PROJETO DE GRADUAÇÃO

Análise de problemas acústicos usando o método dos elementos de contorno

Por,
Fernando Barreto Soares

Brasília, 11 de Maio de 2021

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

Faculdade de Tecnologia

Departamento de Engenharia Mecânica

PROJETO DE GRADUAÇÃO

Análise de problemas acústicos usando o método dos elementos de contorno

POR,

Fernando Barreto Soares

Relatório submetido como requisito parcial para obtenção

do grau de Engenheiro Mecânico

Banca Examinadora

Prof. Eder Lima de Albuquerque UnB/ ENM (Orientador)	_____
Álvaro Campos Ferreira UnB/ ENM (Co-orientador)	_____
Prof. Marcus Vinicius Girão de Morais UnB/ ENM	_____
Prof. Marcela Rodrigues Machado UnB/ ENM	_____

Brasília, 11 de Maio de 2021

FICHA CATALOGRÁFICA
FERNANDO BARRETO SOARES
Análise de problemas acústicos usando o método dos elementos de contorno
2021xv, 125p., 201x297 mm
(ENM/FT/UnB, Engenheiro Mecânico, Engenharia Mecânica, 2021)
Projeto de Graduação - Universidade de Brasília
Faculdade de Tecnologia - Departamento de Engenharia Mecânica

REFERÊNCIA BIBLIOGRÁFICA

FERNANDO BARRETO SOARES (2021) Análise de problemas acústicos usando o método dos elementos de contorno. Projeto de Graduação em Engenharia Mecânica, Publicação xxx/AAAA, Departamento de Engenharia Mecânica, Universidade de Brasília, Brasília, DF, 125p.

CESSÃO DE DIREITOS

AUTOR: Fernando Barreto Soares
TÍTULO: Análise de problemas acústicos usando o método dos elementos de contorno.
GRAU: Engenheiro Mecânico ANO: 2021

Fernando Barreto Soares
fernando.bsoares3@gmail.com

Agradecimentos

Gostaria de agradecer, primeiramente, ao meu orientador Eder Lima de Alburquerque, que mesmo nos atuais tempos de incerteza e dificuldade não mediu esforços para me auxiliar neste trabalho. Também a Álvaro Campos Ferreira, quem me ajudou em reuniões semanais e foi de extrema importância, especialmente na parte de simulação numérica utilizando o programa BB. Também gostaria de agradecer a toda a banca examinadora, em especial ao professor Marcus Vinicius Girão, pela ajuda em corrigir alguns resultados das simulações nas cavidades fechadas. Por último, à minha família e meus amigos, que sempre me deram o suporte e a tranquilidade para que pudesse realizar meus estudos com a dedicação e persistência necessárias.

Resumo

Uma comparação entre dois códigos *open-source* do método dos elementos de contorno (MEC) para problemas acústicos interiores e exteriores é realizada neste trabalho. Os programas utilizado neste projeto foram o Bempp-cl e o BB. Enquanto o Bempp-cl é um programa em Python bem estabelecido desenvolvido na University College London e na Universidade de Cambridge, o BB é um código mais recente, desenvolvido na Universidade de Brasília, escrito na linguagem Julia. Ambos os programas MEC também se destacam pelo fato de permitirem o uso de métodos de aceleração para a montagem da matriz, o que é especialmente importante na resolução de programas de grande escala. Para a geração da malha, um programa também de código aberto, Gmsh, foi usado. Inicialmente, foram resolvidos problemas internos em uma geometria cúbica, tanto em termos da análise da resposta nas frequências ressonantes quanto na análise da resposta em frequência para os três primeiros modos do problema. Para o mesmo problema, também foi realizada uma análise do tempo de processamento, a fim de avaliar a eficiência dos programas mencionados. Por fim, foi obtida a resposta para um ressonador acústico, o *TinyLev*, para mostrar a capacidade de resolver problemas de engenharia mais complexos e de grande escala. As soluções obtidas pelos dois programas foram comparadas entre si e também com soluções analíticas, quando disponíveis. Os resultados para a geometria cúbica mostram que ambos os programas foram capazes de prever, com sucesso, o comportamento dos modos e da análise de frequência, com um consumo de tempo satisfatório. Além disso, o MEC se mostrou capaz de predizer adequadamente a resposta esperada do levitador acústico. Portanto, a utilização do MEC e, principalmente, de ambos os programas estudados neste artigo se comprovaram como sendo ferramentas muito poderosas, disponíveis gratuitamente, para resolver grandes e complexos problemas de engenharia, com grande eficiência e precisão.

PALAVRAS CHAVE: método dos elementos de contorno, acústica.

Abstract

A comparison between two competing free and open-source codes of the boundary element method (BEM) for internal and external acoustic problems is carried out in this work. The BEM programs used were Bempp-cl and BB. While Bempp-cl is a well established Python program developed at University College London and the University of Cambridge, BB is a more recent code, developed at the University of Brasília, written in Julia language. Both BEM programs also stand out by the fact that they allow the use of acceleration methods for the matrix assembly, which is especially important when solving large scale programs using the BEM. For the mesh generation an also open-source program, Gmsh, was used. Initially, internal problems on a cubic geometry were solved, both in terms of response analysis in the resonant frequencies as well as frequency response analysis for the three first modes of the problem. For the same problem, a processing time analysis was also conducted, in order to assess the efficiency of the mentioned programs. Finally, the response for an acoustic resonator, the TinyLev, was obtained, in order to show the capability of solving more complex, large scale engineering problems. The solutions obtained by the two programs were compared with each other as well as with analytical solutions, when available. The results for the cubic geometry show that both programs were able to successfully predict the behavior for the modes and for the frequency analysis, with a satisfactory time consuming. Besides that, both programs were able to properly predict the acoustic levitator expected response. Therefore, the use of BEM and, specially, of both programs studied in this article are very powerful tools, freely available, to solve large and complex engineering problems, with great efficiency and accuracy.

KEYWORDS: boundary element method, acoustics.

Sumário

LISTA DE FIGURAS

LISTA DE SÍMBOLOS

Símbolos Latinos

$\overline{u}$	Condição de campo prescrito
$\overline{u}_n$	Condição de fluxo prescrito
a	Símbolos em **negrito** são vetores
i	Vetor unitário na direção x
j	Vetor unitário na direção y
n	Vetor unitário normal
t	Vetor unitário tangente
v	Velocidade da partícula fluida
θ_{int}	Ângulo interno do contorno
c	Velocidade do som
f	Fonte acústica
G	Solução fundamental
I	Intensidade acústica
IL	Nível de intensidade
k	Número de onda
K'_k	operador *adjoint double-layer*
K_k	operador *double-layer*
p	Pressão acústica
Q	Intensidade da fonte
R	Constante do gás ideal
r	Distância entre o ponto fonte e o ponto campo

$Re()$ Parte real

SPL Nível de pressão sonora

T Temperatura

u^I Onda incidente

u^S Onda espalhada

V_k operador *single-layer*

W_k operador *hypersingular*

Símbolos Gregos

δ Delta de Dirac

η Constante de acoplamento da equação de Burton-Miller

Γ Contorno

γ Razão da capacidade térmica

λ Comprimento de onda

Ω Domínio

ω Frequência angular

ϕ Potencial da velocidade

ρ massa específica

Capítulo 1

INTRODUÇÃO

1.1 Contexto

A acústica está presente diariamente na vida da grande maioria das pessoas. Aquilo que nós, seres humanos, percebemos como som, nada mais são do que oscilações no campo de pressão que ocorrem no ar ao redor de nossos ouvidos. Tal fenômeno, tão presente no ambiente, não poderia deixar de ser objeto de estudo na engenharia, seja para garantir o conforto acústico das pessoas ou seja em aplicações de alta tecnologia. Alguns exemplos se encontram no estudo de automóveis, aeronaves, salas de concerto, submarinos e foguetes. Também existem aplicações menos usuais, como a que foi tratada neste projeto, o levitador acústico (MARZO; BARNES; DRINKWATER, 2017).

Considere, por exemplo, a indústria automobilística. Atualmente, com a crescente competitividade e exigência cada vez maior de qualidade dos automóveis pelos passageiros, não basta que o veículo desempenhe todas as funções esperadas, como também deve garantir um ambiente confortável e agradável. Para garantir tal ambiente, a preocupação em manter os níveis sonoros na cabine interna reduzidos é cada vez maior (BRIZON, 2012; BRANCATI, 2010). Como a preocupação do estudo acústico, neste caso, é na cabine do veículo, ou seja, em um domínio fechado, totalmente limitado pelas superfícies do automóvel, este pode ser classificado como um problema interior.

Ainda na indústria automobilística, o problema acústico exterior também é de interesse de estudo. Um problema exterior é caracterizado por um domínio que não é totalmente cercado por superfícies, sendo assim infinito. Por exemplo, o problema do ruído gerado por um veículo possui como domínio todo o ar exterior às superfícies do automóvel. Especialmente nas grandes metrópoles, milhares de veículos interagem em meio a áreas residenciais com moradores e pedestres, os quais são expostos a níveis de ruído elevados e praticamente constantes. Para amenizar tal problema, automóveis novos devem passar por rígidos testes de nível de ruído para poderem ser produzidos e comercializados (ABNT, 2000). A redução do nível de ruído é feita com o controle da combustão e, principalmente, com o uso de abafadores automotivos, ou *muffler*, como o mostrado na Figura 1.

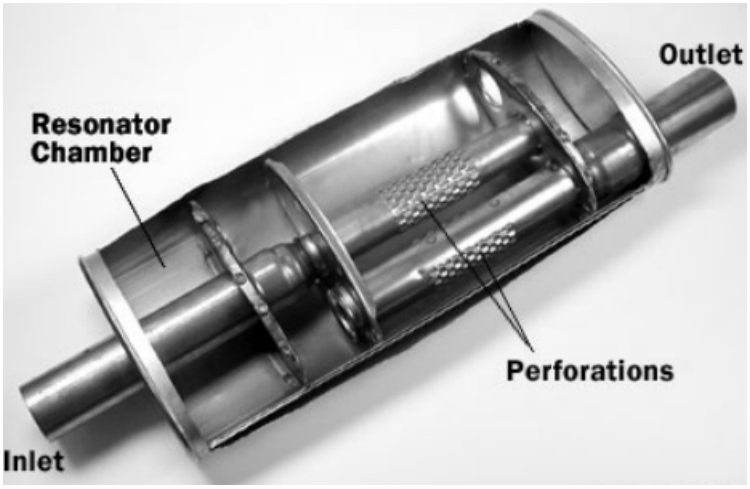


Figura 1: Exemplo de abafador automotivo (POTENTE, 2005)

Outra indústria com grande interesse no estudo da acústica é a aeronáutica. Principalmente quando se trata de voos de longa duração, passageiros são expostos por horas a ruídos internos na aeronave, o que pode piorar significativamente a experiência do passageiro. Uma atenção especial deve ser dada a ruídos entre 2 Hz e 10 Hz uma vez que, mesmo que não sejam percebidos pelo ouvido humano, tendem a ressonar com a frequência natural de diversas partes do corpo. A principal fonte de ruído é o sistema de propulsão, mas também há uma contribuição dos componentes da fuselagem e dos outros passageiros (BRANCATI, 2010).

Na análise acústica de problemas de engenharia utilizam-se, frequentemente, métodos numéricos. Como a engenharia trata de problemas reais, frequentemente soluções puramente analíticas são impossíveis ou muito imprecisas devido a simplificações na formulação, levando a uma quase que necessidade de se utilizar um método numérico. No estudo da acústica, o método dos elementos de contorno (MEC) se destaca como sendo o mais eficiente, principalmente para problemas exteriores, e é utilizado pelos principais pacotes comerciais presentes na indústria (BRANCATI, 2010).

1.2 Objetivos

Este projeto tem como objetivo geral explorar o MEC aplicado à acústica e as ferramentas disponíveis atualmente. Foram utilizados exclusivamente *softwares* de código aberto, disponíveis para qualquer pessoa de forma totalmente gratuita. Além de explorar essas ferramentas, a teoria acústica e a teoria do MEC foi explicada de forma sucinta, o que certamente fará deste projeto uma boa leitura inicial para um leitor interessado no assunto.

Em especial, o programa de código aberto BB[1], desenvolvido na Universidade de Brasília pelo grupo UnBEM, foi utilizado para resolver alguns problemas acústicos. Como se trata de um programa relativamente novo, ainda há muito o que ser desenvolvido e explorado.

[1]Disponível em: <https://github.com/alvarocafe/BB>.

Em sequência, o programa Bempp[2], desenvolvido no *University College London* e na Universidade de Cambridge, foi comparado ao software BB. Tal comparação foi feita tanto em termos de eficiência quanto em termos de precisão dos resultados obtidos. Uma vez que o Bempp é um programa já relativamente bem estabelecido, servirá simultaneamente como *benchmark* e validação para o código desenvolvido na UnB.

De forma mais direta, os objetivos específicos podem ser sintetizados em:

- Comparar as respostas para problemas acústicos utilizando os códigos abertos Bempp e BB, além de comparar com respostas obtidas analiticamente;
- Implementar o problema levitador acústico, conforme apresentado por Marzo, Barnes e Drinkwater (2017), no Bempp. O problema já foi, anteriormente, implementado no BB (FERREIRA, 2020). Em seguida, os resultados obtidos em cada programa foram comparados.

A comparação das soluções geradas pelos programas com soluções analíticas serve para quantificar a precisão desses programas, servindo como base para avaliar a confiabilidade dos resultados. Além disso, considerando que o Bempp é um programa mais antigo e bem estabelecido, este projeto visa compará-lo ao programa BB, de forma a avaliar a precisão do programa desenvolvido na Universidade de Brasília com relação a um software renomado. Essa comparação não se limitou à precisão dos resultados, mas também ao tempo de processamento necessário para alcançar tal solução, já que esse parâmetro também é de suma importância para a viabilidade de um programa de simulação numérica. Também será interessante notar a diferença de desempenho considerando que os dois programas são escritos em linguagens diferentes, utilizam métodos diferentes do MEC e possuem possibilidade de aceleração na geração das matrizes utilizando métodos distintos.

1.3 Metodologia

Antes de iniciar os procedimentos práticos, é necessário se estudar e se conhecer com solidez a parte teórica. Com a difusão cada vez maior de *softwares* de simulação numérica, é cada vez mais comum que usuários resolvam problemas de engenharia sem conhecer, de forma adequada, a teoria por trás dos programas. Isso pode, frequentemente, resultar em resultados imprecisos ou, até mesmo, incorretos. Sendo assim, torna-se essencial o estudo da teoria do MEC. E, uma vez que este projeto é focado em resolver problemas de acústica, também é necessário um conhecimento mínimo da teoria acústica. Por esses motivos, neste projeto, antes das simulações numéricas, as teorias relevantes foram apresentadas.

Exposta a teoria, são apresentados os resultados. Este trabalho utilizará de programas de código aberto para resolver problemas acústicos utilizando-se o MEC. Os dois programas utilizados foram o Bempp e o BB. Problemas bi e tri-dimensionais foram explorados. Inicialmente foram resolvidos e comparados problemas simples. Posteriormente, foram realizados problemas

[2]Disponível em: <http://bempp.com>.

mais complexos. Em destaque, o problema do levitador acústico foi resolvido e comparado. Deu-se preferência a problemas cuja solução analítica é conhecida, de forma que a precisão dos resultados obtidos possa ser avaliada para ambos os programas.

1.4 Organização do trabalho

Este projeto está dividido em seis capítulos. Este capítulo, a introdução, serve como prévia para o restante do projeto. O segundo capítulo irá apresentar o básico da teoria acústica. O terceiro capítulo tratará do MEC aplicado à acústica. Apresentada a teoria relevante, no capítulo quatro foram realizadas simulações numéricas utilizando o código BB e o Bempp. Por último, a conclusão traz um resumo e os principais resultados deste projeto. Nos apêndices estão disponíveis os códigos utilizados pelo autor. Nos apêndices estão disponíveis os principais códigos utilizados neste projeto, e nos anexos são apresentados alguns fundamentos matemáticos necessários para o desenvolvimento da teoria do MEC e alguns conceitos acústicos adicionais que podem ser de interesse para alguns leitores.

Capítulo 2

ACÚSTICA

A teoria acústica é o estudo das oscilações das propriedades termodinâmicas em um meio, em especial, da pressão. Essas oscilações se propagam na forma de ondas sonoras, e são percebidas diariamente na forma de ruído.

Neste capítulo, é feita uma breve revisão de alguns conceitos fundamentais da teoria acústica, para que, posteriormente, essa teoria seja inserida no contexto dos métodos numéricos, ou, mais especificamente, do MEC.

2.1 A corda vibrante

Antes de tratar da propagação de ondas sonoras, serão discutidas algumas equações e alguns conceitos básicos, utilizando-se do que se trata, provavelmente, da forma mais simples de propagação de ondas: a propagação de ondas transversais em uma corda. Muitos dos conceitos e equações apresentados nesta seção, como o número de onda, a velocidade de propagação da onda, a onda estacionária, as ressonâncias e anti-ressonâncias e as autofrequências, são também válidos para casos mais complexos, como a propagação de ondas sonoras.

Primeiramente, por hora sem nenhuma demonstração matemática, apresenta-se a equação da onda uni-dimensional:

$$\frac{\partial^2 y}{\partial x^2} = \frac{1}{c^2}\frac{\partial^2 y}{\partial t^2}, \tag{2.1}$$

em que x é a direção longitudinal da corda, y é o deslocamento transversal e c é uma constante real entendida como sendo a velocidade de propagação da onda na corda.

Além da velocidade de propagação da onda, c, é importante definir a constante k, chamada de número de onda:

$$k = \frac{\omega}{c}, \tag{2.2}$$

onde ω é a frequência em radianos por segundo, dada por $\omega = 2\pi f$, sendo f a frequência em Hz.

Já o comprimento de onda, dado por λ, é definido, em uma onda harmônica, por

$$\lambda = \frac{2\pi}{k}. \tag{2.3}$$

Assim, é trivial demonstrar que

$$c = \lambda f. \tag{2.4}$$

Considere, agora, uma corda que possui comprimento finito, igual a L. Em regime permanente, a solução da equação 2.1 pode ser expressa em termos de duas ondas harmônicas viajando em direções opostas:

$$y(x,t) = Ae^{j(\omega t - kx)} + Be^{j(\omega t + kx)}, \tag{2.5}$$

onde $j = \sqrt{-1}$ e A e B são as amplitudes complexas, determinadas aplicando-se as condições de contorno.

2.1.1 A corda forçada, fixa

A corda forçada, fixa, é conduzida por um forçamento F em sua extremidade $x = 0$ e fixada na extremidade $x = L$. Aplicando-se tais condições de contorno, obtém-se:

$$y(x,t) = \frac{F}{kT}\frac{sin[k(L-x)]}{coskL}e^{j\omega t}, \tag{2.6}$$

onde T é o período.

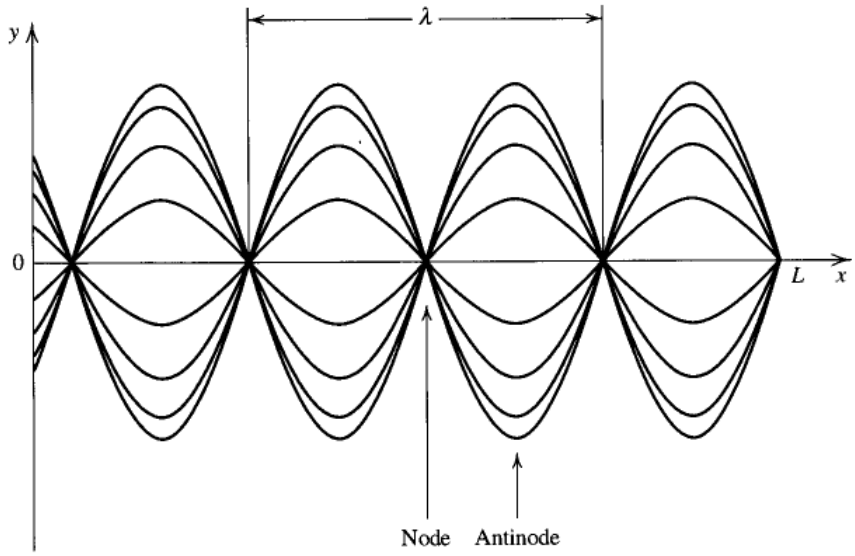


Figura 2: Forma de onda de uma corda de comprimento L para diferentes tempos (KINSLER et al., 2000)

Analisando a equação 2.6, bem como a Figura 2, é possível notar que a forma da onda não se propaga ao longo da corda. Ao invés disso, a corda oscila enquanto a sua forma é preservada. Esse tipo de onda é chamada de onda estacionária, e é caracterizada matematicamente por uma amplitude que depende da posição longitudinal da corda. Esse tipo de onda está presente em diversas aplicações na acústica, como por exemplo no levitador acústico *TinyLev*, que será apresentado neste trabalho.

Dessa forma, percebe-se que esta onda já foi descrita de duas formas: combinação de ondas viajando em direções opostas e onda estacionária. O entendimento de ondas estacionárias como uma combinação de duas ondas harmônicas é utilizado frequentemente no estudo da acústica.

A posição dos nós e anti-nós varia em função da frequência. Tal mudança na posição também tem um forte efeito na amplitude máxima dos anti-nós. Dessa forma, definem-se as frequências de ressonância f_{rn}, que são as frequência que resultam na máxima amplitude de oscilação da corda. No caso da equação 2.6, as frequências de ressonância são determinadas quando $\cos kL = 0$, ou seja,

$$f_{rn} = \frac{(2n-1)}{4}\frac{c}{L}, \tag{2.7}$$

onde $n = 1, 2, 3, ...$

De forma similar, também é possível definir as frequências de anti-ressonância f_{arn}, como as frequências em que se tem a amplitude mínima:

$$f_{arn} = \frac{n}{2}\frac{c}{L}. \tag{2.8}$$

2.1.2 A corda fixa, fixa

A corda fixa, fixa, é fixada em ambas suas extremidades: $x = 0$ e $x = L$. Ao invés de seu movimento ser criado por um forçamento, considere que a excitação é gerada por um determinado deslocamento inicial.

A imposição das condições de contorno na equação 2.5 resulta na equação 2.9

$$2jA \sin kL = 0. \tag{2.9}$$

Como $A = 0$ resulta na solução trivial de movimento inexistente, torna-se necessário que $\sin kL = 0$, ou seja:

$$kL = n\pi, \tag{2.10}$$

onde $n = 1, 2, 3, ...$

Assim, as soluções para o problema estão restritas às seguintes frequências:

$$f_n = \frac{\omega_n}{2\pi} = \frac{n/2}{c/L}. \tag{2.11}$$

Tais frequências únicas são chamadas de autofrequência ou frequência natural. O entendimento da existência de tais frequências será essencial para o estudo da acústica em cavidades fechadas e do método de Burton-Miller.

Além disso, a aplicação das condições de contorno limitou as soluções viáveis da equação da onda a uma série de equações:

$$y_n(x,t) = (A_n \cos \omega_n t + B_n \sin \omega_n t) \sin k_n x, \tag{2.12}$$

chamadas de autofunções ou modos normais. O modo normal de menor frequência natural ($n = 1$) é chamado de modo fundamental. A frequência natural associada é chamada de frequência fundamental ou primeiro harmônico. As frequências naturais para $n = 2, 3, ...$ são chamadas de sobretons.

2.2 A equação da onda sonora tri-dimensional

Nesta seção, é apresentada a equação da onda, a qual é a base da teoria acústica. Devido ao fim deste projeto, não foi feita uma demonstração completa demonstrando passo a passo, e sim uma demonstração simplificada.

As propriedades acústicas oscilatórias são caracterizadas por um valor de equilíbrio e uma flutuação em torno deste valor. A soma dos dois é o valor instantâneo da propriedade. Algumas dessas propriedades são:

$$\mathbf{v} = \mathbf{V} - \mathbf{V}_0; \tag{2.13}$$

$$p = P - P_0; \tag{2.14}$$

$$\rho = \rho' - \rho_0; \tag{2.15}$$

onde $\mathbf{v}$ é a velocidade da partícula fluida, p é a pressão acústica e ρ é a massa específica (ou densidade). As variáveis em letras maiúsculas, como P e $\mathbf{V}$, e a variável ρ' representam o valor instantâneo, as variáveis marcadas com X_0, como P_o, $\mathbf{V}_0$ e ρ_0, representam o valor de equilíbrio (ou valor médio) e as variáveis sem marcação alguma representam a flutuação em torno do ponto de equilíbrio.

Também é útil definir o potencial da velocidade, dado por:

$$\mathbf{v} = \nabla\Phi. \tag{2.16}$$

Apesar de ser uma quantidade razoavelmente abstrata, o potencial Φ é conveniente para diversos cálculos em acústica. A equação 2.16 só é válida para fluidos irrotacionais. Em geral, essa propriedade é imposta pela definição do campo acústico (RIENSTRA; HIRSCHBERG, 2016).

Antes de proceder, é essencial expor quais as hipóteses simplificadoras adotadas para se obter a equação da onda. São elas: fluído invíscido, não existência de escoamento, médias da densidade e da pressão são uniformes pelo fluido e a flutuação das propriedades é muito menor que seu valor médio (FERREIRA, 2012).

Agora, são apresentadas algumas equações clássicas de mecânica dos fluidos, utilizadas para se obter a equação da onda sonora. Primeiramente, a equação da continuidade é dada por:

$$\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{v}) = 0. \tag{2.17}$$

Aplicando as equações 2.13 e 2.15, bem como as simplificações citadas, é possível obter a equação da continuidade linearizada:

$$\frac{\partial\rho}{\partial t} + \rho_0(\nabla\cdot\mathbf{v}) = 0. \tag{2.18}$$

De forma semelhante, utilizando as mesmas hipóteses, é possível obter a equação da quantidade de movimento linearizada, ou equação de Euler linearizada:

$$\rho_0\frac{\partial\mathbf{v}}{\partial t} + \nabla p = 0. \tag{2.19}$$

Por fim, é necessário determinar a equação de estado. Considerando as hipóteses utilizadas anteriormente, adicionando a hipótese de um processo isentrópico, tem-se que

$$p = c^2\rho, \tag{2.20}$$

onde c é a velocidade termodinâmica do som no meio.

Utilizando-se das equações 2.18 a 2.20, além de mais algumas operações matemáticas, chega-se na equação da onda sonora linearizada sem perdas, ou, simplesmente, equação da onda:

$$\nabla^2 p = \frac{1}{c^2}\frac{\partial^2 p}{\partial t^2}. \tag{2.21}$$

2.3 Ondas planas harmônicas

Inicialmente, faz-se necessário apresentar um tipo de onda muito importante para o estudo da acústica: a onda harmônica. A onda harmônica é aquela de caráter senoidal, que se repete periodicamente no tempo. Utilizando a notação complexa de quantidades acústicas tem-se, para um onda harmônica:

$$p(\mathbf{x}, t) = u(\mathbf{x})e^{-j\omega t}. \tag{2.22}$$

Lembrando-se de que, para obter a pressão física do problema, é necessário considerar apenas a parte real, tem-se que

$$p(\mathbf{x}, t) = \mathrm{Re}(u(\mathbf{x})e^{-j\omega t}), \tag{2.23}$$

onde Re() é a parte real da quantidade complexa e $u(\mathbf{x})$ é uma função que representa a parte dependente da posição ($\mathbf{x} = x\hat{\mathbf{i}} + y\hat{\mathbf{j}} + z\hat{\mathbf{k}}$) da pressão complexa.

A onda plana é aquela cujas variáveis acústicas possuem amplitude e fase constante ao longo de qualquer plano perpendicular à direção de propagação da onda. É interessante notar que, como as superfícies de fase constante para qualquer onda divergente se tornam quase planares quando suficientemente longe de sua fonte, é esperado que as propriedades de uma onda divergente se tornem muito semelhantes às propriedades de uma onda plana para grandes distâncias.

Considerando a equação da onda (equação 2.21), para uma onda plana harmônica viajando em uma direção arbitrária, é plausível tentar aplicar uma solução da forma

$$p = Ae^{j(\omega t - \mathbf{k}\cdot\mathbf{r})}, \tag{2.24}$$

onde $\mathbf{k}$ é o vetor de propagação, cujo módulo equivale ao número de onda, A é uma constante complexa e $\mathbf{r}$ é o vetor posição, conforme pode ser visto na Figura 3.

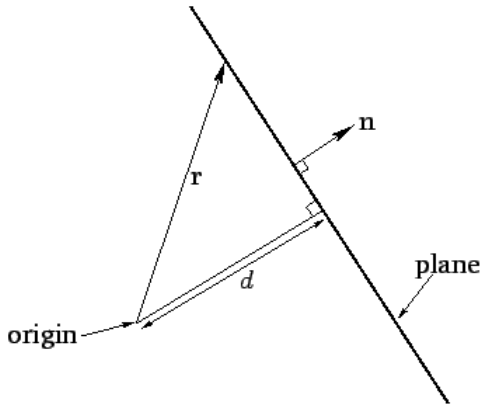


Figura 3: Plano e seus vetores associados (FITZPATRICK, 2010)

As superfícies de fase constante são dadas por $\mathbf{k} \cdot \mathbf{r} = constante$. Agora, considere que os planos de fase constante são perpendiculares ao plano do eixo xy. Além disso, considere ϕ como o ângulo entre o eixo x e o vetor $\mathbf{k}$, conforme mostrado na Figura 4.

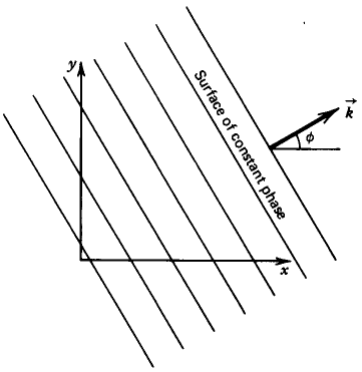


Figura 4: A onda plana (KINSLER et al., 2000)

Sendo assim, tem-se, por fim, que

$$p = Ae^{j(\omega t - kx\cos\phi - ky\sin\phi)}. \tag{2.25}$$

Aplicando a equação 2.25 para uma onda viajando na direção positiva do eixo x, ou seja, $\mathbf{k} = k\hat{\mathbf{i}}$, tem-se:

$$p = Ae^{j(\omega t - kx)}. \tag{2.26}$$

2.4 A equação de Helmholtz

Por mais que a equação da onda seja a base da teoria acústica, quando se trata de resolver problemas acústicos utilizando o MEC, o mais comum é se utilizar a equação de Helmholtz.

Considerando uma onda harmônica, a equação de Helmholtz nada mais é do que a aplicação das equações 2.22 e 2.2 na equação da onda, obtendo-se assim

$$\nabla^2 u + k^2 u = 0. \tag{2.27}$$

Na presença de uma fonte acústica, tem-se a equação de Helmholtz não homogênea:

$$\nabla^2 u + k^2 u = f. \tag{2.28}$$

Caso f seja um ponto fonte, tem-se que

$$f = Q\delta(\mathbf{x}, \mathbf{x}_Q), \tag{2.29}$$

onde $\mathbf{x}_Q$ é a localização do ponto fonte e Q é a intensidade da fonte.

A equação de Helmholtz é convencionalmente utilizada para se calcular os modos normais de um problema. Pode-se dizer que a equação de Helmholtz serve para resolver o problema acústico no domínio da frequência, enquanto a equação da onda serve para resolver o problema no domínio do tempo. Após se obter a solução com a equação de Helmholtz, é possível utilizar a equação 2.22 para se obter a solução no domínio temporal (LIU, 2009).

É interessante ressaltar que a equação de Helmholtz é uma equação do tipo elíptica. Outra conhecida equação deste tipo é a equação de Laplace:

$$\nabla^2 u = 0. \tag{2.30}$$

Sendo assim, é possível notar que a equação de Laplace nada mais é que a equação de Helmholtz com $k = 0$. Isso significa que essas duas equações resguardam muitas semelhanças entre si, e o estudo do MEC para uma equação possui muitas semelhanças com o estudo para a outra equação, sendo que, em geral, os estudos relacionados à equação de Laplace são, evidentemente, mais simples.

2.5 Cavidades fechadas

Para fins de validação dos códigos numéricos, é pertinente que soluções analíticas sejam determinadas para alguns problemas simples. Sendo assim, estudou-se algumas cavidades fechadas cuja solução analítica para a equação da onda seja conhecida. Para demonstrar tais soluções, é apresentado o conceito da matriz de transferência, proposto por Gilbert (1988). Os resultados analíticos para as cavidades fechadas foram baseados nos trabalhos de Ferreira (2004) e Morais (2000).

2.5.1 Matriz de transferência

Considere um conduto reto, definido por um comprimento L. Considere também que a vazão mássica acústica é dada por

$$q = \rho S v_f, \tag{2.31}$$

onde v_f é a velocidade da partícula fluida e S é a seção transversal do conduto. Além disso, é possível demonstrar que

$$q = \frac{jK}{\omega}\frac{dp}{dx}. \tag{2.32}$$

Aplicando a equação 2.32 na equação da onda 2.21 e resolvendo a EDP, tem-se:

$$p(x,\omega) = Asen\left(\frac{\omega}{c}x\right) + Bcos\left(\frac{\omega}{c}x\right) \tag{2.33}$$

e

$$q(x,\omega) = \frac{jS}{c}\left(Acos\left(\frac{\omega}{c}x\right) - Bsen\left(\frac{\omega}{c}x\right)\right). \tag{2.34}$$

Aplicando as equações 2.33 e 2.34 em $x = 0$ e $x = L$, obtém-se a seguinte matriz:

$$\begin{bmatrix} p_2 \\ q_2 \end{bmatrix} = \begin{bmatrix} cos(kL) & \frac{c}{jS}sen(kL) \\ -\frac{jS}{c}sen(kL) & cos(kL) \end{bmatrix} \begin{bmatrix} p_1 \\ q_1 \end{bmatrix} = A \begin{bmatrix} p_1 \\ q_1 \end{bmatrix}, \tag{2.35}$$

onde A é a matriz de transferência.

2.5.2 Cavidade uni-dimensional fechada-fechada

A cavidade fechada-fechada é modelada de forma a se impor uma vazão acústica, q, nula em ambas as extremidades do conduto. Sendo assim, aplicando $q_1 = q_2 = 0$ em 2.35, obtém-se

$$0 = -\frac{jS}{c}sen(kL)p_1 \rightarrow kL = n\pi, \tag{2.36}$$

onde $n = 1, 2, 3, ...$ Portanto, os números de onda relativos às frequências naturais desse problema são dados por

$$k_n = \frac{n\pi}{L}. \tag{2.37}$$

Utilizando-se a definição do número de onda, as frequências naturais são encontradas:

$$\omega_n = \frac{n\pi c}{L}. \tag{2.38}$$

Agora, se ao invés de aplicar a equação 2.35 na seção 1 e 2, e sim na seção 1 e em uma seção genérica x, tem-se a resposta analítica da pressão do interior do conduto, sendo esta dada por

$$\frac{p(x)}{p_1} = cos(kx). \tag{2.39}$$

2.5.3 Cavidade uni-dimensional fechada-aberta

A cavidade fechada-aberta é modelada de forma a se impor $p_2 = 0$ como o lado aberto e $q_1 = 0$ como o lado fechado. Aplicando $p_2 = q_1 = 0$ em 2.35, obtém-se, de forma análoga ao caso fechado-fechado:

$$\omega_n = \frac{c}{L}\pi\left(n - \frac{1}{2}\right), \tag{2.40}$$

onde $n = 1, 2, 3, ...$ Ou, em termos do número de onda:

$$k_n = \frac{(2n-1)\pi}{2L}. \tag{2.41}$$

Também de forma semelhante ao caso fechado-fechado, é possível obter a resposta analítica da pressão no interior do conduto:

$$\frac{p(x)}{p_1} = sen(kx). \tag{2.42}$$

2.5.4 Cavidade uni-dimensional aberta-aberta

Por fim, a cavidade aberta-aberta é modelada de forma a se impor uma pressão nula em ambas as extremidades do conduto, ou seja, $p_1 = p_2 = 0$. Aplicando em 2.35, obtém-se:

$$\omega_n = \frac{n\pi c}{L}, \tag{2.43}$$

para $n = 1, 2, 3...$ Ou, em termos do número de onda:

$$k_n = \frac{n\pi}{L}. \tag{2.44}$$

A resposta analítica da pressão para este caso é dada por

$$\frac{p(x)}{p_1} = sen(kx). \tag{2.45}$$

2.5.5 Cavidade tri-dimensional prismática fechada-fechada

Agora, considere uma geometria prismática, conforme a Figura 5.

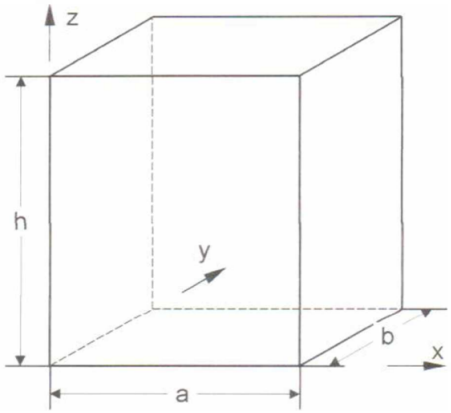


Figura 5: Geometria do prisma (MORAIS, 2000)

No caso da cavidade prismática fechada-fechada, considere que todas as paredes possuem vazão acústica nula. As frequências de ressonância para esse caso são dadas por

$$\omega_n = \pi c \sqrt{\frac{i^2}{a^2} + \frac{j^2}{b^2} + \frac{k^2}{h^2}}, \tag{2.46}$$

ou, em termos do número de onda:

$$k = \pi \sqrt{\frac{i^2}{a^2} + \frac{j^2}{b^2} + \frac{k^2}{h^2}}, \tag{2.47}$$

onde i, j, k[3] $= 1, 2, 3, ...$ são os modos nas direções x, y, z, respectivamente. Além disso, a pressão no interior da cavidade é dada por

$$\frac{p}{\rho} = \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} \sum_{k=0}^{\infty} A_{i,j,k} cos\left(\frac{i\pi x}{a}\right) cos\left(\frac{j\pi y}{b}\right) cos\left(\frac{k\pi z}{h}\right) e^{i\omega t}. \tag{2.48}$$

Considerando-se uma geometria cúbica, note que, para os modos em que $i = j = 0$, tem-se:

$$k_{0,0,n} = \frac{\pi n}{L} \tag{2.49}$$

e

$$\frac{p}{\rho} = A_{0,0,n} cos\left(\frac{n\pi z}{L}\right) e^{i\omega t} = A_{0,0,n} cos(kz) e^{i\omega t}, \tag{2.50}$$

o que é condizente com as equações 2.37 e 2.39 encontradas para o conduto uni-dimensional fechado-fechado.

[3]Uma atenção redobrada deve ser dada ao fato de que k foi utilizado tanto para caracterizar o modo na direção z quanto para o número de onda. Por mais que isso possa ser confuso, optou-se por manter essa notação de forma a evitar o uso de uma notação não usual para alguma dessas variáveis, o que poderia trazer mais problemas de entendimento do que esclarecimentos.

2.5.6 Cavidade tri-dimensional prismática aberta-fechada

A cavidade prismática aberta-fechada é obtida impondo condição de vazão acústica nula em todas as paredes, com exceção da parede em $z = h$, em que a condição é de pressão nula.

Conforme demonstrado por Morais (2000), as frequências de ressonância para esse caso são dadas por

$$\omega_n = \pi c \sqrt{\frac{i^2}{a^2} + \frac{j^2}{b^2} + \frac{k^2}{4h^2}}, \tag{2.51}$$

ou, em termos do número de onda:

$$k = \pi \sqrt{\frac{i^2}{a^2} + \frac{j^2}{b^2} + \frac{k^2}{4h^2}}, \tag{2.52}$$

onde $i, j = 1, 2, 3, ...$ e $k = 1, 3, 5, ...$ são os modos nas direções x, y, z, respectivamente. Novamente, é importante observar que, no caso dos modos para $i, j = 0$, existe concordância com os resultados obtidos para a cavidade unidimensional.

2.5.7 Cavidade tri-dimensional prismática aberta-aberta

A cavidade prismática aberta-aberta é obtida impondo condição de vazão acústica nula em todas as paredes, com exceção das paredes em $z = 0$ e $z = h$, em que a condição é de pressão nula.

Conforme demonstrado por Morais (2000), as frequências de ressonância para esse caso são dadas por

$$\omega_n = \pi c \sqrt{\frac{i^2}{a^2} + \frac{j^2}{b^2} + \frac{k^2}{h^2}}, \tag{2.53}$$

ou, em termos do número de onda:

$$k = \pi \sqrt{\frac{i^2}{a^2} + \frac{j^2}{b^2} + \frac{k^2}{h^2}}, \tag{2.54}$$

onde $i, j, k = 1, 2, 3, ...$ são os modos nas direções x, y, z, respectivamente. Novamente é importante observar que, no caso dos modos para $i, j = 0$, existe concordância com os resultados obtidos para a cavidade unidimensional.

2.6 Levitador acústico TinyLev

A levitação acústica é um fenômeno em que uma partícula solida ou líquida é suspensa no ar utilizando ondas sonoras. Isso é possível devido à presença de uma onda estacionária,

como a da Figura 2. Além da levitação acústica, também são conhecidas a levitação magnética, aerodinâmica, eletroestática, entre outros.

Utilizando-se do fenômeno descrito, pesquisadores da Universidade de Bristol desenvolveram um pequeno levitador acústico, batizado de *TinyLev*. Levitadores acústicos possuem diversas aplicações nos campos de espectroscopia, química, biologia, entre diversos outros.

O *TinyLev* é um levitador acústico uniaxial baseado em múltiplos transdutores ultra sônicos. O levitador opera a 40 kHz no ar e é capaz de suspender objetos de densidade de até 2,2 g/cm^3 e 4 mm de diâmetro. Tal levitador tem como proposta ser estável por longos períodos de tempo, resistente a mudanças de temperatura e umidade, de baixo custo, produzido com elementos facilmente encontrados no mercado e que possa ser facilmente montado, de forma que tal equipamento possa ser reproduzido em diversas Universidades e Instituições de pesquisa pelo mundo, assim como foi feito na Universidade de Brasília (FERREIRA, 2020). Tal aparato experimental, antes restrito a poucos laboratórios, pode ser facilmente reproduzido, democratizando o acesso à tecnologia (MARZO; BARNES; DRINKWATER, 2017).

A Figura 6 mostra o levitador desenvolvido na Universidade de Bristol. Note que a estrutura é feita utilizando-se impressão 3D e placas de MDF. O controle eletrônico dos transdutores é feito utilizando uma placa Arduíno.

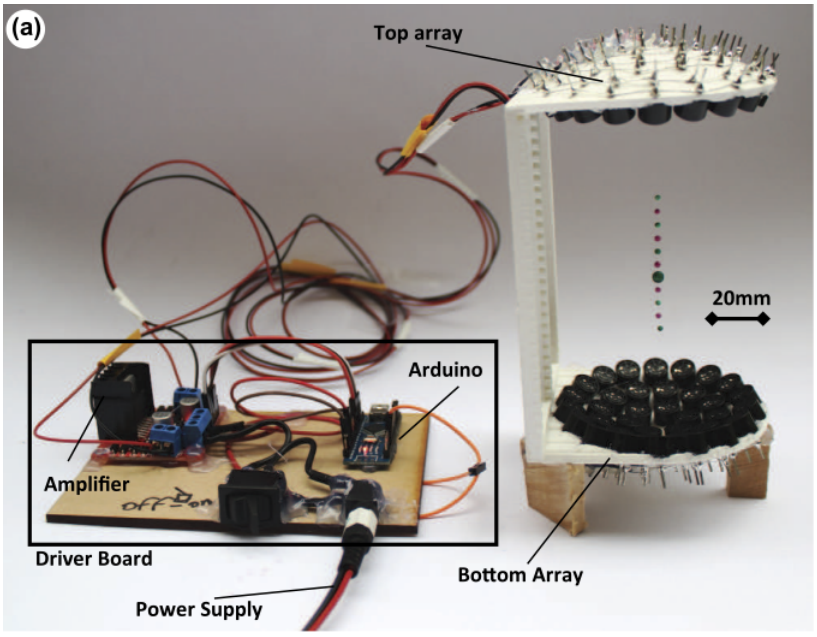


Figura 6: Levitador acústico *TinyLev* (MARZO; BARNES; DRINKWATER, 2017)

2.6.1 Modelagem analítica do levitador acústico *TinyLev*

Os transdutores ultra sônicos são modelados como pistões circulares que oscilam harmonicamente em uma única frequência. A pressão acústica complexa para um pistão circular em um ponto localizado em **r** é dada por:

$$p(\mathbf{r},\theta) = p_0 V \frac{D_f(\theta)}{d} e^{i(\phi+kd)}, \tag{2.55}$$

onde k é o número de onda, c a velocidade de propagação do som, p_0 uma amplitude que define a potência da saída do pistão circular, V é a oscilação pico-a-pico, ϕ a fase da fonte emissora e d é a distância do pistão ao ponto campo. O ângulo θ é definido na Figura 7. A função $D_f = 2J_1(ka\sin(\theta))/(ka\sin(\theta))$ representa a diretividade da fonte, em que J_1 é uma função de Bessel de primeira ordem do primeiro tipo e a é o raio do pistão. Note que o efeito adicional resultante da reflexão das ondas sonoras não é considerado. De acordo com Marzo, Barnes e Drinkwater (2017), tal efeito pode ser ignorado sem perda considerável na precisão do modelo.

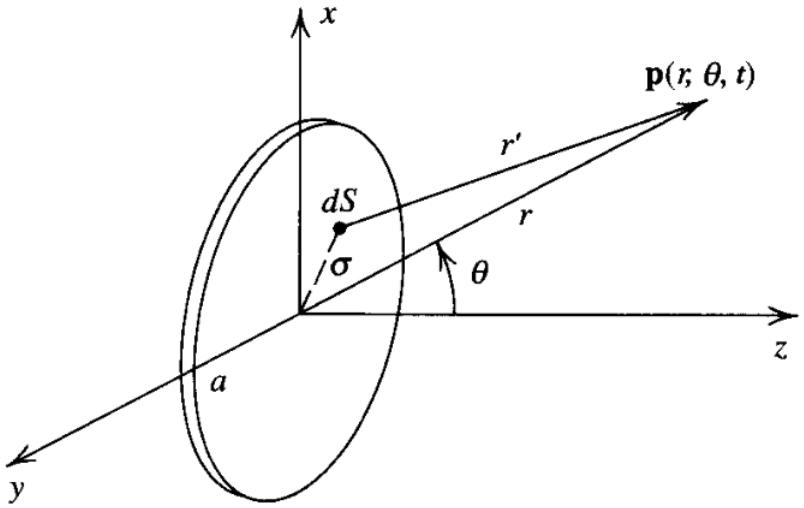


Figura 7: Modelagem de um pistão circular vibrante (KINSLER et al., 2000)

O campo acústico total é a soma da contribuição de cada um dos transdutores acústicos. A força exercida na esfera devido ao campo de pressão complexo é calculado utilizando-se o potencial de Gor'kov $F = -\nabla U$:

$$U = 2K_1|p|^2 - 2K_2(|p_x|^2 + |p_y|^2 + |p_z|^2), \tag{2.56}$$

onde

$$K_1 = \frac{1}{4}V\left(\frac{1}{c_0^2\rho_0} - \frac{1}{c_p^2\rho_p}\right) \tag{2.57}$$

e

$$K_2 = \frac{3}{4}V_p\left(\frac{\rho_0 - \rho_p}{\omega^2\rho_0(\rho_0 + 2\rho_p)}\right), \tag{2.58}$$

onde V_p é o volume da partícula, c_p a velocidade de propagação da partícula, ρ_p a massa específica da partícula, p é a pressão complexa obtida por cada transdutor e p_x é a derivada de p em x.

2.6.2 Resultados experimentais do levitador acústico *TinyLev*

Uma análise experimental do levitador acústico *TinyLev* foi conduzida no levitador construído na Universidade de Brasília, e os resultados de tal análise foram expostos no trabalho de Ferreira et al. (2021). Os resultados foram obtidos utilizando-se um aparato de Schlieren.

O aparato de Schlieren é uma ferramenta simples e direta, que permite uma análise qualitativa visual de problemas acústicos. Tal aparato é composto por um espelho esférico, uma fonte luminosa, um bloqueador de luz e uma câmera fotográfica. Os resultados da análise experimental conduzida estão na Figura 8

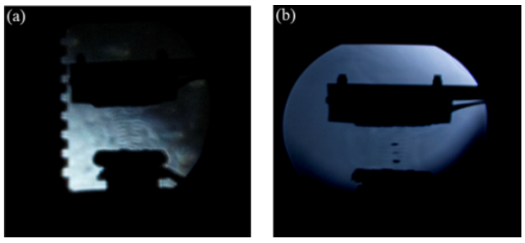


Figura 8: Análise experimental do levitador *TinyLev* utilizando um aparato de Schlieren

Capítulo 3

MÉTODO DOS ELEMENTOS DE CONTORNO

Este capítulo irá tratar do método dos elementos de contorno. Por vezes, também é possível encontrar esse método sendo referido pela sigla BEM, que nada mais é do que a tradução para a língua inglesa, *Boundary Element Method*. Como este projeto é focado em acústica, foi discutida a utilização do método para a equação de Helmholtz. Além disso, é importante ressaltar que o MEC possui diferentes formas de ser implementado e, tendo em vista que futuramente neste projeto foi utilizado o software Bempp (SMIGAJ et al., 2015), um foco maior foi dado, quando apropriado, às implementações adotadas pelo software.

3.1 MEC: vantagens e desvantagens

O MEC é um método numérico para resolver equações diferenciais parciais (EDP) formulado utilizando-se equações integrais de contorno ou, do inglês, *boundary integral equations* (BIE). Engenheiros já familiarizados com o método dos elementos finitos (MEF) podem se questionar qual a necessidade de se produzir mais uma técnica computacional. A resposta é que, mesmo sendo um método extremamente versátil, o MEF se mostrou inadequado ou ineficiente para diversas aplicações em engenharia (BREBBIA; DOMINGUEZ, 1992; ALI; RAJAKUMAR, 2005).

Mesmo com a dominância do MEF nos últimos 50 anos quando se trata de métodos numéricos, as vantagens do MEC se mostram como sendo extremamente relevantes, de forma que atualmente há um forte impeto na comunidade de engenharia no sentido de desenvolver e aperfeiçoar este método (KIRKUP, 2007). O MEC encontrou sua aplicabilidade em diversos tipos de problema, mostrando-se especialmente eficiente e promissor em problemas de domínio infinito como aqueles em acústica, eletroestática, eletromagnetismo e problemas com elevados gradientes de rigidez, como aqueles na mecânica da fratura. Em alguns casos, a combinação do MEF com o MEC para resolver um problema permite explorar as vantagens de ambos os métodos e criar uma rotina ótima (ALI; RAJAKUMAR, 2005).

Uma das maiores vantagens do MEC, se não a maior, está na malha, uma vez que a discretização do problema está confinada ao contorno apenas. Já para o MEF, é necessária uma discretização de todo o domínio. Isso torna o MEC extraordinariamente simples quando se trata de preparar a geometria para estudo, enquanto, para o MEF, os usuários são forçados a passar mais de metade do seu tempo de resolução do problema para preparar a malha de forma adequada. Em termos práticos, a formulação utilizando-se do MEC reduz uma dimensão do problema, ou seja, problemas bi-dimensionais podem ser resolvidos em uma dimensão e problemas tri-dimensionais podem ser resolvidos em duas dimensões.

Ainda sobre a comparação entre o MEF e o MEC, a vantagem do MEC quando se trata de problemas de domínio infinito reside no fato de que a apenas o contorno da superfície precisa ser discretizado, e é relativamente simples aplicar alguma condição de infinidade, como a condição de radiação de Sommerfeld. Já quando se utiliza o MEF para tais problemas, faz-se necessária a fabricação de contornos fictícios, o que, em alguns casos, reduz a precisão ou até mesmo resulta em soluções incorretas (KATSIKADELIS, 2002).

Ainda de acordo com Katsikadelis (2002), outra vantagem do MEC está na análise das derivadas da função campo. Enquanto o MEF é preciso para determinar a função campo, ele é ineficiente em determinar suas derivativas. A precisão cai consideravelmente em áreas de grandes gradientes. Já o MEC permite a avaliação da solução e de suas derivadas em qualquer ponto do problema. Isso é possível pois o método utiliza uma representação integral da solução que é uma expressão matemática contínua, que pode ser diferenciada e utilizada como uma fórmula matemática. Isso é impossível com o FEM, uma vez que a solução é obtida apenas nos pontos nodais.

Entretanto, é evidente que o MEC não possui apenas vantagens e características positivas. O MEF é conhecido por ser extremamente versátil, e é capaz de lidar com não homogeneidades, algo que é difícil quando se utiliza o MEC. Uma outra desvantagem desse método é que exige a obtenção de uma solução fundamental. Sendo assim, o MEC não é aplicável para problemas nos quais a solução fundamental seja desconhecida ou impossível de ser determinada, como por exemplo problemas cujos coeficientes da EDP sejam variáveis (KATSIKADELIS, 2002).

Outra desvantagem reside no fato de que a formulação do MEC gera matrizes cheias (ou densas) e não-simétricas. O sistema linear gerado pode ser facilmente resolvido, porém o custo computacional para montar a matriz do problema é alto (FERREIRA, 2004). Uma comparação do formato geral dos coeficientes das matrizes para o MEF e para o MEC é feita na Figura 9.

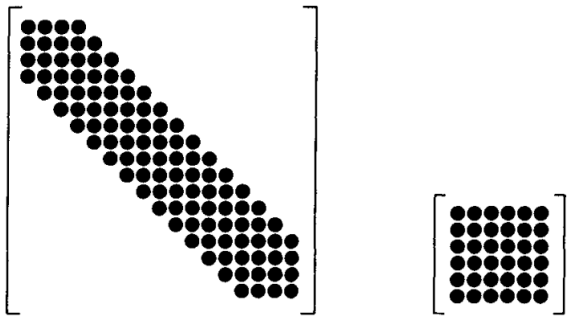


Figura 9: Formato geral dos coeficientes das matrizes para o MEF e para o MEC, respectivamente (KATSIKADELIS, 2002)

Em termos de custo computacional, uma vez que o MEC permite que o problema seja resolvido apenas no contorno, o número de variáveis desconhecidas é reduzido a $0(N^{d-1})$, enquanto métodos baseados no domínio possuem $\mathrm{O}(N^d)$ variáveis a serem determinadas, onde $\mathrm{O}(a)$ indica ordem de grandeza do numero a, N indica os graus de liberdade do sistema e d é o número de dimensões. Por outro lado, as características anteriormente citadas das matrizes envolvidas fazem com o custo do produto matriz-vetor seja maior para o MEC, $\mathrm{O}(N^4)$ para 3-D, em relação ao MEF, $\mathrm{O}(N^3)$ para 3-D. Apesar disso, as chamadas formulações rápidas (como o *Fast Multipole Method* (FMM), ou o *Adaptative Cross-Approximation* (ACA)) permitem reduzir o custo do produto matriz-vetor do MEC para $\mathrm{O}(N^2 log^{\alpha} N)$ para problemas tri-dimensionais, onde α depende da formulação rápida adotada (SMIGAJ et al., 2015).

Enquanto o FMM e o ACA servem para solucionar, ou amenizar, o problema da matriz cheia, o problema da matriz não-simétrica pode ser resolvido utilizando-se o método de Galerkin simétrico. O método de Galerkin consiste, basicamente, em utilizar as funções densidade como sendo iguais às funções de forma. É possível observar que, embora a solução seja, no geral, mais precisa do que para o método da colocação, no método de Galerkin é preciso realizar algumas integrações extras, as quais não são necessárias para a técnica da colocação (FIGUEIREDO, 2019).

3.2 O MEC à acústica

Resolver problemas de acústica é uma das aplicações mais importantes do MEC atualmente, podendo ser usado para prever campos sonoros em aplicações automotivas, aeroespaciais, entre diversos outros. Problemas acústicos geralmente são divididos em duas classes de problemas: problemas de radiação, que consistem de ondas sonoras presentes em um domínio exterior ou interior a uma superfície vibrante, e problemas de espalhamento, que consistem na interação entre um corpo e uma onda incidente que é espalhada pelo corpo (LIU, 2009).

3.2.1 Equações básicas

Primeiramente, é necessário definir 3 entidades geométricas. A primeira é o volume, V, definido com um corpo fechado. A segunda é a superfície S, onde se encontra o contorno, Γ. A terceira é o domínio acústico, Ω. Para problemas externos, Ω é infinito e exterior ao corpo V, enquanto para problemas internos é finito e interior a uma superfície fechada.

Agora, recordando-se da equação de Helmholtz não homogênea (equação 2.28):

$$\nabla^2 u + k^2 u = f, \qquad \forall \mathbf{x} \in \Omega, \tag{3.1}$$

onde $\mathbf{x}$ é o ponto fonte. As condições de contorno são divididas em dois principais tipos:

Condição de Dirichlet (pressão conhecida):

$$u = \overline{u}, \qquad \forall \mathbf{x} \in \Gamma; \tag{3.2}$$

Condição de Neumann (velocidade conhecida):

$$\frac{\partial u}{\partial n} = \overline{u}_n = j\omega\rho v_n, \qquad \forall \mathbf{x} \in \Gamma; \tag{3.3}$$

onde o vetor normal aponta para fora do domínio.

Também existem problemas em que condições de Dirichlet e Neumann são impostas simultaneamente, em diferentes partes do contorno. Considerando a condição de Dirichlet imposta em um pedaço do contorno, Γ_1, e a condição de Neumann imposta no restante do contorno, Γ_2, tem-se uma distribuição como na Figura 10.

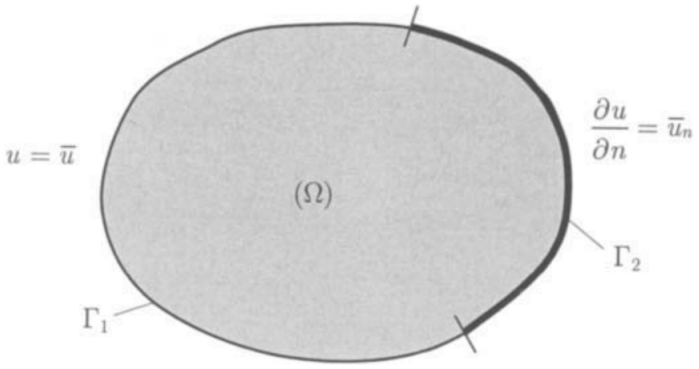


Figura 10: Domínio Ω com condições de Dirichlet e Neumann (KATSIKADELIS, 2002)

Para problemas acústicos de domínio exterior (domínio infinito), além da condição de contorno aplicada em Γ, o campo acústico deve satisfazer a condição de radiação de Sommerfeld:

$$\lim_{R\to\infty} \left[R \left| \frac{\partial u}{\partial R} - jku \right| \right] = 0, \tag{3.4}$$

onde R é o raio de uma esfera fictícia que cobre o domínio Ω. Basicamente, a condição de radiação de Sommerfeld diz que qualquer perturbação causada pelo corpo, seja esta de natureza radiada ou espalhada, deve extinguir-se no infinito baseado em considerações de energia (LIU, 2009).

3.2.2 A solução fundamental

Para se obter a solução fundamental da equação de Helmholtz, deve-se aplicar uma fonte concentrada unitária (ponto pulsante), representada pelo delta de Dirac, no ponto $\mathbf{x}$, chamado de ponto fonte. Assim, a representação matemática da resposta em um outro ponto $\mathbf{y}$, chamado de ponto campo, é chamada de solução fundamental de Green, denotada por $G(\mathbf{x}, \mathbf{y}, \omega)$. Assim, para o caso tri-dimensional, G deve satisfazer a seguinte equação:

$$\nabla^2 G + k^2 G + \delta(\mathbf{x}, \mathbf{y}) = 0, \qquad \forall \mathbf{x}, \mathbf{y} \in R^3. \tag{3.5}$$

A solução de G deve representar uma onda com direção de saída do ponto pulsante. Sendo assim, utilizaremos uma solução da forma da equação 2.5 para coordenadas esféricas, com $B = 0$ (a onda com direção de entrada em relação ao ponto pulsante é inexistente). Além disso, como se trata da equação de Helmholtz, a parte temporal da equação 2.5 deve ser desconsiderada. Assim, tem-se uma solução da forma:

$$G(\mathbf{x}, \mathbf{y}, \omega) = \frac{A}{r} e^{jkr}, \tag{3.6}$$

onde r é a distância entre o ponto fonte e o ponto campo, ou seja, $r = |\mathbf{x} - \mathbf{y}|$ e A é uma constante.

Para determinar A, integra-se 3.5 sobre um domínio esférico $\Omega_\varepsilon(\mathbf{x})$ centrado em $\mathbf{x}$, com um pequeno raio ε e contorno $\Gamma_\varepsilon(\mathbf{x})$ (note que, por se tratar do caso tri-dimensional, $\int_\Omega$ implica uma integral tripla e $\int_\Gamma$ implica uma integral dupla):

$$\int_{\Omega_\varepsilon(\mathbf{x})} \left[\nabla^2 G + k^2 G\right] d\Omega(\mathbf{y}) = -\int_{\Omega_\varepsilon(\mathbf{x})} \delta(\mathbf{x}, \mathbf{y}) d\Omega(\mathbf{y}) = -1. \tag{3.7}$$

Aplicando-se o teorema de Gauss-Green e a equação 3.6, tem-se:

$$\int_{\Omega_\varepsilon(\mathbf{x})} \nabla^2 G d\Omega(\mathbf{y}) = \int_{\Gamma_\varepsilon(\mathbf{x})} \frac{\partial G}{\partial n} d\Gamma(\mathbf{y}) = 4\pi A(jk\varepsilon - 1)e^{jk\varepsilon}. \tag{3.8}$$

De forma similar:

$$\int_{\Omega_\varepsilon(\mathbf{x})} k^2 G d\Omega(\mathbf{y}) = k^2 A \int_{\Omega_\varepsilon(\mathbf{x})} \frac{1}{r} e^{jkr} r^2 \sin\varphi d\varphi d\Theta dr = 4\pi A[(1 - jk\varepsilon)e^{jk\varepsilon} - 1]. \tag{3.9}$$

Substituindo as equações 3.8 e 3.9 na equação 3.7, obtém-se $A = 1/4\pi$. Assim, a solução

fundamental para problemas de onda acústica tri-dimensional é dada por

$$G(\mathbf{x}, \mathbf{y}, \omega) = \frac{1}{4\pi r} e^{jkr}. \tag{3.10}$$

É interessante notar que, aplicando-se $k = 0$ na solução fundamental, tem-se a solução fundamental para a equação de Laplace. Isso já era esperado uma vez que, conforme discutido anteriormente, para $k = 0$ a equação de Helmholtz se reduz à equação de Laplace.

3.2.3 Equações integrais de contorno

Para encontrar as equações integrais de contorno (BIE) correspondentes à equação de Helmholtz, primeiro aplica-se a segunda identidade de Green (A.7), tomando $u = G(\mathbf{x}, \mathbf{y}, \omega)$ e $v = u(\mathbf{x})$, obtendo-se, assim:

$$\int_\Omega [G\nabla^2 u - u\nabla^2 G] d\Omega = \int_{\Gamma \cup \Gamma_R} \left[G\frac{\partial u}{\partial n} - u\frac{\partial G}{\partial n} \right] d\Gamma, \tag{3.11}$$

em que Γ_R é o contorno de uma esfera que envolve o domínio Ω. Aplicando-se agora as equações 3.5 e 2.28, tem-se:

$$u(\mathbf{x}) = \int_{\Gamma \cup \Gamma_R} \left[G\frac{\partial u}{\partial n} - \frac{\partial G}{\partial n} u \right] d\Gamma + \int_\Omega [Gf] d\Omega \qquad \forall \mathbf{x} \in \Omega. \tag{3.12}$$

Agora, considere um problema de radiação. Além disso, considere a integral em Γ_R com $R \to \infty$. Utilizando-se de propriedades matemáticas de inequações, bem como a condição de Sommerfeld e a hipótese de que u deve desaparecer no infinito, tem-se que

$$\lim_{R\to\infty} \left| \int_{\Gamma_R} \left[G\frac{\partial u}{\partial n} - \frac{\partial G}{\partial n} u \right] d\Gamma \right| = \lim_{R\to\infty} \left[R \left| \frac{\partial u}{\partial R} - jku \right| \right] + \lim_{R\to\infty} |u| = 0. \tag{3.13}$$

Agora, considere um problema de espalhamento. Para tal classe de problema, o campo total u é dado pela soma da onda incidente, u^I, e da onda espalhada, u^S, ou seja, $u = u^I + u^S$. Da mesma forma que para um problema de radiação, a onda espalhada também deve respeitar a condição de Sommerfeld e, sendo assim, sua integral deve ser nula no infinito da mesma forma que para a onda do problema de radiação. Sendo assim, novamente para Γ_R com $R \to \infty$, tem-se:

$$\int_{\Gamma_R} \left[G\frac{\partial u}{\partial n} - \frac{\partial G}{\partial n} u \right] d\Gamma = \int_{\Gamma_R} \left[G\frac{\partial u^I}{\partial n} - \frac{\partial G}{\partial n} u^I \right] d\Gamma + \int_{\Gamma_R} \left[G\frac{\partial u^S}{\partial n} - \frac{\partial G}{\partial n} u^S \right] d\Gamma. \tag{3.14}$$

Assim, por 3.12, é possível notar que:

$$\int_{\Gamma_R} \left[G\frac{\partial u}{\partial n} - \frac{\partial G}{\partial n} u \right] d\Gamma = \int_{\Gamma_R} \left[G\frac{\partial u^I}{\partial n} - \frac{\partial G}{\partial n} u^I \right] d\Gamma = u^I. \tag{3.15}$$

Finalmente, aplicando-se os resultados obtidos para o problema de radiação (3.13) e para o problema de espalhamento (3.15), aproximando o ponto fonte $\mathbf{x}$ do contorno Γ, tem-se a equação integral de contorno convencional (CBIE) para problemas acústicos:

$$c(\mathbf{x})u(\mathbf{x}) = \int_\Gamma \left[G\frac{\partial u(\mathbf{y})}{\partial n} - \frac{\partial G}{\partial n}u(\mathbf{y}) \right] d\Gamma(\mathbf{y}) + u^I(\mathbf{x}) + \int_\Omega f d\Omega. \tag{3.16}$$

De modo a generalizar a equação 3.16, faz-se as seguintes considerações para c($\mathbf{x}$):

$$c(\mathbf{x}) = \begin{cases} 1, & \text{se } \mathbf{x} \in \Omega; \\ \frac{\theta_{int}}{2\pi}, & \text{se } \mathbf{x} \in \Gamma; \\ 0, & \text{se } \mathbf{x} \notin \Omega, \Gamma. \end{cases} \tag{3.17}$$

Quando o ponto fonte se encontrar em um ponto suave do domínio, ou seja, não em um canto (ALBUQUERQUE, 2020), tem-se:

$$c = \frac{\theta_{int}}{2\pi} = \frac{1}{2}. \tag{3.18}$$

Sendo assim, deve-se utilizar a equação 3.16 em Γ para se determinar os valores no contorno. Com os valores no contorno conhecidos, é possível utilizar 3.16 com um ponto fonte em Ω para se determinar u em qualquer ponto do domínio.

Para se determinar o fluxo, basta aplicar a derivada normal na equação 3.16, obtendo-se a equação integral de contorno hipersingular (HBIE) para problemas acústicos:

$$c(\mathbf{x})\frac{\partial u(\mathbf{x})}{\partial n} = \int_\Gamma \left[\frac{\partial G}{\partial n(\mathbf{x})}\frac{\partial u(\mathbf{y})}{\partial n} - \frac{\partial^2 G}{\partial n(\mathbf{x})\partial n(\mathbf{y})}u(\mathbf{y}) \right] d\Gamma(\mathbf{y}) + \frac{\partial u^I(\mathbf{x})}{\partial n} + \int_\Omega \frac{\partial f}{\partial n(\mathbf{x})} d\Omega. \tag{3.19}$$

3.2.4 Formulação de Burton-Miller

É fato conhecido que essa CBIE possui uma grande falha para a análise de problemas de domínio exteriores, que está na não unicidade da solução para um conjunto de autofrequências associadas às frequências de ressonância correspondentes ao problema interno (URSELL, 1973). Sendo assim, é fato bem conhecido que, para problemas de Dirichlet, $k^2 = \lambda$ corresponde a um autovalor do problema interior, enquanto para problemas de Neumann, $k^2 = \mu$ corresponde a um autovalor do problema interior, conforme as equações (ENGLEDER; STEINBACH, 2008):

$$-\nabla^2 u_\lambda(\mathbf{x}) = \lambda u_\lambda(\mathbf{x}) \quad \mathbf{x} \in \Omega, \quad u_\lambda(\mathbf{x}) = 0 \quad \mathbf{x} \in \Gamma \tag{3.20}$$

e

$$-\nabla^2 u_\mu(\mathbf{x}) = \mu u_\mu(\mathbf{x}) \quad \mathbf{x} \in \Omega, \quad \frac{\partial}{\partial n_x} u_\mu(\mathbf{x}) = 0 \quad \mathbf{x} \in \Gamma. \tag{3.21}$$

Para se obter uma solução única para todas as frequências, algumas possíveis soluções são utilizar a formulação de Brakhage–Werner (BRAKHAGE; WERNER, 1965) ou utilizar as equações de Helmholtz estabilizadas (ou modificadas) (ENGLEDER; STEINBACH, 2008). Contudo, possivelmente a solução mais utilizada, e a que foi adotada neste projeto, é a formulação de Burton-Miller (BURTON; MILLER, 1971), por vezes também referida por dual BIE (LIU, 2009), a qual consiste em utilizar uma combinação da CBIE com a HBIE, da forma:

$$CBIE + \eta HBIE = 0, \tag{3.22}$$

em que η é a constante de acoplamento. O valor mais comum para tal constante é $\eta = \pm j/k$. O sinal da constante é, por muitas vezes, escolhido de forma incorreta, podendo resultar soluções imprecisas ou até mesmo erradas (MARBURG, 2016). Como o sinal adotado neste projeto para a solução fundamental é positivo, isto é, $\alpha\beta = 1$, a escolha correta do sinal da constante de acoplamento é $\eta = j/k$.

Para ilustrar esse problema e a forma como a formulação de Burton-Miller funciona, considere o exemplo da esfera pulsante. Trata-se de um caso simples de radiação, cuja solução analítica é conhecida. O uso apenas da BIE convencional (CBIE) resulta em soluções incorretas nas autofrequências, que são, para esse caso, dadas por $ka = n\pi$, onde a é o raio da esfera e $n = 1, 2, 3, ...$ O fenômeno descrito pode ser visto na Figura 11.

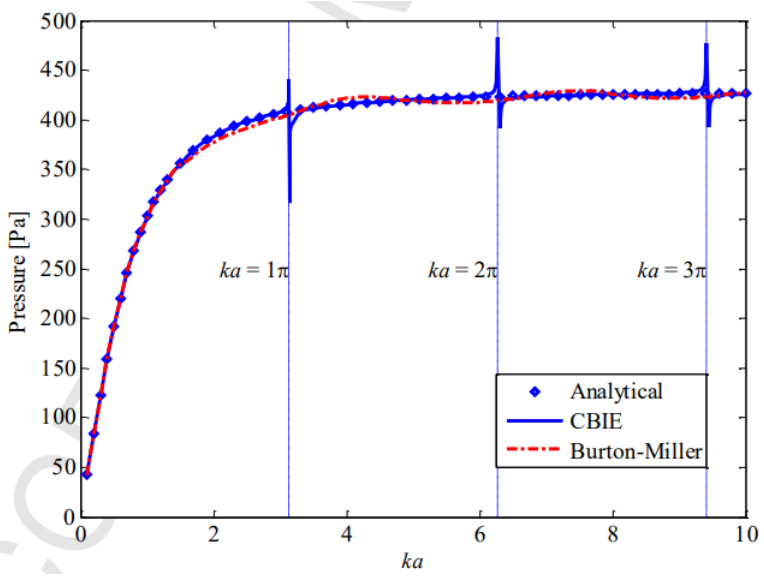


Figura 11: Comparação da solução para o caso da esfera pulsante utilizando a formulação convencional e Burton-miller (WU; YE; JIANG, 2017)

3.3 O problema da singularidade das equações de contorno

Voltando a atenção, por um momento, para a solução fundamental (equação 3.5), é possível notar que, quando o ponto fonte coincide com o ponto campo, há um problema de singularidade. A equação fundamental G possui uma singularidade fraca, enquanto sua derivada

normal possui uma singularidade forte (LIU, 2009).

O tratamento dessas integrais demanda um certo esforço computacional, e essa é uma das desvantagens do MEC. Uma possível solução para esse problema é resolver analiticamente as integrais de contorno quando estas forem singulares. Contudo, essa solução só é viável quando se utiliza elementos de baixa ordem, como os chamados elementos constantes.

Quando se utiliza elementos de ordem maior, como elementos quadráticos, é necessário tratar as integrais de forma que elas se tornem, no máximo, fracamente singulares, podendo assim serem mais facilmente resolvidas. Para tornar uma integral fortemente singular em uma integral fracamente singular, uma alternativa é introduzir o kernel estático da solução fundamental, conforme (LIU, 2009). Os detalhes de tal abordagem fogem do escorpo deste projeto.

Já as singularidades fracas podem ser tratadas mais facilmente, seja por transformação de variáveis cúbicas (TELLES, 1987) ou pela própria quadratura de Gauss-Green.

3.4 Tratamento da integral de domínio

Note que, nas equações referentes à CBIE (3.16) e à HBIE (3.19) tem-se, ainda, uma integral de domínio. Essa integral de domínio aparece quando há a presença de um termo não homogêneo não nulo na EDP. No caso do MEC, a presença dessa integral de domínio, se não tratada, dificulta a aplicação do método e, além de tudo, estraga o conceito básico desse método que é utilizar apenas integrais de contorno.

Sendo assim, precisa-se lidar com essa integral de domínio. Uma das soluções é simplesmente discretizar o domínio e resolver, assim, a integral. A malha do domínio não possui necessidade de ser bem refinada para esse caso. Mesmo assim, essa solução quase não é utilizada, por ir contra o próprio conceito fundamental do MEC.

Já que a solução proposta não é aceitável, o caminho a ser tomado é transformar a integral de domínio em uma de contorno. Caso a não homogeneidade se trate de uma fonte concentrada, dada por $Q\delta(\mathbf{x}, \mathbf{x}_Q)$, onde Q é a intensidade da fonte e $\mathbf{x}_Q$ o ponto onde esta se encontra, a solução do problema é relativamente simples. A integral de domínio aplicada ao delta de Dirac, por definição, resulta na solução fundamental. Assim, tem-se

$$\int_\Omega f d\Omega = QG(\mathbf{x}, \mathbf{y}, \omega). \tag{3.23}$$

Caso a função f não seja uma fonte concentrada, é possível tratá-la utilizando-se o teorema de Green (A.7). Entretanto, para tal, é, em geral, necessário que $\nabla^2 f$ seja conhecido (KATSIKADELIS, 2002).

3.5 Discretização das equações integrais de contorno

Para que se possa resolver numericamente as BIE, é necessário discretizar o contorno em um número finito de segmentos, chamados de elementos de contorno. Os elementos de contorno mais frequentemente utilizados são elementos constantes, elemento linear e o elemento quadrático (ou parabólico). Em cada elemento, distinguem-se os pontos de extremidade e os nós. Os nós podem ou não coincidir com os pontos de extremidade, como mostra a Figura 12.

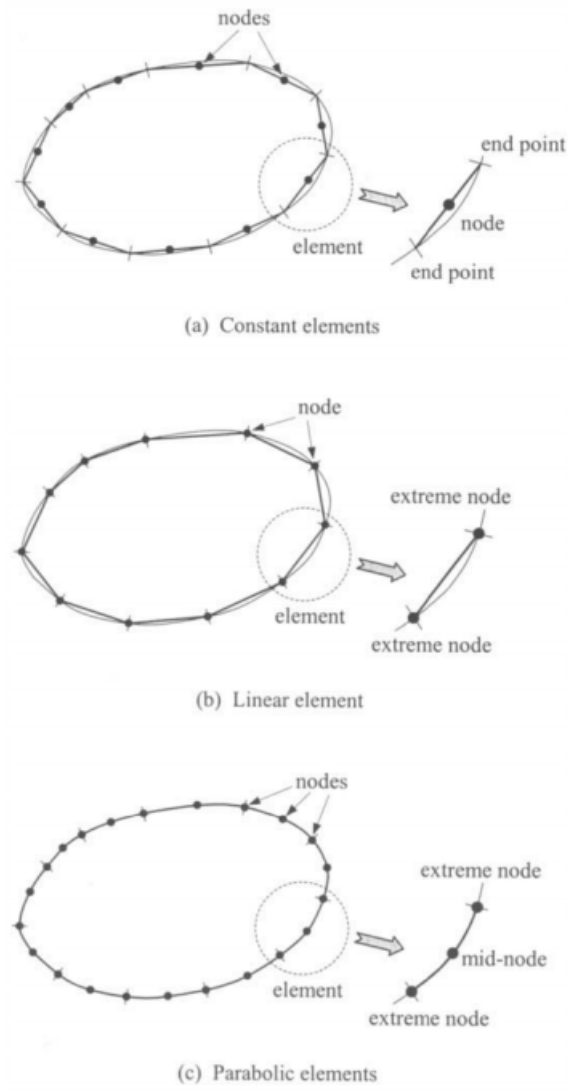


Figura 12: Nós e pontos extremos para alguns elementos típicos (KATSIKADELIS, 2002)

A implementação de elementos constantes é relativamente simples e direta, além de permitir integração analítica quando desejável. Contudo, o uso de elementos lineares ou quadráticos é mais preciso e eficiente. O uso de elementos quadráticos, em especial, é bastante vantajoso em termos de precisão, mesmo com o uso de um número menor de elementos, devido à sua acurácia e flexibilidade em modelar superfícies curvas (LIU, 2009).

No caso tri-dimensional, além de classificar os elementos como constantes, lineares ou quadráticos, deve-se considerá-los como sendo triangulares ou quadrilaterais. Um resumo dos

tipos de elementos para o caso tri-dimensional pode ser visto na Figura 13.

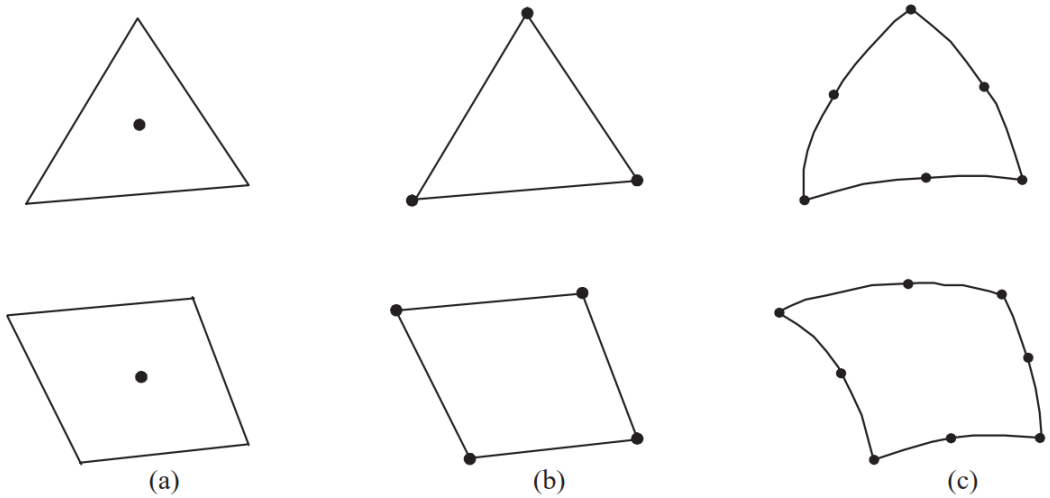


Figura 13: Elementos 3-D: (a)constante, (b)linear, (c)quadrático (LIU, 2009)

Cada tipo de elemento possui suas funções de forma (ou funções de interpolação). Tais funções de forma nada mais são do que a transformada das posições dos nós dos elementos das coordenadas cartesianas (x, y) para as coordenadas naturais (ξ, η). Como um exemplo genérico, as funções de forma para um elemento quadrilateral linear são:

$$\begin{aligned} N_1(\xi,\eta) &= \frac{1}{4}(1-\xi)(1-\eta); \\ N_2(\xi,\eta) &= \frac{1}{4}(1+\xi)(1-\eta); \\ N_3(\xi,\eta) &= \frac{1}{4}(1+\xi)(1+\eta); \\ N_4(\xi,\eta) &= \frac{1}{4}(1-\xi)(1+\eta). \end{aligned} \tag{3.24}$$

3.6 A formulação indireta do método dos elementos de contorno

Uma outra formulação do MEC é utilizar a solução fundamental para construir diretamente as equações integrais de contorno. As BIE construídas dessa forma possuem uma função densidade, σ, que não possui significado físico direto. Portanto, formulações desse tipo são chamadas de formulações indiretas. No caso em que se utiliza o método de Galerkin, as funções densidade utilizadas são as próprias funções de forma.

3.6.1 Operadores de contorno

Os operadores de contorno são as integrais de contorno que são utilizadas para se resolver o problema no contorno. Tais operadores são:

- Operador *single-layer*

$$[V_k\phi](\mathbf{x}) = \int_\Gamma G_k(\mathbf{x},\mathbf{y})\sigma(\mathbf{y})d\Gamma(\mathbf{y}) \qquad \mathbf{x}\in\Gamma \tag{3.25}$$

- Operador *double-layer*

$$[K_k\phi](\mathbf{x}) = \int_\Gamma \frac{\partial G_k(\mathbf{x},\mathbf{y})}{\partial n(\mathbf{y})}\sigma(\mathbf{y})d\Gamma(\mathbf{y}), \qquad \mathbf{x}\in\Gamma; \tag{3.26}$$

- Operador *adjoint double-layer*

$$[K'_k\phi](\mathbf{x}) = \int_\Gamma \frac{\partial G_k(\mathbf{x},\mathbf{y})}{\partial n(\mathbf{x})}\sigma(\mathbf{y})d\Gamma(\mathbf{y}), \qquad \mathbf{x}\in\Gamma; \tag{3.27}$$

- Operador *hypersingular*

$$[W_k\phi](\mathbf{x}) = -\int_\Gamma \frac{\partial^2 G_k(\mathbf{x},\mathbf{y})}{\partial n(\mathbf{x})\partial n(\mathbf{y})}\sigma(\mathbf{y})d\Gamma(\mathbf{y}), \qquad \mathbf{x}\in\Gamma. \tag{3.28}$$

3.6.2 Operadores Potenciais

Depois que o problema for resolvido no contorno Γ, e todas as variáveis forem conhecidas em tal, utiliza-se os operadores potenciais para determinar o valor do potencial e do fluxo em qualquer ponto do domínio Ω. Os operadores potenciais são:

- Potencial *single-layer*

$$[\nu_k\phi](\mathbf{x}) = \int_\Gamma G_k(\mathbf{x},\mathbf{y})\sigma(\mathbf{y})d\Gamma(\mathbf{y}), \qquad \mathbf{x}\in\Gamma; \tag{3.29}$$

- Potencial *double-layer*

$$[\kappa_k\phi](\mathbf{x}) = \int_\Gamma \frac{\partial G_k(\mathbf{x},\mathbf{y})}{\partial n(\mathbf{y})}\sigma(\mathbf{y})d\Gamma(\mathbf{y}), \qquad \mathbf{x}\in\Gamma. \tag{3.30}$$

3.6.3 A projeção de Calderón

Utilizando-se a CBIE (3.16) e os operadores potenciais, tem-se:

$$u_\Omega = \nu_k u_n - \kappa_k u_\Gamma + u^I. \tag{3.31}$$

A partir da equação 3.31, tomando o traço de ambos os lados, seguido da derivada na direção normal, obtém-se a projeção de Calderón:

$$\begin{bmatrix} u \\ \frac{\partial u}{\partial n} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}I - K & V \\ W & \frac{1}{2}I + K' \end{bmatrix} \begin{bmatrix} u \\ \frac{\partial u}{\partial n} \end{bmatrix}. \tag{3.32}$$

Da projeção de Calderón tem-se, para problemas de Dirichlet:

$$V\frac{\partial u}{\partial n} = (\frac{1}{2}I + K)\overline{u}. \tag{3.33}$$

E, para problemas de Neumann:

$$Wu = (\frac{1}{2}I - K')\overline{u}_n. \tag{3.34}$$

Para o caso em que se tenha uma combinação de condições de contorno de Dirichlet e Neumann deve-se utilizar (SMIGAJ et al., 2015):

$$V\frac{\partial u}{\partial n} - Ku = (\frac{1}{2}I + K)\overline{u} - V\overline{u}_n, \qquad \mathbf{x} \in \Gamma_1; \tag{3.35}$$

$$Wu + K'\frac{\partial u}{\partial n} = (\frac{1}{2}I - K')\overline{u}_n - W\overline{u}, \qquad \mathbf{x} \in \Gamma_2. \tag{3.36}$$

As definições das variáveis relativo às condições de contorno estão de acordo com a Figura 10, em que em Γ_1 são aplicadas as condições de Dirichlet e em Γ_2 são aplicadas as condições de Neumann.

3.6.4 Problemas exteriores

Considere, agora, que o domínio Ω pode ser interpretado tanto como interno, denotando-se para tal Ω_-, como externo, denotando-se Ω_+. O domínio externo não é limitado por nenhuma superfície, sendo assim, por definição, infinito. Para este domínio, conforme demonstrado por Amini e Kirkup (1995), a aplicação da segunda identidade de Green (A.7), resulta na seguinte equação integral:

$$u_\Omega = -\nu_k u_n + \kappa_k u_\Gamma. \tag{3.37}$$

De forma semelhante ao problema interior, a projeção de Calderón para o problema exterior resulta, para problemas de Dirichlet:

$$V\frac{\partial u}{\partial n} = (-\frac{1}{2}I + K)\overline{u}. \tag{3.38}$$

E, para problemas de Neumann:

$$-Wu = (\frac{1}{2}I + K')\overline{u}_n. \tag{3.39}$$

Para evitar o problema da não unicidade da solução para as autofrequências, é possível utilizar a formulação de Burton-Miller (BURTON; MILLER, 1971). No caso da formulação indireta, considerando-se problemas de espalhamento, tem-se as seguintes equações integrais de

contorno (LUKASHIN; STRIJHAK; SHCHEGLOV, 2017):

- Problema de Dirichlet utilizando formulação de Burton-Miller:

$$(\frac{1}{2}I + K' - \eta V)\frac{\partial u}{\partial n} = \frac{\partial u^I}{\partial n} - \eta u^I; \tag{3.40}$$

- Problema de Neumann utilizando formulação de Burton-Miller:

$$(\frac{1}{2}I - K + \eta W)u = u^I + \eta\frac{\partial u^I}{\partial n}. \tag{3.41}$$

Com as variáveis determinadas no contorno, para determinar a solução no domínio, tem-se as seguintes equações:

- Avaliação no domínio para problema de Dirichlet utilizando Burton-Miller:

$$u_\Omega = u^I - \nu_k u_n; \tag{3.42}$$

- Avaliação no domínio para problema de Neumann utilizando Burton-Miller:

$$u_\Omega = u^I + \kappa_k u_\Gamma. \tag{3.43}$$

3.7 Fluxo de trabalho para a resolução de um problema usando o método dos elementos de contorno

Para se resolver um problema utilizando o método de elementos de contorno, é necessário estabelecer uma sequência lógica que envolvem, em termos gerais, um momento de preparação, um de resolução e outro de análise. Tal sequência lógica pode ser chamada de fluxograma de trabalho. Para ambos os softwares que foram utilizados, é recomendável adotar um fluxograma de trabalho que possua repetibilidade e transparência.

Primeiramente, é necessário ter, bem estabelecido, qual é o problema a ser resolvido. Deve-se conhecer as equações relevantes envolvidas e como utilizá-las no software para a resolver o problema. No caso de problemas acústicos, será resolvida a equação de Helmholtz. Caso o MEC seja utilizado, é importante escolher, entre outros fatores, o tipo de elemento a ser utilizado, a implementação do MEC adotada e se serão utilizados métodos de aceleração.

A geometria pode ser criada utilizado o software de CAD FreeCad e exportada para o software de geração de malha, o GMSH. Para tal, é necessário utilizar um formato de arquivo intercambiável, como BREP (BUZOGANY, 2017). Geometrias mais simples podem também serem geradas diretamente no GMSH. Durante a geração da malha no GMSH é recomendável determinar as superfícies físicas, que irão conter as informações de contorno. Tal malha deve ser exportada para o programa de elementos de contorno, as condições de contorno devem ser definidas nas superfícies físicas adequadas e o problema pode ser resolvido. Após a resolução do

problema é necessário analisar a resposta obtida. Uma análise gráfica é possível utilizando-se o software ParaView ou mesmo o GMSH (FERREIRA, 2020).

3.7.1 O BEM_base

O BB (ou BEM_base) é uma implementação em código aberto do MEC por colocação, que permite a aceleração por matrizes hierárquicas e o uso de processamento paralelo, desenvolvido pelo grupo de elementos de contorno da UnB (UnBEM). O programa é capaz de resolver as equações de Laplace e de Helmholtz. Para permitir uma metodologia de trabalho facilmente replicável, que utilize apenas programas gratuitos e livremente disponíveis, o BB permite a integração com o Gmsh, FreeCad e ParaView.

A Universidade de Brasília e o grupo UnBEM desenvolve implementações do MEC para problemas diversos a mais de 20 anos. Neste tempo foram produzidos uma considerável quantidade de software para problemas específicos, especialmente utilizando as linguagens computacionais Fortran, GNU Octave, Matlab, Python e Julia. O BB é, em grande parte, uma união desses programas em um só software, escrito em Julia. Julia é uma linguagem de programação razoavelmente recente, desenvolvida de 2012 em diante no MIT, de alta performance, tão rápida quanto compiladores dedicados de linguagens de programação de baixo nível como C ou Fortran (FERREIRA, 2020).

O programa é construído a partir de 3 programas de MEC, chamados de módulos, cada um presente em uma pasta. Todos os módulos podem ser carregados rodando o código "BEM _base.jl". A fonte (ou *source*) do programa está na pasta "src", onde se encontram as implementações para resolver as equações integrais de contorno para elementos lineares 2D constantes, elementos NURBS 2D e elementos triangulares bilineares 3D constantes. O programa permite a utilização do método de aceleração por matrizes hierárquicas, com ACA+ e aproximação por polinômios de Langrange e nós de Chebyshev. Já a pasta "tests"contêm arquivos em Julia para testar as equações de Helmholtz e Laplace. Para isso, são utilizados testes simples cujas soluções analíticas são conhecidas. Além disso, a pasta "notebooks"possui a solução de alguns problemas utilizando-se o Jupyter Notebook (FERREIRA, 2020).

3.7.2 O Bempp

O Bempp é uma plataforma de código aberto para resolver problemas utilizando-se o MEC, em especial problemas de condução de calor, eletroestática, eletromagnetismo e acústica. O software foi desenvolvido no Colégio Universitário de Londres (*University College London*) e na Universidade de Cambridge (*the University of Cambridge*). Atualmente o time de desenvolvimento é composto por Time Betcke e Matthew Scroggs. O programa também permite integração com o método de elementos finitos, utilizando a interface para o programa FEniCS.

Inicialmente, o programa, chamado de BEM++, era uma biblioteca em C++ com interface em Python. Contudo, a nova versão, Bempp-cl é inteiramente escrita em Python. Uma vez que o Python é uma linguagem interpretada, este costuma ser mais lento do que linguagens

compiladas, como é o caso do Julia. Por esse motivo, o Bempp utiliza compiladores *Just in Time* (JIT) para tornar a linguagem mais veloz. O *PyOpenCL* é utilizado por padrão, permitindo paralelização do código utilizando a CPU ou GPU. Alternativamente, também é possível utilizar o *numba*.

O Bempp é dividido em duas partes: bempp.api e bempp.core. As funcionalidades *user-friendly* da biblioteca estão contidas em bempp.api, enquanto as rotinas de computação rápida estão em bempp.core.

O software utiliza o que, neste projeto, foi descrito como formulação indireta do MEC. Além disso, a discretização dos potenciais é feita pelo método de Galerkin simétrico. Também é possível utilizar as formulações rápidas ACA e FMM, por meio de interface com rotinas externas.

A biblioteca do BEM é composto de cinco partes principais, ilustradas na Figura 14. Uma explicação bem simplificada de cada umas das partes é a seguinte:

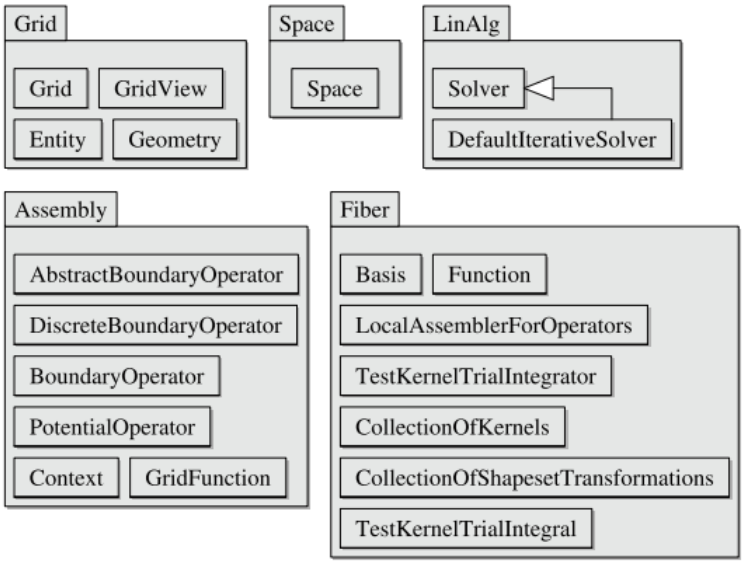


Figura 14: Módulos do Bempp (SMIGAJ et al., 2015)

- *Grid*: responsável por lidar com a malha. Permite importar malha do *Gmsh* ou utilizar malhas já prontas para geometrias simples.
- *Fiber*: é onde estão as rotinas rápidas de integração de elementos de contorno.
- *Space*: é o módulo onde se cria as funções espaço dos elementos, que podem ser elementos constantes ou lineares, por exemplo.
- *Assembly*: é a maior parte da biblioteca. Responsável pela formulação das matrizes discretizadas. Diretamente ligado ao módulo *Fiber*.
- *LinAlg*: interface com uma diversidade de *solvers*, como por exemplo os *solvers GMRES* ou *CG*.

Capítulo 4

RESULTADOS NUMÉRICOS

Neste capítulo são apresentados os resultados das simulações numéricas. Inicialmente alguns problemas simples foram explorados, com a finalidade de validação dos programas de código aberto que foram estudados. Após os problemas simples, o problema do levitador acústico foi explorado com maior profundidade.

4.1 Problemas bi-dimensionais em um quadrado unitário

Inicialmente, alguns problemas bastante simples foram explorados. A utilidade de testar tais problemas é, principalmente, verificar se os resultados obtidos pelas simulações numéricas estão coerentes com aquilo que é esperado. Um benefício adicional de testar problemas simples, quando se trata de um usuário novo a um programa, é o ganho de familiaridade com o código e as formas de implementação.

Os problemas simples inicialmente estudados são todos bi-dimensionais. O domínio é um quadrado unitário, com segmentos indexados de 1 a 4, conforme a Figura 15. Foi utilizado o programa BB, desenvolvido pelo grupo UnBEM na Universidade de Brasília. Para o MEC, foram utilizados elementos constantes, pelo método convencional (sem nenhum tipo de acelerador, como matrizes hierárquicas). Não foi possível, para estes problemas, realizar uma comparação com resultados numéricos utilizando o Bempp, uma vez que o Bempp apenas resolve problemas para malhas bi-dimensionais com elementos triangulares, ou seja, problemas com geometrias tri-dimensionais.

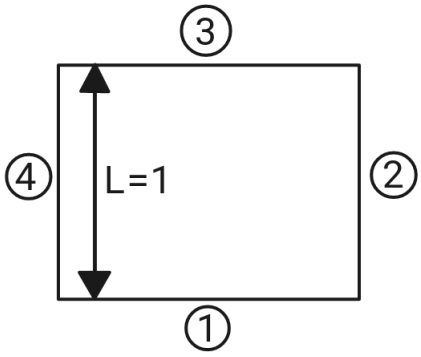


Figura 15: Quadrado unitário

4.1.1 Caso (i): Quadrado fechado-fechado

O caso (i) é o quadrado fechado-fechado. Sendo assim, as seguintes condições de contorno foram aplicadas no programa:

- Segmento 1: $\overline{u}_n = 0$;
- Segmento 2: $\overline{u}_n = 0$;
- Segmento 3: $\overline{u}_n = 0$;
- Segmento 4: $\overline{u}_n = 1$;

onde $\overline{u}_n$ é a derivada normal de u, ou condição de vazão acústica. Para tais condições de contorno, conforme 2.37 têm-se os seguintes números de onda associados às frequências naturais:

$$k_n = \frac{n\pi}{L}, \qquad n = 1, 2, ... \tag{4.1}$$

Para plotar a curva de superfície do potencial de campo, u, Figura 17, cada lado do quadrado foi subdividido em 50 segmentos de tamanhos iguais. A malha uni-domensional resultante pode ser vista na Figura 16. A resposta foi avaliada em 10000 pontos internos distribuídos de forma simétrica. Além das condições de contorno expostas, a simulação foi realizada utilizando-se o quarto sobretom, ou seja, a frequência natural para $n = 5$.

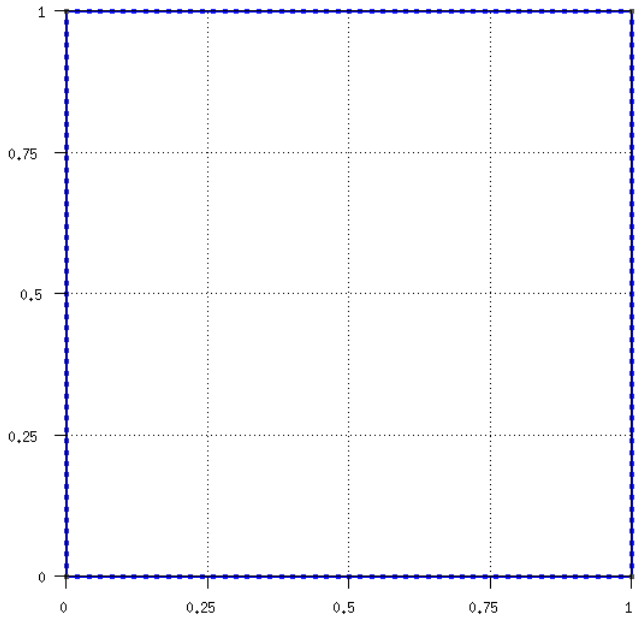


Figura 16: Malha no contorno do quadrado unitário

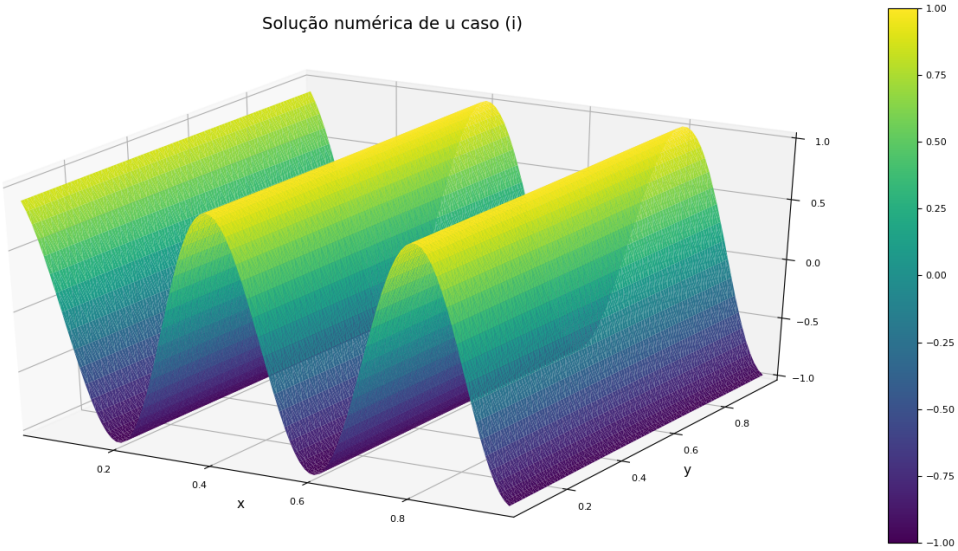


Figura 17: Curva de superfície obtida numericamente para o caso (i)

Para este caso, foi feita uma comparação qualitativa com a resposta analítica do problema. A curva de superfície para a solução analítica pode ser vista na Figura 18. Como esperado, pela avaliação qualitativa percebe-se uma boa concordância entre os resultados numéricos e analíticos.

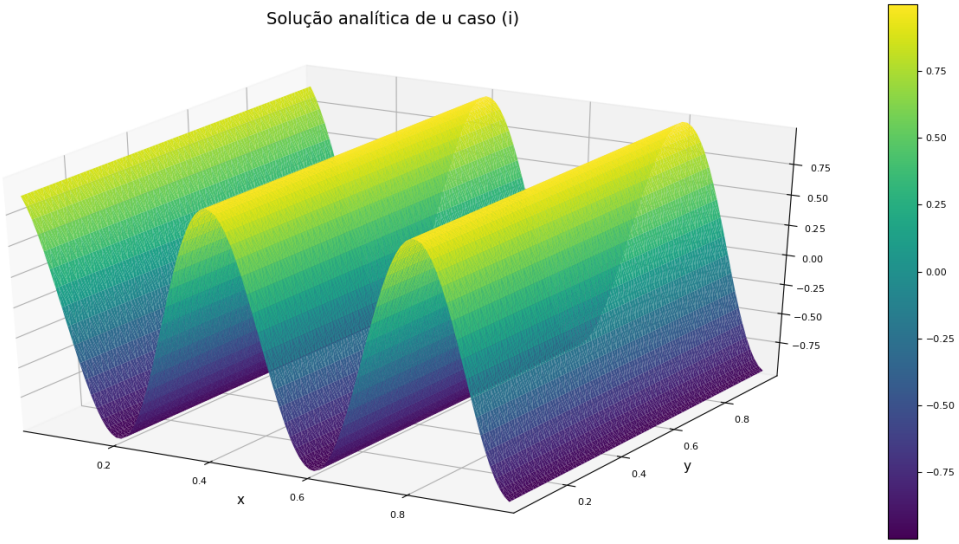


Figura 18: Curva de superfície obtida analiticamente para o caso (i)

Por fim, também realizou-se um gráfico de contorno. Como o gráfico de contorno requer uma resolução maior, cado lado o quadrado foi subdividido em 200 segmentos retilíneos de tamanhos idênticos. O resultado pode ser visto na Figura 19.

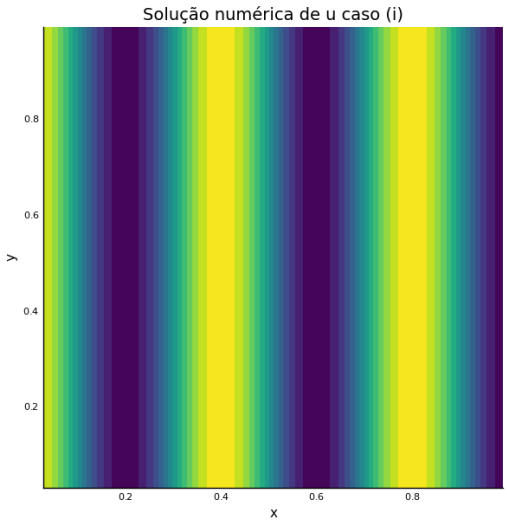


Figura 19: Curva de contorno obtida numericamente para o caso (i)

4.1.2 Caso (ii): Quadrado fechado-aberto

O caso (ii) é o quadrado fechado-aberto. Sendo assim, as seguintes condições de contorno foram aplicadas no programa:

- Segmento 1: $\overline{u}_n = 0$;
- Segmento 2: $\overline{u}_n = 0$;

- Segmento 3: $\overline{u}_n = 0$;
- Segmento 4: $\overline{u} = 1$;

Para tais condições de contorno, conforme a equação 2.41 têm-se os seguintes números de onda associados às frequências naturais:

$$k_n = \frac{(2n-1)\pi}{2L}, \qquad n = 1, 2, ... \tag{4.2}$$

As curvas de superfície, Figura 20, e de contorno, Figura 21, foram plotadas utilizando os mesmo parâmetros do caso (i), exceto que a frequência utilizada foi o quarto sobretom para o caso (ii), ou seja, $n = 5$ aplicado à equação 4.2.

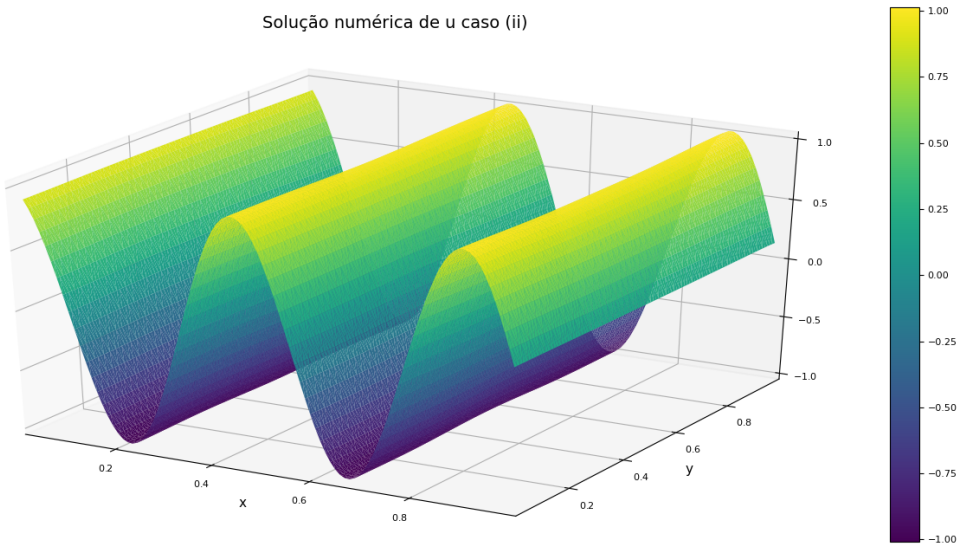


Figura 20: Curva de superfície obtida numericamente para o caso (ii)

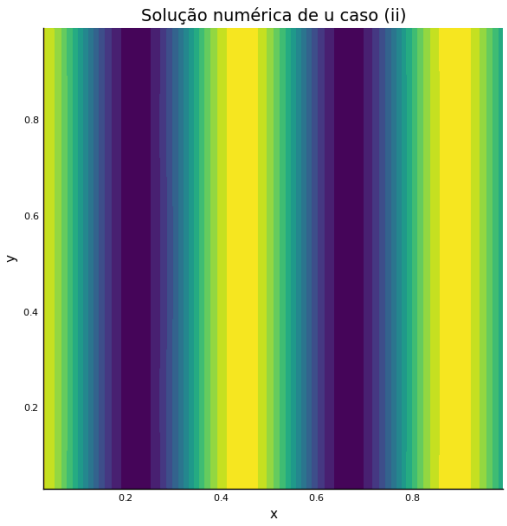


Figura 21: Curva de contorno obtida numericamente para o caso (ii)

4.1.3 Caso (iii): Quadrado aberto-aberto

O caso (iii) é o quadrado aberto-aberto. Sendo assim, as seguintes condições de contorno foram aplicadas no programa:

- Segmento 1: $\overline{u}_n = 0$;
- Segmento 2: $\overline{u} = 0$;
- Segmento 3: $\overline{u}_n = 0$;
- Segmento 4: $\overline{u} = 1$;

Para tal caso, conforme a equação 2.44 têm-se os seguintes números de onda associados às frequências naturais:

$$k_n = \frac{n\pi}{L}, \qquad n = 1, 2, ... \tag{4.3}$$

As curvas de superfície, Figura 22, e de contorno, Figura 23, foram plotadas utilizando os mesmo parâmetros do caso (i) e (ii), exceto que a frequência utilizada foi o primeiro sobretom para o caso (iii), ou seja, $n = 2$ aplicado à equação 4.3.

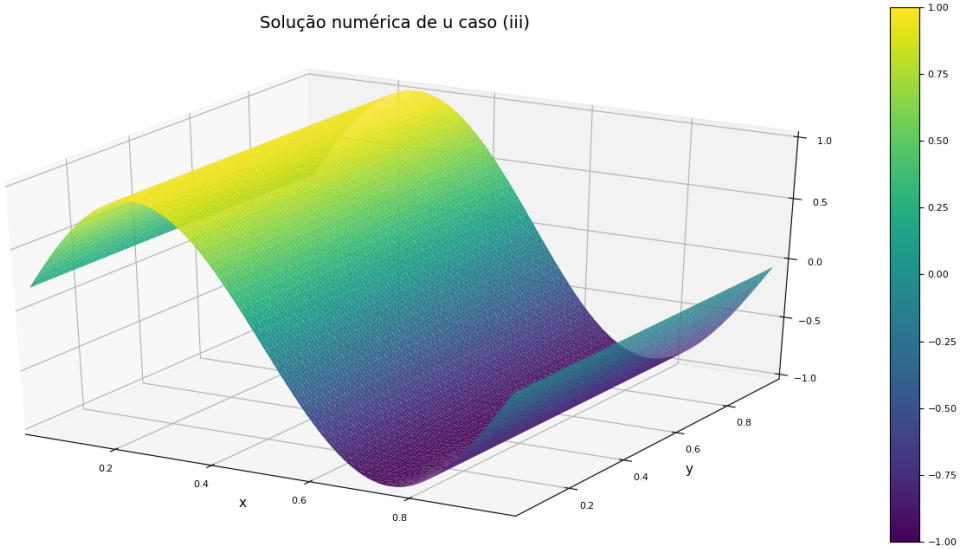


Figura 22: Curva de superfície obtida numericamente para o caso (iii)

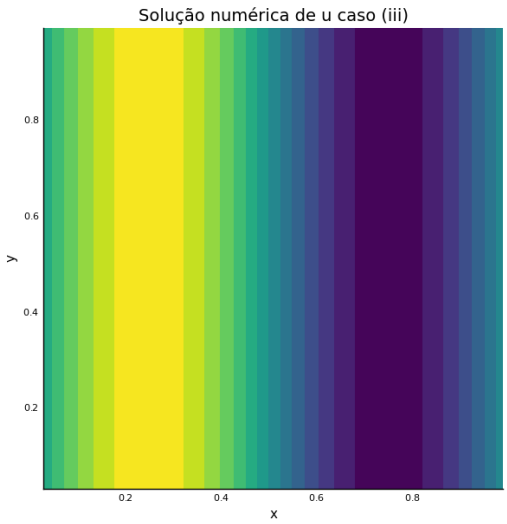


Figura 23: Curva de contorno obtida numericamente para o caso (iii)

4.2 Problemas tri-dimensionais em um cubo

Após a resolução bem sucedida de problemas bi-dimensionais, problemas semelhantes serão resolvidos, mas agora em uma geometria prismática tri-dimensional. Além da comparação dos resultados numéricos obtidos com o BB com a resposta analítica, também foi feita uma comparação com a resposta numérica obtida utilizando-se o programa Bempp.

A geometria estudada foi um cubo de lado igual a 10, com segmentos indexados de 1 a 6. A superfície do cubo foi discretizada em elementos triangulares de comprimento igual a 1, utilizando o programa Gmsh[4]. A malha resultante possui 1476 elementos e 740 nós, conforme pode ser visto na Figura 24.

[4]Disponível em: <https://gmsh.info/>

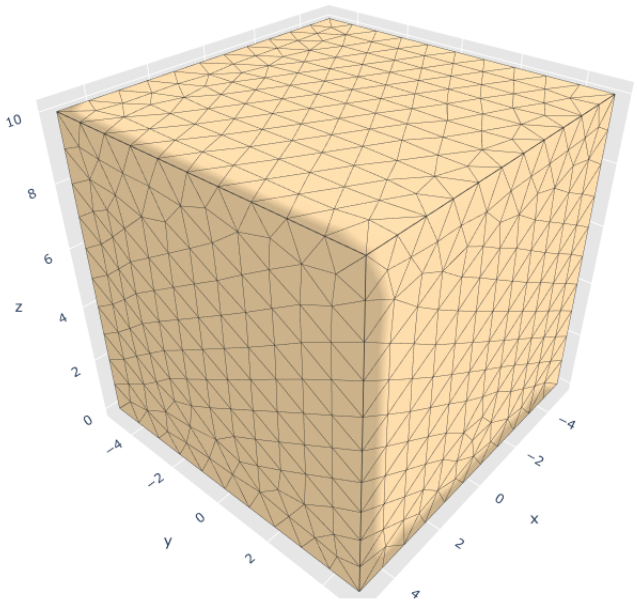


Figura 24: Malha do cubo

O MEC utilizado pelo BB, para a resolução destes problemas, foi o método convencional com elementos constantes. Para as integrações, utilizou-se o método de Gauss-Legendre com 12 pontos de integração.

Já para o Bempp foram utilizados elementos lineares contínuos para as faces onde foram aplicadas condições de contorno de Neumann e elementos constantes para as faces onde foram aplicadas condições de Dirichlet. Por hora, não foram utilizados métodos de aceleração, como o FMM. A solução foi obtida utilizando-se do Método do Resíduo Mínimo Generalizado (GMRES).

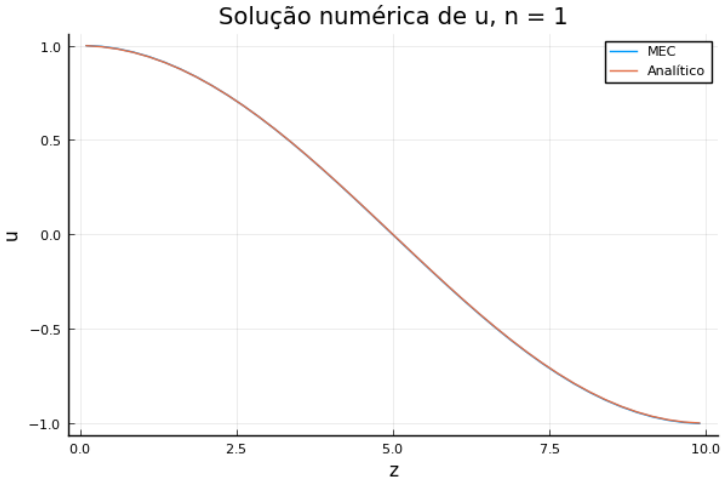
Para o cálculo da solução no interior do cubo, foi utilizada uma reta discretizada em 40 pontos. Tal reta percorre o eixo "z"do cubo e está está localizada no centro da face cuja normal está na direção do eixo "z".

4.2.1 Caso (i): Cubo fechado-fechado

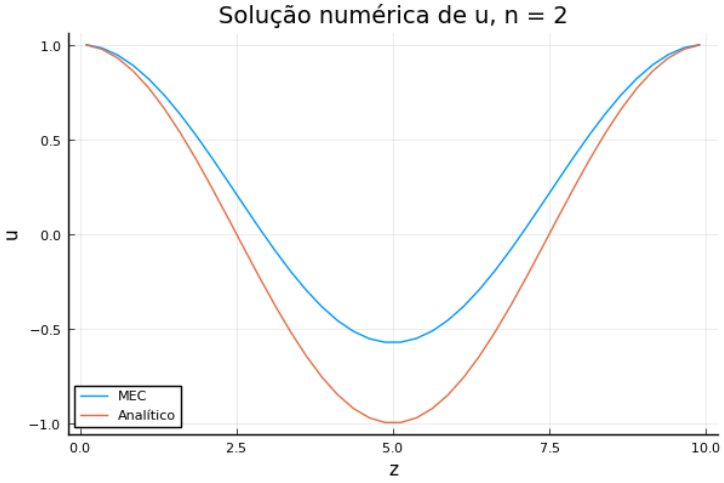
De forma análoga aos problemas simples bi-dimensionais, o caso (i) é o do cubo fechado-fechado. As seguintes condições de contorno foram aplicas:

- Face 1: $\overline{u}_n = 0$;
- Face 2: $\overline{u}_n = 0$;
- Face 3: $\overline{u}_n = 1$;
- Face 4: $\overline{u}_n = 0$;
- Face 5: $\overline{u}_n = 0$;
- Face 6: $\overline{u}_n = 0$;

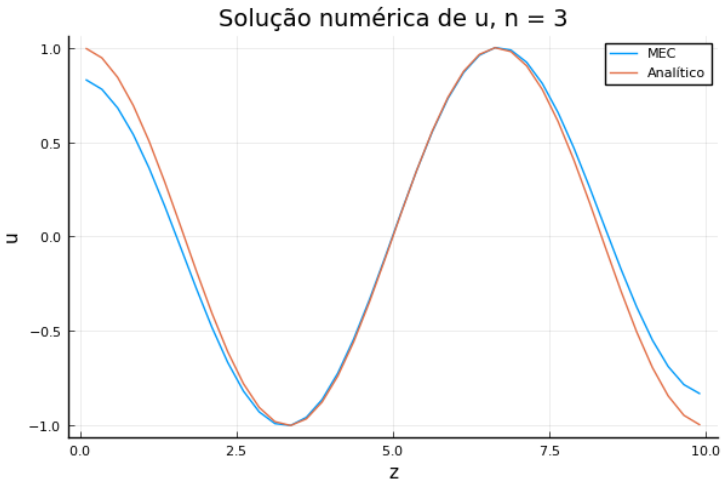
onde $\overline{u}_n$ é a derivada normal de u, ou condição de vazão acústica. Para tal caso, os números de onda associados às frequências naturais são dados pela equação 2.47. Foram plotados os gráficos para os 3 primeiros modos de ressonância, nos pontos internos descritos anteriormente, utilizando-se o programa BB. Além disso, foi realizada uma comparação com a solução analítica, como pode ser visto na Figura 25.



(a) Primeiro modo



(b) Segundo modo



(c) Terceiro modo

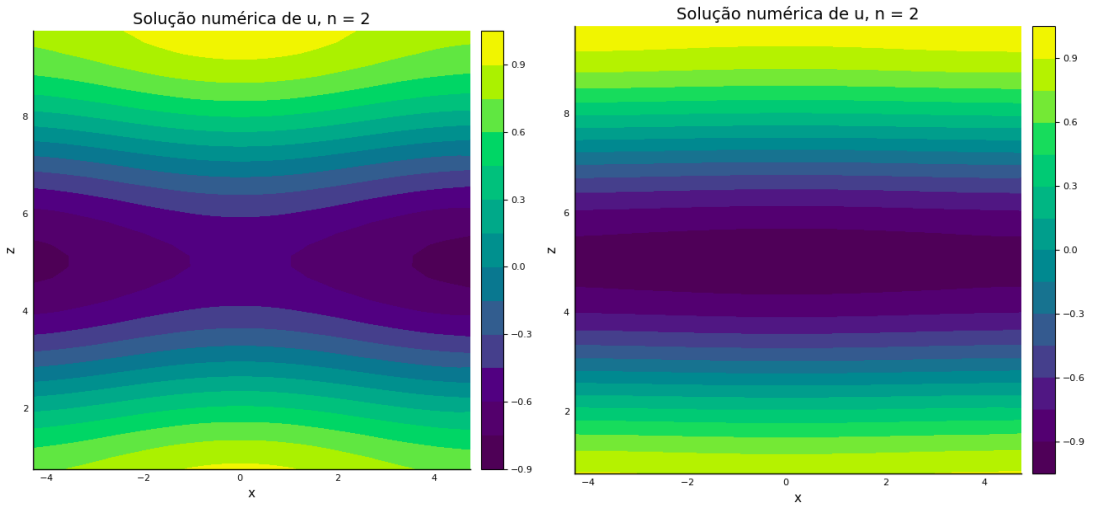
Figura 25: Resposta para os 3 primeiros modos do problema fechado-fechado

É possível observar que, para os modos 2 e 3, o resultado numérico divergiu consideravelmente do resultado analítico. Após uma investigação detalhada, verificou-se que o problema era resultado da simetria do cubo. O mesmo problema já havia sido anteriormente identificado por Ferreira (2012). Sendo assim, para resolver esse problema, basta criar uma geometria que se aproxime bastante de um cubo, com uma leve assimetria. Foi criada uma nova geometria prismática, muito semelhante à geometria cúbica inicial. As dimensões de tal geometria se encontram na tabela 1.

Tabela 1: Geometria prismática assimétrica

a	b	h
10,1	9,9	10,0

O motivo da divergência dos resultados pode ser observado utilizando-se um gráfico de calor. Tal gráfico foi plotado em um plano que corta o centro da geometria, paralelo ao plano "yz", cuja normal está na direção do eixo "x". É possível notar que, para a geometria simétrica, ocorre um abaulamento da solução, o que causa o erro numérico. A introdução de uma assimetria diminui o nível de abaulamento, sendo que quanto maior a assimetria, menor o problema do abaulamento. É curioso notar que esse problema só se apresenta para determinados modos. A Figura 26 mostra o gráfico de calor obtido inicialmente e o gráfico obtido com a geometria assimétrica.



(a) Cubo (b) Prisma assimétrico

Figura 26: Gráfico de calor da resposta para u

É possível notar que ainda existe um pequeno abaulamento da resposta, mesmo para o prisma assimétrico. Contudo, o resultado obtido já se tornou satisfatoriamente próximo à resposta analítica. Da mesma forma que foi feita no BB, implementou-se a geometria assimétrica no Bempp, e obteve-se a resposta para os três primeiros modos. A comparação entre a solução numérica obtida por ambos os programas e a solução analítica para o segundo modo pode ser vista na Figura 27.

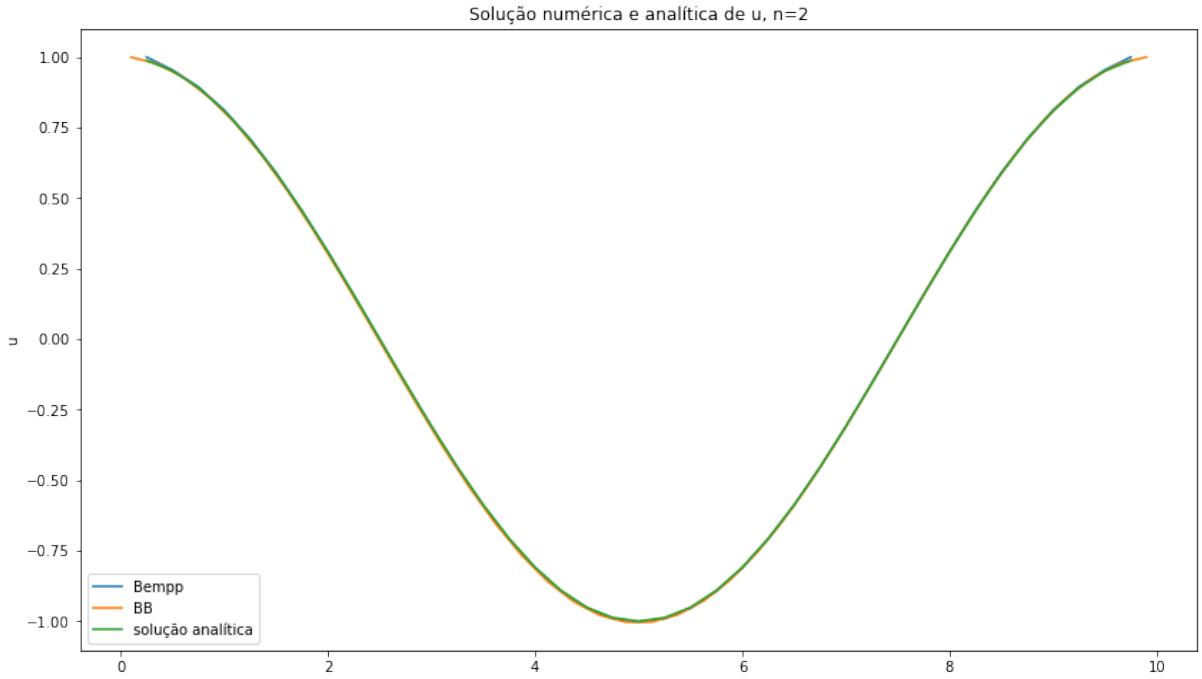


Figura 27: Comparação resposta analítica e numérica utilizando o BB e o Bempp

O erro entre a resposta numérica e a analítica foi avaliado por meio da raiz do valor quadrático médio (RMS) do erro normalizado, conforme a equação A.16. Os erros para cada

modo, entre a resposta numérica e a analítica, podem ser vistos na Tabela 2.

Tabela 2: Erro das soluções numéricas obtidas pelo BB e pelo Bempp

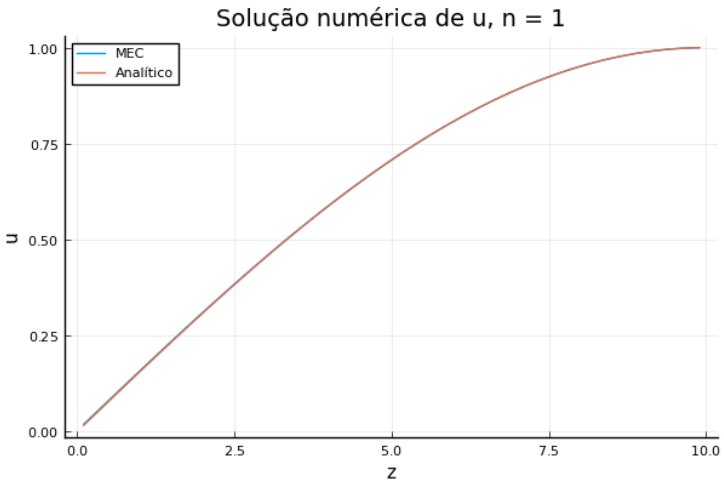
	n=1	n=2	n=3
BB	0,0005017	0,0024834	0,0102628
Bempp	0,0017108	0,0018608	0,0018774

4.2.2 Caso (ii): Cubo aberto-fechado

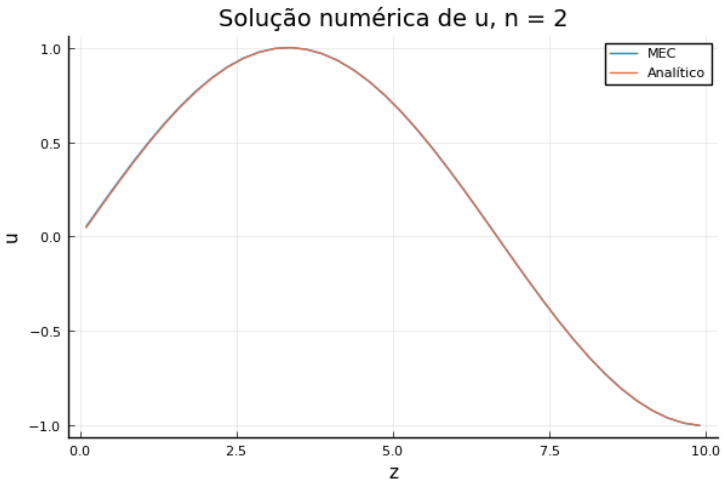
De forma análoga aos problemas simples bi-dimensionais, o caso (ii) é o do cubo aberto-fechado. As seguintes condições de contorno foram aplicas:

- Face 1: $\overline{u}_n = 0$;
- Face 2: $\overline{u}_n = 0$;
- Face 3: $\overline{u} = 1$;
- Face 4: $\overline{u}_n = 0$;
- Face 5: $\overline{u}_n = 0$;
- Face 6: $\overline{u}_n = 0$;

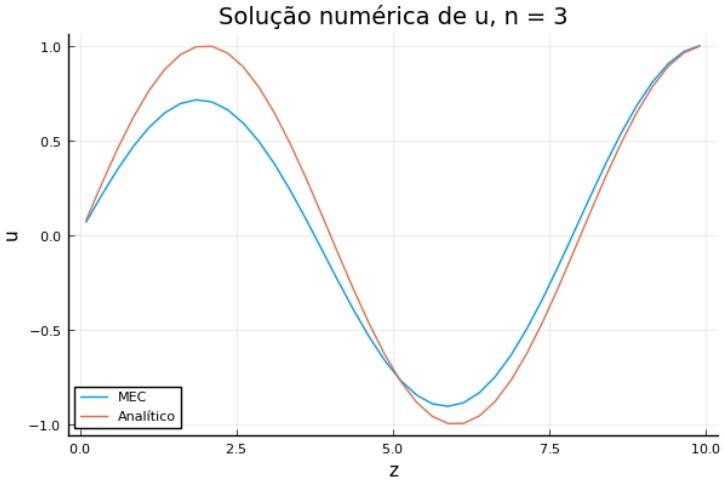
onde $\overline{u}_n$ é a derivada normal de u, ou condição de vazão acústica. Para tal caso, os números de onda associados às frequências naturais são dados pela equação 2.52. Foram plotados os gráficos para os 3 primeiros modos de ressonância, nos pontos internos descritos anteriormente, utilizando-se o programa BB. Além disso, foi realizada uma comparação com a solução analítica, como pode ser visto na Figura 28.



(a) Primeiro modo



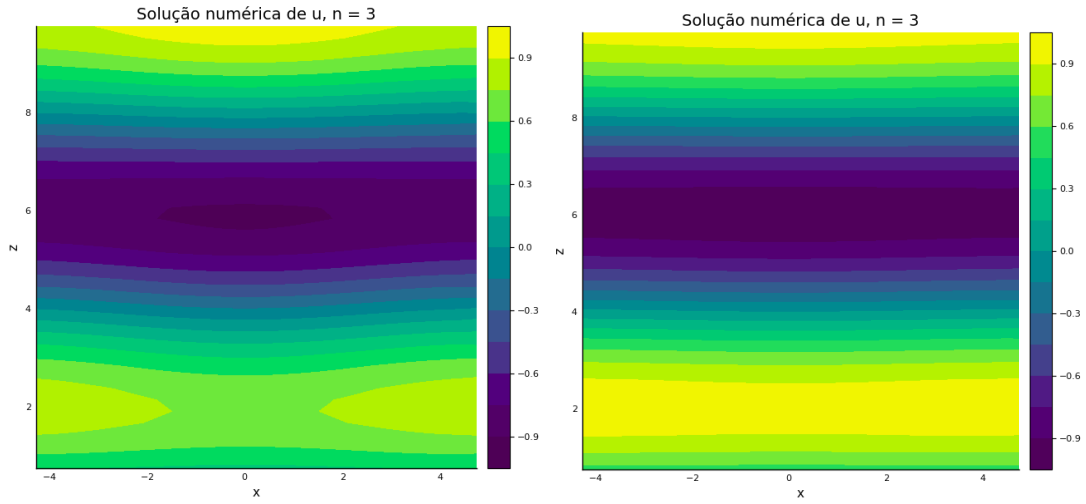
(b) Segundo modo



(c) Terceiro modo

Figura 28: Resposta para os 3 primeiros modos do problema aberto-fechado

Nota-se que, assim como ocorreu com o caso (i), um dos modos apresentou uma resposta consideravelmente divergente da analítica. O motivo de tal divergência é o mesmo do que o do caso (i). Sendo assim, a mesma geometria prismática foi utilizada para resolver o caso (ii). O gráfico de calor, mostrando o abaulamento da resposta antes e depois do uso da geometria assimétrica, pode ser visto em 29.



(a) Cubo (b) Prisma assimétrico

Figura 29: Gráfico de calor da resposta para u

É possível observar uma notável redução no abaulamento da resposta. As respostas para os modos utilizando a geometria assimétrica foram calculadas utilizando tanto o BB quanto o BB. A resposta para o terceiro modo pode ser vista na Figura 30.

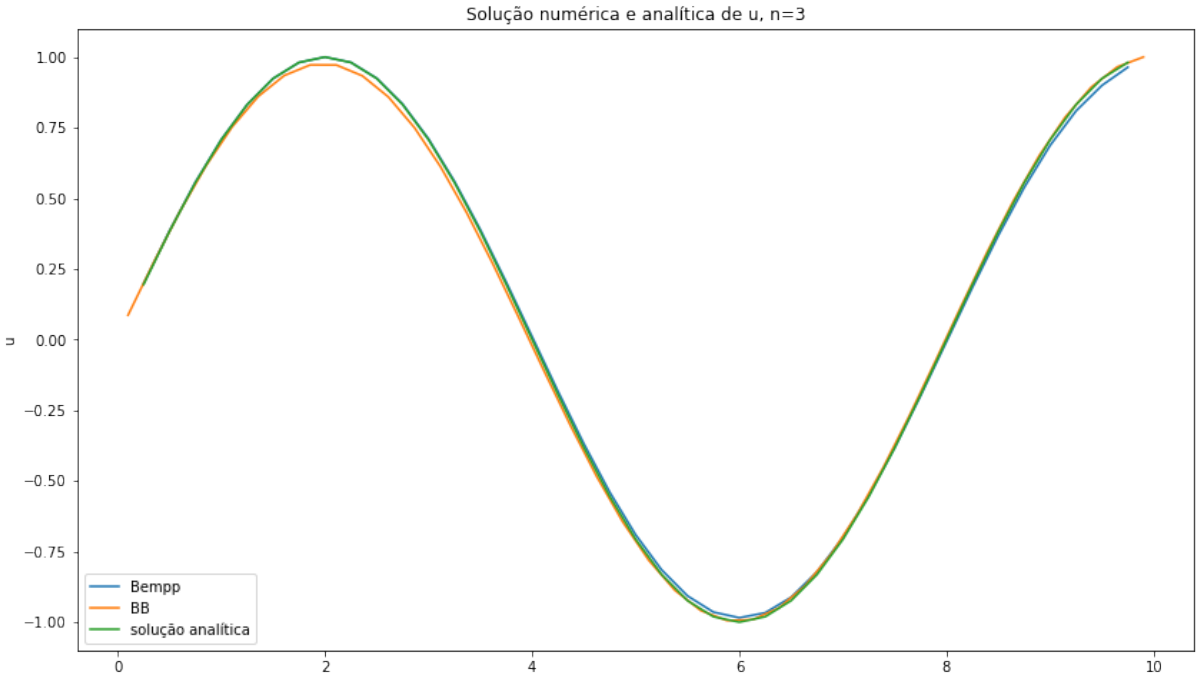


Figura 30: Comparação resposta analítica e numérica utilizando o BB e o Bempp

O erro entre a resposta numérica e a analítica foi avaliado por meio da raiz do valor quadrático médio (RMS) do erro normalizado. Os erros para cada modo, entre a resposta numérica e a analítica, podem ser vistos na Tabela 3.

Tabela 3: Erro das soluções numéricas obtidas pelo BB e pelo Bempp

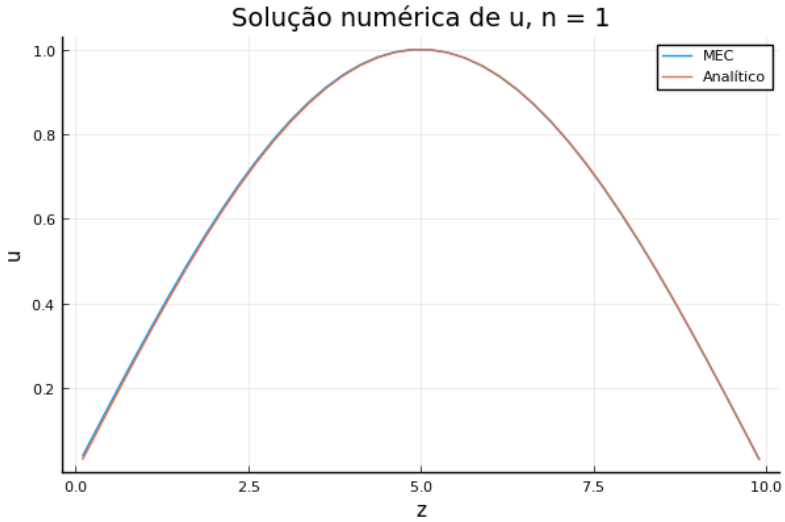
	n=1	n=2	n=3
BB	0,0010254	0,0012278	0,0074104
Bempp	0,0052056	0,0007647	0,0057125

4.2.3 Caso (iii): Cubo aberto-aberto

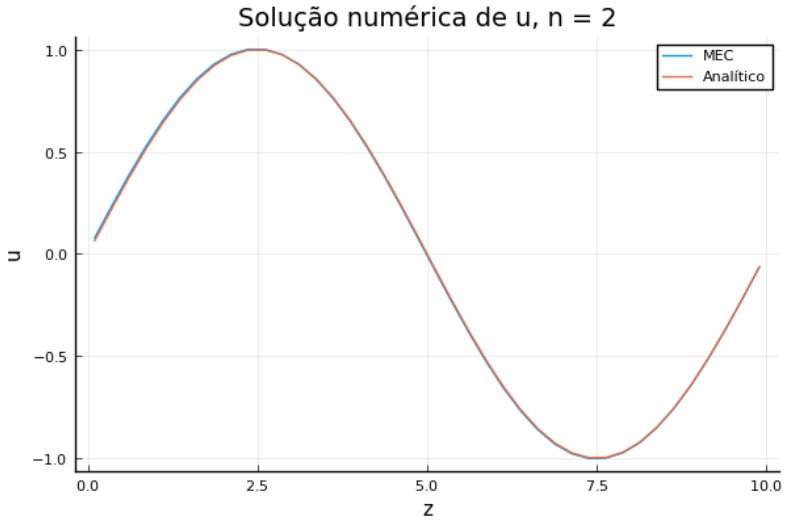
De forma análoga aos problemas simples bi-dimensionais, o caso (iii) é o do cubo aberto-aberto. As seguintes condições de contorno foram aplicas:

- Face 1: $\overline{u} = 0$;
- Face 2: $\overline{u}_n = 0$;
- Face 3: $\overline{u} = 1$;
- Face 4: $\overline{u}_n = 0$;
- Face 5: $\overline{u}_n = 0$;
- Face 6: $\overline{u}_n = 0$;

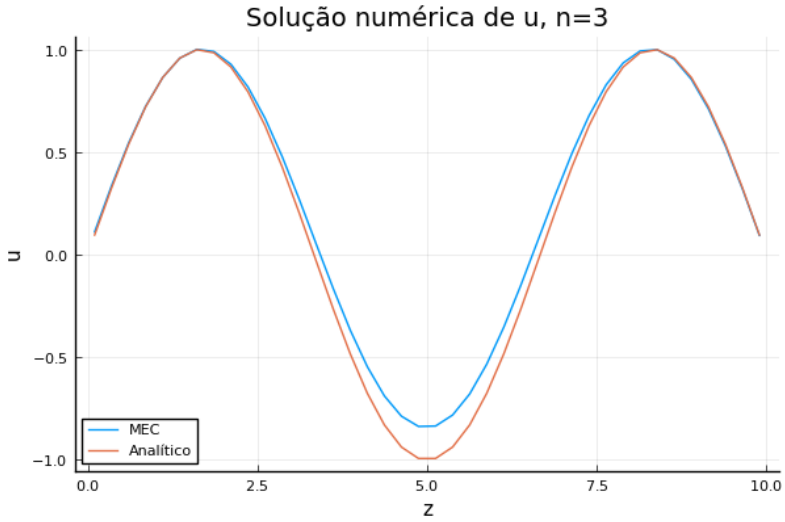
onde $\overline{u}_n$ é a derivada normal de u, ou condição de vazão acústica. Para tal caso, os números de onda associados às frequências naturais são dados pela equação 2.54. Foram plotados os gráficos para os 3 primeiros modos de ressonância, nos pontos internos descritos anteriormente, utilizando-se o programa BB. Além disso, foi realizada uma comparação com a solução analítica, como pode ser visto na Figura 31.



(a) Primeiro modo



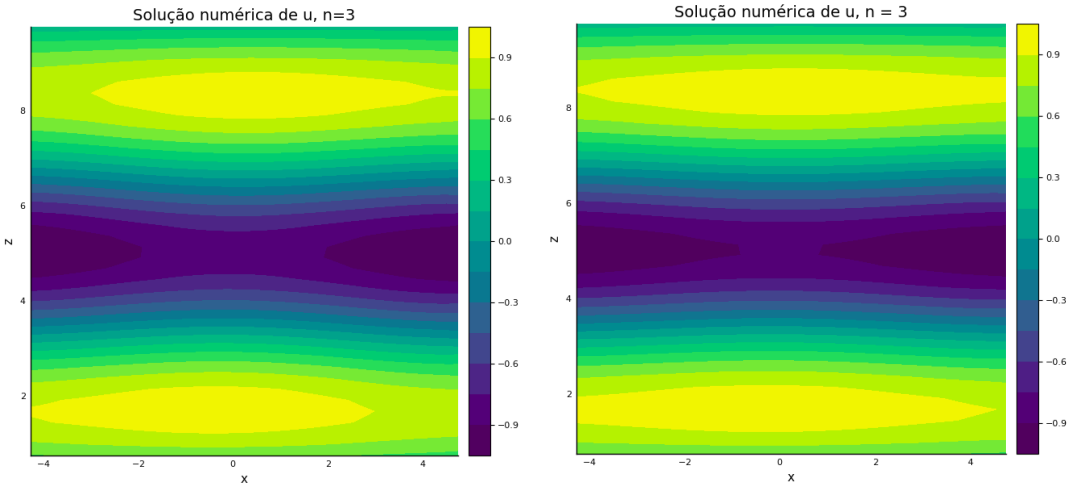
(b) Segundo modo



(c) Terceiro modo

Figura 31: Resposta para os 3 primeiros modos do problema aberto-aberto

É possível observar, para o terceiro modo, a presença um notável erro. Novamente, com a utilização da geometria assimétrica foi possível reduzir o abaulamento, conforme a Figura 32.



(a) Cubo (b) Prisma assimétrico

Figura 32: Gráfico de calor da resposta para u

Mesmo com a introdução da assimetria, a resposta ainda possui um certo abaulamento, apesar de que houve uma melhora. As respostas para os modos utilizando a geometria assimétrica foram calculadas utilizando tanto o BB quanto o BB. A resposta para o terceiro modo pode ser vista na Figura 33, em que é possível observar uma pequena melhora na precisão da resposta.

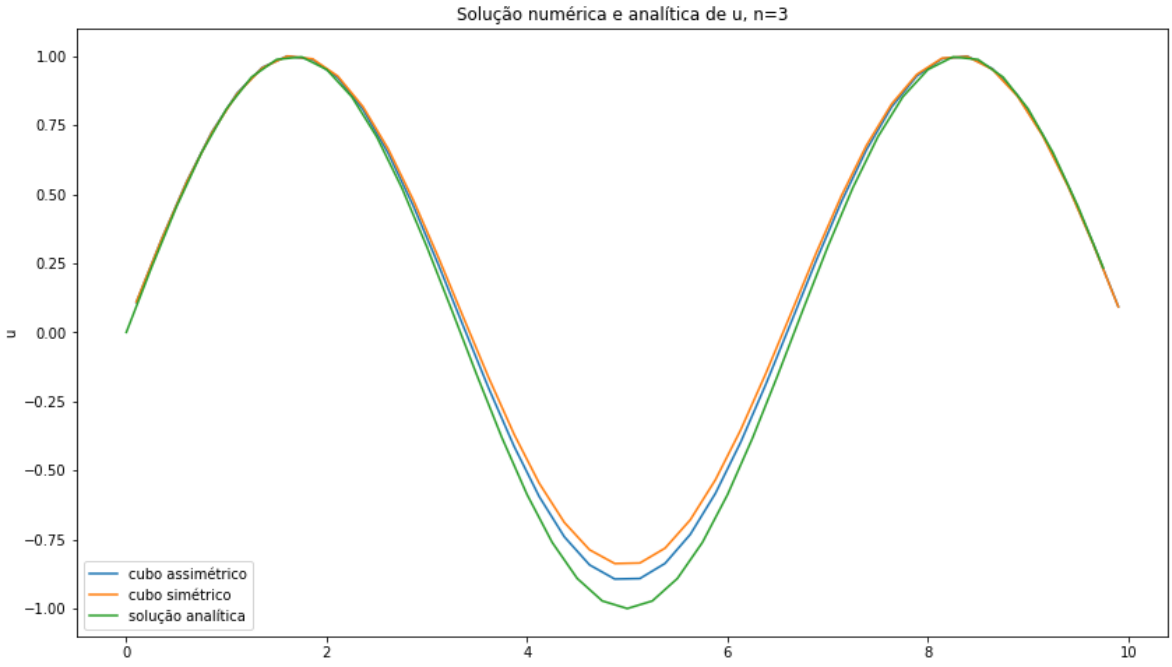


Figura 33: Comparação resposta analítica e numérica utilizando o BB e o Bempp

O erro entre a resposta numérica e a analítica foi avaliado por meio da raiz do valor quadrático médio (RMS) do erro normalizado. Os erros para cada modo, entre a resposta numérica e a analítica, podem ser vistos na Tabela 4.

Tabela 4: Erro das soluções numéricas obtidas pelo BB e pelo Bempp

	n=1	n=2	n=3
BB	0,0033344	0,0028213	0,0265669
Bempp	0,0003359	0,0004193	0,1237878

4.3 Resposta em frequência para os problemas tri-dimensionais em um cubo

Além da análise dos modos no interior do cubo, foi realizada a análise da resposta em frequência para os três casos. A resposta em frequência nada mais é do que obter a pressão para uma determinada faixa de frequências. O principal objetivo de dita análise é identificar as frequências de ressonância de determinado problema.

A resposta foi obtida para 400 frequências, em uma faixa de 45 [rad/s] a 330 [rad/s]. A análise foi realizada em 5 pontos internos do cubo: $z = 0,10$, $z = 2,55$, $z = 5,00$, $z = 7,45$, e $z = 9,90$. Foram utilizados ambos os programas BB e Bempp. A geometria utilizada foi o mesmo cubo utilizado para a análise dos modos, Figura 24.

O MEC utilizado pelo BB foi o método convencional com elementos constantes. Para as integrações, utilizou-se o método de Gauss-Legendre com 12 pontos de integração.

Já para o Bempp foram utilizados elementos lineares contínuos para as faces onde foram aplicadas condições de contorno de Neumann e elementos constantes para as faces onde foram aplicadas condições de Dirichlet. A solução foi obtida utilizando-se do Método do Resíduo Mínimo Generalizado (GMRES).

4.3.1 Caso (i): Cubo fechado-fechado

O caso (i) é o caso fechado-fechado. As condições de contorno utilizadas foram as mesmas utilizadas para a análise dos modos. Para este caso, os gráficos apresentados serão referentes à resposta para o ponto $z = 0,10$.

Inicialmente, apresenta-se a resposta real, na Figura 34. Como se sabe, a parte real de u representa a pressão física obtida. As frequências de ressonância apresentadas como linhas verticais são aquelas dadas pelos números de onda dados pela equação 2.47 para $k = 1, 2, 3$.

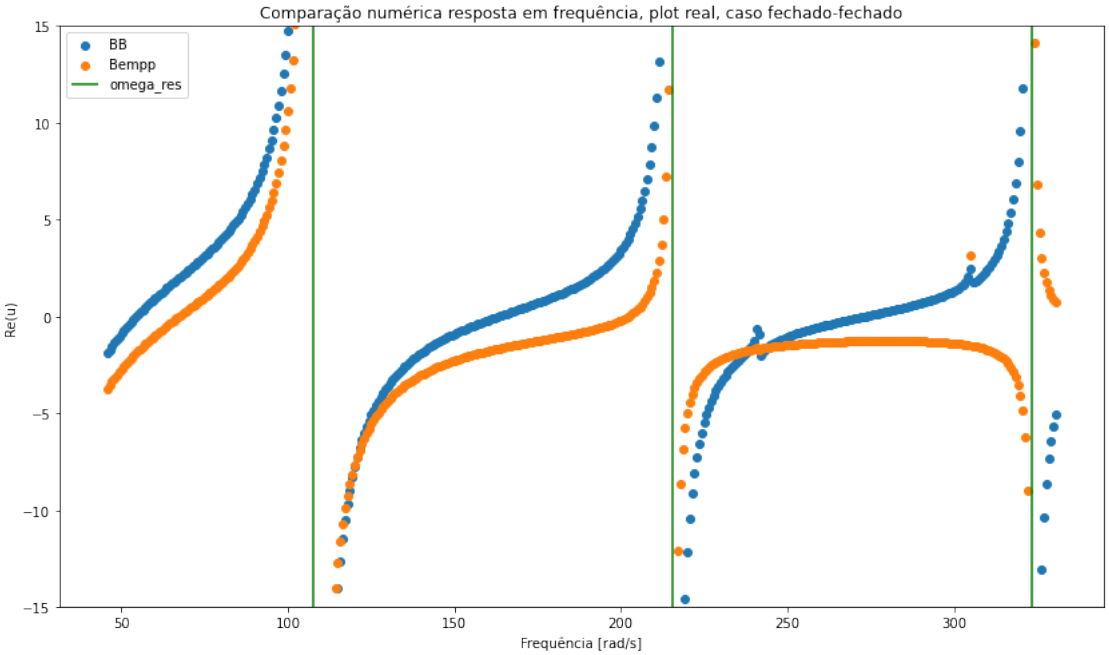


Figura 34: Parte real da resposta em frequência

De forma a se observar os resultados com maior clareza, é comumente utilizado o gráfico absoluto, que considera o valor absoluto da resposta, conforme a Figura 35. Assim, as ressonâncias serão sempre positivas e, portanto, mais facilmente identificáveis.

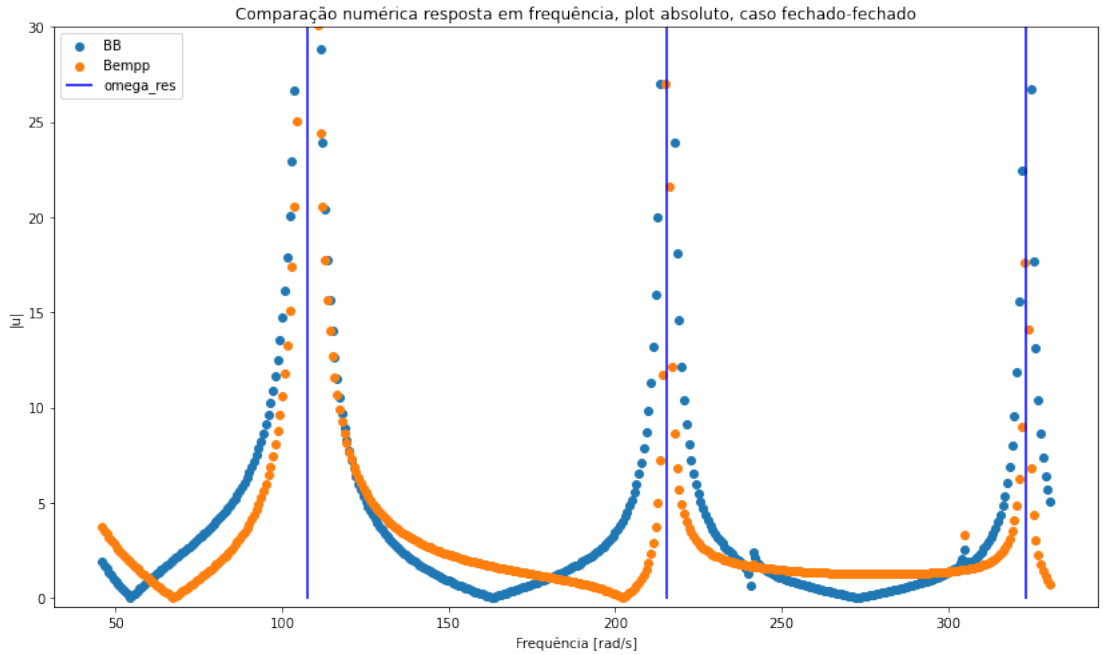


Figura 35: Parte absoluta da resposta em frequência

É possível notar que houve uma boa concordância dos resultados utilizando o BB e o Bempp, especialmente em relação à identificação das frequências de ressonância. Um outro gráfico que permite uma análise ainda mais aprofundada é o gráfico semi-log da resposta absoluta,

que pode ser visto na Figura 36. Além das frequências de ressonância, esse gráfico permite melhor analisar as frequências de anti-ressonância. Em geral, espera-se que tais frequências se encontrem exatamente no meio entre duas frequências de ressonância.

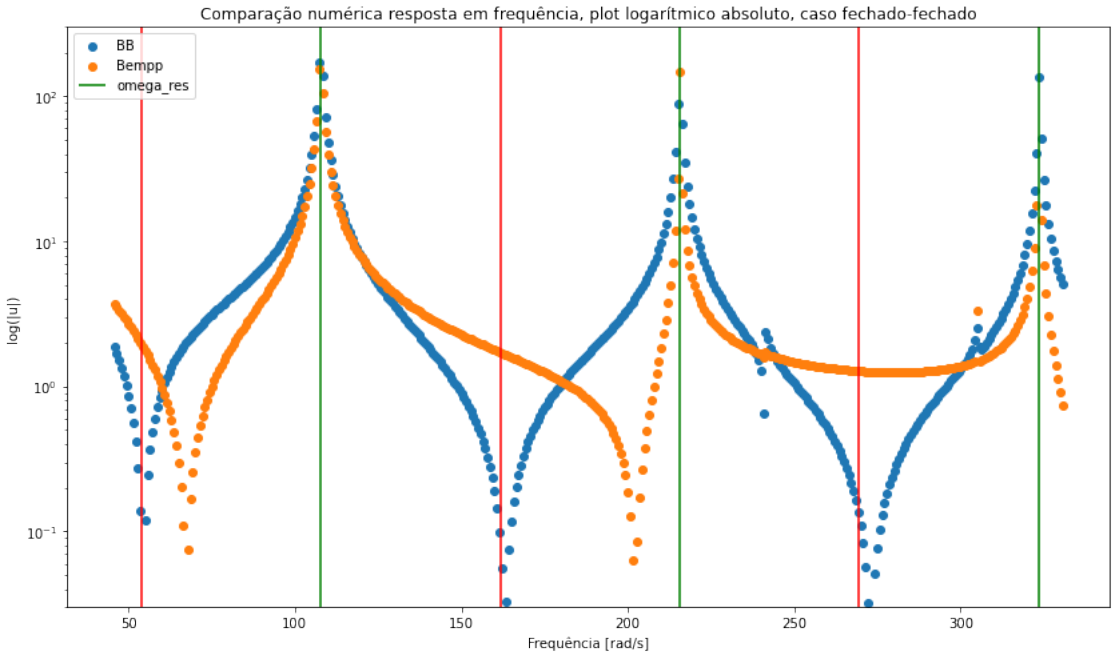


Figura 36: Parte absoluta da resposta em frequência em escala logarítmica

Pelo gráfico, é interessante notar que, aparentemente, o programa BB fez um trabalho melhor em identificar as frequências de anti-ressonância. Por último, é necessário quantificar o erro de cada programa em identificar a frequência de ressonância. O erro foi calculado em termos percentuais, conforme a equação:

$$erro = \frac{\omega_t - \omega_a}{\omega_a} \tag{4.4}$$

em que ω_t é a frequência calculada numericamente e ω_a é a frequência analítica. O método utilizado para obter a frequência de ressonância numérica foi identificar a frequência cuja resposta possuía maior magnitude do valor absoluto. O erro para identificar as 3 primeiras frequências de ressonância pode ser visto na Tabela 5.

Tabela 5: Erro das frequências de ressonância obtidas pelo BB e pelo Bempp

	n=1	n=2	n=3
BB	0,137%	0,087%	0,071%
Bempp	0,137%	0,087%	0,071%

Os erros encontrados foram exatamente os mesmos. Note que a frequência foi discretizada em 400 pontos. Portanto, o erro ser exatamente o mesmo quer dizer que ambos os programas obtiveram suas respostas máximas para os mesmos pontos de frequência. É esperado que, quanto mais pontos de frequência forem utilizados, melhor será a precisão em identificar as frequências

de ressonância.

4.3.2 Caso(ii): Cubo aberto-fechado

O caso (ii) é o caso aberto-fechado. As condições de contorno utilizadas foram as mesmas utilizadas para a análise dos modos. Para este caso, os gráficos apresentados serão referentes à resposta para o ponto $z = 5,00$.

Inicialmente, apresenta-se a resposta real, na Figura 37. As frequências de ressonância apresentadas como linhas verticais são aquelas dadas pelos números de onda dados pela equação 2.52 para $k = 1, 3, 5$.

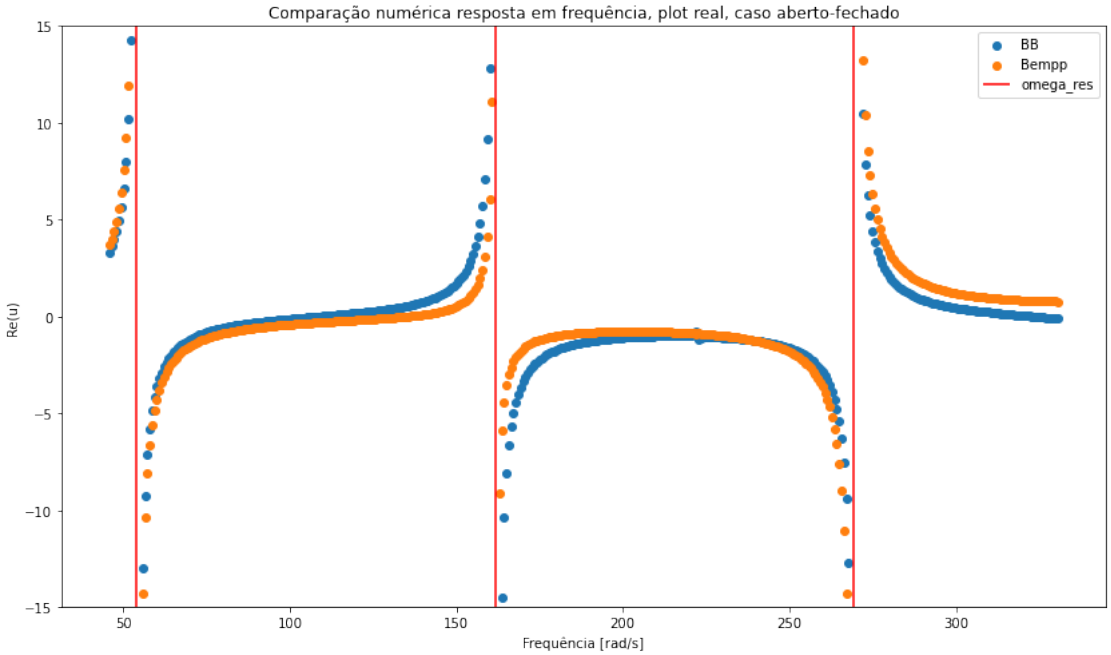


Figura 37: Parte real da resposta em frequência

De forma a se observar os resultados com maior clareza, é comumente utilizado o gráfico absoluto, que considera o valor absoluto da resposta, conforme a Figura 38. Assim, as ressonâncias serão sempre positivas e, portanto, mais facilmente identificáveis.

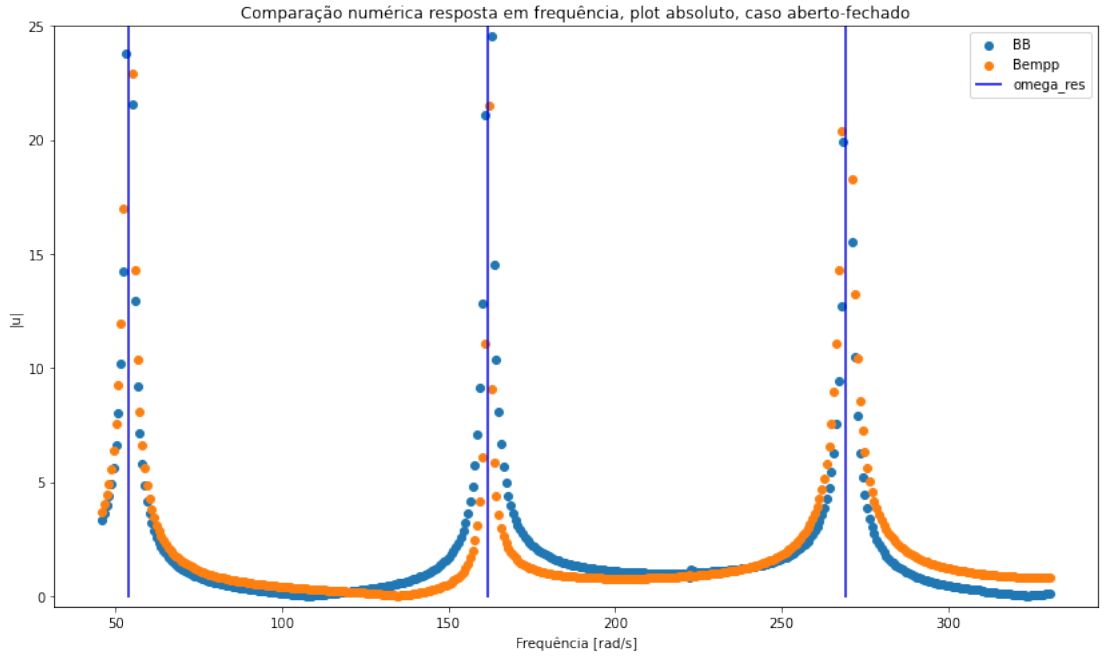


Figura 38: Parte absoluta da resposta em frequência

É possível notar que houve uma boa concordância dos resultados utilizando o BB e o Bempp, especialmente em relação à identificação das frequências de ressonância. Um outro gráfico que permite uma análise ainda mais aprofundada é o gráfico semi-log da resposta absoluta, que pode ser visto na Figura 39.

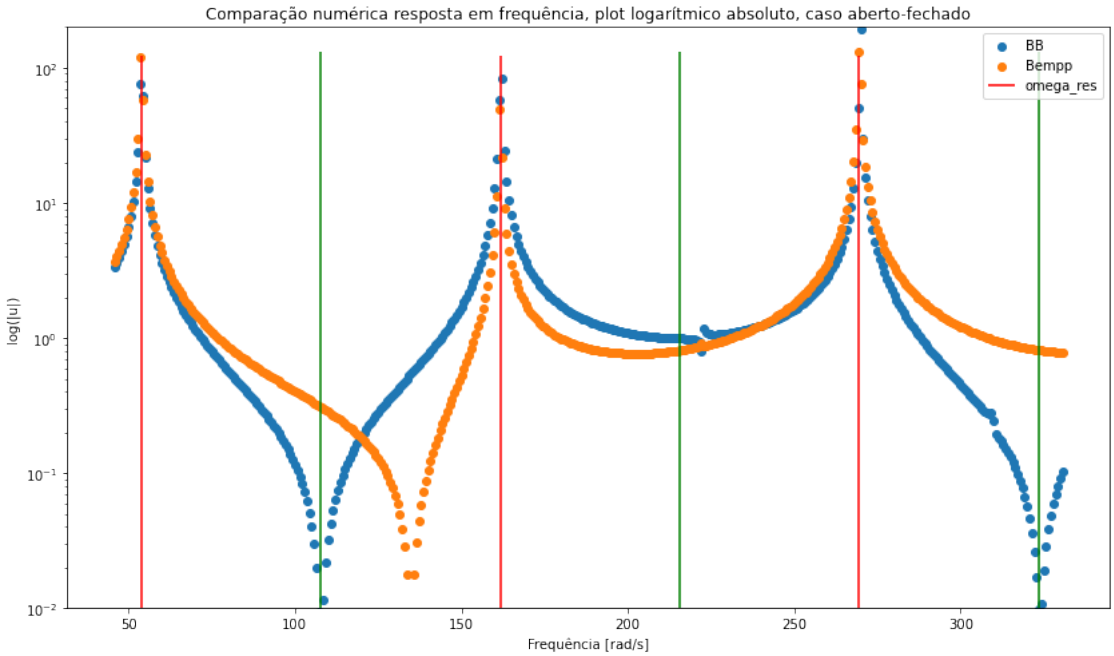


Figura 39: Parte absoluta da resposta em frequência em escala logarítmica

Novamente, o BB fez um melhor trabalho em identificar as frequências de anti-ressonância. Contudo, nenhum dos programas identificou anti-ressonância entre o segundo e terceiro modo.

Por último, é necessário quantificar o erro de cada programa em identificar a frequência de ressonância. O método utilizado para obter a frequência de ressonância numérica foi identificar a frequência cuja resposta possuía maior magnitude do valor absoluto. O erro para identificar as 3 primeiras frequências de ressonância pode ser visto na Tabela 6.

Tabela 6: Erro das frequências de ressonância obtidas pelo BB e pelo Bempp

	n=1	n=2	n=3
BB	0,425%	0,325%	0,21%
Bempp	0,425%	0,117%	0,055%

4.3.3 Caso(iii): Cubo aberto-aberto

O caso (iii) é o caso aberto-aberto. As condições de contorno utilizadas foram as mesmas utilizadas para a análise dos modos. Para este caso, os gráficos apresentados serão referentes à resposta para o ponto $z = 7,45$.

Inicialmente, apresenta-se a resposta real, na Figura 40. As frequências de ressonância apresentadas como linhas verticais são aquelas dadas pelos números de onda dados pela equação 2.54 para $k = 1, 2, 3$.

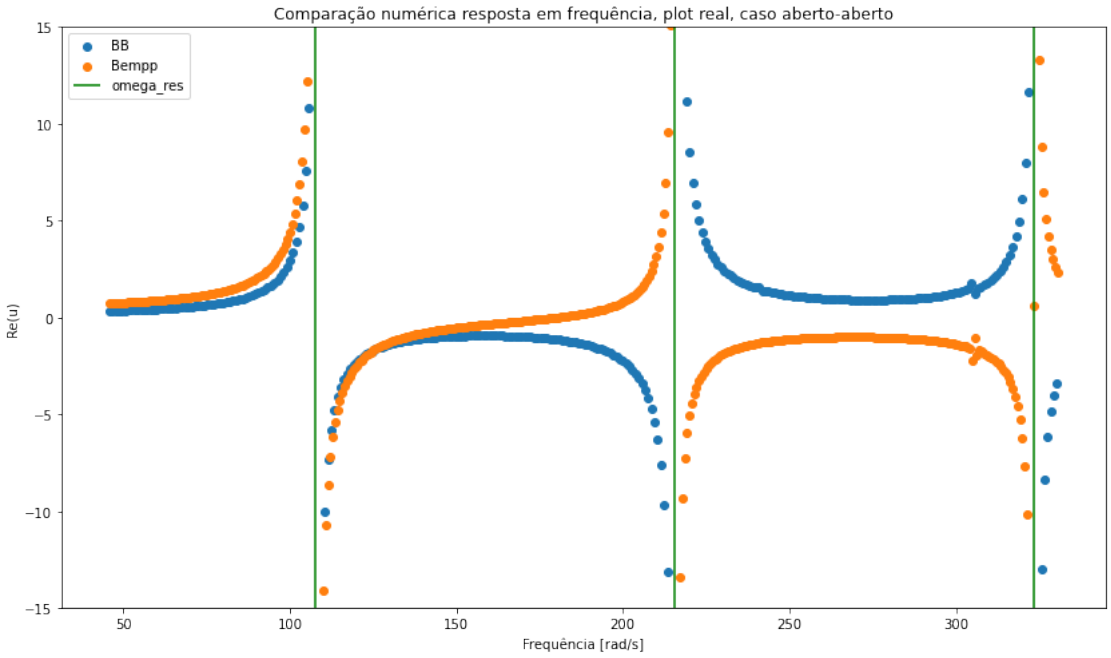


Figura 40: Parte real da resposta em frequência

De forma a se observar os resultados com maior clareza, é comumente utilizado o gráfico absoluto, que considera o valor absoluto da resposta, conforme a Figura 41. Assim, as ressonâncias serão sempre positivas e, portanto, mais facilmente identificáveis.

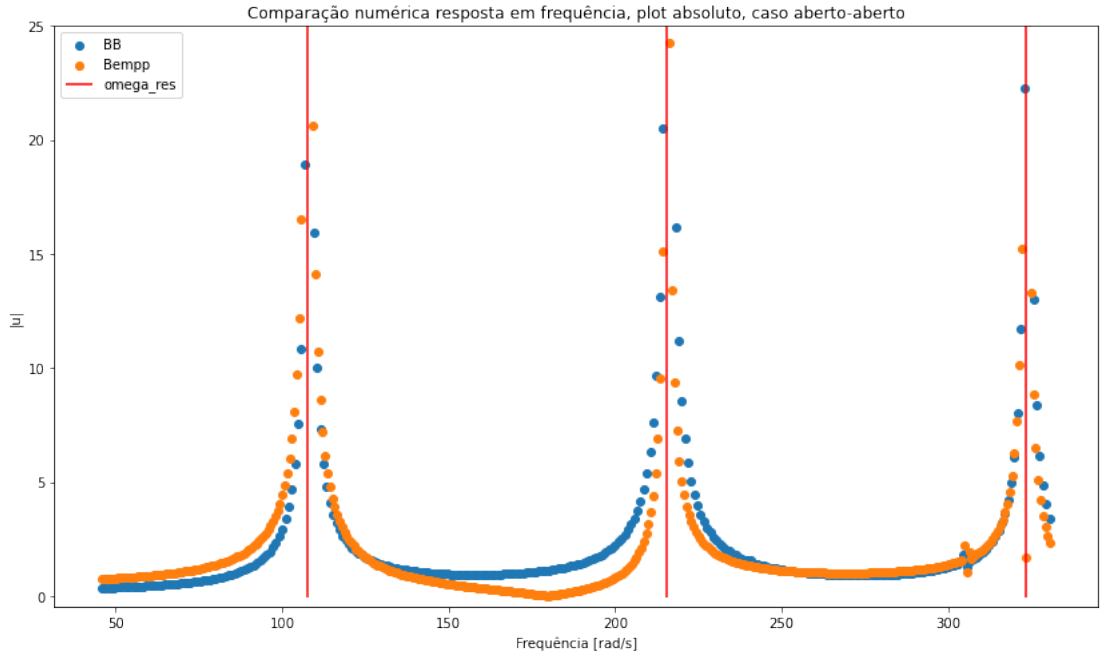


Figura 41: Parte absoluta da resposta em frequência

É possível notar que houve uma boa concordância dos resultados utilizando o BB e o Bempp, especialmente em relação à identificação das frequências de ressonância. Um outro gráfico que permite uma análise ainda mais aprofundada é o gráfico semi-log da resposta absoluta, que pode ser visto na Figura 42.

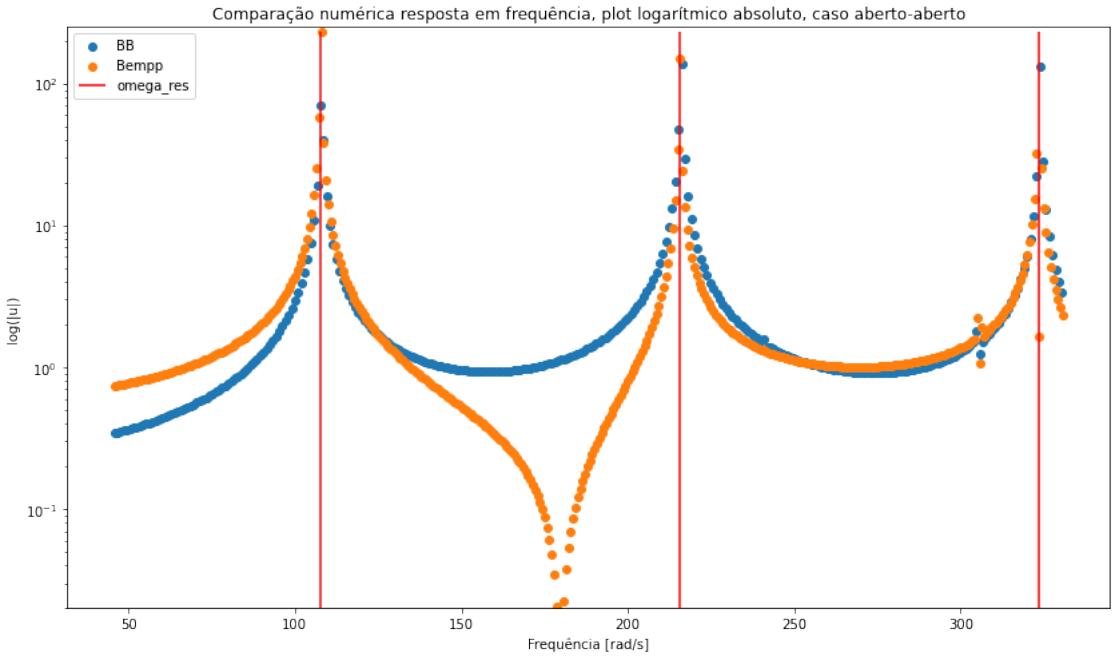


Figura 42: Parte absoluta da resposta em frequência em escala logarítmica

Para este caso, as frequências de anti-ressonância não foram corretamente identificadas por nenhum dos programas. O Bempp apresenta um vale em seu gráfico, contudo este não é

simétrico em relação às ressonâncias do primeiro e segundo modo.

Por último, é necessário quantificar o erro de cada programa em identificar a frequência de ressonância. O método utilizado para obter a frequência de ressonância numérica foi identificar a frequência cuja resposta possuía maior magnitude do valor absoluto. O erro para identificar as 3 primeiras frequências de ressonância pode ser visto na Tabela 7.

Tabela 7: Erro das frequências de ressonância obtidas pelo BB e pelo Bempp

	n=1	n=2	n=3
BB	0,083%	0,308%	0,144%
Bempp	0,137%	0,087%	0,15%

4.4 Estudo comparativo dos tempos de processamento

Uma análise comparativa dos tempos de processamento foi realizada utilizando os programas BB e Bempp. Para tal, o problema do cubo fechado-fechado, caso(i), foi resolvido utilizando 5 malhas com diferentes níveis de refinamento, conforme mostra a Tabela 8.

Tabela 8: Número de elementos de cada uma das malhas

	1	2	3	4	5
Número de elementos	410	1476	6226	10378	13526

O programa BB é escrito em Julia, uma linguagem que, por ser compilada, é mais rápida do que o Python, que é uma linguagem interpretada. Contudo, o Bempp utiliza o Pyopencl, o qual permite compilação *Just-in-Time*, podendo alcançar velocidades semelhantes a linguagens interpretadas. Além disso, outra diferença primordial é que o Bempp utiliza o método de Galerkin simétrico, consequentemente obtendo matrizes simétricas que podem permitir um processamento em menor tempo.

A comparação entre os dois programas em relação ao tempo de processamento é mais complexa do que seria de se imaginar, devido às diferenças de implementação do MEC para cada um dos *softwares*. Por exemplo, o uso do BB requer que se especifique o número de pontos de Gauss usados para a integração numérica, sendo que essa escolha resulta em um impacto significativo no tempo de processamento. Outra diferença é que o sistema linear no BB pôde ser resolvido de forma direta, enquanto no Bempp houve a necessidade de utilizar um *solver* iterativo. Apesar do Bempp permitir o uso de *solvers* diretos, isso requer que as matrizes sejam montadas de forma densa, sendo que essa não é a forma padrão e recomendada do Bempp. Mais uma diferença reside no fato de que foram utilizados elementos constantes no BB, enquanto elementos lineares tiveram de ser utilizados no Bempp, devido a particularidades na implementação que não permitem o uso exclusivo de elementos constantes para resolver problemas de Neumann.

Tendo em vista todas essas diferenças, a solução para prosseguir com a comparação

dos tempos de processamento foi manter as configurações que foram utilizadas nas secções anteriores para o caso (i), fechado-fechado. Essa escolha resguarda as particularidades de cada um dos programas, utilizando a formulação que é mais adequada a cada um deles. Como a implementação é a mesma utilizada anteriormente, é coerente afirmar que a comparação de tempo de processamento foi realizada para um determinado nível de precisão dos resultados, apresentado anteriormente na forma de erro em relação à solução analítica.

Um aspecto que deve ser ressaltado é que não foi possível utilizar a paralelização do código BB. Tal impossibilidade tem origem no fato de que um programa de código aberto utiliza diversas bibliotecas externas. Atualizações dessas bibliotecas, bem como atualizações da própria linguagem de programação, são extremamente comuns e frequentes. Tais atualizações podem fazer com que determinadas partes do programa deixem de funcionar corretamente, sendo necessário uma atualização do *software* para adequar tais mudanças. Para que um programa se mantenha completamente funcional, é necessária uma atualização contínua e constante deste, o que é, certamente, uma tarefa que requer um grande esforço e dedicação de tempo. Infelizmente, no momento da escrita deste projeto, a paralelização não estava funcionando adequadamente, por mais que funcionasse anteriormente, como por exemplo foi demonstrado em Ferreira (2020). Caso a paralelização estivesse funcionando, provavelmente tornaria o programa BB ainda mais rápido.

Outro fator de grande importância para a comparação entre programas da ótica do tempo de processamento são as especificações da máquina sendo utilizada. Evidentemente, uma comparação justa deve ser realizada utilizando-se máquinas com mesma capacidade de processamento para ambos os programas. Além disso, durante a realização deste projeto, observou-se que existe uma certa variabilidade dos resultados comparativos dependendo da máquina usada. Dessa forma, visando obter um resultado de maior confiabilidade, a comparação dos tempos de processamento foi efetuada utilizando-se 3 máquinas distintas.

O primeiro comparativo foi feito utilizando o *Google Colaboratory*, um serviço *Jupyter Notebook* hospedado que providencia acesso gratuito a recursos computacionais por meio da nuvem do *Google*, incluindo GPU, para qualquer computador. Os processadores disponível são dois Intel(R) Xeon(R) com *clock speed* de 2,3 GHz, sendo que cada processador físico possui 2 *siblings*. Além disso, possui disponível 12 GB de memória RAM. Os resultados do comparativo, em segundos, podem ser vistos na Figura 43

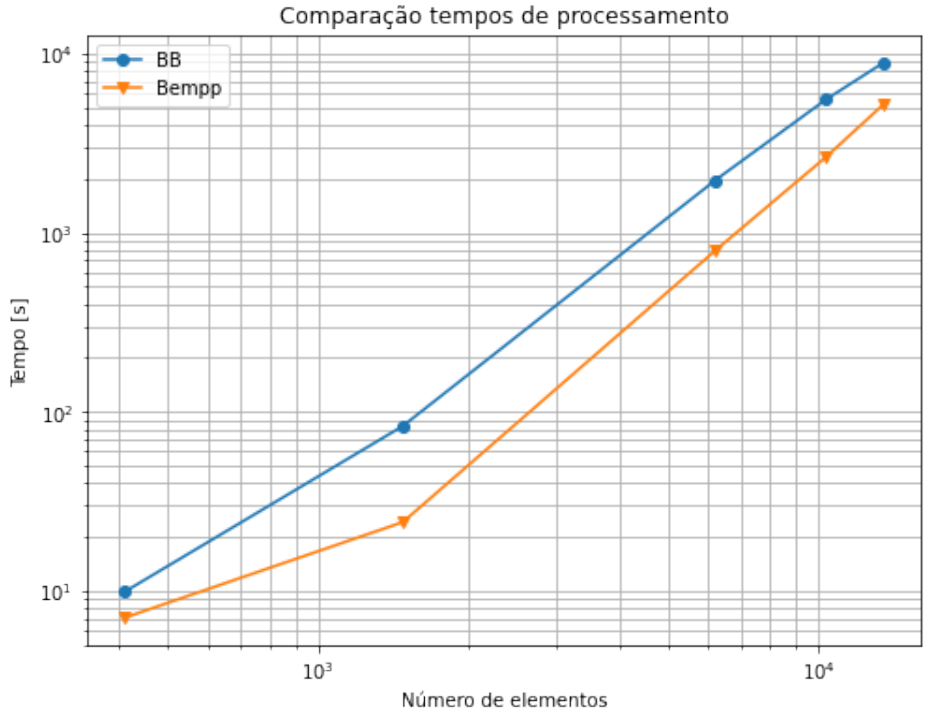


Figura 43: Relação entre o tempo de processamento e o número de elementos usando o *Google Colab* (máquina 1)

É notável que o Bempp possuiu, para todos os números de elementos, uma notável vantagem em termos de tempo de processamento. Em média, o Bempp foi cerca de 50% mais rápido em relação ao BB. Por mais que esse processamento inferior possa significar simplesmente que um programa é mais eficiente que o outro, é possível que, pelo fato de o *Google Colaboratory* ser uma ferramenta voltada para a programação em Python, isso tenha influenciado de alguma forma os resultados comparativos.

A segunda máquina utilizada para o estudo comparativo foi um PC com placa-mãe AMD FX(tm)-8300 com processador de oito núcleos, 16 GB de RAM e um SSD SanDisk de 120 GB. O sistema operacional é o Ubuntu 18.04 e a versão do Julia a 1.1. Os resultados comparativos para esse computador podem ser vistos na Figura 44.

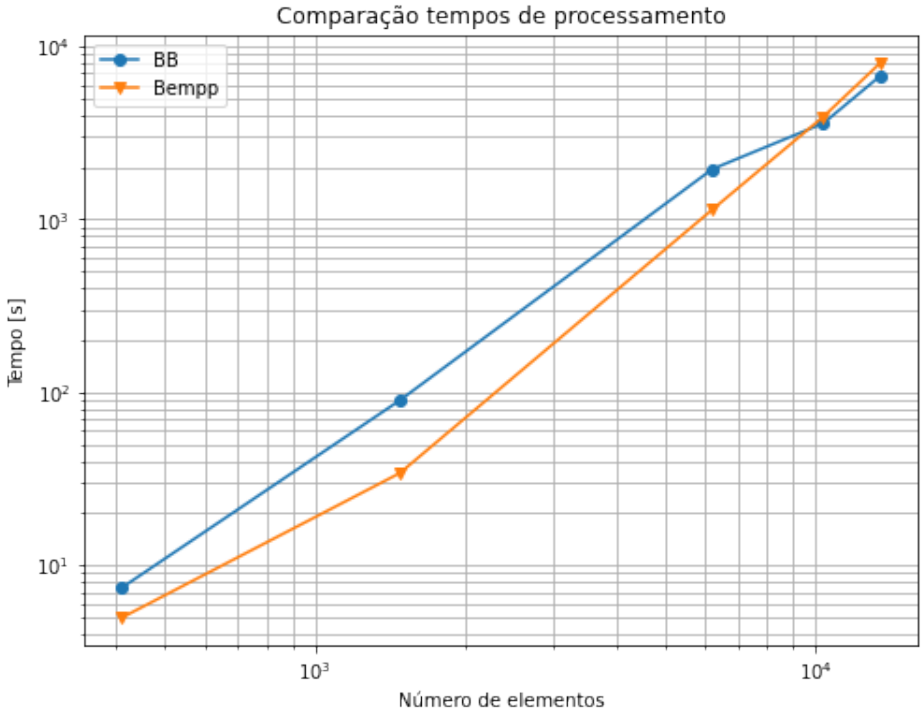


Figura 44: Relação entre o tempo de processamento e o número de elementos da segunda máquina

Para a segunda máquina utilizada neste estudo, o BB e o Bempp se alternaram como sendo mais eficientes. O Bempp se mostrou mais rápido para problemas com baixo número de elementos. Entretanto, em algum ponto entre 6226 elementos e 10378 elementos o BB se tornou mais rápido. O motivo para tal inversão no comportamento não é trivial, e certamente é um interessante objeto de estudo futuro.

O último conjunto de dados foi obtido utilizando um *notebook* equipado com um processador Intel(R)Core(TM)i7-7500U CPU@2,70GHz e 8 GB de memória RAM. É importante notar que a memória de 8 GB não foi suficiente para rodar a malha com 13526 elementos utilizando o BB, indicando um uso superior de memória deste programa. A comparação dos tempos de processamento pode ser visto na Figura 45.

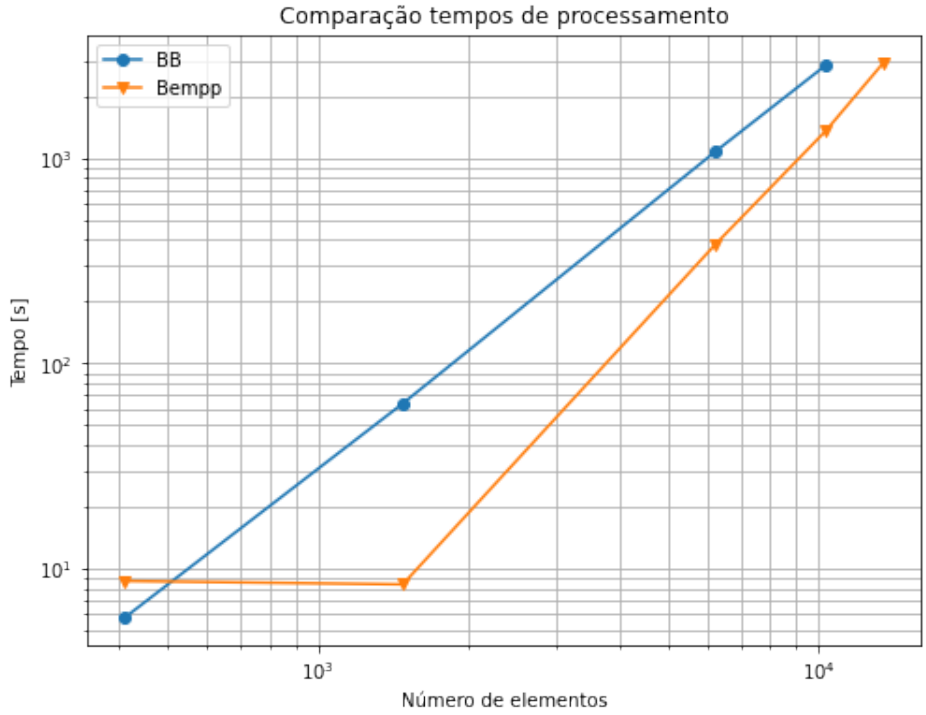


Figura 45: Relação entre o tempo de processamento e o número de elementos do terceiro computador

Para o último computador utilizado, o Bempp se mostrou, em geral, mas rápido do que o BB. Uma observação interessante é que as duas primeiras malhas para o Bempp demoraram aproximadamente o mesmo tempo. Entretanto, esse não foi um problema de carregamento de pacotes no início da rotina, uma vez que o programa foi feito de forma a se descartar o primeiro resultado obtido para evitar esse tipo de erro.

Tendo em mãos os resultados obtidos para os três computadores, montou-se a Tabela 9, um resumo comparativo dos resultados obtidos, em segundos.

Tabela 9: Comparativo dos resultados de tempo de processamento entre Bempp e BB

		Máquina 1	Máquina 2	Máquina 3
Malha 1	Bempp	7,10	4,98	8,73
	BB	9,92	7,43	5,79
Malha 2	Bempp	24,24	34,16	8,40
	BB	83,39	90,02	63,91
Malha 3	Bempp	795,54	1139,47	380,07
	BB	1963,36	1953,34	1076,65
Malha 4	Bempp	2652,93	3919,44	1366,21
	BB	5553,90	3590,34	2848,80
Malha 5	Bempp	5222,41	8065,73	2927,81
	BB	8908,38	6788,67	-

Analisando-se a Tabela 9, é possível notar que, em geral, o tempo de processamento do Bempp foi inferior ao do BB. Entretanto, como exposto anteriormente, o número de pontos

de Gauss utilizados no BB para as integrações numéricas possui grande influência no tempo de processamento. Para esta análise de tempo de processamento, foram utilizados 12 pontos de Gauss, de forma a ser condizente com o número de pontos usado no Capítulo 4.2. Com o intuito de investigar de forma mais profunda os resultados de tempo de processamento, fez-se um teste para entender qual seria a influência de se usar apenas 6 pontos de Gauss. Com isso, espera-se que o tempo de processamento seja reduzido, mas também a precisão dos resultados seja menor. Os resultados de tempo de processamento utilizando 12 e 6 pontos de Gauss obtidos pela máquina 2 estão disponíveis na Tabela 10.

Tabela 10: Tempos de processamento com o BB usando 12 e 6 pontos de Gauss

	1	2	3	4	5
BB (12 pontos de Gauss)	7,43	90,02	1953,34	3590,34	6788,67
B(6 pontos de Gauss)	2,85	30,45	664,30	1233,60	1737,44

Além disso, o erro RMS em relação à solução analítica foi calculado, da mesma forma como foi feito no Capítulo 4.2. Os erros RMS usando apenas 6 pontos de Gauss para o caso fechado-fechado, comparados com os erros usando 12 pontos de Gauss, podem ser vistos na Tabela 11.

Tabela 11: Erro das soluções numéricas obtidas pelo BB com 12 e 6 pontos de Gauss

	n=1	n=2	n=3
BB (12 pontos de Gauss)	0,0005017	0,0024834	0,0102628
BB (6 pontos de Gauss)	0.0020887	0.0043891	0.0128213

Por meio das tabelas 10 e 11 é possível notar uma considerável economia de tempo de processamento. Na realidade, observou-se que a utilização de 6 pontos de Gauss resultou em um tempo de processamento 33,2% inferior em relação ao tempo com 12 pontos, enquanto o erro RMS foi 39,3% superior. Os novos tempos de processamento, quando comparados aos tempos obtidos pelo Bempp, são notoriamente inferiores para todas as malhas. Essa breve análise mostra que a quantidade de pontos de Gauss é um parâmetro de extrema relevância, e um estudo para analisar a influência dessa escolha sobre a precisão e o tempo de processamento seria extremamente interessante.

Por último, como já exposto anteriormente, ambos os programas possuem ainda a possibilidade de utilizar métodos de aceleração para problemas de larga escala, sendo que o BB utiliza matrizes hierárquicas e o Bempp utiliza do FMM. A utilização de métodos de aceleração para o MEC é de grande importância para que este seja competitivo em termos de velocidade com outros métodos numéricos, especialmente quando se trata de problemas de larga escala. Contudo, para este projeto, uma comparação utilizando tais métodos não foi possível.

Em relação ao Bempp, o uso do FMM é possível, mas não existem exemplos da imple-

mentação deste disponível no *site*[5]. Em tentativa própria do autor de resolver o problema do cubo fechado-fechado utilizando o FMM, foi possível obter resultados coerentes com a resposta analítica esperada. Entretanto, contrário ao propósito do método de aceleração, o tempo de processamento foi extremamente alto quando comparado a resolução utilizando o método convencional, especialmente para problemas de larga escala. Por exemplo, a resolução do problema para a malha com 13526 elementos foi abortada pois já demorava mais de 17 horas. Por esse motivo, não foi traçado nenhum comparativo no Bempp entre o método convencional e o FMM.

Já em relação ao BB, como já comentado, a constante atualização de bibliotecas externas pode fazer com que determinada parte do programa deixe de funcionar. No momento da escrita deste projeto, o uso das matrizes hierárquicas não está inteiramente funcional, por mais que a utilização correta já tenha sido realizada anteriormente, como por exemplo em Ferreira (2020). A resposta obtida não foi coerente com a resposta analítica esperada, apesar de não ser, também, totalmente aleatória, possuindo características de uma onda harmônica. Para malhas mais refinadas, observou-se que o *solver* iterativo estava com dificuldades em obter uma convergência da solução, sendo que as simulações foram interrompidas após o *solver* ter alcançado o número máximo de iterações definidas, ou seja, 60000 iterações.

4.5 Modelagem numérica do levitador acústico *TinyLev*

Para demonstrar a capacidade dos programas em estudo de resolver problemas de engenharia, foi realizada a análise numérica do levitador acústico *TinyLev*. Esse problema foi resolvido, inicialmente, em uma formulação bi-dimensional simplificada. Em sequência, o campo acústico foi determinado utilizando uma formulação tri-dimensional. A geometria tri-dimensional foi analisada utilizando os programas Bempp e BB, de forma que os resultados obtidos por ambos os programas possam ser comparados.

4.5.1 Modelagem bi-dimensional do *TinyLev*

Para a modelagem numérica bi-dimensional, foi utilizado o programa BB, desenvolvido na Universidade de Brasília. Infelizmente, uma comparação com o programa Bempp não foi possível para o caso bi-dimensional, uma vez que o Bempp só suporta geometrias tri-dimensionais. Utilizou-se uma geometria simplificada, conforme pode ser visto na Figura 46. Geometrias simplificadas de um problema mais complexo são frequentemente utilizadas, principalmente para se ter uma concepção inicial da resposta esperada.

[5]Disponível em: <http://bempp.com>.

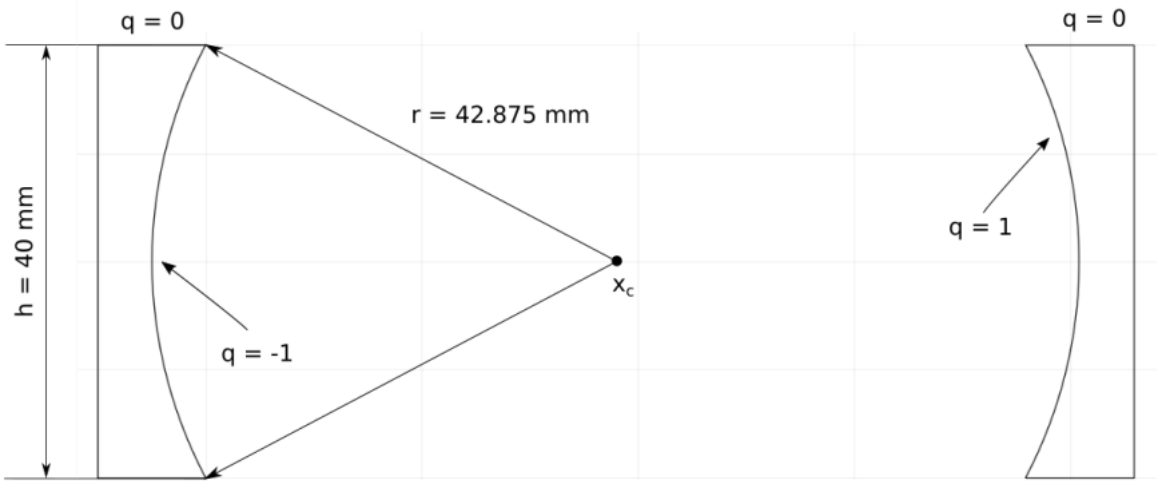


Figura 46: Geometria do levitador acústico (FERREIRA, 2020)

As condições de contorno são de uma superfície rígida, com exceção das faces dos cilindros voltadas para o interior do levitador. Para tal conjunto de faces, aplica-se uma condição de fluxo $\overline{u}_N = \pm 1$. O sinal positivo é adotado para os cilindros de um dos lados e o sinal negativo para os cilindros do lado oposto. O problema foi resolvido para uma frequência de 40 kHz e considerando a velocidade do som como igual a 344 m/s.

Para a solução do problema, utilizou-se de elementos do tipo *Non-uniform rational B-spline* (NURBS). O potencial de campo, u, foi calculado para 2500 pontos internos. Os resultados obtidos podem ser observados nas Figuras 47 e 48.

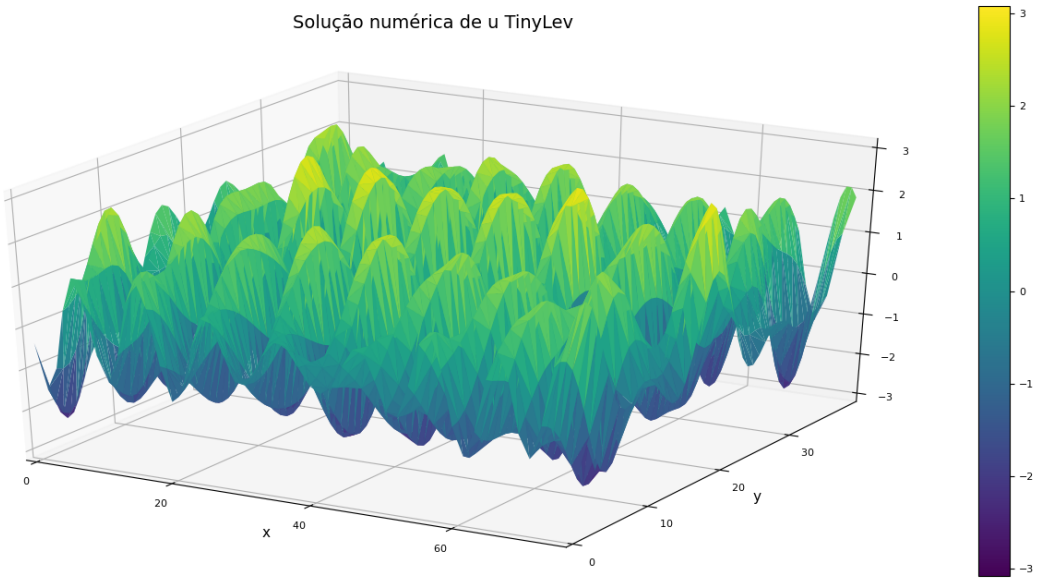


Figura 47: Curva de superfície obtida para o levitador acústico TinyLev

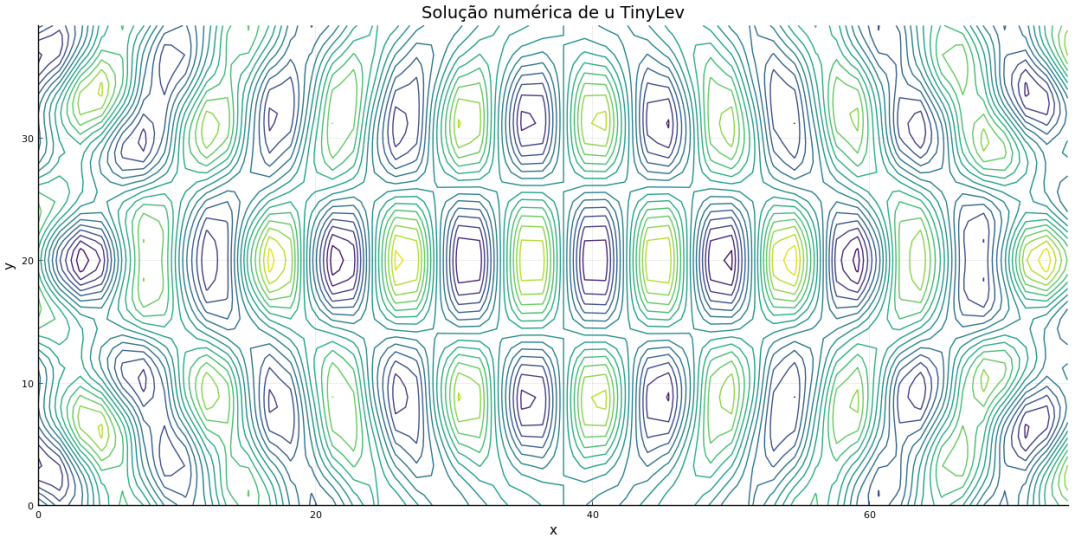


Figura 48: Curva de contorno obtida para o levitador acústico TinyLev

4.5.2 Modelagem tri-dimensional do *TinyLev*

A modelagem tri-dimensional do *TinyLev* foi realizada com uma geometria muito mais fiel à realidade, em que todos os cilindros são representados por completo. Tal geometria foi gerada utilizando o programa Gmsh, mesmo programa que foi utilizado em seguida para a construção da malha nas superfícies da geometria. A malha possui 4314 elementos triangulares e pode ser visualizada na Figura 49.

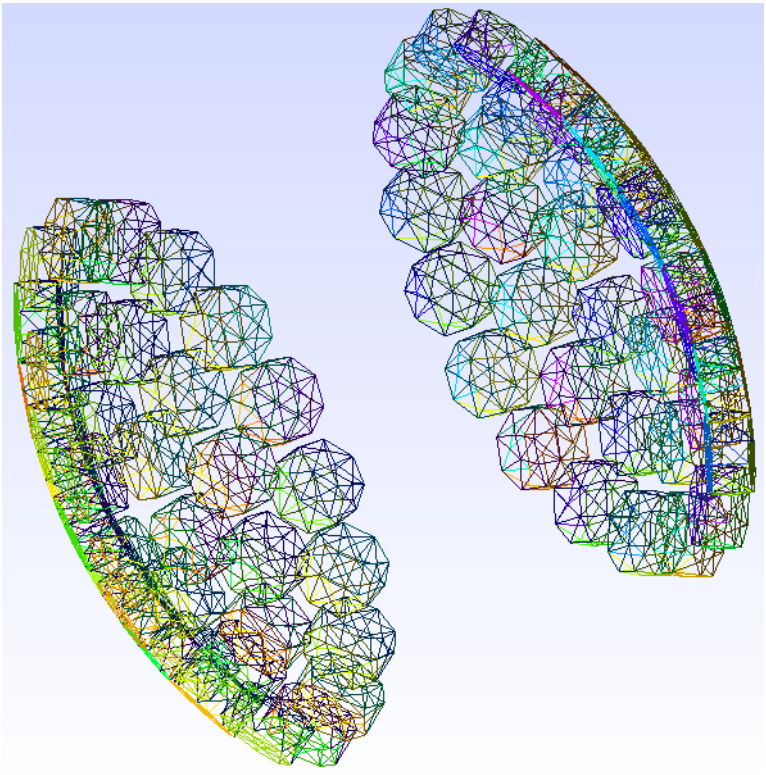


Figura 49: Malha do *TinyLev* tri-dimensional

A simulação foi, primeiramente, efetuada no programa Bempp. Foram utilizados funções espaço do tipo descontínuo constante e do tipo contínuo polinomial de grau 1 (elementos lineares). A frequência e a velocidade do som adotadas foram iguais ao problema bi-dimensional. As condições de contorno aplicadas foram semelhantes àquelas aplicadas para o problema bi-dimensional e podem ser vistas na Figura 50

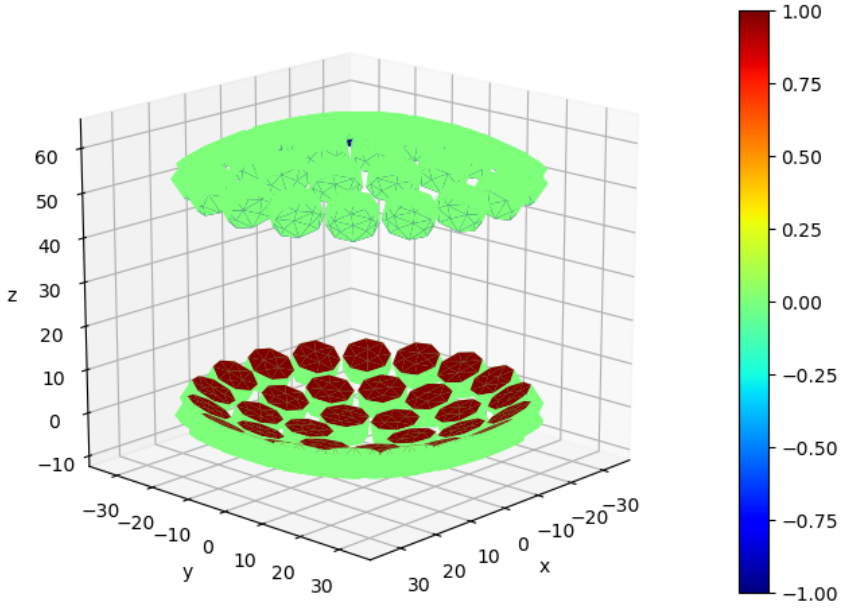


Figura 50: Condições de contorno de fluxo aplicadas para o *TinyLev*

Como só foram utilizadas condições de contorno de fluxo, as variáveis de pressão no contorno foram determinadas utilizando-se a equação 3.39 para problema de Neumann exterior. Com as variáveis determinadas no contorno, a equação 3.37 foi utilizada para se determinar o campo de pressão acústica em 10000 pontos distribuídos em um plano no domínio exterior. Esse plano está localizado de forma que o eixo de simetria axial do levitador está contido a ele. O gráfico de calor normalizado do campo de pressão acústica em tal plano pode ser visto na Figura 51.

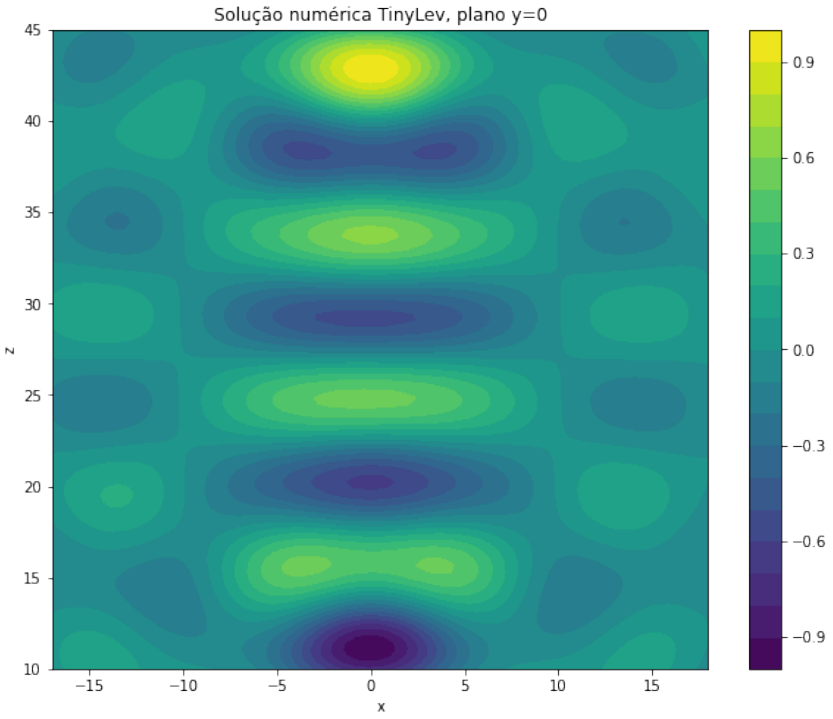


Figura 51: Campo de pressão obtido para o levitador acústico *TinyLev*

O campo de pressão obtido pelo Bempp está de acordo com aquilo que seria esperado de acordo com a literatura e a análise experimental conduzida por Ferreira et al. (2021). Agora, o mesmo problema deve ser resolvido utilizando-se o BB.

A implementação no BB utilizou as mesmas condições de contorno e frequência utilizadas no Bempp. Foram usados elementos constantes em toda a malha. O campo de pressão foi analisado nos mesmos pontos internos utilizados no Bempp. O resultado foi claramente divergente daquele apresentado na Figura 51. Contudo, o motivo de tal divergência não foi, provavelmente, um erro sistemático e generalizado do BB, e sim um erro na implementação, uma vez que um resultado incorreto muito semelhante havia sido obtido anteriormente no Bempp. A comparação do resultado inicial incorreto, obtido no Bempp, com o resultado incorreto obtido no BB pode ser visto nas Figuras 52 e 53

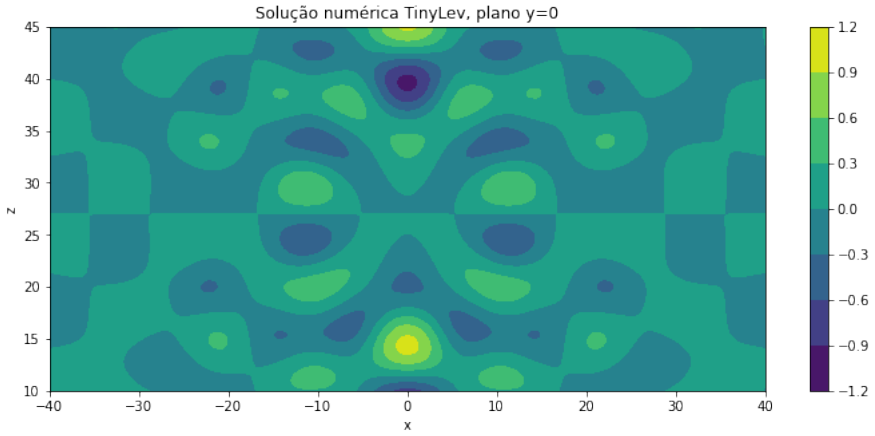


Figura 52: Campo de pressão obtido incorretamente com o Bempp

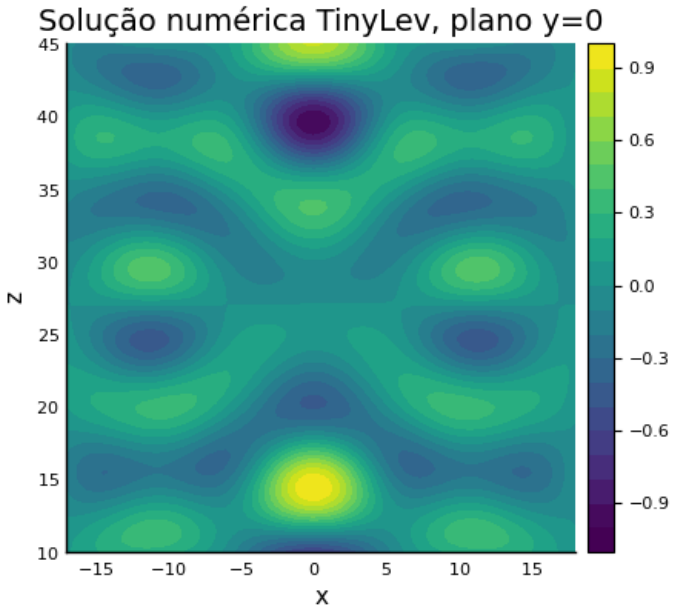


Figura 53: Campo de pressão obtido incorretamente com o BB

Como os resultados incorretos são semelhantes, é de se esperar que a mesma alteração, realizada no Bempp, que solucionou o problema, resolva também o problema no BB. No caso do Bempp, o gráfico da Figura 52 foi obtido utilizando as equações de Neumann para o problema interior, 3.34 e 3.31. A mudança das equações para aquelas adequadas para o problema exterior resultou na obtenção da solução correta. Contudo, até o momento da escrita deste relatório, o autor não conseguiu realizar a implementação adequada no BB.

4.6 Acoplamento do método dos elementos de contorno com o método dos elementos finitos utilizando o Bempp

O acoplamento do MEC com o MEF é uma solução prática e eficiente para se resolver um problema de engenharia. A ideia geral é combinar as duas formulações, de forma que cada

uma seja usada naquilo que é melhor e mais eficiente, gerando assim uma formulação combinada otimizada.

Esse acoplamento pode ser realizado utilizando-se exclusivamente programas de código aberto disponíveis gratuitamente para qualquer um. O Bempp já possui, integrado ao seu código, uma ferramenta para acoplamento com o programa FEniCS Project [6]. O FEniCS é um programa de código aberto extremamente eficiente e completo, utilizado para se resolver equações diferenciais por meio do método de elementos finitos.

4.6.1 Problema de espalhamento em um cubo unitário

Para demonstrar a capacidade de acoplamento das formulações utilizando o Bempp e o FEniCS, apresenta-se o problema de espalhamento em um cubo unitário. A solução para esse problema utilizando o Bempp está disponível livremente *online*[7], na forma de um *Jupyter Notebook*. Neste projeto, foram apresentados os pontos mais importantes da formulação do problema.

Considere um cubo unitário. Seu interior é definido pelo domínio Ω_1, enquanto seu exterior é definido pelo domínio Ω_2. Os parâmetros do material no ambiente interno são diferentes dos parâmetros do material do ambiente externo. Considere também a presença de uma onda incidente plana, dada por:

$$u^I = e^{j\mathbf{k}\cdot\mathbf{x}} = e^{jk\mathbf{x}\cdot\mathbf{d}}, \tag{4.5}$$

em que $\mathbf{d}$ é a direção da onda plana incidente. Para este caso, adotou-se $\mathbf{d} = \frac{1}{\sqrt{3}}(1,1,1)$ e $k = 6$.

Para esta formulação acoplada do problema, a estratégia mais apropriada é optar pelo MEF para resolver o problema em Ω_1 e pelo MEC em Ω_2, uma vez que o MEF é eficiente quando se trata de problemas interiores e é capaz de lidar com a não homogeneidade, enquanto o MEC resolve de forma muito mais eficiente problemas de domínio infinito, desde que este seja homogêneo. Sendo assim, em Ω_1, tem-se:

$$\nabla^2 u + n(\mathbf{x})^2 k^2 u = 0, \tag{4.6}$$

onde, para este exemplo, $n(\mathbf{x}) = 0,5$, sendo $n(\mathbf{x})$ um fator que modifica o número de onda no domínio interno. Mesmo que no caso estudado $n(\mathbf{x})$ seja um valor constante, é também possível que seja um fator cujo valor varie em função da posição. Por não ser o foco deste projeto, a solução utilizando o MEF não foi explorada de forma aprofundada. Sendo assim, considere apenas que, para Ω_1, tem-se:

$$\int_\Omega \nabla^2 u \cdot \nabla^2 v - k^2 \int_\Omega n^2 uv - \int_{d\Omega} v \frac{\partial u}{\partial n} = 0. \tag{4.7}$$

[6] Disponível em: <https://fenicsproject.org>.

[7] Disponível em: <https://nbviewer.jupyter.org/github/bempp/bempp-cl/blob/master/notebooks/helmholtz/simple_helmholtz_fem_bem_coupling.ipynb>.

A equação usando operadores é dada por

$$Au - k^2 Mu - M_\Gamma \frac{\partial u}{\partial n} = 0. \tag{4.8}$$

Já para Ω_2, a equação a ser resolvida é a equação de Helmholtz (2.27) conforme vista anteriormente. Além disso, para problemas de espalhamento, tem-se que $u = u^I + u^S$. Sendo assim, utilizando-se da formulação indireta do MEC, tem-se, para problemas de espalhamento (COLTON; KRESS, 2013):

$$u^S = \kappa u - \nu \frac{\partial u}{\partial n}. \tag{4.9}$$

Tomando o traço no contorno, obtém-se:

$$u^I = (\frac{1}{2} I - K)u + V \frac{\partial u}{\partial n}. \tag{4.10}$$

Sendo assim, unindo as equações 4.10 e 4.8, tem-se a formulação combinada, dada por

$$\begin{bmatrix} A - k^2 M & -M_\Gamma \\ \frac{1}{2} I - K & V \end{bmatrix} \begin{bmatrix} u \\ \frac{\partial u}{\partial n} \end{bmatrix} = \begin{bmatrix} 0 \\ u^I \end{bmatrix}. \tag{4.11}$$

Note que essa formulação não é estável para todas as frequências, uma vez que não utiliza nenhuma metodologia para lidar com o problema da não unicidade da solução, como o método de Burton-Miller. Entretanto, fora das frequências de ressonância do problema interno, a solução apresentada é suficiente e precisa.

Após a formulação do problema, a próxima etapa para a formulação numérica é a geração da malha. O FEniCS possui algumas malhas simples já embutidas em seu código. Sendo assim, foi utilizada a malha embutida do cubo unitário. É importante recordar que, por se tratar de uma geometria tri-dimensional em um programa de elementos finitos, a malha gerada também é tri-dimensional, conforme pode ser visto na Figura 54

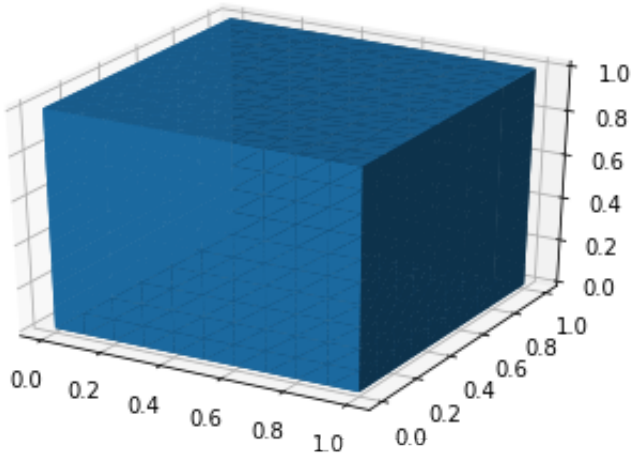


Figura 54: Malha do cubo tri-dimensional gerada pelo FEniCS

O Bempp possui, em seu núcleo, a capacidade de identificar, na malha tri-dimensional gerada pelo FEniCS, os elementos do contorno da malha de elementos finitos e, com isso, construir a malha que foi usada na formulação de elementos de contorno. Os detalhes da implementação numérica podem ser encontrados no *Jupyter Notebook* disponível *online* e no anexo B.5.

Após resolver o problema no contorno do cubo, a solução pode ser plotada em qualquer ponto do domínio exterior, utilizando-se a equação 4.9. A solução também pode ser avaliada no domínio interior por meio do resultado utilizando o MEF. Para a visualização da solução, criou-se um plano em $z = 0.5$, onde o resultado para a variável u^S foi plotado. Note que u^S é a variável que deve ser determinada, uma vez que u^I já é conhecida. Para se obter o campo total, u, basta se somar u^S e u^I. O gráfico de contorno para u^S pode ser visto na Figura 55.

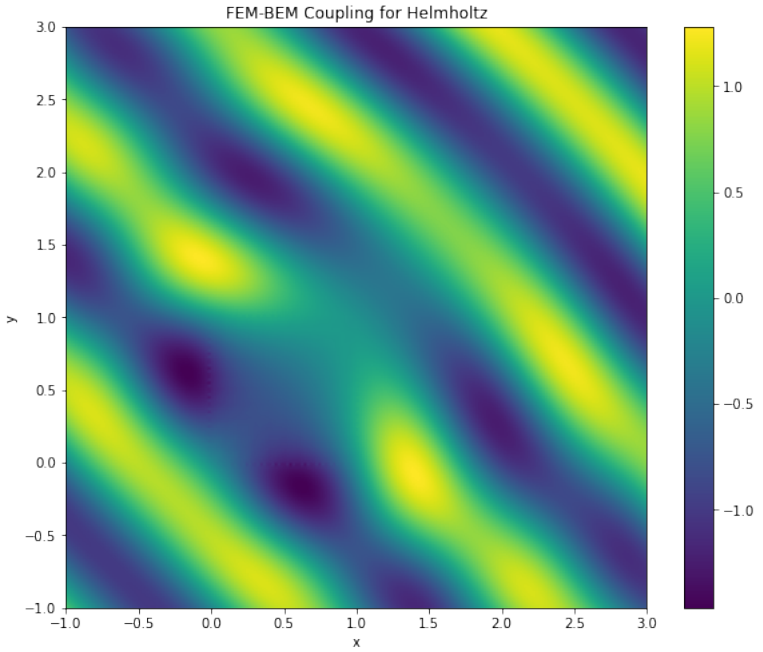


Figura 55: Solução do problema de espalhamento utilizando-se a formulação combinada

Capítulo 5

CONCLUSÃO E TRABALHOS FUTUROS

5.1 Conclusão

Este projeto apresenta a base teórica necessária para a compreensão do método dos elementos de contorno aplicado a problemas de acústica, governados pela equação de Helmholtz. A teoria acústica clássica foi abordada de uma maneira concisa e objetiva, uma vez que não é o foco deste projeto, mas é essencial para o seu desenvolvimento.

Além da base teórica tradicional relativa ao método dos elementos de contorno, foi abordada a formulação de Burton-Miller, como uma forma de resolver a não singularidade do resultado nas autofrequências para problemas exteriores. Devido à utilização do *software* Bempp, também foi explorada a formulação indireta do método dos elementos de contorno, tendo sido apresentados os operadores de contorno e os operadores potenciais relevantes.

Inicialmente foram feitas algumas simulações em um quadrado unitário utilizando o programa BB, que tiveram como maiores utilidades o ganho de familiaridade com o uso do programa e a verificação de se os resultados obtidos estavam coerentes. Os resultados foram mostrados tanto em gráficos de superfície quanto gráficos de contorno.

Após a resolução dos problemas no quadrado unitário, acrescentou-se mais uma dimensão espacial ao analisar problemas semelhantes em uma geometria cúbica. Tais problemas foram resolvidos utilizando-se tanto o Bempp quanto o BB. As respostas também foram comparadas com valores da solução analítica. O BB previu o comportamento dos modos com um erro quadrático médio (RMS) de 0,0061816 e as frequências ressonantes com um erro de 0,199 %, enquanto o Bempp-cl apresentou um erro RMS de 0,0157416 para os modos e 0,141 % para as frequências ressonantes esperadas. Assim, foi possível observar que os dois programas apresentaram uma precisão bastante satisfatória, sendo que o erro RMS ficou dentro de um limite aceitável e as frequências de ressonância foram identificadas com um erro inferior a 0,2%.

Outro fator decisivo para o sucesso de um programa de método numérico é o tempo de processamento. Sendo assim, foi realizada uma análise dos tempos de processamento para

ambos os *softwares* para o cubo em um problema cuja precisão já foi analisada. Em geral, o Bempp se mostrou como sendo mais veloz do que o BB, demonstrando que a utilização o *PyOpenCL* realmente faz bem sua função de acelerar a linguagem *Python*. Entretanto, isso não significa necessariamente que o BB não é competitivo, uma vez que não foi possível utilizar a paralelização do código para a análise neste projeto.

Em seguida, foi explorado um problema real de engenharia. O modelo do levitador acústico TinyLev foi resolvido utilizando-se o BB e o Bempp no contexto bi e tri-dimensional. Os resultados do Bempp foram condizentes com aqueles esperados pela comparação com a literatura e resultados experimentais. Contudo, o programa BB obteve, para o caso tri-dimensional, uma resposta incorreta, fruto de um erro de implementação. Mesmo com tal erro, existem evidências que a resposta do BB é condizente com a resposta obtida no Bempp.

Por fim, foi resolvido o problema de espalhamento no cubo unitário. Esse problema teve como principal objetivo demonstrar a capacidade de integração do MEC com outros métodos numéricos. A integração entre elementos finitos e elementos de contorno foi realizada com os *softwares* FEniCS e Bempp, respectivamente. Esse problema mostra como é possível combinar os dois métodos, de forma a se obter uma formulação otimizada que se beneficia das principais vantagens de cada um deles.

5.2 Trabalhos Futuros

Por mais que esse trabalho tenha contemplado uma boa gama de problemas utilizando o MEC, ainda existem diversos tópicos extremamente interessantes relacionados ao tema deste projeto que podem vir a ser explorados futuramente. Dentre eles:

- Realizar novamente o estudo dos tempos de processamento com a paralelização do BB funcionando;
- Resolver os problemas em relação aos métodos de aceleração para o BB e o Bempp e realizar um estudo comparativo dos tempos de processamento;
- Explorar problemas utilizando valores obtidos experimentalmente de pressão e fluxo, avaliando fatores como intensidade acústica;
- Realizar um estudo de resposta em frequência para um problema exterior, como por exemplo um problema de espalhamento, de forma a aplicar o método de Burton-Miller e verificar o efeito da aplicação do método na resposta obtida;
- Explorar mais problemas acústicos com dois ou mais meios com propriedades distintas, como por exemplo o problema do abafador automotivo que utiliza lã de vidro como estratégia de redução de ruído;
- Realizar uma comparação de tempos de processamento em problemas acústicos interiores e exteriores utilizando-se o MEC e o MEF, por meio do programa de código aberto FEniCS;

- Fazer uma análise do impacto do número de pontos de Gauss utilizados para as integrações numéricas em relação à precisão dos resultados e ao tempo de processamento para malhas com diferentes números de elementos usando o BB. A necessidade desse estudo surgiu da comparação dos tempos de processamento do BB com o Bempp, em que se notou que a escolha do número de pontos de Gauss possuía uma notória relação com os resultados de tempo de processamento e a precisão obtida.

- Desenvolver o BB de forma que ele deixe de ser apenas um programa e se torne um produto de programação, conforme definido por Brooks (1995). No estado presente, o BB é um programa, completo por si próprio, e pronto para ser executado pelo autor no sistema em que foi desenvolvido. Para que o BB seja convertido a um produto de programação, um objeto mais útil, entretanto mais custoso, é necessário que o programa passe por um processo de generalização, testes extensivos, documentação e manutenção contínua, de forma que possa ser executado, testado, reparado e estendido por qualquer pessoa.

Referências Bibliográficas

ABNT (Ed.). *Veículo rodoviário automotor - Ruído emitido na condição parado*. Rio de Janeiro: [s.n.], 2000.

ALBUQUERQUE Éder Lima de. *Introdução ao Método dos Elementos de Contorno*. Brasília, Brasil: Universidade de Brasília, 2020. Notas de aula.

ALI, A.; RAJAKUMAR, C. *The Bounday Element Method Applications in Sound and Vibration*. Leiden, Holanda: A.A.Balkema Publishers, 2005. ISBN 90 5809 657 2.

AMINI, S.; KIRKUP, M. Solution of helmholtz equation in the exterior domain by elementary boundary integral methods. *Journal of Computational Physics*, Universidade de Salford, Grande Manchester, Reino Unido, 1995.

BRAKHAGE, H.; WERNER, P. Über das dirichletsche aussenraumproblem für die helmholtzsche schwingungsgleichung. *Arch. Math.*, p. 325–329, 1965.

BRANCATI, A. *Boundary Element Method for Fast Solution of Acoustic Problems: Active and Passive Noise Control*. Tese (Doutorado) — Imperial College London, Londres, Inglaterra, 2010.

BREBBIA, C. A.; DOMINGUEZ, J. *Boundary Elements An Introductory Course*. 2. ed. Southampton: WIT Press/Computational Mechanics Publications, 1992.

BRIZON, C. J. da S. *Metodologia para avaliação e determinação de índices de conforto acústico em engenharia automobilística*. Tese (Doutorado) — Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil, 2012.

BROOKS, F. P. *The Mythical Man-Month*. Edição de aniversário. Carolina do Norte, Estados Unidos da América: Addison-Wesley, 1995.

BURTON, A. J.; MILLER, G. F. The application of integral equation methods to the numerical solution of some exterior bounday-value problems. *Proc. Roy. Soc. Lond.*, p. 201–210, 1971.

BUZOGANY, A. F. *Análise de problemas tridimensionais usando o método dos elementos de contorno com expansão em multipolos*. Brasília, Brasil: Universidade de Brasília, 2017.

CAMPOS, L. S. *Método dos elementos de contorno isogeométricos acelerado pela aproximação cruzada adaptativa*. Tese (Doutorado) — Universidade de Brasília, Brasília, DF, Brasil, 2016.

COLTON, D.; KRESS, R. *Integral Equation Methods in Scattering Theory*. [S.l.]: Classics in Applied Mathematics, 2013.

ENGLEDER, S.; STEINBACH, O. Stabilized boundary element methods for exterior helmholtz problems. *Numerische Mathematik*, p. 145–160, 2008.

FERREIRA Álvaro C. *Análise harmônica de cavidades acústicas pelo método dos elementos de contorno direto*. Dissertação (Mestrado) — Universidade de Brasília, Brasília, Brasil, 12 2004.

FERREIRA Álvaro C. *Comparação analítica numérica de cavidades acústicas e vibro-acústicas*. Brasília, Brasil: Universidade de Brasília, 2012.

FERREIRA Álvaro C. *Parallel $\mathscr{H}$ -Matrices accelerated isogeometric boundary element method implementation applied to acoustics internal and external problems*. Tese (Doutorado) — Universidade de Brasília, Brasília, DF, Brasil, 2020.

FERREIRA Álvaro C. et al. Experimental and h-matrices accelerated isogeometric boundary element method models of a tinylev inspired acoustic levitator. Universidade de Brasília, Brasília, Brasil, 2021.

FIGUEIREDO, W. H. S. *Elementos de contorno de Galerkin simétrico*. Brasília, Brasil: Universidade de Brasília, 2019.

FITZPATRICK, R. *Quantum Mechanics*. Austin, Texas: Institute for Fusion Studies, University of Texas at Austin, 2010. <http://farside.ph.utexas.edu/teaching/qmech/qmech.html>. Acesso em: 26 de out. 2020.

GILBERT, R. J. *Vibrations des structures - Interactions avec les fluides - Sources d'excitation aleatoires*. Paris, França: Ed. Eyrolles, 1988.

KATSIKADELIS, J. T. *Boundary elements: theory and applications*. Kidlington, Oxford, UK: Elsevier, 2002.

KINSLER, L. E. et al. *Fundamentals of acoustics*. 4. ed. United States: John Wiley & Sons, 2000.

KIRKUP, S. *The Bounday Element Method in Acoustics*. [S.l.]: Integrated Sound Software, 2007. ISBN 0 953 4031 06.

LEIGHTON, T. G. *The Acoustic Bubble*. London: Academis Press, 1994.

LIU, Y. *Fast Multipole Boundary Element Method*. New York: Cambridge University Press, 2009. ISBN 978-0-511-60504-8.

LUKASHIN, P. S.; STRIJHAK, S. V.; SHCHEGLOV, G. A. Validation of open-source bem++ code for simulation of acoustics problems. *Trudy ISP RAN/Proc*, p. 39–52, 2017.

MARBURG, S. The burton and miller method: Unlocking another mystery of its coupling parameter. *Journal of Computational Acoustics*, p. 20, 2016.

MARZO, A.; BARNES, A.; DRINKWATER, B. W. Tinylev: A multi-emitter single-axis acoustic levitator. *Review of Scientific Instruments*, 2017.

MORAIS, M. V. G. de. *Modelagens numéricas pelo método dos elementos finitos em problemas de iteração fluido-estrutura*. Dissertação (Mestrado) — Universidade de Brasília, Brasília, Brasil, 6 2000.

POTENTE, D. General design principles for an automotive muffler. *Australian Acoustical Society*, Busselton, Western Australia, 2005.

RIENSTRA, S. W.; HIRSCHBERG, A. *An Introduction to Acoustics*. Eindhoven University of Technology: [s.n.], 2016.

SMIGAJ, W. et al. Solving boundary integral problems with bem++. *ACM Trans. Math. Softw.*, v. 41, n. 2, p. 40, 2015.

TELLES, J. C. F. A self adptive co-ordinate transformation for efficient numerical evaluation of general boundary element integrals. *International Journal for Numerical Methods in Engineering*, p. 959–973, 1987.

THOMPSON, P. A. *Compressible-Fluid Dynamics*. New York: McGraw-Hill Book Company, 1972.

URSELL, F. On the exterior problems of acoustics. *Mathematical Proceedings of the Cambridge Philosophical Society*, p. 117–125, 1973.

WU, H.; YE, W.; JIANG, W. A collocation bem for 3d acoustic problems based on a non-singular burton-miller formulation with linear continuous elements. *Comput. Methods Appl. Mech. Engrg.*, p. 31, 2017.

Apêndice A

Códigos em Julia

A.1 Problemas Quadrado Simples

```
##################### Test case n ####################
### Problem description
### Analytical solution
### BEM model
### return error
using Plots
using LinearAlgebra
pyplot()
##################### Test case 1-4 ####################
### Acoustic tube
# Consider a square acoustic domain in which the speed of
#sound is approximately 344 [m/s] inside the domain
### Analytical solution
phi_closed(k,x) = cos.(k.*x./L)
q_closed(k,x) = -k.*sin(k.*x)
kclosed(L=1,n=1) = n*pi./(L)

### BEM model
function constclosed2D(ne=10,L=1,k=kclosed())
    #L = 1; # length of the square
    #k = 1; # wave number of the problem
    #The points and segments which describe this geometry are
    POINTS = [1 0 0
              2 L 0
              3 L L
              4 0 L];
```

```
27      SEGMENTS = [1 1 2 0
28                  2 2 3 0
29                  3 3 4 0
30                  4 4 1 0];
31      # Each segment will be meshed by ne elements
32      #ne = 100;
33      MESH = [1 ne
34              2 ne
35              3 ne
36              4 ne];
37      n = 100;
38      PONTOS_dom = zeros(n*n,3);
39      delta = 2*L/n; #distancia entre o primeiro ponto interno e a borda
40      passo = (L-2*delta)/(n-1);
41      iter = 0
42      for i in 1:n
43          for j in 1:n
44              iter += 1
45              PONTOS_dom[iter,:] = [iter delta+i*passo delta+j*passo]
46          end
47      end
48      fc = [0 0 0]
49      BCSeg = [1 1 0
50               2 1 0
51               3 1 0
52               4 0 1];
53      NOS_GEO,NOS,ELEM,CDC,normal =
        ↪  const2D.format_dad(POINTS,SEGMENTS,MESH,BCSeg)
54      nnos = size(NOS,1)  # Number of physical nodes, same as elements when
        ↪  using constant elements
55      info = [NOS_GEO,NOS,ELEM,CDC]
56      t = @elapsed phi, q, phi_dom, phi_dom =
        ↪  const2D.solve(info,PONTOS_dom,fc,BCSeg,k)
57      # Conventional method with approximated influence matrices
58      NOS_GEO,NOS,ELEM,CDC,normal =
        ↪  const2D.format_dad(POINTS,SEGMENTS,MESH,BCSeg)
59      nnos = size(NOS,1)  # Number of physical nodes, same as elements when
        ↪  using constant elements
60      return phi_dom,PONTOS_dom
61  end
62
63  ### Analytical solution
```

```
phi_cup(k,x) = sin.(k.*x)
q_cup(k,x) = -k.*cos(k.*x)
kcup(L=1,n=1) = (2*n-1)*pi./(2*L)
function cup2D(ne=10,L=1,k=1)
    #L = 1; # length of the square
    #k = 1; # wave number of the problem
    #The points and segments which describe this geometry are
    POINTS = [1 0 0
              2 L 0
              3 L L
              4 0 L];
    SEGMENTS = [1 1 2 0
                2 2 3 0
                3 3 4 0
                4 4 1 0];
    # Each segment will be meshed by ne elements
    #ne = 100;
    MESH = [1 ne
            2 ne
            3 ne
            4 ne];
    n = 100;
    PONTOS_dom = zeros(n*n,3);
    delta = 2*L/n; #distancia entre o primeiro ponto interno e a borda
    passo = (L-2*delta)/(n-1);
    iter = 0
    for i in 1:n
        for j in 1:n
            iter += 1
            PONTOS_dom[iter,:] = [iter delta+i*passo delta+j*passo]
        end
    end
    fc = [0 0 0]
    BCSeg = [1 1 0
             2 0 0
             3 1 0
             4 0 1];
    NOS_GEO,NOS,ELEM,CDC,normal =
    ↪ const2D.format_dad(POINTS,SEGMENTS,MESH,BCSeg)
    nnos = size(NOS,1)  # Number of physical nodes, same as elements when
    ↪ using constant elements
    info = [NOS_GEO,NOS,ELEM,CDC]
```

```
104     t = @elapsed phi, q, phi_dom, phi_dom =
        ↪  const2D.solve(info,PONTOS_dom,fc,BCSeg,k)
105     # Conventional method with approximated influence matrices
106     NOS_GEO,NOS,ELEM,CDC,normal =
        ↪  const2D.format_dad(POINTS,SEGMENTS,MESH,BCSeg)
107     nnos = size(NOS,1)  # Number of physical nodes, same as elements when
        ↪  using constant elements
108     return phi_dom,PONTOS_dom
109 end
110
111 ### Analytical solution
112 phi_open(k,x) = sin.(k.*x)
113 q_open(k,x) = -k.*cos(k.*x)
114 function open2D(ne=10,L=1,k=1)
115     #L = 1; # length of the square
116     #k = 1; # wave number of the problem
117     #The points and segments which describe this geometry are
118     POINTS = [1 0 0
119               2 L 0
120               3 L L
121               4 0 L];
122     SEGMENTS = [1 1 2 0
123                 2 2 3 0
124                 3 3 4 0
125                 4 4 1 0];
126     # Each segment will be meshed by ne elements
127     #ne = 100;
128     MESH = [1 ne
129             2 ne
130             3 ne
131             4 ne];
132     n = 100;
133     PONTOS_dom = zeros(n*n,3);
134     delta = 2*L/n; #distancia entre o primeiro ponto interno e a borda
135     passo = (L-2*delta)/(n-1);
136     iter = 0
137     for i in 1:n
138         for j in 1:n
139             iter += 1
140             PONTOS_dom[iter,:] = [iter delta+i*passo delta+j*passo]
141         end
142     end
```

```
    fc = [0 0 0]
    BCSeg = [1 1 0
             2 0 0
             3 1 0
             4 0 1];
    NOS_GEO,NOS,ELEM,CDC,normal =
    ↪ const2D.format_dad(POINTS,SEGMENTS,MESH,BCSeg)
    nnos = size(NOS,1)  # Number of physical nodes, same as elements when
    ↪ using constant elements
    info = [NOS_GEO,NOS,ELEM,CDC]
    t = @elapsed phi, q, phi_dom, phi_dom =
    ↪ const2D.solve(info,PONTOS_dom,fc,BCSeg,k)
    # Conventional method with approximated influence matrices
    NOS_GEO,NOS,ELEM,CDC,normal =
    ↪ const2D.format_dad(POINTS,SEGMENTS,MESH,BCSeg)
    nnos = size(NOS,1)  # Number of physical nodes, same as elements when
    ↪ using constant elements
    return phi_dom,PONTOS_dom
end
```

A.2 Modos Cubo Simples

```
# Boundary element method implementation for the Helmholtz and Laplace
#equations using constant  bidimensional elements
# Authors: Álvaro Campos Ferreira - alvaro.campos.ferreira@gmail.com,
# Fernando Barreto Soares - fernando.bsoares3@gmail.com
# Contains the dependencies for the linear constant element integration.
#The main function is const2D.solve() which builds the influence matrices,
#applies the boundary conditions, solves the linear system and returns the
#value of the potential and its gradient at boundary and domain points.
using PyCall, Plots, JLD
include("../BEM_base.jl")

### Analytical solution
L = 10; # Square length
mode = 2; # First mode number (Note: modes i and j = 0, excitation only on the
↪ "z" axis)
k_closed(n,L) = (n/2)*2*pi/L; # Resonance wavenumber, k = w/c, where c is the
↪ speed of propagation and w is the angular frequency. k_closed
k_cup(n,L) = (2*n-1)*pi/(2*L); #k_cup
```

```
17 phi_cup(k,x) = sin.(k.*x) #closed-open
18 phi_closed(k,x) = cos.(k.*x) #closed-closed BCs
19 phi_open(k,x) = sin.(k.*x)
20
21 CW = 343; # Speed of sound in m/s
22 inc = [0]; # There'll be no incident wave
23
24 # BEM modelling
25 # Gaussian quadrature - generation of points and weights [-1,1]
26 npg=6; # Number of integration points
27 qsi,w = const3D_tri.Gauss_Legendre(-1,1,npg) # Generation of the points and
   ↪ weights
28 # Python - mesh.io
29 meshio = pyimport("meshio")
30 # Build the domain points
31 L = 10; # Length of the cube
32 n_pint = 40;
33 PONTOS_int = zeros(n_pint,4);
34 delta = 0.1; # distance from both ends
35 passo = (L-2*delta)/(n_pint-1);
36 for i = 1:n_pint
37     PONTOS_int[i,:] = [i 0 0 delta+(i-1)*passo];
38 end
39 # Set the boundary conditions for each face. Faces 1 and 3 are perpendicular
   ↪ do "z" axis, 4 and 2 are perpendicular to "x" and 6 and 3 are
   ↪ perpendicular to "y".
40 #BCFace = [1. 1. 0.
41 #          2. 1. 0.
42 #          3. 0. 1.
43 #          4. 1. 0.
44 #          5. 1. 0.
45 #          6. 1. 0.]; #cup
46 #BCFace = [1. 0. -1.
47 #          2. 1. 0.
48 #          3. 0. 1.
49 #          4. 1. 0.
50 #          5. 1. 0.
51 #          6. 1. 0.];
52 BCFace = [1. 0. 0.
53           2. 1. 0.
54           3. 0. 1.
55           4. 1. 0.
```

```
          5. 1. 0.
          6. 1. 0.]; # open
#BCFace = [1. 1. -1.
#          2. 1. 0.
#          3. 1. 1.
#          4. 1. 0.
#          5. 1. 0.
#          6. 1. 0.];
#BCFace = [1. 1. 0.
#          2. 1. 0.
#          3. 1. 1.
#          4. 1. 0.
#          5. 1. 0.
#          6. 1. 0.];  #closed_closed
mshd = "./data/"
file = "cube_coarse.msh"
#t = [];
#Gnelem = [];
mesh = meshio.read(string(mshd,file))
NOS_GEO = [1:size(mesh.points,1) mesh.points]
nelem = size(mesh.cells_dict["triangle"],1)
ELEM = [1:nelem mesh.cells_dict["triangle"].+1
↪  mesh.cell_data["gmsh:geometrical"][1]]
CDC,NOS = const3D_tri.gera_vars(ELEM,BCFace,NOS_GEO)
println("Malha ",file,", nelem = ",nelem)
elemint = []
info = [NOS_GEO,ELEM,elemint,CDC]

#ks = [k_closed(1,L), k_closed(2,L), k_closed(3,L)]
#T_pintc = zeros(ComplexF64,n_pint,length(ks))
#T_analitic = zeros(n_pint,length(ks))

#for i = 1:length(ks)
#  k = ks[i]
#  Tc,qc,T_pint,qz = const3D_tri.solve(info,PONTOS_int,BCFace,k,true)
#  T_pintc[:,i] = T_pint
#  T_analitic[:,i] = phi_open(k,PONTOS_int[:,4])
#end

k = k_closed(mode,L)

Tc,qc,T_pint,qz = const3D_tri.solve(info,PONTOS_int,BCFace,k,true)
```

```
97
98  T_analitic = phi_open(k_closed(mode,L),PONTOS_int[:,4])
99
100 #errorms = zeros(length(ks))
101 #for i = 1:length(ks)
102 #  errorms[i] =
    ↪  sqrt(sum((real(T_pintc[:,i])/maximum(real(T_pintc[:,i]))-T_analitic[:,i]).^2)/n_pint)
    ↪  #pg.74 dissertação lucas campos
103 #end
104
105 #save(string("./data/assymetric_open.jld"),"T_pintc",T_pintc)
106 #save(string("./data/errorms_open.jld"),"errorms",errorms)
107
108 #TH,qH,T_pintH,qzH = const3D_tri.solveH(info,PONTOS_int,BCFace,k,true)
109
110 #Results plotting
111
112 pyplot()
113
114 #use the regular plot to obtain the pressure distribution inside the cube at a
    ↪  given frequency (Line Plot)
115 plot(PONTOS_int[:,4],real(T_pint)/maximum(real(T_pint)),xaxis=("z"),yaxis=("u"),title=(st
    ↪  numérica de u, n = ",mode)),label=("MEC"))
116 plot!(PONTOS_int[:,4],T_analitic,label=("Analítico"))
117 #plot(PONTOS_int[:,4],real(T_pintH)/maximum(real(T_pintH)),xaxis=("z"),yaxis=("u"),title=
    ↪  numérica de u, n = ",mode)),label=("MEC Hmatrix"))
```

A.3 FRF Cubo Simples

```
1  # Boundary element method implementation for the Helmholtz and Laplace
2  #equations using constant  bidimensional elements
3  # Authors: Álvaro Campos Ferreira - alvaro.campos.ferreira@gmail.com,
4  # Fernando Barreto Soares - fernando.bsoares3@gmail.com
5  # Contains the dependencies for the linear constant element integration.
6  #The main function is const2D.solve() which builds the influence matrices,
7  #applies the boundary conditions, solves the linear system and returns the
8  #value of the potential and its gradient at boundary and domain points.
9  using PyCall, Plots, JLD
10 include("../BEM_base.jl")
11
```

```
### Analytical solution
L = 10; # Square length
#n = 1; # First mode number (Note: modes i and j = 0, excitation only on the "z" axis)
phi_cup(k,x) = sin.(k.*x) #closed-open
phi_closed(k,x) = cos.(k.*x) #closed-closed BCs
q_closed(k,x) = k*sin.(k.*x)
k_closed(n,L) = (n/2)*2*pi/L; # Resonance wavenumber, k = w/c, where c is the speed of propagation and w is the angular frequency
k_res(n=1,L=1) = (2*n-1)*pi/(2*L) #k_cup

CW = 343; # Speed of sound in m/s
#k = k_res(n,L);
k1 = k_res(1,L); # Set the frequency in [rad/s]
k2 = k_res(2,L); # Set the frequency in [rad/s]
k3 = k_res(3,L); # Set the frequency in [rad/s]
#FR = k*CW;
inc = [0]; # There'll be no incident wave

# BEM modelling
# Gaussian quadrature - generation of points and weights [-1,1]
npg=6; # Number of integration points
qsi,w = const3D_tri.Gauss_Legendre(-1,1,npg) # Generation of the points and weights
# Python - mesh.io
meshio = pyimport("meshio")
# Build the domain points
L = 10; # Length of the cube
#n_pint = 40; # Number of domain points
n_pint = 5; # Number of domain points
PONTOS_int = zeros(n_pint,4);
delta = 0.1; # distance from both ends
passo = (L-2*delta)/(n_pint-1);
for i = 1:n_pint
    PONTOS_int[i,:] = [i 0 0 delta+(i-1)*passo];
end
# Set the boundary conditions for each face. Faces 1 and 3 are perpendicular do "z" axis, 4 and 2 are perpendicular to "x" and 6 and 5 are perpendicular to "y".
#BCFace = [1. 1. 0.
#          2. 1. 0.
#          3. 0. 1.
```

```
#           4. 1. 0.
#           5. 1. 0.
#           6. 1. 0.];
#BCFace = [1. 0. -1.
#           2. 1. 0.
#           3. 0. 1.
#           4. 1. 0.
#           5. 1. 0.
#           6. 1. 0.];
#BCFace = [1. 0. 0.
#           2. 1. 0.
#           3. 0. 1.
#           4. 1. 0.
#           5. 1. 0.
#           6. 1. 0.];
#BCFace = [1. 1. -1.
#           2. 1. 0.
#           3. 1. 1.
#           4. 1. 0.
#           5. 1. 0.
#           6. 1. 0.];
#BCFace = [1. 1. 0.
#           2. 1. 0.
#           3. 1. 1.
#           4. 1. 0.
#           5. 1. 0.
#           6. 1. 0.];
BCFace = [1. 0. 0.
          2. 1. 0.
          3. 1. 1.
          4. 1. 0.
          5. 1. 0.
          6. 1. 0.];
mshd = "./data/"
file = "cube_coarse.msh"
#t = [];
#Gnelem = [];
mesh = meshio.read(string(mshd,file))
NOS_GEO = [1:size(mesh.points,1) mesh.points]
nelem = size(mesh.cells_dict["triangle"],1)
ELEM = [1:nelem mesh.cells_dict["triangle"].+1
 ↪  mesh.cell_data["gmsh:geometrical"][3]]
```

```
CDC,NOS = const3D_tri.gera_vars(ELEM,BCFace,NOS_GEO)
println("Malha ",file,", nelem = ",nelem)
elemint = []
info = [NOS_GEO,ELEM,elemint,CDC]

pontos_freq = 400; # Number os frequency points
#pontos_freq = 20; # Number os frequency points
kFRFF = zeros(pontos_freq);
#intervalo_freq = k; # Set the frequency range
intervalo_freq = (k3-k1)+1.3*k1
passo_freq = intervalo_freq/pontos_freq;
kFRFF[1]=(k2+k_closed(2,L))/2-intervalo_freq/2; # First frequency value.
#kFRFF[1]=k-intervalo_freq/2;
for i = 2:pontos_freq
  kFRFF[i] = kFRFF[i-1]+passo_freq;
end

TcFRF =[]
T_pintcFRF = zeros(ComplexF64,n_pint,length(kFRFF));
for i = 1:length(kFRFF)
  kFRF = kFRFF[i]
  Tc,qc,T_pintc,qzc = const3D_tri.solve(info,PONTOS_int,BCFace,kFRF,true)
  append!(TcFRF,Tc)
  #append!(T_pintcFRF,T_pintc)
  T_pintcFRF[:,i] = T_pintc;
end
#TccH,qccH,T_pintccH,qzccH =
↪  const3D_tri.solveH(info,PONTOS_int,BCFace,k,true)

save(string("./data/FRF00.11.jld"),"TcFRF",TcFRF,"T_pintcFRF",T_pintcFRF)

# Results plotting

omega=kFRFF*CW;

pyplot()

#use the scatter plot to obtain the FRF
#scatter(omega,real(T_pintcFRF[1,:]),xlabel=("Frequência [rad/s]"),
↪  ylabel=("u"), marker=("o"), fillstyle=("none"),label=("z = 0.10"))
#scatter(omega,real(T_pintcFRF[2,:]),xlabel=("Frequência [rad/s]"),
↪  ylabel=("u"), marker=("o"), fillstyle=("none"),label=("z = 2.55"))
```

```
#scatter(omega,real(T_pintcFRF[3,:]),xlabel=("Frequência [rad/s]"),
↪  ylabel=("u"), marker=("o"), fillstyle=("none"),label=("z = 5.00"))
#scatter(omega,real(T_pintcFRF[4,:]),xlabel=("Frequência [rad/s]"),
↪  ylabel=("u"), marker=("o"), fillstyle=("none"),label=("z = 7.45"))
#scatter(omega,real(T_pintcFRF[5,:]),xlabel=("Frequência [rad/s]"),
↪  ylabel=("u"), marker=("o"), fillstyle=("none"),label=("z = 9.90"))
#For absolute value
internal_point = 2; #select desired internal point position on the PONTOS_int
↪  array for absolute FRF analysis
absoluteT_pintcFRF=zeros(length(kFRFF));
for i = 1:length(kFRFF)
  absoluteT_pintcFRF[i] = abs(T_pintcFRF[internal_point,i]);
end
scatter(omega,absoluteT_pintcFRF,xlabel=("Frequência [rad/s]"), ylabel=("u"),
↪  marker=("o"), fillstyle=("none"),label=("z = 2.55"))
vline!([k_res(1,10)*CW, k_res(2,10)*CW, k_res(3,10)*CW],label=("k_res"))
#vline!([k_closed(1,10)*CW, k_closed(2,10)*CW,
↪  k_closed(3,10)*CW],label=("k_res"))

#use the regular plot to obtain the pressure distribution inside the cube at a
↪  given frequency
#plot(PONTOS_int[:,4],real(T_pintcFRF[:,390]/maximum(real(T_pintcFRF[:,390]))),xaxis=("z
↪  numérica de u"),label=("MEC"))
#plot!(PONTOS_int[:,4],phi_closed(k_closed(3,10),PONTOS_int[:,4]),label=("Analítico"))
```

A.4 TinyLev 2D

```
function TinyLev2D(r=8.575*5,k=40000*2*pi/343000,l=10,h=40)
    #L = 1; # length of the square
    #k = 1; # wave number of the problem
    #The points and segments which describe this geometry are
    xc,yc = wavenurbs2D.calcula_centro(0,0,0,h,-r)
    POINTS =[1 0 0
                    2 -l 0
                    3 -l h
                    4 0 h
             5 2*xc 0
             6 2*xc h
             7 2*xc+l h
             8 2*xc+l 0
```

```
14              ];
15      SEGMENTS = [1 1 2 0
16                  2 2 3 0
17                  3 3 4 0
18                  4 4 1 r
19
20                  5 5 6 r
21                  6 6 7 0
22                  7 7 8 0
23                  8 8 5 0
24                  ];
25      BCSeg = [1 1 0 0
26               2 1 0 0
27               3 1 0 0
28               4 1 1 0
29               5 1 -1 0
30               6 1 0 0
31               7 1 0 0
32               8 1 0 0];
33      crv =  wavenurbs2D.format_dad_iso(POINTS,SEGMENTS)
34      n_pint = 50; # Number of domain points
35      PONTOS_dom = zeros(n_pint*n_pint,3);
36      iniciox = POINTS[1,2];
37      finalx = POINTS[5,2]
38      passox = (finalx-iniciox)/n_pint
39      inicioy = POINTS[1,3];
40      finaly = POINTS[4,3];
41      passoy = (finaly-inicioy)/n_pint
42      global iter = 1
43      for j = 1:n_pint
44          for i = 1:n_pint
45              PONTOS_dom[iter,:] = [iter iniciox+(i-1)*passox
                ↪  inicioy+(j-1)*passoy];
46              iter +=1
47          end
48      end
49      fc = [0 0 0]
50      info = [POINTS,SEGMENTS,BCSeg,k];
51      t = @elapsed phi,q,phi_dom = wavenurbs2D.solveH(info, PONTOS_dom, fc, k)
52      return phi_dom,iniciox,inicioy,passox,passoy
53  end
```

A.5 Estudo dos tempos de processamento

```
# Boundary element method implementation for the Helmholtz and Laplace
#equations using constant  bidimensional elements
# Author: Álvaro Campos Ferreira - alvaro.campos.ferreira@gmail.com
# Contains the dependencies for the linear constant element integration.
#The main function is const2D.solve() which builds the influence matrices,
#applies the boundary conditions, solves the linear system and returns the
#value of the potential and its gradient at boundary and domain points.
using PyCall, Plots, JLD
include("../BEM_base.jl")

#load results from cube_FRF.jl
#TcFRF,T_pintcFRF=load(string("./data/FRF00.01.jld"),"TcFRF","T_pintcFRF");

### Analytical solution
L = 10; # Square length
mode = 1; # First mode number (Note: modes i and j = 0, excitation only on the
↪  "z" axis)
k_closed(n,L) = (n/2)*2*pi/L; # Resonance wavenumber, k = w/c, where c is the
↪  speed of propagation and w is the angular frequency. k_closed
k_cup(n,L) = (2*n-1)*pi/(2*L); #k_cup
phi_cup(k,x) = sin.(k.*x) #closed-open
phi_closed(k,x) = cos.(k.*x) #closed-closed BCs
phi_open(k,x) = sin.(k.*x)

k = k_closed(mode,L)

CW = 343; # Speed of sound in m/s
inc = [0]; # There'll be no incident wave

# BEM modelling
# Gaussian quadrature - generation of points and weights [-1,1]
npg=6; # Number of integration points
qsi,w = const3D_tri.Gauss_Legendre(-1,1,npg) # Generation of the points and
↪  weights
# Python - mesh.io
meshio = pyimport("meshio")
# Build the domain points
n_pint = 40;
PONTOS_int = zeros(n_pint,4);
```

```
delta = 0.1; # distance from both ends
passo = (L-2*delta)/(n_pint-1);
for i = 1:n_pint
    PONTOS_int[i,:] = [i 0 0 delta+(i-1)*passo];
end
# Set the boundary conditions for each face. A cube has 6 faces
BCFace = [1. 1. 0.
          2. 1. 0.
          3. 1. 1.
          4. 1. 0.
          5. 1. 0.
          6. 1. 0.]; #closed
#mshd = "./data/"
#files = ["cube_coarsest.msh" "cube_coarsest.msh" "cube_coarse.msh"
↪  "cube_fine.msh" "cube_finest2.msh" "cube_finest3.msh"]
files = ["cube_coarsest.msh" "cube_finest3.msh"]

t = [];
th=[];
Gnelem = [];
#Res = [];
#Resh = [];

for i in files[1:2]
    mesh = meshio.read(i)
    NOS_GEO = [1:size(mesh.points,1) mesh.points]
    nelem = size(mesh.cells_dict["triangle"],1)
    ELEM = [1:nelem mesh.cells_dict["triangle"].+1
    ↪  mesh.cell_data["gmsh:geometrical"][1]]
    CDC,NOS = const3D_tri.gera_vars(ELEM,BCFace,NOS_GEO)
    println("Malha ",i,", nelem = ",nelem)
    elemint = []
    info = [NOS_GEO,ELEM,elemint,CDC]

    #Hmatrix
#    tsolve = @elapsed TH,qH,T_pintH,qzH =
↪  const3D_tri.solveH(info,PONTOS_int,BCFace,k,true)
#    println("tsolve = ",tsolve)

    #Conventional
    tsolvec = @elapsed Tc,qc,T_pint,qz =
    ↪  const3D_tri.solve(info,PONTOS_int,BCFace,k,true)
```

```
75      println("tsolvec = ",tsolvec)
76
77  #    global Resh = append!(Resh, [TH, T_pintH])
78  #    global Res = append!(Res, [Tc, T_pint])
79
80      global t = append!(t,tsolvec)
81  #    global th = append!(th,tsolve)
82      global Gnelem = append!(Gnelem,nelem)
83  end # files for
84
85  #save(string("./data/convergence_julia.jld"),"t",t,"th",th,"Gnelem",Gnelem)
86  #save("convergence_julia.jld","t",t,"Gnelem",Gnelem)
```

A.6 TinyLev 3D

```
1   # Boundary element method implementation for the Helmholtz and Laplace
2   #equations using constant  bidimensional elements
3   # Authors: Álvaro Campos Ferreira - alvaro.campos.ferreira@gmail.com,
4   # Fernando Barreto Soares - fernando.bsoares3@gmail.com
5   # Contains the dependencies for the linear constant element integration.
6   #The main function is const2D.solve() which builds the influence matrices,
7   #applies the boundary conditions, solves the linear system and returns the
8   #value of the potential and its gradient at boundary and domain points.
9   using JLD
10  using PyPlot
11  using PyCall
12  using Plots
13  @pyimport matplotlib.colors as col
14  @pyimport matplotlib.cm as cm
15  @pyimport mpl_toolkits.mplot3d as mp
16  @pyimport mpl_toolkits.mplot3d.art3d as ar
17  plt=PyPlot
18  plot=Plots
19
20  include("../BEM_base.jl")
21  include("../src/const3D_tri/mostra_resultados2.jl")
22
23  freq = 40e3
24  omega = 2 * pi * freq
25  CW = 344e3
```

```
26 k = omega/CW
27
28 inc = [0]; # There'll be no incident wave
29
30 # BEM modelling
31 # Gaussian quadrature - generation of points and weights [-1,1]
32 npg=6; # Number of integration points
33 qsi,w = const3D_tri.Gauss_Legendre(-1,1,npg) # Generation of the points and
    ↪ weights
34 # Python - mesh.io
35 meshio = pyimport("meshio")
36 # Build the domain points
37 L = 35; # Length of the cube
38 n_pint = 100;
39 PONTOS_int = zeros(n_pint*n_pint,4);
40 passo = L/(n_pint-1);
41 starty = -17
42 startz = 10
43 iter = 0;
44 for i in 1:n_pint
45     for j in 1:n_pint
46         global iter += 1
47         PONTOS_int[iter,:] = [iter 0 starty-passo+i*passo
            ↪ startz-passo+j*passo]
48     end
49 end
50
51 #BCFace = [1. 1. 0.
52 #          2. 1. 1.
53 #          3. 1. 0.
54 #          4. 1. -1.];
55
56 mshd = "./data/"
57 file = "T2L_coarse.msh"
58 #t = [];
59 #Gnelem = [];
60 mesh = meshio.read(string(mshd,file))
61 NOS_GEO = [1:size(mesh.points,1) mesh.points]
62 nelem = size(mesh.cells_dict["triangle"],1)
63 ELEM = [1:nelem mesh.cells_dict["triangle"].+1
    ↪ mesh.cell_data["gmsh:geometrical"][1]]
64
```

```
65  geo_to_physic = [mesh.cell_data["gmsh:physical"][1]
    ↪  mesh.cell_data["gmsh:geometrical"][1]]
66
67  BCFace = zeros(432,3) #There are 432 surfaces
68  for k = 1:432
69          indice = findfirst(x -> x==k, geo_to_physic[:,2])
70          physic_surf = geo_to_physic[indice,1]
71          if physic_surf == 1 || physic_surf == 3
72                  BCFace[k,:] = [k 1 0]
73          elseif physic_surf == 2
74                  BCFace[k,:] = [k 1 1]
75          elseif physic_surf == 4
76                  BCFace[k,:] = [k 1 -1]
77          end
78  end
79
80  CDC,NOS = const3D_tri.gera_vars(ELEM,BCFace,NOS_GEO)
81  println("Malha ",file,", nelem = ",nelem)
82  elemint = []
83  info = [NOS_GEO,ELEM,elemint,CDC]
84
85  #mostra_resultados2(NOS_GEO,ELEM,CDC[:,3])
86
87  Tc,qc,T_pint = const3D_tri.solve_ext(info,PONTOS_int,BCFace,k,true)
88
89  # Results plotting
90
91  pyplot()
92
93  using ColorSchemes
94
95  pontosy = zeros(n_pint,2);
96  for i = 1:n_pint
97      pontosy[i,:] = [i starty-passo+i*passo];
98  end
99  pontosz = zeros(n_pint,2);
100 for i = 1:n_pint
101     pontosz[i,:] = [i startz-passo+i*passo];
102 end
103
104 phi_real=real(T_pint);
105 phi_contour = zeros(n_pint,n_pint);
```

```
106 for j = 1:n_pint
107     index1=n_pint*(j-1)+1
108     index2=n_pint*j
109     phi_contour[:,j] = phi_real[index1:index2]
110 end
111 
112 solar = ColorSchemes.solar.colors
113 plot.contour(pontosy[:,2],pontosz[:,2],phi_contour/maximum(phi_contour),fill=true,colorba
    ↪  numérica TinyLev, plano
    ↪  y=0"),seriescolor=cgrad(ColorSchemes.viridis.colors),aspect_ratio=:equal,levels=20)
114 
115 plot.show()
116 
117 save(string("./data/tinylev3D3.jld"),"T_pint",T_pint,"phi_contour",phi_contour)
```

Apêndice B

Códigos em Python

B.1 Modo Cubo Simples

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import bempp.api
import numpy as np


# Problema 1 - correct closed

# In[2]:


L = 10 #cube length
n = 2 #mode
k = (n/2)*2*np.pi/L #k_closed


# In[3]:


#grid = bempp.api.import_grid('cube_coarse.msh')
#grid = bempp.api.shapes.cube()
grid = bempp.api.shapes.cube(length=10, origin=(0, 0, 0), h=1)
```

```
27
28
29  # In[ ]:
30
31
32  p1_space = bempp.api.function_space(grid, "P", 1)
33
34
35  # In[ ]:
36
37
38  identity = bempp.api.operators.boundary.sparse.identity(
39      p1_space, p1_space, p1_space)
40  adlp = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(
41      p1_space, p1_space, p1_space, k)
42  hyp = bempp.api.operators.boundary.helmholtz.hypersingular(
43      p1_space, p1_space, p1_space, k)
44
45
46  # In[ ]:
47
48
49  @bempp.api.complex_callable
50  def neumann_data(x, n, domain_index, res):
51      if domain_index == 3:
52          res[0] = 1
53      else:
54          res[0] = 0
55
56  neumann_fun = bempp.api.GridFunction(p1_space, fun=neumann_data)
57
58
59  # In[ ]:
60
61
62  lhs = hyp
63  rhs = (0.5 * identity - adlp) * neumann_fun
64
65
66  # In[ ]:
67
68
```

```
dirichlet_fun, info = bempp.api.linalg.gmres(lhs, rhs)


# In[ ]:


dirichlet_fun.plot()


# In[ ]:


n_grid_points = 40
plot_grid = np.mgrid[0:10:n_grid_points*1j, 0:10:n_grid_points*1j]
points = np.vstack((plot_grid[0].ravel(),
                    plot_grid[1].ravel(),
                    np.zeros(plot_grid[0].size)+5))


# In[ ]:


slp_pot = bempp.api.operators.potential.helmholtz.single_layer(
    p1_space, points, k)
dlp_pot = bempp.api.operators.potential.helmholtz.double_layer(
    p1_space, points, k)
u_evaluated = slp_pot * neumann_fun - dlp_pot * dirichlet_fun


# In[ ]:


# The next command ensures that plots are shown within the IPython notebook
get_ipython().run_line_magic('matplotlib', 'inline')

u_evaluated = u_evaluated.reshape((n_grid_points,n_grid_points))

# Plot the image
import matplotlib
matplotlib.rcParams['figure.figsize'] = (10.0, 8.0)

from matplotlib import pylab as plt
```

```
111
112 plt.imshow(np.real(u_evaluated.T)/np.max(np.real(u_evaluated.T)),
    ↪  extent=(0,10,0,10))
113 plt.title('Solução numérica de u, n=2, z=5')
114 plt.colorbar()
115
116
117 # In[ ]:
118
119
120 u_lineplot = np.real(u_evaluated.T[20,:])
121 u_lineplot = np.delete(u_lineplot, 0)
122 u_lineplot = np.delete(u_lineplot, 38)
123 max_lineplot_value = np.max(u_lineplot)
124 u_lineplot = u_lineplot/max_lineplot_value
125 x = np.arange(start=0, stop=L, step=L/np.size(u_lineplot))
126
127
128 # In[ ]:
129
130
131 matplotlib.rcParams['figure.figsize'] = (14.0, 8.0)
132 plt.figure()
133 plt.plot(x,np.flip(u_lineplot))
134 plt.title('Solução numérica de u, n=2, z=5')
135 plt.ylabel('u')
136 plt.show()
137
138
139 # In[ ]:
140
141
142 Tmode = np.flip(u_lineplot)
143 np.save('mode_correct_closed',Tmode) #save real part of T internal
```

B.2 FRF Cubo Simples

```
1 #!/usr/bin/env python
2 # coding: utf-8
3
```

```
# In[1]:


import bempp.api
import numpy as np


# Problema 1 - correct closed

# In[2]:


L = 10 #cube length
#k = (n/2)*2*np.pi/L #k_closed
n=1
k1 = (2*n-1)*np.pi/(2*L) #k_cup
n=2
k2 = (2*n-1)*np.pi/(2*L) #k_cup
n=3
k3= (2*n-1)*np.pi/(2*L) #k_cup

n=1
k_closed1 = (n/2)*2*np.pi/L #k_closed
n=2
k_closed2 = (n/2)*2*np.pi/L #k_closed
n=3
k_closed3= (n/2)*2*np.pi/L #k_closed

CW = 343 # Speed of sound in m/s


# In[3]:


pontos_freq = 400 # Number os frequency points
#pontos_freq = 20; # Number os frequency points
kFRFF = np.zeros(pontos_freq)
intervalo_freq = (k3-k1)+1.3*k1
passo_freq = intervalo_freq/pontos_freq
kFRFF[0]=(k2+((2/2)*2*np.pi/L))/2-intervalo_freq/2 # First frequency value.
for i in range(1,pontos_freq):
  kFRFF[i] = kFRFF[i-1]+passo_freq
```

```
46
47
48  # In[4]:
49
50
51  n_pint = 5
52  x = np.arange(start=0, stop=L, step=L/n_pint)
53  points = np.vstack((x,
54                      np.zeros(n_pint)+5,
55                      np.zeros(n_pint)+5))
56
57
58  # In[5]:
59
60
61  #grid = bempp.api.import_grid('cube_coarse.msh')
62  #grid = bempp.api.shapes.cube()
63  grid = bempp.api.shapes.cube(length=10, origin=(0, 0, 0), h=1)
64
65
66  # In[6]:
67
68
69  p1_space = bempp.api.function_space(grid, "P", 1)
70
71
72  # In[7]:
73
74
75  TpintcFRF = np.zeros((n_pint,np.size(kFRFF)), dtype=np.complex)
76
77  for i in range(0,np.size(kFRFF)):
78
79      k = kFRFF[i]
80
81      identity = bempp.api.operators.boundary.sparse.identity(
82          p1_space, p1_space, p1_space)
83      adlp = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(
84          p1_space, p1_space, p1_space, k)
85      hyp = bempp.api.operators.boundary.helmholtz.hypersingular(
86          p1_space, p1_space, p1_space, k)
87
```

```
88
89      @bempp.api.complex_callable
90      def neumann_data(x, n, domain_index, res):
91          if domain_index == 3:
92              res[0] = 1
93          else:
94              res[0] = 0
95
96      neumann_fun = bempp.api.GridFunction(p1_space, fun=neumann_data)
97
98      lhs = hyp
99      rhs = (0.5 * identity - adlp) * neumann_fun
100
101
102     dirichlet_fun, info = bempp.api.linalg.gmres(lhs, rhs)
103
104
105     slp_pot = bempp.api.operators.potential.helmholtz.single_layer(
106         p1_space, points, k)
107     dlp_pot = bempp.api.operators.potential.helmholtz.double_layer(
108         p1_space, points, k)
109     u_evaluated = slp_pot * neumann_fun - dlp_pot * dirichlet_fun
110
111     TpintcFRF[:,i] = u_evaluated
112
113
114 # In[8]:
115
116
117 np.save('FRF_correct_closed',TpintcFRF)
118
119
120 # In[9]:
121
122
123 internal_point = 2 #select desired internal point position on the PONTOS_int
    ↪  array for absolute FRF analysis
124 absoluteT_pintcFRF=np.zeros(np.size(kFRFF))
125 for i in range(0,np.size(kFRFF)):
126     absoluteT_pintcFRF[i] = abs(TpintcFRF[internal_point,i])
127
128
```

```
# In[10]:


# The next command ensures that plots are shown within the IPython notebook
get_ipython().run_line_magic('matplotlib', 'inline')

# Plot the image
import matplotlib

matplotlib.rcParams['figure.figsize'] = (14.0, 8.0)

from matplotlib import pylab as plt

omega=kFRFF*CW

plt.figure()
plt.scatter(omega,absoluteT_pintcFRF, marker=('o'),label='z = 2.55')
plt.vlines(np.array([k_closed1*CW, k_closed2*CW,
 ↪  k_closed3*CW]),np.min(absoluteT_pintcFRF)-0.1,np.max(absoluteT_pintcFRF)+0.1,label='k
plt.ylabel('u')
plt.xlabel('Frequência [rad/s]')
plt.ylim([-0.5,25])
plt.legend()
plt.savefig('FRF_correct_closed.png')
plt.show()
```

B.3 TinyLev 3D

```
# -*- coding: utf-8 -*-
"""TinyLev.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1Y7K-YonWb7V9rkufa95o4LNSx2PMwuej
"""

from google.colab import drive
drive.mount('/content/drive')

```

```
13 cd /content/drive/'My Drive'/'Colab Notebooks'
14
15 !pip install numpy scipy numba meshio>=4.0.16
16 !pip install plotly pyopencl gmsh==4.6.0
17
18 !git clone https://github.com/bempp/bempp-cl
19
20 cd bempp-cl
21
22 !python setup.py install
23
24 import bempp.api
25 import numpy as np
26
27 freq = 40e3
28 omega = 2 * np.pi * freq
29 CW = 344e3
30 k = omega/CW
31
32 grid = bempp.api.import_grid('T2L_coarse.msh')
33
34 p1_space = bempp.api.function_space(grid, "P", 1)
35 dp0_space = bempp.api.function_space(grid, "DP", 0)
36
37 identity = bempp.api.operators.boundary.sparse.identity(
38     dp0_space, dp0_space, p1_space)
39 adlp = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(
40     dp0_space, dp0_space, p1_space, k)
41 hyp = bempp.api.operators.boundary.helmholtz.hypersingular(
42     p1_space, dp0_space, p1_space, k)
43
44 @bempp.api.real_callable
45 def neumann_data(x, n, domain_index, res):
46     if domain_index == 1:
47         res[0] = 0
48     if domain_index == 2:
49         res[0] = 1
50     if domain_index == 3:
51         res[0] = 0
52     if domain_index == 4:
53         res[0] = -1
54
```

```
neumann_fun = bempp.api.GridFunction(dp0_space, fun=neumann_data,
↪  dual_space=p1_space)

lhs = -hyp
rhs = (0.5 * identity + adlp) * neumann_fun

dirichlet_fun, info = bempp.api.linalg.gmres(lhs, rhs)

n_grid_points = 100
plot_grid = np.mgrid[-17:18:n_grid_points*1j, 10:45:n_grid_points*1j]
points = np.vstack((np.zeros(plot_grid[0].size),
                    plot_grid[0].ravel(),
                    plot_grid[1].ravel()))

slp_pot = bempp.api.operators.potential.helmholtz.single_layer(
    dp0_space, points, k)
dlp_pot = bempp.api.operators.potential.helmholtz.double_layer(
    p1_space, points, k)
u_evaluated = -slp_pot * neumann_fun + dlp_pot * dirichlet_fun

# Commented out IPython magic to ensure Python compatibility.
# The next command ensures that plots are shown within the IPython notebook
# %matplotlib inline

u_evaluated = u_evaluated.reshape((n_grid_points,n_grid_points))

# Plot the image
import matplotlib
matplotlib.rcParams['figure.figsize'] = (10.0, 8.0)

from matplotlib import pylab as plt

plt.imshow(np.real(u_evaluated.T)/np.max(np.real(u_evaluated.T)),
↪  extent=(-17,18,10,45), aspect=('equal'))
plt.ylabel('z')
plt.xlabel('x')
plt.title('Solução numérica TinyLev, plano y=0')
plt.colorbar()

matplotlib.rcParams['figure.figsize'] = (10.0, 8.0)

plt.contourf(np.real(u_evaluated.T)/np.max(np.real(u_evaluated.T)),levels=20,extent=(-17,
```

```
plt.ylabel('z')
plt.xlabel('x')
plt.title('Solução numérica TinyLev, plano y=0')
plt.colorbar()

points.shape
```

B.4 Estudo tempos de processamento

```
# -*- coding: utf-8 -*-
"""Cube_convergence_Bempp.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/18BRw1rcj5_yHTJHgSNw7sTAJGpuNffLO
"""

from google.colab import drive
drive.mount('/content/drive')

!cat /proc/cpuinfo
!cat /proc/meminfo

cd /content/drive/'My Drive'/'Colab Notebooks'

!pip install numpy scipy numba meshio>=4.0.16
!pip install plotly pyopencl gmsh==4.6.0

cd bempp-cl

!python setup.py install

import bempp.api
import numpy as np
import time
import matplotlib.pyplot as plt

L = 10 #cube length
```

```
n = 1 #mode
k = (n/2)*2*np.pi/L #k_closed
#k = (2*n-1)*np.pi/(2*L) #k_cup

CW = 343

grid_files =
↪  ['cube_coarsest.msh','cube_coarsest.msh','cube_coarse.msh','cube_fine.msh','cube_fine
#if desired, use this first to certify everything is working fine:
#grid_files = ['cube_coarsest.msh','cube_coarse.msh']

n_pint = 40
x = np.arange(start=0, stop=L, step=L/n_pint)
points = np.vstack((np.zeros(n_pint),
                    np.zeros(n_pint),
                    x))

t = []
iterations=[]

for i in range(0,np.size(grid_files)):
    grid = bempp.api.import_grid(grid_files[i])

    print(grid_files[i])

    start = time.time()

    p1_space = bempp.api.function_space(grid, "P", 1)

    identity = bempp.api.operators.boundary.sparse.identity(
        p1_space, p1_space, p1_space)
    adlp = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(
        p1_space, p1_space, p1_space, k)
    hyp = bempp.api.operators.boundary.helmholtz.hypersingular(
        p1_space, p1_space, p1_space, k)

    @bempp.api.complex_callable
    def neumann_data(x, n, domain_index, res):
        if domain_index == 3:
            res[0] = 1
        else:
            res[0] = 0
```

```
72
73      neumann_fun = bempp.api.GridFunction(p1_space, fun=neumann_data)
74
75      lhs = hyp
76      rhs = (0.5 * identity - adlp) * neumann_fun
77
78      dirichlet_fun, info, iter_count = bempp.api.linalg.gmres(lhs, rhs,
        ↪  tol=1E-5, return_iteration_count=True)
79
80      slp_pot = bempp.api.operators.potential.helmholtz.single_layer(
81          p1_space, points, k)
82      dlp_pot = bempp.api.operators.potential.helmholtz.double_layer(
83          p1_space, points, k)
84      u_evaluated = slp_pot * neumann_fun - dlp_pot * dirichlet_fun
85
86      end = time.time()
87
88      t_elapsed = end - start
89      print('t = ', t_elapsed)
90      t.append(t_elapsed)
91      print('iterations = ', iter_count)
92      iterations.append(iter_count)
93
94  th = []
95  iterationsh = []
96
97  for i in range(0,np.size(grid_files)):
98      grid = bempp.api.import_grid(grid_files[i])
99
100     print(grid_files[i])
101
102     starth = time.time()
103
104     p1_space = bempp.api.function_space(grid, "P", 1)
105
106     identity = bempp.api.operators.boundary.sparse.identity(
107         p1_space, p1_space, p1_space)
108     adlp = bempp.api.operators.boundary.helmholtz.adjoint_double_layer(
109         p1_space, p1_space, p1_space, k, assembler='fmm')
110     hyp = bempp.api.operators.boundary.helmholtz.hypersingular(
111         p1_space, p1_space, p1_space, k, assembler='fmm')
112
```

```
    @bempp.api.complex_callable
    def neumann_data(x, n, domain_index, res):
        if domain_index == 3:
            res[0] = 1
        else:
            res[0] = 0

    neumann_fun = bempp.api.GridFunction(p1_space, fun=neumann_data)

    lhs = hyp
    rhs = (0.5 * identity - adlp) * neumann_fun

    dirichlet_fun, info, iter_count = bempp.api.linalg.gmres(lhs, rhs,
    ↪  tol=1E-5, return_iteration_count=True)

    slp_pot = bempp.api.operators.potential.helmholtz.single_layer(
        p1_space, points, k)
    dlp_pot = bempp.api.operators.potential.helmholtz.double_layer(
        p1_space, points, k)
    u_evaluated = slp_pot * neumann_fun - dlp_pot * dirichlet_fun

    endh = time.time()

    th_elapsed = endh - starth
    print('th = ', th_elapsed)
    th.append(th_elapsed)
    print('iterations = ', iter_count)
    iterationsh.append(iter_count)

np.savez('cube_convergence',t=t,th=th,iterations=iterations,iterationsh=iterationsh)
#load_results_convergence = np.load('cube_convergence.npz')
#ttest=load_results_convergence['t']
```

B.5 Problema do cubo unitário utilizando formulação combinada

```
# -*- coding: utf-8 -*-
"""Bempp_Fenics.ipynb

```

```
"""
Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1S2BiI0VKc1-9F5zndWyjUbovy0iN1G0T
"""

!apt-get install gmsh
!pip install pyopencl
!pip install meshio
!pip install bempp-cl

!add-apt-repository -y ppa:fenics-packages/fenics
!apt-get update
!apt-get install --no-install-recommends fenics

import dolfin
import bempp.api
import numpy as np

k = 6.
d = np.array([1., 1., 1])
d /= np.linalg.norm(d)

mesh = dolfin.UnitCubeMesh(10, 10, 10)

dolfin.plot(mesh)

from bempp.api.external import fenics

fenics_space = dolfin.FunctionSpace(mesh, "CG", 1)
trace_space, trace_matrix = \
    fenics.fenics_to_bempp_trace_data(fenics_space)
bempp_space = bempp.api.function_space(trace_space.grid, "DP", 0)

print("FEM dofs: {0}".format(mesh.num_vertices()))
print("BEM dofs: {0}".format(bempp_space.global_dof_count))

id_op = bempp.api.operators.boundary.sparse.identity(
    trace_space, bempp_space, bempp_space)
mass = bempp.api.operators.boundary.sparse.identity(
    bempp_space, bempp_space, trace_space)
```

```
dlp = bempp.api.operators.boundary.helmholtz.double_layer(
    trace_space, bempp_space, bempp_space, k)
slp = bempp.api.operators.boundary.helmholtz.single_layer(
    bempp_space, bempp_space, bempp_space, k)

u = dolfin.TrialFunction(fenics_space)
v = dolfin.TestFunction(fenics_space)
n = 0.5

@bempp.api.complex_callable
def u_inc(x, n, domain_index, result):
    result[0] = np.exp(1j * k * np.dot(x, d))
u_inc = bempp.api.GridFunction(bempp_space, fun=u_inc)

# The rhs from the FEM
rhs_fem = np.zeros(mesh.num_vertices())
# The rhs from the BEM
rhs_bem = u_inc.projections(bempp_space)
# The combined rhs
rhs = np.concatenate([rhs_fem, rhs_bem])

from bempp.api.assembly.blocked_operator import BlockedDiscreteOperator
from bempp.api.external.fenics import FenicsOperator
from scipy.sparse.linalg.interface import LinearOperator
blocks = [[None,None],[None,None]]

trace_op = LinearOperator(trace_matrix.shape, lambda x:trace_matrix @ x)

A = FenicsOperator((dolfin.inner(dolfin.nabla_grad(u),
                                 dolfin.nabla_grad(v)) \
    - k**2 * n**2 * u * v) * dolfin.dx)

blocks[0][0] = A.weak_form()
blocks[0][1] = -trace_matrix.T * mass.weak_form().A
blocks[1][0] = (.5 * id_op - dlp).weak_form() * trace_op
blocks[1][1] = slp.weak_form()

blocked = BlockedDiscreteOperator(np.array(blocks))

from bempp.api.assembly.discrete_boundary_operator import
↪  InverseSparseDiscreteBoundaryOperator
from scipy.sparse.linalg import LinearOperator
```

```
86
87  # Compute the sparse inverse of the Helmholtz operator
88  # Although it is not a boundary operator we can use
89  # the SparseInverseDiscreteBoundaryOperator function from
90  # BEM++ to turn its LU decomposition into a linear operator.
91  P1 = InverseSparseDiscreteBoundaryOperator(
92      blocked[0,0].A.tocsc())
93
94  # For the Laplace slp we use a simple mass matrix preconditioner.
95  # This is sufficient for smaller low-frequency problems.
96  P2 = InverseSparseDiscreteBoundaryOperator(
97      bempp.api.operators.boundary.sparse.identity(
98          bempp_space, bempp_space, bempp_space).weak_form())
99
100 # Create a block diagonal preconditioner object using the Scipy LinearOperator
    ↪  class
101 def apply_prec(x):
102     """Apply the block diagonal preconditioner"""
103     m1 = P1.shape[0]
104     m2 = P2.shape[0]
105     n1 = P1.shape[1]
106     n2 = P2.shape[1]
107
108     res1 = P1.dot(x[:n1])
109     res2 = P2.dot(x[n1:])
110     return np.concatenate([res1, res2])
111
112 p_shape = (P1.shape[0] + P2.shape[0], P1.shape[1] + P2.shape[1])
113 P = LinearOperator(p_shape, apply_prec, dtype=np.dtype('complex128'))
114
115 # Create a callback function to count the number of iterations
116 it_count = 0
117 def count_iterations(x):
118     global it_count
119     it_count += 1
120
121 from scipy.sparse.linalg import gmres
122 soln, info = gmres(blocked, rhs, M=P, callback=count_iterations)
123
124 soln_fem = soln[:mesh.num_vertices()]
125 soln_bem = soln[mesh.num_vertices():]
126
```

```
127 print("Number of iterations: {0}".format(it_count))
128
129 # Store the real part of the FEM solution
130 u = dolfin.Function(fenics_space)
131 u.vector()[:] = np.ascontiguousarray(np.real(soln_fem))
132
133 # Solution function with dirichlet data on the boundary
134 dirichlet_data = trace_matrix * soln_fem
135 dirichlet_fun = bempp.api.GridFunction(trace_space,
    ↪  coefficients=dirichlet_data)
136
137 # Solution function with Neumann data on the boundary
138 neumann_fun = bempp.api.GridFunction(bempp_space, coefficients=soln_bem)
139
140 # Commented out IPython magic to ensure Python compatibility.
141 # The next command ensures that plots are shown within the IPython notebook
142 # %matplotlib inline
143
144 Nx=200
145 Ny=200
146 xmin, xmax, ymin, ymax=[-1,3,-1,3]
147 plot_grid = np.mgrid[xmin:xmax:Nx*1j,ymin:ymax:Ny*1j]
148 points = np.vstack((plot_grid[0].ravel(),
149                     plot_grid[1].ravel(),
150                     np.array([0.5]*plot_grid[0].size)))
151 plot_me = np.zeros(points.shape[1], dtype=np.complex128)
152
153 x,y,z = points
154 bem_x = np.logical_not((x>0) * (x<1) * (y>0) * (y<1) * (z>0) * (z<1))
155
156 slp_pot= bempp.api.operators.potential.helmholtz.single_layer(
157     bempp_space, points[:, bem_x], k)
158 dlp_pot= bempp.api.operators.potential.helmholtz.double_layer(
159     trace_space, points[:, bem_x], k)
160
161 plot_me[bem_x] += np.exp(1j * k * (points[0, bem_x] * d[0] \
162                                    + points[1, bem_x] * d[1] \
163                                    + points[2, bem_x] * d[2]))
164 plot_me[bem_x] += dlp_pot.evaluate(dirichlet_fun).flat
165 plot_me[bem_x] -= slp_pot.evaluate(neumann_fun).flat
166
167 fem_points = points[:, np.logical_not(bem_x)].transpose()
```

```
fem_val = np.zeros(len(fem_points))
for p,point in enumerate(fem_points):
    result = np.zeros(1)
    u.eval(result, point)
    fem_val[p] = result[0]

plot_me[np.logical_not(bem_x)] += fem_val

plot_me = plot_me.reshape((Nx, Ny))

plot_me = plot_me.transpose()[::-1]

# Plot the image
from matplotlib import pyplot as plt
fig=plt.figure(figsize=(10, 8))
plt.imshow(np.real(plot_me), extent=[xmin, xmax, ymin, ymax])
plt.xlabel('x')
plt.ylabel('y')
plt.colorbar()
plt.title("FEM-BEM Coupling for Helmholtz")
plt.show()
```

Anexo A

Conceitos matemáticos preliminares

Algumas relações matemáticas básicas são apresentadas neste anexo. Elas são necessárias para se compreender as formulações do MEC que serão posteriormente apresentadas. Também foram apresentados alguns conceitos matemáticos essenciais para o desenvolvimento da teoria acústica.

A.1 O teorema de Gauss-Green

O teorema de Gauss-Green é uma identidade fundamental, em que se relaciona a integral da derivada de uma função sob o domínio Ω com a integral desta função no contorno Γ. O domínio pode ser tanto bi-dimensional quanto tri-dimensional.

Considere uma função $f = f(x, y)$, contínua no domínio. A integral da derivada dessa função pode ser escrcita da forma:

$$\int\int_{\Omega} \frac{\partial f(x,y)}{\partial x} dxdy = \int_{y_1}^{y_2} \left(\int_{x_1}^{x_2} \frac{\partial f(x,y)}{\partial x} \right) dy \tag{A.1}$$

Agora, da figura 56, é possível obter as seguintes relações:

$$\mathbf{t} = \frac{dx}{ds}\mathbf{i} - \frac{dy}{ds}\mathbf{j} \tag{A.2}$$

$$\mathbf{n} = \frac{dy}{ds}\mathbf{i} - \frac{dx}{ds}\mathbf{j} \tag{A.3}$$

onde $\mathbf{t}$, $\mathbf{n}$, $\mathbf{i}$ e $\mathbf{j}$ são os vetores unitários tangente, normal, na direção x e na direção y, respectivamente.

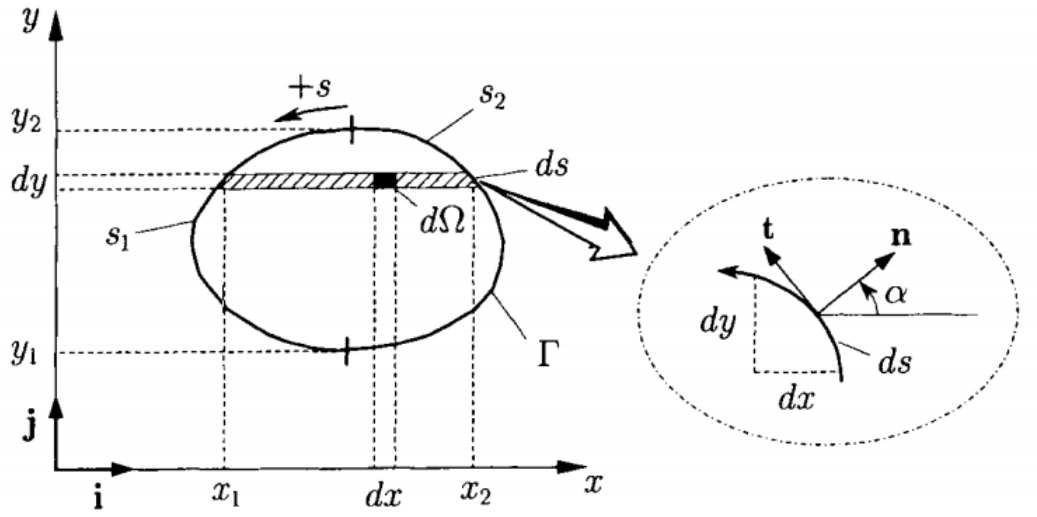


Figura 56: Integração sob um domínio plano Ω cercado por um contorno Γ (KATSIKADELIS, 2002)

Portanto, tem-se:

$$n_x = \frac{dy}{ds} \Rightarrow dy = n_x ds \tag{A.4}$$

$$n_y = -\frac{dx}{ds} \Rightarrow dx = -n_y ds \tag{A.5}$$

Substituindo a equação A.4 na integral do lado esquerdo de A.1, é possível obter o teorema de Gauss-Green, ou, por vezes, referido apenas como o teorema de Green:

$$\int\int_{\Omega} \frac{\partial f(x,y)}{\partial x} dxdy = \int_{\Gamma} f(s) n_x ds \tag{A.6}$$

O mesmo pode ser mostrado para a direção y. Também é possível demonstrar o mesmo teorema para um domínio tri-dimensional.

A partir do teorema de Green, pode-se também encontrar a relação chamada de *segunda identidade de Green*:

$$\int_{\Omega} (u\nabla^2 v - v\nabla^2 u) d\Omega = \int_{\Gamma} \left(u\frac{\partial v}{\partial n} - v\frac{\partial u}{\partial n} \right) ds \tag{A.7}$$

A.2 O Delta de Dirac

O Delta de Dirac pode ser definido como um pulso, ou uma perturbação que ocorre em um único ponto do espaço. Considerando que d é a distância do ponto de aplicação do Delta de Dirac à origem na direção x, tem-se:

$$\delta(x-d) = \begin{cases} 0 & \text{se } x \neq d \\ \infty & \text{se } x = d \end{cases} \tag{A.8}$$

Dado um intervalo $[a, b]$, assim como uma função qualquer $g = g(x)$, as seguintes propriedades do Delta de Dirac são definidas:

$$\int_a^b \delta(x-d) = \begin{cases} 0 & \text{se } d < a \\ 1 & \text{se } a \leq d \leq b \\ 0 & \text{se } d > b \end{cases} \tag{A.9}$$

e

$$\int_a^b g(x)\delta(x-d) = \begin{cases} 0 & \text{se } d < a \\ g(d) & \text{se } a \leq d \leq b \\ 0 & \text{se } d > b \end{cases} \tag{A.10}$$

A.3 Solução de equações diferenciais utilizando números complexos

A representação de fenômenos oscilatórios utilizando números complexos pode ser extremamente útil quando se trata de análises acústicas. Sendo assim, uma breve revisão foi feita para tratar do assunto.

Primeiramente, apresenta-se a identidade de Euler:

$$e^{j\theta} = cos\theta + jsen\theta \tag{A.11}$$

em que $j \equiv \sqrt{-1}$.

Agora, considere uma equação diferencial ordinária (EDO) linear da forma:

$$\frac{d^2x}{dt^2} + k^2x = 0 \tag{A.12}$$

Uma solução complexa é obtida se o deslocamento x for escrito da forma:

$$x = Ae^{jkt} \tag{A.13}$$

onde $A = aj + b$.

A parte real da solução complexa é, por si só, uma solução geral e completa da EDO original.

A notação complexa torna fácil a tarefa de determinar a velocidade e a aceleração complexas:

$$u = jkAe^{jkt} \tag{A.14}$$

$$a = -k^2Ae^{jkt} \tag{A.15}$$

As principais vantagens de se utilizar as quantidades complexas são, quando comparado com a solução trigonométrica, a maior simplicidade matemática e a maior facilidade com que se é possível relacionar as diversas variáveis acústicas e mecânicas. Contudo, deve-se ter cuidado para se obter a parte real da solução complexa para chegar na equação física correta (KINSLER et al., 2000).

A.4 Cálculo do erro para soluções obtidas numericamente

Quando se obtém uma solução numérica para um problema cuja solução analítica é conhecida, é conveniente utilizar de alguma métrica para determinar o erro entre as duas soluções.

Uma possível métrica para se determinar o erro em problemas numéricos é a raiz do valor quadrático médio (RMS) do erro normalizado (CAMPOS, 2016). Esse medidor é dado por:

$$e_{rms} = \frac{\sqrt{\frac{\sum_{t=1}^{n}(y_t - y_a)^2}{n}}}{y_{max} - y_{min}} \tag{A.16}$$

onde y_t é o valor da variável calculada numericamente, y_a é o valor da variável calculada analiticamente, n é o número de pontos em que y_t e y_a foram avaliadas e y_{max} e y_{min} são os valores máximos e mínimos da variável y_a, respectivamente.

Anexo B

Conceitos adicionais de acústica

B.1 A velocidade do som em fluidos

No estudo da acústica, a velocidade do som é uma constante utilizada muito frequentemente. Para um processo isotrópico, da equação de estado tem-se:

$$c^2 = \left(\frac{\partial P}{\partial \rho}\right)_s \tag{B.1}$$

Considerando-se um gás ideal, a seguinte relação é conhecida:

$$\left(\frac{\partial P}{\partial \rho}\right)_s = \gamma \frac{P_0}{\rho_0} \tag{B.2}$$

Sendo assim, combinando as equações B.1 e B.2:

$$c = \sqrt{\gamma \frac{P_0}{\rho_0}} \tag{B.3}$$

Utilizando a equação do gás ideal, tem-se que a razão P_0/ρ_0 é quase independente da pressão. Por substituição:

$$c = \sqrt{\gamma R T} \tag{B.4}$$

Para o som propagando no ar ($\gamma = 1,402$) a $T = 273,15K$ ($T = 0°C$), tem-se $c_0 = 331,5$ m/s. Este valor possui excelente concordância com dados experimentais, reforçando a hipótese de que processos acústicos em um fluido são adiabáticos (KINSLER et al., 2000). Outra velocidade frequentemente utilizada é $c = 343,4$ m/s, que corresponde à temperatura de 20 °C. A umidade relativa no ar possui baixa influência na velocidade do som (THOMPSON, 1972).

Já o cálculo da velocidade do som na água é mais complexo, e frequentemente são utilizadas fórmulas empíricas ou tabelas para se determinar esse valor. Além disso, a presença de sal ou de bolhas de ar na água afetam consideravelmente a velocidade do som (RIENSTRA;

HIRSCHBERG, 2016; LEIGHTON, 1994).

Uma vez que a água é menos compressível do que o ar, a velocidade de propagação do som é mais alta. Para efeito de comparação, a velocidade do som na água fresca a $20°C$ é igual a 1481 m/s (KINSLER et al., 2000).

B.2 Intensidade acústica e a escala decibel

A intensidade acústica instantânea $I(t)$ é definida como a taxa por unidade de área a qual um elemento de fluido realiza trabalho sobre o elemento adjacente. Ela é dada por $I(t) = pv[\mathrm{W/m^2}]$. A intensidade acústica I é a média temporal de $I(t)$, ou seja, a taxa de energia média transmitida através de uma área unitária normal à direção de propagação da onda.

$$I = \frac{1}{T}\int_0^T pv dt \tag{B.5}$$

Considerando-se uma onda plana viajando na direção positiva de x, utilizando a equação 2.19, tem-se que $p = \rho_0 c u$ e, portanto:

$$I = \frac{P^2}{2\rho_0 c} \tag{B.6}$$

Utilizando-se do conceito de impedância acústica específica, quantidade complexa definida por $z = p/u$, a equação B.6 pode ser reescrita da forma:

$$I = \frac{P^2}{2z} \tag{B.7}$$

Note que isso significa que, para ondas planas, $z = \rho_0 c$. A impedância acústica característica para o ar a 20°C é $z = 415[\mathrm{Pa \cdot s/m}]$ e para a água a 20°C é $z = 1,48 \cdot 10^6[\mathrm{Pa \cdot s/m}]$ (KINSLER et al., 2000).

Ainda é preciso definir a pressão efetiva, a qual, para ondas harmônicas, é dada por:

$$P_e = \frac{P}{\sqrt{2}} \tag{B.8}$$

e, consequentemente:

$$I = \frac{P_e^2}{\rho_0 c} \tag{B.9}$$

Seguido da definição de intensidade acústica, um nível sonoro nada mais é do que uma escala logarítmica da razão entre duas intensidades acústicas. A escala de nível sonora mais utilizada é a escala decibel (dB). Primeiramente, define-se nível de intensidade IL:

$$IL = 10 \log \left(\frac{I}{I_{ref}} \right) \tag{B.10}$$

onde I_{ref} é uma intensidade de referência e log é o logaritmo na base 10.

Utilizando-se da equação B.9, define-se o nível de pressão sonora SPL:

$$SPL = 20 \log \left(\frac{P_e}{P_{ref}} \right) \tag{B.11}$$

A referência padrão para sons no ar é $I_{ref} = 10^{-12}[\mathrm{W/m^2}]$, que é aproximadamente a intensidade a uma frequência de 1 kHz a qual um indivíduo com audição normal consegue ouvir um som com mínima clareza. Utilizando-se da equação B.9 tem-se a pressão efetiva de referência para o ar $P_{ref} = 20[\mu\mathrm{Pa}]$. Para água os valores padrões são $P_{ref} = 1[\mu\mathrm{Pa}]$ e $I_{ref} = 6,76^{-19}[\mathrm{W/m^2}]$.

Para ondas planas ou esféricas, IL e SPL são idênticos em valor. Entretanto, para campos sonoros mais complexos, esta afirmativa não é válida. O SPL é mais frequentemente utilizado, uma vez que é mensurável utilizando microfones.

Utilizando a equação de Helmholtz com perdas (consultar (KINSLER et al., 2000)), é possível demonstrar que a a perda de intensidade como função da distância para uma onda plana é [dB]:

$$IL(0) - IL(x) = 8,7\alpha x \tag{B.12}$$

Já para uma onda esférica é [dB]:

$$IL(0) - IL(r) = 20log(r) + 8,7\alpha r \tag{B.13}$$

onde $\alpha / f^2 = 1,37 \cdot 10^{-11}$ e f é a frequência, para o ar seco a 20°C.