2.4.1 Método dos Mínimos Quadrados

O método dos mínimos quadrados é amplamente utilizado para aproximar uma função $f(x)$ por uma função $F(x)$, que é uma combinação linear de funções conhecidas. O objetivo é minimizar a distância entre $f(x)$ e $F(x)$, de modo que $F(x)$ seja a melhor aproximação possível de $f(x)$. Essa função aproximada é expressa como:

$$F(x) = a_0 g_0(x) + a_1 g_1(x) + \cdots + a_m g_m(x),$$

em que $g_0(x), g_1(x), \ldots, g_m(x)$ são funções conhecidas, e $a_0, a_1, \ldots, a_m$ são os coeficientes que minimizam a soma dos quadrados dos desvios. O método é essencial quando $f(x)$ é definida por processos não finitos, como integrais ou somas de séries, ou quando $f(x)$ é conhecida por um conjunto de pares de pontos obtidos experimentalmente (FRANCO, 2006).

Existem dois casos principais para a aplicação do método dos mínimos quadrados: o caso contínuo e o caso discreto.

- **Caso Contínuo**:

No caso contínuo, a função $f(x)$ pertence ao espaço das funções contínuas $C[a, b]$, e busca-se aproximar $f(x)$ por um polinômio $P_m(x)$ de grau no máximo m. A função $P_m(x)$ é obtida minimizando a seguinte integral, que representa a soma dos quadrados das diferenças entre $f(x)$ e $P_m(x)$:

$$Q = \int_a^b (f(x) - P_m(x))^2 \, dx. \tag{2.1}$$

Aqui, a solução corresponde à projeção ortogonal de $f(x)$ sobre o subespaço dos polinômios de grau até m, o que garante que $P_m(x)$ seja a melhor aproximação de $f(x)$ nesse espaço (FRANCO, 2006).

- **Caso Discreto**:

No caso discreto, a função $f(x)$ é dada por $n+1$ pares de pontos $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n))$, onde $y_i = f(x_i)$, e deseja-se ajustar um polinômio $P_m(x)$ de grau m (com $m < n$) que melhor se ajuste a esses pontos. O objetivo é minimizar a soma dos quadrados dos desvios $(y_i - P_m(x_i))^2$ entre os valores da função e os valores do polinômio ajustado nos pontos x_i, ou seja, minimizar:

$$Q = \sum_{i=0}^{n} (y_i - P_m(x_i))^2. \tag{2.2}$$

Isso resulta em um sistema linear de equações, no qual os coeficientes $a_0, a_1, \ldots, a_m$ do polinômio são determinados de modo a minimizar o erro de aproximação. Esse método é amplamente utilizado em ajuste de curvas experimentais e análise de dados, nos quais é necessário encontrar a curva que melhor descreve os dados observados (FRANCO, 2006).

2.4.2 Métodos AsLS e airPLS

A suavização é uma técnica fundamental para reduzir o ruído em sinais espectrais, permitindo uma melhor visualização das tendências subjacentes nos dados. Em métodos de suavização baseados em mínimos quadrados penalizados, o objetivo é equilibrar a fidelidade ao sinal original com a suavidade desejada no resultado final. Isso é alcançado minimizando uma função de custo que combina dois termos: a diferença entre o sinal suavizado e o sinal original, e um termo de penalidade que controla a suavidade do ajuste.

A função de minimização é dada por:

$$S\left(\mathbf{z}\right) = \left(\mathbf{y} - \mathbf{z}\right)^T \left(\mathbf{y} - \mathbf{z}\right) + \lambda \mathbf{z}^T \mathbf{D}^T \mathbf{D} \mathbf{z}, \tag{2.3}$$

na qual $\mathbf{y}$ é o sinal original, $\mathbf{z}$ é o sinal suavizado e λ é um parâmetro de suavidade. O termo $\mathbf{D}$ refere-se a uma matriz de diferenças que calcula as variações no sinal para garantir que ele permaneça suave. A matriz $\mathbf{D}$ é definida como uma matriz de diferenças de segunda ordem[1], conforme apresentado a seguir:

$$\mathbf{D} = \begin{bmatrix} 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & -2 & 1 \end{bmatrix}. \tag{2.4}$$

Para melhorar o ajuste da *baseline* em dados espectrais, um vetor de pesos $\mathbf{W}$ é introduzido na função de minimização, permitindo que diferentes partes do sinal recebam diferentes níveis de atenção. Isso é especialmente útil para evitar que regiões de pico sejam tratadas da mesma forma que regiões de ruído. A função de suavização modificada com pesos é expressa por:

$$S\left(\mathbf{z}\right) = \left(\mathbf{y} - \mathbf{z}\right)^T \mathbf{W} \left(\mathbf{y} - \mathbf{z}\right) + \lambda \mathbf{z}^T \mathbf{D}^T \mathbf{D} \mathbf{z}. \tag{2.5}$$

A solução da equação de minimização é obtida por meio da derivada da função de suavização em relação a $\mathbf{z}$ ($\frac{\partial S}{\partial \mathbf{z}^T} = 0$), que leva à seguinte expressão:

[1] Uma matriz de diferenças de segunda ordem é uma forma de aproximar numericamente a segunda derivada de uma função, usada comumente em problemas de diferenças finitas e soluções numéricas de equações diferenciais (LEVEQUE, 2007).

$$\frac{\partial S}{\partial \mathbf{z}^T} = -2\mathbf{W}(\mathbf{y} - \mathbf{z}) + 2\lambda \mathbf{D}^T \mathbf{D}\mathbf{z} = 0 \tag{2.6}$$

$$\mathbf{z} = \left(\mathbf{W} + \lambda \mathbf{D}^T \mathbf{D}\right)^{-1} \mathbf{W}\mathbf{y}. \tag{2.7}$$

O método AsLS (*Asymmetric Least Squares*), conforme proposto por (EILERS; BOELENS, 2005), introduz um parâmetro de assimetria p que ajusta os pesos de forma não simétrica, permitindo uma correção eficiente da *baseline* sem a necessidade de identificar previamente as regiões de pico. Os pesos w_i são ajustados iterativamente conforme:

$$w_i = \begin{cases} p, & y_i > z_i \\ 1 - p, & y_i \leq z_i \end{cases}. \tag{2.8}$$

O parâmetro p é geralmente escolhido em um intervalo de valores entre 0,001 e 0,1, e a suavização é realizada de forma iterativa até que os pesos se estabilizem.

O método airPLS (*Adaptive Iteratively Reweighted Penalized Least Squares*) foi proposto para superar algumas limitações do AsLS, como a necessidade de ajustar os parâmetros de assimetria. No airPLS, os pesos são ajustados iterativamente de maneira adaptativa. A ideia principal do método é que, se uma amostra y_i estiver acima da *baseline* estimada z_i, ela será tratada como parte de um pico e receberá um peso zero. Para amostras abaixo da *baseline*, os pesos são atualizados conforme a seguinte equação:

$$w_i = \begin{cases} 0, & y_i \geq z_i \\ e^{t(y_i - z_i)/|\mathbf{d}|}, & y_i < z_i \end{cases}, \tag{2.9}$$

em que $\mathbf{d}$ contém os elementos negativos da diferença entre o sinal original e a *baseline*. O processo é repetido iterativamente até que as mudanças nos pesos sejam pequenas o suficiente para atender a uma condição de convergência, dada por:

$$|\mathbf{d}| < 0,001 \cdot |\mathbf{y}|. \tag{2.10}$$

O método airPLS oferece uma solução mais robusta para espectros com ruídos, ajustando a *baseline* de forma adaptativa e iterativa, o que o torna ideal para casos em que há uma variação significativa entre picos e a *baseline*.

2.4.3 O Método arPLS

Embora o método airPLS tenha melhorado a adaptação da *baseline*, o método arPLS (*Asymmetrically Reweighted Penalized Least Squares*), descrito por (BAEK et al.,

2015), foi desenvolvido para resolver problemas em que o arPLS subestima a *baseline* em regiões de picos elevados. O arPLS ajusta os pesos de forma adaptativa, corrigindo a *baseline* de maneira mais precisa em dados com ruídos significativos.

No método arPLS, uma função *logistic* generalizada é utilizada para ajustar os pesos de maneira assimétrica:

$$w_i = \begin{cases} logistic\left(y_i - z_i, m_{\mathbf{d}^-}, \sigma_{\mathbf{d}^-}\right), & y_i \geq z_i \\ 1, & y_i \leq z_i \end{cases}. \tag{2.11}$$

Essa função logística equilibra os pesos atribuídos ao sinal, de modo que regiões com picos significativos sejam tratadas de forma diferenciada. A função logística é expressa por:

$$logistic\left(d, m, \sigma\right) = \frac{1}{1 + e^{\frac{2(d-(-m+2\sigma))}{\sigma}}}, \tag{2.12}$$

onde m e σ representam a média e o desvio padrão das diferenças entre o sinal e a *baseline* ajustada. A função é utilizada iterativamente até que as mudanças nos pesos sejam mínimas, garantindo uma correção eficaz da *baseline* sem subestimar picos significativos.

2.5 Método *SumThreshold*

O método *SumThreshold* é amplamente utilizado na radioastronomia para a mitigação de interferências de radiofrequência (RFI). Esta técnica se destaca pela sua capacidade de detectar sinais interferentes ao mesmo tempo que preserva os sinais astrofísicos autênticos. Em observatórios de rádio, nos quais a precisão é crucial, o *SumThreshold* é um dos métodos preferidos devido ao seu sistema adaptativo de limiares de detecção, que se ajusta automaticamente às características do ruído de fundo nos dados.

2.5.1 Principais Conceitos e Estrutura do *SumThreshold*

O *SumThreshold* opera a partir de uma série de iterações, em que, em cada etapa, o método calcula e aplica um limiar T de detecção que discrimina valores de RFI em relação ao ruído de fundo natural dos dados observacionais. O limiar T é baseado na Distribuição Normal (ou Gaussiana), que descreve estatisticamente o comportamento da maioria dos ruídos de fundo em dados de rádio. Esse modelo matemático permite uma abordagem confiável e adaptativa, no qual o método identifica variações bruscas de intensidade como prováveis interferências.

2.5.2 Descrição Matemática da Distribuição Gaussiana

Para modelar o ruído de fundo nos dados observacionais, a função densidade de probabilidade (PDF) da Distribuição Gaussiana é definida como:

$$\phi(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \tag{2.13}$$

com as seguintes definições:

- x representa o valor de uma amostra,
- μ é a média dos dados (centro da distribuição),
- σ é o desvio padrão, que indica a dispersão dos valores em torno da média.

A curva da Distribuição Gaussiana possui uma forma de sino simétrico, no qual aproximadamente 68% dos valores estão dentro de um desvio padrão da média ($\mu \pm \sigma$), 95% dentro de dois desvios padrão ($\mu \pm 2\sigma$) e 99,7% dentro de três desvios padrão ($\mu \pm 3\sigma$). Essa propriedade é fundamental para o *SumThreshold*, pois fornece uma base confiável para definir limiares de exclusão de RFI, descartando valores que se desviam de μ em função de σ como interferências.

2.5.3 Aplicação da Distribuição Gaussiana no *SumThreshold*

No *SumThreshold*, o limiar T é calculado com base na média μ e no desvio padrão σ dos dados em cada iteração, conforme a expressão:

$$T_i(\nu, t) = \mu_i(\nu, t) + k \cdot \sigma_i(\nu, t), \tag{2.14}$$

em que:

- $T_i(\nu, t)$ representa o limiar adaptado ao ponto $V(\nu, t)$,
- $\mu_i(\nu, t)$ e $\sigma_i(\nu, t)$ são a média e o desvio padrão do sinal na i-ésima iteração,
- k é um fator de sensibilidade ajustável, geralmente entre 3 e 5, que determina o nível de exclusão de RFI.

Ao usar essa definição, o *SumThreshold* consegue ajustar-se ao ruído de fundo, definindo um intervalo $\mu \pm k\sigma$ que é seguro para a identificação de RFI. Valores que excedem esse intervalo são considerados RFI e descartados, o que preserva a integridade dos dados astrofísicos genuínos (OFFRINGA et al., 2010; BAAN et al., 2010; OFFRINGA, 2012).

2.5.4 Processo Iterativo do *SumThreshold*

O *SumThreshold* é um método iterativo, o que significa que, a cada nova etapa, o limiar T é recalculado com base nos dados restantes, excluindo-se as amostras já identificadas como RFI nas etapas anteriores. Esse processo permite que o método ajuste continuamente o limiar com base na média μ e no desvio padrão σ das subseções de $V(\nu, t)$ analisadas, convergindo para uma configuração que maximize a exclusão de RFI e minimize a remoção de dados astrofísicos.

2.5.5 Divisão do Sinal em Subsequências

Para melhorar a precisão, o *SumThreshold* segmenta o sinal $V(\nu, t)$ em subsequências, o que facilita a identificação de padrões específicos de interferência que podem ser localizados em regiões distintas de tempo ou frequência. Cada subsequência S_j é processada com uma janela deslizante que calcula μ e σ localmente, identificando os valores que ultrapassam o limiar T_j e aplicando uma máscara de RFI para essas amostras. Esse refinamento torna o método eficaz para detectar tanto RFI persistente quanto transitória.

2.5.6 Ajuste da Sensibilidade com o Parâmetro k

O parâmetro k é crucial para controlar a sensibilidade do *SumThreshold*. Em ambientes com baixa interferência, valores mais baixos de k podem ser suficientes, enquanto ambientes com alta densidade de RFI podem exigir valores maiores para evitar que ruídos legítimos sejam marcados como interferência. O processo iterativo, juntamente com o ajuste de k, assegura que o método se adapte de forma eficaz a uma ampla gama de condições de interferência.

2.5.7 Convergência e Robustez do *SumThreshold*

O *SumThreshold* utiliza a natureza estatística da Distribuição Gaussiana para reduzir a probabilidade de falsos positivos, garantindo que apenas os picos significativos de intensidade sejam detectados como RFI. Ao recalibrar μ e σ em cada iteração com base nos dados restantes, o método converge para um limiar otimizado, o que facilita uma separação robusta entre RFI e sinais astrofísicos legítimos.

Essa adaptação ao longo de $V(\nu, t)$, com a aplicação de máscaras iterativas e ajustes dinâmicos de T, torna o *SumThreshold* uma ferramenta poderosa para radioastronomia, permitindo uma análise precisa e confiável dos dados observacionais.

2.6 Filtro Gaussiano

Filtros Gaussianos são uma ferramenta essencial no processamento de imagens digitais, aplicados para suavizar detalhes indesejados e reduzir ruídos de alta frequência, enquanto preservam as principais estruturas visuais. Esses filtros são baseados na função Gaussiana, uma função matemática contínua que distribui os valores de maneira suave e simétrica em torno de um ponto central. A suavização realizada pelos filtros Gaussianos é ideal para reduzir o ruído sem perder detalhes importantes das bordas e estruturas da imagem (GONZALEZ; WOODS, 2018).

Em contextos como a astronomia, nos quais as imagens digitais geralmente são armazenadas em arquivos *Flexible Image Transport System* (FITS), a aplicação de filtros Gaussianos permite a melhoria da qualidade da imagem e facilita a detecção de fontes astrofísicas. Esses arquivos contêm dados sensíveis capturados por telescópios, e a filtragem adequada pode remover ruídos indesejados sem comprometer a integridade das informações astrofísicas relevantes (AURICH; GARTH; LINKE, 2023).

2.6.1 Fundamentos Teóricos

A função Gaussiana em duas dimensões, que é usada para construir o filtro, pode ser expressa matematicamente como:

$$G(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

Nesta equação, x e y são as coordenadas espaciais relativas ao ponto central do filtro, e σ é o desvio padrão da função Gaussiana, o qual define a largura do filtro. Quanto maior o valor de σ, maior será o efeito de suavização da imagem, pois mais pixels vizinhos serão considerados na média ponderada durante a filtragem. Esse comportamento isotrópico, em que a suavização ocorre de maneira uniforme em todas as direções, é uma das razões pelas quais os filtros Gaussianos são amplamente preferidos em várias aplicações (BURGER; BURGE, 2022).

2.6.2 Operação de Convolução

A aplicação de um filtro Gaussiano em uma imagem é realizada através de uma operação de convolução. A convolução de uma imagem $I(x, y)$ com o filtro Gaussiano $G(x, y)$ é definida como:

$$I'(x, y) = I(x, y) * G(x, y)$$

A convolução calcula uma média ponderada dos valores dos pixels em torno de um ponto específico, na qual os pesos são determinados pela função Gaussiana. Essa operação suaviza as variações bruscas de intensidade, como o ruído, enquanto preserva as estruturas maiores, como bordas e objetos importantes. É uma técnica fundamental para a pré-processamento de imagens em tarefas como segmentação, detecção de bordas e reconhecimento de padrões (GONZALEZ; WOODS, 2018).

2.6.3 Propriedades no Domínio da Frequência

Uma propriedade notável dos filtros Gaussianos é que sua transformada de Fourier também é uma função Gaussiana. Isso os torna filtros *passa-baixa*, o que significa que eles atenuam as altas frequências (associadas ao ruído) e preservam as baixas frequências, que contêm as principais características estruturais da imagem. A relação entre o desvio padrão no domínio espacial (σ_t) e no domínio da frequência (σ_f) é dada por:

$$\sigma_t \cdot \sigma_f = \frac{1}{2\pi}$$

Essa característica faz com que os filtros Gaussianos sejam adequados para aplicações que exigem suavização sem distorção significativa das bordas, como nas imagens astronômicas, em que é essencial preservar as estruturas celestes para análise detalhada (BURGER; BURGE, 2022; KUMAR et al., 2024).

3 Metodologia, Simulação e Análise dos Resultados

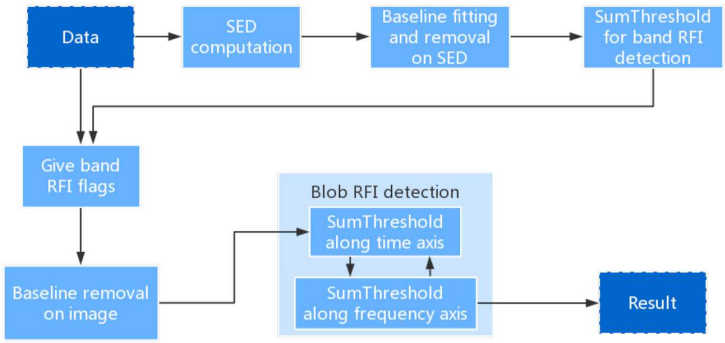
Neste capítulo, descrevem-se a metodologia empregada, as simulações realizadas e a análise dos resultados obtidos com o objetivo de mitigar a interferência nos sinais astrofísicos. Os experimentos realizados visam demonstrar a eficácia de algoritmos específicos no tratamento de dados espectrais. As formas de interferência de radiofrequência (RFI) analisadas incluem ruído senoidal, ruído rosa, ruído gaussiano, banda larga e ruído impulsivo. Cada simulação foi projetada para replicar cenários que aproximam o contexto real de interferência nas observações, permitindo, assim, testar a capacidade dos métodos ArPLS e *SumThreshold* na identificação e remoção de ruídos, além de recuperar informações da *baseline*.

O esquema ArPLS-ST combina duas abordagens centrais: o método ArPLS , que ajusta e remove a *baseline* do sinal, e o algoritmo *SumThreshold* (ST), especializado na detecção de interferências. A combinação dessas técnicas proporciona um tratamento robusto e específico para diferentes tipos de RFI.

O fluxograma do ArPLS-ST, ilustrado na Figura 3 deste estudo, organiza os procedimentos principais em quatro etapas:

- **Ajuste e remoção de *baseline* em SED**: O método ArPLS é empregado para corrigir a *baseline* de sinais espectrais (SED), minimizando os desvios causados por ruídos de fundo e preservando a integridade dos sinais de interesse.
- **SumThreshold para detecção de RFI de banda**: Este procedimento utiliza o algoritmo ST para identificar interferências distribuídas em faixas específicas de frequência, como as de banda larga ou estreita.
- **Remoção de *baseline* na imagem**: Após a correção do SED, a *baseline* também é ajustada nas imagens, aumentando a precisão na identificação das regiões contaminadas por RFI.
- **Detecção de RFI de blob**: Esta etapa, também baseada no ST, é projetada para detectar interferências pontuais e transitórias, conhecidas como *blobs* (regiões localizadas de interferência intensa).

Figura 3 – Um fluxograma do esquema ArPLS-ST.



Fonte: (ZENG et al., 2020)

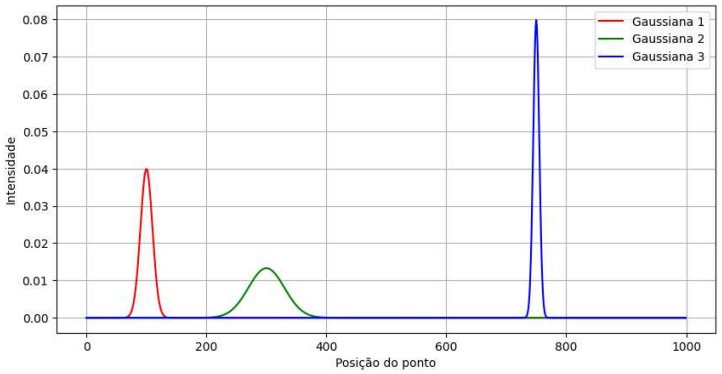
3.1 Experimento 01: Determinação e Correção da *Baseline*

A *baseline* foi determinada usando o método *ARPLS* (Penalized Least Squares Reweighted Asymmetric Smoothing), frequentemente usada para eliminar as tendências de fundo nos dados espectrais. A função `ArPLS` no arquivo `baseline1.py` destina-se à suavização penalizada assimétrica. Há um ajuste iterativo dos pesos para cada observação, de modo a manter as principais características do sinal. A suavização continua até a convergência. Portanto, a *baseline* obtida é precisa e eficaz.

Além disso, os dados corrigidos são submetidos a uma filtração inicial, na qual a *baseline* ajustada é subtraída dos dados originais. Nesse ponto, os componentes ruins dos dados são reduzidos antes da aplicação de técnicas de mitigação de de-interferência mais complexas.

Portanto, na Figura 4 estão as três gaussianas individuais que compõem o sinal sintético original. Como mostrado nas cores diferentes, cada componente representa um pico "observado" em experimentos espectroscópicos. A visualização contrastante dos picos entre si mostra as definições iniciais desses sinais antes que a *baseline* e o ruído sejam adicionados.

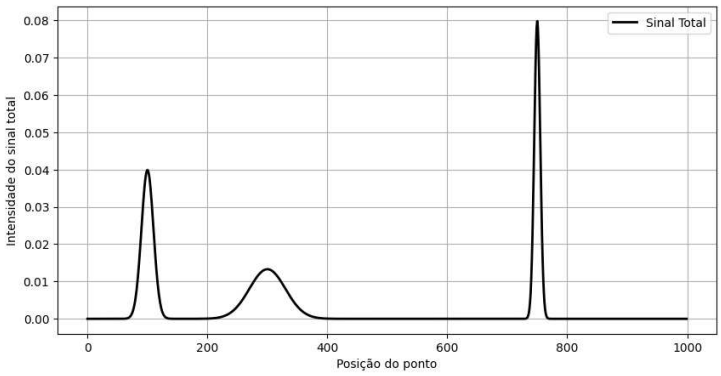
Figura 4 – Cada gaussiana representa um pico distinto do sinal sintético original.



Fonte: Própria, 2024.

A Figura 5 ilustra o sinal resultante da soma das três gaussianas, sem a adição de *baseline* ou ruído. Esse gráfico serve como uma referência para a análise dos resultados da correção, representando o sinal ideal que se deseja recuperar após o processamento.

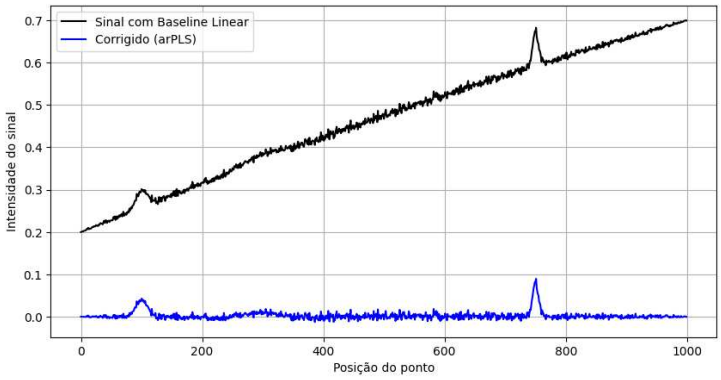
Figura 5 – Sinal total composto pelas três gaussianas, antes da adição da *baseline* e ruído.



S Fonte: Própria, 2024.

A Figura 6 retrata a versão contaminada do sinal com um *baseline* linear, junto com a versão corrigida usando o algoritmo *arPLS*. A linha preta retrata a versão contaminada enquanto a azul é o resultado após a execução do *arPLS*. Observa-se que o método *arPLS* foi eficaz na remoção da interferência, com os picos sendo restaurados e sem incisos notáveis que comprometem o resultado final.

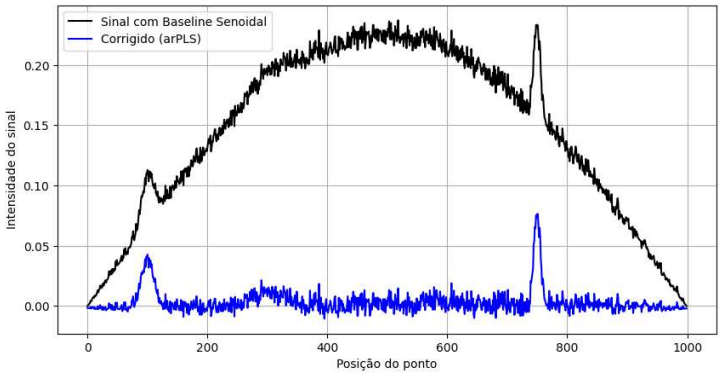
Figura 6 – Sinal contaminado com *baseline* linear e sua correção pelo arPLS.



Fonte: Própria, 2024.

Na Figura 7, encontra-se o sinal com um *baseline* senoidal e sua forma corrigida após o arPLS. Como o *baseline* possui um perfil senoidal, mantém-se o desafio extra, imitando flutuações periódicas num ambiente experimental. Após a correção com o arPLS, observa-se que o *baseline* foi removido quase completamente, com uma recuperação precisa dos picos.

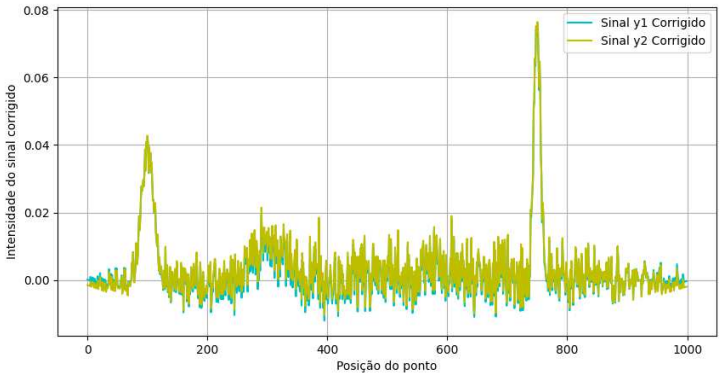
Figura 7 – Sinal contaminado com *baseline* senoidal e sua correção pelo arPLS.



Fonte: Própria, 2024.

A Figura 8 é apresentado os sinais y_1 e y_2 após correção com arPLS. As linhas azul e amarela representam, respectivamente, os sinais cujo *baseline* foi corrigido linearmente e corrigido com uma função senoidal. Ambos os sinais apresentam picos bem definidos, próximos ao sinal de referência ideal. Portanto, o arPLS foi bem-sucedido independentemente da forma do *baseline*.

Figura 8 – Sinais y_1 e y_2 após correção com arPLS.



Fonte: Própria, 2024.

3.2 Experimento 02: Ruído RFI Senoidal

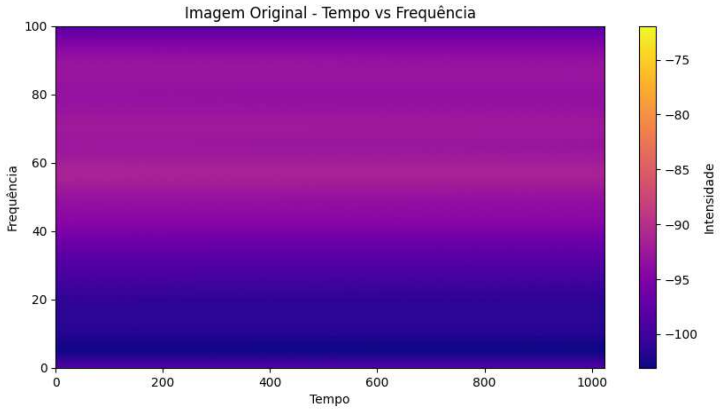
Neste experimento, o objetivo foi simular e entender, por meio de um conjunto de dados espectrais, a interferência de radiofrequência utilizando ruído senoidal. Por meio da contaminação seletiva, é possível adicionar o ruído em colunas das imagens para simular a situação em que a interferência externa somente ocorre em determinadas regiões do sinal. Com a contaminação, as imagens do sinal original e do sinal contaminado são comparadas, avaliando a identificação da RFI pelo método arPLS-ST.

3.2.1 Procedimento Experimental

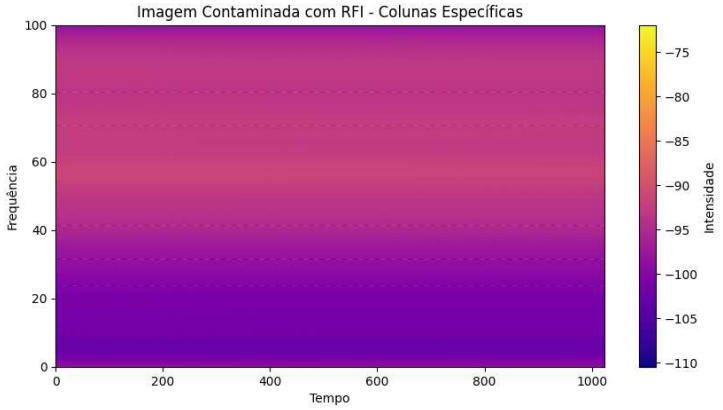
A princípio, é aberto e processado um arquivo.fits contendo dados de espectros reais para selecionar as colunas a serem contaminadas. Em seguida, é gerado um ruído senoidal com uma frequência de 50 Hz e aplicado às colunas especificamente escolhidas. A amplitude do ruído é posteriormente calculada como 10% do valor máximo da linha e somada aos dados. Em seguida, os dados contaminados são salvos em um novo arquivo.fits para preservar a integridade dos dados precedentes. Em segundo lugar, para ver o efeito da contaminação, são criados dois gráficos das colunas antes e depois da contaminação e um indicador adicional de quais seções a contaminar. Para tal, foi aplicado o método arPLS para correção do *baseline* no sinal de interesse, seguido pela técnica *SumThreshold* para mascarar as áreas afetadas por RFI.

A Figura 9(a) apresenta a imagem contínua antes do acréscimo de ruído, sem nenhuma interferência de RFI, ou seja, a imagem "ideal" que se deseja restaurar depois da remoção da interferência. Por outro lado, a Figura 9(b) indica o sinal após a contaminação de ruído senoidal nas colunas especificadas, e a figura descreve o padrão de interferência que se desenvolve e danifica a integridade do sinal e dos dados.

Figura 9 – Imagem sem ruído e com ruído em colunas especificas.



(a) Imagem sem o ruído senoidal.

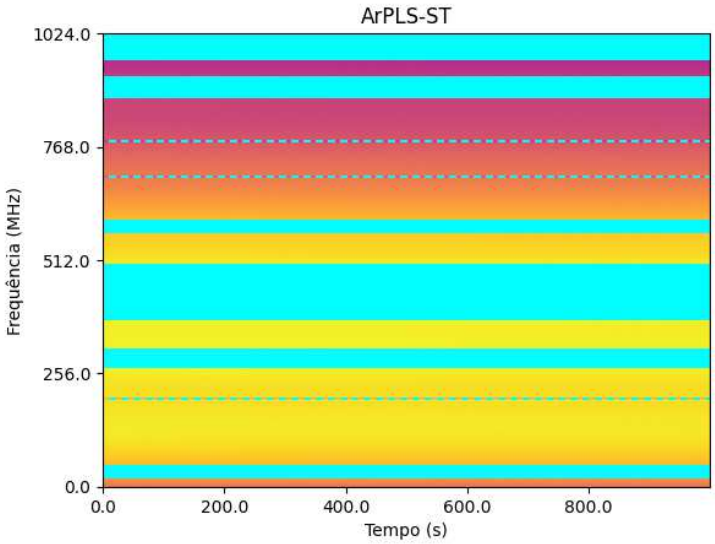


(b) Imagem com o ruído senoidal.

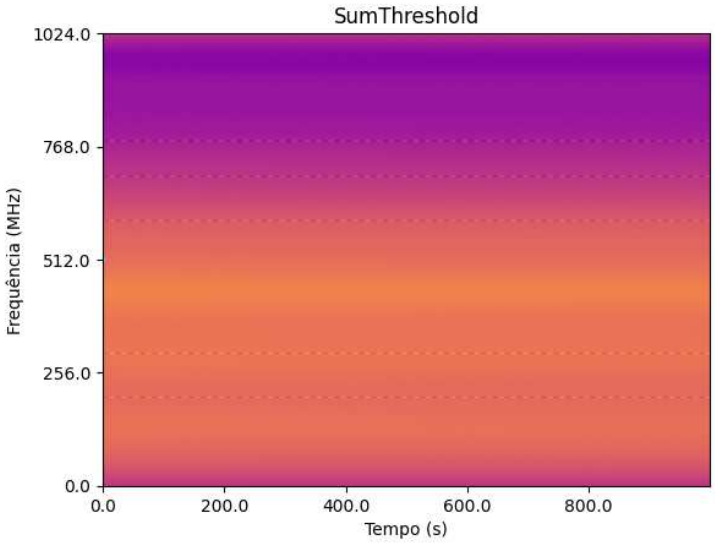
Fonte: Própria, 2024.

A Figura 10(a) apresenta o resultado do mascaramento de RFI com o uso do método arPLS-ST, que aplica uma ponderação assimétrica que modela e corrige o *baseline*, preservando as características principais do sinal e diminuindo as interferências da mesma forma. Para melhor visualização, as áreas em que o ruído foi detectado é adicionado uma mascara com uma cor ciano. Por outro lado, a Figura 10(b) mostra o resultado técnica *SumThreshold*, que mascara pontos de alta intensidade. Neste caso, não houve adição de mascara caracterizando a não identificação do ruído, isto porque ele já foi detectado no arPLS-ST.

Figura 10 – Imagens com mascara identificando a presença de ruído.



(a) arPLS-ST.



(b) SumThreshold.

Fonte: Própria, 2024.

3.3 Experimento 03: RFI com Ruído Rosa

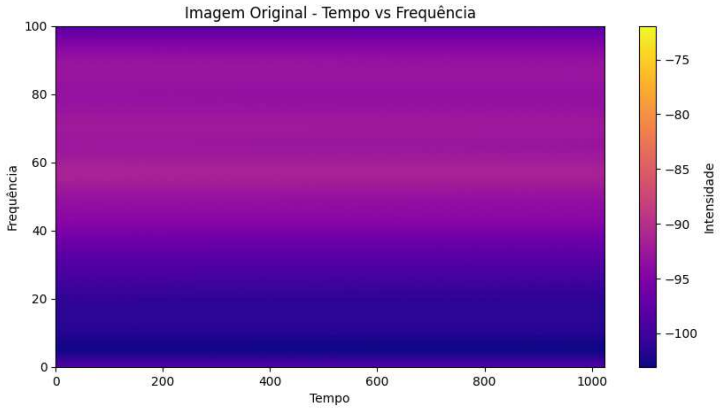
O ruído rosa, expresso pela intensidade inversamente decaindo com a frequência, o que é um denominador comum de muitas formas de sinal de interferência de radiofrequência. O ruído foi gerado com a frequência de 50 Hz e amplitude correspondente a 12% do valor máximo do ponto do sinal e adicionado a colunas particulares de dados, o que designava a interferência esparsa e intermitente.

Após a aplicação do ruído rosa, o sinal original contaminado foi salvo em um novo

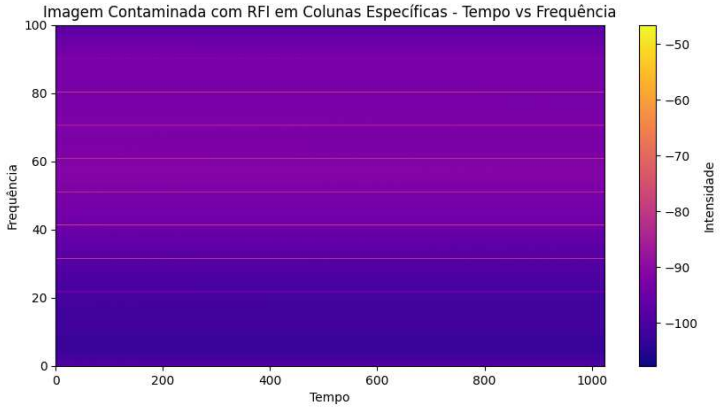
arquivo `.fits`, utilizado para análise posterior e comparação com os dados originais.

Na Figura 11(a) está sendo ilustrado o sinal original, sem contaminação por RFI, apresentando uma intensidade uniforme distribuída ao longo do eixo da frequência sem interferência clara. Esta figura é a base de referência para o experimento, retratando o cenário ideal a ser recuperado depois da correção do *baseline*. Em comparação, na Figura 11(b), o sinal é contaminado por ruído rosa em colunas intercaladas, depositado em um padrão intermitente de interferência, o que apresenta efeitos maiores que o normal, resultando em módulos recortados. Tal padrão é indicativo do impacto da contaminação do RFI no sinal.

Figura 11 – Comparação entre o sinal original e o sinal com ruido rosa



(a) Imagem sem o ruído rosa.



(b) Imagem com o ruído rosa.

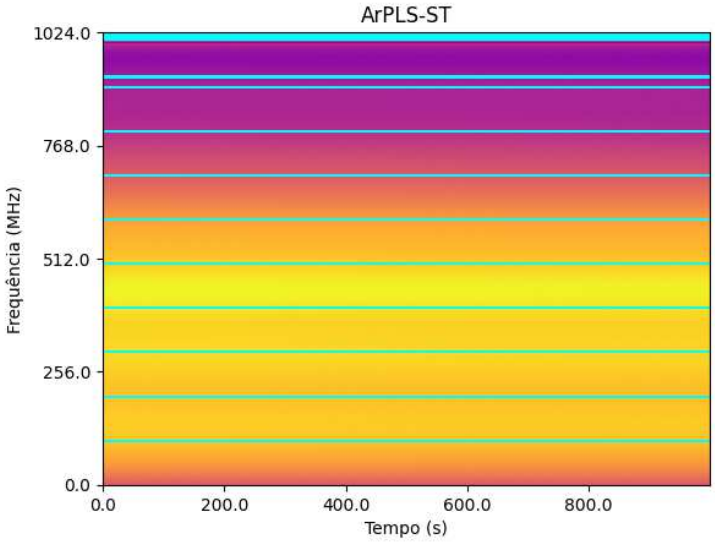
Fonte: Própria, 2024.

A Figura 12(a) representa o resultado do mascaramento de RFI através do método arPLS-ST. O método emprega pesos assimétricos empregados na modelagem do baseline, permitindo a preservação das áreas de interesse e a eliminação da interferência de forma seletiva. É evidente a coloração ciano devidamente espaçada, nas quais o ruído foi captado. É possível observar que o algoritmo de arPLS-ST identificou adequadamente a maioria

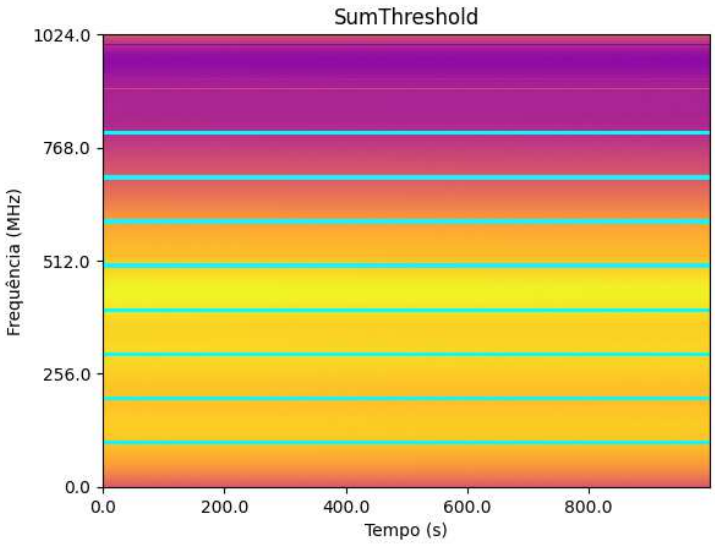
das regiões com interferência e as devidamente mascarou.

A Figura 12(b) é ilustrado a máscara produzida a partir do método *SumThreshold*, que realça os pontos muito intensos. Entretanto, apesar de funcionar no rastreamento de interferências, ele pode marcar excessivamente e impactar os dados reais.

Figura 12 – Mascara identificando a presença de ruído nas colunas intercaladas.



(a) arPLS-ST.



(b) SumThreshold.

Fonte: Própria, 2024.

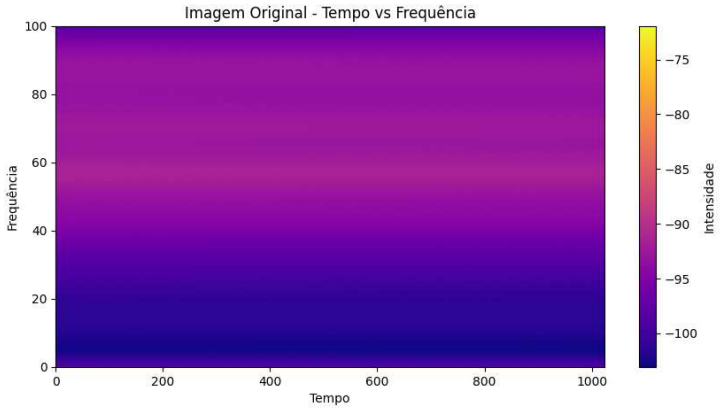
3.4 Experimento 04: Ruído Gaussiano

No presente experimento, foi adicionado ruído gaussiano ao sinal original, o que simula tanto a interferência que tem uma distribuição normal quanto a flutuação aleatória,

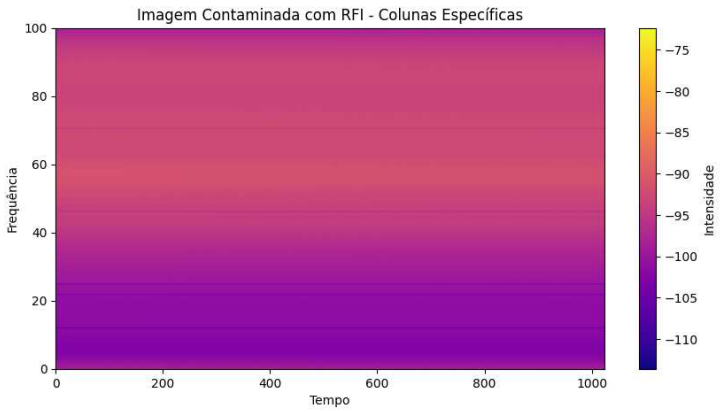
pois são comuns em interferência de radiofrequência. A amplitude de ruído é então estabelecida com 12% do valor de pico e média zero e desvio padrão igual a 1. O ruído adicionado a colunas específicas é internamente gasto com a simulação de interferência específica como interferência localizada.

Na Figura 13(a), observa-se o sinal original, não contaminado por RFI, com uma distribuição regular da intensidade em frequência e nenhuma interferência visual, o que serve como um sinal de referência para examinar o efeito do ruído gaussiano. Por outro lado, a Figura 13(b) apresenta o sinal após a contaminação de colunas selecionadas com RFI gaussiano, no qual as flutuações de intensidade nos locais contaminados começam a aparecer, refletindo o padrão intermitente da fonte característica do ruído gaussiano.

Figura 13 – Comparação entre o sinal original e o sinal com ruido gaussiano



(a) Imagem sem o ruído gaussiano.



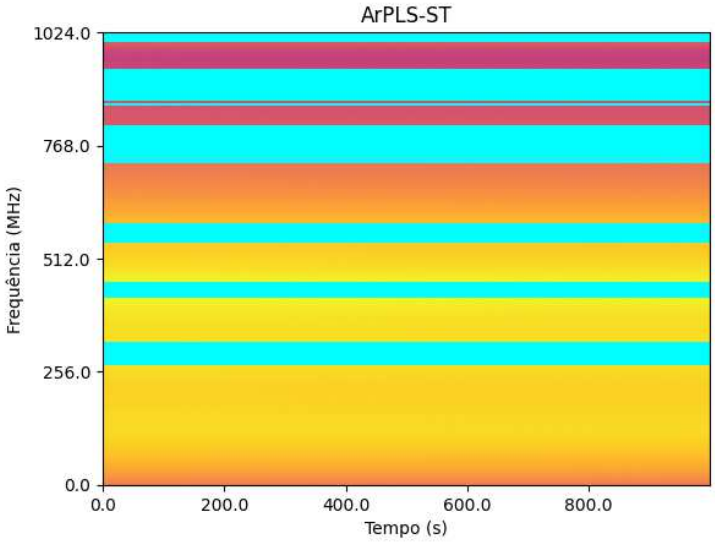
(b) Imagem com o ruído gaussiano.

Fonte: Própria, 2024.

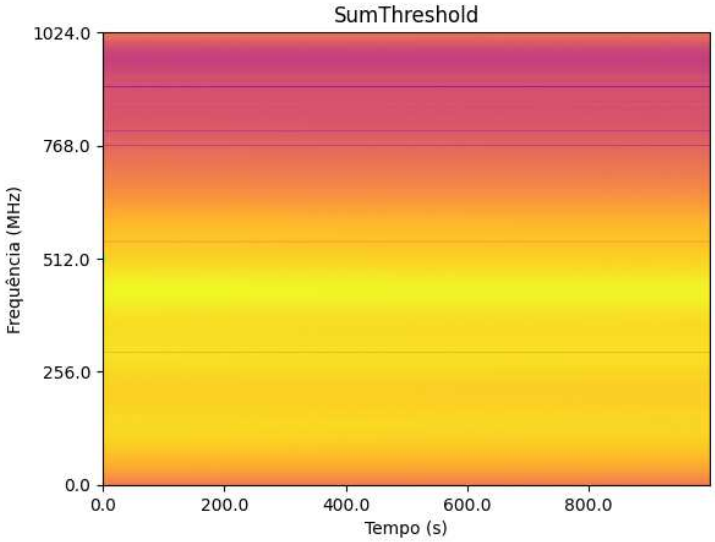
A Figura 14(a) apresenta o resultado do arPLS-ST aplicado ao trabalho de remoção de *baseline*, com a região de interferência colorida em ciano. Pode-se ver que o método é capaz de priorizar as áreas contaminadas por ruído gaussiano, já que o *baseline* é removido de maneira precisa e os picos de sinal genuínos são mantidos. Não é observado o efeito do

mascaramento na Figura 14(b) após a passagem pelo *SumThreshold*.

Figura 14 – Mascara identificando a presença de ruído nas colunas intercaladas.



(a) arPLS-ST.



(b) SumThreshold.

Fonte: Própria, 2024.

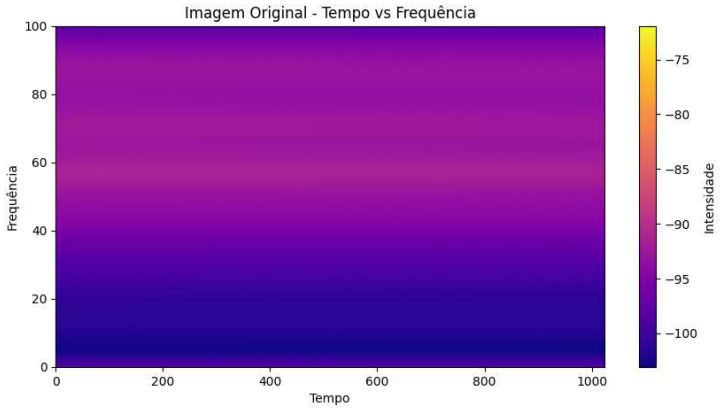
3.5 Experimento 05: Ruído de Banda Larga

Já o ruído de banda larga foi simulado para representar uma outra interferência, mas que ocorre em várias frequências simultâneas, muito comum em ambientes de comunicação. Foi gerado com uma frequência central de 60 Hz e uma largura de banda de 50 Hz, assim, uma faixa de 35 Hz até 85 Hz foi abordada, com uma amplitude de 10% da intensidade máxima dos dados originais. Por fim, esse tipo de ruído foi aplicado em algumas colunas,

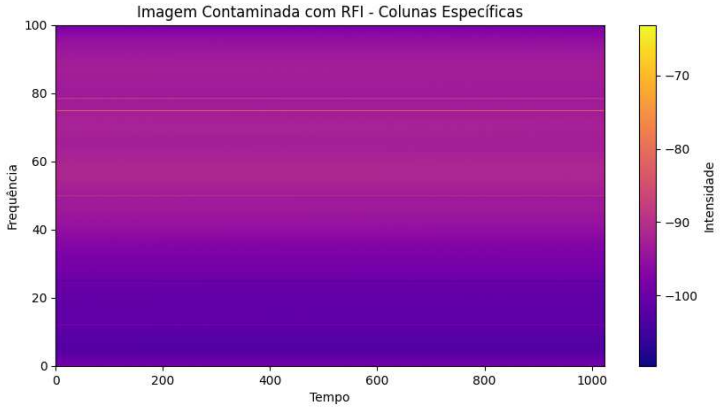
o que correspondia a uma interferência em alguma região delimitada dos dados.

A Figura 15(a) apresenta os dados espectrais originais, sem a introdução de ruído, com uma distribuição uniforme de intensidades ao longo dos eixos de tempo e frequência. Em contraste, a Figura 15(b) ilustra os dados espectrais após a adição de ruído de banda larga nas colunas especificadas, evidenciando a distorção nas intensidades ao longo da faixa de frequências.

Figura 15 – Comparação entre o sinal original e o sinal com ruido banda larga



(a) Imagem sem o ruído banda larga.

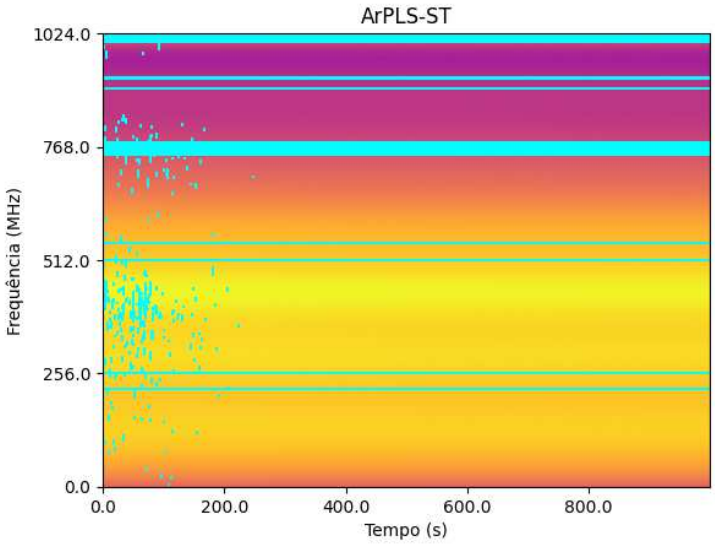


(b) Imagem com o ruído banda larga.

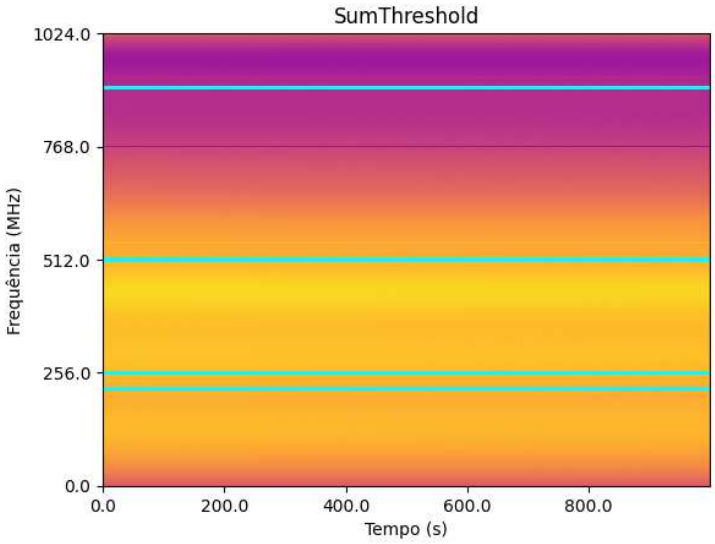
Fonte: Própria, 2024.

O resultado da aplicação da técnica ArPLP-ST, para mascaramento da contribuição do ruído banda larga ao sinal, é mostrado na Figura 16(a), além de ser marcado com a mascara a região do ruído que foi acrescentado é nota-se a possível presença do ruído do tipo *Blob*, caracterizado por uma granulação espalhada. A Figura 16(b) nota-se a máscara gerada pelo *SumThreshold* no mesmo conjunto de dados contaminados, não verifica-se o ruído do tipo *Blob*.

Figura 16 – Mascara identificando a presença de ruído nas colunas intercaladas.



(a) arPLS-ST.



(b) SumThreshold.

Fonte: Própria, 2024.

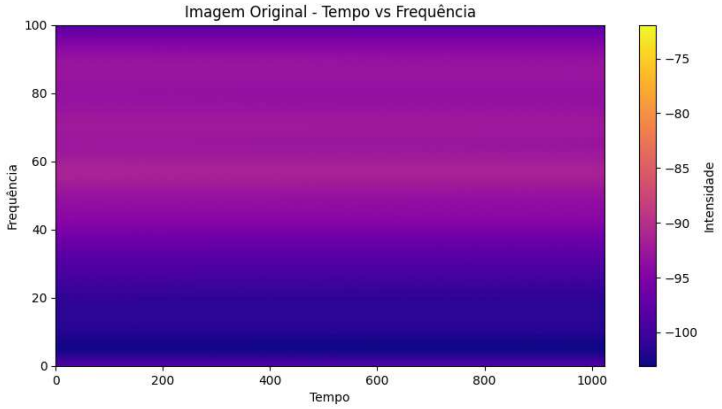
3.6 Experimento 06: Ruído Impulsivo

O ruído impulsivo foi adicionado para simular interferências de alta intensidade e baixa frequência, características de fontes artificiais. Utilizou-se uma distribuição binomial com probabilidade de 0,6 para introduzir impulsos aleatórios, em uma amplitude média de 12% do valor máximo dos dados. Esse ruído foi aplicado em colunas específicas, simulando interferências esparsas.

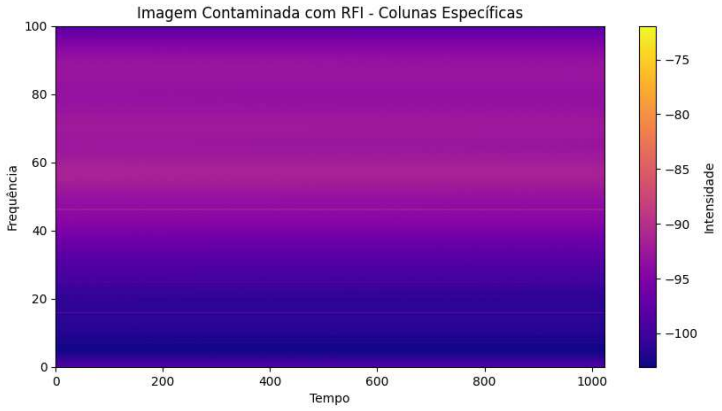
A Figura 17(a) apresenta o sinal original sem contaminação por ruído impulsivo,

funcionando como referência para as comparações posteriores à introdução da interferência. Já a Figura 17(b) mostra o sinal após a adição do ruído impulsivo em colunas específicas, simulando interferências de alta intensidade em áreas delimitadas.

Figura 17 – Comparação entre o sinal original e o sinal com ruido impulsivo



(a) Imagem sem o ruído impulsivo.

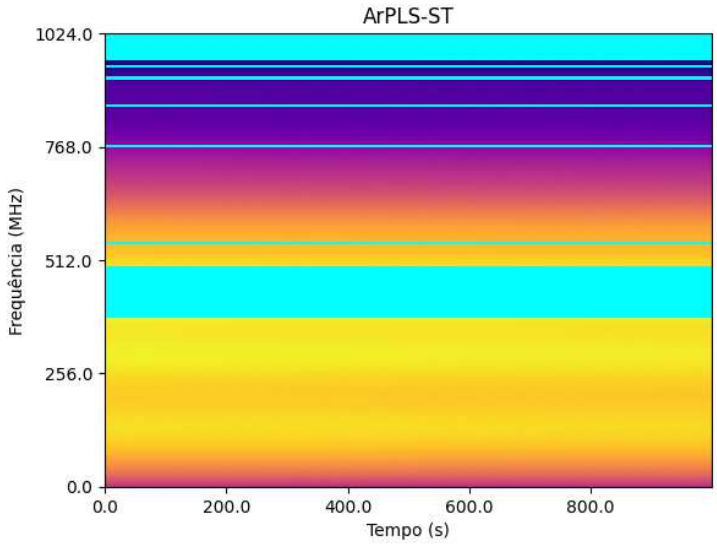


(b) Imagem com o ruído impulsivo.

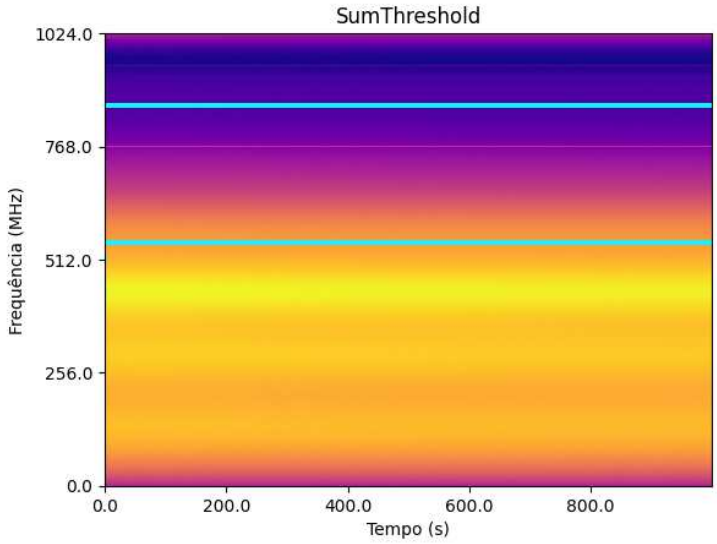
Fonte: Própria, 2024.

A Figura 18(a) apresenta o resultado da correção de *baseline* e do mascaramento de ruído impulsivo pelo método arPLS-ST, como pode ser visto em ciano. Já a Figura 18(b) mostra a máscara gerada pelo método *SumThreshold*, sendo visto apenas duas linhas destacadas em ciano.

Figura 18 – Mascara identificando a presença de ruído nas colunas intercaladas.



(a) arPLS-ST.



(b) SumThreshold.

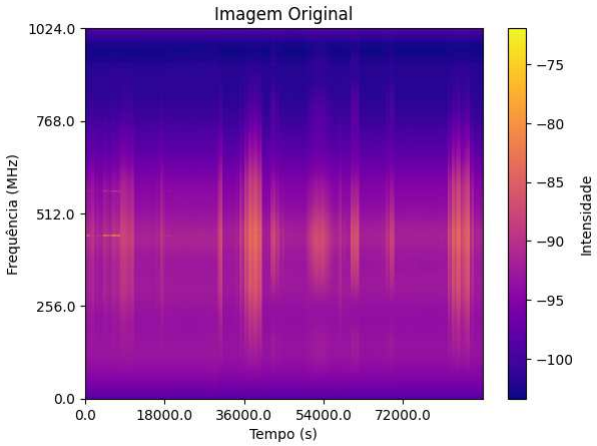
Fonte: Própria, 2024.

3.7 Experimento 07: Verificação da *baseline* usando o método arPLS

Este experimento visa avaliar a eficácia do método arPLS para a correção da *baseline* em dados astrofísicos contaminados por interferências de radiofrequência (RFI). Foram geradas e interpretadas várias visualizações que ilustram o estado dos dados antes e após o processo de mitigação, destacando as melhorias na qualidade do sinal para análises subsequentes.

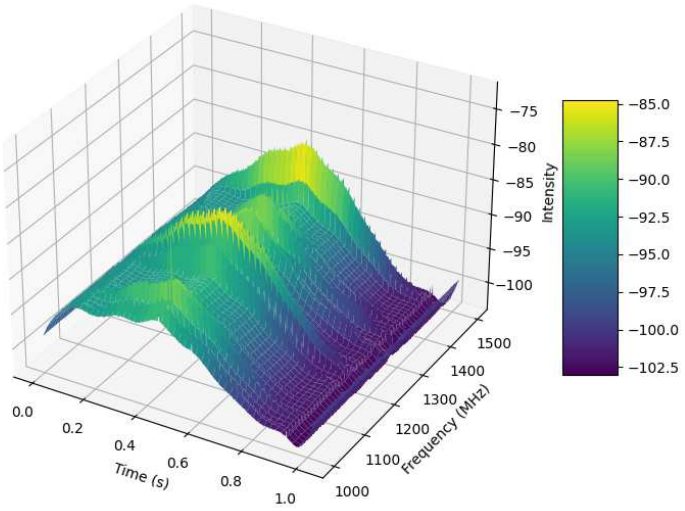
Na Figura 19, apresenta-se uma concatenação de várias imagens representando os dados iniciais, antes de qualquer procedimento de mitigação. Nela, é possível observar a distribuição da intensidade dos sinais em função do tempo e da frequência, com padrões que evidenciam as interferências de radiofrequência (RFI). Essa visualização atua como referência para examinar a amplitude e a distribuição das RFI, servindo como base para avaliar a eficácia das técnicas de tratamento aplicadas.

Figura 19 – Visualização dos dados astrofísicos originais, antes do tratamento.



A representação tridimensional dos dados originais, ilustrada na Figura 20 , utiliza os eixos de tempo, frequência e intensidade para destacar picos de interferência e áreas de maior intensidade, sinalizando a presença de RFI. Esse modelo permite uma observação detalhada das oscilações e padrões de interferência ao longo do espectro.

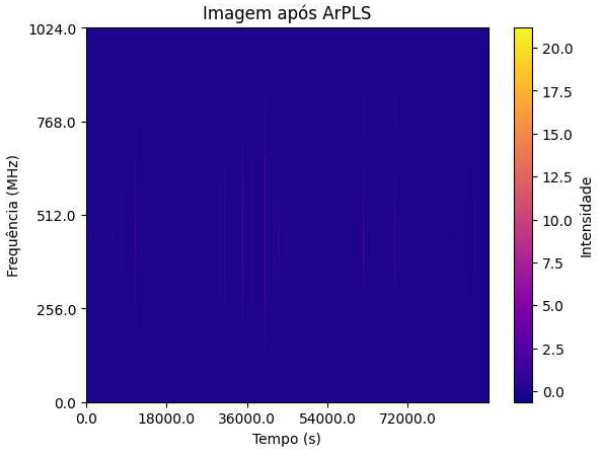
Figura 20 – Superfície tridimensional dos dados originais, evidenciando os picos de interferência.



Após a aplicação do método arPLS, Figura 21 , observa-se uma suavização da

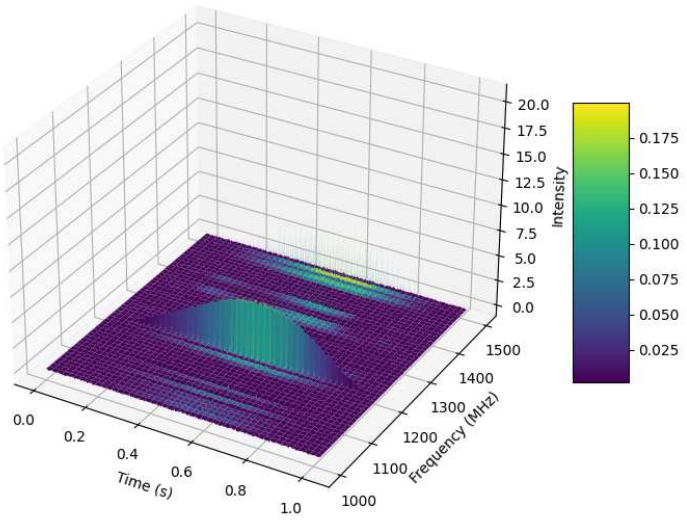
baseline dos sinais e uma redução das interferências de baixa frequência. Em comparação com a imagem inicial, as variações indesejadas diminuem significativamente, preservando os sinais astrofísicos de interesse. O arPLS demonstra, assim, sua eficácia em minimizar desvios na *baseline* e atenuar o impacto das RFI.

Figura 21 – Dados após a aplicação do método arPLS para correção da *baseline*.



Finalmente, a superfície tridimensional dos dados, ilustrada na Figura 22, revela o efeito do arPLS na correção da *baseline*. Após o tratamento, a superfície apresenta uma uniformidade maior, com uma *baseline* estabilizada e sem os picos abruptos característicos de RFI. Essa regularidade evidencia a eficácia do método em preservar os sinais astrofísicos, removendo seletivamente apenas as interferências indesejadas.

Figura 22 – Superfície tridimensional dos dados corrigidos pelo arPLS, com redução das interferências.



Considerações finais

Este estudo investigou a aplicação de técnicas de correção de *baseline* e mitigação de interferência de radiofrequência (RFI) no radiotelescópio BINGO, com foco nos métodos arPLS e *SumThreshold*. Os experimentos realizados demonstraram que ambos os métodos são efetivos na redução de interferências e na preservação dos sinais astrofísicos relevantes. A utilização dessas técnicas foi fundamental para garantir a integridade dos dados coletados, especialmente em um ambiente saturado por sinais artificiais, que pode comprometer a precisão das análises em radioastronomia.

Em resumo, os resultados indicam que os métodos empregados apresentaram desempenho satisfatório na mitigação de interferências e na preservação dos sinais astrofísicos de interesse. O método arPLS destacou-se por sua capacidade de suavizar a *baseline* sem comprometer componentes significativos do sinal, enquanto o *SumThreshold* demonstrou eficácia na identificação de ruídos em regiões específicas, especialmente em áreas de alta intensidade. Dessa forma, a combinação dessas técnicas reforça a aplicabilidade desses métodos para aprimorar a qualidade das observações em radioastronomia.

Diante do exposto, o presente estudo reforça a necessidade de utilizar métodos de mitigação de RFI na captura de dados durante observações radioastronômicas, contribuindo para o desenvolvimento de abordagens que aumentem a robustez desses dados. A partir das contribuições realizadas, sugere-se que trabalhos futuros explorem a integração dos métodos propostos com outras técnicas de processamento de sinais, ampliando as possibilidades de filtragem e classificação de eventos astrofísicos. Esse aprimoramento fortaleceria a coleta de dados pelo radiotelescópio BINGO e impulsionaria a pesquisa em radioastronomia.

Contudo, vale destacar que não foi possível realizar um experimento que simulasse o ruído do tipo *blob*, o que impediu a avaliação dos métodos arPLS e *SumThreshold* em relação a esse tipo específico de interferência. Essa limitação restringe a análise completa da eficácia das técnicas no tratamento de ruídos transitórios e pontuais, características comuns e desafiadoras em observações de radioastronomia. Esse aspecto poderá ser abordado em estudos futuros, possibilitando uma compreensão mais ampla dos métodos em diferentes cenários de RFI.

Referências

ABDALLA, E. et al. The bingo project i: Baryon acoustic oscillations from integrated neutral gas observations. *Astronomy & Astrophysics*, EDP Sciences, v. 664, p. A14, ago. 2022. ISSN 1432-0746. Disponível em: <http://dx.doi.org/10.1051/0004-6361/202140883>.

AURICH, J. C.; GARTH, C.; LINKE, B. S. *Proceedings of the 3rd Conference on Physical Modeling for Virtual Manufacturing Systems and Processes*. [S.l.]: Springer, 2023. ISBN 3031357787; 9783031357787.

BAAN, W. A. et al. Rfi mitigation in radio astronomy. *Proceedings of Science*, RFI2010, 2010.

BAEK, S.-J. et al. Baseline correction using asymmetrically reweighted penalized least squares smoothing. *Analyst*, The Royal Society of Chemistry, v. 140, p. 250–257, 2015. Disponível em: <http://dx.doi.org/10.1039/C4AN01061B>.

BUKE, B. F.; GRAHAM-SMITH, F.; WILKINSON, P. N. *An Introduction to Radio Astronomy*. 4. ed. New Yourk: Cambridge, 2019.

BURGER, W.; BURGE, M. J. *Digital Image Processing: An Algorithmic Introduction*. 3rd. ed. [S.l.]: Springer, 2022.

CONDON, J. J.; RANSOM, S. M. *Essential Radio Astronomy*. National Radio Astronomy Observatory, 2024. Disponível em: <https://www.cv.nrao.edu/~sransom/web/Ch1.html>.

EILERS, P.; BOELENS, H. Baseline correction with asymmetric least squares smoothing. *Unpubl. Manuscr*, 11 2005.

FRANCO, N. M. B. *Cálculo Numérico*. 1. ed. São Paulo: PEARSON, 2006.

GARY, D. *An Introduction to Radio Astronomy*. [S.l.]: NJIT, 2024.

GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing*. 4th. ed. Pearson, 2018. Global Edition. ISBN 978-1-292-22304-9. Disponível em: <https://www.pearsonglobaleditions.com>.

KUMAR, S. et al. *Fourth Congress on Intelligent Systems: CIS 2023, Volume 1 (Lecture Notes in Networks and Systems, 868)*. 2024. ed. [S.l.]: Springer, 2024.

LEVEQUE, R. J. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. 1. ed. [S.l.]: SIAM, 2007.

OFFRINGA, A. et al. A lofar rfi detection pipeline and its first results. In: *RFI mitigation workshop*. [S.l.: s.n.], 2010. p. 1–10.

OFFRINGA, A. R. *Algorithms for Radio Interference Detection and Removal*. Tese (Doutorado) — University of Groningen, 2012.

OFFRINGA, A. R. et al. Post-correlation radio frequency interference classification methods. *Monthly Notices of the Royal Astronomical Society*, v. 405, n. 1, p. 155–167, 2010.

RAMALHO, R. N. *Impactos de Satélites GNSS no Radiotelescópio BINGO*. Dissertação (Mestrado) — Universidade Federal de Campina Grande, 2023.

THOMPSON, A.; MORAN, J.; JR, G. S. Interferometry and synthesis in radio astronomy. *Springer*, 2014.

ZAINUD-DIN, Z. et al. The study on the effect of population density on radio frequency interference (rfi) analysis on dynamic spectrum at selected callisto stations. *Physical Sciences and Technology*, v. 8, n. 1-2, p. 53–61, 2021.

ZENG, Q. et al. Radio frequency interference mitigation based on the asymmetrically reweighted penalized least squares and sumthreshold method. *Monthly Notices of the Royal Astronomical Society*, v. 500, p. 2969–2978, 12 2020.

ZHANG, Y.; LI, J.; LIU, C. False positives in the search for extraterrestrial technosignatures: A cautionary tale. *Astronomical Journal*, v. 164, n. 2, p. 34, 2022.

Anexos

ANEXO A – Códigos utilizados

Os arquivos fornecidos têm um propósito em comum: processar dados astronômicos, especificamente dados de rádio obtidos de telescópios, para identificar e remover interferências de radiofrequência (RFI). O objetivo do código é aplicar técnicas de mitigação de RFI, utilizando métodos como *arPLS* (Penalização Assimétrica Reponderada de Mínimos Quadrados) e *SumThreshold*, além de exibir os resultados em gráficos.

Os arquivos principais são:

- **baseline1.py**: Contém funções para suavização de dados, como o método *arPLS* e um filtro gaussiano.
- **mitigation1.py**: Implementa métodos para mitigação de RFI, como *SumThreshold* e máscaras de RFI.
- **main1.py**: Arquivo que executa o código principal, chamando as funções dos outros arquivos para processar e exibir os dados.

Funções no Arquivo `baseline1.py`

Função `arPLS(y, lam=1e4, ratio=0.05, itermax=10)`

- **Objetivo**: Estimar a *baseline* de um conjunto de dados utilizando o método *arPLS*, uma técnica iterativa que suaviza o fundo dos dados, penalizando mais fortemente as regiões que desviam do comportamento geral.
- **Parâmetros**:
  - `y`: Dados de entrada (vetor ou curva) dos quais a *baseline* deve ser removida.
  - `lam`: Parâmetro de suavização. Quanto maior, mais suave será a *baseline* estimada.
  - `ratio`: Critério de convergência. Controla a mudança permitida nos pesos entre iterações.
  - `itermax`: Número máximo de iterações do algoritmo.
- **Conexão**: Essa função é chamada dentro do método `arPLS_mask` no arquivo `mitigation1.py` para corrigir a *baseline* dos dados antes de aplicar técnicas de mitigação de RFI.

```
def ArPLS(y, lam=1e4, ratio=0.05, itermax=10):
    '''
    Copiado de https://irfpy.irf.se/projects/ica/_modules/irfpy/ica/baseline.html

    Correção de \textit{baseline} usando suavização penalizada assimetricamente
    reponderada de mínimos quadrados
    Sung-June Baek, Aaron Park, Young-Jin Ahna e Jaebum Choo,
    Analyst, 2015, 140, 250 (2015)

    Entradas:
        y:
            dados de entrada (ou seja, curva SED)
        lam:
            parâmetro que pode ser ajustado pelo usuário. Quanto maior lambda,
            mais suave será o fundo resultante, z
        ratio:
            ponderação das variações: 0 < ratio < 1, valores menores permitem menos valores negativos
        itermax:
            número máximo de iterações
    Saída:
        o vetor de fundo ajustado
    '''

    N = len(y)  # Pega o tamanho dos dados
    D = sparse.eye(N, format='csc')  # Cria uma matriz identidade esparsa (grande, com muitos zeros)
    D = D[1:] - D[:-1]  # Calcula as diferenças entre os pontos
    D = D[1:] - D[:-1]  # Faz isso duas vezes para melhorar a suavidade

    D = D.T  # Transpõe a matriz
    w = np.ones(N)  # Inicializa os pesos com 1
    for i in range(itermax):  # Loop para ajustar a suavização
        W = sparse.diags(w, 0, shape=(N, N))  # Cria uma matriz de pesos
        Z = W + lam * D.dot(D.T)  # Aplica os pesos na matriz de suavização
        z = spsolve(Z, w * y)  # Resolve o sistema de equações
        d = y - z  # Calcula a diferença entre os dados originais e o fundo suavizado
        dn = d[d < 0]  # Seleciona as partes negativas
        m = np.mean(dn)  # Calcula a média das partes negativas
        s = np.std(dn)  # Calcula o desvio padrão das partes negativas
        wt = 1. / (1 + np.exp(2 * (d - (2 * s - m)) / s))  # Atualiza os pesos com base nos desvios
```

```
40          if np.linalg.norm(w - wt) / np.linalg.norm(w) < ratio:  #
    Verifica se deve parar o loop
41              break
42          w = wt  # Atualiza os pesos para a próxima iteração
43
44      return z  # Retorna o fundo suavizado
```

Função `gaussian_filter(V, mask, M=40, N=20, sigma_m=0.5, sigma_n=0.5)`

- **Objetivo**: Aplicar um filtro gaussiano em uma matriz de dados, com a opção de ignorar valores mascarados.
- **Parâmetros**:
  - `V`: Matriz de dados a ser suavizada.
  - `mask`: Matriz booleana indicando quais pontos devem ser ignorados na suavização.
  - `M`, `N`: Dimensões do kernel gaussiano (janelas de suavização) nos eixos horizontal e vertical, respectivamente.
  - `sigma_m`, `sigma_n`: Desvios padrão do kernel gaussiano nos eixos horizontal e vertical.
- **Conexão**: Essa função é utilizada no processo de suavização dentro do método `st_mask` no arquivo `mitigation1.py` para remover o fundo dos dados antes da aplicação de limiarização (*thresholding*).

```
1  def gaussian_filter(V, mask, M=40, N=20, sigma_m=0.5, sigma_n=0.5):
2      """
3      Aplica um filtro gaussiano (suavização) na matriz fornecida, levando
       em conta os valores mascarados.
4      Copiado de https://github.com/cosmo-ethz/seek/blob/master/seek/utils
       /filter.py
5
6      :param V: a matriz de valores a ser suavizada
7      :param mask: matriz booleana que define os valores mascarados
8      :param M: tamanho da janela do kernel no eixo=1
9      :param N: tamanho da janela do kernel no eixo=0
10     :param sigma_m: sigma do kernel no eixo=1
11     :param sigma_n: sigma do kernel no eixo=0
12
13     :return vs: a matriz suavizada
14     """
15
16     def wd(n, m, sigma_n, sigma_m):
```

```
        return np.exp(-n**2/(2*sigma_n**2) - m**2/(2*sigma_m**2))  # Fun
    ção para calcular o peso gaussiano

    # Adiciona bordas à matriz original
    Vp = np.zeros((V.shape[0]+N, V.shape[1]+M))
    Vp[N//2:-N//2,M//2:-M//2] = V[:]  # Preenche o meio da matriz com os
    valores originais

    # Cria uma máscara estendida
    Wfp = np.zeros((V.shape[0]+N, V.shape[1]+M))
    Wfp[N//2:-N//2,M//2:-M//2] = ~mask[:]  # Inverte a máscara de
    entrada
    Vh = np.zeros((V.shape[0]+N, V.shape[1]+M))  # Cria uma matriz para
    armazenar os valores suavizados
    Vh2 = np.zeros((V.shape[0]+N, V.shape[1]+M))  # Outra matriz para o
    segundo passo de suavização

    n = np.arange(-N/2, N/2+1)  # Define os valores para o eixo N
    m = np.arange(-M/2, M/2+1)  # Define os valores para o eixo M
    kernel_0 = wd(n, 0, sigma_n=sigma_n, sigma_m=sigma_m).T  # Calcula o
    kernel gaussiano para o eixo N
    kernel_1 = wd(0, m, sigma_n=sigma_n, sigma_m=sigma_m).T  # Calcula o
    kernel gaussiano para o eixo M

    # Aplica o filtro gaussiano nos dados
    Vh = _gaussian_filter(Vp, V.shape[0], V.shape[1], Wfp, mask, Vh, Vh2
    , kernel_0, kernel_1, M, N)
    Vh = Vh[N//2:-N//2,M//2:-M//2]  # Remove as bordas
    Vh[mask] = V[mask]  # Restaura os valores mascarados originais
    return Vh  # Retorna a matriz suavizada
```

Funções no Arquivo `mitigation1.py`

Função `arpls_mask(data, eta_i)`

- **Objetivo**: Aplicar o método *ArPLS* nos dados para estimar a *baseline* e, em seguida, identificar a RFI aplicando o método de limiarização com base no valor de desvio padrão (*SumThreshold*).
- **Parâmetros**:
  - `data`: Matriz de dados de entrada, contendo o sinal e a RFI.
  - `eta_i`: Lista de sensibilidades para a aplicação do *SumThreshold*.
- **Conexão**: Essa função utiliza o método *ArPLS* (importado de `baseline1.py`) para estimar a *baseline*, suaviza os dados e aplica uma máscara para detectar interferências

de RFI. O resultado é uma máscara binária que indica a presença de RFI. É chamada no arquivo `main1.py` para aplicar a primeira técnica de mitigação.

```
def arpls_mask(data, eta_i=[0.5, 0.55, 0.62, 0.75, 1]):
    """
    Cria uma máscara para cobrir a RFI nos dados com base em ArPLS-ST.
    data: array contendo o sinal e a RFI.
    eta_i: lista de sensibilidades.
    Retorna: máscara que cobre a RFI identificada.
    """

    freq_mean = data.mean(axis=1)  # Calcula a média das frequências.
    bl = ArPLS(freq_mean, lam=100000)  # Usa o ArPLS para estimar a \
   textit{baseline}.
    diff = freq_mean - bl  # Calcula a diferença entre a curva SED e a \
   textit{baseline}.

    popt = ksigma(diff)  # Calcula o primeiro valor de limiar baseado em
    K sigma.
    line_mask = _run_sumthreshold_arpls(diff, 2 * popt)  # Aplica o
   sumthreshold para mitigar o RFI da banda.

    line_index = np.where(line_mask == True)[0]  # Encontra os índices
   de RFI na banda.
    final_curve = freq_mean.copy()  # Copia a curva de frequência.
    final_curve[line_index] = bl[line_index]  # Substitui os pontos de
   RFI pela \textit{baseline}.

    valid_index = np.where(line_mask == False)[0]  # Encontra os índices
    válidos.
    valid_data = data - final_curve[:, np.newaxis]  # Subtrai a curva
   final dos dados.
    valid_data = valid_data[valid_index]  # Mantém apenas os dados vá
   lidos.

    popt_point = ksigma(valid_data)  # Calcula o valor de limiar para os
    pontos de RFI.
    mask = blob_mitigation(data, final_curve, line_mask, 5 * popt_point)
     # Aplica a mitigação de blobs.

    mask[line_index] = True  # Marca os índices da banda na máscara.

    return mask  # Retorna a máscara final.
```

Função `st_mask(data, eta_i, chi_1, sm_kwargs, di_kwargs)`

- **Objetivo**: Aplicar o método *SumThreshold* para gerar uma máscara de RFI nos dados suavizados.
- **Parâmetros**:
  - `data`: Matriz contendo os dados e a RFI.
  - `eta_i`: Lista de sensibilidades para ajustar a detecção de RFI.
  - `chi_1`: Primeiro valor de limiar usado no processo de *thresholding*.
  - `sm_kwargs`: Argumentos relacionados à suavização dos dados (passados pela função `get_sm_kwargs`).
  - `di_kwargs`: Argumentos relacionados à dilatação da máscara (passados pela função `get_di_kwargs`).
- **Conexão**: Essa função utiliza o método de suavização gaussiana e, posteriormente, aplica a técnica de limiarização para detectar e mascarar RFI. É usada no arquivo `main1.py` como o segundo método de mitigação de RFI.

```
def st_mask(data, eta_i=[0.5, 0.55, 0.62, 0.75, 1], chi_1=35000,
    sm_kwargs=None, di_kwargs=None):
    """
    Cria uma máscara para cobrir a RFI nos dados.
    data: array contendo o sinal e a RFI.
    chi_1: Primeiro valor de limiar.
    eta_i: lista de sensibilidades.
    sm_kwargs: argumentos para suavização.
    di_kwargs: argumentos para dilatação.
    Retorna: máscara que cobre a RFI identificada.
    """

    mask = np.full(data.shape, False)  # Inicializa a máscara como "
    False".
    chi_i = chi_1 / p**np.log2(m)  # Calcula os valores de chi baseados
    em chi_1.

    st_mask = mask  # Inicializa a máscara de SumThreshold.
    for eta in eta_i:
        st_mask = _run_sumthreshold(data, st_mask, eta, M, chi_i,
    sm_kwargs)  # Executa o SumThreshold.

    dilated_mask = st_mask  # Copia a máscara resultante.

    if di_kwargs is not None:  # Se houver parâmetros de dilatação...
```

```
22          dilated_mask = binary_mask_dilation(dilated_mask.astype(np.float
    ) - mask.astype(np.float), **di_kwargs)  # Aplica dilatação.
23
24      return dilated_mask + mask  # Retorna a máscara final combinada.
```

Funções de Auxílio (get_sm_kwargs, get_di_kwargs)

- **Objetivo**: Retornar parâmetros ajustáveis para os métodos de suavização e dilatação usados na função `st_mask`.
- **Conexão**: Essas funções são usadas para modular os parâmetros de suavização e dilatação, facilitando a configuração do algoritmo *SumThreshold*. Elas são chamadas no arquivo `main1.py` durante a execução da função `st_mask`.

```
1  def st_mask(data, eta_i=[0.5, 0.55, 0.62, 0.75, 1], chi_1=35000,
       sm_kwargs=None, di_kwargs=None):
2      """
3      Cria uma máscara para cobrir a RFI nos dados.
4      data: array contendo o sinal e a RFI.
5      chi_1: Primeiro valor de limiar.
6      eta_i: lista de sensibilidades.
7      sm_kwargs: argumentos para suavização.
8      di_kwargs: argumentos para dilatação.
9      Retorna: máscara que cobre a RFI identificada.
10     """
11
12     mask = np.full(data.shape, False)  # Inicializa a máscara como "
       False".
13     chi_i = chi_1 / p**np.log2(m)  # Calcula os valores de chi baseados
       em chi_1.
14
15     st_mask = mask  # Inicializa a máscara de SumThreshold.
16     for eta in eta_i:
17         st_mask = _run_sumthreshold(data, st_mask, eta, M, chi_i,
       sm_kwargs)  # Executa o SumThreshold.
18
19     dilated_mask = st_mask  # Copia a máscara resultante.
20
21     if di_kwargs is not None:  # Se houver parâmetros de dilatação...
22         dilated_mask = binary_mask_dilation(dilated_mask.astype(np.float
       ) - mask.astype(np.float), **di_kwargs)  # Aplica dilatação.
23
24     return dilated_mask + mask  # Retorna a máscara final combinada.
25
26
```

```
def get_sm_kwargs(kernel_m=KERNEL_M, kernel_n=KERNEL_N, sigma_m=SIGMA_M,
    sigma_n=SIGMA_N):
    """
    Cria um dicionário com as palavras-chave para a suavização.
    kernel_m: tamanho da janela do kernel no eixo M.
    kernel_n: tamanho da janela do kernel no eixo N.
    sigma_m: sigma do kernel no eixo M.
    sigma_n: sigma do kernel no eixo N.
    Retorna: dicionário com as palavras-chave para suavização.
    """
    return dict(M=kernel_m, N=kernel_n, sigma_m=sigma_m, sigma_n=sigma_n
   )
```

Função `_run_sumthreshold(data, init_mask, eta, M, chi_i, sm_kwargs)`

- **Objetivo**: Executar o método *SumThreshold* em uma série de iterações, aplicando suavização e limiarização para detectar regiões com RFI.
- **Parâmetros**:
  - `data`: Dados de entrada.
  - `init_mask`: Máscara inicial para ignorar certas regiões durante a detecção.
  - `eta`: Fator de ajuste para o valor de limiar.
  - `M`: Número de iterações.
  - `chi_i`: Critérios de limiarização para cada iteração.
  - `sm_kwargs`: Parâmetros de suavização.
- **Conexão**: Essa função é utilizada dentro de `st_mask` para realizar a detecção iterativa de RFI em diferentes janelas de suavização e limiarização.

```
def _run_sumthreshold_arpls(diff_temp, chi_1=3):
    '''
    Função para aplicar o sumthreshold para uma lista de valores de
   limiar.
    diff_temp: diferença entre os dados e a \textit{baseline}.
    chi_1: o primeiro valor de limiar.
    '''
    res = diff_temp.copy()  # Cria uma cópia da diferença dos dados.
    chi_i = chi_1 / p**np.log2(m)  # Calcula a lista de valores de
   limiar com base no primeiro chi.

    if len(res.shape) == 1:
        res = res[:, np.newaxis]  # Ajusta a forma do array se for
   unidimensional.
```

```
12
13      st_mask = np.full(res.shape, False)  # Inicializa a máscara de RFI
    como "False".
14
15      for mi, chi in zip(M, chi_i):  # Itera pelos valores de M e chi.
16          if mi == 1:
17              st_mask = st_mask | (chi <= res)  # Atualiza a máscara
    baseando-se no limiar.
18          else:
19              if diff_temp.shape[-1] != 1:
20                  st_mask = _sumthreshold(res, st_mask, mi, chi, *res.
    shape)  # Aplica sumthreshold nas linhas.
21              st_mask = _sumthreshold(res.T, st_mask.T, mi, chi, *res.T.
    shape).T  # Aplica sumthreshold nas colunas.
22
23      return st_mask  # Retorna a máscara final.
```

Funções no Arquivo `main1.py`

Função `plot_subint(image, h, title, cmap)`

- **Objetivo**: Exibir os dados de entrada em formato gráfico, com eixos rotulados em tempo e frequência.
- **Parâmetros**:
  - `image`: Dados de entrada a serem exibidos.
  - `h`: Cabeçalho dos dados FITS, contendo informações como o tempo entre amostras (`TBIN`) e largura de banda dos canais (`CHAN_BW`).
  - `title`: Título da imagem.
  - `cmap`: Colormap utilizado para exibir a imagem (esquema de cores).
- **Conexão**: Essa função é chamada repetidamente no script principal para exibir tanto os dados originais quanto os resultados processados após a aplicação dos métodos de mitigação de RFI.

```
1 def plot_subint(image, h, title, cmap):
2     m,n = image.data.shape  # Pega o tamanho da imagem (m linhas, n
    colunas)
3     plt.figure()  # Cria um novo gráfico
4     plt.imshow(image, aspect='auto', cmap=cmap)  # Mostra a imagem
5     # Define as marcações no gráfico para o tempo e a frequência
6     plt.xticks(np.arange(0, n, n//5), np.round(np.arange(0, h['TBIN']*(n
    +1), h['TBIN']*n/5), decimals=2))
```

```
    plt.yticks(np.arange(0, m, m//5), np.arange(0, 1600, 100)[::-1])
    plt.title(title)  # Dá um título para o gráfico
    plt.xlabel('Time (s)')  # Rótulo para o eixo do tempo
    plt.ylabel('Frequency (MHz)')  # Rótulo para o eixo da frequência
```

Execução Principal

O arquivo `main1.py` é responsável pela execução do *pipeline* completo. Ele lê os dados de um arquivo FITS, aplica as máscaras de mitigação de RFI (`arpls_mask` e `st_mask`), e exibe os resultados.

Fluxo:

1. Inicialmente, os dados são lidos e exibidos em sua forma original.
2. Em seguida, o método *ArPLS-ST* é aplicado, e o resultado é exibido.
3. Finalmente, o método *SumThreshold* é aplicado, e o resultado também é exibido.

```
if __name__ == '__main__':
    # Lê o arquivo de imagem do telescópio (é como carregar uma foto)
    fname = '../code/G69.04+0.00_tracking-M14_0040.fits'
    sub = 0  # Escolhe uma parte específica do arquivo para ler
    hdu, h= fitsio.read(fname, header=True, rows=sub, ext=-1)  # Lê a
    imagem e as informações extras
    data = hdu[0]['DATA'][:, 0, :, 0]  # Pega os dados da imagem
    data = data.T[::-1]  # Ajusta a orientação da imagem

    # Mostra a imagem original
    plot_subint(data, h, 'Original image', 'hot')

    # Primeiro método para remover o ruído: ArPLS-ST
    print('RFI flagging based on ArPLS-ST...')
    mask1 = arpls_mask(data)  # Aplica a máscara de remoção de ruído
    palette = copy(plt.cm.hot)  # Copia o esquema de cores
    palette.set_bad('cyan', 1.0)  # Define a cor ciano para as partes
    ruins (ruído)
    plot_subint(np.ma.array(data, mask=mask1), h, 'ArPLS-ST', palette)
    # Mostra o resultado com a máscara

    # Segundo método para remover o ruído: SumThreshold
    print('RFI flagging based on SumThreshold...')
    mask2 = st_mask(data, chi_1=20, sm_kwargs=get_sm_kwargs(80, 50, 10,
    30), di_kwargs=get_di_kwargs(3, 3))  # Aplica outra máscara
    plot_subint(np.ma.array(data, mask=mask2), h, 'SumThreshold',
    palette)  # Mostra o resultado com a segunda máscara
```

```
# Nome dos arquivos FITS
arquivo_fits_original = '/content/drive/MyDrive/TCC/data/UIRAPURU_1270_20230306_093620_59.fit'
arquivo_fits_contaminado6 = 'arquivo_contaminado1.fits'  # Nome do arquivo contaminado

# Abrir o arquivo FITS original
with fits.open(arquivo_fits_original) as hdul:
    dados = hdul[0].data
    header = hdul[0].header

    # Fazer uma cópia dos dados originais para exibição posterior
    dados_originais = np.copy(dados)

    # Parâmetros do ruído RFI
    frequencia_rfi = 50  # Frequência do ruído RFI em Hz
    amplitude_rfi = np.max(dados) * 0.12  # Amplitude é 12% do valor máximo dos dados
    num_pontos = dados.shape[-1]  # Número de colunas (tempo)
    tempo = np.arange(num_pontos)

    # Escolha das colunas específicas para serem contaminadas
    colunas_especificas = [300, 512, 550, 768, 800, 860, 900, 950]
    print(f"Colunas escolhidas para contaminação: {colunas_especificas}")

    # Verificar se as colunas escolhidas estão dentro dos limites
    if any(coluna >= num_pontos for coluna in colunas_especificas):
        raise ValueError(f"Alguma coluna escolhida excede o número de colunas ({num_pontos}).")

    # Gerar os diferentes tipos de ruído
    rfi_sin = amplitude_rfi * np.sin(2 * np.pi * frequencia_rfi * tempo / num_pontos)
    rfi_gaussiano = amplitude_rfi * np.random.normal(loc=0, scale=1, size=num_pontos)
    rfi_pulso = amplitude_rfi * np.sign(np.sin(2 * np.pi * frequencia_rfi * tempo / num_pontos))
    rfi_impulsivo = amplitude_rfi * (np.random.rand(num_pontos) > 0.95)
    rfi_banda_larga = amplitude_rfi * (
        np.sin(2 * np.pi * frequencia_rfi * tempo / num_pontos) +
        np.sin(2 * np.pi * (frequencia_rfi + 20) * tempo / num_pontos) +
        np.sin(2 * np.pi * (frequencia_rfi + 40) * tempo / num_pontos)
    )
    frequencias = np.fft.fftfreq(num_pontos)
    ruido_espectral = (np.random.randn(num_pontos) + 1j * np.random.randn(num_pontos)) / (frequencias + 0.01)
```

```
    rfi_rosa = np.fft.ifft(ruido_espectral).real * amplitude_rfi

    # Escolher o tipo de ruído que deseja aplicar
    tipo_ruido = 'impulsivo'  # Alterar para: 'gaussiano', 'pulso', 'impulsivo', 'banda_larga', 'rosa'
    print(f"Tipo de ruído escolhido: {tipo_ruido}")

    if tipo_ruido == 'sinodal':
        rfi = rfi_sin
    elif tipo_ruido == 'gaussiano':
        rfi = rfi_gaussiano
    elif tipo_ruido == 'pulso':
        rfi = rfi_pulso
    elif tipo_ruido == 'impulsivo':
        rfi = rfi_impulsivo
    elif tipo_ruido == 'banda_larga':
        rfi = rfi_banda_larga
    elif tipo_ruido == 'rosa':
        rfi = rfi_rosa
    else:
        raise ValueError("Tipo de ruído desconhecido. Escolha entre 'sinodal', 'gaussiano', 'pulso', 'impulsivo', 'banda_larga', 'rosa'.")

    # Aplicar a contaminação nas colunas selecionadas
    for coluna in colunas_especificas:
        if coluna < num_pontos:
            dados[:, coluna] += rfi[coluna % len(rfi)]

    # Salvar os dados contaminados em um novo arquivo FITS
    hdul[0].data = dados
    hdul.writeto(arquivo_fits_contaminado6, overwrite=True)
    print(f"Arquivo contaminado salvo em {arquivo_fits_contaminado6}")

# Definir os eixos de tempo e frequência (simulação)
time = np.linspace(0, num_pontos, num_pontos)
frequency = np.linspace(0, 100, dados.shape[0])

# Exibir a imagem original (sem contaminação)
plt.figure(figsize=(10, 5))
plt.imshow(dados_originais.T, aspect='auto', cmap='plasma', extent=[min(time), max(time), min(frequency), max(frequency)])
plt.colorbar(label='Intensidade')
plt.xlabel('Tempo')
plt.ylabel('Frequência')
plt.title('Imagem Original - Tempo vs Frequência')
plt.show()

```

```
# Exibir a imagem contaminada (com RFI em colunas específicas)
plt.figure(figsize=(10, 5))
plt.imshow(dados.T, aspect='auto', cmap='plasma', extent=[min(time), max(time), min(frequency), max(frequency)])
plt.colorbar(label='Intensidade')
plt.xlabel('Tempo')
plt.ylabel('Frequência')
plt.title('Imagem Contaminada com RFI - Colunas Específicas')
plt.show()
```

```
from mpl_toolkits.mplot3d import Axes3D

# Caminho para o arquivo FITS
file_path = '/content/drive/MyDrive/TCC/data1/concatenated_output.fits'  # Substitua pelo caminho correto do arquivo FITS

# Abrir o arquivo FITS
hdul = fits.open(file_path)

# Verificar a estrutura dos dados (header)
hdul.info()

# Supondo que os dados estão na primeira extensão
data = hdul[0].data
header = hdul[0].header

# Fechar o arquivo FITS
hdul.close()

# Definir os eixos: tempo (X), frequência (Y) e intensidade (Z)
num_frequencias, num_tempos = data.shape

# Se houver informações sobre o tempo no cabeçalho, usar
tbin = header.get('TBIN', 1e-3)  # Usar o valor de 'TBIN' (tempo entre amostras) do cabeçalho, se disponível
tempo = np.arange(0, num_tempos) * tbin  # Gera todo o eixo de tempo

# Frequência, ajustar conforme necessário ou com base em metadados do cabeçalho
frequencia = np.linspace(1000, 1500, num_frequencias)  # Simulando frequências entre 1000 e 1500 MHz

# Criar uma grade de valores para os eixos X e Y
X, Y = np.meshgrid(tempo, frequencia)
Z = data  # Intensidade (valores do arquivo FITS)

# Criar a figura 3D
```

```
fig = plt.figure(figsize=(10, 7))
ax = fig.add_subplot(111, projection='3d')

# Plotar a superfície 3D
surf = ax.plot_surface(X, Y, Z, cmap='viridis', edgecolor='none')

# Adicionar rótulos aos eixos
ax.set_xlabel('Time (s)')
ax.set_ylabel('Frequency (MHz)')
ax.set_zlabel('Intensity')

# Adicionar uma barra de cores para a intensidade
fig.colorbar(surf, ax=ax, shrink=0.5, aspect=5)

# Exibir o gráfico
plt.show()
```

```
# Função para plotar as imagens
def plot_subint(image, h, title, cmap):
    m, n = image.shape
    plt.figure()
    plt.imshow(image, aspect='auto', cmap=cmap)

    # Verificar se 'TBIN' está no header e definir um valor padrão se não estiver
    tbin = h.get('TBIN', 1.0)  # Usar 1.0 como valor padrão se 'TBIN' não estiver presente

    # Corrigir a geração de ticks e labels para que o número de elementos coincida
    plt.xticks(np.arange(0, n, n // 5), np.round(np.arange(0, tbin * n, tbin * n / 5), decimals=2))

    # Verificar se 'CHAN_BW' está no header e definir um valor padrão se não estiver
    chan_bw = h.get('CHAN_BW', 1.0)  # Usar 1.0 como valor padrão se 'CHAN_BW' não estiver presente

    # Gerar os yticks e rótulos de forma que tenham o mesmo tamanho
    yticks = np.linspace(0, m, 5)  # Gerar 5 valores de tick no eixo Y
    ytick_labels = np.round(np.linspace(0, m * chan_bw, 5)[::-1], decimals=2)  # Gerar 5 rótulos correspondentes

    plt.yticks(yticks, ytick_labels)

    plt.title(title)
    plt.xlabel('Tempo (s)')
```

```
    plt.ylabel('Frequência (MHz)')
    plt.colorbar(label='Intensidade')
    plt.show()

if __name__ == '__main__':
    # Caminho do arquivo FITS
    fname = '/content/drive/MyDrive/TCC/data1/concatenated_output.fits'

    # Abrir o arquivo FITS usando astropy para obter o header
    with fits.open(fname) as hdul:
        header = hdul[0].header  # Extrair o cabeçalho usando astropy
        data = hdul[0].data  # Extrair os dados diretamente com astropy

    # Verificar as dimensões dos dados
    print(f"Dimensões do array: {data.shape}")

    # Se os dados forem 2D, usá-los diretamente
    if len(data.shape) == 2:
        data = data  # O dado já é 2D, basta copiá-lo
    else:
        print("Os dados não são 2D como esperado. Ajuste necessário para acessar o array corretamente.")

    # Inverter os dados no eixo vertical
    data = data.T[::-1]  # Transposição e inversão

    # Plotar a imagem original
    plot_subint(data, header, 'Imagem Original', 'plasma')

    # Aplicar o ArPLS
    print('Aplicando ArPLS para remoção da linha de base...')
    corrected_data = np.zeros_like(data)
    for i in range(data.shape[0]):
        corrected_data[i, :] = data[i, :] - ArPLS(data[i, :])

    # Plotar a imagem após a aplicação do ArPLS
    plot_subint(corrected_data, header, 'Imagem após ArPLS', 'plasma')

    # Salvar os dados corrigidos em um novo arquivo FITS
    output_file = '/content/drive/MyDrive/TCC/data/UIRAPURU_1270_20230306_093620_59_arpls2_corrected.fits'
    hdu = fits.PrimaryHDU(data=corrected_data.T, header=header)  # Usar o header correto extraído com astropy
    hdu.writeto(output_file, overwrite=True)

    print(f"Arquivo corrigido salvo em: {output_file}")
```

```
1
2 """
3 You should have received a copy of the GNU Lesser General Public License
4 along with this program.  If not, see <http://www.gnu.org/licenses/>
5 """
6
7
8 import numpy as np
9 from scipy.sparse.linalg import spsolve
10 from scipy.linalg import cholesky
11 # from scikits.sparse.cholmod import cholesky
12 from scipy import sparse
13
14 from scipy.stats import norm
15 import matplotlib.pyplot as pl
16
17 def arpls(y, lam=1e4, ratio=0.05, itermax=100):
18     """
19     Baseline correction using asymmetrically
20     reweighted penalized least squares smoothing
21     Sung-June Baek, Aaron Park, Young-Jin Ahna and Jaebum Choo,
22     Analyst, 2015, 140, 250 (2015)
23
24     Abstract
25
26     Baseline correction methods based on penalized least squares are
    successfully
27     applied to various spectral analyses. The methods change the weights
     iteratively
28     by estimating a baseline. If a signal is below a previously fitted
    baseline,
29     large weight is given. On the other hand, no weight or small weight
    is given
30     when a signal is above a fitted baseline as it could be assumed to
    be a part
31     of the peak. As noise is distributed above the baseline as well as
    below the
32     baseline, however, it is desirable to give the same or similar
    weights in
33     either case. For the purpose, we propose a new weighting scheme
    based on the
34     generalized logistic function. The proposed method estimates the
    noise level
35     iteratively and adjusts the weights correspondingly. According to
    the
36     experimental results with simulated spectra and measured Raman
    spectra, the
```

```
    proposed method outperforms the existing methods for baseline correction and
    peak height estimation.

    Inputs:
        y:
            input data (i.e. chromatogram of spectrum)
        lam:
            parameter that can be adjusted by user. The larger lambda is,
            the smoother the resulting background, z
        ratio:
            wheighting deviations: 0 < ratio < 1, smaller values allow less negative values
        itermax:
            number of iterations to perform
    Output:
        the fitted background vector

    """
    N = len(y)
#   D = sparse.csc_matrix(np.diff(np.eye(N), 2))
    D = sparse.eye(N, format='csc')
    D = D[1:] - D[:-1]  # numpy.diff( ,2) does not work with sparse matrix. This is a workaround.
    D = D[1:] - D[:-1]

    H = lam * D.T * D
    w = np.ones(N)
    for i in range(itermax):
        W = sparse.diags(w, 0, shape=(N, N))
        WH = sparse.csc_matrix(W + H)
        C = sparse.csc_matrix(cholesky(WH.todense()))
        z = spsolve(C, spsolve(C.T, w * y))
        d = y - z
        dn = d[d < 0]
        m = np.mean(dn)
        s = np.std(dn)
        wt = 1. / (1 + np.exp(2 * (d - (2 * s - m)) / s))
        if np.linalg.norm(w - wt) / np.linalg.norm(w) < ratio:
            break
        w = wt
    return z

# Função arPLS - insira aqui a função arPLS se não estiver importando de outro módulo

```

```
79 if __name__ == '__main__':
80     print('Iniciando teste com arPLS...\n')
81 
82     # Dados
83     x = np.arange(0, 1000, 1)
84 
85     # Três Gaussianas como sinal
86     g1 = norm(loc=100, scale=10.0).pdf(x)
87     g2 = norm(loc=300, scale=30.0).pdf(x)
88     g3 = norm(loc=750, scale=5.0).pdf(x)
89     signal = g1 + g2 + g3
90 
91     # Baselines
92     baseline1 = 5e-4 * x + 0.2  # baseline linear
93     baseline2 = 0.2 * np.sin(np.pi * x / x.max())  # baseline senoidal
94 
95     # Ruído
96     noise = np.random.poisson(100 * baseline2, size=x.shape[0]) / 800
97 
98     # Sinais com baselines e ruído
99     y1 = signal + baseline1 + noise
100    y2 = signal + baseline2 + noise
101 
102    # Remoção de baseline usando apenas arPLS
103    corrected_y1 = y1 - arpls(y1, 1e7, 0.1)  # arPLS no sinal com
    baseline linear
104    corrected_y2 = y2 - arpls(y2, 1e7, 0.01)  # arPLS no sinal com
    baseline senoidal
105 
106    # Gráfico das Gaussianas Individuais
107    pl.figure(figsize=(10, 5))
108    pl.plot(x, g1, '-r', label='Gaussiana 1')
109    pl.plot(x, g2, '-g', label='Gaussiana 2')
110    pl.plot(x, g3, '-b', label='Gaussiana 3')
111    pl.xlabel('Posição do ponto')
112    pl.ylabel('Intensidade')
113    pl.legend()
114    pl.grid(True)
115    pl.show()
116 
117    # Gráfico do Sinal Total
118    pl.figure(figsize=(10, 5))
119    pl.plot(x, signal, '-k', label='Sinal Total', linewidth=2)
120    pl.xlabel('Posição do ponto')
121    pl.ylabel('Intensidade do sinal total')
122    pl.legend()
123    pl.grid(True)
```

```
    pl.show()

    # # Gráfico do Sinal com Baseline Linear (y1) e Correção arPLS
    # pl.figure(figsize=(10, 5))
    # pl.plot(x, y1, '-k', label='Sinal com Baseline Linear')
    # pl.plot(x, corrected_y1, '-b', label='Corrigido (arPLS)')
    # pl.xlabel('Posição do ponto')
    # pl.ylabel('Intensidade do sinal')
    # pl.legend()
    # pl.grid(True)
    # pl.show()

    # # Gráfico do Sinal com Baseline Senoidal (y2) e Correção arPLS
    # pl.figure(figsize=(10, 5))
    # pl.plot(x, y2, '-k', label='Sinal com Baseline Senoidal')
    # pl.plot(x, corrected_y2, '-b', label='Corrigido (arPLS)')
    # pl.xlabel('Posição do ponto')
    # pl.ylabel('Intensidade do sinal')
    # pl.legend()
    # pl.grid(True)
    # pl.show()

    # # Gráfico dos Sinais y1 e y2 Corrigidos com arPLS
    # pl.figure(figsize=(10, 5))
    # pl.plot(x, corrected_y1, '-c', label='Sinal y1 Corrigido')
    # pl.plot(x, corrected_y2, '-y', label='Sinal y2 Corrigido')
    # pl.xlabel('Posição do ponto')
    # pl.ylabel('Intensidade do sinal corrigido')
    # pl.legend()
    # pl.grid(True)
    # pl.show()
    # Criação de uma única figura para os três gráficos sobrepostos
    pl.figure(figsize=(14, 7))

    # Gráfico do Sinal com Baseline Linear (y1) e Correção arPLS
    pl.plot(x, y1, '-k', label='Sinal com Baseline Linear')
    pl.plot(x, corrected_y1, '-b', label='Corrigido (arPLS - Linear)')

    # Gráfico do Sinal com Baseline Senoidal (y2) e Correção arPLS
    pl.plot(x, y2, '-r', label='Sinal com Baseline Senoidal')
    pl.plot(x, corrected_y2, '-g', label='Corrigido (arPLS - Senoidal)')

    # Gráfico dos Sinais y1 e y2 Corrigidos com arPLS
    # pl.plot(x, corrected_y1, '-c', label='Sinal y1 Corrigido (Extra)')
    # pl.plot(x, corrected_y2, '-y', label='Sinal y2 Corrigido (Extra)')

    # Configurações gerais
```

```
171     pl.xlabel('Posição do ponto')
172     pl.ylabel('Intensidade do sinal')
173     pl.legend()
174     pl.grid(True)
175     #pl.title('Gráficos dos Sinais com Baseline e Correção arPLS')
176 
177     # Exibição do gráfico
178     pl.show()
179 
180     print('\nFinalizado!')
```