MAC0499 - Trabalho de Formatura Supervisionado

Projeto Acústico Ótimo de Salas de Escuta*

Luciana Maria Gregolin Dias

Orientador: Prof. Dr. Marcelo Gomes de Queiroz

Departamento de Ciência da Computação - IME - USP

Resumo

O presente trabalho se refere a um projeto de iniciação científica desenvolvido durante a graduação, que compreendeu o estudo de técnicas de modelagem acústica de ambientes de escuta musical. O trabalho corresponde à simulação computacional do comportamento acústico de salas de escuta de geometria cubóide, com a presença de uma fonte sonora e um ouvinte. A partir deste estudo, foi produzido um programa que, através de métodos de otimização global, é capaz de obter as localizações mais adequadas para a fonte e o ouvinte, de modo a minimizar a distorção harmônica causada pelas interferências entre as ondas sonoras que atingem o ouvinte diretamente ou através de reflexões.

*Este trabalho foi financiado pela FAPESP - processo 04/01184-9

Conteúdo

Parte I

O Projeto de Iniciação Científica

Capítulo 1

Introdução

Projetos de salas de escuta são problemas práticos que envolvem uma variedade de áreas, como Acústica, Matemática, Computação e Música. No presente trabalho, foi tratada especificamente a modelagem de salas cubóides, onde são dados as dimensões da sala e os coeficientes de absorção de suas superfícies. A partir desses dados, o objetivo é determinar as posições de uma fonte sonora e de um ouvinte que minimizam as distorções do som no âmbito das freqüências.

Este trabalho também visou contribuir com as pesquisas referentes ao projeto temático *ACMUS*. Este último tem desenvolvido ferramentas computacionais para cálculo, análise e simulação acústica de salas destinadas à música, com a contribuição adicional de disponibilizar os códigos-fonte de todas as ferramentas implementadas.

Os estudos efetuados e os progressos obtidos neste trabalho estão detalhados nas páginas a seguir. As explicações estão organizadas de acordo com a ordem cronológica do projeto.

Dessa forma, primeiramente discorremos rapidamente sobre as principais técnicas da *Análise de Fourier*, fundamentais para o estudo de ondas sonoras. A partir desta teoria, é possível analisar muitas propriedades de um sinal sonoro.

Em seguida, há algumas explicações sobre *Acústica*, assunto muito relacionado com o projeto. Foi feito um estudo e implementação de um modelo de acústica geométrica considerado robusto para a simulação da *resposta de freqüência*[1] da sala, o modelo de *fontes*

[1] A resposta de freqüência de uma sala é uma função que descreve, para cada freqüência em Hertz, o fator de atenuação ou amplificação que ela sofreria naquela sala.

virtuais. A partir da simulação computacional desta função, foi possível quantificar a distorção sofrida por um sinal sonoro quando emitido em uma sala com dimensões e coeficientes de absorção conhecidos. Dessa forma, a ferramenta desenvolvida tornou possível determinar o comportamento acústico da sala para uma fonte sonora e um ouvinte em posições fixas.

Na seção sobre *Otimização Global*, estão detalhados os estudos sobre otimização contínua necessários para a realização de uma busca automática de localizações ótimas para a fonte e o ouvinte. Esta busca foi possível através da aplicação de métodos de otimização contínua em conjunto com a ferramenta de simulação da resposta de freqüência de salas produzido anteriormente. Ela é feita a partir de certas posições iniciais de fonte e ouvinte, cobrindo um número cada vez maior de localizações tentando tornar a resposta de freqüência da sala o mais próxima possível de uma resposta plana, isto é, ausente de distorções.

No capítulo seguinte, estão detalhadas as principais atividades realizadas durante a iniciação. Nesta parte, descrevemos os principais componentes do programa produzido para o projeto, bem como os testes realizados e resultados obtidos. Ela também inclui o atual andamento da integração do programa com o sistema do projeto ACMUS.

Capítulo 2

Conceitos Estudados

2.1 Análise de Fourier

2.1.1 Introdução

A *Análise de Fourier* é uma ferramenta fundamental no processamento de sinais musicais. Por isso, um estudo detalhado deste conjunto de técnicas é fundamental para o que segue.

Para estudar a microestrutura de um som (vocal, instrumental ou sintético), aplicamos técnicas de *análise espectral*. O *espectro* de um sinal sonoro é uma propriedade física que pode ser caracterizada como a distribuição da energia do sinal como função da freqüência. A análise espectral possui muitas aplicações. Quando feita em tempo real, por exemplo, serve como "escuta eletrônica" em sistemas musicais interativos, ajudando na identificação eletrônica de timbres e separação de sons produzidos por múltiplas fontes sonoras.

A Análise de Fourier é uma família de técnicas para a estimação espectral. Métodos baseados na teoria de Fourier modelam o som a ser analisado como uma soma de componentes senoidais harmonicamente relacionadas.

2.1.2 A Teoria de Fourier

Para a manipulação de ondas sonoras computacionalmente, costuma-se usar representações aproximadas de seus formatos através de funções matemáticas adequadas. Para sons periódicos, podemos usar o *Teorema de Fourier*, enunciado e verificado a seguir.

Teorema 1 (Teorema de Fourier) *Toda função $f : \mathbb{R} \to \mathbb{R}$ periódica de período T que satisfaz a certas hipóteses técnicas*[1] *pode ser representada através de sua* ***série de Fourier****, que é uma soma de componentes senoidais harmonicamente relacionadas, com diferentes amplitudes e fases:*

$$f(t) = \sum_{k=0}^{\infty} A_k \sin(k\omega t + \phi_k) \tag{2.1}$$

onde

A_k *é a amplitude da k-ésima componente senoidal de* $f(t)$

ω *é a freqüência fundamental de* $f(t)$ $\qquad (= 2\pi/T)$

ϕ_k *é o deslocamento de fase da k-ésima componente senoidal de* $f(t)$

A equação 2.1 é equivalente a

$$f(t) = \sum_{k=0}^{\infty} (a_k \cos k\omega t + b_k \sin k\omega t) \tag{2.2}$$

onde $a_k = A_k \sin \phi_k$ *e* $b_k = A_k \cos \phi_k$*, e podem ser calculadas pelas expressões*

$$a_k = \frac{2}{T} \int_0^T f(t) \cos k\omega t \mathrm{d}t,$$

$$b_k = \frac{2}{T} \int_0^T f(t) \sin k\omega t \mathrm{d}t.$$

Observe que

$$f(t) = a_0 \cos 0 + b_0 \sin 0 + a_1 \cos \omega t + b_1 \sin \omega t + \ldots + a_k \cos k\omega t + b_k \sin k\omega t + \ldots$$

é uma série trigonométrica em função de t (ω, a_k e b_k são constantes). Cada um dos termos dessa série tem a propriedade de repetir-se em intervalos de $T = 2\pi/\omega$:

[1] $f(\cdot)$ é integrável e absolutamente integrável, diferenciável a menos de um número finito de pontos no intervalo $[0, T]$ e para cada ponto z no qual $f(\cdot)$ é descontínua, vale que

$$f(z) = \frac{1}{2} \lim_{t \to z^-} f(t) + \frac{1}{2} \lim_{t \to z^+} f(t)$$

$\cos(k\omega(t+T)) = \cos(k\omega t + k2\pi) = \cos(k\omega t) \qquad \sin(k\omega(t+T)) = \sin(k\omega t + k2\pi) = \sin(k\omega t)$

Segue-se então que a soma $f(t)$ também deve ser periódica com período T. Substituindo ω por $\frac{2\pi}{T}$ na equação 2.2, temos que

$$f(t) = \sum_{k=0}^{\infty} \left(a_k \cos \frac{2\pi kt}{T} + b_k \sin \frac{2\pi kt}{T} \right) \tag{2.3}$$

é periódica de período T.

Suponhamos agora que a função periódica $f(t)$ de período T seja representada pela série trigonométrica acima (a validade desta hipótese é dada pelas hipóteses técnicas do teorema e não entraremos nestes detalhes). Usando as *relações de ortogonalidade* abaixo

$$\int_0^T \cos \tfrac{2\pi kt}{T} \sin \tfrac{2\pi nt}{T} dt = 0 \text{ se } n, k >= 1$$

$$\int_0^T \cos \tfrac{2\pi kt}{T} \cos \tfrac{2\pi nt}{T} dt = \begin{cases} T/2 & \text{se } k = n \\ 0 & \text{se } k \neq n \end{cases}$$

$$\int_0^T \sin \tfrac{2\pi kt}{T} \sin \tfrac{2\pi nt}{T} dt = \begin{cases} T/2 & \text{se } k = n \\ 0 & \text{se } k \neq n \end{cases}$$

podemos multiplicar a expressão 2.3 por $\sin \frac{2\pi kt}{T}$ e integrar de 0 a T, obtendo

$$\int_0^T f(t) \sin \frac{2\pi kt}{T} dt = b_k \frac{T}{2}.$$

Analogamente, fazendo o produto entre a expressão 2.3 e $\cos \frac{2\pi kt}{T}$ e integrando de 0 a T, obtemos

$$\int_0^T f(t) \cos \frac{2\pi kt}{T} dt = a_k \frac{T}{2}.$$

Destas expressões seguem as relações entre $f(t)$, a_k e b_k do enunciado do teorema.

Transformada de Fourier

Suponhamos agora que queremos analisar o sinal de tempo contínuo $f(t)$, de duração infinita (figura 2.1). A *Transformada de Fourier (Fourier Transform - FT)* é uma operação que mapeia um sinal de tempo contínuo por uma soma de infinitas séries de Fourier de

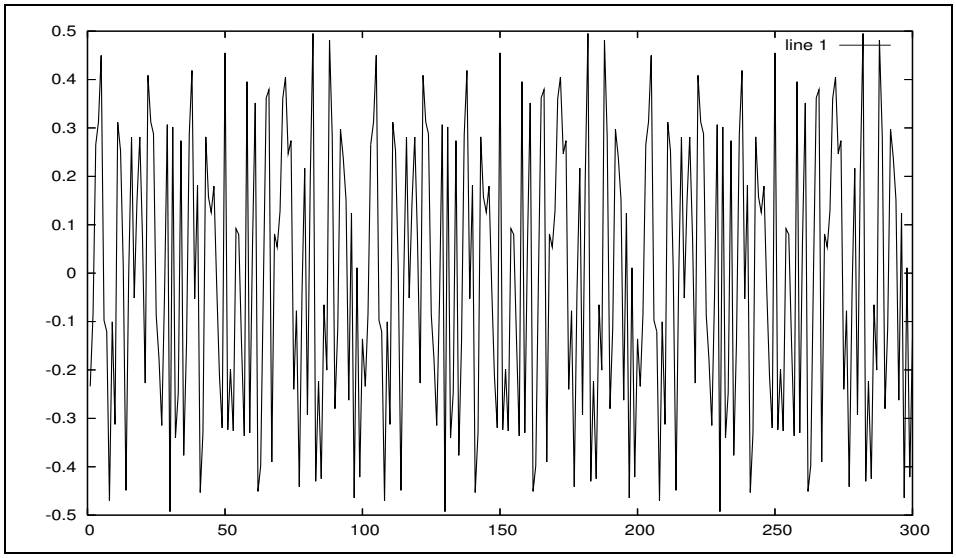

Figura 2.1: Representação de um sinal contínuo no domínio do tempo ([2])

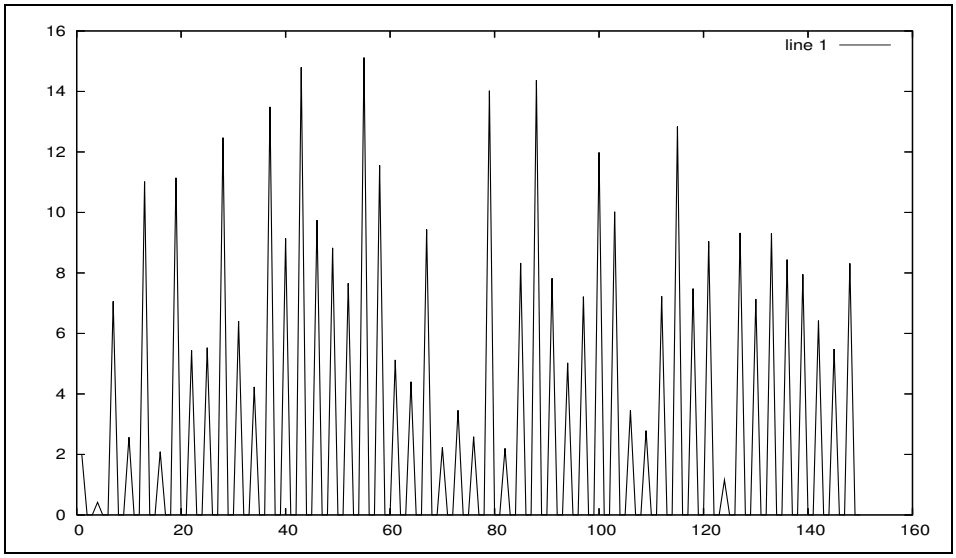

Figura 2.2: Espectro de magnitude do sinal de entrada da figura 2.1 ([2])

senoidais elementares, com fases e amplitudes diferentes. Ou seja, a FT converte o sinal de entrada em uma representação espectral correspondente.

O *espectro* da FT representa todas as freqüências de 0 a ∞ Hz com imagem refletida nas freqüências negativas (figura 2.2). Denotaremos esse espectro por $F(\omega)$, onde $\omega = 2\pi f$. A fórmula da FT é:

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt.$$

Cada valor de $F(\omega)$ é um número complexo da forma $a + bi$, de onde se pode extrair a magnitude ($\sqrt{a^2 + b^2}$) e a fase ($\arctan(b/a)$) da componente senoidal com freqüência f Hz.

Um aspecto digno de nota é que a FT divide a amplitude igualmente entre os componentes de freqüência positiva e negativa. Assim, a amplitude de todos os componentes é igual à metade de seu valor real.

Transformada de Fourier de Curta-duração (STFT)

A *Transformada de Fourier de Curta-duração (STFT)* é uma moldagem da FT para adaptar a teoria de Fourier ao mundo prático dos sinais finitos e não-periódicos. Recorremos assim à STFT para capturar as características do som ao longo do tempo. A STFT "quebra" o sinal de entrada em segmentos de curta-duração, isto é, impõe uma seqüência de "janelas de tempo" sobre o sinal de entrada. Esse processo é conhecido como *janelamento* (figura 2.3). Ele limita a duração dos sinais, para que uma FT digital possa lidar com sinais reais.

Uma *janela* é uma função matemática que é não-nula apenas em um intervalo de tempo limitado. A duração da janela é em geral de 1 ms a 1 s, sendo que quanto maior a janela, melhor a resolução de freqüência na análise. Analisando o espectro de cada segmento separadamente, obtemos uma seqüência de medidas que representam a variação do espectro no tempo. O janelamento tem o efeito colateral de distorcer a edição do espectro, pois o analisador mede não o sinal, mas o produto do sinal pela função da janela.

O primeiro passo da STFT é o janelamento do sinal, que acabamos de descrever. Em seguida, é aplicada a *Transformada de Fourier Discreta (DFT)* em cada segmento janelado. A DFT é uma FT de tempo discreto. Detalharemos a DFT e a FFT, que é uma implementação eficiente da DFT, nas seções seguintes. A DFT de um sinal janelado atua no produto *sinal X janela*, produzindo um espectro que é a *convolução* do espectro do sinal original com o

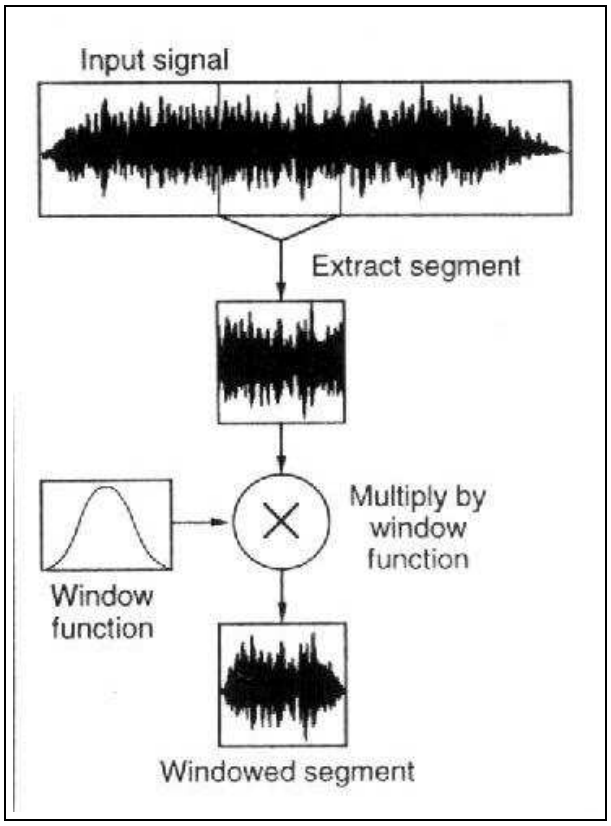

Figura 2.3: Janelamento de um sinal de entrada ([2])

da função da janela. A seção seguinte explica mais detalhadamente a teoria que envolve o processo de convolução.

Existem várias técnicas de janelamento. A mais simples delas consiste na seleção de uma parte do sinal de entrada $f(n)$ e atribuição de valores nulos para todo o resto do sinal. Outra função de janela, mais elaborada e muito conhecida, é a janela de Hamming, como podemos ver na figura 2.4.

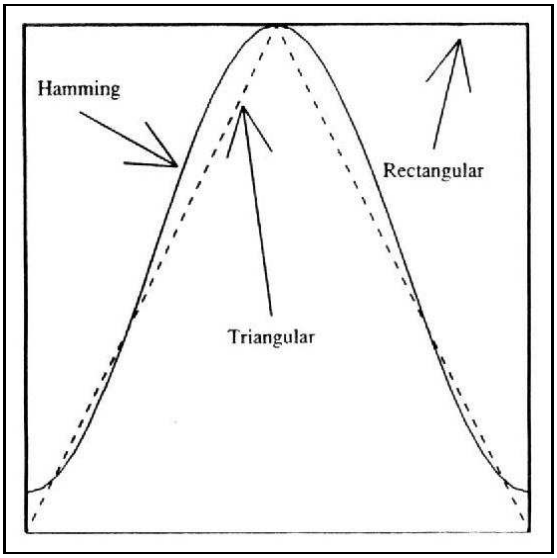

Figura 2.4: Janelas retangular, triangular e de Hamming ([1])

Em geral, as melhores janelas possuem espectros com maior concentração de energia no centro (próximo de 0 Hz) e menores "caudas" em seus extremos (maior taxa de decaimento). Em geral, as melhores janelas possuem espectros com maior concentração de energia na banda central, isto é, próximo de 0 Hz, e menos energia nas bandas laterais. A largura da banda central e a taxa de decaimento de energia nas bandas laterais têm relação direta com a qualidade do espectro: uma banda central muito larga ou bandas laterais muito altas causam muita distorção no espectro obtido pela STFT. Para estimação de freqüências específicas em sinais quase-periódicos é melhor escolher uma janela cujo espectro possua banda central estreita; para obter uma informação mais global do espectro, o melhor é garantir que as bandas laterais são baixas.

Toda análise espectral com janelamento é dificultada pelo *princípio de incerteza entre a*

resolução do tempo e freqüência. Se queremos alta resolução no domínio do tempo, precisamos de janelas pequenas, e com isso sacrificamos a resolução da freqüência (neste caso, poderíamos dizer que um evento ocorreu em um instante preciso, mas não poderíamos dizer exatamente quais frequências ele contém). Por outro lado, se queremos alta resolução no domínio da freqüência, precisamos de janelas grandes (neste caso, poderíamos estimar com alta precisão a freqüência de um evento, mas não saberíamos em que pontos da janela ela começou ou terminou).

Na figura 2.5, temos uma esquematização da STFT.

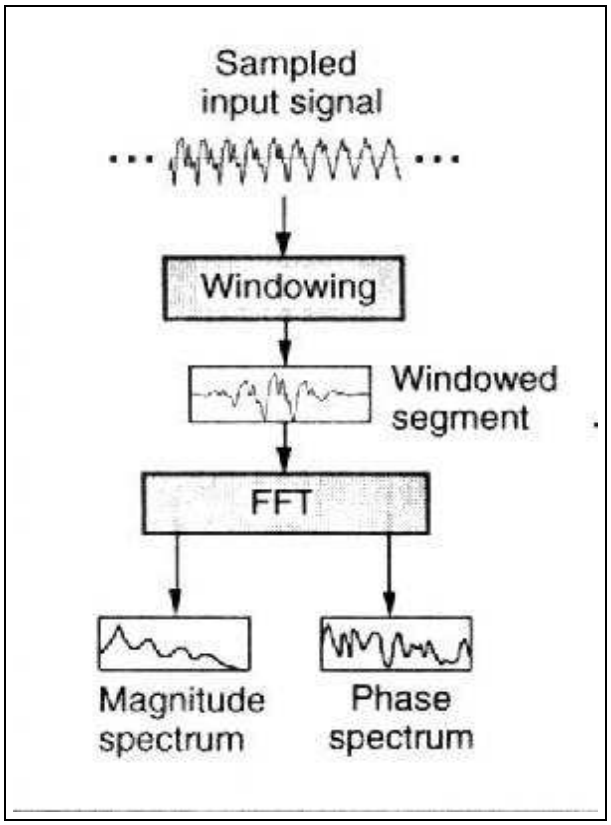

Figura 2.5: Esquematização da STFT ([2])

Transformada de Fourier Discreta (DFT)

Para aplicar os métodos da Análise de Fourier a sinais sonoros usando métodos computacionais, é necessário discretizá-los, pois computadores só são capazes de manipular informações

em conjuntos discretos e finitos de bits. Assim, quando digitalizamos o processo do cálculo da FT, em vez de trabalhar com uma função contínua $f(t)$, é necessário representar os sinais digitais como uma lista finita de valores discretos indexados por um índice amostral n.

Para o processo do cálculo da FT de sinais sonoros discretizados, temos um método conhecido como Transformada de Fourier Discreta (*Discrete Fourier Transform - DFT*). A *DFT* possibilita a obtenção de *amostras* do espectro contínuo da onda contínua, e trata as N amostras da onda sonora como um único período de uma onda infinitamente periódica. Assim, a DFT permite uma análise amostrada nas magnitudes e fases do espectro do sinal discretizado $f(n)$, determinando uma correspondência biunívoca entre o sinal de entrada e suas componentes senoidais.

Tendo N amostras de um sinal digital $f(n)$, a fórmula para o cálculo de sua DFT, para cada amostra k do espectro, é a seguinte:

$$F(k) = \sum_{n=0}^{N-1} f(n) e^{-i2\pi k \frac{n}{N}}, k = 0, \ldots, N-1. \tag{2.4}$$

O processo inverso, conhecido como *Transformada de Fourier Discreta Inversa (Inverse Discrete Fourier Transform - IDFT)*, é dado pela fórmula

$$f(n) = \frac{1}{N} \sum_{k=0}^{N-1} F(k) e^{i2\pi n \frac{k}{N}}, n = 0, \ldots, N-1. \tag{2.5}$$

As equações 2.4 e 2.5 são chamadas de *equaçao de análise* e *equação de síntese*, respectivamente. Através da IDFT, podemos reconstruir cada segmento janelado da onda a partir de seus componentes espectrais, para ressintetizar o sinal original no domínio do tempo.

Notemos que $F(k)$ é um conjunto de N números complexos, da forma $a + bi$. O valor da amplitude para cada amostra pode ser obtido através do cálculo do módulo dos números complexos obtidos. Notemos que, como a FT divide a amplitude igualmente entre os componentes positivos e negativos, a DFT atribuirá metade da amplitude para o componente negativo de freqüência e metade para o positivo. Assim, o espectro de magnitude da onda digitalizada é sempre simétrico ao redor de 0 Hz.

Podemos notar também que a DFT realiza $O(N^2)$ multiplicações complexas. Portanto, como em geral os processamentos computacionais de sinais sonoros envolvem grande número de amostras, mostrou-se necessária uma maneira mais rápida de fazer o cálculo da DFT de

uma função. Um método conhecido capaz de realizar este cálculo realizando menor número de multiplicações complexas do que a DFT é a Transformada Rápida de Fourier (*Fast Fourier Transform - FFT*), que executa $O(Nlog_2N)$ multiplicações complexas. Devido à eficiência deste método, ele foi estudado detalhadamente durante o projeto e posteriormente implementado para ser utilizado no programa produzido.

2.1.3 Convolução

A convolução é uma operação em processamento de sinais de áudio. Com ela podemos, por exemplo, fazer filtração de áudio e produzir efeitos digitais. Pode-se entender esta operação através da DFT/FFT quando duas ou mais ondas são combinadas.

Quando duas ondas são adicionadas, o espectro da onda resultante também será a soma dos dois espectros. O mesmo acontece quando multiplicamos a onda por uma constante. Assim, nesses casos, teríamos:

$$DFT\Big(af(n) + bg(n)\Big) = aF(k) + bG(k)$$

onde $f(n)$ e $g(n)$ são sinais discretizados no domínio do tempo, a e b são constantes e $F(k)$ e $G(k)$ representam as transformadas de Fourier de $f(n)$ e $g(n)$, respectivamente.

Já quando duas ondas são *multiplicadas*, o espectro resultante não é o produto dos dois espectros, e sim sua *convolução* (aqui, estamos representando a operação de convolução pelo símbolo $*$):

$$DFT\Big(f(n)g(n)\Big) = F(k) * G(k). \tag{2.6}$$

Sendo $f(n)$ e $g(n)$ duas seqüências de tamanhos N_f e N_g, a fórmula da convolução de $f(n)$ e $g(n)$ é:

$$h(n) = f(n) * g(n) = \sum_{k=0}^{N_f-1} f(k)g(n-k), \text{ para } n = 0, \ldots, N_g - 1.$$

Se $f(\cdot)$ e $g(\cdot)$ possuem valores nulos fora do intervalo 0 a $N_f - 1$ e 0 a $N_g - 1$, então $h(\cdot)$ terá tamanho $N_f + N_g - 1$, e a convolução é dita *linear*. Temos ainda que se $f(\cdot)$ e $g(\cdot)$ são periódicas de período N_f e N_g, a convolução resultante é periódica, e $h(\cdot)$ tem período $N_f + N_g - 1$.

Uma das propriedades da convolução é o fato de que a convolução de uma função qualquer $f(n)$ pela *função amostra unitária* $u(n)$ resulta na própria função:

$$f(n) * u(n) = f(n),$$

onde $u(n)$ é uma função tal que

$$u(n) = \begin{cases} 1 & \text{se } n = 0 \\ 0 & \text{se } n \neq 0. \end{cases}$$

Além disso, a convolução é comutativa:

$$f(n) * g(n) = g(n) * f(n).$$

Alguns exemplos do efeito da convolução: um sinal a contendo dois impulsos bem espaçados, quando convoluído com qualquer sinal b , resulta em duas instâncias do sinal, dando um efeito de eco. Em outra situação, se os dois impulsos de a estão bem próximos um dos outro, as repetições do sinal b sobrepõem-se, causando um efeito de reverberação (veja a figura 2.6).

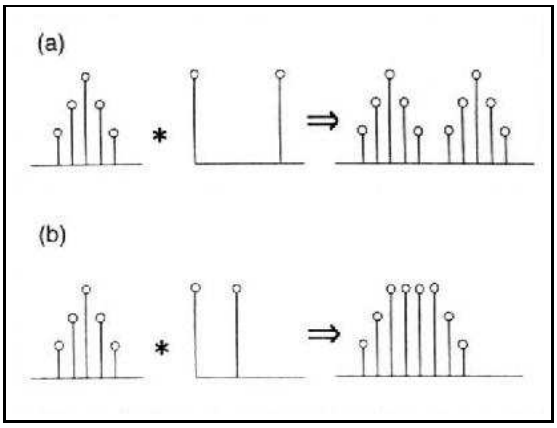

Figura 2.6: Efeitos da convolução no domínio do tempo ([2]). (a) Convolução com dois impulsos bem espaçados produz efeito de eco. (b) Convolução com dois impulsos muito próximos produz efeito de reverberação.

Aplicando esses conceitos, vamos verificar a propriedade 2.6 para o caso de dois sinais $f(n)$ e $g(n)$.

Sejam $f(n)$ e $g(n)$ duas seqüências periódicas de período N. Suas transformadas de Fourier são dadas por:

$$F(k) = \sum_{n=0}^{N-1} f(n) e^{-i2\pi k \frac{n}{N}}$$
$$G(k) = \sum_{n=0}^{N-1} g(n) e^{-i2\pi k \frac{n}{N}}.$$

Quando os espectros de f e g são multiplicados, deve acontecer uma operação de convolução entre f e g. Vamos verificar este fato.

Seja $h(n)$ a seqüência definida por $h(n) = f(n) * g(n)$, e seja $H(k)$ seu espectro. Pela equação de análise, sabemos que

$$H(k) = \sum_{n=0}^{N-1} h(n) e^{-i2\pi k \frac{n}{N}}.$$

Então:

$$\begin{aligned}
H(k) &= \sum_{n=0}^{N-1} h(n) e^{-i2\pi k \frac{n}{N}} \\
&= \sum_{n=0}^{N-1} (f(n) * g(n)) e^{-i2\pi k \frac{n}{N}} \\
&= \sum_{n=0}^{N-1} \sum_{j=0}^{N-1} f(j) g(n-j) e^{-i2\pi k \frac{n}{N}} \\
&= \sum_{n=0}^{N-1} \sum_{j=0}^{N-1} f(j) e^{-i2\pi k \frac{j}{N}} g(n-j) e^{-i2\pi k \frac{n-j}{N}} \\
&= \sum_{j=0}^{N-1} \Big(f(j) e^{-i2\pi k \frac{j}{N}} \sum_{z=-j}^{N-1-j} g(z) e^{-i2\pi k \frac{z}{N}} \Big) \\
&= \Big(\sum_{j=0}^{N-1} f(j) e^{-i2\pi k \frac{j}{N}} \Big) \cdot \Big(\sum_{z=0}^{N-1} g(z) e^{-i2\pi k \frac{z}{N}} \Big) \\
&= F(k) \cdot G(k)
\end{aligned}$$

2.1.4 Transformada Rápida de Fourier (*Fast Fourier Transform - FFT*)

A *FFT* tem como premissa que o número de amostras do sinal de entrada deve necessariamente ser uma potência de 2. Segue uma explicação resumida da idéia central do algoritmo. Primeiramente, a seqüência de entrada é dividida em duas seqüências menores, de tamanho $N/2$, onde uma é formada pelas amostras de índice ímpar e outra pelas de índice par. A DFT dessas duas seqüências pode ser combinada para obter a DFT da seqüência original. Isso reduz o número de multiplicações complexas à metade. Pode-se repetir recursivamente essa bifurcação, até que restem apenas duas amostras, cujo cálculo da DFT é trivial. Como são $log_2 N$ bifurcações, o número total de multiplicações complexas é $O(N log_2 N)$.

Com base no livro [3], foi analisado passo a passo o algoritmo da FFT, aplicado a um sinal de entrada $f[n]$ contendo N amostras, onde N deve ser uma potência de 2. Ao final do algoritmo, o espectro do sinal de entrada estará em F Descrevemos o algoritmo a seguir (a função *invertebit(j)* inverte a ordem dos r bits da representação binária de j).

[0] $\sigma \leftarrow log_2 N$, $i \leftarrow 1$, $s \leftarrow N/2$, $W \leftarrow e^{-2\pi i/N}$

[1] Para n de 0 a $N-1$, faça $F[n] \leftarrow f[n] + 0.0i$

[2] Enquanto $i \leq \sigma$ faça:

[2.1] $j \leftarrow 0$; $p \leftarrow invertebit(j)$

[2.2] Enquanto $(j \cdot s < N)$ faça:

[2.2.1] $k \leftarrow (j \cdot s)$

[2.2.2] Enquanto $(k < (j+1) \cdot j$ faça:

[A] $aux \leftarrow W^p \cdot F[k+s]$

[B] $F[k+s] \leftarrow F[k] - aux$

[C] $F[k] \leftarrow F[k] + aux$

[D] $k \leftarrow k + 1$

[2.2.3] $j \leftarrow j + 2$

[2.3] $s \leftarrow s/2$;

[2.4] $i \leftarrow i + 1$

[3] Inversão final de bits.

[3.1] $j \leftarrow 0$.

[3.2] Enquanto $j < N$ faça:

[3.2.1] $r \leftarrow invertebit(j)$

[3.2.2] Se $j < r$, troque os conteúdos dos elementos $F[j]$ e $F[r]$ do vetor.

[3.2.3] $j \leftarrow j + 1$

No algoritmo acima, temos que:

- A variável σ representa o número de estágios do algoritmo.
- As operações centrais do algoritmo (passos B e C) são efetuadas entre elementos cujos índices têm distância de s. Esta variável é dividida por 2 a cada passo.

O algoritmo foi implementado na linguagem Java, e o programa produzido recebe N amostras de um sinal de entrada e produz N números complexos, amostras do espectro contínuo do sinal de entrada.

2.2 Simulação Acústica Computacional

2.2.1 Visão Geral

Uma *sala de escuta* é um ambiente que pode ser caracterizado por possuir superfícies com diferentes características de reflexão e absorção do som, sendo portando um recinto *reverberante*. Quando uma fonte sonora é situada em um ambiente como esse, o volume de ar encerrado entre as superfícies limites da sala é excitado não só pelas ondas sonoras provenientes diretamente da fonte, mas também pelas ondas que são refletidas pelas superfícies. Veja na figura 2.7 uma representação de uma sala de escuta cubóide no espaço tridimensional.

Durante muito tempo, a forma mais usada para a modelagem acústica de ambientes de escuta musical foi a execução de medições no próprio local, ou a construção de réplicas de dimensões proporcionais às do ambiente original. A utilização de réplicas é viável quando não se pretende fazer modificações na sala de escuta, ou quando o número de modificações é

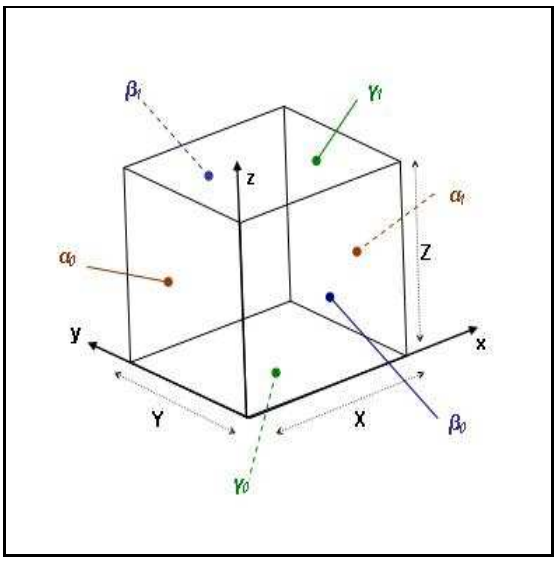

Figura 2.7: Modelo de uma sala de escuta de geometria cubóide

pequeno. No entanto, ao considerar o problema de *projeto* de salas, observa-se a dificuldade em utilizar medições, pois durante a fase de projeto se deseja testar uma enorme quantidade de combinações dos materiais e de posições de fontes ou ouvintes, entre outros parâmetros. Atualmente, a aparição de novos modelos geométricos para a simulação de campos sonoros tem permitido o desenvolvimento de programas que facilitam uma aproximação do comportamento acústico de salas.

Para simular computacionalmente o comportamento acústico de uma sala de escuta, é necessário executar simulações de sua *resposta de freqüência* a partir de suas dimensões e dos coeficientes de absorção de suas superfícies, simulações estas imprescindíveis para uma busca automática de localizações ótimas para fontes sonoras e ouvinte em uma sala de escuta. Neste contexto, interpretamos a resposta de freqüência de uma sala como sendo a quantificação da atenuação ou amplificação sofrida por cada freqüência para uma certa disposição de fonte sonora e ouvinte. Essas diferenças no comportamento de cada freqüência são conseqüência da ressonância do recinto, das diferentes velocidades de amortecimento dos sinais e dos modos normais de vibração.

Dois dos principais modelos geométricos teóricos empregados para a simulação acústica computacional são o *modelo de traçado de raios* e o *modelo de fontes virtuais*. Ambos baseiam-

se na teoria da acústica geométrica e tratam as ondas sonoras como raios que incidem e são refletidos nas superfícies do recinto, pretendendo conhecer o comportamento do campo sonoro no local. Para o presente trabalho, o modelo considerado mais adequado para implementação foi o modelo de fontes virtuais, visto que é o mais eficiente para salas de geometria cubóide.

2.2.2 O Modelo de Fontes Virtuais

Como o próprio nome indica, este modelo está baseado na teoria geométrica e ótica, e só pode ser aplicado a ambientes de superfícies planas e com modelos de reflexões especulares, que é o caso dos ambientes tratados neste projeto. Em comparação com a solução da equação de onda, que corresponde à solução de uma equação diferencial por elementos finitos em um número muito grande de pontos da sala, este modelo geométrico tem como principal vantagem a sua enorme eficiência computacional, visto que toda informação processada diz respeito à localização do ouvinte e às reflexões que atingem aquele ponto. Este modelo geométrico é bastante acurado para altas freqüências, mas sua principal desvantagem é fornecer uma pior aproximação em baixas freqüências.

O modelo das fontes virtuais, como mencionado anteriormente, trata a onda sonora como um raio, e a reflexão da onda nas superfícies do ambiente é considerada como sendo especular. Isto significa que cada reflexão do som em uma parede é tratada como se tivesse sido gerada por uma fonte sonora "virtual" que é imagem simétrica da fonte principal (veja a figura 2.8) .

Na modelagem para o problema em questão, cada fonte virtual está localizada em uma sala virtual, identificada por um índice (i, j, k), que indica a distância da sala virtual em relação à sala real em número de salas, sobre um eixo tridimensional. No caso, a variável i representa a distância na direção do eixo x, a variável j representa a distância na direção do eixo y, a variável k representa a distância na direção do eixo z. Por exemplo, a sala virtual imediatamente à direita da sala real na direção do eixo x possui índice $(1, 0, 0)$.

Uma fonte sonora virtual pode, por sua vez, ter outra imagem em outra parede, e assim sucessivamente, do mesmo modo que um espelho frente a outro produziriam uma cadeia de imagens infinita.

A figura 2.9 mostra uma representação de algumas salas virtuais no plano (x, y). Para efeito de cálculo computacional, é tomado um número limitado de salas virtuais - apenas as

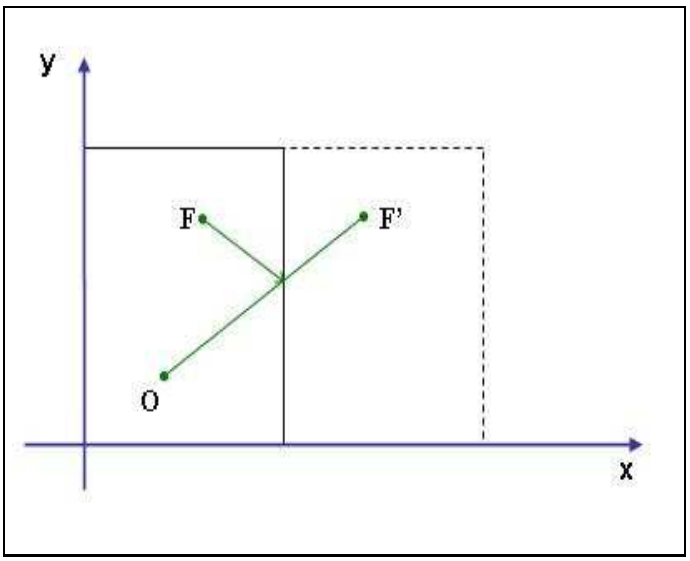

Figura 2.8: Representação de uma reflexão no plano

situadas a uma distância menor do que um certo raio máximo da sala real. Essa distância foi tomada como sendo a somatória das componentes dos índices (i, j, k), isto é, a soma $|i| + |j| + |k|$.

Segundo o modelo de fontes virtuais, o sinal da fonte sonora refletido por uma parede é idêntico ao de uma onda direta criada por uma fonte virtual situada do outro lado da parede. Ou seja, de acordo com este modelo, as fontes virtuais irradiam ondas sonoras sincronizadas com a fonte principal (real). Seguindo o mesmo processo para todas as fontes virtuais nas salas virtuais adjacentes, podemos encontrar todas as reflexões de primeira ordem que passam por um determinado ponto do ambiente. Esse processo se repete para reflexões de segunda ordem, terceira, etc. Inicialmente, este método é mais indicado pra salas de geometrias simples, como as de forma cubóide, já que para geometrias mais complexas é muito difícil identificar as posições das fontes virtuais.

Veremos na seção seguinte que, para os cálculos necessários para a simulação, que corresponderão aos cálculos da *resposta impulsiva* da sala, deve-se considerar também a absorção das paredes, que diminui o valor da intensidade sonora, bem como a absorção do ar.

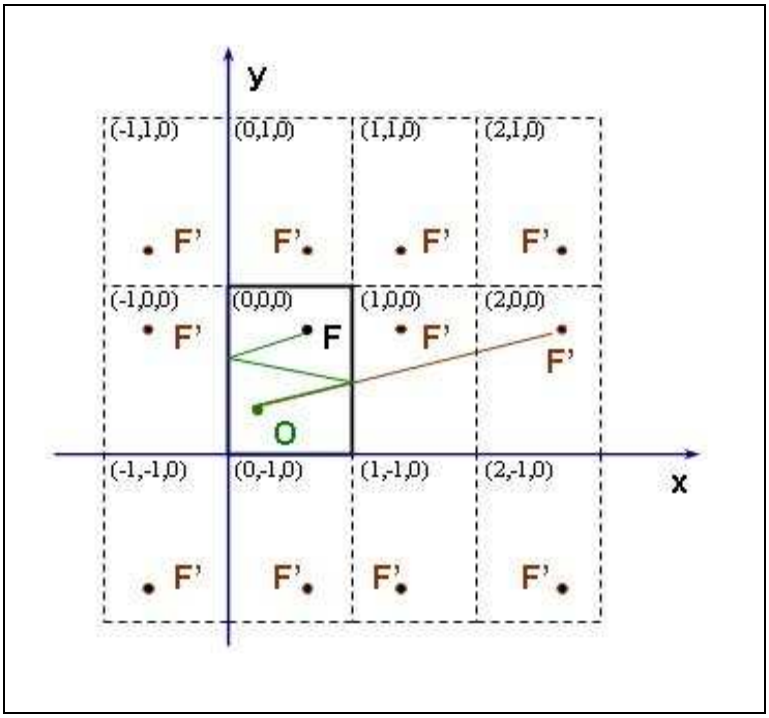

Figura 2.9: Algumas salas virtuais no plano (x,y)

2.2.3 Resposta Impulsiva

Usando o modelo de fontes virtuais, descrito na seção anterior, prosseguimos na simulação da resposta de freqüência da sala através do cálculo de sua *resposta impulsiva*.

A resposta impulsiva de um ambiente de escuta musical corresponde à resposta produzida por ele para um impulso produzido pela fonte sonora. Mais especificamente, a resposta impulsiva nos informa o intante e a intensidade com que cada reflexão chega a um ponto determinado, onde está situado o ouvinte (veja a figura 2.10).

Para o cálculo da resposta impulsiva, é necessário primeiramente saber *quando* cada reflexão chega ao ouvinte. Para isso, suponha que a posição da fonte sonora da sala real no espaço tridimensional seja (x_F, y_F, z_F), e que as dimensões da sala sejam dadas por (x, y, z). Então, a posição da fonte virtual da sala (i, j, k) no espaço cartesiano é dada pela fórmula abaixo:

$$(2x\lceil i/2\rceil + (-1)^{|i|}x_F, 2y\lceil j/2\rceil + (-1)^{|j|}y_F, 2z\lceil k/2\rceil + (-1)^{|k|}z_F).$$

Tendo a posição de cada fonte virtual no espaço, mais a posição do ouvinte, torna-se

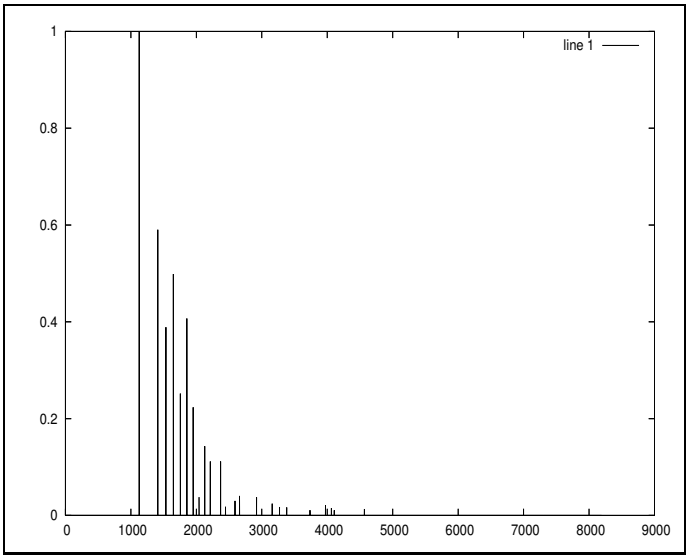

Figura 2.10: Resposta Impulsiva

possível calcular a distância percorrida por cada reflexão considerada até chegar ao ouvinte. Tendo essa distância em mãos, é fácil calcular o tempo gasto pelas reflexões até atingir o ouvinte, dividindo-se a distância pela velocidade do som no ar.

Para a simulação da resposta impulsiva, é preciso levar também em conta a absorção sofrida pelas ondas sonoras ao serem refletidas pelas superfícies da sala, bem como o efeito de atenuação devido à distância percorrida. O fator de atenuação do sinal em relação à distância percorrida d é $O(1/d)$. Além disso, um sinal sonoro saído da fonte localizada na sala virtual de índice (i, j, k) também sofre uma atenuação causada pelos coeficientes de absorção das superfícies, dada pela seguinte expressão:

$$\alpha_0^{|\lfloor i/2 \rfloor|} * \alpha_1^{|\lceil i/2 \rceil|} * \beta_0^{|\lfloor j/2 \rfloor|} * \beta_1^{|\lceil j/2 \rceil|} * \gamma_0^{|\lfloor k/2 \rfloor|} * \gamma_1^{|\lceil k/2 \rceil|}$$

onde os expoentes indicam justamente quantas vezes o sinal proveniente da sala (i, j, k) é refletido em cada uma das superfícies.

2.2.4 Resposta de Freqüência

Observamos, no início desta seção, que a resposta de freqüência de uma sala é a quantificação da atenuação ou amplificação sofrida por cada freqüência para uma certa disposição de fonte sonora e ouvinte (veja a figura 2.11). Vimos também que a resposta impulsiva é

uma função no domínio do tempo que nos dá a intensidade e o atraso de cada reflexão que atinge o ouvinte. Usando as técnicas baseadas na teoria de Fourier, podemos obter um espectro da resposta impulsiva através da aplicação da Transformada de Fourier a essa função. Assim, obteremos uma nova função complexa no domínio das freqüências, que corresponderá justamente à resposta de freqüência do ambiente.

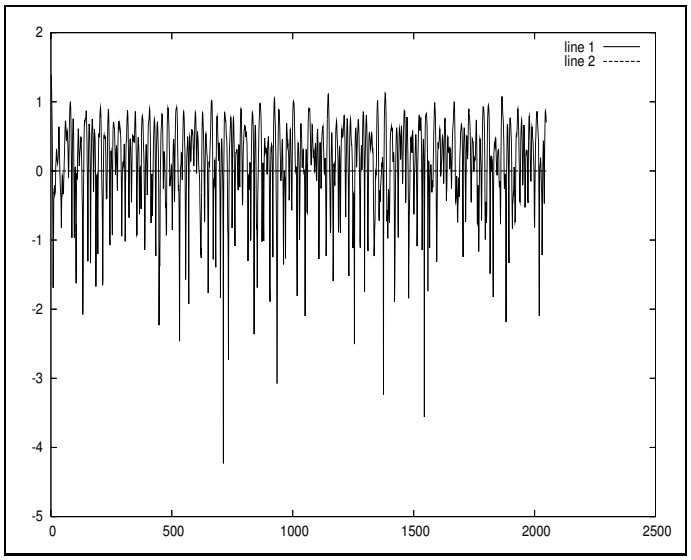

Figura 2.11: Resposta de Freqüência (em escala logarítmica)

A metodologia descrita nas seções anteriores permite a obtenção de *amostras* da resposta impulsiva da sala. Poderíamos, assim, aplicar a Transformada de Fourier Discreta (DFT) para obter o resultado desejado. Porém, vimos que existe um método capaz de obter o mesmo resultado da DFT muito mais eficientemente - a *Transformada Rápida de Fourier (FFT).* Dessa forma, basta aplicar a FFT à resposta impulsiva produzida para obter a resposta de freqüência da sala.

O vetor que contém a resposta impulsiva, para servir de entrada ao algoritmo da FFT, precisa ser "tratado", já que o algoritmo precisa receber como entrada N amostras, sendo N uma potência de 2. Para isso, o conjunto de amostras da resposta impulsiva é então subdividido em N partes e seus valores não-nulos são aproximados para corresponderem a uma dessas subdivisões.

2.2.5 Estimativa da Distorção Harmônica

Tendo a resposta de freqüência amostrada para determinadas posições de fonte sonora e ouvinte, podemos calcular a quantidade de distorção presente nessa resposta em relação a uma resposta de freqüência "neutra", ou seja, a *resposta plana*.

A resposta plana corresponde à situação hipotética em que nenhuma freqüência sofreria qualquer atenuação ou amplificação, a não ser aquela devida à absorção do ar. Esta situação hipotética só pode ser obtida num ambiente sem superfícies refletoras (ao ar livre ou em um ambiente com 100% de absorção), e não é sequer uma situação ideal, visto que alguma reverberação é de fato desejável em um ambiente de escuta musical.

Por outro lado a resposta plana fornece um bom parâmetro de comparação entre duas duas respostas de freqüência "reais", pois aquela que resultar em maior distorção estará também mais distante da resposta plana.

Podemos assim definir, para uma dada resposta de freqüência armazenada em um vetor F com N amostras, a medida de distorção d(F) dada por

$$d(F) = \sqrt{\sum_{i=0}^{N-1}(|F[i] - \mu|)^2}$$

onde $\mu = \frac{1}{N}\sum_{i=0}^{N-1} F[i]$ é a média dos fatores de atenuação/amplificação. Observe que esta expressão corresponde à distância entre o gráfico de F e o gráfico de resposta constante e igual a μ utilizando a norma Euclidiana.

Calculamos a distorção a partir da resposta logarítmica de freqüência, pois esta corresponde melhor à nossa percepção de distorção: intuitivamente, podemos pensar que uma amplificação s e um fator de atenuação $1/s$ são distorções igualmente "ruins".

2.3 Otimização Global

2.3.1 Introdução

O estudo e implementação da simulação da resposta de freqüência de salas com dimensões e coeficientes de absorção conhecidos torna possível uma busca automática de localizações ótimas para fonte sonora e ouvinte.

Para efetuar uma busca dessa natureza, é necessário aplicar um algoritmo de otimização. O objetivo de tal algoritmo é encontrar localizações para fonte e ouvinte associadas a uma resposta de freqüência o mais plana possível, isto é, a uma quantidade mínima de distorções no âmbito das freqüências.

Desta forma, em cada passo do algoritmo, analisamos uma certa localização da fonte sonora e do ouvinte e aplicamos a simulação da resposta de freqüência para obter a distorção correspondente, que tem o papel de função objetivo a ser minimizada. Ao final do algoritmo, desejamos obter as localizações mais adequadas para o posicionamento da fonte sonora e ouvinte, isto é, as posições que minimizam a distorção da resposta de freqüência.

Existem diversos tipos de procedimentos de otimização, cada um deles mais adequado do que outros para um determinado tipo de problema. Dessa forma, torna-se necessário analisar alguns dos principais modelos de otimização e suas aplicações, a fim de optar pelo mais adequado aos fins do projeto. Uma característica fundamental do nosso problema é o alto custo computacional envolvido no cálculo da função objetivo, que depende de uma simulação das reflexões sonoras e do cálculo da FFT; devemos, assim, evitar um número excessivamente grande de cômputos da função objetivo.

2.3.2 Algoritmos de Otimização Global

Muitos problemas de otimização contínua possuem várias soluções ótimas locais. Para esses casos, principalmente quando as condições do escopo "tradicional" de otimização contínua (diferenciabilidade e convexidade) não são satisfeitas ou verificáveis, costuma-se usar estratégias de otimização global.

O problema geral de otimização global contínua é denotado por

$$(P) \begin{cases} \min\ f(x) \\ \text{s.a. } x \in S \end{cases}$$

onde $f(x)$ é a função objetivo e S é o conjunto de soluções viáveis.

Em princípio, gostaríamos de encontrar um ótimo global, ou seja, um $x^* \in S$ tal que $f(x^*) \leq f(x), \forall x \in S$. Quando S é finito, podemos tentar fazer uma busca exaustiva, embora isso seja computacionalmente inviável na maioria dos casos. Quando S não é finito, em geral

utilizamos métodos iterativos, ou seja, métodos que geram uma seqüência de soluções que aproximam-se da solução ótima.

Os métodos de otimização global podem ser classificados como *determinísticos* ou *estocásticos*, bem como *heurísticos* ou *exatos*.

Nos métodos determinísticos, a partir dos mesmos dados iniciais, chegamos sempre à mesma solução. Os métodos baseados em grade (*Grid-based methods*) são exemplos de métodos determinísticos.

Já os métodos estocásticos possuem, em geral, algum componente de busca global aleatória. Por suas características, acabam sendo preferíveis em dimensões maiores e sem um modelo estrutural especial. Estes métodos podem ser utilizados em conjunto com estratégias determinísticas.

A distinção entre métodos exatos e heurísticos tem relação com a existência de garantias teóricas de convergência a uma solução global. Poucos problemas de otimização global são suficientemente bem estruturados a ponto de admitirem um método de solução exata; muitos problemas reais e relevantes só podem ser resolvidos aproximadamente. Mesmo métodos exatos de programação não-linear que garantem convergência a um mínimo local só podem ser considerados como sendo heurísticos na busca de um ótimo global.

Veremos a seguir alguns métodos estocásticos (ver [9]) e o método determinístico baseado em grades (ver [15]).

Métodos Estocásticos

Os métodos estocásticos são métodos que contêm algum elemento aleatório. Veremos duas classes de métodos estocásticos:

- Métodos de Busca Aleatória: visam encontrar o ótimo global, geram uma seqüência de pontos seguindo uma distribuição de probabilidade. Bons quando o número de ótimos locais é grande mas as flutuações na função objetivo não são tão grandes.

- Métodos de Duas Fases: usam, em geral, uma amostragem aleatória (fase global) e um procedimento de otimização local (fase local). São uma boa escolha quando o número de ótimos locais não é tão grande, ou quando o objetivo é encontrar todos os ótimos locais.

Métodos de Busca Aleatória

Os métodos desta família objetivam encontrar o ótimo global do problema. Em geral, eles geram uma seqüência de pontos seguindo uma distribuição de probabilidade, e a distribuição pode ser atualizada a cada iteração. Entretanto, é muito difícil uma implementação eficiente desses algoritmos.

Vimos anteriormente que o método da *Busca Aleatória Pura (PRS)* é um método de busca aleatória, onde o ótimo será encontrado com probabilidade 1 quando o tamanho da amostra tende a infinito. Entretanto, esta é uma estratégia nada eficiente. Os algoritmos de busca aleatória a seguir buscam uma melhoria a cada iteração.

A idéia geral de um algoritmo de busca aleatória é gerar uma seqüência de distribuições de probabilidade em $\mathbb{R}^d$, onde d é a dimensão do problema. A cada passo i, é escolhido um ponto x_i pertencente à distribuição atual, e dependendo do valor da função objetivo no ponto escolhido, a distribuição seguinte pode ser modificada, passando a ser um subconjunto da anterior.

A. Busca Aleatória Pura (*Pure RandomSearch - PRS*)

A PRS é o mais simples método estocástico para otimização global. Ele possui apenas uma fase global, e o algoritmo consiste basicamente no seguinte:

[1] $n \leftarrow 1$, $y_o \leftarrow +\infty$

[2] Gerar um ponto x da distribuição uniforme sobre S.

[3] Se $f(x)$ for melhor do que y_{n-1}, então $y_n \leftarrow f(x)$ e $x_n \leftarrow x$; senão, $y_n \leftarrow y_{n-1}$ e $x_n \leftarrow x_{n-1}$.

[4] Incrementar n e voltar para o passo 2.

A probabilidade de ter encontrado o ótimo, usando a PRS, tende a 1 quando $n \rightarrow \infty$. Assim, esse algoritmo é bastante ineficiente: o número de possibilidades por sorteio cresce exponencialmente com a dimensão do problema.

B. Busca Adaptativa Pura (*Pure Adaptive Search - PAS*)

Neste método, a cada passo, só são sorteados pontos que melhoram o valor da função objetivo em relação ao valor obtido pelo ponto anterior. Dessa forma, a tendência é que o conjunto de possibilidades para o próximo sorteio seja gradativamente reduzido.

A estratégia deste algoritmo é bastante eficiente na teoria. Porém, a construção dos sucessivos subconjuntos do conjunto viável S e a geração de pontos contidos nesses subconjuntos é muito difícil de ser obtida computacionalmente. Uma solução possível para este problema seria gerar os pontos sempre a partir de uma distribuição uniforme sobre S. Porém, é fácil notar que com esse procedimento o número de tentativas feitas para a obtenção de pontos que estejam dentro dos subconjuntos desejados cresce enormemente.

C. Busca Adaptativa (*Adaptive Search - AS*)

O método AS viabiliza a estratégia do algoritmo PAS através da geração de pontos a partir de uma seqüência de distribuições de *Boltzmann* sobre o conjunto viável. Através dessa seqüência de distribuições, é possível simular a seqüência de subconjuntos de S do PAS. É possível mostrar que o número de iterações do AS é menor do que o número que teríamos no PAS.

O próximo algoritmo, o *Simulated Annealing* pode ser visto como uma implementação aproximada do AS utilizando técnicas de cadeias de Markov.

D. *Simulated Annealing*

O *Simulated Annealing* é uma técnica de busca aleatória que tenta evitar que o procedimento fique "preso" em mínimos locais, aceitando, além de transições que melhoram o valor da função objetivo, transições que pioram, de forma limitada por um critério de aceitação probabilístico.

Durante o algoritmo, a probabilidade de aceitar passos que pioram o valor da função objetivo vai lentamente caindo para 0. Com essas "pioras", é possível sair de uma região de atração local e explorar a região viável S em sua integridade.

Este método baseia-se na mesma propriedade das distribuições de Boltzmann em que se baseia o AS. Como o AS é um algoritmo basicamente conceitual, o *Simulated Annealing* tenta

aproximar sua idéia geral.

Em linhas gerais, este algoritmo consiste no seguinte: tendo um passeio aleatório em S, que converge para uma distribuição uniforme, podemos filtrar esse passeio da maneira descrita nos passos abaixo.

[1] Seja x_n o ponto atual.

[2] Geramos y_{n+1} de acordo com a distribuição de probabilidade $R(x_n, \cdot)$ (distribuição de transição, dado que a cadeia de Markov está no estado x_n).

[3] Caso y_{n+1} melhore o valor da função objetivo, o aceitamos. Senão, o aceitamos com probabilidade $e^{(f(y_{n+1})-f(x_n))/T}$. (*critério de Metropolis*).

[4] Caso y_{n+1} não seja aceito, $x_{n+1} \leftarrow x_n$ e repetimos o procedimento.

No algoritmo, T é um número positivo "pequeno", que tende a 0 quando $n \to \infty$.

É possível verificar que existe um conjunto de condições sob as quais a seqüência gerada pelo algoritmo será eventualmente absorvida por vizinhanças arbitrariamente pequenas do mínimo global. Para maiores detalhes sobre o método *Simulated Annealing*, veja [9].

Métodos de Duas Fases

Como mencionado acima, os métodos de duas fases possuem uma fase global e uma local. Na fase global, é feita uma amostragem aleatória de pontos nos quais a função é calculada. Esses pontos são gerados a partir de uma distribuição uniforme em S. Já na fase local, um procedimento de otimização local é aplicado a cada ponto, obtendo, dessa forma, vários mínimos locais. Note-se que os métodos desta família objetivam encontrar *todos* os ótimos locais do problema P. Em nosso contexto, nos contentaremos em encontrar um grande número de mínimos locais, para selecionar o melhor deles.

A seguir, veremos alguns dos métodos de duas fases. Note-se que o método da Busca Aleatória Pura não pertence a essa família, mas será mencionado em primeiro lugar para efeito de comparação.

A. Multistart

O algoritmo Multistart parte da busca aleatória pura (PRS), porém utiliza um procedimento de busca local L. Ele é bastante parecido como PRS, mas aplica L aos pontos sorteados, obtendo assim o mínimo local de suas respectivas regiões de atração.

O Multistart é um pouco melhor do que o PRS, pois a partir dos pontos sorteados é possível obter alguns pontos locais ótimos. Porém, ele ainda é ineficiente, pois pode encontrar o mesmo ótimo local muitas vezes. Como o procedimento L costuma ser a parte mais cara do algoritmo, L deveria ser chamado apenas uma vez para cada região de atração.

B. Métodos *Clustering*

A idéia dos métodos desse tipo é gerar uma amostra usando a distribuição uniforme sobre o conjunto viável. Antes de aplicar L a cada ponto, são criados grupos com pontos relativamente próximos, e aplica-se L apenas uma vez para cada grupo.

Costuma-se "preparar" a amostra para a aplicação desses métodos de duas formas:

- Redução: tomamos uma fração da amostra apenas com os pontos com melhor valor da função objetivo.
- Concentração: modifica-se a amostra aplicando alguns passos de Cauchy para cada ponto.

O método básico para identificação de *clusters* é tomar o melhor ponto da amostra e adicionar outros pontos usando uma *regra de clustering*.

B.(a) *Density Clustering*

Este método baseia-se na idéia de que a região de atração de um mínimo local pode ser aproximada por uma elipsóide, isto é, a função objetivo pode ser localmente aproximada por uma função quadrática. O sucesso desse método depende do quão boa é essa aproximação.

O algoritmo consiste basicamente no seguinte:

[0] $k \leftarrow 1$, $X^* \leftarrow \emptyset$, $i \leftarrow 1$, $j \leftarrow 0$

[1] Gerar N pontos a partir da distribuição uniforme sobre o conjunto viável e determinar os γkN melhores pontos, para alguma fração γ (método da redução).

[2] Se todos os pontos da amostra reduzida já estão em algum *cluster*, vá para o passo *4*.
Se $j \leq |X^*|$, escolha o j-ésimo mínimo local em X^* como o próximo "ponto semente" e vá para o passo *3*.
Se $j > |X^*|$, aplique um método de otimização local L ao melhor dos pontos da amostra reduzida que não estiver em nenhum *cluster*, que chamaremos $\overline{x}$.
Se o mínimo local resultante x_* for um elemento de X^*, designe $\overline{x}$ ao *cluster* de x_* e volte para o passo *2*.
Se $x_* \notin X^*$, adicione x_* a X^* e faça x_* ser o próximo "ponto semente".

[3] Adicione todos os pontos sem *cluster* da amostra reduzida que estejam dentro de uma distância $r_i(x_*)$ do ponto semente ao *cluster* iniciado por x_*. Se para este valor específico de $r_i(x_*)$ nenhum ponto tiver sido adicionado, incrementar j e voltar para o passo *2*. Senão, incrementar i e repetir o passo *3*.

[4] Incrementar k e voltar para o passo *1*.

No algoritmo acima, a distância crítica $r_i(.)$ é dada por Rinnoy Kan e Timmer [13] como sendo igual à seguinte equação:

$$r_i(x) = \frac{1}{\pi}\left(i\Gamma\left(1+\frac{d}{2}\right)\cdot\sqrt{-detH(x)}\cdot m(S)\cdot\frac{\zeta ln(kN)}{kN}\right)^{1/d}$$

onde d é a dimensão do problema, $m(S)$ é a medida de Lebesgue, $H(x)$ é a Hessiana no ponto x, $\zeta > 0$ é um parâmetro positivo do algoritmo e $\Gamma(\cdot) = \int_{t=0}^{\infty} t^{z-1}e^{-t}dt$.

B.(b) *Single Linkage Clustering*

No método *Single Linkage*, diferentemente do *Density Clustering*, os clusters não possuem forma fixa a priori. A idéia geral desse algoritmo é, tendo iniciado um cluster C por um *ponto semente*, encontramos um ponto x qualquer que esteja fora de qualquer cluster tal que a distância

$$d(x, C) = \min_{y \in C} ||x - y||$$

seja mínima, isto é, um ponto x que esteja o mais próximo possível de algum ponto do cluster.

O *Single Linkage* vai prepetindo esse processo até que $d(x, C)$ exceda um valor crítico r_k. O valor de r_k também é sugerido por Rinnoy Kan e Timmer ([13]) como sendo:

$$r_k = \frac{1}{\pi}\left(\Gamma\left(1+\frac{d}{2}\right)\cdot m(S)\cdot\frac{\zeta ln(kN)}{kN}\right)^{1/d}.$$

Para uma esquematização detalhada deste método, veja [9].

Experimentos sugerem que esse método obtém uma aproximação da forma dos clusters melhor que o *Density Clustering*. Porém, como é possível que, através desse método, alguns pontos seja adicionados a clusters que não correspondem efetivamente à sua região de atração, alguns ótimos locais podem ser perdidos.

C. Multi Level Single Linkage

Este método tenta reunir em um único algoritmo a eficiência dos métodos *clustering* e as virtudes teóricas do *Multistart*. Aqui, o procedimento de busca local L é aplicado a todos os pontos da amostra, a menos que nela haja outro ponto com menor valor da função objetivo, dentro de uma certa distância crítica. Veja maiores detalhes sobre a estrutura deste método em [9].

Nesse algoritmo, a probabilidade de L ser aplicado a x tende a 0 quando o número de iterações tende a infinito. Assim, todo ótimo local será encontrado em um número finito de passos com probabilidade 1.

Método baseado em grades

O método de otimização baseado em grades é um método determinístico. Apresentaremos aqui dois modelos de otimização irrestrita baseados em grades.

Os algoritmos considerados buscam um minimizador da função objetivo examinando-a em uma seqüência de grades sucessivamente refinadas. Cada grade $G^{(m)}$ é definida por um conjunto de n vetores-base linearmente independentes $V^{(m)}$ onde

$$V^{(m)} = \{v_i^{(m)} \in \mathbb{R}^n : i = 1, \ldots, n\}.$$

Os pontos da grade $G^{(m)}$ são

$$G^{(m)} = \{x \in \mathbb{R}^n : x = x_o^{(m)} + h^{(m)} \sum_{i=1}^{n} \eta_i v_i^{(m)}\}$$

onde η é um inteiro e $h^{(m)}$ é um número positivo que representa o tamanho da malha, que é ajustado quando m cresce para assegurar o refinamento das malhas.

A base $V^{(m)}$ é usada para formar uma base positiva $V_+^{(m)}$ tal que todo vetor em $\mathbb{R}^n$ é combinação linear positiva dos vetores em V_+ e nenhum elemento de V_+ é combinação linear dos outros. Se $V^{(m)}$ é a base canônica, podemos construir uma base positiva fazendo

$$V_+^{(m)} = \{V_1^{(m)}, \ldots, V_n^{(m)}, -V_1^{(m)}, \ldots, -V_n^{(m)}\}.$$

Outra possibilidade é fazer

$$V_+^{(m)} = \{V_1^{(m)}, \ldots, V_n^{(m)}, -\sum_{i=1}^{n} V_i^{(m)}\}.$$

Por definição, temos que $x \in G^{(m)}$ é um *minimizador local de grades* com respeito à base positiva $V_+^{(m)}$ se e somente se

$$f(x + h^{(m)} v_i) \geq f(x), \forall v_i \in V_+^{(m)}.$$

O modelo de algoritmo A, que usa bases positivas, possui dois laços. O primeiro deles seleciona cada grade, e checa as condições de parada. O mais interno executa buscas finitas usando cada membro da base positiva até que p buscas consecutivas falhem. Quuando isso ocorre, um minimizador local da grade foi encontrado, o laço mais interno termina e o mais externo seleciona uma nova grade. Este algoritmo gera um novo iterado $x(k)$ somente quando um novo ponto com valor de função estritamente menor é encontrado.

Modelo A

[0] Inicializar $m \leftarrow 1$, $k \leftarrow 1$. Estabelecer um ponto de partida pertencente ao conjunto viável e atribuir seu valor a $x^{(1)}$.

[1] Enquanto as condições de parada forem falsas, faça:

[1.A] Escolher o tamanho da grade atual, $h^{(m)}$, e uma base positiva $V_+^{(m)}$. Inicializar $r \leftarrow 0$ e $p^{(m)} \leftarrow |V_+^{(m)}|$.

[1.B] Enquanto $r < p^{(m)}$ faça:

- Calcular f nos pontos $x^{(k)} + h^{(m)} v_i^{(m)}$.
- Se algum ponto menor do que $x^{(k)}$ for encontrado, $x^{(k+1)}$ recebe o menor desses pontos. Incrementar k. Reinicializar $r \leftarrow 0$. Senão, incrementar r.

[1.C] Incremente m e volte para o início do laço *1*.

Para o modelo de algoritmo A, a convergência para a solução só pode ser verificada para subseqüências dos minimizadores locais. A seguir, veremos o modelo B, uma especialização de A, para o qual pode-se mostrar convergência da seqüência completa de iterações. Nele, uma busca em profundidade é feita na direção $s^{(k)} \in V_+^{(m)}$, que satisfaz

$$f(x^{(k)} + h^{(m)} s^{(k)}) \leq f(x^{(k)} + h^{(m)} v), \forall v \in V_+^{(m)}. \tag{2.7}$$

Vale observar que a determinação de $s^{(k)}$ requer $p^{(m)}$ avaliações da função objetivo. A busca na direção $s^{(k)}$ avalia f na seqüência de pontos $x^{(k)} + \alpha_i h^{(m)} s^{(k)}$, onde α_i é uma seqüência de inteiros. A busca deve terminar quando é encontrado um minimizador local.

Modelo B

[0] Inicializar $m \leftarrow 1$, $k \leftarrow 1$ e $x^{(1)}$ com um ponto de partida.

[1] Enquanto as condições de parada forem falsas, faça:

[1.A] Escolher o tamanho da grade atual, $h^{(m)}$, e uma base positiva $V_+^{(m)}$. Inicializar $i \leftarrow 1$, $r \leftarrow 0$, e $p^{(m)} \leftarrow |V_+^{(m)}|$.

[1.B] Enquanto $x^{(k)}$ não é um minimizador local da grade, faça:

- Calcular a melhor direção de descida $s^{(k)}$, satisfazendo a expressão 2.7.
- Se $f(x^{(k)}) \leq f(x^{(k)} + h^{(m)} s^{(k)})$, sair deste laço, com $x^{(k)}$ como minimizador.

- Escolher uma seqüência crescente de inteiros $\alpha_0 = 1 < \alpha_1 < \alpha_2 \ldots$ e determinar o menor índice l que satisfaz

$$f(x^{(k)} + \alpha_{l+1} h^{(m)} s^{(k)}) \geq f(x^{(k)} + \alpha_l h^{(m)} s^{(k)}).$$

- $x^{(k+1)} \leftarrow x^{(k)} + \alpha_l h^{(m)} s^{(k)}$.
- Incrementar k.

[1.C] Incremente m.

É possível provar que, sob determinadas condições, qualquer algoritmo que esteja de acordo com o modelo B gera uma seqüência de pontos que converge para algum ponto estacionário de f. Para mais detalhes, veja o artigo [15].

2.3.3 Aplicação dos métodos de otimização ao projeto

Estes métodos de otimização global são algumas das possíveis abordagens para a busca de localizações ideais de fonte e ouvinte. Dentre ele, destacamos três, que são os métodos estocásticos *Density Clustering* e *Simulated Annealing*, e o método determinístico de busca em grades. Decidimos, em nosso projeto, testar e comparar o método estocástico *Density Clustering* com o método determinístico de busca em grades, para assim verificar qual seria o mais adequado para o problema em questão.

Capítulo 3

Atividades Realizadas

3.1 Implementação

3.1.1 Introdução

Um dos objetivos deste trabalho é contribuir com as pesquisas do projeto *ACMUS*, que visa o desenvolvimento de software para cálculo, análise e simulação de acústica de salas para prática musical (ver [5] e [6]).

Tendo em vista este propósito, foi desenvolvido um software para o Projeto Acústico Ótimo de Salas de Escuta (PAOSE), que será integrado às ferramentas de medição e simulação de comportamento acústico produzidas no projeto ACMUS.

A linguagem escolhida para a implementação dos métodos estudados neste projeto é a linguagem *Java* [16]. As vantagens dessa linguagem de programação concentram-se em sua robustez e versatilidade, permitindo aos desenvolvedores:

- Escrever um software em uma plataforma e executá-lo em outra.
- Criar programas para serem rodados dentro de um navegador Web (programas que possuem esta característica são conhecidos como *applets*).

Essas possibilidades vão de encontro aos propósitos do ACMUS, que pretende que as ferramentas produzidas sejam facilmente acessíveis.

3.1.2 Algoritmo da FFT

O primeiro algoritmo implementado foi a Transformada Rápida de Fourier (*Fast Fourier Transform - FFT*). Como detalhado anteriormente, este é um importante procedimento dentre as técnicas de Análise de Fourier, necessário para a transformação da resposta impulsiva da sala em resposta de freqüência.

O algoritmo foi implementado principalmente com base nas informações contidas em [3]. No código fonte do programa final, esta implementação encontra-se no método *fft(f, F)* do arquivo *FFT.java*. Este método recebe um sinal discretizado f, que é um vetor contendo a resposta impulsiva, de tamanho igual a uma potência de 2. O espectro correspondente produzido é armazenado no vetor F.

3.1.3 Simulação da Resposta Impulsiva

O cálculo da resposta impulsiva da sala é feito no método *generateImpulseResponse*, que se encontra no arquivo *ImpulseResponse.java*. Seus dados de entrada consistem nas características de uma sala de escuta cubóide (dimensões e coeficiente de absorção de cada uma de suas superfícies) e posições fixadas para 1 fonte sonora e 1 ouvinte. A partir dessas informações, é gerada computacionalmente uma série de posições para "fontes virtuais", situadas em salas virtuais ao redor da sala real e que são imagens especulares da fonte original.

O número máximo de fontes virtuais gerado é também passado como parâmetro, e é representado por um "raio máximo" de salas em torno da sala real. Em outras palavras, para um raio máximo igual a N, serão geradas todas as fontes virtuais contidas em salas cujos índices (i, j, k), indicadores de suas posições em relação à sala real, satisfaçam a condição $|i| + |j| + |k| \leq N$.

A partir da lista das posições das fontes virtuais, obtemos os intantes em que os raios sonoros provenientes de cada uma delas atinge o ouvinte. Adicionando a esta informação a atenuação sofrida por cada onda devido à distância percorrida e aos coeficientes de absorção das superfícies atingidas, é fácil calcular a intensidade de cada reflexão. Com isso, obtemos um mapa acústico da sala, tendo uma lista das intensidades das reflexões que chegam ao ouvinte a cada instante.

Contudo, a lista gerada ainda não está pronta para servir de entrada ao algoritmo da FFT,

uma vez que ele só aceita como entrada conjuntos de amostras em quantidade igual a uma potência de 2. Assim, a resposta impulsiva ainda passa por um tratamento de aproximação das amostras.

Na implementação, essa aproximação é feita no método *generateFFTInputArray*, que está no arquivo *FrequencyResponse.java*. Como os instantes de chegada de cada reflexão, para salas pequenas, são menores que 1, foram tomadas as x casas decimais seguintes como sendo os índices daquela reflexão em um vetor adequado de tamanho igual a uma potência de 2. Aqui, x é igual a $-\log_{10}\epsilon$, onde ϵ é uma constante real próxima de 0[1].

A resposta impulsiva discretizada é uma função com muitas descontinuidades, o que gera no espectro uma quantidade muito grande de ruído. Uma maneira de compensar este efeito colateral é aplicar um filtro à resposta de freqüência produzida. Por isso, o método *applyFilter*, também em *FrequencyResponse.java*, implementa e aplica à resposta impulsiva um filtro simples. Nesta implementação, foi usado um cálculo relativamente direto para suavizar as descontinuidades: cada amostra da resposta impulsiva é substituída por uma média de duas amostrasss adjacentes. Isso corresponde a um filtro "passa-baixa"(ver [1]).

3.1.4 Simulação da Resposta de Freqüência

Unindo os dois procedimentos descritos nas seções anteriores (FFT e cálculo da resposta impulsiva), torna-se simples obter a simulação da resposta de freqüência da sala. Isso porque a aplicação da FFT à resposta impulsiva gera um espectro correspondente, que indica a intensidade de cada uma das freqüências ouvidas pelo ouvinte naquela sala.

Para calcular a distorção, é feita uma estimativa da resposta de freqüência ideal (*resposta plana*), isto é, a resposta de freqüência que obteríamos na ausência de distorções. Para isso, poderíamos calcular o desvio padrão da resposta de freqüência obtida na simulação (em relação à média dos valores obtidos).

Nossa audição, entretanto, responde de forma logarítmica aos valores de amplitude do sinal sonoro; por exemplo, um fator de distorção igual a 4 (uma amplificação de 4 vezes) é comparativamente tão grande quanto um fator de $\frac{1}{4}$ (uma atenuação de 4 vezes), em relação

[1] *Observação:* os valores para o raio máximo de salas e para o erro máximo permitido estão definidos no arquivo *Defs.java*, nas variáveis *maxSum* e *epsilon*, respectivamente.

à nossa percepção. Por isso, consideramos a resposta de freqüência logarítmica, em que um fator de 1 (ausência de distorção) corresponde ao valor 0. Fazendo dessa forma, consideramos como resposta plana a função constante igual a 0.

O método *stdDeviation*, no arquivo *FrequencyResponse.java*, é responsável por realizar o cálculo da resposta impulsiva, gerar a partir deste resultado a entrada adequada para o algoritmo da FFT e obter a resposta de freqüência, tudo isso através de métodos auxiliares. Este método foi encapsulado na classe *ObjectiveFunctionPAOSE* para que pudesse ser chamado a partir dos algoritmos de otimização.

3.1.5 Algoritmos de Otimização

Como mencionado anteriormente, foram implementados dois algoritmos de otimização: o *Density Clustering* e o *Grid-based method*. Ambos foram escritos de forma que pudessem ser reutilizados para a resolução de outros problemas, permanecendo independentes desta aplicação específica.

Foi ainda criada uma classe *Cache*, responsável por armazenar os valores da função objetivo em todos os pontos já percorridos pelo algoritmo. Essa estratégia foi escolhida porque cada cálculo do valor da função objetivo é relativamente caro; por isso, é necessário aproveitar todos os cálculos já efetuados.

O algoritmo *Density Clustering* foi implementado no método *executeAlgorithm* do arquivo *DensityClustering.java*. Ele executa uma série de iterações buscando o ótimo global de um problema.

Para a execução deste método, é necessário o conhecimento dos seguintes valores: a dimensão d do problema, o número N de pontos a serem sorteados a cada iteração, a fração de redução γ, o número máximo de iterações desejado para o algoritmo, o número máximo de iterações para o método de otimização local, uma matriz contendo os limites de cada uma das d coordenadas dos pontos a serem sorteados, um objeto do tipo *ObjectiveFunction*, que deve encapsular a chamada da função objetivo desejada, a medida de Lebesgue do conjunto viável (que para este problema de otimização consiste simplesmente no produto das dimensões da sala) e o objeto *Cache* que fará o armazenamento dos pontos visitados.

O procedimento de busca de ótimos locais usado nesta implementação do *Density Cluste-*

ring foi o próprio *Grid-based method*, implementado no método *executeAlgorithm* do arquivo *GridMethod.java*.

Também implementamos a opção de usar o *Grid-based method* como algoritmo de otimização global para o problema. Neste caso, ele é chamado diretamente com os seguintes parâmentros: a dimensão d do problema, o ponto de partida do algoritmo *xInit*, uma matriz contendo os limites de cada uma das d coordenadas dos pontos a serem sorteados, um objeto do tipo *ObjectiveFunction*, que deve encapsular a chamada da função objetivo desejada, o número máximo de iterações permitido, o tamanho da malha inicial, e o objeto *Cache* que fará o armazenamento dos pontos visitados.

3.1.6 O programa

O método principal do programa do Projeto Acústico Ótimo de Salas de Escuta (PAOSE) encontra-se no arquivo *MainPAOSE.java*. A interface (em linha de comando) do programa é

`java MainPAOSE arquivo.txt [OPÇÃO]`.

O arquivo.txt deve conter os dados da sala (dimensões e coeficientes de absorção) no seguinte formato:

```
X Y Z
alpha0 alpha1 beta0 beta1 gamma0 gamma1
```

onde X, Y e Z são as dimensões da sala e alpha0, ..., gamma1 são os coeficientes de absorção de suas superfícies.

O segundo argumento consiste em um código que define o procedimento de otimização a ser utilizado no programa. As opções são:

`-d` para que seja aplicado o método *Density Clustering* com o algoritmo baseado em grades como método de otimização local.

`-g` para que seja aplicado o método baseado em grades como método de otimização global.

O programa imprime na saída padrão as seguintes informações:

- o tempo de execução até a obtenção da resposta;
- a maior quantidade de distorção dentre as geradas nos pontos visitados pelo método de otimização;
- as localizações ótimas encontradas para fonte e ouvinte;
- a quantidade de distorção gerada nas localizações ótimas encontradas.

Ainda, fazendo com que o valor da variável booleana *debug* do arquivo *Defs.java* seja *true* será impresso na saída padrão o caminho completo percorrido pelo algoritmo de otimização.

3.2 Testes e Resultados

Foram realizados alguns testes para salas hipotéticas. Inicialmente, consideramos uma única fonte e um único ouvinte, não colocando mais restrições (além dos limites da sala) às suas possíveis posições. Testamos para uma sala de dimensões $X = 3m$, $Y = 4m$ e $Z = 5m$ e coeficientes de absorção variados, todos com valores entre 0 e 1. Foram realizados testes com ambos os métodos de otimização implementados. A partir desses testes, foram feitas sucessivas melhorias nos algoritmos, de modo a diminuir o tempo de execução e aumentar a precisão dos resultados. Os dados obtidos em tais testes, aplicando respectivamente o método baseado em grades (*Grid-based method*) e o método estocástico *Density Clustering*, foram os seguintes:

```
- Grid-based Method -
Número de avaliações da função objetivo: 57
Tempo de resposta: 185.326 s
Valor do pior ponto sorteado: 0.7797158869018019
Localizações ótimas encontradas:
   Fonte: (2.25, 0.0, 0.0)
   Ouvinte: (2.25, 2.0, 2.5)
Quantidade de distorção nas localizaçõs ótimas: 0.3958269857678017

- Density Clustering -
```

```
Tempo de resposta: 2552.477 s
Valor do pior ponto sorteado: 1.7976931348623157E308
Localizações ótimas encontradas:
   Fonte: (1.6875, 1.75, 2.5)
   Ouvinte: (1.6875, 1.75, 2.490234375)
Quantidade de distorção nas localizações ótimas: 0.36589052199761113
```

Podemos observar a partir destes resultados o comportamento típico dos dois métodos. O Density encontrou uma solução melhor que a encontrada pelo Grid, em tempo também maior.

Entretanto, tais testes não são capazes de refletir com precisão uma situação real, visto que não faz sentido posicionar uma fonte e um ouvinte em qualquer ponto da sala. Por isso, fizemos algumas alterações no programa para que pudéssemos considerar o problema de localizar duas fontes e um ouvinte, dispostos simetricamente em uma sala de escuta de dimensões idênticas às dos testes anteriores. Uma das fontes ocupa a posição (x_F, y_F, z_F) e a outra a posição $(X - x_F, y_F, z_F)$, onde (x_F, y_F, z_F) satisfaz $0 \leq x_F \leq X/2$, $Y/2 \leq y_F \leq Y$ e $0 \leq z_F \leq Z$. Já o ouvinte deve ser posicionado em (x_O, y_O, z_O), onde $x_F = X/2$ (restringindo assim a localização do ouvinte ao eixo central da sala), $0 \leq y_O \leq Y/2$, $0 \leq z_O \leq z$. Este é um problema que pode ser resolvido considerando-se apenas a primeira fonte e o ouvinte, dado que a resposta impulsiva produzida pela segunda fonte é idêntica à produzida pela primeira, e portanto a distorção harmônica produzida pela segunda fonte é também idéntica.

O método de otimização baseado em grades, ao ser aplicado aos dados acima, produziu a seguinte resposta:

```
- Grid-based Method -
Número de avaliações da função objetivo: 75
Tempo de resposta: 87.509 s
Valor do pior ponto sorteado: 1.4202278753437534
Localizações ótimas encontradas:
   Fonte: (0.2109375, 3.5, 4.84375)
   Ouvinte: (1.5, 0.5, 0.0)
Quantidade de distorção nas localizações ótimas: 0.5617007729474769
```

Neste exemplo, observa-se uma variação de 252.84% entre a pior e a melhor localização encontrada. Vale a pena mencionar também que os pontos de partida do método foram $(x_F = 0.0, y_F = 0.0, z_F = 0.0)$ e $(x_O = 1.5, y_O = 0.0, z_O = 0.0)$, nos quais a distorção total era de 1.339057806330562 (238.38% acima do ótimo).

Já o método de otimização *Density Clustering* produziu a resposta:

```
- Density Clustering -
Tempo de resposta: 1378.040 s
Valor do pior ponto sorteado: 0.9838447204945652
Localizações ótimas encontradas:
   Fonte: (1.499908447265625, 3.5316162109375, 4.10125732421875)
   Ouvinte: (1.5, 0.8748779296875, 0.00152587890625)
Quantidade de distorção nas localizações ótimas: 0.5761921592717776
```

Observe que o tempo de resposta é bem mais alto, o que já era esperado, visto que o Grid é uma subrotina do Density, e é chamado um grande número de vezes. Neste teste em particular, a solução final encontrada pelo Density não era melhor do que a do Grid, o que pode ser considerado como "sorte" deste último, ao encontrar um mínimo local tão bom em apenas uma tentativa.

Levando-se em consideração os resultados destes testes, notamos que várias melhorias ainda poderiam ser feitas. Considerando que o método Grid busca apenas um minimizador local da função, é natural que seu tempo de execução seja muito menor que o do Density. É natural supor que um usuário escolherá o método de busca global apenas quando tiver tempo para esperar a resposta. Assim, algumas variantes do método podem ser exploradas.

Uma destas variantes consiste em substituir a base positiva do método baseado em grades $V_+^{(m)} = \{V_1^{(m)}, \ldots, V_n^{(m)}, -\sum_{i=1}^n V_i^{(m)}\}$, escolhida para minimizar o número de chamadas da função objetivo, pela base $V_+^{(m)} = \{V_1^{(m)}, \ldots, V_n^{(m)}, -V_1^{(m)}, \ldots, -V_n^{(m)}\}$. Esta última fornece mais liberdade de movimentação para o método e aumenta a chance de se encontrar uma direção de descida a partir do ponto atual. Após feita esta modificação, os mesmos dados de descrição da sala foram fornecidos novamente aos dois métodos. Os resultados obtidos a partir da aplicação do Grid e do Density foram, respectivamente, os seguintes:

```
- Grid-based method -
```

```
Número de avaliações da função objetivo: 150
Tempo de resposta: 182.401 s
Valor do pior ponto sorteado: 0.9292232704290136
Localizações ótimas encontradas:
   Fonte: (0.04541015625, 2.998046875, 0.0)
   Ouvinte: (1.5, 1.0, 4.6875)
Quantidade de distorção nas localizações ótimas: 0.5905246503402999

- Density Clustering -
Tempo de resposta: 2543.094 s
Valor do pior ponto sorteado: 1.1790867848297337
Localizações ótimas encontradas:
   Fonte: (0.61083984375, 3.181640625, 4.844207763671875)
   Ouvinte: (1.5, 0.001953125, 0.15625)
Quantidade de distorção: 0.5005439899823095
```

Nestes testes, notamos novamente que o Density, sendo um método de busca global, encontrou uma solução melhor do que a do baseado em grades. Entretanto, percebemos que a resposta encontrada pelo Grid foi bem próxima à encontrada pelo Density, o que podemos atribuir à maior quantidade de direções de busca possíveis proporcionada pela nova base.

Outras extensões deste trabalho ainda podem ser exploradas, as quais serão descritas no caítulo seguinte.

3.3 Integração com o software do projeto ACMUS

O desenvolvimento do software do projeto ACMUS tem acontecido progressivamente, à medida em que os diversos projetos que visam contribuir com ele vão sendo concluídos. Atualmente, este desenvolvimento está sendo conduzido por Leo Kazuhiro Ueda, sob orientação do prof. Fábio Kon.

Um de nossos objetivos era o de integrar o programa produzido nesta iniciação ao ACMUS ainda no ano de 2005. Entretanto, viu-se em reunião a impossibilidade de realizar esta integração nesse período, visto que há outros projetos que foram concluídos anteriormente a este

e cuja integração com o ACMUS ainda está sendo realizada, e portanto é prioritária. Dessa forma, a solução encontrada foi modularizar e documentar o nosso programa o melhor possível para que ele seja integrado posteriormente, no momento em que houver disponibilidade para isso.

3.4 Participação no Colóquio de Iniciação Científica

Acreditando ser este projeto interessante a outros alunos do Instituto, visto seu cunho interdisciplinar envolvendo aplicações de matemática à música e desenvolvimento de software, participei, sob recomendação do orientador, do XVIII Colóquio de Iniciação Científica do Instituto de Matemática e Estatística. Este Colóquio teve realização nos dias 29 e 30 de setembro de 2004, e minha participação no evento ocorreu através de painel e de uma apresentação oral.

Capítulo 4

Conclusão

Pode-se dizer que, em geral, os progressos realizados durante o período desta iniciação científica estiveram de acordo com o planejado. Certamente, os resultados computacionais obtidos sugerem várias possibilidades de melhoria, tanto no modelo quanto na implementação. No entanto, são contribuições originais deste trabalho o uso de estratégias de otimização global e a disponibilização da implementação em código aberto.

A partir do presente trabalho são possíveis algumas extensões. Uma delas corresponde a melhorias no modelo da sala e no resultado da simulação por acústica geométrica. A consideração com maior prioridade de inclusão é a representação dos possíveis coeficientes de absorção das superfícies como funções da freqüência. Isso se daria através da utilização de tabelas de materiais com valores de coeficiente de absorção por banda de freqüência, ou ainda aproximações baseadas em modelos matemáticos das curvas de absorção, com preferência para a opção que representar o menor overhead computacional para o método. Coeficientes de absorção variam também em função do ângulo de incidência; tal dependência poderia ser levada em consideração através de valores tabulados ou de modelos analíticos.

Também apresenta grande interesse prático a consideração de outras geometrias de sala além da cubóide. Entretanto, isso apresenta grande dificuldade computacional e teórica. Isso porque ao considerarmos salas de geometria poliédrica, o modelo das fontes virtuais deixa de possuir fórmulas fechadas para a localização das fontes e passa a depender de técnicas de Álgebra Linear. Também não é desprezível o fato de que em geral, para salas poliédricas, o número de fontes virtuais de ordem n cresce exponencialmente em função de n (para geometria

cubóide, este crescimento é polinomial). Uma alternativa é utilizar o modelo de traçado de raios (*ray tracing*), que gera uma grande quantidade de raios saindo da fonte real em todas as direções, e acompanha o caminho geométrico desses raios até atingirem o ouvinte. Existe ainda a possibilidade de utilizar uma estratégia mista de fontes virtuais e traçado de raios.

Finalmente, poderíamos considerar ainda outros fenômenos acústicos, como difusão, difração e sombreamento acústico. A modelagem desses outros fenômenos, entretanto, traria como custo um aumento considerável tanto na complexidade do modelo matemático como no tempo de execução. Embora seja esperado que estas considerações tornem o modelo mais robusto e condizente com a realidade, no contexto da automatização do projeto de salas tais inclusões devem ser avaliadas em relação à melhoria real da solução obtida pelo método de otimização e o acréscimo de custo computacional envolvido.

Parte II

Experiência Pessoal

Capítulo 5

5.1 Desafios e Frustrações

As atividades referentes a este projeto de Iniciação Científica, que incluíram estudos, implementação e a redação dos relatórios científicos para a FAPESP, proporcionaram uma série de desafios.

Primeiramente, com relação ao estudo da teoria envolvida, posso dizer que exigiram muita dedicação os assuntos da Análise de Fourier e de Otimização Global, por envolverem uma base matemática bastante complexa. Já os estudos para a simulação da acústica de salas foram muito interessantes; através do modelo de fontes virtuais, pude deduzir e aplicar fórmulas que possibilitaram simular com boa precisão o comportamento de cada reflexão que atinge o ouvinte.

Quanto à implementação, o principal obstáculo surgido foi o de manter a eficiência do programa. A linguagem escolhida para o desenvolvimento do software foi a linguagem *Java*, para facilitar a compatibilidade com outras ferramentas desenvolvidas pelo projeto ACMUS. Entretanto, esta linguagem é interpretada, além de ser orientada a objetos e possuir uma estrutura complexa e pesada. Assim, como o programa efetua muitos cálculos matemáticos, foi relativamente complicado adequar sua estrutura a essa linguagem, e uma série de adaptações dos algoritmos tiveram de ser feitas a fim de assegurar a performance do programa em tempo hábil.

A impossibilidade de integrar nosso programa com o software do ACMUS ainda este ano, mencionada em "Atividades Realizadas", foi, para mim, um pouco frustrante, já que durante todo o desenvolvimento do programa fui tentando prepará-lo para esse processo.

De qualquer forma, fico satisfeita por ter contribuído um pouco com a parte de simulação acústica do projeto temático. Há, também, outros projetos contribuintes muito interessantes, relacionados, por exemplo, com a medição de parâmentros acústicos de uma sala. Acredito que, quando concluído, o ACMUS será uma das ferramentas mais poderosas existentes para auxiliar projetos de salas destinadas à prática musical.

A elaboração dos relatórios para a FAPESP foi também uma experiência bastante desafiadora, já que eu nunca havia produzido documentos científicos antes. Foi bastante gratificante tê-los aprovados por este órgão de fomento.

Por fim, a principal dificuldade enfrentada certamente foi a de conciliar o tempo entre o BCC e as atividades da iniciação. Havia a necessidade de cumprir o compromisso assumido com o ACMUS e com a FAPESP, e tive que me esforçar bastante para não deixar cair o meu desempenho nas disciplinas.

5.2 O BCC e a Iniciação Científica

A formação que recebi durante o curso de Ciência da Computação foi, para mim, fundamental em diversos aspectos. Durante o BCC, tive não só a oportunidade de adquirir conhecimentos em computação e matemática, mas também de desenvolver meu raciocínio e, principalmente, de me "educar" para o aprendizado contínuo de novos assuntos e tecnologias.

Para este trabalho, além de terem sido necessários estudos fora da área de Computação, por se tratar de um projeto multidisciplinar, as disciplinas oferecidas pelo BCC foram muito importantes. Dentre elas, vale destacar as seguintes:

- **Programação Linear** e **Programação Não-Linear:** O conteúdo ministrado nestas disciplinas foi fundamental na segunda etapa do projeto, na qual foram estudados e implementados métodos de otimização contínua para a determinação das localizações ótimas de fonte e ouvinte.

- **Computação Musical:** Apesar de ter cursado esta disciplina durante o período final do projeto, pude, através dela, entender melhor muitos dos conceitos que havia estudado, como por exemplo o conjunto de técnicas de análise de sinais sonoros baseadas na Teoria

de Fourier. Além disso, foi muito interessante aprender outros assuntos da área do meu projeto, como filtros e síntese de sons.

- **Cálculo Diferencial e Integral I e IV:** Todos os Cálculos foram importantes para o embasamento matemático do projeto. Contudo, o I e o IV em particular foram muito úteis durante os estudos sobre Análise de Fourier, por abrangerem respectivamente os princípios fundamentais do Cálculo e a teoria sobre as Séries de Fourier.

- **Física II:** Nesta matéria pude aprender mais sobre Ondulatória e Acústica e entender melhor a estrutura do som e de seu espectro.

- **Introdução à Computação** e **Princípios de Desenvolvimento de Algoritmos:** Disciplinas introdutórias da programação, foram fundamentais não só para este projeto, mas para a minha formação como um todo. Tive a oportunidade de cursá-las com excelentes professores, e através delas pude aprender os conceitos básicos da lógica de programação, a modelagem correta de algoritmos e diversas técnicas para implementá-los. Acredito que o conteúdo teórico da disciplina **Estrutura de Dados** também teria enriquecido muito a minha formação, porém infelizmente ela não foi ministrada com a profundidade necessária na ocasião em que a cursei.

- **Introdução à Probabilidade e Estatística II:** Apliquei alguns conceitos dessa disciplina para realizar a estimativa da distorção harmônica, e também para os estudos de métodos estocásticos de otimização global.

- **Análise de Algoritmos:** Uma característica fundamental do problema tratado neste projeto é o alto custo computacional envolvido no cálculo da função objetivo, que depende de uma simulação das reflexões sonoras e do cálculo da FFT. Dessa forma, o conteúdo desta matéria foi útil para a escolha dos algoritmos a serem implementados.

- **Laboratório de Programação I e II:** Nestas matérias tive as primeiras experiências em desenvolvimento de sistemas grandes, divididos em várias fases. Foi ainda em Laboratório de Programação II que aprendi os conceitos fundamentais de programação orientada a objetos, como encapsulamento e técnicas de reaproveitamento de código,

além de ter também começado a aprender a programar na linguagem Java, utilizada na implementação deste projeto.

- **Programação Orientada a Objetos** e **Tópicos de Programação Orientada a Objetos:** Embora a maior parte do conteúdo visto nestas matérias não tenha sido aplicado no projeto diretamente, pude aqui aprofundar meus conhecimentos sobre programação orientada a objetos, e com isso tornar a estrutura do programa mais modular, generalizada e extensível.

5.3 Interação com o orientador e com o grupo de pesquisa

Durante os estudos teóricos para a realização do projeto, o meu orientador, o prof. Marcelo Queiroz, pôde acompanhar esses estudos através de reuniões periódicas, e a cada assunto estudado eu fazia uma explanação do mesmo para o próprio orientador. Esses seminários foram importantes tanto para que o professor avaliasse o quão bem eu havia assimilado o tópico estudado, quanto para que eu tirasse dúvidas e fixasse melhor o conteúdo. Além disso, ao me preparar para participar do Colóquio de Iniciação Científica, fiz para ele uma prévia da minha apresentação oral, o que me ajudou a preparar melhor tanto a forma quanto o conteúdo apresentado. Além desses seminários, o orientador também acompanhou a implementação de cada componente do programa, tanto através de reuniões quanto via e-mail. Essas interações foram importantes para discutir o andamento do projeto e decidir os próximos passos.

Também passei por uma experiência interessante ao interagir com outros orientandos do prof. Marcelo, os mestrandos Mariana Zaparolli Martins e Cezar Monteiro Pirajá Neto, que estão atualmente realizando pesquisas também na área de Computação Musical. Cada um de nós apresentou aos outros um seminário relacionado ao seu trabalho, o que foi, para mim, muito útil para conhecer outros ramos de pesquisa dentro da área do meu projeto.

Participei, também, de algumas reuniões do grupo ACMUS, conduzidas pelos professores Fábio Kon (IME) e Fernando Iazzetta (ECA), coordenadores do ACMUS. Nessas reuniões, todos os pesquisadores participantes do grupo compartilham com os outros o andamento de seus trabalhos, e discutem sobre a maneira com que cada um deles contribuirá com o projeto temático como um todo. Também é nesses momentos que são decididos os próximos passos

da integração dos diversos projetos com o software do ACMUS.

5.4 Próximos Passos

Para o futuro próximo, pretendo documentar e organizar melhor o programa que desenvolvi durante o projeto para deixá-lo pronto para entregá-lo ao grupo de pesquisa ACMUS. Além disso, também pretendo realizar alguns testes de comparação do programa com o *Room Optimizer*([4]), um software proprietário para Windows de propósitos parecidos com os do presente projeto, ou seja, otimizar o posicionamente do ouvinte, fontes sonoras e tratamento acústico de superfícies em salas de escuta.

Caso fosse continuar atuando nessa área de pesquisa, gostaria de aprofundar os estudos abrangendo algumas de suas possíveis extensões, como a consideração de mais fontes e ouvintes e de outras geometrias de salas, ampliando, dessa forma, suas aplicações e seu interesse prático.

5.5 Agradecimentos

Ao IME, por toda a formação e estrutura oferecida; à FAPESP, pelo apoio financeiro.

Ao meu orientador, o Prof. Marcelo Queiroz, por todos os conhecimentos transmitidos e todo o auxílio prestado durante o desenvolvimento deste projeto de iniciação científica. Também ao grupo de pesquisa ACMUS, pela oportunidade que me foi dada de contribuir com o projeto temático.

Finalmente, agradeço à minha família, que sempre me apoiou em meus estudos e durante toda a minha vida. A todos os meus amigos e colegas que estiveram presentes durante esses quatro anos, pelos inesquecíveis momentos passados juntos. E ao Mateus, por estar sempre ao meu lado.

Bibliografia

[1] Moore, F. R. (1990). *Elements of Computer Music*. Prentice-Hall.

[2] Roads, C. & Strawn, J. (1996). *The Computer Music Tutorial*. Mit Press.

[3] Morrison, N. (1994). *Introduction to Fourier Analysis*. Wiley-Interscience.

[4] D'Antonio, P. & Cox, T.J. (1997) *ROOM OPTIMIZER: A Computer Program to Optimize the Placement of Listener, Loudspeakers, Acoustical Surface Treatment and Room Dimensions in Critical Listening Rooms*, 103rd Convention of the Audio Engineering Society, Preprint 4555, Paper H-6, New York.

[5] Iazzetta, F. H.; Kon, F. & Silva F. (2001). *ACMUS: Design and Simulation of Musical Listening Environments*, Proceedings of the 8th Brazilian Symposium on Computer Music, Fortaleza.

[6] AcMus
`http://gsd.ime.usp.br/acmus/projeto.html`

[7] Queiroz, M. (2003) *Some Optimization Models for Listening Room Design*, Proceedings of the 9th Brazilian Symposium on Computer Music, Campinas.

[8] Beranek, Leo L. (1993) *Acoustics*, New York: Acoustical Society of America.

[9] Horst, R. & Pardalos, P. M. (1995) *Handbook of Global Optimization*, Kluwer Academic Publishers.

[10] Rindel, J. H. (1997) *Computer Simulation Techniques for the Acoustical Design of Rooms - how to treat reflections in sound field simulation*, Proceedings of ASVA'97, pp. 201-208, Tokyo.

[11] Rindel, J. H. (2000) *The Use of Computer Modeling in Room Acoustics*, Journal of Vibroengineering, Vol. 3, No. 4, pp. 41-72.

[12] Warusfel, O. (1995) *Predictive acoustics software and computer aided optimization in room acoustics*, 15th Intl. Congress on Acoustics, Trondheim, Norway, pp.693-696.

[13] A. H. G. Rinnoykan & G. T. Timmer, "Stochastic Global Optimization Methods. Part I: clustering methods,"*Mathematical Programming* 39 (1987) 27-56.

[14] López, M. R. (2001) *Acondicionamiento Acústico*. Paraninfo.

[15] I. D. Coope & C. J. Price, "On the convergence of grid-based methods for unconstrained optimization", *SIAM J. Optim.*, 11 (2001), No. 4, pp. 859-869.

[16] Java Technology `http://java.sun.com`