

FIA 2020/22

XII CONGRESSO/CONGRESO IBEROAMERICANO DE ACÚSTICA
XXIX ENCONTRO DA SOCIEDADE BRASILEIRA DE ACÚSTICA - SOBRAC

Florianópolis, SC, Brasil

Toolbox educacional para simulação de fontes sonoras estáticas e em movimento

Gomes, L. M.¹; Fonseca, W. D'A.¹; Mello, F. R.¹; Mareze, P. H.¹

¹ Engenharia Acústica, Universidade Federal de Santa Maria, Santa Maria, RS, Brasil,
{lucas.gomes, will.fonseca, felipe.mello, paulo.mareze}@eac.ufsm.br

Resumo

No ensino e pesquisa em acústica, frequentemente, tem-se como objeto de estudo medições de fontes sonoras e processamento dos sinais dos áudios obtidos. Porém, em várias circunstâncias essas medições não são possíveis de serem realizadas, ocasionando assim a falta dos dados para o estudo, o que pode, eventualmente, resultar em um desinteresse do estudante com o tema. Desse modo, este trabalho apresenta um toolbox (conjunto de códigos e funções), desenvolvidos em Matlab (linguagem de programação interpretada), para geração de dados simulados similares às situações de medições de fontes sonoras, sejam elas estáticas ou em movimento. A ideia principal do trabalho é a de permitir a construção desses dados sintéticos com diversas alternativas, tais como, ruído de banda larga, ruído de banda estreita (filtrado), tonais ou ainda, a partir de uma faixa de áudio real e estática, colocá-la virtualmente em movimento. Logo, o Efeito Doppler será incluído, dependendo da velocidade entre fonte e receptor. Na utilização do toolbox para a geração de fontes sonoras em movimento, os dados criados podem ser usados, por exemplo, no estudo de técnicas de processamento para sinais transientes, como espectrograma, janelamento temporal, estimativa de frequência instantânea, técnicas de dedoplerização, análise com ondaletas, método de Welch, entre outros. O toolbox viabiliza a escolha de diferentes tipos de receptores, podendo se estabelecer manualmente posições e quantidade de relações fonte-receptor. Além disso, pode-se gerar também arranjos de receptores, com geometrias pré-estabelecidas (como espiral, circular, linear, entre outras), possibilitando assim, sua aplicação no estudo de técnicas de imageamento acústico, por exemplo. No artigo, encontra-se um pequeno tutorial passo a passo, ensinando como iniciar e desenvolver seus próprios sinais. Esta iniciativa nasceu na Engenharia Acústica (EAC) da Universidade Federal de Santa Maria (UFSM), sendo de livre uso para comunidade, discentes e docentes.

Palavras-chave: simulação experimental, efeito doppler, arranjo de microfones, processamento digital de sinais.

PACS: 43.10.Sv, 43.60.-c, 43.35.Sx, 43.28.Py.

Educational toolbox for simulating static and moving sound sources

Abstract

In acoustics teaching and research, the study object is frequently measurements of sound sources and the obtained audio signals processing. However, in several situations, such measurements cannot be held leading to a lack of data to study, which in turn can result in students becoming disinterested in the theme. Thus, this work presents a toolbox developed in Matlab for the generation of simulated data similar to sound source measurements, being the source static or in movement. The work's main idea is to allow for the construction of the synthetic data by considering various alternatives, such as broadband noise, narrowband noise, tones, or, from a static real audio signal, virtually putting it in motion. Therefore, the Doppler Effect is included according to the velocity between the source and receiver. When using the toolbox for generating moving sound sources, the created data can be used, for example, in studying processing techniques for transient signals, such as the spectrogram, time windowing, instantaneous frequency estimation, dedopplerization techniques, wavelet analysis, and Welch's method, and others. The toolbox allows for choosing different receiver types, being possible to define positions and the number of source-receiver relations. Furthermore, it is possible to create receiver arrays using pre-defined geometries (spiral, circular, linear, and others), thus allowing their application in the study of acoustic imaging techniques, for example. This article is a brief tutorial showing how to create and design your own signals. This initiative was born in the Acoustical Engineering Program (EAC) at the Federal University of Santa Maria, its use being free for the community, students, and professors.

Keywords: measurement simulation, doppler effect, microphone array, digital signal processing.



1. INTRODUÇÃO

Este trabalho busca apresentar algoritmos de simulações de medições sonoras, com a proposta de serem utilizadas no ensino e pesquisa de acústica. São apresentadas as metodologias e configurações para a realização de simulações para fontes sonoras estáticas e em movimento, em relação a receptores estáticos.

2. FUNDAMENTOS

Esta Seção trata dos fundamentos teóricos que formam a base das ferramentas apresentadas neste tutorial. Especificamente, a Seção 2.1 aborda o conceito de fontes sonoras estáticas e móveis; a Seção 2.2 discute o efeito Doppler e sua formulação matemática; finalmente, a Seção 2.3 apresenta conceitos-chave em processamento de sinais.

2.1 Fontes sonoras

Frequentemente para fins de simulação ou cálculos, as fontes sonoras podem ser apresentadas como um monopolo emitindo ondas acústicas. Nesta seção serão apresentadas as formulações para os monopolos estático (Seção 2.1.1) e móvel (Seção 2.1.2).

2.1.1 Monopolo estático

A pressão gerada por um monopolo e experienciada por um receptor estático é dada por:

$$p(t) = \frac{\rho}{4\pi r} q'(t - r/c_0), \quad (1)$$

em que $p(t)$ é a pressão sonora gerada, r a distância entre a fonte e o receptor, ρ a densidade do ar, c_0 a velocidade de propagação da onda no ar e q' a derivada temporal da velocidade de volume. O argumento r/c_0 , em q' , representa a diferença de tempo entre a geração e recepção da onda sonora [1].

A característica da fonte sonora é representada, na Equação 1, pelo termo $\rho q'(t - r/c_0)$, que descreve a *força da fonte*¹. Por simplificação, pode-se substituí-lo por $s(t - r/c_0)$ ², representando assim o sinal sonoro emitido pela fonte em um determinado instante de tempo.

Desse modo, obtém-se

$$p(t) = \frac{s(t - r/c_0)}{4\pi r} \quad (2)$$

e

$$p(\omega) = \frac{s(\omega) e^{j\omega \frac{r}{c_0}}}{4\pi r}, \quad (3)$$

que descrevem a pressão sonora gerada por uma fonte sonora e recebida por um receptor, nos domínios do tempo e da frequência.

2.1.2 Monopolo móvel

Para representar a pressão sonora gerada por uma fonte em movimento e recebida por um receptor estático, é usada a mesma simplificação feita anteriormente, ou seja, assume-se que a fonte sonora é um monopolo.

A pressão sonora gerada por uma fonte em movimento é dada por:

$$p(t) = \frac{s\left(t - \frac{R(t)}{c}\right)}{4\pi R(t)(1 - M \cos[\theta(t)])^2}, \quad (4)$$

em que t , em segundos, é o instante de tempo que receptor recebe a onda sonora, s ² representa a pressão sonora da fonte em Pascal, c é a velocidade de propagação das ondas sonoras no ar em m/s, $R(t)$ é distância entre a fonte e o receptor em metros, M é o número de Mach (v/c , adimensional), dado pela razão entre a velocidade da fonte (v) e a velocidade de propagação das ondas sonoras no ar, $\theta(t)$ é o ângulo entre a direção da trajetória da fonte e a direção fonte-receptor, em radianos, e $p(t)$ ² a pressão sonora gerada no receptor em Pascal [2]³.

O termo $(t - R(t)/c)$, em s , significa o instante de tempo que a onda sonora foi emitida pela fonte, sendo $(R(t)/c)$ a diferença temporal entre emissão e recepção. Por simplificação, escreve-se $t_e = (t - R(t)/c)$ (tempo de emissão) e calcula-se as demais variáveis relacionados à fonte, como R e θ , em função de t_e ,

$$p(t) = \frac{s(t_e)}{4\pi R(t_e)(1 - M \cos[\theta(t_e)])^2}. \quad (5)$$

¹Do inglês *source strength*, em tradução literal.

² s é uma abreviação para *source*, que significa fonte em português.

²O valor relativo à pascal.

³Formulação feita por Morse [3], e posteriormente revista e corrigida por Meng, Jax & Vorländer [2].

2.2 Efeito Doppler

O Efeito Doppler é um evento acústico que muitas pessoas entendem ou vivenciam intuitivamente. Ele é percebido como uma mudança da frequência sonora percebida por um ouvinte, quando há movimento relativo entre uma fonte sonora e um receptor. Esse fenômeno pode ser experienciado, por exemplo, quando ambulâncias ou carros de polícia estão passando com suas sirenes acionadas. Em razão da movimentação da fonte, e consequentemente aproximação com o receptor, as ondas sonoras passam a chegar com uma taxa de tempo variada, gerando, assim, uma alteração na frequência e amplitude que será percebida pelo ouvinte.

2.3 Processamento sinais

Processamento de sinais é um ramo da ciência que estuda técnicas para analisar, modificar ou sintetizar sinais. Essa área do conhecimento lida com a interação entre sinais e sistemas. Os primeiros podem ser definidos como funções (contendo uma ou mais variáveis independentes) que carregam informações relacionadas a algum fenômeno e circulam entre sistemas, ou seja, entidades capazes de modificar um sinal, gerando outros [4]. Em processamento de sinais, muitas vezes é de interesse conhecer o comportamento de um sinal (ou sistema) no domínio da frequência. A principal ferramenta matemática para esse fim é a Transformada de Fourier, definida no domínio contínuo por:

$$X(j\omega) = \int_{-\infty}^{\infty} x(n) e^{-j\omega n} dt. \quad (6)$$

Atualmente, grande parte dos processamentos ocorrem no domínio digital, no qual o sinal temporal contínuo é amostrado em intervalos regulares de tempo e, deste modo, torna-se discreto. Neste caso, utiliza-se a Transformada Discreta de Fourier (TDF), definida por:

$$X(j\omega) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n}. \quad (7)$$

Quando se considera sinais determinísticos⁴ (cuja função matemática é conhecida), é comum aplicar diretamente a transformada de Fourier e, assim, obter seu conteúdo na frequência. Todavia, a

⁴Uma varredura exponencial, por exemplo.

maioria dos sinais observados no mundo real possuem uma natureza aleatória⁵ (não-determinística, cujo futuro não pode ser definido com exatidão), e sua análise exige uma abordagem estatística. Desse modo, busca-se estimar as características espectrais do sinal, ou seja, como sua energia se distribui ao longo das frequências.

Para sinais não-determinísticos, uma característica de particular interesse é sua estacionariedade. Sinais estacionários são aqueles cuja média e variância são constantes, ou variam muito pouco, ao longo do tempo. Essa característica é importante pois, quando satisfeita, permite dividir um único sinal gravado em vários segmentos e utilizá-los para estimar as propriedades espectrais do fenômeno observado (ao invés de gravar várias amostras). Nesse caso, o método de Welch é um dos mais utilizados. Quando um sinal é não-estacionário, deve-se utilizar outras técnicas para sua análise, sendo o espectrograma uma das mais conhecidas. Para mais informações sobre processamento de sinais, ver [4, 5].

3. TOOLBOX

Com o intuito de facilitar o acesso a sinais acústicos à estudantes de Engenharia Acústica (para uso didático em disciplinas como processamento de sinais, por exemplo), desenvolveu-se um conjunto de códigos capazes de simular medições acústicas em diferentes cenários, considerando diversos tipos de fontes sonoras e número de receptores. As rotinas foram escritas em Matlab⁶, por ser uma linguagem comumente utilizada no curso de Engenharia Acústica e indústria no geral, bem como por possuir um grande acervo de bibliotecas nativas com ferramentas prontamente aplicáveis à acústica. O programa pode ser obtido no site da [Mathworks](#) nas versões pagas (com valores diferenciados para estudantes) e de teste. Ademais, universidades como a UFSM costumam fornecer o *software* em seus laboratórios de informática.

Para construir as simulações, o *toolbox* possui objetos⁷ do tipo *fonte sonora*, responsáveis por gerar

⁵Uma bomba de água em operação, por exemplo.

⁶Linguagem muito utilizada em análise numérica e visualização de dados.

⁷O *toolbox* foi construído utilizando o paradigma de Programação Orientada ao Objeto, em que projeta-se *classes*, as quais podem ser atribuídas dados que são processados por *métodos*. Para mais informações sobre o tema, veja



os sinais que serão analisados, e do tipo *receptores*, que capturam os sinais da fonte em um ponto definido no espaço virtual. Mais detalhes sobre as ferramentas são explicados de forma prática na Seção 4, apresentando exemplos de código e resultados obtidos. Finalmente, o *toolbox* está disponível ao público via [repositório no GitHub](#). Para utilizá-lo, deve-se adicionar a pasta com os códigos no ambiente (*path*) do Matlab, como apresentado em sua documentação⁸. Após isso, todas as funções estarão disponíveis para uso em novas rotinas.

4. CONFIGURAÇÃO DAS SIMULAÇÕES

Esta Seção apresenta as ferramentas de simulação contidas no *toolbox*, explicando seu funcionamento e demonstrando como utilizá-las. Em particular, as Seções 4.1 e 4.2 apresentam os elementos de *fontes sonoras* e *receptores*, respectivamente; em seguida, a Seção 4.3 demonstram exemplos de simulações com fontes estática e em movimento, respectivamente.

4.1 Fontes sonoras

No *toolbox*, as fontes sonoras são simuladas por meio da classe *Source* (tradução literal de “fonte”, em inglês). Esta recebe parâmetros como amplitude e frequência, bem como posição no espaço virtual, e possui métodos para geração de sinal sonoro a partir das configurações do usuário. Para ilustrar seu funcionamento, os Códigos 1, 2 e 3 demonstram como construir uma fonte de ruído tonal, gaussiano e gaussiano filtrado (via filtro passa-bandas), respectivamente. Ademais, as Figuras 1, 2 e 3 contêm gráficos com os respectivos sinais, tanto no domínio do tempo quanto da frequência. Por fim, ainda é possível gerar uma fonte a partir de arquivos de áudio previamente gravados, como mostra o Código 4.

Utilizando-se os tipos de fontes sonoras citados, é possível criar cenários de medições simuladas. Assim, com as fontes de ruído gaussiano, por exemplo, é possível representar a passagem de automóveis ou aviões. Já com as fontes sonoras tonais, pode-se combiná-las a fim de criar um sinal sonoro complexo específico.

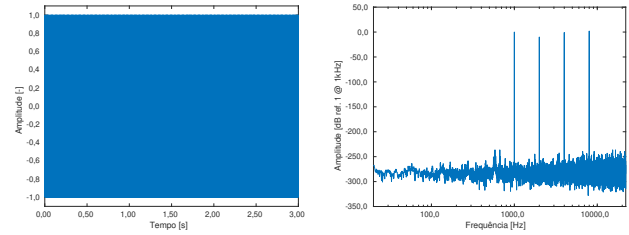


Figura 1: Gráfico no domínio do tempo e na frequência de uma fonte sonora composta de tonais em diversas frequências.

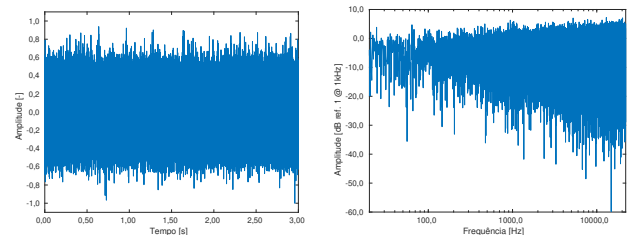


Figura 2: Gráfico no domínio do tempo e na frequência de uma fonte sonora de ruído gaussiano.

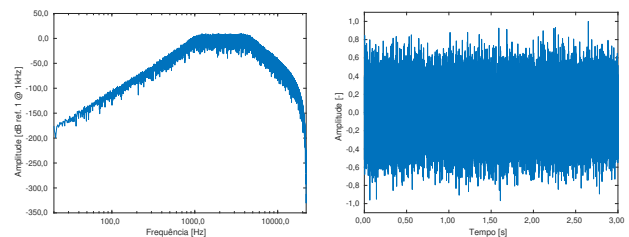


Figura 3: Gráfico no domínio do tempo e na frequência de uma fonte sonora de ruído gaussiano.

4.2 Receptores

Com o intuito de gerar os receptores para as simulações foi criado uma classe chamada *MicArray*. Como o nome já sugere, trata-se de uma abstração para representar um conjunto de microfones (receptores). O objeto dessa classe pode ser inicializado passando-se a posição de cada um dos microfones nos eixos, x, y, z, como mostra o Código 5 (ilustrado na Figura 4 (a)). Também há a possibilidade de gerar arranjos de microfones a partir de configurações pré-definidas, considerando as geometrias espirais e circulares, explicitado nos Códigos 6 e 7, e Figuras 4 (b) e 4 (c). Essa opção pode ser utilizada para em simulações para fins de imageamento acústico.

Finalmente, o método `.plot` da classe *MicArray* plota a figura com a geometria do arranjo gerado.

[6].

⁸Vide <https://bit.ly/3HMqMGz>.

Código 1: Geração de uma fonte tonal

```
freq_values = [1000, 2000, 4000, 8000]; % Frequências tonais
amplitude_values = [80, 85, 79, 82]; % Amplitude cada tonal
sound_source = Source('x',1,'y',1,'Amp',amplitude_values,'Freq', freq_values);
audio = sound_source.CreateAudio('T',3, 'Fs',44100);
audio.plot_freq()
audio.plot_time()
```

Código 2: Geração de uma fonte de ruído gaussiano

```
sound_source = Source('x',0.5,'y',1.7,'z',5,'GaussianNoise',true);
audio = sound_source.CreateAudio('T', 3, 'Fs', 44100)
audio.plot_time()
audio.plot_freq()
```

Código 3: Geração de uma fonte de ruído filtrado

```
filter_noise = [1000, 4500] % Filtrar ruído entre 1000 e 4500Hz
sound_source = Source('x',1,'y',1,'Amp',80,'GaussianNoise', true, 'FilterNoise', 2
    ↳ filter_noise);
audio = sound_source.CreateAudio('T',3, 'Fs',44100);
```

Código 4: Configurando fonte sonora utilizando faixa de áudio

```
sound_source = Source('x',0.5,'y',1.7,'z',5,'LoadAudio', 'audio.wav');
```

Código 5: Configuração manual de um arranjo de microfones.

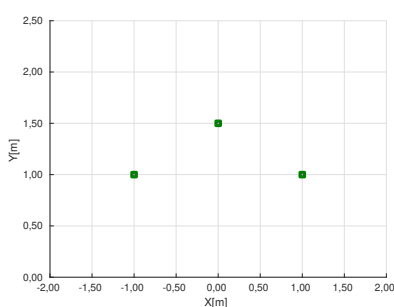
```
array = MicArray('x', [-1.0, 0.0, 1.0], 'y', [1.0, 1.5, 1.0], 'z', [0,0,0]);
figure()
array.plot()
```

Código 6: Configuração de arranjo de microfones com geometria espiral pré-definida.

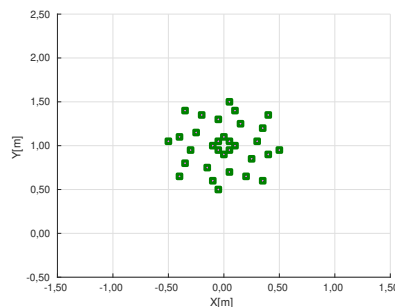
```
array = MicArray;
array.GenerateArray('spiral','H',1);
figure()
array.plot()
```

Código 7: Configuração de arranjo de microfones com geometria circular pré-definida.

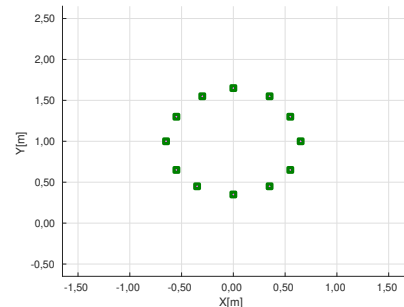
```
array = MicArray;
array.GenerateArray('circle','H',1,'MicpCircle', 12)
figure()
array.plot()
```



(a) Exemplo de arranjo com três microfones gerado manualmente.



(b) Exemplo de arranjo espiral.



(c) Exemplo de arranjo circular.

Figura 4: Exemplos de receptores.



4.3 Simulações de fontes estática e em movimento

Uma vez com as fontes e receptores configurados, é possível combiná-los a fim de gerar uma medição acústica simulada. Aqui, existem duas possibilidades, sendo uma analisar a fonte em repouso (estática), e a outra, analisá-la em movimento. O primeiro caso está demonstrado no Código 8 e ilustrado na Figura 5 (a).

No segundo caso, além de configurar a geometria e posição dos receptores, faz-se necessário definir a trajetória do plano em que as fontes estão posicionadas. Valendo-se das mesmas configurações de sinal sonoro e posição dos receptores para o

caso da fonte estática, o Código 9 apresenta como definir a trajetória de fonte em movimento e rodar a simulação.

No exemplo, as fontes movimentam-se linearmente, na mesma direção e plano, por uma distância de 160 metros em um intervalo de 8 segundos (ou seja, a uma velocidade de 15 m/s). A Figura 5 (b) ilustra o cenário. Ademais, cabe pontuar que a essa velocidade a informação sonora que chega aos receptores sofre o Efeito Doppler (previamente explicado na Seção 2.2).

Código 8: Gerando fontes e receptores da simulação

```
##### Fontes sonoras da simulação
sources = {};

%% Fonte de ruído
sources{1} = Source('x',-.5,'y',.4,'Amp',60,'GaussianNoise', true);

%% Fonte tonal

freq_values = [1000, 2000, 4000, 8000]; % Frequências tonais
amplitude_values = [80, 70, 79, 82]; % Amplitude de cada tonal [dB]
sources{2} = Source('x',0,'y',1,'Amp',amplitude_values,'Freq', freq_values);

%% Fonte de ruído filtrado
filter_noise = [1000, 4500];
sources{3} = Source('x',0.7,'y',1.2,'Amp',65,'GaussianNoise', true, 'FilterNoise', \
    \ filter_noise);

##### Receptores da simulação
microphones = MicArray('x', [-.1, .1], 'y', [1.7, 1.7], 'z', [5, 5]);

##### Rodando a simulação
fs = 44100; % Taxa de amostragem [Hz]
time = 10; % Tempo da simulação
simulation = simulate_measurement_ss(sources,microphones,10,44100);
simulation.plot_freq()
simulation.plot_time()
```

Código 9: Configurando trajetória da fontes fontes sonoras

```
##### Configuração da trajetória da fonte
d_sim = 5; % Distância entre o plano das fontes sonoras e o plano dos microfones

t_initial = 0; % Instante inicial [s]
t_final = 8; % Instante final [s]
x_initial = -60; % Posição inicial [m]
x_final = 60; % Posição final [m]

time_vector = [t_initial t_final];
sources_trajectory = ...
    Trajectory(time_vector,'x',[x_initial x_final],'z',[d_sim,d_sim]);

##### Rodando a simulação
[simulation, sources, simulated_traj] = ...
    simulate_measurement_mvs(sources, microphones, sources_trajectory,'Noise', 100);
```

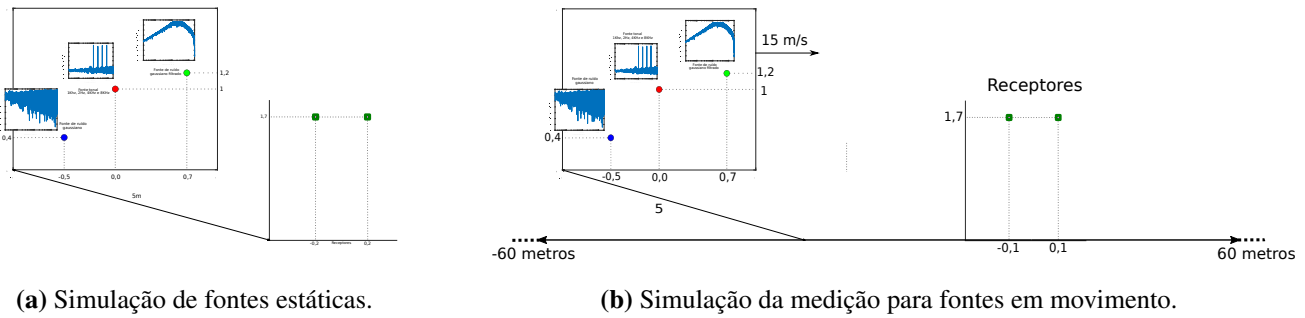


Figura 5: Esquema demonstrando as simulações configuradas.

5. PÓS-PROCESSAMENTO E ANÁLISE

Na Seção 4 foram apresentadas rotinas escritas em Matlab para geração de medições acústicas simuladas, considerando fontes sonoras estáticas e em movimento e utilizando diferentes configurações de fonte/receptor. Uma vez com os dados simulados, pode-se processá-los no intuito de tecer análises a seu respeito. Desse modo, a Seção 5.1 comenta sobre possibilidades de pós-processamento dos sinais de fonte estática, enquanto que a Seção 5.2 apresenta exemplos de processamento para as fontes em movimento.

5.1 Simulação de fonte estática

Com a simulação de fontes sonoras estáticas, há a possibilidade de gerar medições com arranjos de microfones no intuito de testar algoritmos de imageamento acústico, como nos exemplos mostrados nas Figuras 6 (a) e 6 (b). Ainda, ao utilizar a configuração de dois receptores, é possível simular as posições das orelhas de um indivíduo e, juntamente com um banco de Funções de Transferência Relativas à Cabeça (HRTF⁹), gerar arquivos de áudio biauriculares.

5.2 Simulação de fonte em movimento

As medições simuladas com fontes sonoras que estão em movimento possibilitam estudar técnicas de processamentos relacionados a sinais transientes, como o janelamento e espectrograma. Assim, é possível identificar mudanças no sinal de áudio, tanto no domínio do tempo quanto frequência, para cada instante da medição.

A Figura 7, por exemplo, demonstra a variação de amplitude de um sinal no tempo devido ao Efeito

Doppler. Seu contraponto no domínio da frequência pode ser visualizado na Figura 8. Nessa, nota-se o surgimento de dois novos picos em torno das frequências tonais originais da fonte sonora (presentes na Figura 2), ocasionados pela movimentação da fonte em relação ao receptor. Ademais, a Figura 9 apresenta em detalhe cada tonal capturado pelo receptor (em conjunto com o espectro original da fonte).

Uma forma mais intuitiva de observar o Efeito Doppler é utilizando o espectrograma. Como pode ser observado na Figura 14, o espectrograma é capaz de ilustrar a variação de frequência em função do tempo, de modo que torna-se fácil perceber os efeitos da fonte sonora em movimento. Essas Figuras foram geradas por meio da função `spectrogram`, nativa do Matlab, aplicada aos sinais oriundos da simulação (como demonstrado no Código 10).

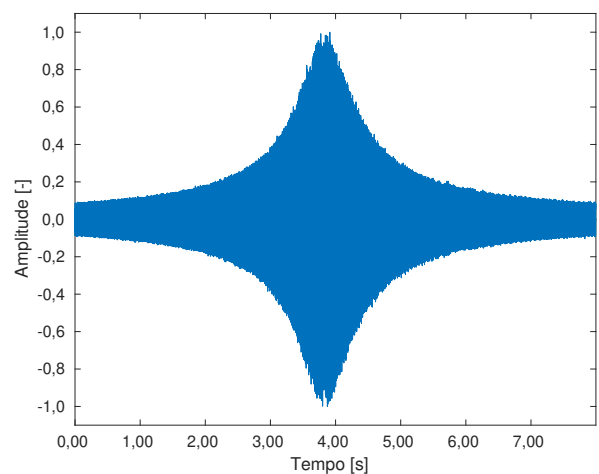
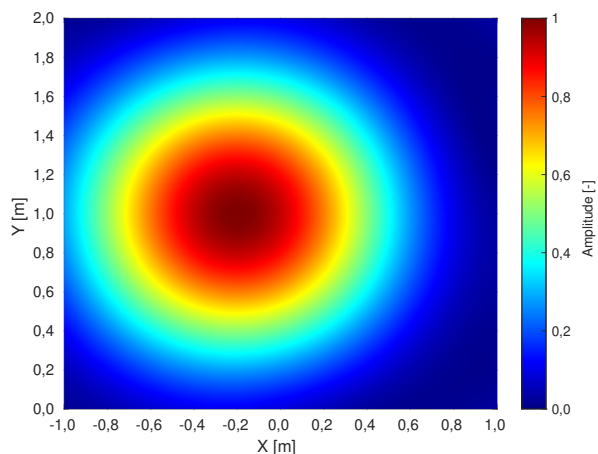
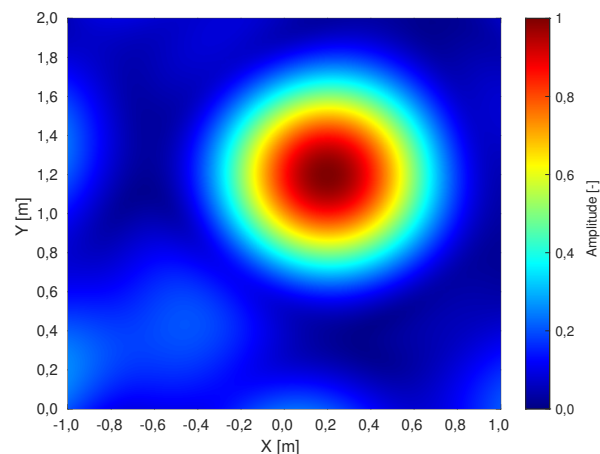


Figura 7: Gráfico no domínio do tempo da simulação da medição para fontes sonoras em movimento.

⁹Do inglês *Head Related Transfer Functions*.



(a) Imagem acústica na frequência de 2 kHz, mostrando a fonte sonora tonal.



(b) Imagem acústica na frequência de 3 kHz, mostrando a fonte de ruído gaussiano filtrado.

Figura 6: Imagens acústicas de uma simulação com um arranjo de microfones espiral, utilizando a mesma configuração de fontes sonoras dos exemplos.

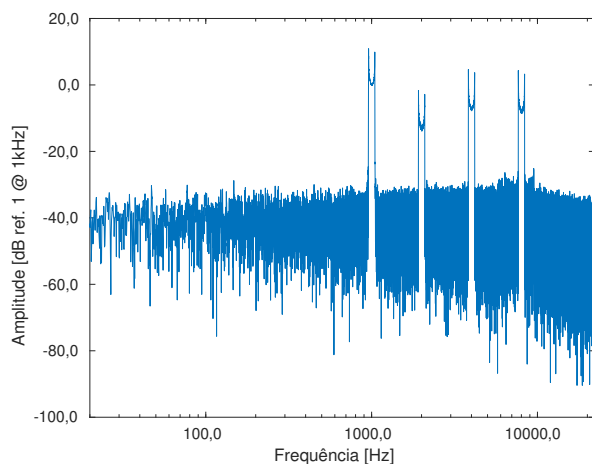


Figura 8: Gráfico no domínio da frequência da simulação da medição para fontes sonoras em movimento.

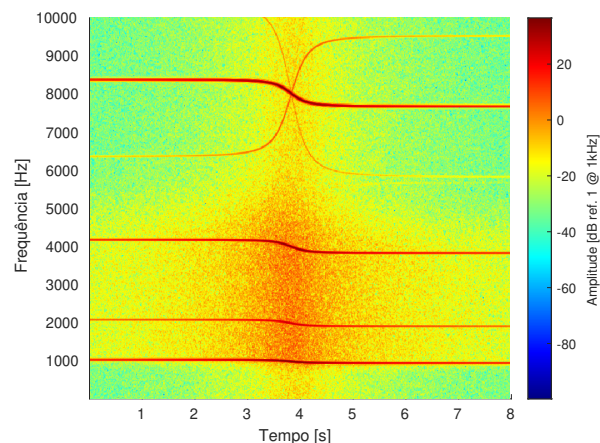


Figura 10: Espectrograma da simulação da medição para fontes sonoras em movimento.

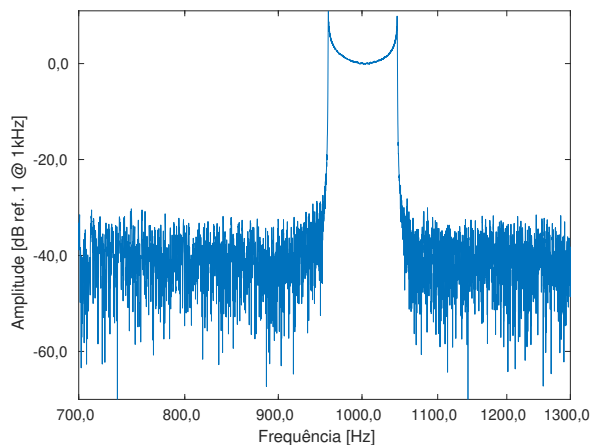


Figura 9: Fazer carrousel com os outros plots com zoom (1000hz, 2000hz, 4000hz, 8000hz)

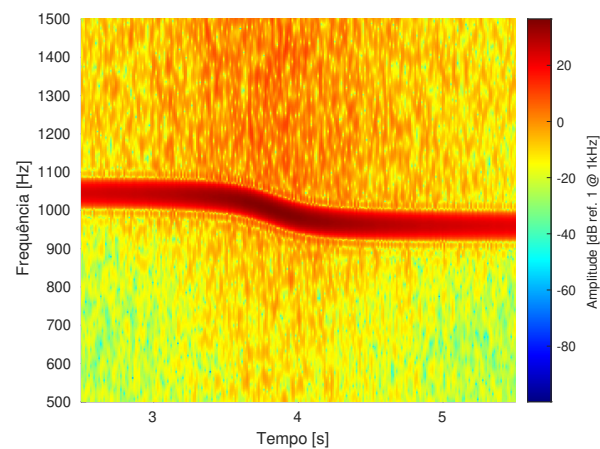


Figura 11: Espectrograma da simulação da medição para fontes sonoras em movimento, em zoom o tonal de de 1 kHz emitido pela fonte sonora.

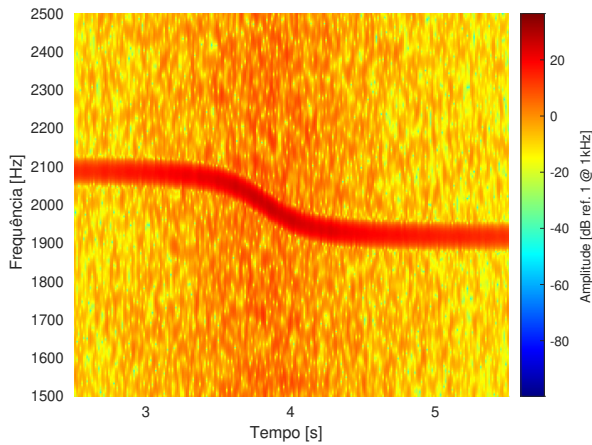


Figura 12: Espectrograma da simulação da medição para fontes sonoras em movimento, em zoom o tonal de de 2 kHz emitido pela fonte sonora.

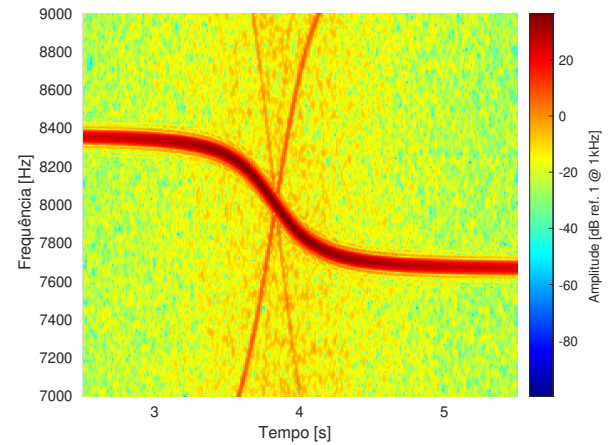


Figura 14: Espectrograma da simulação da medição para fontes sonoras em movimento, em zoom o tonal de de 8 kHz emitido pela fonte sonora.

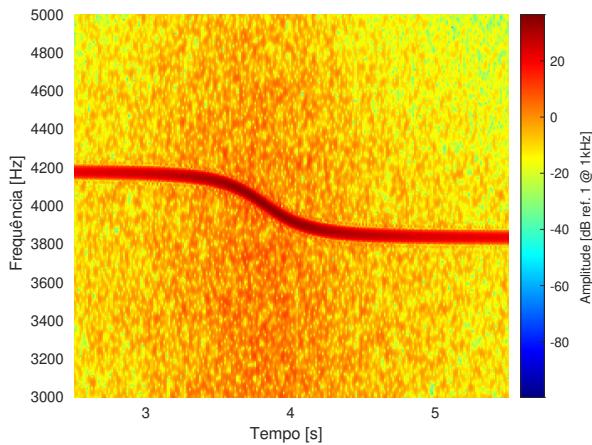


Figura 13: Espectrograma da simulação da medição para fontes sonoras em movimento, em zoom o tonal de de 4 kHz emitido pela fonte sonora.

Para observar em detalhes a variação em frequência de cada um dos tonais emitidos pela fonte, pode-se utilizar funções para identificar seus picos e, aplicando uma janela temporal, acompanhá-los ao longo do tempo, obtendo assim uma função de *frequência instantânea*. O Código 11 demonstra como realizar o processamento. Ademais, as Figuras 15 e 18 apresentam os resultados.

Alguns outros trabalhos, utilizaram-se de estratégias de simulações desse tipo, para testar algoritmos de processamento, ou gerar áudios auralizados. Ferguson e Quinn [7], por exemplo, apresentaram métodos de rastreamento de fontes

Código 10: Processando os dados para as simulações de fontes em movimento

```
n_mic = 1; % Número do microfone
audio_data = simulation.time_data(1, :); % Dados de áudio
fs = simulation.Fs; % Frequência de amostragem

%% Gerando espectrograma
%%% Parâmetros
Δ_f = 1; % Discretização do vetor de frequências
f_range = 0:Δ_f:round(fs/2); % Valores de frequência

l_win = 2048; %Tamanho da janela escolhida
win_data = hann(l_win); % Janela escolhida para o espectrograma
overlap_percentage = 0.75; % Porcentagem de overlap
n_overlaps = round(l_win*overlap_percentage); %Número de samples para overlap

%% Processando espectrograma
[s,f,t] = spectrogram(audio_data,win_data,n_overlaps,f_range,fs);

%% Plotando espectrograma

spectrogram(audio_data,win_data,n_overlaps,f_range,fs, 'yaxis');
```



sonoras, para medições com um único receptor, utilizando o processamento de identificação de frequências instantâneas oriundas Efeito Doppler.

No trabalho de Tapolezai et al. [8], realizou-se uma medição em câmara anecoica de um *drone* para identificação da frequência principal emitida pelas pás. Com essa informação, gerou-se um cenário simulado com fonte similar em movimento, capturada por um arranjo de 24 microfones virtuais, no intuito de testar métodos de rastreamento desse tipo de fonte sonora.

Focado na área de auralização, o trabalho de Gomes et al. [9] fez uso do mesmo método de simulação utilizado no *toolbox*, com o objetivo de gerar áudios binauriculares referentes a fontes de movimento. Para isso, identificou-se os ângulos e distâncias da fonte sonora em relação aos receptores, configurados para simular a posição das orelhas de um indivíduo. Os dados medidos foram, então, convoluídos com HRTFs correspondentes às direções de chegada ao longo do trajeto da fonte.

Ademais, no trabalho de Masiero et al. [10],

utilizou-se técnicas para identificação de fontes sonoras em uma medição de ruído de passagem. Nesse caso, os sons emitidos por um automóvel em movimento foram capturados utilizando um arranjo de microfones e, a partir de técnicas de processamento, as fontes de interesse foram identificadas e combinadas com HRTFs para geração de faixas de áudio binauriculares.

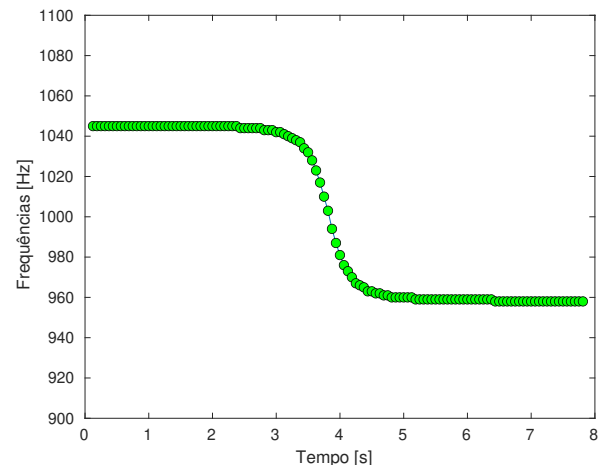


Figura 15: Frequência instantânea para rastrear o tonal de 1 kHz emitido pela fonte sonora

Código 11: Processando frequência instantânea

```
n_mic = 1; % Número do receptor
audio_data = simulation.time_data(1, :); % Dados de áudio
fs = simulation.Fs; % Frequência de amostragem

f_zoom = [800, 1200]; % Intervalo de frequências para procurar os picos máximos
f_dist = 1; % Discretização do vetor de frequências

f_range = f_zoom(1):f_dist:f_zoom(2);

t_interval = .25; % Tamanho da janela [s]
l_win = round(t_interval*fs); % Tamanho da janela [samples]
win_data = hann(l_win); % Janela escolhida para o espectrograma
overlap_percentage = 0.75;
n_overlaps = round(l_win*overlap_percentage); % Número de samples para overlap

[s,f,t] = spectrogram(audio_data,win_data,n_overlaps,f_range,fs);

f_max_values = zeros(length(t),1);
f_max_amplitudes = zeros(length(t),1);

for n=1:1:length(t)
    [max_amp, max_idx] = max(s(:, n));
    f_max_values(n) = f(max_idx);
    f_max_amplitudes(n) = s(max_idx, n);
end

figure()
plot(t, f_max_values, 'o-', 'MarkerSize', 6, ...
     'MarkerEdgeColor', 'green', ...
     'MarkerFaceColor', [1 .6 .6])
xlabel("Tempo [s]")
ylabel("Frequências [Hz]")
```

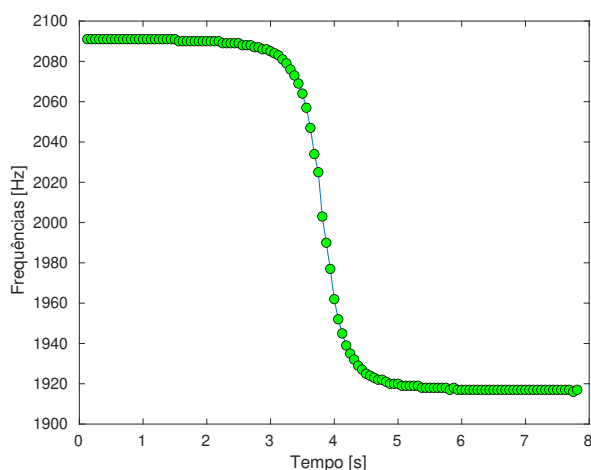


Figura 16: Frequência instantânea para rastrear o tonal de 2 kHz emitido pela fonte sonora

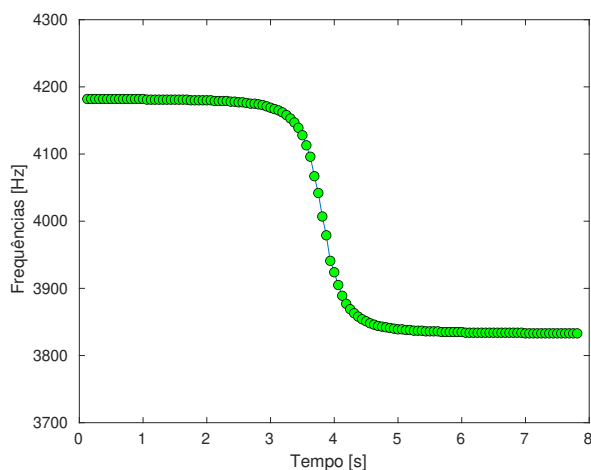


Figura 17: Frequência instantânea para rastrear o tonal de 4 kHz emitido pela fonte sonora

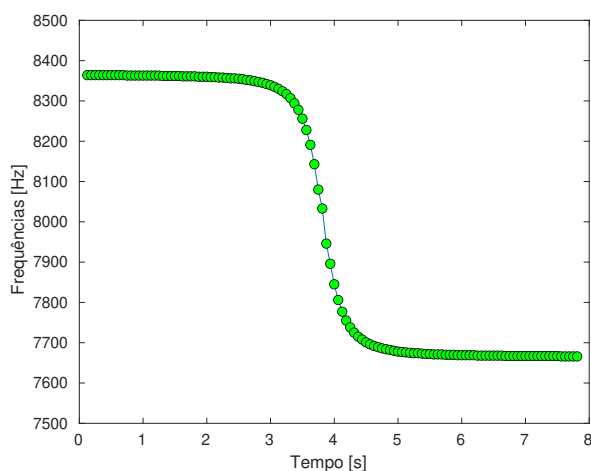


Figura 18: Frequência instantânea para rastrear o tonal de 8 kHz emitido pela fonte sonora

6. CONSIDERAÇÕES FINAIS

No presente trabalho foram apresentadas algoritmos desenvolvidos para simulação de medições para diversos tipos de fontes sonoras e configurações de receptores. Também foram demonstrados rotinas completas, no formato de tutorial, para que o leitor possa gerar suas próprias simulações.

Além disso também foram demonstrados alguns casos reais que podem ser simulados, bem como algumas possíveis métodos de processamento que podem ser testados utilizando os dados simulados.

7. AGRADECIMENTOS

Os autores agradecem ao curso de Engenharia Acústica (EAC), da Universidade Federal de Santa Maria (UFSM), pelo apoio e infraestrutura fornecidos. Agradecemos também aos seus programas internos FIPE-CT e FIEEX-CT, por financiarem e apoiarem este projeto.

REFERÊNCIAS

- [1] Rossing, Thomas. *Springer handbook of acoustics*. Springer Science & Business Media, 2007.
- [2] Meng, Fanyu; Jax, Peter Johannes e Vorländer, Michael. Modeling of moving sound sources based on array measurements. Technical report, Lehrstuhl und Institut für Technische Akustik, 2018.
- [3] Morse, Philip McCord e Ingard, K Uno. *Theoretical acoustics*. Princeton university press, 1986.
- [4] Oppenheim, Alan V. e Schafer, Ronald W. *Digital Signal Processing*. 1ª ed. Prentice-Hall, 1975. ISBN 978-0132146357.
- [5] Shin, Kihong e Hammond, Prof Joseph. *Fundamentals of Signal Processing for Sound and Vibration Engineers*. John Wiley & Sons, 2008. ISBN 978-0470511886; 0470511885.
- [6] Matlab. Object-oriented programming in matlab. Disponível em: <https://www.mathworks.com/products/matlab/object-oriented-programming.html>.
- [7] Ferguson, Brian G e Quinn, Barry G. Application of the short-time fourier transform and the wigner-ville distribution to the acoustic localization of aircraft. *The Journal of the Acoustical Society of America*, 96(2): 821–827, 1994.
- [8] Tapolczai, Péter; Fiala, Péter; Firtha, Gergely e Rucz, Péter. Simulation framework for detecting and tracking moving sound sources using acoustical beam-forming methods. Em *Conference: DAGA2019 45. Jahrestagung für AkustikAt: Rostock, Germany*, 2019. Disponível em: https://www.researchgate.net/publication/332371929_Simulation_framework_for_detec



[ting_and_tracking_moving_sound_sources_using_a_coustical_beamforming_methods?isFromSharing=1](#).

- [9] Gomes, L M; Fonseca, W D'Andrea; Carvalho, D e H Mareze. RENDERING BINAURAL SIGNALS FOR MOVING SOURCES. Em *REPRODUCED SOUND 2020*. Institute of Acoustics, Dezembro 2020. doi: [10.25144/13386](https://doi.org/10.25144/13386). Disponível em: <https://doi.org/10.25144/13386>.
- [10] Masiero, Bruno S; Fonseca, William D; Müller-Trapet, Markus e Dietrich, Pascal. Auralization of pass-by beamforming measurements. 2010. doi: [10.13140/RG.2.1.4455.9208](https://doi.org/10.13140/RG.2.1.4455.9208). Disponível em: <http://rgdoi.net/10.13140/RG.2.1.4455.9208>.