

**Universidade de São Paulo
Instituto de Física de São Carlos
Departamento de Física e Informática
Grupo de Física Computacional e Instrumentação Aplicada**

RENATO FABBRI

**Música no áudio digital: descrição psicofísica e
caixa de ferramentas**

São Carlos

2013

RENATO FABBRI

Música no áudio digital: descrição psicofísica e caixa de ferramentas

Dissertação apresentada ao Programa de Pós-graduação em Física do Instituto de Física de São Carlos da Universidade de São Paulo, para a obtenção do título de Mestre em Ciências.

Área de Concentração: Física Aplicada

Orientador: Prof. Dr. Osvaldo Novais de Oliveira Junior

Colaborador: Prof. Dr. Luciano da Fontoura Costa

Versão Corrigida

(versão original disponível na Unidade que aloja o Programa)

São Carlos

2013

AUTORIZO A REPRODUÇÃO E DIVULGAÇÃO TOTAL OU PARCIAL DESTE TRABALHO, POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO PARA FINS DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

Ficha catalográfica elaborada pelo Serviço de Biblioteca e Informação do IFSC,
com os dados fornecidos pelo(a) autor(a)

Fabbri, Renato

Música no áudio digital: descrição psicofísica e
caixa de ferramentas / Renato Fabbri; orientador
Osvaldo Novais de Oliveira Junior - versão corrigida
-- São Carlos, 2013.
253 p.

Dissertação (Mestrado - Programa de Pós-Graduação em
Física Aplicada) -- Instituto de Física de São Carlos,
Universidade de São Paulo, 2013.

1. Som. 2. Áudio digital. 3. Música. 4.
Psicofísica. 5. Arte e tecnologia. I. Oliveira
Junior, Osvaldo Novais de, orient. II. Título.

AGRADECIMENTOS

Agradeço ao Prof. Luciano da Fontoura Costa e ao Prof. Osvaldo Novais de Oliveira Junior pela oportunidade e pela orientação deste trabalho.

Agradeço ao corpo de funcionários do IFSC, pela prestatividade e eficiência em todos os momentos que precisei.

Agradeço aos Prof. Rafael Santos Mendes e Prof. Adolfo Maia Junior pelas orientações na gênese deste trabalho.

Agradeço à minha família, em especial à minha esposa Thaís Teixeira Fabbri e ao meu filho Antônio Anzoategui Fabbri.

Agradeço ao meu irmão Ricardo Fabbri e ao amigo Vilson Vieira da Silva Junior pelas profícuas colaborações acadêmicas, artísticas e socialmente engajadas.

Agradeço aos participantes do labMacambira.sf.net pelas colaborações diretas e indiretas nos *software*, apresentações e articulações desde junho de 2011. Em especial agradeço ao Daniel Penalva, Caleb Macarenhas, Chico Simões, Fábio Simões, Daniel Marostegan, Marcos Mendonça, Geraldo Magela Rocha, Guilherme Lunhani, Patrícia Ferraz, Glerm Soares, Danilo Shiga, Edson "Presto" Corrêa, Vanessa Ferreira e Sérgio Teixeira de Carvalho. Agradeço aos demais colaboradores das listas de email, IRC, AA e outros canais.

Agradeço ao Cultura Viva e aos pontos de cultura pelo apoio fundamental a este e outros trabalhos. Agradeço em especial a estes Pontões e Cultura Digital e Ação Griô pelo suporte a este trabalho em ações regionais e nacionais: CDTL (PE), JuntaDados.org (BA), Nós Digitais (SP), Casa dos Meninos (SP), Nina Griô (SP), Pontão da Eco (RJ), Casa de Cultura Tainã (SP) e à Comissão Nacional dos Pontos de Cultura - CNPdC (GO).

Agradeço às comunidades de cultura e software livre por todos os conhecimentos e tecnologias repassados e que compõem esta contribuição.

*"Music is a hidden arithmetic exercise of the soul,
which does not know that it is counting."*
— GOTTFRIED LEIBNIZ (1646-1716)

*"Music is a hidden metaphysical exercise of the soul,
which does not know that it is philosophizing."*
— ARTHUR SCHOPENHAUER (1788-1860)

*"from Pantheon import Obatalá
from World import sharing, kitten
while True:
 if sharing == 0:
 Obatalá.kill(kitten)"*
— AUTORIA COLETIVA E ANÔNIMA (2010)

RESUMO

FABBRI, R. **Música no áudio digital: descrição psicofísica e caixa de ferramentas**. 2013. 253p. Dissertação (Mestrado) - Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos, 2013.

A representação dos elementos básicos da música - tais como notas, ornamentos e estruturas intervalares - em termos do som discretizado é bastante utilizada em *software* e rotinas para criação musical e tratamento sonoro. Não há, entretanto, uma abordagem concisa que relacione estes elementos às amostras sonoras. Nesta dissertação, cada elemento musical é descrito por equações que resultam diretamente nas sequências temporais do som em sua representação discretizada. O elemento fundamental, a nota musical básica com duração, volume, altura e timbre, é relacionado quantitativamente às características do sinal digital. As variações internas, como *tremolos*, vibratos e flutuações espectrais, também são contempladas, o que permite sintetizar notas com inspiração nos instrumentos musicais reais além de sonoridades novas. A partir desta representação das notas, dispomos de recursos para a geração de estruturas musicais, como a métrica rítmica, os intervalos de altura e os ciclos. As equações deram origem a uma caixa de ferramentas computacionais: *scripts* que realizam cada equação e geram exemplos sonoros simples. Nomeamos MASSA, Música e Áudio em Sequências e Séries Amostrais, este ferramental e sua eficácia foi comprovada com a síntese de pequenas peças usando notas básicas, notas incrementadas e notas em música. É possível, também, sintetizar álbuns inteiros através de colagens dos *scripts* e parametrização especificada pelo usuário. Com o paradigma de implementação em código aberto, a (caixa de ferramentas) MASSA pode ser expandida em processos de co-autoria e usada livremente por músicos, engenheiros e outros interessados. De fato, o sistema já foi empregado por usuários externos para a produção de músicas, apresentações artísticas, experimentos psico-acústicos e a difusão da linguagem computacional através do apelo lúdico dos artefatos audiovisuais.

Palavras-chave: Som. Áudio digital. Música. Psicofísica. Arte e tecnologia.

ABSTRACT

FABBRI, R. **Music on digital audio: psychophysical description and toolbox**. 2013. 253p. Dissertação (Mestrado) - Instituto de Física de São Carlos, Universidade de São Paulo, São Carlos, 2013.

The representation of the basic elements of music - such as notes, ornaments and intervalar structures - in terms of discrete audio signal is often used in software for music creation and design. Nevertheless, there is no unified approach that relates these elements to the sound discrete samples. In this dissertation, each musical element is described by equations that represent the sonic time samples, which are then implemented in scripts within a software toolbox, referred to as MASSA (Music and Audio in Sequences and Samples). The fundamental element, the musical note with duration, volume, pitch and timbre, is related quantitatively to the characteristics of the digital signal. Internal variations, such as tremolos, vibratos and spectral fluctuations, are also considered, which enables the synthesis of notes inspired by real instruments and new sonorities. With this representation of notes, resources are provided for the generation of musical structures, such as rhythmic meter, pitch intervals and cycles. The efficacy of MASSA was confirmed by the synthesis of small musical pieces using basic notes, incremented notes and notes in music. It is possible to synthesize whole albums through collage of the scripts and parameterization specified by the user. With the paradigm of open source implementation, MASSA toolbox can be expanded in co-authorship processes and used freely by musicians, engineers and other interested parties. In fact, MASSA has already been employed by external users for diverse purposes which include music production, artistic presentations, psychoacoustic experiments and computer language diffusion where the appeal of audiovisual artifacts is exploited.

Keywords: Sound. Digital audio. Music. Psychophysics. Art and technology.

LISTA DE FIGURAS E TABELAS

Figura 1.1 -	Som digital em modulação por código de pulsos (PCM): 25 amostras representadas por 4 bits cada uma.	35
Figura 2.1 -	Formas de onda musicais básicas. As formas de onda sintéticas estão em (a) e as formas de onda reais estão em (b).	44
Figura 2.2 -	Espectros das ondas sonoras musicais artificiais básicas. A senoide tem o espectro puntual, a triangular apresenta somente os harmônicos ímpares, caindo a 6dB por oitava; a onda quadrada tem somente os harmônicos ímpares, caindo a 12dB por oitava; a onda dente de serra apresenta todos os harmônicos, caindo a 6dB por oitava.	45
Figura 2.3 -	Espectros das ondas sonoras de uma nota de oboé natural e de período amostrado. O som natural possui flutuações nos harmônicos e ruídos, já o som de período amostrado possui espectro perfeitamente harmônico.	47
Figura 2.4 -	Oscilação de 2 amostras (frequência máxima em qualquer f_a). O primeiro coeficiente reflete o deslocamento (<i>offset</i> ou <i>bias</i>) e o segundo coeficiente especifica a amplitude da oscilação. . . .	49
Figura 2.5 -	3 amostras fixas apresentam uma só frequência não nula. $c_1 = c_2^*$ e $w_1 \equiv w_2$	50

Figura 2.6 -	Componentes frequenciais em 4 amostras.	51
Figura 2.7 -	Formas de onda básicas em 4 amostras.	52
Figura 2.8 -	Componentes frequenciais em 6 amostras: 3 senoides se somam ao <i>bias</i>	53
Figura 2.9 -	Formas de onda básicas em 6 amostras: as ondas triangular e quadrada possuem os harmônicos ímpares, mas em proporções e fases diferentes; a dente de serra possui também o harmônico par.	53
Figura 2.10 -	Detecção de localização espacial de fonte sonora: esquema uti- lizado para cálculo da diferença de tempo interaural (DTI) e da diferença de intensidade interaural (DII).	55
Figura 2.11 -	Mixagem de três sequências sonoras. As amplitudes são sobre- postas diretamente.	60
Figura 2.12 -	Concatenação de três sequências sonoras através da justaposição temporal de suas amostras.	61
Figura 2.13 -	Procedimento de busca em tabela (conhecido como <i>Lookup Ta- ble</i>) para síntese de sons em frequências diferentes a partir de uma única forma de onda em alta resolução.	64
Figura 2.14 -	Transições de intensidade para diferentes valores de α (veja equa- ções 2.39 e 2.40).	67
Figura 2.15 -	Interpretação gráfica da convolução. Cada amostra resultante é a soma das amostras anteriores de um sinal uma a uma multipli- cadas pelas amostras retrógradadas do outro sinal.	70

Figura 2.16 -	Convolução com o impulso: deslocamento (a), linhas de delays (b) e síntese granular (c). Dispostos em ordem crescente de densidade de pulsos.	72
Figura 2.17 -	Módulos da resposta em frequência (a), (b), (c) e (d) respectivamente dos filtros IIR das equações 2.45, 2.46, 2.48 e 2.49 para diferentes frequências de corte, frequências centrais e larguras de banda.	74
Figura 2.18 -	Ruídos coloridos realizados através das equações 2.50, 2.51, 2.52, 2.53, 2.54: espectros e ondas sonoras resultantes.	77
Figura 2.19 -	Espectrograma de um som com vibrato senoidal de $3Hz$ e profundidade de uma oitava em uma dente de serra de $1000Hz$ (considerada $f_a = 44.1kHz$).	81
Figura 2.20 -	Tremolo de profundidade $V_{dB} = 12dB$ com padrão oscilatório de uma dente de serra em $f' = 1.5Hz$ em uma senoide de $f = 40Hz$ (considerada taxa de amostragem $f_a = 44,1kHz$).	83
Figura 2.21 -	Envoltória ADSR (<i>Attack, Decay, Sustain, Release</i>) e uma sequência sonora arbitrária submetida à envoltória. A variação linear de amplitude está acima. Abaixo a variação de amplitude é exponencial.	93

Tabela 2.22 -	Intervalos musicais, suas notações tradicionais, classificações básicas de dissonância e número de semitons. As consonâncias perfeitas são os uníssonos, as quintas e as oitavas justas (J). As consonâncias imperfeitas são as terças e as sextas maiores (M) e menores (m). As dissonâncias fortes são as segundas menores e sétimas maiores. As dissonâncias brandas são as segundas maiores e as sétimas menores. O primeiro caso especial consiste na quarta justa, que é consonante perfeita se considerada uma inversão da quinta justa, caso contrário pode ser considerada uma dissonância ou uma consonância imperfeita. O segundo caso especial é o trítono (4aum, 5dim, tri). Este é consonante em algumas culturas. Já para a música tonal, o trítono indica dominante e busca sua resolução em uma terça ou sexta e, por esta instabilidade, é considerado intervalo dissonante.	97
Tabela 2.23 -	Resumo das funções harmônicas tonais para a escala maior. A tônica é o centro da música, a dominante tende à tônica e a subdominante se distancia da tônica. Os três acordes podem, a princípio, serem substituídos livremente pelas respectivas relativas ou anti-relativas.	106
Figura 2.24 -	Movimentos diferenciados pelo contraponto com vistas a preservar a independência entre as vozes. 3 tipos de movimentos: direto, contrário e oblíquo, categorizam as possibilidades. O movimento paralelo é um tipo de movimento direto.	109
Tabela 2.25 -	Transição das durações ouvidas individualmente para alturas. . .	110

Figura 2.26 -	Divisões e aglomerações do pulso musical para estabelecimento de métrica. Ao lado esquerdo estão as divisões da semínima estabelecida como pulso. Ao lado direito, fórmulas de compasso que especificam as mesmas métricas, mas na escala das aglomerações do pulso musical.	111
Figura 2.27 -	Distinções canônicas do clímax musical em uma melodia e outros domínios. As possibilidades diferenciadas são: clímax no começo, clímax na primeira metade, clímax no meio, clímax na segunda metade, clímax no fim. Não está especificado o eixo das ordenadas pois pode não haver variação paramétrica real, neste caso a estrutura é uma referência.	117
Tabela 2.28 -	Change Ringing: <i>Peal</i> (padrão) com 3 sinos. As permutações estão entre as ordenações. Cada linha é uma ordenação dos sinos, cada ordenação é tocada, uma linha por vez.	119

LISTA DE RELAÇÕES ANALÍTICAS DESCRITAS E IMPLEMENTADAS COMPUTACIONALMENTE NO APÊNDICE A

Equação	2.1 - Sequência de amostras de um áudio PCM.
Equação	2.2 - Potência.
Equação	2.3 - Decibels entre dois áudios.
Equação	2.4 - Amplitude dobrada em decibels.
Equação	2.5 - Potência dobrada em decibels.
Equação	2.7 - Variação de amplitude no volume dobrado (10 dBs).
Equação	2.8 - Conversão de decibels em variação de amplitude.
Equação	2.9 - Representação básica de uma sequência periódica.
Equação	2.10 - Sequência infinita senoidal.
Equação	2.11 - Sequência infinita de uma onda 'dente de serra'.
Equação	2.12 - Sequência infinita de uma onda 'triangular'.

Equação	2.13 - Sequência infinita de uma onda 'quadrada'.
Equação	2.14 - Sequência infinita de uma forma de onda <i>sampleada</i> .
Equação	2.15 - Recomposição das amostras temporais com base nos coeficientes espectrais.
Equação	2.16 - Recomposição das amostras temporais reais em termos do módulo e fase.
Equação	2.17 - Número de pares de coeficientes relativos à mesma frequência.
Equação	2.18 - Coeficientes equivalentes em termos das frequências que representam.
Equação	2.19 - Equivalências dos módulos dos coeficientes espectrais.
Equação	2.20 - Equivalências das fases dos coeficientes espectrais.
Equação	2.21 - Reconstrução das amostras temporais em termos dos coeficientes pareados e independentes.
Equação	2.22 - Nota básica com duração, frequência e altura.
Equação	2.23 - Forma de onda sintética ou amostrada.
Equação	2.24 - Nota básica com forma de onda especificada.
Equação	2.25 - Distância de uma fonte sonora a cada ouvido.
Equação	2.26 - Diferença de Tempo Interaural (DTI).
Equação	2.27 - Diferença de Intensidade Interaural (DII).
Equação	2.28 - Sequência binaural PCM com DTI e DII para localização espacial.
Equação	2.29 - Ângulo resolvido pela implementação da DTI e DII.

Equação	2.30 - Mixagem de N sequências.
Equação	2.31 - Concatenação de N sequências.
Equação	2.32 - Procedimento de busca em tabelas (<i>Lookup Table</i>).
Equação	2.33 - Frequências em cada amostra em uma variação linear.
Equação	2.34 - Índices para busca na tabela em uma variação linear de frequência.
Equação	2.35 - Sequência amostral em uma variação linear de frequências.
Equação	2.36 - Frequências em cada amostra em uma variação exponencial.
Equação	2.37 - Índices para busca na tabela em uma variação exponencial de frequência.
Equação	2.38 - Sequência amostral em uma variação exponencial de frequências.
Equação	2.39 - Sequência de amplitudes em uma variação exponencial.
Equação	2.40 - Sequência amostral em uma variação exponencial de amplitude.
Equação	2.41 - Sequência de amplitudes em uma variação linear de amplitude.
Equação	2.42 - Sequência amostral em uma variação exponencial de amplitude dada em decibels.
Equação	2.43 - Convolução de sequências reais e finitas.
Equação	2.44 - Equação a diferenças para aplicação temporal de filtros IIR.
Equação	2.45 - Coeficientes de um filtro passa-baixas bem comportado de primeira ordem com frequência de corte variável.
Equação	2.46 - Coeficientes de um filtro passa-altas bem comportado de primeira ordem com frequência de corte variável.

Equação	2.47 - Variáveis auxiliares de um filtro nó de segunda ordem.
Equação	2.48 - Coeficientes de um filtro passa-banda de segunda ordem com frequência central e largura de banda variáveis.
Equação	2.49 - Coeficientes de um filtro rejeita-banda de segunda ordem com frequência central e largura de banda variáveis.
Equação	2.50 - Coeficientes espectrais para síntese de ruído branco.
Equação	2.51 - Coeficientes espectrais para síntese de ruído rosa.
Equação	2.52 - Coeficientes espectrais para síntese de ruído marrom.
Equação	2.53 - Coeficientes espectrais para síntese de ruído azul.
Equação	2.54 - Coeficientes espectrais para síntese de ruído violeta.
Equação	2.55 - Coeficientes espectrais para síntese de ruído preto.
Equação	2.56 - Índices auxiliares para um vibrato de frequência variável.
Equação	2.57 - Expoentes auxiliares para um vibrato de frequência variável.
Equação	2.58 - Frequências por amostra de um vibrato de frequência e profundidade variáveis.
Equação	2.59 - Índices um vibrato de frequência e profundidade variáveis.
Equação	2.60 - Amostras de um áudio com vibrato de frequência e profundidade variáveis.
Equação	2.61 - Amplitudes por amostra de um tremolo de frequência e profundidade variáveis.

Equação	2.62 - Amostras de um áudio com tremolo de frequência e profundidade variáveis.
Equação	2.63 - Espectro da síntese FM.
Equação	2.65 - Espectro da síntese AM.
Equação	2.66 - Índices auxiliares para síntese FM.
Equação	2.67 - Sequência auxiliar para síntese FM.
Equação	2.68 - Sequência de frequências para cada amostra para síntese FM.
Equação	2.69 - Índices para síntese FM.
Equação	2.70 - Sequência amostral resultante da síntese FM.
Equação	2.71 - Sequência de amplitudes por amostra na síntese AM.
Equação	2.72 - Sequência amostral resultante da síntese AM.
Equação	2.73 - Exemplo de vínculo entre a frequência e parâmetros do tremolo e do vibrato em um som.
Equação	2.74 - Equação básica da frequência observada no efeito Doppler.
Equação	2.75 - Amplitude observada no efeito Doppler.
Equação	2.76 - Frequência efetiva observada no efeito Doppler.
Equação	2.77 - Primeiro período de uma reverberação.
Equação	2.78 - Segundo período de uma reverberação.
Equação	2.79 - Resposta ao impulso de uma reverberação.

Equação	2.80 - Envoltória ADSR com variações lineares e logarítmicas.
Equação	2.81 - Aplicação da envoltória ADSR em uma sequência arbitrária.
Equação	2.22 - Intervalos musicais em número de semitons.
Equação	2.82 - Escalas simétricas na oitava dividida em 12 notas.
Equação	2.83 - Escalas diatônicas.
Equação	2.84 - Sucessão de intervalos em uma escala diatônica.
Equação	2.85 - Escalas menores natural, harmônica e melódica.
Equação	2.86 - Tríades maior, menor, diminuta e aumentada.
Subseção	2.3.1 - Relações microtonais através de frações de semitons ou através de quantidades inteiras de divisões arbitrárias da oitava.
Subseção	2.3.2 - Relações básicas de harmonia tonal.
Subseção	2.3.3 - Regras básicas de condução de vozes com independência.
Subseção	2.3.4 - Relações básicas de métrica e rítmica.
Subseção	2.3.6 - Formação de arcos musicais através de estruturas direcionais.
Subseção	2.3.7 - Estruturas cíclicas para a síntese musical.

Elementos de notação

Hz abreviação de Herz, medida de frequência, número de ocorrências por segundo.

kHz abreviação de kilo Hertz, i.e. mil Herz.

m metro.

mm milímetro.

s segundo.

m/s metros por segundo.

f_a frequência de amostragem, taxa de amostragem.

λ_a duração da separação temporal entre um par de amostras consecutivas.

PCM sigla de *Pulse Code Modulation*, veja seção 1.1

S_i sequência S indexada em i .

$\{s_i\}_x^y$ sequência com elementos s_i com índices de x a y incluso.

$\lfloor x \rfloor$ parte inteira de x .

V_{dB} volume em decibels.

\Rightarrow então.

\therefore portanto.

$:$ tal que.

\forall para todo.

\approx aproximadamente.

\equiv equivalente.

$\%$ operação módulo, resto da divisão.

Λ comprimento em amostras.

$\widetilde{\Lambda}$ comprimento em amostras de período de onda.

Δ duração temporal.

a_i fator multiplicativo de amplitude da i -ésima amostra.

Sumário

1	Introdução	33
1.1	Som em áudio digital	33
1.2	Arte sonora e teoria musical	35
1.3	Implementação computacional	36
1.4	Objetivos	37
1.5	Trabalhos relacionados	37
2	Música no som digitalizado	39
2.1	Caracterização da nota musical em tempo discreto	39
2.1.1	Duração	40
2.1.2	Volume	40
2.1.3	Altura	42
2.1.4	Timbre	42
2.1.5	O espectro no som amostrado	47
2.1.6	A nota básica	53
2.1.7	Localização espacial e espacialização	54

2.1.8	Usos musicais	59
2.2	Variações na nota musical básica	63
2.2.1	Tabela de busca	63
2.2.2	Variações incrementais de frequência e intensidade	65
2.2.3	Aplicação de filtros digitais	69
2.2.4	Ruídos	76
2.2.5	Tremolo e vibrato, AM e FM	80
2.2.6	Usos musicais	86
2.3	Organização de notas em música	95
2.3.1	Afinação, intervalos, escalas e acordes	95
2.3.2	Harmonias atonal, tonal, expansão e modulação	103
2.3.3	Contraponto	109
2.3.4	Ritmo	110
2.3.5	Repetição e variação: motivos e unidades maiores	113
2.3.6	Estruturas direcionais	116
2.3.7	Estruturas cíclicas	118
2.3.8	Idioma musical?	120
2.3.9	Usos musicais	121

REFERÊNCIAS **125**

Apêndice A – Código computacional dos procedimentos expostos no capítulo 2 **133**

A.1	Código Python das relações descritas na seção 2.1	133
A.2	Código Python das relações descritas na seção 2.2	138
A.3	Código Python das relações descritas na seção 2.3	150

Apêndice B – Código Computacional das Peças Musicais **157**

B.1	Peças referentes à seção 2.1	158
B.1.1	Quadros sonoros	158
B.1.2	Reduced-fi	161
B.2	Peças referentes à seção 2.2	163
B.2.1	Transita para metro	163
B.2.2	Vibra e treme	167
B.2.3	Tremolos, vibratos e a frequência	167
B.2.4	Trenzinho de caipiras impulsivos	167
B.2.5	Ruidosa faixa	167
B.2.6	Bela Rugosi	173
B.2.7	Chorus infantil	176
B.2.8	ADa e SaRa	176
B.3	Peças referentes à seção 2.3	177

B.3.1	Intervalos entre alturas	177
B.3.2	Cristais	177
B.3.3	Micro tom	180
B.3.4	Acorde cedo	183
B.3.5	Conta ponto	183
B.3.6	Poli Hit Mia	183
B.3.7	Dirracional	183
 Apêndice C – <i>Finite Groups in Granular and Unit Synthesis</i> e a síntese de um EP		187
C.1	FIGGUS	187
C.1.1	FIGGUS.py	187
C.1.2	tables.py	187
C.1.3	__init__.py	187
C.2	PPEPPS: músicas de um EP solvente	188
C.2.1	RUNME make EP MUSIC.py	188
C.2.2	Éter	189
C.2.3	Benzina	191
C.2.4	Clorofórmio	194
 Apêndice D – Síntese FM e AM em escala logarítmica		199
 Apêndice E – Análise sonora e musical		201

E.1	Espectro em frequência	201
E.2	Eventos no tempo	202
E.3	Análise de partituras	202

Apêndice F – Música digital em domínios não digitais **203**

F.1	Experimentos abertos em áudio: LADSPAs, Wavelets e Redes Complexas . . .	203
F.2	Áudio e música	206
F.2.1	Música em tempo diferido: <i>minimum-fi</i> e FIGGUS	206
F.2.2	Música em tempo real: <i>Livecoding</i> e ABeatTracker (ABT)	217
F.2.3	Música na matéria: EKP e AHT	224
F.2.4	Música no tecido social: Sabrina Kawahara, Audioexperiments, Estu- dioLivre.org, CDTL, juntaDados.org, Devolts.org, MSST, LabMacam- bira.sf.net	226
F.3	Materiais didáticos	229
F.3.1	Tutoriais em texto e código: python, filtros e nyquist, plugins lv2, me- trics, carta mídias livres, contra-cultura digital	229
F.3.2	<i>Screencasts</i> e outros materiais em video	232
F.4	Web	233
F.4.1	Tecnologias sociais: Sítios, Conteúdos e Articulação	233
F.5	Momento atual e previsões	236

Apêndice G – Trabalhos relacionados e caracterização das contribuições deste tra-

balho **239**

G.1 Livros 240

G.2 Bibliotecas, linguagens e conjuntos de implementações computacionais volta-
dos para música 249

G.3 Aprofundamento sobre esta dissertação com base nos trabalhos visitados nas
subseções G.1 e G.2 253

1 *Introdução*

"Tradicionalmente a notação musical é vista como um código através do qual sons, ideias musicais ou indicações para execução musical são registrados sob forma escrita."

Edson S. Zampronha.(1)

Representar estruturas e artifícios musicais através das características do som discretizado é a proposta deste trabalho. Os resultados são relações matemáticas e suas implementações computacionais. Uma descrição teórica está no capítulo 2 e o conjunto de *scripts* disponibilizados no Apêndice A e *online*. A caixa de ferramentas (*toolbox*) recebeu o nome MASSA (música e áudio em sequências e séries amostrais) e foi utilizada para fazer pequenas peças e montagens focadas nos princípios expostos. O Apêndice B possui uma relação destas montagens assim como o diretório *exemplos_de_uso* da MASSA.(2)

1.1 Som em áudio digital

O som é uma onda mecânica longitudinal de pressão. A banda de frequências compreendida entre 20Hz e 20kHz é apreciada pelo aparelho auditivo humano com variações dependentes da pessoa, das condições climáticas e do som em si. Considerada a velocidade do som no ar

$\approx 343.2m/s$, estes limites correspondem respectivamente aos comprimentos de onda $\frac{343.2}{20} = 17.16m$ e $\frac{343.2}{20000} = 17.16mm$.(3)

A percepção humana do som envolve captações pelos ossos, estômago e orelha, funções de transferência da cabeça e dorso e processamento pelo sistema nervoso. Além disso, o ouvido é um órgão dedicado à captura destas ondas. Seu funcionamento decompõe o som em seu espectro senoidal e passa para o sistema nervoso.(3) Estas componentes senoidais são cruciais para os fenômenos musicais, como se pode observar tanto na composição dos sons de interesse para a música quanto nas afinações e escalas.(4) A subseção 2.1 expõe a presença de senoides no som discretizado e caracteriza a nota musical básica.

A representação do som é o áudio¹ e este pode provir da captura do som por microfones ou da síntese. Muitas vezes, o áudio digital é especificado através de protocolos que facilitam o armazenamento e transferência dos arquivos. A representação digital do som pode consistir em amostras igualmente espaçadas no tempo e cujas amplitudes individuais são registradas com um mesmo número de *bits*. Estas amostras separadas por intervalos regulares λ_a constituem a forma padrão de representação do som em tempo discreto, chamada de modulação por código de pulsos (PCM do inglês *Pulse Code Modulation*). Um som digital PCM é caracterizado pela frequência de amostragem $f_a = \frac{1}{\lambda_a}$, também chamada de taxa de amostragem, e a profundidade de *bit* que é o número de *bits* utilizados para representar a amplitude de cada amostra. A figura 1.1 exibe 25 amostras de um áudio PCM com 4 *bits* cada. Os $2^4 = 16$ graus para a amplitude de cada amostra junto ao espaçamento regular λ_a introduzem um erro de quantização. O ruído causado por estes erros diminuem com a diminuição destes espaçamentos.(6)

Pelo teorema de Nyquist, constata-se que a metade da frequência de amostragem é a frequência máxima do sinal. Assim, para apreender as frequências audíveis, é necessária uma taxa de amostragem que seja ao menos o dobro da frequência mais aguda $f_a \geq 2 \times 20kHz = 40kHz$. Este

¹ Os termos som e áudio são muitas vezes usados de forma intercambiável.(5)

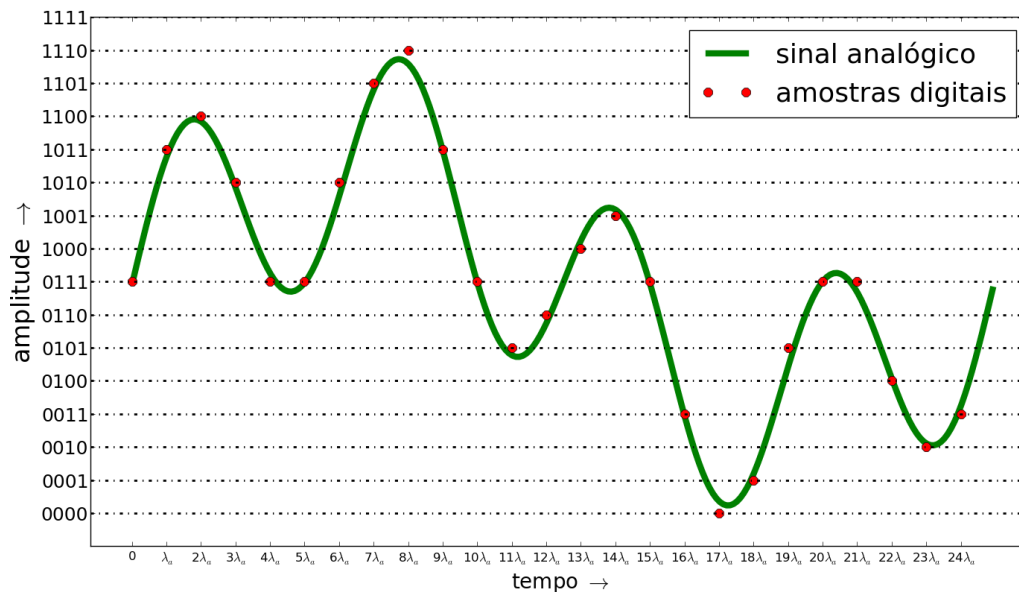


Figura 1.1 – Som digital em modulação por código de pulsos (PCM): 25 amostras representadas por 4 bits cada uma.

raciocínio está na base da utilização das frequências de amostragem $f_a = 44.1kHz$ e $f_a = 48kHz$, ambas padrão em *Compact Disks* (CDs) e em sistemas de Rádio e TV, respectivamente.(6)

1.2 Arte sonora e teoria musical

A música é definida como a arte manifesta pelos sons e silêncios. Para um ouvinte comum - e boa parte dos especialistas - uma 'música que seja música' pressupõe também uma métrica rítmica e organizações de alturas que formem melodias e harmonias como explicadas na seção 2.3. A música do século XX ampliou esta concepção tradicional de música. Isso ocorreu na música de concerto, especialmente nas correntes concreta, eletrônica e eletroacústica. Já na década de 90 era evidente que também a música popular, especialmente as músicas eletrônicas de dança, tinham incorporado sons sem altura definida e organizações temporais fora de métricas simples. Mesmo assim, a nota permanece paradigmática como 'unidade fundamental' das estruturas musicais e, na prática, pode se desdobrar em sons que contemplam estes desenvol-

vimentos recentes. A definição e expansão da nota como unidade fundamental da música são abordados nas seções 2.2 e 2.1, respectivamente. A seção 2.3 trata da organização das notas em estruturas de alto nível.(7–11)

A teoria musical engloba assuntos tão diversos quanto psico-acústica, manifestações culturais e formalismos. O texto do capítulo 2 aborda estes assuntos mediante necessidade e assinala complementos externos.(12–14)

1.3 Implementação computacional

Os resultados apresentados desta dissertação incluem *scripts*, i.e. pequenos programas para melhor disponibilidade e validação das tecnologias. Estes constituem a caixa de ferramentas MASSA, disponibilizada em domínio público através de repositórios Git abertos.(15) Os *scripts* estão em Python e fazem uso das bibliotecas externas Numpy e Scikits/Audiolab que realizam chamadas à linguagem Fortran para maior eficiência computacional. Parte deste código foi transcrita para JavaScript e Python nativos com facilidade, o que aponta para um uso destas contribuições em navegadores como o Firefox e o Chromium.(16–19)

Estas tecnologias são todas abertas, i.e. estão publicadas em licenças que permitem o uso, cópia, distribuição e utilização de quaisquer partes para estudo e geração de produtos derivados. Desta forma, o trabalho aqui descrito está disponível e facilita os processos de co-autoria².

² A comunidade e movimento chamada 'Open Source' entende a publicação de código computacional (e outras tecnologias) em licenças abertas como uma vantagem pragmática que facilita o desenvolvimento de *software* e apresenta vantagens pedagógicas e mercadológicas. A comunidade e movimento chamada 'Free Software' engloba este entendimento, mas adiciona a abordagem filosófica da liberdade e compartilhamento, dando ênfase a isso. Ambas as correntes reforçam o entendimento de que o código computacional é o 'bem mais precioso produzido atualmente' pois consiste em tecnologia condensada, reativa (executa, processa ou gera resultados), modular (partes são copiadas e reutilizadas eficientemente) e replicada sem custo adicional (a cópia de texto tem custo baixíssimo).(20, 21)

1.4 Objetivos

O objetivo principal desta dissertação é apresentar de forma unificada relações entre elementos básicos da música e as sequências amostrais do áudio PCM. O capítulo seguinte é um texto conciso em que os elementos musicais são apresentados junto às amostras temporais resultantes. Para validação e compartilhamento, as implementações em código computacional destas relações e de pequenas peças musicais³ foram reunidas em uma *toolbox* chamada MASSA e disponibilizadas *online* e parcialmente nos Apêndices A e B.

Dos objetivos secundários, destaca-se a difusão da compreensão do código computacional através de práticas lúdicas, no caso a música. Outro objetivo considerado é a apresentação de um arcabouço de síntese sonora e musical com controle amostral, para o qual há potenciais usos em experimentos psico-acústicos e síntese em alta definição (*hi-fi*). Também é considerada a apresentação destes conteúdos de forma didática, quase um tutorial, o que possibilita compreensão e uso facilitados. Esta exposição amistosa faz-se significativa pois os assuntos tratados são de reconhecida complexidade: processamento de sinais, música, psico-acústica, para citar somente alguns exemplos. Deste ponto de vista pedagógico, também se presta a apresentação destes resultados na forma de hipertexto, em que cada *script* e exemplo sonoro/musical seja acessível junto ao material teórico.

1.5 Trabalhos relacionados

Dado o interesse humano pela música e a multidisciplinaridade inerente a esta dissertação, os trabalhos relacionados são numerosos. Assim, o Apêndice G é dedicado aos livros e implementações computacionais de interesse ou que apresentem similaridades com a descrição de elementos musicais em termos do áudio digital. Não há ênfase em artigos pois foram poucos os

³ Ou, de forma menos pretensiosa, montagens musicais, sequências musicais.

encontrados. A visita indica aspectos inéditos deste trabalho, em especial a descrição analítica de elementos musicais básicos em termos das amostras sonoras e a descrição natural, formal e concisa de técnicas tradicionais da música.

2 *Música no som digitalizado*

'The increasing dominance of graphic interfaces for music software obscured the continuing presence of the command-line tradition, the code writer, the hacker. The code writing of deferred time computer programming may be assembled out of time order, debugged and optimized.'

Simon Emmerson, Living electronic music.(22)

2.1 **Caracterização da nota musical em tempo discreto**

Em diversos contextos artísticos e teóricos, a música é pensada através de unidades chamadas notas e estas unidades compreendidas como "átomos" constituintes da música.(7, 8, 23) Hoje, estas notas são tidas como um paradigma de proposta musical e, de um ponto de vista cognitivo, como discretizações que facilitam e enriquecem o fluxo de informação através da música.(3, 11) Canonicamente, as notas possuem ao menos duração, volume, altura e timbre.(11) Estas são qualidades tratáveis quantitativamente e ditadas pelas amostras, igualmente espaçadas no tempo, da onda sonora (veja seção 1.1 sobre áudio PCM).(3)

Todas as relações desta seção estão no Apêndice A.1, as montagens musicais *Quadros sonoros* e *Reduced-fi* estão nos Apêndices B.1.1 e B.1.2. Estas implementações estão também disponíveis online como parte do *toolbox* MASSA.(2)

2.1.1 Duração

A frequência (ou taxa) de amostragem f_a é definida como o número de amostras por segundo. Seja a sequência $T_i = \{t_i\}$ um conjunto ordenado de amostras reais separadas por $\delta_a = 1/f_a$ segundos. Uma nota musical de duração Δ se apresenta como uma sequência de $\lfloor \Delta \cdot f_a \rfloor$ amostras¹:

$$T_i^\Delta = \{t_i\}_{i=0}^{\lfloor \Delta \cdot f_a \rfloor - 1} \quad (2.1)$$

Seja $\Lambda = \lfloor \Delta \cdot f_a \rfloor$ o número de amostras da sequência, de forma que $T_i = \{t_i\}_0^{\Lambda-1}$.

2.1.2 Volume

A sensação de volume sonoro depende da reverberação e distribuição dos harmônicos, dentre outras características trabalhadas na seção 2.2. Pode-se obter variações do volume através da potência da onda (24):

$$pot(T_i) = \frac{\sum_{i=0}^{\Lambda-1} t_i^2}{\Lambda} \quad (2.2)$$

O volume final dependerá sempre da amplificação do sinal nos alto-falantes, assim o crucial é a potência relativa de uma nota em relação às outras ou de um trecho da música em relação ao resto. As diferenças de volume são medidas em decibels, e estes são calculados diretamente com as amplitudes através das energias ou potências²:

$$V_{dB} = 10 \log_{10} \frac{pot(T'_i)}{pot(T_i)} \quad (2.3)$$

¹ O limite superior de uma sequência é um número natural, mas $\Delta \cdot f_a$ só satisfaz esta condição em casos muito excepcionais. É necessário escolher um inteiro próximo de $\Delta \cdot f_a$ e admitir algum erro. Por simplicidade, será considerada sempre a parte inteira da multiplicação, descrita por $\lfloor \Delta \cdot f_a \rfloor$ e aceito o erro de até $-\delta_a$ segundos. Por exemplo, $\delta_a = 1/44100 \approx 2,3 \cdot 10^{-5}$, por volta de 23 microssegundos, o que é razoável para usos musicais.

² Lembrando que, devido à percepção logarítmica, em um som de volume v a redução da potência p para uma mesma fração $v \cdot p$ com $v \in [0, 1]$ é sentido como a mesma diminuição κ do volume $v - \kappa$ com $\kappa \geq 0$.

A quantidade V_{dB} possui a unidade decibel (dB). A cada 10 dB se atribui a sensação de "volume dobrado". Referências úteis são os 10 dB por grado na escala de intensidades: *pianissimo*, *piano*, *mezzoforte*, *forte* e *fortissimo*. Valores ainda mais cruciais são equivalentes em dB de se dobrar a amplitude ou a potência:

$$\text{se } t'_i = 2.t_i \Rightarrow \text{pot}(T'_i) = 4.\text{pot}(T_i) \Rightarrow V'_{dB} = 10\log_{10}4 \approx 6\text{ dB} \quad (2.4)$$

$$\text{se } \text{pot}(T'_i) = 2\text{pot}(T_i) \Rightarrow V'_{dB} = 10\log_{10}2 \approx 3\text{ dB} \quad (2.5)$$

e o ganho de amplitude necessário para que uma sequência tenha o volume dobrado (10 dB a mais):

$$10\log_{10}\frac{\text{pot}(T'_i)}{\text{pot}(T_i)} = 10 \Rightarrow \sum_{i=0}^{\lfloor \Delta.f_a \rfloor - 1} t_i'^2 = 10 \sum_{i=0}^{\Lambda-1} t_i^2 = \sum_{i=0}^{\Lambda-1} (\sqrt{10}.t_i)^2 \quad (2.6)$$

$$\therefore t'_i = \sqrt{10}t_i \Rightarrow t'_i \approx 3,16t_i \quad (2.7)$$

Ou seja, é necessário pouco mais que triplicar a amplitude para um volume dobrado. Estes valores servem de guia para os aumentos e diminuições dos valores absolutos que compõem as sequências de amostras sonoras com propósitos musicais. A conversão direta de decibels em ganho ou atenuação de amplitude se dá da seguinte forma:

$$A = 10^{\frac{V_{dB}}{20}} \quad (2.8)$$

Onde A é o fator multiplicativo que relaciona as amplitudes do sinal antes e depois da amplificação.

2.1.3 Altura

Recapitulando, a partícula musical (nota) é uma sequência T_i cuja duração e volume correspondem ao tamanho da sequência e amplitude de suas amostras. A altura é especificada pela frequência fundamental f_0 com ciclo de duração $\delta_{f_0} = 1/f_0$. Esta duração multiplicada pela frequência de amostragem f_a resulta no número de amostras do ciclo $\lambda_{f_0} = f_a \cdot \delta_{f_0} = f_a/f_0$.

Por motivos didáticos, seja f_0 tal que divida f_a e λ_{f_0} resulte inteiro. Se T_i^f é uma sequência sonora de frequência fundamental f , então:

$$T_i^f = \{t_i^f\} = \left\{t_{i+\lambda_f}^f\right\} = \left\{t_{i+\frac{f_a}{f}}^f\right\} \quad (2.9)$$

Na seção seguinte serão contempladas frequências f que não dividem f_a e esta restrição não implica na perda de generalidade do conteúdo desta seção.

2.1.4 Timbre

Enquanto o período da onda corresponde a uma frequência fundamental, o percurso da onda sonora dentro do período - chamado de forma de onda - define um espectro harmônico e portanto um timbre³. Musicalmente, importa que espectros sonoros com diferenças mínimas resultam em timbres com diferenças expressivamente cruciais e que, portanto, pode-se produzir timbres diferentes através de espectros diferentes.(3)

O caso mais simples (e mais importante, como mostra o texto que segue) é o do espectro que consiste somente em sua própria fundamental f . Este é o caso da senoide, frequência em

³ O timbre é uma característica subjetiva e complexa. Fisicamente, o timbre é multidimensional e dado pelo comportamento temporalmente dinâmico de energias em componentes espectrais tanto harmônicas quanto ruidosas. Além disso, a palavra *timbre* é utilizada para designar coisas diferentes: uma mesma nota possui diferentes timbres, um mesmo instrumento possui diferentes timbres, dois instrumentos da mesma família possuem o mesmo timbre que a caracteriza mas possuem timbres diferentes porque são instrumentos diferentes. Vale salientar que nem tudo o que se atribui ao timbre se acha manifesto em diferenças espectrais e que até aspectos culturais ou circunstanciais alteram nossa percepção do timbre.

movimento oscilatório puro chamado movimento harmônico simples. Seja S_i^f uma sequência cujas amostras s_i^f descrevem uma senoide de frequência f :

$$S_i^f = \{s_i^f\} = \left\{ \sin\left(2\pi \frac{i}{\lambda_f}\right) \right\} = \left\{ \sin\left(2\pi f \frac{i}{f_a}\right) \right\} \quad (2.10)$$

Onde $\lambda_f = \frac{f_a}{f} = \frac{\delta_f}{\lambda_a}$ é o número de amostras do período⁴.

De forma semelhante, outras formas de onda são utilizadas na música por suas qualidades espectrais e simplicidade. Enquanto a senoide é um ponto isolado no espectro, estas ondas apresentam cadeias de componentes harmônicas. As formas de onda especificadas nas equações 2.10, 2.11, 2.12 e 2.13 estão na figura 2.1. São as formas de onda artificiais tradicionalmente usadas na música para síntese e controle oscilatório de variáveis e apresentam diversos usos também fora da música.(25)

A dente de serra apresenta todas as componentes da série harmônica com energia decrescente de $-6dB/oitava$. A sequência de amostras temporais pode ser descrita da seguinte forma:

$$D_i^f = \{d_i^f\} = \left\{ 2 \frac{i \% \lambda_f}{\lambda_f} - 1 \right\} \quad (2.11)$$

A forma de onda triangular apresenta somente os harmônicos ímpares caindo a $-12dB/oitava$:

$$T_i^f = \{t_i^f\} = \left\{ 1 - \left| 2 - 4 \frac{i \% \lambda_f}{\lambda_f} \right| \right\} \quad (2.12)$$

A onda quadrada apresenta somente os harmônicos ímpares caindo a $-6dB/oitava$:

$$Q_i^f = \{q_i^f\} = \begin{cases} 1 & \text{para } (i \% \lambda_f) < \lambda_f/2 \\ -1 & \text{caso contrário} \end{cases} \quad (2.13)$$

A dente de serra é um ponto de partida comum para a síntese subtrativa pois possui ambos os

⁴ Neste ponto já se tem toda a base para música *Reduced-fi* do Apêndice B.1.2.

harmônicos pares e ímpares e em grande quantidade. Para fins musicais, estas formas de onda são excessivamente ricas em harmônicos agudos e uma filtragem atenuante nos médios e agudos é útil para que o som ganhe naturalidade e fique mais agradável. Os harmônicos relativamente atenuados da onda triangular a fazem a mais funcional - dentre as citadas - para ser usada sem nenhum tratamento na síntese de notas musicais.

Já a onda quadrada pode ser usada na síntese subtrativa que vise a imitar um clarinete. Este instrumento também só apresenta os componentes ímpares do espectro harmônico e a onda quadrada convém com sua energia abundante nas altas frequências.

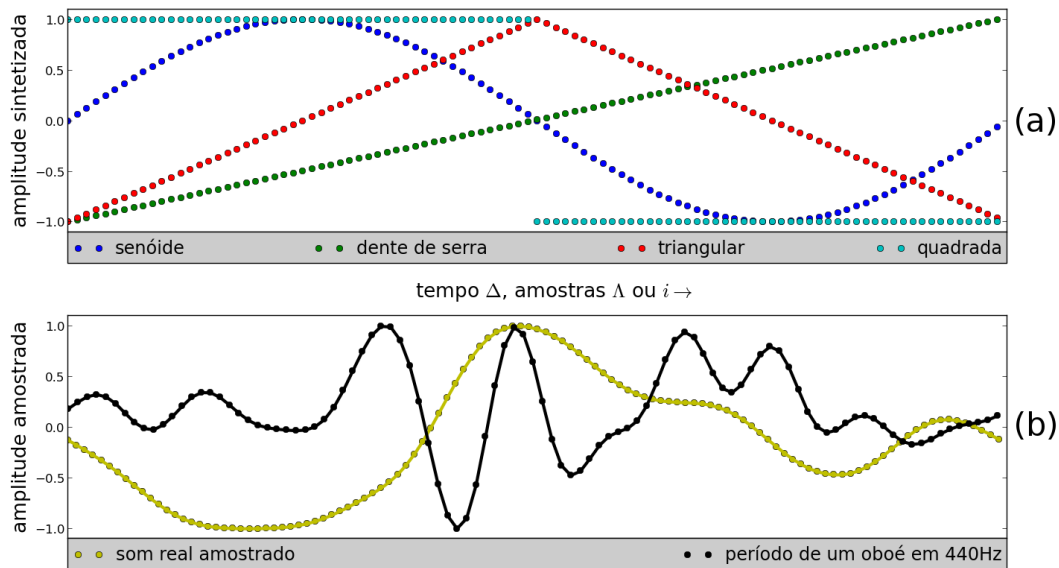


Figura 2.1 – Formas de onda musicais básicas. As formas de onda sintéticas estão em (a) e as formas de onda reais estão em (b).

A figura 2.1 apresenta as formas de onda descritas nas equações 2.10, 2.11, 2.12 e 2.13 para $\lambda_f = 100$ (período de 100 amostras). Se $t_a = 44,1kHz$, como no padrão PCM de *Compact Disks*, a onda possui frequência fundamental $f = \frac{f_a}{\lambda_f} = \frac{44100}{100} = 441 \text{ Herz}$. Um lá⁵ seja qual for a forma de onda.

⁵ Um lá 4, logo acima do dó central, no segundo espaço do pentagrama na clave de sol comum.

O espectro de cada forma de onda básica está na figura 2.2. As componentes isoladas e exatamente harmônicas dos espectros correspondem a um período rigorosamente fixo. A senoide consiste de um nódulo único no espectro, frequência pura. A dente de serra é a única com a série harmônica completa (pares e ímpares). Já as ondas triangular e quadrada possuem as mesmas componentes espectrais, mas com decaimentos de $-12\text{dB}/\text{oitava}$ e $-6\text{dB}/\text{oitava}$ respectivamente.

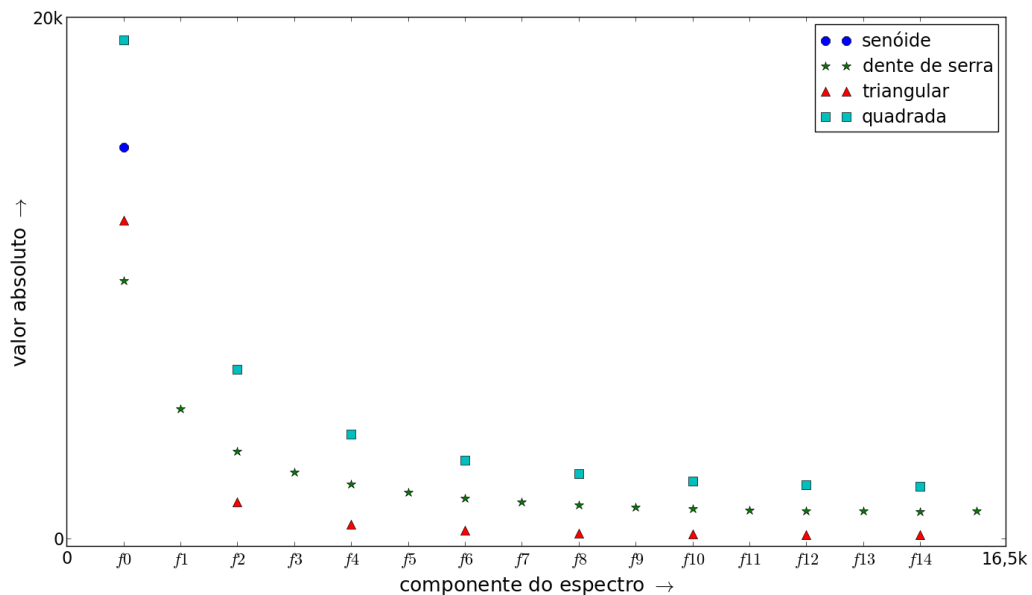


Figura 2.2 – Espectros das ondas sonoras musicais artificiais básicas. A senoide tem o espectro puntual, a triangular apresenta somente os harmônicos ímpares, caindo a 6dB por oitava; a onda quadrada tem somente os harmônicos ímpares, caindo a 12dB por oitava; a onda dente de serra apresenta todos os harmônicos, caindo a 6dB por oitava.

O espectro harmônico é formado pelas frequências múltiplas da frequência fundamental $f_n = (n + 1) \cdot f_0$. Como a percepção humana de altura segue uma progressão geométrica de frequências, o espectro possui notas diferentes da frequência fundamental. Além disso, o número de harmônicos será limitado pela frequência máxima $f_a/2$ (pelo Teorema de Nyquist).

Musicalmente crucial aqui é internalizar que a presença de energia em uma componente de frequência f_n significa uma oscilação na constituição do som, puramente harmônica e naquela

frequência f_n . Esta energia concentrada especificamente na frequência f_n é separada pelo ouvido para adentrar em um nível cognitivo de processamento⁶. As componentes senoidais são geralmente as principais responsáveis pela qualidade chamada timbre. Caso não se apresentem em proporções harmônicas (relações de pequenos números), o som é percebido como ruidoso ou dissonante e não com uma sonoridade de frequência fundamental estabelecida de forma unívoca. Além disso, a noção de altura absoluta em um complexo sonoro é baseada na semelhança do espectro com a série harmônica.(3)

No caso de uma forma de onda fixa e de tamanho fixo, o espectro é sempre harmônico e estático. Cada forma de onda é composta de proporções específicas das componentes harmônicas e quanto maior a curvatura do trecho na forma de onda, maior a contribuição do trecho para a concentração de energia nos harmônicos agudos. Pode-se constatar isso em sons reais. A onda rotulada como “som real amostrado” na figura 2.1 é um período de $\lambda_f = 114$ amostras extraído de um som real relativamente comportado. A onda de oboé foi amostrada de um lá 4 também em $44,1kHz$. O período escolhido para a amostragem é relativamente curto, com 98 amostras corresponde a uma frequência de $\frac{44100}{98} = 450Hz$. Pode-se perceber, através das curvaturas, o espectro rico em frequências agudas do oboé e o espectro mais grave do som real.

A sequência $R_i = \{r_i\}_0^{\lambda_f-1}$ de amostras do som real da figura 2.1 pode ser tomada como base para um som T_i^f da seguinte forma:

$$T_i^f = \{t_i^f\} = \{r_{(i \% \lambda_f)}\} \quad (2.14)$$

O som resultante possui o espectro momentâneo do som original. Por ser repetido de forma idêntica, seu espectro é perfeitamente harmônico, sem os ruídos e variações típicas do fenômeno natural. Isso pode ser visto na figura 2.3, que mostra os espectros da nota original do

⁶ Esta separação em frequência é realizada por diversas espécies através de mecanismos similares à cóclea humana.(3)

oboé e de uma nota artificial de mesma duração e cujas amostras consistem no mesmo período da figura 2.1. O espectro natural possui variações nas frequências dos harmônicos, nas suas intensidades e uma quantidade de ruído. Já a nota cujo período foi amostrado possui espectro perfeitamente harmônico.

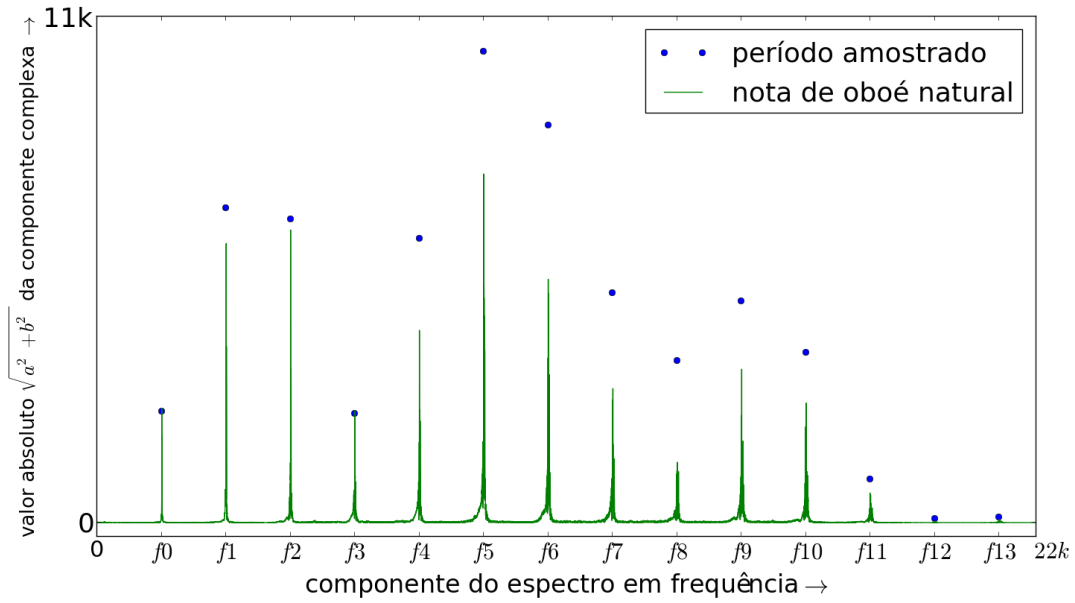


Figura 2.3 – Espectros das ondas sonoras de uma nota de oboé natural e de período amostrado. O som natural possui flutuações nos harmônicos e ruídos, já o som de período amostrado possui espectro perfeitamente harmônico.

2.1.5 O espectro no som amostrado

A presença e comportamento destas componentes senoidais no som discretizado possui particularidades. Considere um sinal T_i e sua decomposição de Fourier $\mathcal{F}\langle T_i \rangle = C_i = \{c_i\}_0^{\Lambda-1}$. A recomposição é a soma das componentes frequenciais em amostras temporais⁷:

$$t_i = \frac{1}{\Lambda} \sum_{k=0}^{\Lambda-1} c_k e^{j \frac{2\pi k}{\Lambda} i} = \frac{1}{\Lambda} \sum_{k=0}^{\Lambda-1} (a_k + j.b_k) [\cos(w_k i) + j.\text{sen}(w_k i)] \quad (2.15)$$

⁷ Lembrando que o fator $\frac{1}{\Lambda}$ pode ser distribuído dentre a transformada e a reconstrução como preferir.

Onde $c_k = a_k + j.b_k$ dita a amplitude e fase de cada frequência: $w_k = \frac{2\pi}{\Lambda}k$ em radianos ou $f_k = w_k \frac{f_a}{2\pi} = \frac{f_a}{\Lambda}k$ em Hertz, com atenção para os respectivos limites em π e em $\frac{f_a}{2}$ dados pelo Teorema de Nyquist.

No caso específico de um sinal sonoro, as amostras t_i são reais e dadas pela parte real da equação 2.15:

$$\begin{aligned} t_i &= \frac{1}{\Lambda} \sum_{k=0}^{\Lambda-1} [a_k \cos(w_k i) - b_k \sin(w_k i)] \\ &= \frac{1}{\Lambda} \sum_{k=0}^{\Lambda-1} \sqrt{a_k^2 + b_k^2} \cos \left[w_k i - \tan^{-1} \left(\frac{b_k}{a_k} \right) \right] \end{aligned} \quad (2.16)$$

A equação 2.16 mostra que o termo imaginário de c_k acrescenta uma fase à senoide real, i.e. os termos imaginários b_k da decomposição espectral por Fourier proporcionam a varredura de fase $\left[-\frac{\pi}{2}, +\frac{\pi}{2}\right]$ dada pelo termo $\tan^{-1} \left(\frac{b_k}{a_k} \right)$ que possui esta imagem. O sinal de a_k especifica o lado direito ou esquerdo do círculo trigonométrico, o que completa a varredura de fase: $\left[-\frac{\pi}{2}, +\frac{\pi}{2}\right] \cup \left[\frac{\pi}{2}, \frac{3\pi}{2}\right] \equiv [2\pi]$.

A figura 2.4 exhibe duas amostras e as componentes espectrais que contêm. A decomposição de Fourier possui neste caso um único par de coeficientes $\{c_k = a_k - j.b_k\}_0^{\Lambda-1=1}$ relativos às frequências $\{f_k\}_0^1 = \left\{w_k \frac{f_a}{2\pi}\right\}_0^1 = \left\{k \frac{f_a}{\Lambda=2}\right\}_0^1 = \left\{0, \frac{f_a}{2} = f_{\text{máx}}\right\}$ com energias $e_k = \frac{(c_k)^2}{\Lambda=2}$. O papel das amplitudes a_k é nítido com $\frac{a_0}{2}$ o deslocamento fixo⁸ e $\frac{a_1}{2}$ a amplitude da oscilação em si, dada pela relação $f_k = k \frac{f_a}{\Lambda=2}$. Este caso é de especial importância pois o mínimo necessário para representar uma oscilação são 2 amostras e disso resulta a frequência de Nyquist $f_{\text{máx}} = \frac{f_a}{2}$. Esta é a frequência máxima presente em um som amostrado com f_a amostras por segundo⁹.

Todas as sequências fixas T_i de apenas 3 amostras também apresentam somente 1 frequência, pois sua primeira harmônica usaria 1,5 amostras e ultrapassa o limite inferior de 2 amostras mí-

⁸ Chamado de *bias* ou *offset*.

⁹ Qualquer sinal amostrado possui esta característica, não somente o som digitalizado.

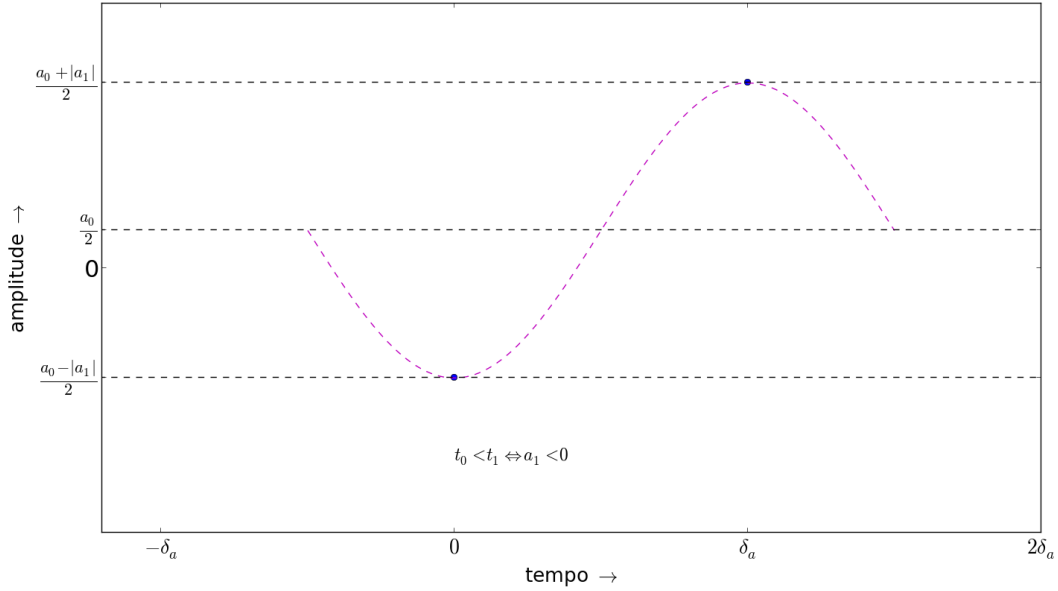


Figura 2.4 – Oscilação de 2 amostras (frequência máxima em qualquer f_a). O primeiro coeficiente reflete o deslocamento (offset ou bias) e o segundo coeficiente especifica a amplitude da oscilação.

nimas, i.e. a frequência da harmônica ultrapassaria a de Nyquist pois: $\frac{2 \cdot f_a}{3} > \frac{f_a}{2}$. Os coeficientes $\{c_k\}_{0}^{\Lambda-1=2}$ apresentam-se em 3 componentes frequenciais. Uma delas é relativa à frequência zero (c_0), as outras duas (c_1 e c_2) contribuem de forma igual na reconstrução da senoide com $f = f_a/3$.

Λ amostras reais t_i resultam em Λ coeficientes complexos $c_k = a_k + j.b_k$. Os coeficientes c_k se equivalem dois a dois correspondendo às mesmas frequências e com contribuições idênticas¹⁰. Lembrando que $f_k = k \frac{f_a}{\Lambda}$, $k \in \{0, \dots, \lfloor \frac{\Lambda}{2} \rfloor\}$. Quando $k > \frac{\Lambda}{2}$, a frequência f_k é espelhada em $\frac{f_a}{2}$ da seguinte forma $f_k = \frac{f_a}{2} - (f_k - \frac{f_a}{2}) = f_a - f_k = f_a - k \frac{f_a}{\Lambda} = (\Lambda - k) \frac{f_a}{\Lambda} \Rightarrow f_k \equiv f_{\Lambda-k}$, $\forall k < \Lambda$.

O mesmo pode ser observado com $w_k = f_k \cdot \frac{2\pi}{f_a}$ e lembrando da periodicidade 2π , que resulta em $w_k = -w_{\Lambda-k}$. Como o cosseno é uma função par e a tangente inversa é ímpar, as componentes em w_k e $w_{\Lambda-k}$ se somam na equação de reconstrução das amostras reais disposta na equação 2.15.

¹⁰ Parte real igual e imaginária com sinal trocado: $a_{k1} = a_{k2}$ e $b_{k1} = -b_{k2}$. Como consequência os módulos são iguais e as fases possuem sinais opostos

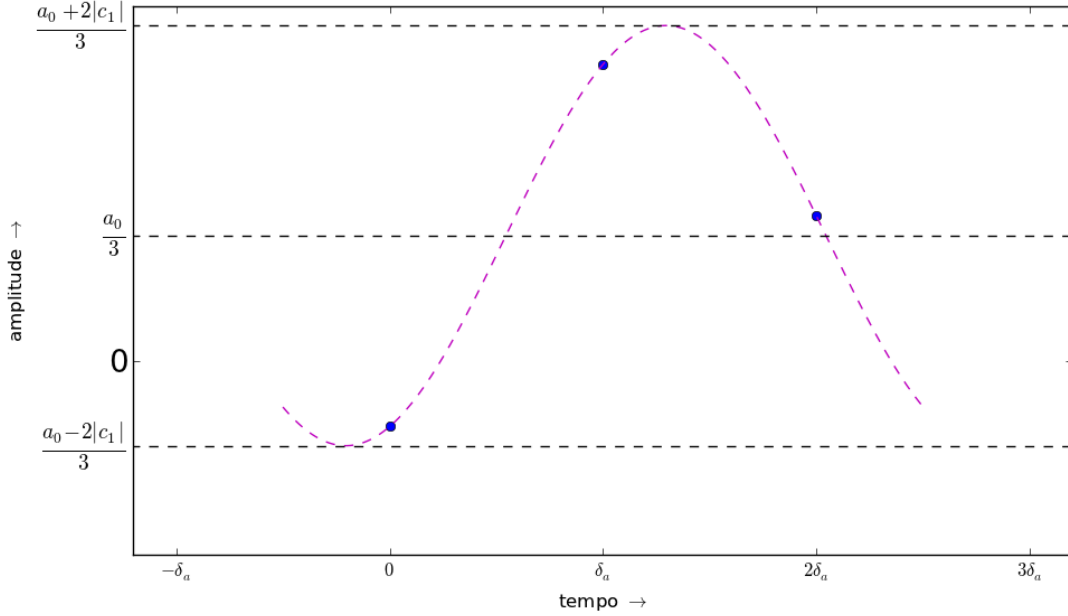


Figura 2.5 – 3 amostras fixas apresentam uma só frequência não nula. $c_1 = c_2^*$ e $w_1 \equiv w_2$.

Ou seja, em uma decomposição de Λ amostras, as Λ componentes frequenciais $\{c_i\}_0^{\Lambda-1}$ resultantes são equivalentes em pares. Exceção para f_0 e, no caso de Λ ser par, de $f_{\Lambda/2} = f_{\text{máx}} = \frac{f_a}{2}$, ambas as componentes são isoladas, i.e. não existe outra componente na frequência f_0 ou $f_{\Lambda/2}$ (se Λ par) além dela mesma. Pois $f_{\Lambda/2} = f_{(\Lambda-\Lambda/2)=\Lambda/2}$ e $f_0 = f_{(\Lambda-0)=\Lambda} = f_0$. Além disso, estas duas frequências (a frequência zero e a frequência máxima) não são representadas com variação de fase e, portanto, são estritamente reais. Assim, pode-se concluir que o número τ de pares de coeficientes equivalentes é:

$$\tau = \frac{\Lambda - \Lambda \% 2}{2} + \Lambda \% 2 - 1 \quad (2.17)$$

e ficam evidentes as equivalências 2.18, 2.19 e 2.20:

$$f_k \equiv f_{\Lambda-k}, \quad w_k \equiv -w_{\Lambda-k}, \quad \forall \quad 1 \leq k \leq \tau \quad (2.18)$$

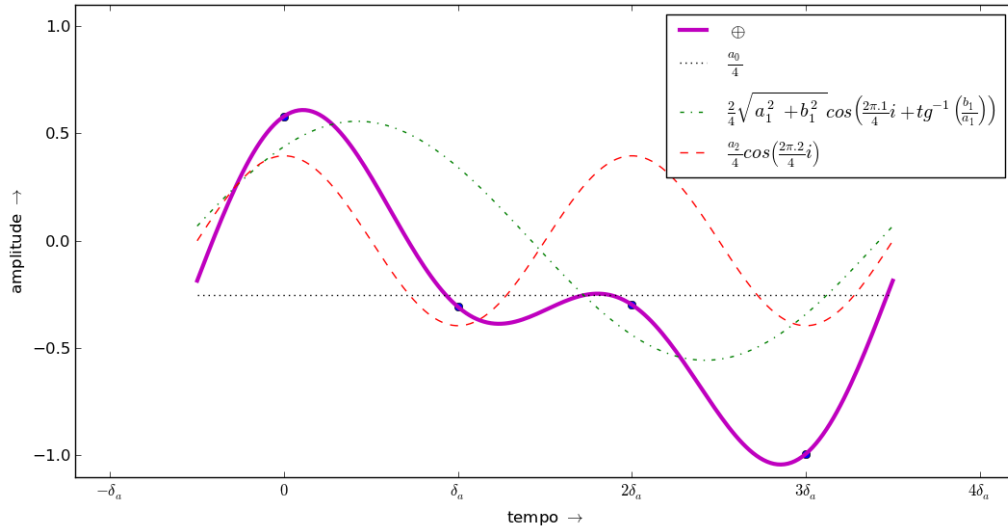


Figura 2.6 – Componentes frequenciais em 4 amostras.

Como $a_k = a_{\Lambda-k}$ e $b_k = -b_{\Lambda-k}$:

$$\sqrt{a_k^2 + b_k^2} = \sqrt{a_{\Lambda-k}^2 + b_{\Lambda-k}^2}, \quad \forall \quad 1 \leq k \leq \tau \quad (2.19)$$

$$tg^{-1}\left(\frac{b_k}{a_k}\right) = -tg^{-1}\left(\frac{b_{\Lambda-k}}{a_{\Lambda-k}}\right), \quad \forall \quad 1 \leq k \leq \tau \quad (2.20)$$

Com $k \in \mathbb{N}$.

A observação da equação de reconstrução para o sinal real 2.16 em conjunto com as equivalências dos módulos e fases 2.19 e 2.20, o número de coeficientes pareados 2.17 e equivalência de pares de frequências 2.18 expõe o caso geral da combinação das componentes em cada amostra t_i :

$$t_i = \frac{a_0}{\Lambda} + \frac{2}{\Lambda} \sum_{k=1}^{\tau} \sqrt{a_k^2 + b_k^2} \cos\left[w_k i - tg^{-1}\left(\frac{b_k}{a_k}\right)\right] + \frac{a_{\Lambda/2}}{\Lambda} \cdot (1 - \Lambda \% 2) \quad (2.21)$$

Assim, a exemplo da figura 2.5, a transformada de Fourier de 3 amostras possui 2 coeficientes

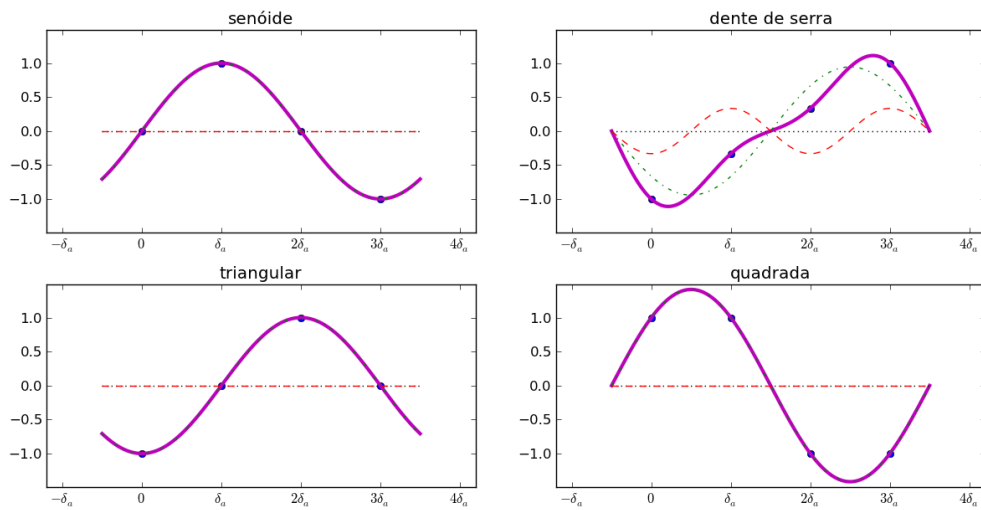


Figura 2.7 – Formas de onda básicas em 4 amostras.

frequenciais com quantidades iguais de energia na mesma frequência.

Com 4 amostras, pode-se representar 1 ou 2 frequências em proporções quaisquer. A figura 2.6 mostra uma forma de onda de 4 amostras e suas duas componentes. As contribuições individuais se somam na forma de onda original, e uma breve inspeção revela que as curvaturas maiores são fruto da frequência mais aguda, enquanto um deslocamento fixo da somatória das componentes advém da componente na frequência zero.

A figura 2.7 explicita os harmônicos em 4 amostras nas formas de onda básicas das equações 2.10, 2.11, 2.12 e 2.13. Todas consistem em apenas 1 senoíde, com exceção da dente de serra que possui os harmônicos pares.

A figura 2.8 mostra uma decomposição senoidal para o caso de 6 amostras e a figura 2.9 decompõe as formas de onda básicas. Neste caso todas as ondas se diferenciam no espectro: as quadrada e triangular possuem as mesmas componentes, mas em proporções diferentes, já a dente de serra possui uma componente a mais.

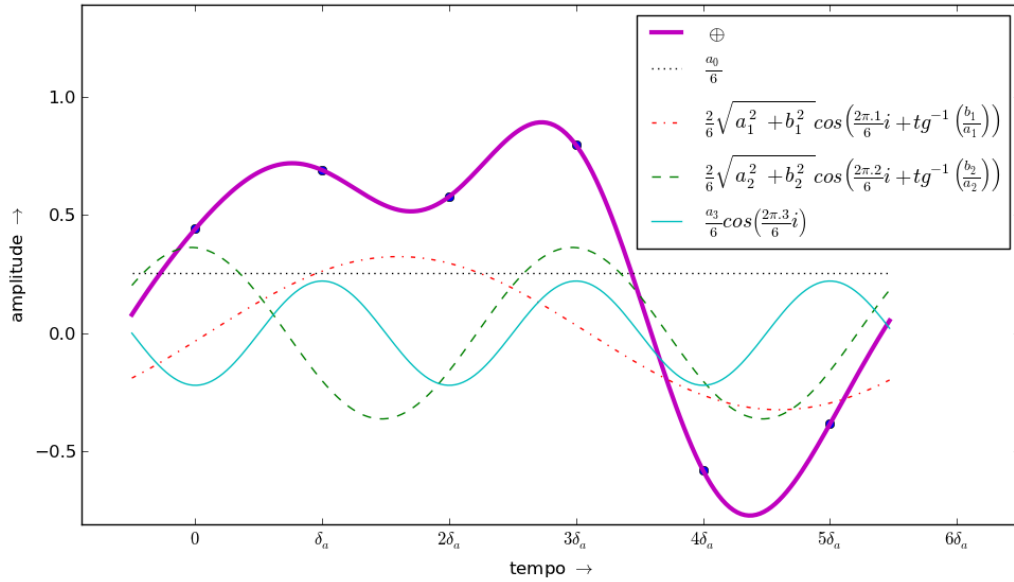


Figura 2.8 – Componentes frequenciais em 6 amostras: 3 senóides se somam ao bias.

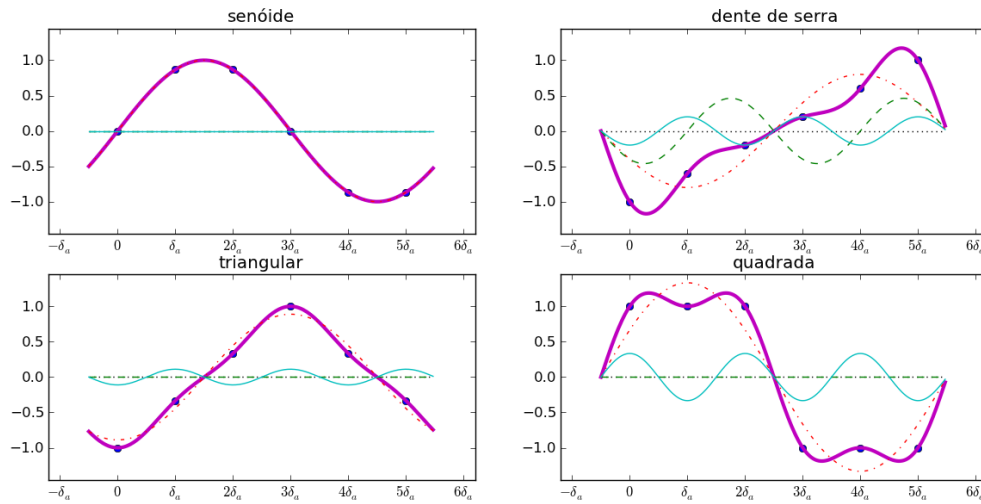


Figura 2.9 – Formas de onda básicas em 6 amostras: as ondas triangular e quadrada possuem os harmônicos ímpares, mas em proporções e fases diferentes; a dente de serra possui também o harmônico par.

2.1.6 A nota básica

Seja f tal que f divida f_a ¹¹. Uma sequência T_i de amostras sonoras separadas por $\delta_a = 1/f_a$ descreve uma nota musical de frequência f Hertz e duração Δ segundos se, e somente se, possuir

¹¹ Como apontado anteriormente, esta limitação facilita a exposição sem perda de generalidade. A limitação será superada no início da próxima seção.

a periodicidade $\lambda_f = f_a/f$ e tamanho $\Lambda = \lfloor f_a \cdot \Delta \rfloor$:

$$T_i^{f, \Delta} = \{t_i \% \lambda_f\}_0^{\Lambda-1} = \left\{ t_i^f \% \left(\frac{f_a}{f} \right) \right\}_0^{\Lambda-1} \quad (2.22)$$

A nota por si só não especifica um timbre. Mesmo assim, faz-se necessária a escolha de uma forma de onda para que as amostras t_i tenham um valor estabelecido individualmente. Um único período dentre as ondas básicas pode ser utilizado para a especificação da nota da seguinte forma:

$\lambda_f = \frac{f_a}{f}$ é o número de amostras do período. Seja L_i^{f, δ_f} a sequência que descreve um período da onda $L_i^f \in \{S_i^f, Q_i^f, T_i^f, D_i^f, R_i^f\}$ de duração $\delta_f = 1/f$, dadas pelas equações 2.10, 2.11, 2.12 e 2.13 e onde R_i^f é uma onda real amostrada:

$$L_i^{f, \delta_f} = \{l_i^f\}_0^{\delta_f \cdot f_a - 1} = \{l_i^f\}_0^{\lambda_f - 1} \quad (2.23)$$

Então a sequência T_i consistirá em uma nota de duração Δ e frequência f se:

$$T_i^{f, \Delta} = \{t_i^f\}_0^{\lfloor f_a \cdot \Delta \rfloor - 1} = \left\{ t_i^f \% \left(\frac{f_a}{f} \right) \right\}_0^{\Lambda-1} \quad (2.24)$$

2.1.7 Localização espacial e espacialização

Embora não seja uma das quatro qualidades básicas tradicionais de uma nota musical, esta possui sempre uma localização espacial, que é a posição da fonte que a emitiu, no espaço físico tridimensional ordinário. Além disso, há um ambiente que reverbera a nota emitida, assunto ao qual a 'espacialização' é dedicada. Ambas, a espacialização e a localização espacial, são bastante valorizadas por audiófilos e pela indústria fonográfica.(4)

Localização espacial

Acredita-se que a percepção da localização espacial do som se dê em nosso sistema nervoso através destas três informações: o atraso de chegada do som entre um ouvido e o outro, a diferença de intensidade do som direto em cada ouvido e a filtragem realizada pelo corpo, incluindo tórax, cabeça e orelhas.(3, 26, 27)

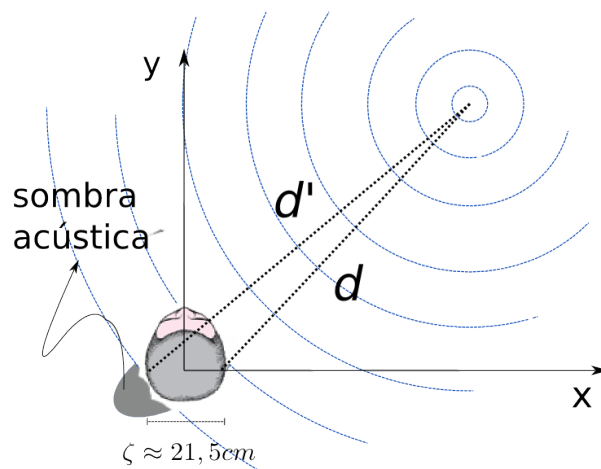


Figura 2.10 – Detecção de localização espacial de fonte sonora: esquema utilizado para cálculo da diferença de tempo interaural (DTI) e da diferença de intensidade interaural (DII).

Se consideradas somente as incidências diretas em cada ouvido, as equações são simples. Dada a separação ζ entre os ouvidos¹², um objeto localizado em (x,y) conforme a figura 2.10 está distante de cada ouvido:

$$\begin{aligned} d &= \sqrt{\left(x - \frac{\zeta}{2}\right)^2 + y^2} \\ d' &= \sqrt{\left(x + \frac{\zeta}{2}\right)^2 + y^2} \end{aligned} \quad (2.25)$$

e cálculos imediatos resultam na Diferença de Tempo Interaural:

¹² Constata-se que $\zeta \approx 21,5\text{cm}$ para um humano adulto.

$$DTI = \frac{d' - d}{v_{som\ no\ ar} \approx 343.2} \quad \text{segundos} \quad (2.26)$$

e na Diferença de Intensidade Interaural:

$$DII = 20 \log_{10} \left(\frac{d}{d'} \right) \quad \text{decibels} \quad (2.27)$$

Convertendo para amplitude, obtém-se $DII_a = \frac{d}{d'}$. A DII_a pode ser utilizada como constante multiplicativa do canal direito de um sinal sonoro estéreo: $\{t'_i\}_0^{\Lambda-1} = \{DII_a \cdot t_i\}_0^{\Lambda-1}$. Pode-se utilizar a DII junto à DTI como adiantamento no tempo do canal direito com relação ao esquerdo, vínculo crucial para a localização em sons graves e em sonoridades percussivas.(27) Considerando $\Lambda_{DTI} = \lfloor DTI \cdot f_a \rfloor$:

$$\begin{aligned} \Lambda_{DTI} &= \left\lfloor \frac{d' - d}{343,2} f_a \right\rfloor \\ DII_a &= \frac{d}{d'} \\ \{t'_{(i+\Lambda_{DTI})}\}_{\Lambda_{DTI}}^{\Lambda+\Lambda_{DTI}-1} &= \{DII_a \cdot t_i\}_0^{\Lambda-1} \\ \{t'_i\}_0^{\Lambda_{DTI}-1} &= 0 \end{aligned} \quad (2.28)$$

Com t_i o canal direito e t'_i o canal esquerdo. Caso $\Lambda_{DTI} < 0$, basta trocar t_i por t'_i e utilizar $\Lambda'_{DTI} = |\Lambda_{DTI}|$.

Embora consideravelmente simples até aqui, a localização espacial depende drasticamente de outras pistas. Pela DTI e DII especifica-se somente o ângulo horizontal (azimutal) θ dado por:

$$\theta = \tan^{-1} \left(\frac{y}{x} \right) \quad (2.29)$$

com x, y tais como representados na figura 2.10. Mesmo assim, há dificuldades quando θ in-

cide sobre o chamado "cone de confusão" em que um mesmo par de especificações DTI, DII resultam de vários dos pontos do cone. Nestes pontos, a inferência do ângulo azimutal depende especialmente da filtragem atenuante nos agudos, pois a cabeça interfere um tanto mais nas ondas mecânicas agudas do que nas graves.(26, 27) Também pertinente à audição de fonte lateral, quando o som é grave o suficiente, há uma difração e a onda chega ao ouvido $\approx 0,7ms$ depois.(4)

A figura 2.10 mostra também esta sombra acústica do crânio, importante para a percepção do ângulo azimutal da fonte no cone de confusão. O cone em si não foi disposto na figura pois não é exatamente um cone e suas dimensões precisas não foram encontradas na literatura visitada e não são facilmente concebíveis, dadas as filtrações e a difração dependente do espectro do som em si. De toda forma, o cone de confusão pode ser entendido como um cone com o ápice no meio da cabeça e saindo por cada uma das orelhas.(26)

Já a localização completa, incluindo distância e elevação da fonte sonora, é dada pela função de transferência de cabeça (HRTF - do inglês *Head Related Transfer Function*).(26) Existem bases abertas e conhecidas de HRTF como a CIPIC e pode-se aplicar estas funções de transferência em um som por convolução (veja equação 2.43).(28) O corpo do indivíduo altera bastante as filtrações realizadas e existem técnicas para gerar HRTFs que sejam - como proposta - utilizáveis de forma universal.(29)

Espacialização

Já a espacialização é o resultado das reflexões e absorções do som nas superfícies do recinto/paisagem no qual a nota foi emitida. O som se propaga no ar a $\approx 343,2m/s$, e pode ser emitido da fonte com qualquer padrão de direcionalidade. Quando uma frente sonora encontra uma superfície, há uma reflexão. Nesta reflexão ocorrem tanto 1) a inversão da componente da velocidade de propagação que é perpendicular à superfície, quanto 2) a absorção de energia, especialmente

nos agudos. As ondas se propagam até atingirem níveis inaudíveis. Quando alguma frente de onda atingir o ouvido, pode ser descrita com o momento de chegada ao ouvido e os filtros de absorção de cada superfície que atingiu. Pode-se simular reverberações não possíveis em sistemas reais. Para experimentações, pode-se usar reflexões assimétricas com relação ao eixo perpendicular à superfície, ou ainda ganhos em determinadas bandas de frequência (tidos como 'ressonâncias'), ambas as características não são encontradas em sistemas reais.

Existem algumas modelagens de reverberação menos atreladas ao cálculo de cada reflexão, exploram informações valiosas do ponto de vista auditivo. De fato, a reverberação pode ser modelada com um conjunto de 2 características temporais e no espectro:

- Primeiro período: as 'primeiras reflexões' são mais intensas e esparsas.
- Segundo período: a 'reverberação tardia' é praticamente uma sucessão densa de atrasos indistintos com um decaimento exponencial e ocorrências estatísticas.
- Primeira banda: o grave possui algumas frequências de ressonância relativamente espalhadas.
- Segunda banda: o médio e agudo possuem um decaimento progressivo e suave com flutuações estatísticas.

Smith III aponta que boas salas de concerto possuem um tempo total de reverberação de aproximadamente 1,9 segundos. Aponta também o período das primeiras reflexões de 0,1 segundos. Estas quantidades sugerem que, nas condições contempladas, há frentes de onda perceptíveis que se propagam até 652,08 metros (83,79k amostras em $f_a = 44,1kHz$) antes de atingirem o ouvido. Além disso, as reflexões do som formam, após a propagação por 34,32 metros (4,41k amostras em $f_a = 44,1kHz$), um emaranhado cujas incidências são pouco distintas na audição. Estas primeiras reflexões são particularmente importantes para a sensação de espaço. A

primeira incidência é o som direto, descrito por DTI e DII das equações 2.26 e 2.27. Admitindo que cada uma das primeiras reflexões, antes de chegar ao ouvido, se propagará, ao menos, $3 - 30m$ dependentes das dimensões da sala, a separação entre as primeiras reflexões é de, ao menos, $8 - 90$ milissegundos ($\approx 350 - 4000$ amostras em $f_a = 44.1kHz$). Verifica-se experimentalmente que o número de reflexões aumenta em proporção quadrática $\approx k.n^2$. Apontamentos do uso de convoluções e filtragens para facilitar estas implementações estão na subseção 2.2.6, especialmente nos parágrafos sobre reverberação.

2.1.8 Usos musicais

A partir da nota básica, cabe realizar estruturas musicais com sequências destas partículas. A soma dos elementos de mesmo índice de N sequências $T_{k,i} = \{t_{k,i}\}_{k=0}^{N-1}$ de mesmo tamanho Λ resulta em seus conteúdos espectrais sobrepostos em um processo de mixagem sonora:

$$\{t_i\}_0^{\Lambda-1} = \left\{ \sum_{k=0}^{N-1} t_{k,i} \right\}_0^{\Lambda-1} \quad (2.30)$$

A figura 2.11 ilustra este processo de superposição de ondas sonoras discretizadas. A figura dispõe 100 amostras, de onde pode-se concluir que, se $f_a = 44.1kHz$, as frequências da dente de serra, da onda quadrada e da senoide são, respectivamente, $\frac{f_a}{100/2} = 882Hz$, $\frac{f_a}{100/4} = 1764Hz$ e $\frac{f_a}{100/5} = 2205Hz$. A duração do trecho é bastante curto $\frac{f_a=44.1kHz}{100} \approx 2$ milissegundos. Basta completar com zeros para somar sequências de tamanhos diferentes.

As notas mixadas são em grande parte separadas pelo ouvido por leis físicas de ressonância e pelo sistema nervoso.(3) O resultado da mixagem de notas musicais é a harmonia musical, cujos intervalos entre as frequências e os acordes de notas simultâneas regem aspectos subjetivos e abstratos da música e sua apreciação.(30)

As sequências podem também ser concatenadas no tempo. Caso as sequências $\{t_{k,i}\}_0^{\Lambda_k-1}$ de

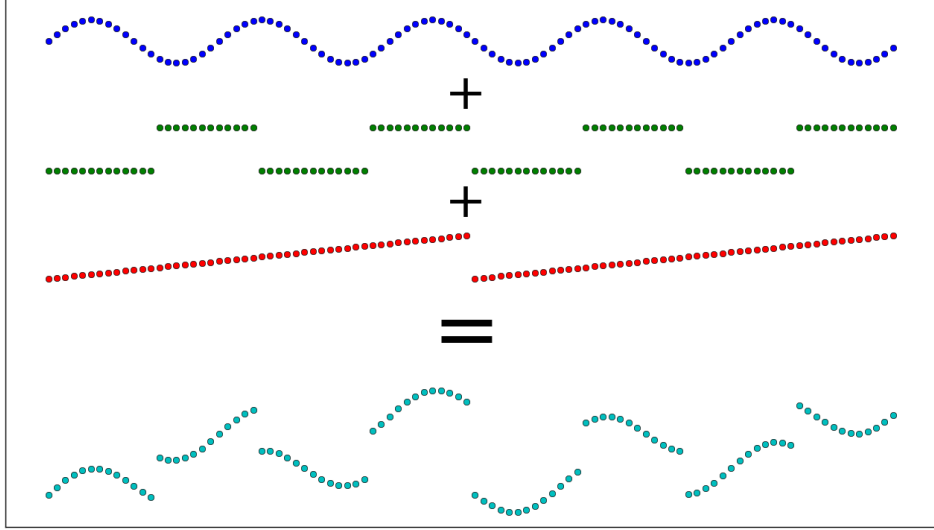


Figura 2.11 – Mixagem de três seqüências sonoras. As amplitudes são sobrepostas diretamente.

tamanhos Λ_k representem k notas musicais, sua concatenação em uma única seqüência T_i é em uma seqüência musical simples ou melodia:

$$\{t_i\}_0^{\sum \Delta_k - 1} = \{t_{l,i}\}_0^{\sum \Delta_k - 1}, \quad l \text{ menor inteiro} : \quad \Lambda_l > i - \sum_{j=0}^{l-1} \Lambda_j \quad (2.31)$$

Este mecanismo é demonstrado de forma ilustrativa na figura 2.12 com as mesmas seqüências da figura 2.11. As seqüências são curtas para as taxas de amostragem usuais, mas pode-se observar a concatenação de seqüências sonoras. Além disso, cada nota tem a duração maior que $100ms$ se $f_a < 1kHz$.

A montagem musical *reduced-fi* explora de forma isolada este uso de justaposição temporal das notas, resultando em uma peça homofônica. O princípio vertical está demonstrado nos *quadros sonoros*, sons estáticos com espectros peculiares. Ambas as peças estão em código Python nos Apêndices B.1.1 e B.1.2 e estão disponíveis como parte da *toolbox* MASSA.(2)

Está descrita a nota musical digital básica e a seção seguinte desenvolve a evolução temporal

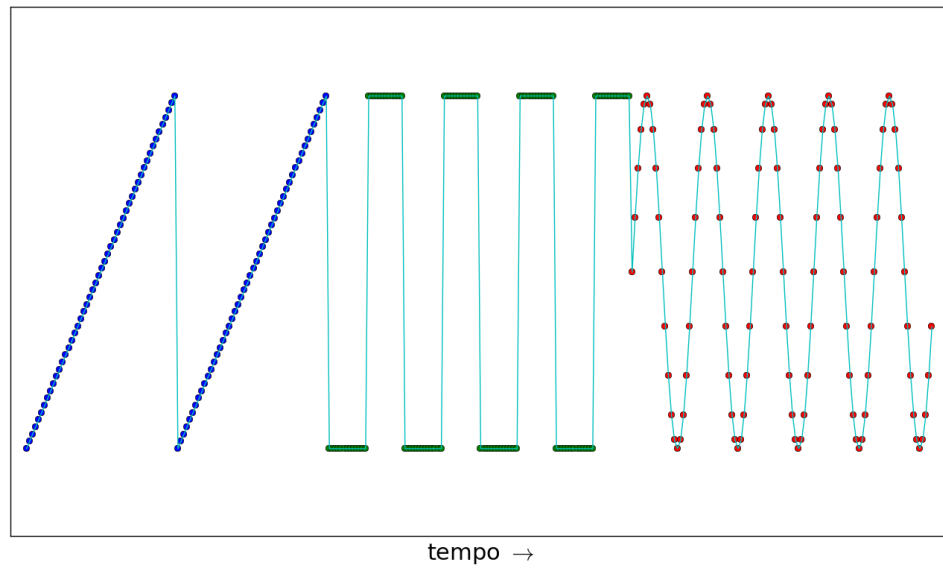


Figura 2.12 – Concatenação de três sequências sonoras através da justaposição temporal de suas amostras.

de seus conteúdos, como nos *glissandi* e nas envoltórias de volume. A filtragem de componentes espectrais e a geração dos ruídos completam a constituição da nota musical como unidade isolada e se desdobra na seção 2.3, dedicada à estruturação destas notas em música através de métricas e trajetórias.

2.2 Variações na nota musical básica

A nota musical digital básica foi definida na seção 2.1 com os parâmetros: duração, altura, intensidade (volume) e timbre. Esta é uma modelagem útil e paradigmática, mas não esgota o que se entende por uma nota musical.

Em primeiro lugar, as características da nota se modificam no decorrer da própria nota.⁽²⁴⁾ Por exemplo, uma nota de piano de 3 segundos tem a intensidade com início abrupto e decaimento progressivo, além de variações do espectro, com harmônicos que decaem antes dos outros e alguns que aparecem com o tempo. Estas variações não são obrigatórias e sim orientações da síntese sonora para usos musicais, pois é como os sons se apresentam na natureza¹³. Explorar todas as formas pelas quais estas variações ocorrem está fora do escopo de qualquer trabalho dada a considerável sensibilidade do ouvido humano e a complexidade da nossa cognição sonora. A seguir, serão apontados recursos primários para estas variações das características na nota básica. Todas as relações descritas nesta seção estão implementadas em Python no Apêndice A.2. As montagens musicais *Transita para metro*, *Vibra e treme*, *Tremolos*, *vibratos e a frequência*, *Trenzinho de caipiras impulsivos*, *Ruidosa faixa*, *Bela rugosi*, *Chorus infantil*, *ADa e SaRa* estão nos Apêndices B.2.1, B.2.2, B.2.3, B.2.4, B.2.5, B.2.6, B.2.7 e B.2.8. Estes códigos são parte da caixa de ferramentas MASSA, disponível online.⁽²⁾

2.2.1 Tabela de busca

Mais conhecida pelo termo em inglês, a *Lookup Table* (ou simplesmente LUT), é uma estrutura de dados para consultas indexadas usada frequentemente para reduzir a complexidade computacional e por permitir o uso de funções sem possibilidade de cálculo direto, como amostras recolhidas da natureza. Na música seu uso transcende estes primeiros, facilitando as operações

¹³ A regra de ouro aqui é: para que um som isolado desperte interesse por si só, faça com que tenha variações internas.⁽³⁾

e permitindo que um único período de onda possa ser usado para sintetizar sons em toda a banda de frequências audíveis, qualquer que seja a forma de onda amostrada.

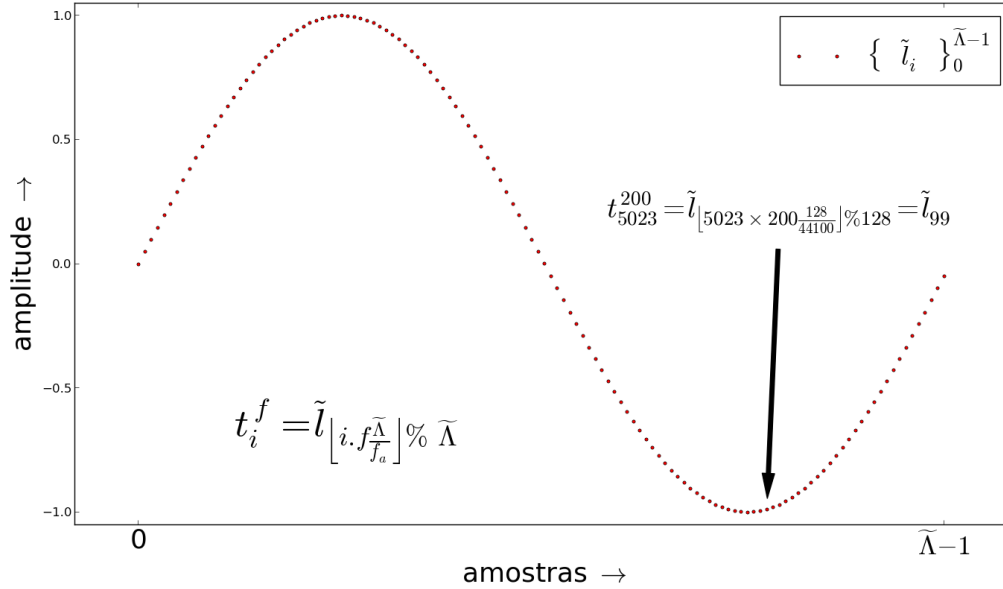


Figura 2.13 – Procedimento de busca em tabela (conhecido como Lookup Table) para síntese de sons em frequências diferentes a partir de uma única forma de onda em alta resolução.

Seja $\tilde{\Lambda}$ o tamanho do período e $\tilde{L}_i = \{\tilde{l}_i\}_0^{\tilde{\Lambda}-1}$ os elementos \tilde{l}_i de um período de onda qualquer (veja equação 2.23). Uma sequência $T_i^{f,\Delta}$ com amostras de um som de frequência f e duração Δ pode ser obtida a partir de \tilde{L}_i da seguinte forma:

$$T_i^{f,\Delta} = \{t_i^f\}_0^{\lfloor f_a \cdot \Delta \rfloor - 1} = \{\tilde{l}_{\gamma_i \% \tilde{\Lambda}}\}_0^{\Delta-1}, \quad \text{onde } \gamma_i = \left\lfloor i \cdot f \frac{\tilde{\Lambda}}{f_a} \right\rfloor \quad (2.32)$$

Ou seja, com os índices corretos ($\gamma_i \% \tilde{\Lambda}$) da LUT, pode-se sintetizar o som em qualquer frequência. A figura 2.13 ilustra o cálculo de uma amostra de $\{t_i\}$ a partir de $\{\tilde{l}_i\}$ para uma frequência de $f = 200\text{Hz}$, $\tilde{\Lambda} = 128$ e considerada a taxa de amostragem em $f_a = 44.1\text{kHz}$. Esta não é uma configuração praticável, como assinalado abaixo, mas possibilita uma disposição gráfica do procedimento.

O cálculo do inteiro γ_i introduz um ruído, e este diminui com o aumento de $\tilde{\Lambda}$. Para fins de

síntese, em $f_a = 44.1kHz$ o padrão é usar $\tilde{\Lambda} = 1024$ amostras, pois já não gera ruído relevante no espectro audível. O método de arredondamento ou interpolação não é decisivo.(31)

A expressão que define a variável γ_i pode ser compreendida da seguinte forma: i é acrescida de f_a a cada 1 segundo. Caso seja dividida pela frequência de amostragem, resulta $\frac{i}{f_a}$, que é acrescida de 1 a cada 1 segundo. Multiplicada pelo comprimento do período, resulta $i\frac{\tilde{\Lambda}}{f_a}$ que varre o período em 1 segundo. Por fim, com a frequência f , resulta $i.f\frac{\tilde{\Lambda}}{f_a}$ que completa f varreduras do período $\tilde{\Lambda}$ em 1 segundo, i.e. a sequência resultante apresenta a frequência fundamental f .

Importantes considerações: f é qualquer, só há limitantes nas frequências graves quando o tamanho da tabela $\tilde{\Lambda}$ não é suficientemente grande para a taxa de amostragem f_a . O procedimento de busca em tabela é computacionalmente bastante barato, substituindo cálculos por buscas simples (por isso geralmente é entendido como um processo de otimização). Salvo quando assinalado, no texto que usará este procedimento para todos os casos cabíveis pois simplifica as rotinas e é computacionalmente coerente.

O uso de LUTs é bastante difundido nas implementações computacionais voltadas para música e um uso clássico que explora com ênfase as LUTs na síntese sonora musical, é a chamada *Wavetable Synthesis* que consiste em várias LUTs utilizadas em conjunto através da mixagem para gerar uma nota musical quasi-periódica. (10, 32).

2.2.2 Variações incrementais de frequência e intensidade

Segundo a lei de Weber e Fechner, a percepção humana tem uma relação logarítmica com o estímulo que a causa.(33) Em outras palavras, um estímulo em progressão exponencial é percebido como linear. Por razões didáticas e dado o uso nas AM e FM (veja subseção 2.2.5), a variação linear será abordada primeiro.

Em uma nota de duração $\Delta = \frac{\Lambda}{f_a}$, a frequência $f = f_i$ varia de f_0 até $f_{\Lambda-1}$ linearmente. Pode-se escrever:

$$F_i = \{f_i\}_0^{\Lambda-1} = \left\{ f_0 + (f_{\Lambda-1} - f_0) \frac{i}{\Lambda-1} \right\}_0^{\Lambda-1} \quad (2.33)$$

$$\Delta_{\gamma_i} = f_i \frac{\widetilde{\Lambda}}{f_a} \Rightarrow \gamma_i = \left\lfloor \sum_{j=0}^i f_j \frac{\widetilde{\Lambda}}{f_a} \right\rfloor = \left\lfloor \sum_{j=0}^i \frac{\widetilde{\Lambda}}{f_a} \left[f_0 + (f_{\Lambda-1} - f_0) \frac{j}{\Lambda-1} \right] \right\rfloor \quad (2.34)$$

$$\left\{ \overline{t_i^{f_0, f_{\Lambda-1}}} \right\}_0^{\Lambda-1} = \left\{ \widetilde{t}_{\gamma_i \% \widetilde{\Lambda}} \right\}_0^{\Lambda-1} \quad (2.35)$$

Onde $\Delta_{\gamma_i} = f_i \frac{\widetilde{\Lambda}}{f_a}$ é o incremento da LUT entre duas amostras dada a frequência do som na primeira amostra.

Desta forma, pode-se calcular os elementos $\overline{t_i^{f_0, f_{\Lambda-1}}}$ com base no período $\left\{ \widetilde{t}_i \right\}_0^{\Lambda-1}$.

As equações 2.33, 2.34 e 2.35 são relativas à progressão linear da frequência. Como assinalado para o caso geral, também aqui uma progressão de frequência *percebida* como linear segue uma progressão exponencial¹⁴. Pode-se escrever que: $f_i = f_0 \cdot 2^{\frac{i}{\Lambda-1} n_8}$ onde $n_8 = \log_2 \frac{f_{\Lambda-1}}{f_0}$ é o número de oitavas entre f_0 e $f_{\Lambda-1}$. De forma que $f_i = f_0 \cdot 2^{\frac{i}{\Lambda-1} \log_2 \frac{f_{\Lambda-1}}{f_0}} = f_0 \cdot 2^{\log_2 \left(\frac{f_{\Lambda-1}}{f_0} \right)^{\frac{i}{\Lambda-1}}} = f_0 \left(\frac{f_{\Lambda-1}}{f_0} \right)^{\frac{i}{\Lambda-1}}$.

Portanto, as equações de transições de frequência lineares para o ouvido são:

$$F_i = \{f_i\}_0^{\Lambda-1} = \left\{ f_0 \left(\frac{f_{\Lambda-1}}{f_0} \right)^{\frac{i}{\Lambda-1}} \right\}_0^{\Lambda-1} \quad (2.36)$$

$$\Delta_{\gamma_i} = f_i \frac{\widetilde{\Lambda}}{f_a} \Rightarrow \gamma_i = \left\lfloor \sum_{j=0}^i f_j \frac{\widetilde{\Lambda}}{f_a} \right\rfloor = \left\lfloor \sum_{j=0}^i f_0 \frac{\widetilde{\Lambda}}{f_a} \left(\frac{f_{\Lambda-1}}{f_0} \right)^{\frac{j}{\Lambda-1}} \right\rfloor \quad (2.37)$$

¹⁴ Ou, dito ainda de outra forma, uma progressão geométrica da frequência é percebida como uma progressão aritmética de alturas.

$$\left\{ \overline{t_i^{f_0, f_{\Lambda-1}}} \right\}_0^{\Lambda-1} = \left\{ \widetilde{t}_{\gamma_i \% \widetilde{\Lambda}} \right\}_0^{\Lambda-1} \quad (2.38)$$

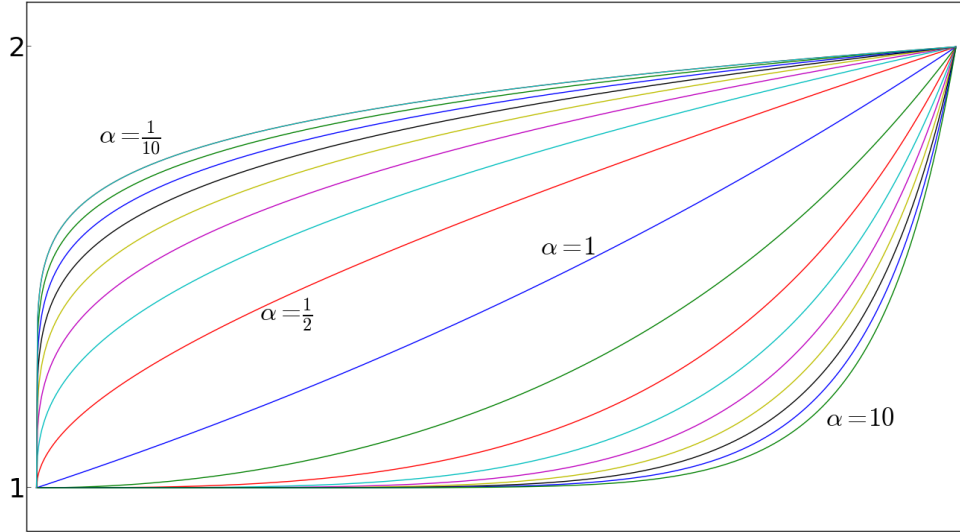


Figura 2.14 – Transições de intensidade para diferentes valores de α (veja equações 2.39 e 2.40).

O termo $\frac{i}{\Lambda-1}$ varre o intervalo $[0, 1]$ e pode-se elevá-lo a uma potência para que o início da transição seja mais suave ou abrupto. Este procedimento é útil para variações de energia da onda vibratória para alteração do volume¹⁵. Basta multiplicar a sequência original (seja ela gerada ou pré-estabelecida) pela sequência $a_{\Lambda-1}^{\left(\frac{i}{\Lambda-1}\right)^\alpha}$ onde α é o coeficiente citado e $a_{\Lambda-1}$ é fração da amplitude original que se visa atingir ao final da transição.

Assim, para variações de amplitude:

$$\{a_i\}_0^{\Lambda-1} = \left\{ a_0 \left(\frac{a_{\Lambda-1}}{a_0} \right)^{\left(\frac{i}{\Lambda-1}\right)^\alpha} \right\}_0^{\Lambda-1} = \left\{ (a_{\Lambda-1})^{\left(\frac{i}{\Lambda-1}\right)^\alpha} \right\}_0^{\Lambda-1} \text{ com } a_0 = 1 \quad (2.39)$$

¹⁵ A mudança do volume (qualidade psicofísica) ocorre através de diferentes características do som, como a reverberação e a concentração de harmônicos agudos, dentre as quais está a energia da onda. A manipulada com mais facilidade é a energia da onda (veja equação 2.2) e esta também pode variar de diferentes formas. Uma forma mais simples é variar a amplitude através da multiplicação da sequência toda por um número real. O aumento de energia sem variação de amplitude é a *compressão sonora*, útil na produção musical atual.(34)

$$T'_i = T_i \odot A_i = \{t_i \cdot a_i\}_0^{\Lambda-1} = \left\{ t_i \cdot (a_{\Lambda-1})^{\left(\frac{i}{\Lambda-1}\right)^\alpha} \right\}_0^{\Lambda-1} \quad (2.40)$$

Pode-se tomar $a_0 = 1$ para iniciar a nova sequência com a amplitude original e então ir modificando com o decorrer das amostras. Esta restrição faz com que o termo $a_{\Lambda-1}$ seja a variação da amplitude. Caso $\alpha = 1$, a variação de amplitude segue exatamente a progressão geométrica que caracteriza a percepção linear. A figura 2.14 exibe as transições para diferentes valores de α e para a transição entre os valores 1 e 2, um ganho de $\approx 6dB$ segundo a equação 2.4.

Alguns cuidados são necessários para lidar com $a = 0$. Na equação 2.39, se $a_0 = 0$ há divisão por zero e se $a_{\Lambda-1} = 0$, há uma multiplicação por zero. Ambos os casos tornam o procedimento inútil pois nenhum número diferente de zero pode ser representado como uma proporção com relação ao zero. Pode-se resolver isso escolhendo um número suficientemente pequeno como $-80dB \Rightarrow a = 10^{\frac{-80}{20}} = 10^{-4}$ como o volume mínimo no caso de um *fade in* ($a_0 = 10^{-4}$) ou de um *fade out* ($a_{\Lambda-1} = 10^{-4}$).

Para uma amplificação linear, mas não linear para a percepção, basta usar uma sequência $\{a_i\}$ adequada:

$$a_i = a_0 + (a_{\Lambda-1} - a_0) \frac{i}{\Lambda - 1} \quad (2.41)$$

Aqui convém a conversão de decibels para amplitude. Assim, as equações 2.8 e 2.40 especificam a transição de V_{dB} decibels:

$$T'_i = \left\{ t_i 10^{\frac{V_{dB}}{20} \left(\frac{i}{\Lambda-1}\right)^\alpha} \right\}_0^{\Lambda-1} \quad (2.42)$$

para o caso geral de variações de amplitude segundo a progressão geométrica. Quanto maior o valor de α , mais suave é a introdução do som e mais intenso o final da transição. $\alpha > 1$ resulta

em transições de volume muitas vezes chamadas de *slow fade* enquanto $\alpha < 1$ resulta em *fast fade*.(34)

As transições lineares serão usadas para as sínteses AM e FM e a aplicação das transições logarítmicas para os tremolos e vibratos. Uma exploração não oscilatória destas variações está na montagem musical *Transita para metro*, cujo código está no Apêndice B.2.1 e online na MASSA.(2)

2.2.3 Aplicação de filtros digitais

Esta subseção limita-se a uma descrição do processamento das sequências, por convolução e equação a diferenças, e em aplicações imediatas, pois a complexidade facilmente foge ao escopo¹⁶. A aplicação de filtros pode ser parte constituinte da síntese ou feita posteriormente como parte dos processos tipicamente chamados de tratamento sonoro.

- Convolução e filtros de resposta ao impulso finita (FIR)

Os filtros aplicados por convolução são conhecidos pela sua sigla FIR (do inglês Finite Impulse Response) e são caracterizados por possuírem uma representação amostral finita no tempo. Esta representação amostral é chamada de 'resposta ao impulso' $\{h_i\}$. Os filtros FIR são aplicados no domínio temporal ao som digitalizado pela convolução do som com a resposta ao impulso do filtro¹⁷. Para os fins deste trabalho, a convolução fica definida como:

¹⁶ A elaboração de filtros constitui uma área reconhecidamente complexa, com literatura e pacotes de software dedicados. Recomendamos ao leitor interessado uma visita à nossa bibliografia.(25, 35)

¹⁷ Pode-se aplicar o filtro do domínio espectral através da multiplicação das transformadas de Fourier de ambos o som e a resposta ao impulso, e então realizada a transformada inversa de Fourier do espectro resultante.(25)

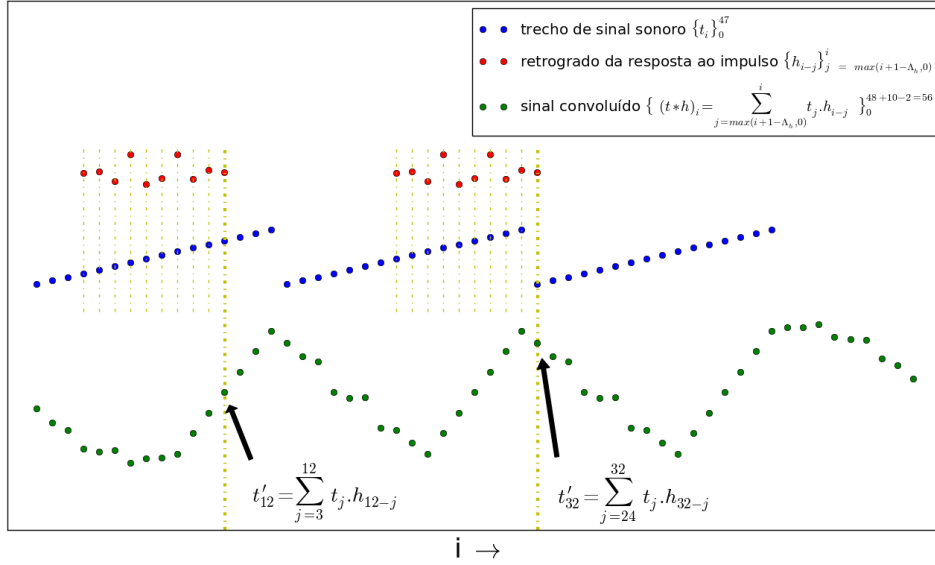


Figura 2.15 – Interpretação gráfica da convolução. Cada amostra resultante é a soma das amostras anteriores de um sinal uma a uma multiplicadas pelas amostras retrógradas do outro sinal.

$$\begin{aligned}
 \{t'_i\}_0^{\Lambda_t + \Lambda_h - 2 = \Lambda_{t'} - 1} &= \{(T_j * H_j)_i\}_0^{\Lambda_{t'} - 1} = \left\{ \sum_{j=0}^{\min(\Lambda_h - 1, i)} h_j \cdot t_{i-j} \right\}_0^{\Lambda_{t'} - 1} \\
 &= \left\{ \sum_{j=\max(i+1-\Lambda_h, 0)}^i t_j \cdot h_{i-j} \right\}_0^{\Lambda_{t'} - 1}
 \end{aligned} \tag{2.43}$$

Onde $t_i = 0$ para as amostras não definidas de antemão. Ou seja, o som $\{t'_i\}$ resultante da convolução de $\{t_i\}$ com a resposta ao impulso $\{h_i\}$ tem cada i -ésima amostra t_i substituída pela soma de suas últimas Λ_h amostras $\{t_{i-j}\}_{j=0}^{\Lambda_h-1}$ multiplicadas uma a uma pelas amostras da resposta ao impulso $\{h_i\}_0^{\Lambda_h-1}$. Este procedimento está ilustrado na figura 2.15, onde a resposta ao impulso $\{h_i\}$ é percorrida na forma retrógrada e t'_{12} e t'_{32} são duas amostras calculadas pela convolução $(T_j * H_j)_i = t'_i$. O sinal resultante possui sempre o tamanho $\Lambda_t + \Lambda_h - 1 = \Lambda_{t'}$.

Com este procedimento pode-se aplicar reverberadores, equalizadores, *delays* e vários

outros tipos de filtros para fins de tratamento sonoro ou efeitos musicais/artísticos.

A resposta ao impulso pode provir de medições físicas ou da síntese. Uma resposta ao impulso para a aplicação de reverberação pode resultar da gravação sonora em um ambiente ao disparar um estalo que se assemelhe a um impulso ou de uma varredura em senoide, que transformada se aproxima da resposta em frequência. Ambas são respostas ao impulso que, convoluidas com a sequência sonora, resultam na própria sequência com uma reverberação que se assemelha àquela do ambiente em que ocorreu a medição.(10)

A transformada inversa de Fourier de uma envoltória par e real é uma resposta ao impulso de um FIR. Este realiza uma filtragem em frequência com a envoltória. Quanto maior o número de amostras maior a resolução da envoltória e também o processamento computacional, pois a convolução é cara.

Uma propriedade importante é o deslocamento temporal causado pela convolução com o impulso deslocado. Embora caro computacionalmente, pode-se criar linhas de *delays* através da convolução do som com uma resposta ao impulso que possui um impulso para cada reincidência do som. Na figura 2.16 pode-se observar o deslocamento causado pela convolução com o impulso. Dependendo da densidade dos impulsos, o resultado é de caráter rítmico (20 impulsos por segundo ou menos) ou de amálgama sonoro (20-40 impulsos por segundo ou mais). Neste último caso, ocorrem processos tipicamente vinculados à síntese granular, delays, reverbs e equalizações.

- Filtros de resposta ao impulso infinita (IIR)

Esta classe de filtros é conhecida pela sigla IIR (do inglês Infinite Impulse Response) e é caracterizada por possuir uma representação temporal infinita, i.e. a resposta ao impulso não converge para zero. Sua aplicação é usualmente feita pela equação:

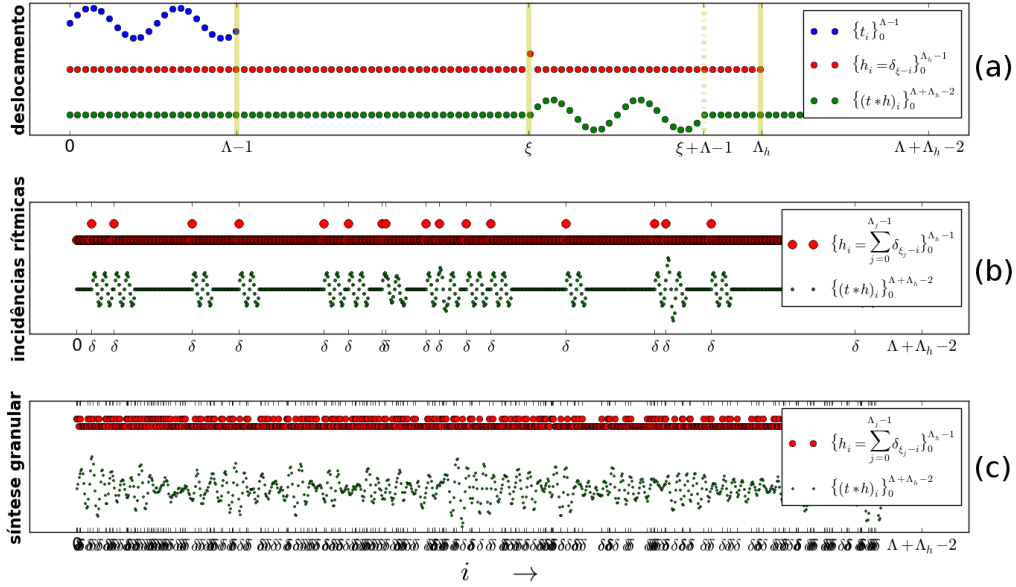


Figura 2.16 – Convolução com o impulso: deslocamento (a), linhas de delays (b) e síntese granular (c).
Dispostos em ordem crescente de densidade de pulsos.

$$t'_i = \frac{1}{b_0} \left(\sum_{j=0}^J a_j \cdot t_{i-j} + \sum_{k=1}^K b_k \cdot t'_{i-k} \right) \quad (2.44)$$

com $b_0 = 1$ na grande maioria dos casos pois pode-se normalizar as variáveis: $a'_j = \frac{a_j}{b_0}$ e $b'_k = \frac{b_k}{b_0} \Rightarrow b'_0 = 1$. A equação 2.44 é chamada 'equação a diferenças' por exibir as amostras resultantes $\{t'_i\}$ através das diferenças entre as amostras originais $\{t_i\}$ e as amostras resultantes anteriores $\{t'_{i-k}\}$.

Existem diversos métodos e ferramentas para a elaboração de filtros IIR e segue abaixo uma seleção com fins didáticos e para consulta futura por utilidade. São filtros bem comportados e cujas filtragens estão na figura 2.17.

No caso dos filtros de ordem simples, a frequência de corte f_c é onde o filtro realiza uma atenuação de $-3dB \approx 0.707$ da amplitude original. No caso dos filtros passa e rejeita banda, esta mesma atenuação é resultado de duas especificações: f_c (neste caso mais bem compreendida como 'frequência central') e a largura de banda bw , em ambas as frequên-

cias $f_c \pm bw$ há uma atenuação de ≈ 0.707 da amplitude original. Existe amplificação do som no caso dos filtros passa e rejeita banda quando a frequência de corte é baixa e a largura de banda é grande o suficiente. Nos agudos, estes filtros apresentam somente um desvio do perfil esperado, expandindo a envoltória para o lado grave da banda em evidência.

Para filtros cujas respostas em frequência possuem outras envoltórias (para o módulo), pode-se realizar cascatas destes filtros aplicando-os sucessivamente. Outra possibilidade é utilizar alguma receita de filtro biquad¹⁸ ou rotinas para cálculo de coeficientes de filtros Chebichev¹⁹. Ambas as possibilidades são exploradas por títulos em nossas referências, em especial (35, 36) e a coleção de filtros da comunidade *Music-DSP*, da Universidade de Columbia.(25, 37)

1. Passa-baixas de polo simples com módulo da resposta em frequência no canto superior esquerdo da figura 2.17. A fórmula geral tem por referência da frequência de corte $f_c \in (0, \frac{1}{2})$, fração da frequência de amostragem f_a em que há aproximadamente uma atenuação de $3dB$. Os coeficientes do filtro IIR a_0 e b_1 são dados através da variável intermediária $x \in [e^{-\pi}, 1]$:

$$\begin{aligned} x &= e^{-2\pi f_c} \\ a_0 &= 1 - x \\ b_1 &= x \end{aligned} \tag{2.45}$$

2. Passa-altas de polo simples com o módulo da resposta em frequência no canto superior direito da figura 2.17. A fórmula geral, com frequência de corte $f_c \in (0, \frac{1}{2})$, é calculada

¹⁸ Abreviação de 'biquadrado' pois sua função de transferência possui dois polos e dois zeros, i.e. sua forma normal consiste em dois polinômios quadráticos formando uma fração: $\mathbb{H}(z) = \frac{a_0 + a_1.z^{-1} + a_2.z^{-2}}{1 - b_1.z^{-1} - b_2.z^{-2}}$.

¹⁹ Filtros Butterworth e Elípticos podem ser considerados como casos específicos dos Filtros do tipo Chebichev.(25, 35)

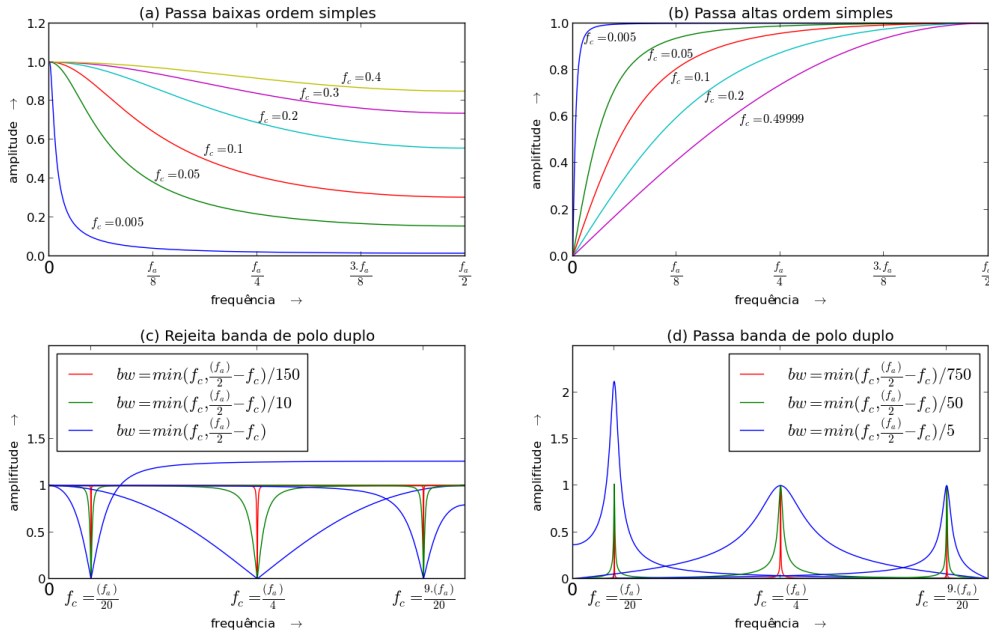


Figura 2.17 – Módulos da resposta em frequência (a), (b), (c) e (d) respectivamente dos filtros IIR das equações 2.45, 2.46, 2.48 e 2.49 para diferentes frequências de corte, frequências centrais e larguras de banda.

através da variável intermediária $x \in [e^{-\pi}, 1]$:

$$\begin{aligned}
 x &= e^{-2\pi f_c} \\
 a_0 &= \frac{x+1}{2} \\
 a_1 &= -\frac{x+1}{2} \\
 b_1 &= x
 \end{aligned} \tag{2.46}$$

3. Nó (*notch filter*). Este filtro é parametrizado pela frequência central²⁰ f_c e a largura de banda bw - $f_c \pm bw$, que resultam em 0.707 da amplitude, i.e. atenuação de 3dB - ambos dados como frações de f_a , portanto $f, bw \in (0, 0.5)$.

Por facilidade, sejam as variáveis auxiliares K e R :

²⁰ Atenção com a frequência de corte também f_c nos filtros passa baixas e passa altas.

$$\begin{aligned}
 R &= 1 - 3bw \\
 K &= \frac{1 - 2R \cos(2\pi f_c) + R^2}{2 - 2 \cos(2\pi f_c)}
 \end{aligned}
 \tag{2.47}$$

O filtro passa banda do canto inferior esquerdo da figura 2.17 possui os seguintes coeficientes para a equação 2.44:

$$\begin{aligned}
 a_0 &= 1 - K \\
 a_1 &= 2(K - R) \cos(2\pi f_c) \\
 a_2 &= R^2 - K \\
 b_1 &= 2R \cos(2\pi f_c) \\
 b_2 &= -R^2
 \end{aligned}
 \tag{2.48}$$

Os coeficientes do filtro rejeita banda são:

$$\begin{aligned}
 a_0 &= K \\
 a_1 &= -2K \cos(2\pi f_c) \\
 a_2 &= K \\
 b_1 &= 2R \cos(2\pi f_c) \\
 b_2 &= -R^2
 \end{aligned}
 \tag{2.49}$$

com o módulo de sua resposta em frequência disposto na parte inferior esquerda da figura 2.17.

2.2.4 Ruídos

De forma geral, os sons sem altura definida são chamados ruídos.⁽¹¹⁾ Estes são constituintes importantes dos sons musicais de altura definida, como os ruídos presentes nas notas do piano, do violino, etc. Além disso, os instrumentos de percussão, em grande parte, não possuem altura definida e seus sons são em geral compreendidos como ruídos.⁽³⁾ Na música eletrônica, incluindo a eletroacústica e gêneros de pista de dança, os ruídos possuem usos diversificados e comumente característicos do estilo musical.⁽¹⁰⁾

A ausência de uma altura definida é fruto da ausência de uma organização harmônica perceptível nas componentes senoidais que formam o som. Assim, são incontáveis as possibilidades de gerar ruídos. A utilização de valores aleatórios para a geração da sequência sonora T_i é um método atraente, mas os resultados não são tão úteis, tendendo geralmente ao ruído branco.⁽¹⁰⁾

Outra possibilidade é a geração de ruído através do espectro desejado, a partir do qual executamos a transformada inversa de Fourier. A distribuição espectral deve ser feita com cuidado pois caso se utilize a mesma fase, ou fases com forte correlação, o som sintetizado possuirá energia bastante concentrada em alguns trechos de sua duração.

Abaixo elencamos alguns ruídos de espectro estático. São chamados *coloridos* por terem sido associados a cores. A figura 2.18 mostra lado a lado o perfil espectral e a sequência sonora. Os ruídos foram gerados com a mesma fase, então pode-se observar o resultado das contribuições em diferentes regiões do espectro.

- O ruído branco deve seu nome por possuir energia distribuída igualmente por todas as frequências. Pode-se realizar o ruído branco com a transformada inversa dos seguintes coeficientes:

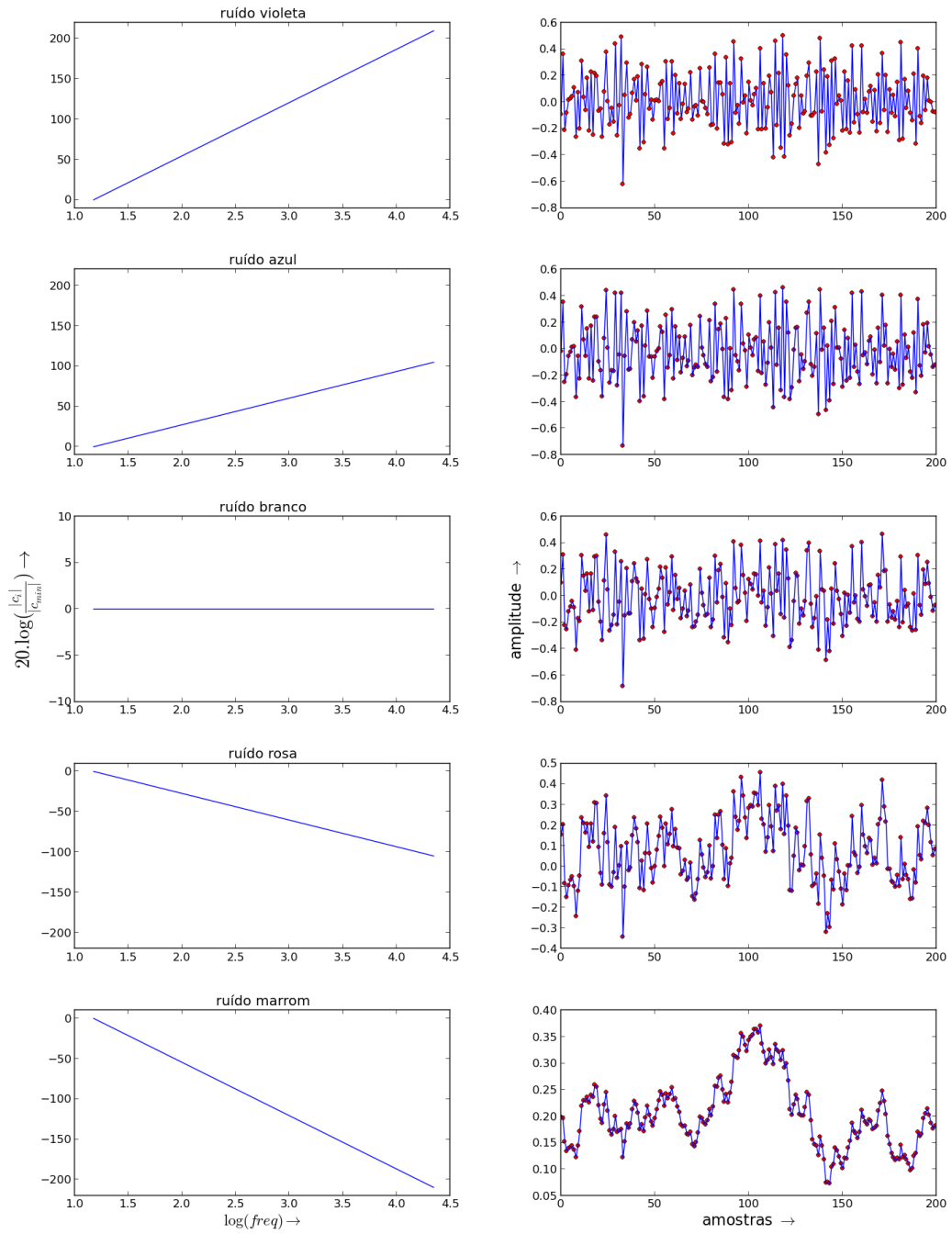


Figura 2.18 – Ruídos coloridos realizados através das equações 2.50, 2.51, 2.52, 2.53, 2.54: espectros e ondas sonoras resultantes.

$$\begin{aligned}
c_0 &= 0 \quad \text{pois evita-se bias} \\
c_i &= e^{j.x}, \quad j^2 = -1, \quad x \text{ randômico} \in [0, 2\pi], \quad i \in \left[1, \frac{\Lambda}{2} - 1\right] \\
c_{\Lambda/2} &= 1 \quad (\text{se } \Lambda \text{ par}) \\
c_i &= c_{\Lambda-i}^*, \quad \text{para } i > \frac{\Lambda}{2}
\end{aligned} \tag{2.50}$$

O valor de c_i calculado pela exponencial é apenas um artifício para resultar em módulo unitário e fase aleatória. Já $c_{\Lambda/2}$ é sempre puramente real (como vimos na seção anterior).

- O ruído rosa possui uma queda de $3dB$ por oitava. Este ruído é muito usual no teste de equipamentos e montagens de aparelhos além de presença destacada na natureza (3).

$$\begin{aligned}
f_{\min} &\approx 15Hz \\
f_i &= i \frac{f_a}{\Lambda}, \quad i \leq \frac{\Lambda}{2}, \quad i \in \mathbb{N} \\
\alpha_i &= \left(10^{-\frac{3}{20}}\right)^{\log_2\left(\frac{f_i}{f_{\min}}\right)} \\
c_i &= 0, \quad \forall i : f_i < f_{\min} \\
c_i &= e^{j.x} \cdot \alpha_i, \quad j^2 = -1, \quad x \text{ randômico} \in [0, 2\pi], \quad \forall i : f_{\min} \leq f_i < f_{\lceil \Lambda/2 - 1 \rceil} \\
c_{\Lambda/2} &= \alpha_{\Lambda/2} \quad (\text{se } \Lambda \text{ par}) \\
c_i &= c_{\Lambda-i}^*, \quad \text{para } i > \Lambda/2
\end{aligned} \tag{2.51}$$

A frequência mínima f_{\min} pode ser escolhida com base no limite da audição, pois não se escuta como altura uma componente sonora cuja frequência esteja abaixo de $\approx 20Hz$.

Os ruídos restantes podem ser feitos com base no procedimento descrito para o ruído rosa, bastando que modificar detalhes, em especial a equação que define α_i .

- O ruído marrom deve seu nome a Robert Brown, que descreveu o movimento browniano.

Embora esta origem seja um tanto díspar do que pode-se considerar motivo para uma associação com a cor marrom, o ruído sonoro ficou consagrado com este nome. De qualquer forma, é bastante comum declarar satisfatória a associação do ruído com a cor marrom, uma vez que os ruídos branco e rosa são mais estridentes e relacionados a cores mais intensas (10, 34).

O que caracteriza este ruído é a queda de $6dB$ por oitava. Desta forma, α_i no conjunto 2.51 fica:

$$\alpha_i = (10^{-\frac{6}{20}})^{\log_2\left(\frac{f_i}{f_{\min}}\right)} \quad (2.52)$$

- No ruído azul há ganho de $3dB$ por oitava em uma banda limitada pela frequência mínima f_{\min} e a frequência máxima f_{\max} . Assim, também com base no conjunto de equações 2.51:

$$\alpha_i = (10^{\frac{3}{20}})^{\log_2\left(\frac{f_i}{f_{\min}}\right)} \quad (2.53)$$

$$c_i = 0, \quad \forall i : f_i < f_{\min} \text{ ou } f_i > f_{\max}$$

- O ruído violeta é similar ao ruído azul, mas o ganho é de $6dB$ por oitava:

$$\alpha_i = (10^{\frac{6}{20}})^{\log_2\left(\frac{f_i}{f_{\min}}\right)}, \quad f_{\min} \approx 15Hz \quad (2.54)$$

- O ruído preto possui perdas maiores que $6dB$ por oitava, assim:

$$\alpha_i = (10^{-\frac{\beta}{20}})^{\log_2\left(\frac{f_i}{f_{\min}}\right)}, \quad \beta > 6 \quad (2.55)$$

- O ruído cinza é definido como um ruído branco sujeito a uma das curvas iso-audíveis. Estas curvas são resultados experimentais e necessárias para a obtenção de α_i . Uma implementação da ISO 226, a última revisão destas curvas, está na toolbox MASSA.(2)

Foram expostos somente ruídos com espectro estático. Existem classificações de ruídos com variações do espectro no decorrer do tempo. Existem também ruídos que são fundamentalmente transientes, como os clicks e os chirps. O primeiro é modelado facilmente por um impulso relativamente isolado, enquanto o segundo não é um ruído, mas uma varredura rápida de alguma banda de frequência.(10)

Os ruídos das equações 2.50, 2.51, 2.52, 2.53, 2.54 estão na figura 2.18. Os espectros foram feitos com a mesma fase em cada coeficiente de mesma frequência, de forma que se pode observar a contribuição dos harmônicos agudos e das frequências graves.

2.2.5 Tremolo e vibrato, AM e FM

O vibrato é uma variação periódica de altura (frequência) e o tremolo é uma variação periódica de volume (intensidade).²¹ Para o caso geral, o vibrato é descrito da seguinte forma:

$$\gamma'_i = \left[i f' \frac{\tilde{\Lambda}_M}{f_a} \right] \quad (2.56)$$

$$t'_i = \tilde{m}_{\gamma'_i \% \tilde{\Lambda}_M} \quad (2.57)$$

$$f_i = f \left(\frac{f + \mu}{f} \right)^{t'_i} = f \cdot 2^{t'_i \frac{\nu}{12}} \quad (2.58)$$

$$\Delta_{\gamma_i} = f_i \frac{\tilde{\Lambda}}{f_a} \Rightarrow \gamma_i = \left[\sum_{j=0}^i f_j \frac{\tilde{\Lambda}}{f_a} \right] = \left[\sum_{j=0}^i \frac{\tilde{\Lambda}}{f_a} f \left(\frac{f + \mu}{f} \right)^{t'_j} \right] = \left[\sum_{j=0}^i \frac{\tilde{\Lambda}}{f_a} f \cdot 2^{t'_j \frac{\nu}{12}} \right] \quad (2.59)$$

²¹ Alguns instrumentos e contextos musicais usam nomenclaturas diferentes. Por exemplo, no piano, o chamado tremolo é um vibrato e um tremolo segundo a classificação aqui utilizada. As definições presentes neste trabalho tem por base uma literatura mais abrangente do que a utilizada para um único instrumento, prática ou tradição musical e comum em contextos de teoria musical e musica eletrônica.(11, 30)

$$T_i^{f,vbr(f',v)} = \left\{ t_i^{f,vbr(f',v)} \right\}_0^{\Lambda-1} = \left\{ \tilde{t}_{\gamma_i \% \tilde{\Lambda}} \right\}_0^{\Lambda-1} \quad (2.60)$$

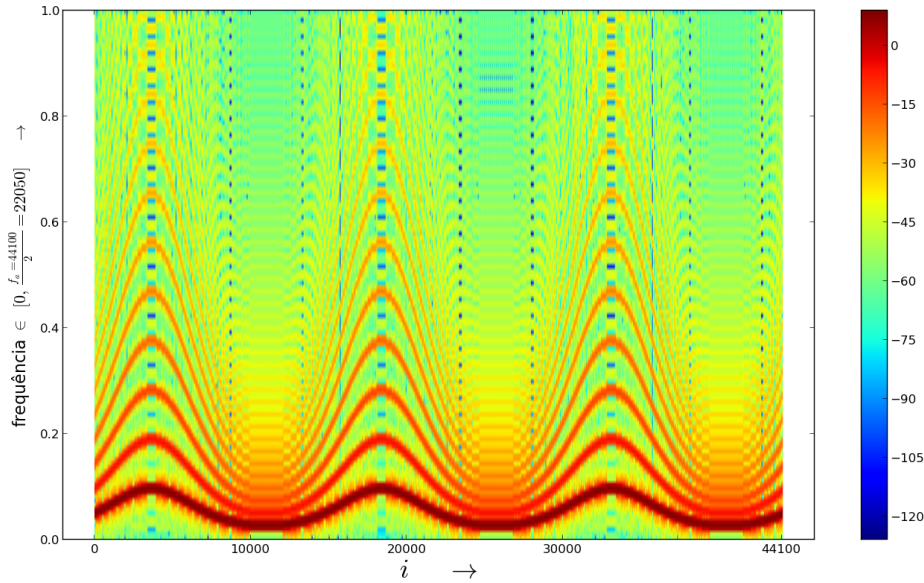


Figura 2.19 – Espectrograma de um som com vibrato senoidal de 3Hz e profundidade de uma oitava em uma dente de serra de 1000Hz (considerada $f_a = 44.1kHz$).

Para a correta realização do vibrato, é importante atenção para as duas tabelas e sequências. A tabela \tilde{M}_i de tamanho $\tilde{\Lambda}_M$ e a sequência de índices γ'_i formam a sequência t'_i que é o padrão da oscilação da frequência enquanto a tabela \tilde{L}_i de tamanho $\tilde{\Lambda}$ e a sequência de índices γ_i formam t_i que é o som em si. As variáveis μ e ν quantificam a intensidade do vibrato: μ é uma medida direta da quantidade de Hertz envolvidos no limite superior da oscilação e ν é a medida direta de semitons envolvidos na oscilação (2ν é o número de semitons entre os picos superiores e inferiores de oscilação da frequência do som $\{t_i\}$ causada pelo vibrato). É conveniente $\nu = \log_2 \frac{f+\mu}{f}$ neste caso pois o aumento máximo de frequência não equivale à diminuição máxima, mas a variação de semitons se mantém.

A Figura 2.19 é o espectrograma de um vibrato artificial de uma nota em 1000Hz (entre um si e um dó) e cujo desvio da frequência atinge uma oitava para cima e para baixo. Qualquer forma

de onda pode ser utilizada para gerar o som e o padrão de oscilação do vibrato em quaisquer frequência de oscilação e desvio de altura envolvidos²². Estas oscilações com formas precisas e amplitudes arbitrárias não são praticáveis em instrumentos musicais tradicionais, introduzindo novidade nas possibilidades artísticas.

O caso do tremolo é semelhante: f' , γ'_i e t'_i permanecem os mesmos. A sequência de amplitudes a serem multiplicadas pela sequência original t_i fica:

$$a_i = 10^{\frac{V_{dB}}{20}} t'_i = a'_{i\text{máx}} \quad (2.61)$$

$$T_i^{tr(f')} = \{t_i^{tr(f')}\}_0^{\Lambda-1} = \{t_i \cdot a_i\}_0^{\Lambda-1} = \left\{t_i \cdot 10^{\frac{V_{dB}}{20}}\right\}_0^{\Lambda-1} = \left\{t_i \cdot a'_{i\text{máx}}\right\}_0^{\Lambda-1} \quad (2.62)$$

Onde V_{dB} é a profundidade da oscilação em decibels do tremolo e $a_{\text{máx}} = 10^{\frac{V_{dB}}{20}}$ é o ganho máximo de amplitude envolvido. A medição em decibels é pertinente pois o aumento máximo de amplitude não equivale à diminuição máxima relacionada, enquanto a diferença em decibels se mantém.

A figura 2.20 mostra a amplitude das sequências $\{a_i\}_0^{\Lambda-1}$ e $\{t'_i\}_0^{\Lambda-1}$ para três oscilações de um tremolo com forma da dente de serra. A curvatura é devida à progressão logarítmica de intensidade. A frequência do tremolo é de $1,5Hz$ pois $f_a = 44,1kHz \Rightarrow \text{duração} = \frac{i_{\text{máx}}=82000}{f_a} = 2s \Rightarrow \frac{3\text{oscilações}}{2s} = 1,5 \text{ oscilações por segundo (Hz)}$.

A montagem musical *Vibra e treme* explora estes recursos dos tremolos e vibratos em associação e isoladamente com frequências f' e profundidades (v e V_{dB}) diferentes, variações progressivas dos parâmetros²³. A peça desenvolve também uma comparação entre os vibratos e tremolos em escala logarítmica e em escala linear para uma apreciação qualitativa. Seu código está no

²² O desvio de altura é chamado profundidade do vibrato e é geralmente dado por conveniência em semitons ou cents.

²³ Os tremolos e vibratos ocorrem muitas vezes juntos em instrumentos tradicionais e na voz.

Apêndice B.2.2 e disponível online como parte da *toolbox* MASSA.

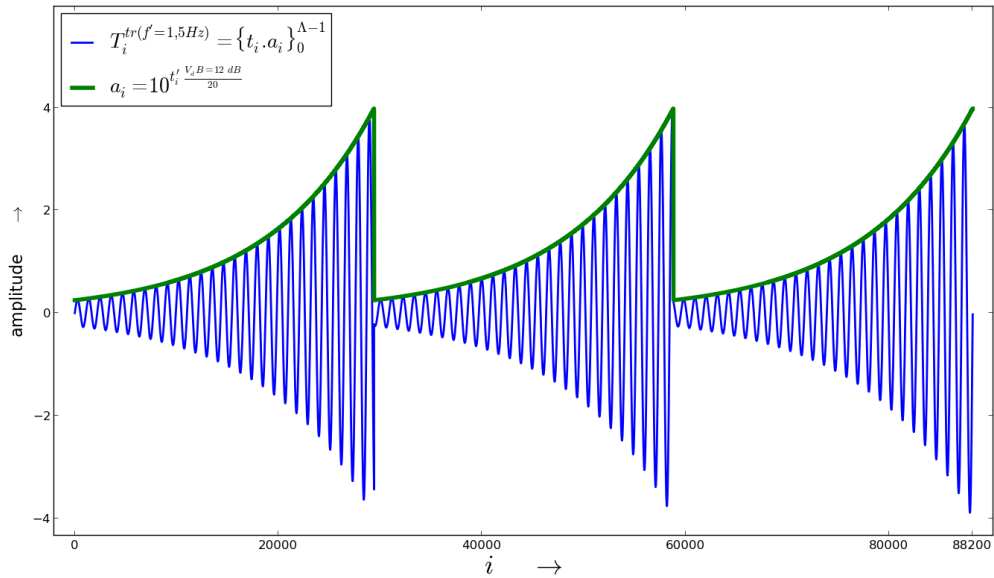


Figura 2.20 – Tremolo de profundidade $V_{dB} = 12dB$ com padrão oscilatório de uma dente de serra em $f' = 1.5Hz$ em uma senoide de $f = 40Hz$ (considerada taxa de amostragem $f_a = 44,1kHz$).

No aumento progressivo de f' , a aproximação do limiar de frequência para audição do fenômeno sonoro como altura ($\approx 20Hz$) gera rugosidades para ambos tremolos e vibratos. Estas rugosidades são muito apreciadas tanto na tradição erudita quanto na música eletrônica atual, especialmente no *Dubstep*. A rugosidade também é gerada através de conteúdos espectrais que geram batimentos.(38, 39) A sequência *Bela Rugosi* explora este limiar com concomitâncias de tremolos e vibratos na mesma voz, com intensidades e formas de onda diferentes. Seu código está o Apêndice B.2.6 e disponível online como parte da MASSA.

Aumentando ainda mais a frequência, estas oscilações deixam de ser eventos identificáveis. Neste caso, as oscilações são audíveis como altura. Assim, f' , μ e a forma de onda realizam alterações espectrais no som original T_i de formas diferentes para os tremolos e para os vibratos. São as chamadas sínteses AM (*Amplitude Modulation*) e FM (*Frequency Modulation*), respectivamente. Estas são técnicas conhecidas, com aplicações em sintetizadores como o Ya-

maha DX7, e com aplicações fora da música, como em telecomunicações para transmissão de informação via ondas eletromagnéticas (ex. rádios AM e FM).

Para fins musicais e em resumo, pode-se entender a síntese FM através do caso entre senoides e decompor os sinais em seus espectros de Fourier (i.e. senoidais) para casos mais complexos. Assim, a síntese FM realizada com um 'vibrato' senoidal de frequência f' e profundidade μ em um som também senoidal T_i de frequência f gera bandas centradas em f e distantes f' entre si:

$$\begin{aligned} \{t'_i\} &= \left\{ \cos \left[f \cdot 2\pi \frac{i}{f_a - 1} + \mu \cdot \text{sen} \left(f' \cdot 2\pi \frac{i}{f_a - 1} \right) \right] \right\} = \\ &= \left\{ \sum_{k=-\infty}^{+\infty} J_k(\mu) \cos \left[f \cdot 2\pi \frac{i}{f_a - 1} + k \cdot f' \cdot 2\pi \frac{i}{f_a - 1} \right] \right\} = \\ &= \left\{ \sum_{k=-\infty}^{+\infty} J_k(\mu) \cos \left[(f + k \cdot f') \cdot 2\pi \frac{i}{f_a - 1} \right] \right\} \end{aligned} \quad (2.63)$$

onde

$$J_k(\mu) = \frac{2}{\pi} \int_0^{\frac{\pi}{2}} \left[\cos \left(\bar{k} \frac{\pi}{2} + \mu \cdot \sin w \right) \cdot \cos \left(\bar{k} \frac{\pi}{2} + k \cdot w \right) \right] dw \quad , \quad \bar{k} = k \% 2 \quad , \quad k \in \mathbb{N} \quad (2.64)$$

é a função de Bessel (36, 40) que na FM especifica a amplitude de cada componente.

Nestas equações, a variação de frequência introduzida por $\{t'_i\}$ não respeita a progressão geométrica de frequência que acompanha a percepção de altura, mas sim a equação 2.33. A utilização das equações 2.58 para a FM está no Apêndice D, onde é calculado o conteúdo espectral da síntese FM obtida com oscilações na escala logarítmica. De fato, o comportamento simples que torna a FM atraente é obtido somente com as variações lineares em 2.63.

O caso da modulação de amplitude (AM) é mais simples:

$$\begin{aligned}
\{t'_i\}_0^{\Lambda-1} &= \{(1+a_i).t_i\}_0^{\Lambda-1} = \left\{ \left[1 + M \cdot \sin\left(f' \cdot 2\pi \frac{i}{f_a-1}\right) \right] \cdot P \cdot \sin\left(f \cdot 2\pi \frac{i}{f_a-1}\right) \right\}_0^{\Lambda-1} = \\
&= \left\{ P \cdot \sin\left(f \cdot 2\pi \frac{i}{f_a-1}\right) + \frac{P \cdot M}{2} \left[\sin\left((f-f') \cdot 2\pi \frac{i}{f_a-1}\right) + \sin\left((f+f') \cdot 2\pi \frac{i}{f_a-1}\right) \right] \right\}_0^{\Lambda-1}
\end{aligned} \tag{2.65}$$

Ou seja, o som resultante é o original e a reprodução de seu conteúdo espectral acima e abaixo da frequência original, distantes f' de f . Novamente, isso é obtido com a variação na escala linear de amplitude. No Apêndice D está uma exposição do espectro da AM realizada com a oscilação na escala logarítmica de amplitude. Esta também perde o comportamento simples.

A sequência T_i de frequência f , chamada portadora, é modulada pela f' , chamada moduladora. No jargão de FM e AM, μ e $\alpha = 10^{\frac{V_{dB}}{20}}$ são chamados índices de modulação. Assim, para o padrão de vibração da moduladora $\{t'_i\}$ as equações:

$$\gamma'_i = \left\lfloor i f' \frac{\widetilde{\Lambda}_M}{f_a} \right\rfloor \tag{2.66}$$

$$t'_i = \widetilde{m}_{\gamma'_i \% \widetilde{\Lambda}_M} \tag{2.67}$$

Para aplicação da moduladora $\{t'_i\}$ na portadora $\{t_i\}$ por FM:

$$f_i = f + \mu \cdot t'_i \tag{2.68}$$

$$\Delta\gamma_i = f_i \frac{\widetilde{\Lambda}}{f_a} \Rightarrow \gamma_i = \left\lfloor \sum_{j=0}^i f_j \frac{\widetilde{\Lambda}}{f_a} \right\rfloor = \left\lfloor \sum_{j=0}^i \frac{\widetilde{\Lambda}}{f_a} (f + \mu \cdot t'_j) \right\rfloor \tag{2.69}$$

$$T_i^{f, FM(f', \mu)} = \{t_i^{f, FM(f', \mu)}\}_0^{\Lambda-1} = \{\tilde{t}_{\gamma_i \% \tilde{\Lambda}}\}_0^{\Lambda-1} \quad (2.70)$$

Onde \tilde{t} é um período da forma de onda de comprimento $\tilde{\Lambda}$ da portadora.

Para realizar a AM, basta modular $\{t_i\}$ com $\{t'_i\}$ através das equações:

$$a_i = 1 + \alpha \cdot t'_i \quad (2.71)$$

$$T_i^{f, AM(f', \alpha)} = \{t_i^{f, AM(f', \alpha)}\}_0^{\Lambda-1} = \{t_i \cdot a_i\}_0^{\Lambda-1} = \{t_i \cdot (1 + \alpha \cdot t'_i)\}_0^{\Lambda-1} \quad (2.72)$$

2.2.6 Usos musicais

A este ponto as possibilidades musicais explodiram. As características de altura (dada pela frequência), timbre (dado pela forma de onda e filtragens), volume (dado pela intensidade) e duração (dada pelo número de amostras) podem ser consideradas de forma absoluta ou tratadas ao longo de sua duração, com a única exceção da duração em si.

Desta forma, os usos musicais aqui dispostos são uma coleção de possibilidades com o objetivo de exemplificar manipulações sonoras que resultem algo musical, por razões variadas e aprofundadas na próxima seção.

Vínculos entre características

Uma outra possibilidade interessante é usar vínculos entre os parâmetros do tremolo e do vibrato e algum parâmetro da nota básica, como a frequência. Assim, pode-se estabelecer que a frequência do vibrato é diretamente proporcional à altura, e a profundidade do tremolo é inversamente proporcional à altura. Desta forma, com as equações 2.56, 2.58 e 2.61 pode-se

escrever:

$$\begin{aligned} f^{vbr} &= f^{tr} = func_a(f) \\ v &= func_b(f) \\ V_{dB} &= func_c(f) \end{aligned} \tag{2.73}$$

Com f^{vbr} e f^{tr} como f' nas equações de referência, ou seja, a frequência de oscilação do vibrato e do tremolo da equação 2.56. Já v e V_{dB} são as profundidades do vibrato e do tremolo, respectivamente. As funções $func_a$, $func_b$ e $func_c$ são arbitrárias e dependentes das intenções musicais. A montagem *Tremolos, vibratos e a frequência* explora recursos como este e variações da forma de onda da oscilação com vínculos, de modo a formar um *idioma musical*²⁴. Seu código está no Apêndice B.2.3 e também disponível online como parte da caixa de ferramentas MASSA.

Com relação à convolução, pode-se estabelecer uma duração como pulso musical - a exemplo de um pulso BPM - e distribuir impulsos no decorrer deste pulso, de forma a estabelecer métricas e ritmos²⁵. Por exemplo, 2 impulsos igualmente espaçados fazem uma divisão binária básica do pulso. Dois sinais, um com 2 pulsos e outro com 3 pulsos, ambos com os impulsos igualmente espaçados na duração do pulso, resultam na manutenção do pulso, com uma marcação rítmica usada tanto em divisões binárias quanto ternárias em diversos estilos de música étnica e tradicional.(41) Os próprios valores absolutos destes impulsos resultam em proporções entre as amplitudes dos sinais convoluídos. Este recurso da métrica estabelecida pela convolução com impulsos é explorado na montagem *Trenzinho de caipiras impulsivos*. Os recursos explorados incluem a criação de amálgamas sonoras provenientes da síntese granular e esta montagem é já uma ponte para a próxima seção. Veja especialmente a figura 2.26. O código da peça está na

²⁴ Veja na próxima seção.

²⁵ Lembrando que a convolução com o impulso resulta no som deslocado ao instante de ocorrência do impulso.

seção B.2.4 e disponível como parte da toolbox MASSA.

Fonte em movimento, receptor em movimento, efeito Doppler

Retomada a exposição da subseção 2.1.7, quando uma fonte ou um receptor está em movimento, suas características idealmente são atualizadas a cada amostra do sinal digital. As velocidades são decompostas com relação à direção de cada ouvido. Assim, dada a velocidade v_s da fonte (s de *source*), positiva se a fonte se move no sentido de se afastar do receptor e a velocidade v_r do receptor, positiva no sentido de se aproximar da fonte, a frequência é dada pela conhecida fórmula do efeito Doppler:

$$f = \left(\frac{v_{som} + v_r}{v_{som} + v_s} \right) f_0 \quad (2.74)$$

Com esta frequência e com as relações dadas pela nova DII da nova localização da fonte, pode-se realizar o efeito Doppler. Há um acréscimo para maior fidelidade ao fenômeno físico: um aumento da potência recebida. Pode-se entender que este aumento em potência é proporcional à velocidade relativa, que, a cada segundo, acrescenta aquele trecho percorrido, pela forma de onda, de potência: $\Delta P = P_0 \left(\frac{v_r - v_s}{343,2} \right)$, com P_0 a potência do sinal.

Assim, pode-se obter a amplitude e frequência de uma fonte em movimento. Esteja esta fonte à frente do receptor a uma distância horizontal de y_0 metros e altura z_0 metros. A distância é dada por $D_i = \left\{ d_i = \sqrt{y_i^2 + z_0^2} \right\}_0^{\Lambda-1}$ onde $y_i = y_0 + v_s - v_r$ se consideradas v_s e v_r horizontais. A amplitude muda com a distância e com o fator de potência citado acima (veja a subseção 2.1.2 para a conversão de potência para amplitude):

$$A_i = \left\{ \frac{z_0}{d_i} A_{\Delta P} \right\}_0^{\Lambda-1} = \left\{ \frac{z_0}{\sqrt{y_i^2 + z_0^2}} \sqrt{\frac{v_r - v_s}{343,2} + 1} \right\}_0^{\Lambda-1} \quad (2.75)$$

Observe que o fator alterador de amplitude devido à distância é par enquanto a o fator devido à variação de potência é antissimétrico com relação ao cruzamento da fonte e do receptor. Já a frequência tem uma progressão simétrica com relação à altura, ou seja, os mesmos semitons (ou frações) acrescentados na aproximação são decrescidos no afastamento. Além disso, a transição é abrupta caso a fonte e o receptor se cruzem com exatidão, caso contrário, há uma progressão monotônica. No caso considerado, em que há uma altura fixa z_0 , é necessário observar a componente de velocidade na direção entre o observador e a fonte:

$$F_i = \{f_i\}_0^{\Lambda-1} = \left\{ \frac{v_{som} + v_r \frac{y_i}{\sqrt{z_0^2 + y_i^2}}}{v_{som} + v_s \frac{y_i}{\sqrt{z_0^2 + y_i^2}}} f_0 \right\}_0^{\Lambda-1} \quad (2.76)$$

No Apêndice A.2 está uma implementação em Python do efeito Doppler como descrito acima, com as alterações de frequência e amplitude, e com o cruzamento da fonte com o receptor.

Filtragens e ruídos (subseções 2.2.4 e 2.2.3)

Com os filtros as possibilidades explodem ainda mais vertiginosamente. Pode-se convoluir um sinal para reverberá-lo, para remover algum ruído, para gerar distorções ou para tratamento com intuito estético mesmo. Por exemplo, a aplicação de um filtro passa banda em que deixa passar somente entre $1kHz$ e $3kHz$, resulta em sons que parecem de telefone ou de televisão antiga. Ao remover com alguma precisão somente a frequência de oscilação da rede elétrica (usualmente $50Hz$ ou $60Hz$) e harmônicas, pode-se remover ruídos introduzidos por equipamentos de áudio. Um uso mais incrementado e propriamente musical é realizar filtragens em bandas específicas e usar estas bandas pré-estabelecidas como um parâmetro adicional das notas.

Inspirado nos instrumentos tradicionais, pode-se aplicar uma a filtragem dependente do tempo (3).

Cascatas destes filtros podem realizar filtragens complexas e mais precisas. A montagem *Ruidosa faixa* explora este recursos, realizando filtragens em ruídos diversos e síntese de ruídos

diversos. O código da peça está no Apêndice B.2.5 e disponível online como parte da MASSA.

Estes recursos utilizados em conjunto podem incidir na realização de um efeito chamado *chorus*. A exemplo do que ocorre com um coro de cantores, neste efeito o som é realizado com diversas pequenas modificações, potencialmente aleatórias, em parâmetros como frequência central, presença (ou ausência) de vibrato e tremolo e suas características, equalizações, volume etc. Para o resultado final, estas versões do som inicial são então mixadas (ver equação 2.30). A peça *Chorus infantil* realiza chorus de formas diferentes em sons diferentes e seu código está no Apêndice B.2.7. A *toolbox* MASSA disponibiliza online este *script*.

Reverberação

Com a mesma nomenclatura da subseção 2.1.7 sobre espacialização, a reverberação tardia pode ser modelada como a convolução com um trecho de ruído colorido rosa, marrom ou preto, em decaimento exponencial de amplitude com relação também ao tempo. Assim, a atenuação nos agudos e irregularidade suave na resposta em frequência são contempladas de forma bastante satisfatória.(34, 42) Linhas de atrasos podem ser adicionadas como prefixo ao ruído com o decaimento, assim contemplado as duas partes temporais da reverberação: as primeiras reflexões e a reverberação tardia. Pode-se, para melhora de qualidade, calcular geometricamente a localização da última superfície em que cada frente de onda refletiu antes de chegar ao ouvido nos primeiros 100 – 200 milissegundos e aplicar um filtro passa-baixas como em 2.2.3. O ruído colorido pode ser introduzido gradualmente, desde o instante inicial dado pela incidência direta (i.e. sem reflexão alguma e dada pela DTI e DII), em *fade-in*, atingindo o máximo no começo da 'reverberação tardia', quando as incidências geométricas perdem sua importância para as estatísticas do ruído em decaimento.

Como um exemplo de implementação, seja Δ_1 a duração do primeiro período e Δ_R a duração total da reverberação ($\Lambda_1 = \Delta_1 f_a$, $\Lambda_R = \Delta_R f_a$). Pode-se associar uma probabilidade p_i de uma

reincidência do som na i -ésima amostra com amplitude em decaimento exponencial. Seguindo as indicações na subseção 2.1.7, a reverberação R_i^1 do primeiro período pode ser descrita desta forma:

$$R_i^1 = \{r_i^1\}_0^{\Lambda_1-1} : r_i^1 = \begin{cases} 10^{\frac{V_{dB}}{20} \frac{i}{\Lambda_1-1}} & \text{com probabilidade } p_i = \left(\frac{i}{\Lambda_1}\right)^2 \\ 0 & \text{com probabilidade } 1 - p_i \end{cases} \quad (2.77)$$

Onde V_{dB} é o decaimento total considerado em decibels, tipicamente $-80dB$ ou $-120dB$. Já a reverberação R_i^2 , do segundo período, pode ser emulada por um ruído marrom R_i^m (ou rosa R_i^r) em decaimento exponencial:

$$R_i^2 = \{r_i^2\}_{\Lambda_1}^{\Lambda_R-1} = \left\{ 10^{\frac{V_{dB}}{20} \frac{i}{\Lambda_R-1}} \cdot r_i^m \right\}_{\Lambda_1}^{\Lambda_R-1} \quad (2.78)$$

Como:

$$R_i = \{r_i\}_0^{\Lambda_R-1} : r_i = \begin{cases} 1 & \text{se } i = 0 \\ r_i^1 & \text{se } 1 \leq i < \Lambda_1 - 1 \\ r_i^2 & \text{se } \Lambda_1 \leq i < \Lambda_R - 1 \end{cases} \quad (2.79)$$

a aplicação da reverberação representada por R_i é a simples convolução de R_i (chamada 'resposta ao impulso' da reverberação) com a sequência sonora T_i , tal como descrito na subseção 2.2.3.

A reverberação é conhecida por causar bastante interesse no ouvinte e tornar as sonoridades mais agradáveis. Além disso, a modificação do espaço em que são reverberadas a música, ou as sonoridades individualmente, constitui um macete (quase um clichê) para causar surpresa e interesse no ouvinte.

Envoltória ADSR

A variação de volume no decorrer de um som é crucial para nossa percepção de timbre. A envoltória de volume chamada ADSR (sigla de *Attack-Decay-Sustain-Release*) possui numerosas implementações em sintetizadores em hardware e software. Uma implementação pioneira pode ser encontrada no Hammond Novachord de 1938 e algumas variantes são citadas logo abaixo.(43)

A envoltória ADSR escolástica é caracterizada por 4 parâmetros: duração do ataque (tempo que o som demora para atingir seu volume máximo), duração do decaimento (segue ao ataque imediatamente), nível de volume de sustentação (em que o volume fica estável após o decaimento) e duração de soltura (após a sustentação, nesta duração o volume decai a zero). Note que o tempo de sustentação não é especificado, pois é resultante da duração em si menos os tempos de ataque, decaimento e soltura.

A aplicação da envoltória ADSR com durações Δ_A , Δ_D e Δ_R , duração total Δ e nível de sustentação a_S , dado como fração da amplitude máxima, em uma sequência sonora t_i qualquer, pode ser feita da seguinte forma:

$$\begin{aligned}
 \{a_i\}_0^{\Lambda_A-1} &= \left\{ \xi \left(\frac{1}{\xi} \right)^{\frac{i}{\Lambda_A-1}} \right\}_0^{\Lambda_A-1} & \text{ou} & \quad \left\{ \frac{i}{\Lambda_A-1} \right\}_0^{\Lambda_A} \\
 \{a_i\}_{\Lambda_A}^{\Lambda_A+\Lambda_D-1} &= \left\{ a_S^{\frac{i-\Lambda_A}{\Lambda_D-1}} \right\}_{\Lambda_A}^{\Lambda_A+\Lambda_D-1} & \text{ou} & \quad \left\{ 1 - (1-a_S) \frac{i-\Lambda_A}{\Lambda_D-1} \right\}_{\Lambda_A}^{\Lambda_A+\Lambda_D-1} \\
 \{a_i\}_{\Lambda_A+\Lambda_D}^{\Lambda-\Lambda_R-1} &= \{a_S\}_{\Lambda_A+\Lambda_D}^{\Lambda-\Lambda_R-1} \\
 \{a_i\}_{\Lambda-\Lambda_R}^{\Lambda-1} &= \left\{ a_S \left(\frac{\xi}{a_S} \right)^{\frac{i-(\Lambda-\Lambda_R)}{\Lambda_R-1}} \right\}_{\Lambda-\Lambda_R}^{\Lambda-1} & \text{ou} & \quad \left\{ a_S - a_S \frac{i+\Lambda_R-\Lambda}{\Lambda_R-1} \right\}_{\Lambda-\Lambda_R}^{\Lambda-1}
 \end{aligned} \tag{2.80}$$

Com $\Lambda_X = \lfloor \Delta \cdot f_a \rfloor \quad \forall \quad X \in (A, D, R,)$ e ξ um valor pequeno que torne o *fade in* e o *fade out*

satisfatórios, e.g. $\xi = 10^{\frac{-80}{20}} = 10^{-4}$ ou $\xi = 10^{\frac{-40}{20}} = 10^{-2}$. Quanto menor for ξ mais lento é o *fade*, a exemplo de α da figura 2.14. Já os termos do lado direito de 2.80 podem realizar as entrada e saída do som a partir da intensidade zero, por serem lineares. Esquemáticamente, a figura 2.21 mostra esta envoltória ADSR, uma implementação clássica que comporta variações diversas. Por exemplo, entre o ataque e o decaimento, pode-se adicionar uma partição adicional em que a amplitude máxima perdura. Outro exemplo comum é o uso de traçados mais elaborados para o ataque ou para o decaimento. A montagem musical *ADa e SaRa*, disponível no Apêndice B.2.8 e na MASSA, explora diversas destas configurações da envoltória ADSR

$$\{t_i^{ADSR}\}_0^{\Lambda-1} = \{t_i \cdot a_i\}_0^{\Lambda-1} \quad (2.81)$$

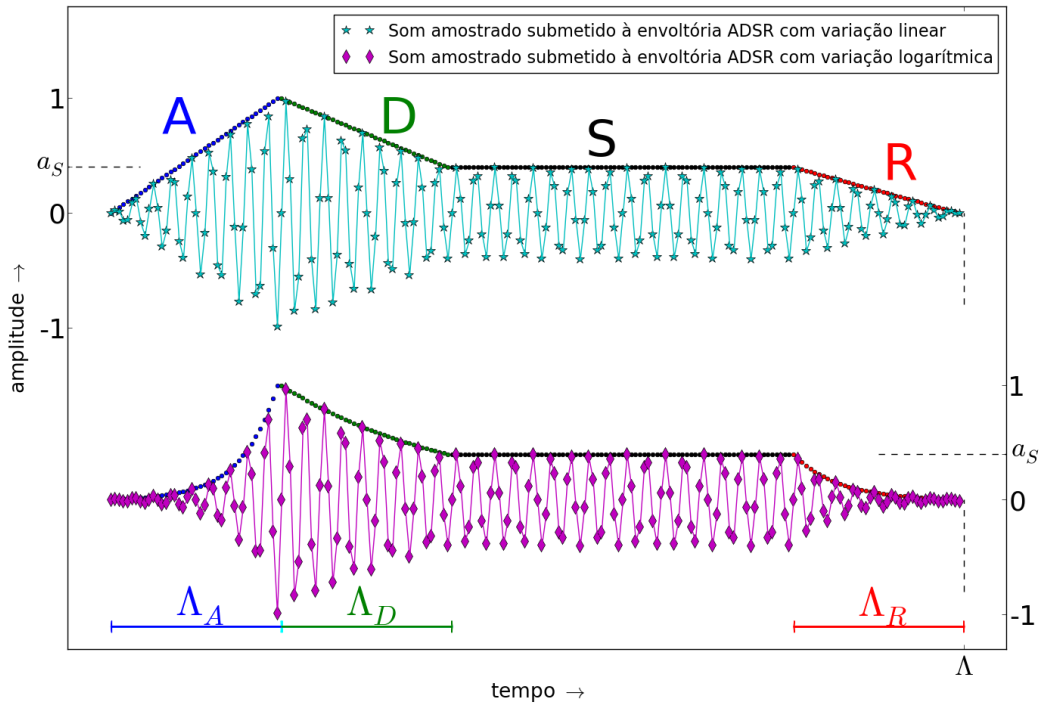


Figura 2.21 – Envoltória ADSR (Attack, Decay, Sustain, Release) e uma sequência sonora arbitrária submetida à envoltória. A variação linear de amplitude está acima. Abaixo a variação de amplitude é exponencial.

2.3 Organização de notas em música

Seja $S_j = \left\{ s_j = T_i^j = \{t_i^j\}_{i=0}^{\Lambda_j-1} \right\}_{j=0}^{H-1}$ uma sequência S_j de H eventos musicais s_j . Seja S_j chamada uma 'estrutura musical' composta de eventos s_j que são também estruturas musicais, p.ex. notas. Esta seção é dedicada às técnicas que tornam S_j interessante e agradável na audição.

Os elementos de S_j podem ser sobrepostos por mixagem, como na equação 2.30 e figura 2.11, formando intervalos e acordes. Este é o 'pensamento vertical' da música. A concatenação de S_j , como na equação 2.31 e na figura 2.12, forma sequências melódicas e ritmos, associados ao 'pensamento horizontal' na música. A frequência fundamental f e o momento de início (ataque) são, em geral, as características mais importantes dos elementos de S_j . Estas viabilizam músicas de alturas (harmonia e melodia) e a presença da métrica temporal e ritmos, respectivamente.

2.3.1 Afinação, intervalos, escalas e acordes

Afinação

O dobro da frequência é uma oitava ascendente ($f = 2f_0$). A divisão da oitava em doze notas é o cânone da música ocidental clássica, além de usos cerimoniais/religiosos e étnicos observados fora da tradição ocidental.(7) Doze semitons equidistantes para o ouvido formam uma oitava, portanto, se $f = 2^{\frac{1}{12}} f_0$, há um semitom entre f_0 e f . O fator $\varepsilon = 2^{\frac{1}{12}}$, um semitom, forma uma grade de notas no espectro audível. Fixada uma frequência f , as frequências fundamentais possíveis estão separadas por intervalos múltiplos de ε . Esta precisão absoluta é característica de implementações computacionais, os instrumentos reais possuem desvios destas frequências para melhor compatibilizar os harmônicos de suas notas. Além disso, a referência fixa $\varepsilon = 2^{\frac{1}{12}}$ caracteriza a afinação de temperamento por igual. Há afinações com intervalos propostos como razões de inteiros de baixa ordem, com fruto da observação de comportamentos físicos. As

primeiras destas afinações foram formalizadas por volta de 2 mil anos antes do advento do temperamento por igual.(3)

Duas afinações emblemáticas são:

- A **intonação justa** consiste em notas da escala diatônica associadas a razões de inteiros de pequena ordem como apontados pela série harmônica. As razões básicas estão contidas no modo jônico (dó a dó nas teclas brancas do piano, veja abaixo na subseção 2.3.1): 1, 9/8, 5/4, 4/3, 3/2, 5/3, 15/8, 2/1. Os intervalos são considerados com relação às notas da escala e são usados o semitom 16/15, o 'tom menor' 10/9 e o 'tom maior' 9/8. Há diferentes formas de realizar uma divisão de 12 notas.
- A **afinação pitagórica** baseia-se no intervalo 3/2 (quinta justa). O modo jônico fica: 1, 9/8, 81/64, 4/3, 3/2, 27/16, 243/128, 2/1. Os intervalos são também considerados com relação às notas da escala. Além dos intervalos do modo, são usados a segunda menor 256/243, a terça menor 32/27, a quarta aumentada 729/512, a quinta diminuta 1024/729, a sexta menor 128/81 e a sétima menor 16/9.

Para a realização de microtonalidade²⁶, pode-se usar reais não inteiros para a sequência de alturas, ou modificar o fator $\varepsilon = 2^{\frac{1}{12}}$ e continuar usando inteiros. Por exemplo, uma afinação bastante próxima da série harmônica em si é proposta na forma da divisão da oitava em 53 notas: $\varepsilon_2 = 2^{\frac{1}{53}}$.(44) As notas nesta divisão da oitava em 53 notas se relacionam por inteiros com ε_2 . Note que se S_i é uma sequência de alturas relacionadas por ε_1 , um mapeamento para notas relacionadas por ε_2 constitui uma nova sequência $S'_i = \{s'_i\} = \{s_i \frac{\varepsilon_1}{\varepsilon_2}\}$. A montagem musical *Micro tom* explora recursos microtonais e seu código está no Apêndice B.3.3, assim como na MASSA online.

²⁶ O uso de intervalos menores que o semitom é chamado microtonalidade e tem usos ornamentais e estruturantes da música. A divisão da oitava em 12 notas possui fundamentos físicos mas não deixa de ser uma *convenção* adotada inclusive pela música erudita clássica de origem europeia. Outras afinações são incidentes. Para citar somente um exemplo, a música tradicional tailandesa utiliza uma divisão da oitava em sete notas igualmente espaçadas ($\varepsilon = 2^{\frac{1}{7}}$), resultando em intervalos que pouco se assemelham aos intervalos na divisão de doze notas.(7)

Intervalos

Em proporções de $\varepsilon = 2^{\frac{1}{12}}$ entre as frequências das notas (i.e. um semitom), os intervalos do sistema de 12 notas são representados por inteiros. A tabela 2.22 resume as características de cada intervalo: sua nomenclatura tradicional, características de consonância e dissonância e número de semitons de cada um.

Tabela 2.22 – Intervalos musicais, suas notações tradicionais, classificações básicas de dissonância e número de semitons. As consonâncias perfeitas são os uníssonos, as quintas e as oitavas justas (J). As consonâncias imperfeitas são as terças e as sextas maiores (M) e menores (m). As dissonâncias fortes são as segundas menores e sétimas maiores. As dissonâncias brandas são as segundas maiores e as sétimas menores. O primeiro caso especial consiste na quarta justa, que é consonante perfeita se considerada uma inversão da quinta justa, caso contrário pode ser considerada uma dissonância ou uma consonância imperfeita. O segundo caso especial é o trítono (4aum, 5dim, tri). Este é consonante em algumas culturas. Já para a música tonal, o trítono indica dominante e busca sua resolução em uma terça ou sexta e, por esta instabilidade, é considerado intervalo dissonante.

consonâncias		
perfeitas:	notação tradicional	número de semitons
	1J, 5J, 8J	0, 7, 12
imperfeitas:	3m, 3M, 6m, 6M	3, 4, 8, 9
dissonâncias		
fortes:	notação tradicional	número de semitons
	2m, 7M	1, 11
brandas:	2M, 7m	2, 10
casos especiais		
consonante ou dissonante:	notação tradicional	número de semitons
	4J	5
dissonante na tradição ocidental:	trítono, 4aum, 5dim	6

A nomenclatura, com base em imposições e conveniências do sistema tonal, e de aspectos práticos da manipulação de notas, pode ser especificada assim:(3, 7)

- Intervalos por número de grados entre as notas: primeira (uníssonos), segunda, terça, quarta, quinta, sexta, sétima, oitava. Nona, décima, décima primeira, etc, são os intervalos compostos, de uma ou mais oitavas + um intervalo dentro da oitava, que caracteriza o intervalo composto. Os intervalos são representados pelos dígitos numéricos, e.g. 1, 3,

5 é um uníssono, uma terça e uma quinta.

- Qualidades de cada intervalo: as consonâncias perfeitas - i.e. uníssono, quarta, quinta e oitava - são 'justas'. As consonâncias imperfeitas - i.e. terças e sextas - e as dissonâncias - i.e. segundas e sétimas - podem ser maiores ou menores. Exceção para o trítono.
- A quarta justa é tida como consonante perfeita ou dissonante de acordo com o contexto e arcabouço teórico. Como regra geral, pode ser considerada consonante salvo casos em que é usada de passagem para uma quinta ou terça como resolução.
- O trítono é dissonante na música ocidental por caracterizar a "dominante" no sistema tonal (veja abaixo na subseção 2.3.2) e representar instabilidade. Algumas culturas entoam o intervalo como consonante.
- Um intervalo maior, decrescido de um semitom, resulta em um intervalo menor. Um intervalo menor, acrescido de um semitom, resulta em um intervalo maior.
- Um intervalo justo (uníssono, quarta justa, quinta justa, oitava justa), ou um intervalo maior (segunda maior 2M, terça maior 3M, sexta maior 6M ou sétima maior 7M), acrescido de um semitom, resulta em um intervalo aumentado (p.ex. terça aumentada 3aum com cinco semitons). A quarta aumentada é também chamada de trítono (4aum tri).
- Um intervalo justo, ou um intervalo menor (segunda menor 2m, terça menor 3m, sexta menor 6m ou sétima menor 7m), decrescido de um semitom, resulta em um intervalo diminuto. A quinta diminuta é também chamada de trítono (5dim tri).
- Um intervalo aumentado, acrescido de um semitom, resulta em um intervalo 'mais que aumentado' e um intervalo diminuto, decrescido de um semitom, resulta em um intervalo 'mais que diminuto'.
- Caso as notas soem simultaneamente, o intervalo é harmônico.

- Caso as notas soem em sequência no tempo, o intervalo é melódico. A ordem das notas, primeiro a nota mais grave ou a mais aguda, resulta em um intervalo ascendente ou descendente, respectivamente.
- Passada a nota mais grave para a oitava acima, ou a nota mais aguda uma oitava para baixo, o intervalo é invertido. Um intervalo, somado à sua inversão, resulta 9 ($7m$ inverte para $2M$: $7m + 2M = 9-$). Um intervalo maior invertido resulta em um intervalo menor e vice-versa. Um intervalo aumentado invertido resulta diminuto e vice-versa, assim como o mais que aumentado resulta em mais que diminuto e vice-versa. Um intervalo justo invertido resulta igualmente justo.
- Um intervalo maior que a oitava é dito 'intervalo composto' e é classificado como o intervalo entre as mesmas notas, mas na mesma oitava. São também especificados por 7 acrescido deste intervalo: $11J$ é uma oitava mais uma quarta ($7+4J=11J$), $9M$ é uma oitava mais uma segunda maior ($7+2M=9M$).

Os intervalos aumentados/diminutos e mais que aumentados/diminutos são consequências do sistema tonal. Os graus da escala (veja abaixo na subseção 2.3.1) são realmente notas diferentes, com funções e usos específicos. Assim, em uma escala de dó bemol maior, a tônica - primeiro grau - é dó bemol, não si, e a sensível - sétimo grau - é si bemol, não lá sustenido ou dó duplo bemol. De forma semelhante, o segundo grau de uma escala pode estar a um semitom do primeiro grau, assim como a sensível (sétimo grau a um semitom ascendente do primeiro grau), momento no qual há uma terça diminuta entre os dois semitons do sétimo e segundo grau da escala pois o primeiro grau está entre os dois graus próximos a ele: segundo e sensível.(11)

Esta descrição resume a teoria tradicional dos intervalos musicais.(11) A montagem *Intervalos entre alturas* explora estes intervalos de formas isoladas e diversas. O código está no Apêndice B.3.1 e disponível online com a *toolbox* MASSA.(2)

Escalas

Uma escala é um conjunto ordenado de alturas. Usualmente, as escalas se repetem a cada oitava. A sequência ascendente com todas as notas da divisão da oitava em 12 intervalos iguais, separados pela razão $\varepsilon = 2^{\frac{1}{12}}$, é a escala cromática de temperamento por igual. Há 5 divisões perfeitamente simétricas da oitava dentro da escala cromática. Estas divisões são consideradas escalas pelos usos fáceis e peculiares que disso provém. Como inteiros aos quais $\varepsilon = 2^{\frac{1}{12}}$ é elevado para multiplicar f_0 , as escalas são:

$$\begin{aligned}
 \text{cromática} &= E_i^c = \{e_i^c\}_0^{11} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\} = \{i\}_0^{11} \\
 \text{tons inteiros} &= E_i^t = \{e_i^t\}_0^5 = \{0, 2, 4, 6, 8, 10\} = \{2.i\}_0^5 \\
 \text{terças menores} &= E_i^{tm} = \{e_i^{tm}\}_0^3 = \{0, 3, 6, 9\} = \{3.i\}_0^3 \\
 \text{terças maiores} &= E_i^{tM} = \{e_i^{tM}\}_0^2 = \{0, 4, 8\} = \{4.i\}_0^2 \\
 \text{trítonos} &= E_i^{tt} = \{e_i^{tt}\}_0^1 = \{0, 6\} = \{6.i\}_0^1
 \end{aligned} \tag{2.82}$$

Assim, a terceira nota da escala em tons inteiros com $f_0 = 200\text{Hz}$ é $f_3 = \varepsilon^{e_3^t} \cdot f_0 = 2^{\frac{6}{12}} \cdot 200 \approx 282.843\text{Hz}$. Estas 'escalas' ou padrões geram estruturas estáveis pelas simetrias internas, e podem ser repetidas de forma eficiente e sustentada. Abaixo, na seção 2.3.7, será retomado o assunto das simetrias. A montagem *Cristais* expõe cada uma destas escalas tanto melódica quanto harmonicamente e seu código está no Apêndice B.3.2. Como parte da MASSA, este código é disponibilizado online.

As escalas diatônicas:

$$\begin{aligned}
\text{menor natural} &= \text{modo eólico} = E_i^m = \{e_i^m\}_0^6 = \{0, 2, 3, 5, 7, 8, 10\} \\
\text{modo lócrio} &= E_i^{mlo} = \{e_i^{mlo}\}_0^6 = \{0, 1, 3, 5, 6, 8, 10\} \\
\text{maior} &= \text{modo jônico} = E_i^M = \{e_i^M\}_0^6 = \{0, 2, 4, 5, 7, 9, 11\} \\
\text{modo dórico} &= E_i^{md} = \{e_i^{md}\}_0^6 = \{0, 2, 3, 5, 7, 9, 10\} \\
\text{modo frígio} &= E_i^{mf} = \{e_i^{mf}\}_0^6 = \{0, 1, 3, 5, 7, 8, 10\} \\
\text{modo lídio} &= E_i^{ml} = \{e_i^{ml}\}_0^6 = \{0, 2, 4, 6, 7, 9, 11\} \\
\text{modo mixolídio} &= E_i^{mmi} = \{e_i^{mmi}\}_0^6 = \{0, 2, 4, 5, 7, 9, 10\}
\end{aligned} \tag{2.83}$$

possuem apenas intervalos maiores, menores e justos. Única exceção para o trítono, que se apresenta como quarta aumenta ou quinta diminuta.

Todas as escalas diatônicas seguem o padrão de intervalos sucessivos tom, tom, semitom, tom, tom, tom, semitom, e pode-se escrever:

$$\begin{aligned}
\{d_i\} &= \{2, 2, 1, 2, 2, 2, 1\} \\
e_0 &= 0 \\
e_i &= d_{(i+\kappa)\%7} + e_{i-1} \quad \text{para } i > 0
\end{aligned} \tag{2.84}$$

Com $\kappa \in \mathbb{N}$. Para cada modo, há um único valor de $\kappa \in [0, 6]$. Por exemplo, uma breve inspeção revela que $e_i^{ml} = d_{(i+2)\%7} + e_{i-1}^{ml}$. Portanto, $\kappa = 2$ para o modo lídio.

A escala menor possui duas formas adicionais, melódica e harmônica:

$$\begin{aligned}
\text{menor natural (igual acima)} &= E_i^m = \{e_i^m\}_0^6 = \{0, 2, 3, 5, 7, 8, 10\} \\
\text{menor harmônica} &= E_i^{mh} = \{e_i^{mh}\}_0^6 = \{0, 2, 3, 5, 7, 8, 11\} \\
\text{menor melódica} &= E_i^{mm} = \{e_i^{mm}\}_0^{14} = \{0, 2, 3, 5, 7, 9, 11, 12, 10, 8, 7, 5, 3, 2, 0\}
\end{aligned} \tag{2.85}$$

O traçado ascendente e descendente da escala menor melódica é necessário para a presença da sensível (sétimo e último grau, separado por um semitom da oitava, realça a polarização na tônica) no trajeto ascendente, o que não é necessário quando descende, retomando a forma natural. Já a escala harmônica apresenta a sensível, mas não evita o intervalo de segunda aumentada entre o sexto e sétimo graus, pois não precisa contemplar trajetória melódica, apenas apresentar a sensível tão cara ao sistema tonal (a sensível tende à tônica, afirmando-a).(30) Outras escalas podem ser representadas da mesma forma, como as pentatônicas e os modos de transposição limitados de Messiaen.(45)

Acordes

A ocorrência simultânea de notas é observada através dos acordes. Destes, a base na música tonal são as tríades. Estas constituem-se de duas terças sucessivas, em 3 notas: fundamental, terça e quinta. Um acorde invertido é aquele que apresenta no grave outra nota que não a fundamental. A posição fechada é aquela em que não cabe nota alguma do acorde entre quaisquer duas notas consecutivas.(11) As tríades, sem inversão, na forma fechada e com a fundamental em 0, são:

$$\begin{aligned}
 \text{tríade maior} &= A_i^M = \{a_i^M\}_0^2 = \{0, 4, 7\} \\
 \text{tríade menor} &= A_i^m = \{a_i^m\}_0^2 = \{0, 3, 7\} \\
 \text{tríade diminuta} &= A_i^d = \{a_i^d\}_0^2 = \{0, 3, 6\} \\
 \text{tríade aumentada} &= A_i^a = \{a_i^a\}_0^2 = \{0, 4, 8\}
 \end{aligned}
 \tag{2.86}$$

Para considerar outra terça sobreposta à quinta, basta acrescentar 10 ao final para a téttrade com sétima menor ou 11 para a téttrade com sétima maior. As inversões e posições abertas podem ser obtidas com a adição seletiva de 12 às componentes.

Acordes triádicos incompletos, com notas adicionais (acordes 'sujos'), e não triádicos são comuns. Orientações gerais são:

- Uma quinta constitui uma fundamental confirmada pelo intervalo.
- A terça maior ou menor aponta a qualidade maior ou menor do acorde.
- Todo trítone, especialmente se formado entre uma terça maior e uma sétima menor, tende a resolver em uma terça ou uma sexta.
- Evita-se a duplicação de nota. Na necessidade de duplicação, a ordem de preferência é: a fundamental, a quinta, a terça e a sétima.
- Pode-se formar acordes com notas diferentes das triádicas, especialmente se dentro de alguma lógica recorrente ou encadeamento musical que justifique as notas diferentes.
- Acordes formados por sucessões de intervalos diferentes das terças, como as quartas ou as segundas, são recorrentes dentre composições de tonalismo avançado ou experimentais.
- A repetição de sucessões de acordes (ou suas características) fixa uma trajetória pela recorrência e permite a introdução de formações exóticas sem que haja incoerência.

2.3.2 Harmonias atonal, tonal, expansão e modulação

A omissão dos encadeamentos básicos do sistema tonal é chave para a obtenção de harmonias modal e atonal. No caso desta ausência de estruturas tonais mínimas, se as notas coincidirem com alguma escala diatônica (veja as equações 2.83) ou forem em número pequeno, pode-se dizer que a harmonia é modal. Caso encadeamentos tonais básicos estejam ausentes, as notas não coincidam com alguma das escalas diatônicas e forem diversas e dissonantes (com relação às outras notas) o suficiente para evitar redução por polarizações, a harmonia é atonal. Nesta classificação, a harmonia modal não é tonal e não é atonal. A harmonia modal está reduzida à

incidência de notas dentro de escalas diatônicas (ou simplificações) e à ausência de estruturas tonais. Pode-se perceber, pelo conceito, que a harmonia atonal é difícil de ser realizada.(46)

Harmonia atonal

De fato, as técnicas de música atonal visam estruturas que evitam o vínculo da audição a modos e relações tonais. A dificuldade é tamanha que o dodecafonismo surgiu. A proposta do dodecafonismo é usar um conjunto de notas, idealmente usa-se as 12 notas, e executar uma a uma destas notas na mesma ordem. Neste contexto, a tônica fica difícil de se estabelecer. Mesmo assim, a audição ocidental procura traços tonais na música e facilmente os encontra por caminhos inesperados e por vezes tortuosos. A utilização de intervalos dissonantes, especialmente trítonos sem resoluções, nonas, segundas e sétimas, reforça a ausência de tonalidade. Neste contexto, para criação da peça, pode-se:

- Repetir notas. Ao considerar a repetição imediata um prolongamento da incidência anterior, notas iguais, sem outras notas entre elas, não adicionam informação relevante.
- Soar notas adjacentes ao mesmo tempo, formando intervalos harmônicos e acordes.
- Apresentar durações livremente, desde que respeitada a ordem de aparição das notas.
- Para variação, além dos recursos de ampliação, transposição e translação, são usados o inverso, o retrógrado e o retrógrado do inverso. Veja nas subseções 2.3.5 e 2.3.9 para maiores detalhes.
- Variações de instrumentação, articulação, espacialização e outras possibilidades de apresentação da estrutura de notas.

A harmonia atonal pode ser observada, de forma paradigmática, dentro destas condições. Vale apontar que boa parte do que escreveram os compositores emblemáticos para o dodecafonismo,

como Alban Berg e o próprio Schoenberg, consiste de aplicações parciais e livres destas técnicas. Várias peças fazem até mesmo pontes propositalmente entre técnicas tonais e atonais.

Harmonia tonal

No século XX, músicas rítmicas e com ênfase em sonoridades/timbres ampliaram as concepções de tonalidade e harmonia. Ainda assim, a harmonia tonal tem forte presença nas vertentes artísticas e comerciais. Considera-se o próprio dodecafonismo de natureza tonal pois consiste na negação das características tonais de polarização.

Na música tonal ou modal, acordes, como os listados nas equações 2.86, constituídos com a fundamental em cada grau de uma escala, como das equações 2.83, formam os pilares do campo harmônico. A observação de progressões incidentes de acordes e regras para encadeamentos é o objeto de estudo da harmonia musical. Mesmo uma melodia monofônica gera campos harmônicos e pode-se observar acordes sugeridos por cada passagem.

Na 'harmonia tonal tradicional', a escala pode ter como tônica (primeiro grau da escala) qualquer nota e pode ser maior (com as mesmas notas do modo jônico) ou menor (as notas do eólico constituem a escala "menor natural" e esta possui versões harmônica e melódica, veja a equação 2.85). A escala escolhida é base para tríades, cada uma com a fundamental em um grau da escala: $\hat{1}, \hat{2}, \hat{3}, \hat{4}, \hat{5}, \hat{6}, \hat{7}$. Para a formação do acorde, são consideradas a terceira nota e a quinta nota acima da fundamental, considerando a própria nota como a primeira e utilizando as notas da escala. Pode-se anotar $\hat{1}, \hat{3}, \hat{5}$ para o acorde do primeiro grau, formado pelo primeiro grau da escala e central para uma música tonal. Secundários são os acordes do quinto grau $\hat{5}, \hat{7}, \hat{2}$ ($\hat{7}$ sustenido no caso da escala menor) e do quarto grau $\hat{4}, \hat{6}, \hat{1}$. Depois são considerados os outros graus. A harmonia tradicional consiste em convenções e técnicas estilísticas de encadeamento destes acordes, formados em cada grau da escala.(30)

A 'harmonia funcional' atribui funções a estes três acordes centrais e busca compreender seus

usos através destas funções. O acorde formado sobre o primeiro grau é o acorde de tônica (T ou t se tônica maior ou menor) e tem a função de manter um centro, um "chão" na música. O acorde formado sobre o quinto grau é a dominante (D, a dominante é sempre maior) e tem a função de tender à tônica, direcionar a música para ela. A tríade formada sobre o quarto grau é a subdominante (S ou s se subdominante maior ou menor) e tem a função de distanciar a música da tônica. O sistema se baseia em afirmar a tônica através de encadeamentos tônica-dominante-tônica expandidos com outros acordes das formas mais diversas.

A estes três acordes, são associadas as outras tríades. Na escala maior, a associada relativa (tônica relativa Tr, subdominante relativa Sr e dominante relativa Dr) é a tríade formada uma terça abaixo e a associada anti-relativa (tônica anti-relativa Ta, subdominante anti-relativa Sa e a dominante anti-relativa Da) é a tríade formada na terça acima. Na escala menor ocorre o mesmo, mas a tríade a uma terça abaixo é chamada anti-relativa (tA, sA) e a tríade a uma terça acima é chamada de relativa (tR, sR). As exatas funções e efeitos musicais destes acordes é motivo de bastante controvérsia. A tabela 2.23 mostra a relação entre as tríades formadas em cada grau da escala maior.

Tabela 2.23 – *Resumo das funções harmônicas tonais para a escala maior. A tônica é o centro da música, a dominante tende à tônica e a subdominante se distancia da tônica. Os três acordes podem, a princípio, serem substituídos livremente pelas respectivas relativas ou anti-relativas.*

relativa	acorde principal da função	anti-relativa
6, 1, 3	tônica: 1, 3, 5	3, 5, 7
3, 5, 7	dominante: 5, 7, 2	[7, 2, 4#]
2, 4, 6	subdominante: 4, 6, 1	6, 1, 3

É necessário que a dominante anti-relativa forme um acorde menor, por isso a alteração do quarto grau um semitom para cima 7#. O acorde diminuto 7, 2, 4, geralmente é considerado uma 'dominante com sétima e sem fundamental'.(47) Em modo menor, há a alteração do 7 por um semitom ascendente para que haja somente um semitom de separação com a tônica, permitindo a dominante (que deve ser maior e tender à tônica). Assim, a dominante é sempre

maior, tanto em escalas maior ou menor e, por este motivo, mesmo em tom menor, a dominante relativa permanece a uma terça abaixo e a antirrelativa a uma terça acima.

Expansão tonal: funções individuais e medianas cromáticas

Cada um destes acordes pode ser confirmado e se desenvolver com uma execução de sua dominante ou subdominante individual, que é o acorde com base na tríade formada a uma quinta acima ou uma quinta abaixo respectivamente. Estas dominantes/subdominantes individuais, por sua vez, possuem também subdominantes e dominantes individuais passíveis de uso. Assim, em uma dada tonalidade, pode ocorrer qualquer acorde, por mais distante que seja do campo harmônico e das notas da escala, desde que a ocorrência apresente um percurso coerente de dominantes e subdominantes até a tonalidade de origem.

As medianas, ou 'medianas cromáticas', são duas para cada acorde: a mediana cromática superior, formada com a fundamental na terça do acorde original, e a inferior, formada com a quinta na terça do acorde original. São acordes formados também a uma terça, mas com uma alteração cromática com relação ao acorde de origem. Caso haja duas alterações cromáticas, i.e. duas notas alteradas por um semitom cada com relação ao acorde original, a mediana é chamada 'duplamente cromática'. Também são duas para cada acorde: a superior, com terça na quinta do acorde original, e a inferior, com terça na fundamental da tríade original. Observe que um acorde maior possui medianas maiores e medianas duplamente cromáticas maiores. Um acorde menor possui medianas menores e medianas duplamente cromáticas menores. Esta relação entre acordes é considerada de tonalismo avançado, por vezes até de expansão e dissolução do tonalismo, e tem efeitos fortes e marcantes embora perfeitamente consonantes. As medianas foram utilizadas a partir do final do romantismo por Wagner, Litz, Richard Strauss dentre outros e são bastante simples de serem realizadas.(30, 48)

Modulação

A modulação é a mudança da tonalidade em que se encontra a música. Caracteriza-se a modulação através da observação das tonalidades de partida e chegada e da forma de transição. As tonalidades são sempre tidas como relacionadas por quintas e suas relativas e antirrelativas. São formas de efetuar a modulação:

- A transposição do discurso para a nova tonalidade, sem preparação alguma. É procedimento típico do barroco embora seja incidente em outros períodos. Por vezes chamada de modulação frasal ou modulação sem preparação.
- O uso cuidadoso de uma dominante individual, e possivelmente também a subdominante individual, para afirmar a mudança da tônica e campo harmônico.
- Uso de alterações cromáticas para atingir um acorde da nova tonalidade a partir de algum acorde da tonalidade anterior. Chamada de modulação cromática.
- O destaque para uma única nota, possivelmente repetida ou suspensa sem acompanhamento, comum às tonalidades de saída e chegada, constitui uma forma peculiar de introduzir o novo campo harmônico.
- Mudança da função, sem a modificação das notas em si, de um acorde para contemplar nova tonalidade. Procedimento chamado de enarmonia.
- A manutenção do centro tonal e mudança da qualidade maior para menor (ou vice-versa) da tonalidade é a modulação paralela. A tonalidade de mesma tônica e outra qualidade é chamada homônima.

A importância da dominante a torna pivô natural das modulações, o que desemboca no círculo das quintas.(30, 47–49) A montagem musical *Acorde cedo* explora estas relações entre acordes. Seu código está disponível do Apêndice B.3.4 e online como parte da MASSA.(2)

2.3.3 Contraponto

A condução de linhas melódicas simultâneas, chamadas vozes, é o contraponto. A bibliografia percorre formas sistemáticas de condução de vozes e desemboca em gêneros escolásticos como cânones, invenções e fugas. É possível resumir regras principais do contraponto e é dito que o próprio Beethoven - dentre outros - esboçou uma síntese deste tipo.

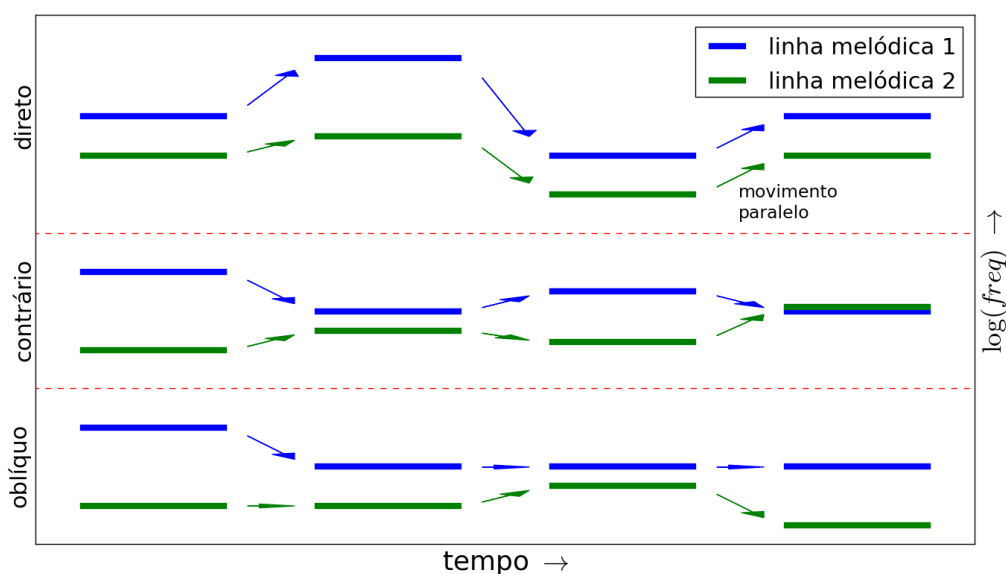


Figura 2.24 – Movimentos diferenciados pelo contraponto com vistas a preservar a independência entre as vozes. 3 tipos de movimentos: direto, contrário e oblíquo, categorizam as possibilidades. O movimento paralelo é um tipo de movimento direto.

O propósito do contraponto é conduzir as vozes de forma que soem independentes. Cruciais para isso são as movimentações relativas, das vozes duas a duas, categorizadas em: movimento direto, oblíquo e contrário conforme a figura 2.24. O movimento paralelo é um movimento oblíquo. A regra de ouro é cuidar que os movimentos diretos não terminem em consonância perfeita. O movimento paralelo deve só ocorrer entre consonâncias imperfeitas e não mais do que três vezes consecutivas. As dissonâncias podem ser não admitidas ou usadas seguidas e precedidas de consonâncias em graus conjuntos, i.e. notas vizinhas na escala. Os movimentos

que levam a nota a uma vizinha soam mais coerentes. Na presença de 3 ou mais vozes, a importância melódica recai sobre as vozes mais aguda e mais grave, nesta ordem.(50–52)

Estas regras foram usadas na montagem *Conta ponto*, o código está no Apêndice B.3.5 e disponível online junto à MASSA.

2.3.4 Ritmo

A noção rítmica é dependente de eventos separados por durações.(11) Estes eventos podem ser ouvidos individualmente se espaçados por ao menos $50 - 63ms$. Para que a separação temporal entre eles possa ser apreciada como duração, ela deve ser maior, por volta de $100ms$.(14) Pode-se sumarizar a transição de durações ouvidas como alturas para a apreciação em ritmo da seguinte forma:(14, 53)

Tabela 2.25 – Transição das durações ouvidas individualmente para alturas.

duração (s) frequência (Hz)	zona de percepção de durações em ritmo										transição	-				
	...	32,	16,	8,	4,	2,	1,	1/2,	1/4,	1/8,	$\left\ \frac{1}{16} = 62,5ms, \frac{1}{20} = 50ms \right\ $	1/40	1/80	1/160	1/320	1/640
	...	1/32,	1/16,	1/8,	1/4,	1/2,	1,	2,	4,	8,	16,	40	80	160	320	640
											transição	zona de percepção de durações em altura				

A banda de durações marcada como transição está minimizada pois os limites não são bem definidos: a duração em que se começa a perceber uma frequência fundamental ou uma separação entre as ocorrências é dependente da pessoa e de características do som.(3, 14)

A métrica rítmica costuma se basear em uma duração básica chamada pulso. O pulso tipicamente compreende durações entre $0.25 - 1.5s$ (respectivamente 240 e 40BPM). Na educação musical e estudos cognitivistas, costuma-se associar esta gama de frequências de pulsação às durações entre batidas do coração, da inspiração/expiração ou entre os passos ao caminhar.(3, 11)

O pulso é subdividido em partes iguais e também é repetido sequencialmente. Estas relações (de divisão e de concatenação) costumam seguir relações de números inteiros de baixa ordem

²⁷. Uma exposição esquemática está na figura 2.26.

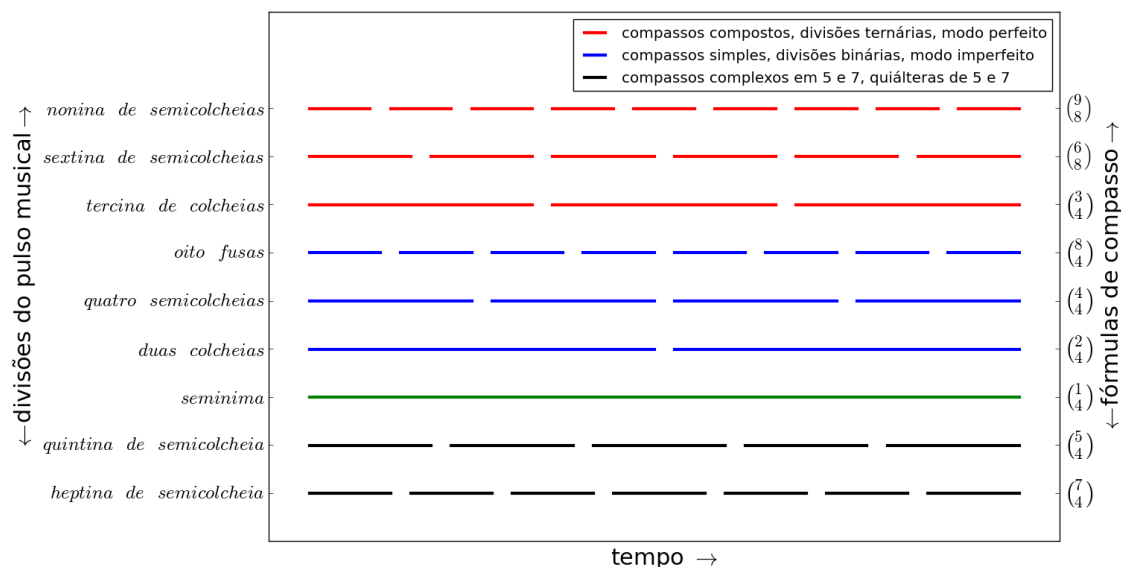


Figura 2.26 – Divisões e aglomerações do pulso musical para estabelecimento de métrica. Ao lado esquerdo estão as divisões da semínima estabelecida como pulso. Ao lado direito, fórmulas de compasso que especificam as mesmas métricas, mas na escala das aglomerações do pulso musical.

As relações duais (compassos simples e divisões binárias) costumam ocorrer em ritmos de dança e ocasiões festivas, e são chamadas imperfeitas. As relações ternárias incidem mais na música ritualística e relacionada ao sagrado e são ditas perfeitas.

As unidades mais fortes (acentuadas) são as que consistem nas 'cabeças das divisões'. A cabeça de uma unidade é a primeira parte da subdivisão. Nas divisões binárias (2, 4 e 8 dos casos considerados), as unidades consideradas fortes se revezam com as fracas (e.g. a divisão em 4 é forte, fraco, meio-forte, fraco). Nas divisões ternárias (3, 6 e 9) à unidade forte (primeira) se

²⁷ Em ordem crescente de ocorrência na música escrita e étnica, as divisões do pulso musical e seus agrupamentos sequenciais no tempo são: 2, 4 e 8, depois 3, 6 (dois grupos de 3 ou 3 grupos de 2) e 9 e 12 (3 e 4 grupos de 3). Por último os primos 5 e 7, completando 1-9 e 12. Outras métricas são menos usuais, como divisões ou agrupamentos em 13, 17, etc, e são incidentes principalmente em contextos de música experimental e erudita do século XX e XXI. Por mais complexas que pareçam, as métricas costumam ser composições e decomposições de 1-9 partes iguais. (3, 41).

sucedem 2 unidades fracas (e.g. a divisão em 3 é forte, fraco fraco)²⁸.

A acentuação em tempo fraco é o contratempo, uma nota iniciada em tempo fraco e cuja duração se prolonga sobre um tempo forte é uma síncopa.

As notas podem ocorrer dentro e fora destas divisões da '*métrica musical*'. Nos casos mais comportados, as notas ocorrem exatamente nestas divisões, com maior incidência em ataques nos tempos fortes. Em casos extremos, não se pode perceber a métrica.(3) Variações pequenas na grade ajudam a compor a interpretação musical e diferenças entre estilos.(10)

Seja o pulso o nível $j = 0$ de agrupamento, o nível $j = -1$ a primeira subdivisão do pulso, o nível $j = 1$ a primeira aglomeração dos pulsos e assim por diante. Desta forma, P_i^j é a i -ésima unidade de pulsos no nível j de agrupamento: P_{10}^0 é o décimo pulso, P_3^1 é a terceira unidade de agrupamento de pulsos (é possível que seja o terceiro compasso), P_2^{-1} é a segunda unidade da subdivisão do pulso.

Especial atenção para os limites de j : as divisões do pulso são durações apreciáveis como ritmo; além disso, as junções do pulso somam, no máximo do escopo, uma música ou um conjunto coeso de músicas. Dito de outra forma: a duração de $P_i^{min(j)}$, $\forall i$, é maior que 50ms e as durações somadas $\sum_{\forall i} P_i^{máx(j)}$ é menor do que alguns minutos ou, no máximo, poucas horas.

Cada nível j possui alguns índices i . Sempre que o índice i possuir três valores diferentes (ou múltiplo de três), há uma relação perfeita. Quando i possuir somente múltiplo dois, quatro ou oito valores diferentes, há uma relação imperfeita, como na figura 2.26.

Qualquer unidade (nota), de uma dada sequência musical que tenha métrica pode ser univocamente assim especificada:

²⁸ A divisão em 6 é considerada composta mas pode ocorrer também como uma divisão binária. Uma divisão binária que sofre então uma divisão ternária resulta em duas unidades divididas em três unidades cada: forte (subdividido em forte fraco, fraco) e fraco (subdividido em forte, fraco, fraco). A outra forma de ocorrer a divisão em 6 é através de uma divisão ternária que sofre então uma divisão binária, resultando em: uma unidade forte (subdividido em forte e fraco) e duas unidades fracas (subdivididas em forte e fraco cada).

$$P_{\{i_k\}}^{\{j_k\}} \quad (2.87)$$

em que j_k é o nível de aglomeração e i_k é a ordem da unidade em si.

Como um exemplo, $P_{3,2,2}^{-1,0,1}$ é a terceira subdivisão P_3^{-1} do segundo pulso P_2^0 do segundo aglomerado de pulsos P_2^1 . Cada unidade ou conjunto de unidades P_i^j pode ser associada a uma sequência de amostras temporais T_i que forma uma nota musical.

A montagem *Poli Hit Mia* utiliza as diferentes métricas e está no Apêndice B.3.6, disponível online como parte da MASSA.

2.3.5 Repetição e variação: motivos e unidades maiores

Dadas as estruturas musicais básicas tanto frequenciais (acordes e escalas) quanto rítmicas (divisões e aglomerações simples, compostas e complexas), é natural apresentar estas estruturas de forma que tenham coesão e sentido.(54) Para tal, é fundamental o conceito de arcos: partindo de algum lugar e voltando, forma-se um arco. A audição de linhas melódicas e harmônicas é permeada de arcos musicais pela natureza cognitiva da escuta musical. Pode-se considerar a nota o menor arco, cada motivo e melodia também um arco. Cada tempo e cada subdivisão, cada compasso e secção da música, constitui um arco próprio. Uma música, cujos arcos não apresentam consistência entre si, pode ser compreendida como uma música sem coesão. A sensação de coerência provém, em grande parte, do tratamento hábil dos arcos de uma peça.

Os arcos musicais são estruturas abstratas e passíveis de operações básicas. Um arco espectral, como um acorde, pode ser invertido, ampliado e permutado, por exemplo. Os arcos temporais, como uma melodia, um motivo, um compasso ou uma nota, são igualmente passíveis de variações. Lembrando que $S_j = \left\{ s_j = T_i^j = \{t_i^j\}_0^{\Lambda_j-1} \right\}_0^{H-1}$ é uma sequência de H eventos musicais s_j , cada evento com suas Λ_j amostras t_i^j (veja no início desta seção 2.3), as técnicas básicas podem

ser descritas assim:

- A translação temporal é o deslocamento δ do material para um outro instante $\Gamma' = \Gamma + \delta$ da música. É uma variação do material com modificação na localização no decorrer da música: $\{s'_j\} = \{s_j^{\Gamma'}\} = \{s_j^{\Gamma+\delta}\}$ onde Γ é a duração entre o começo da peça (ou trecho considerado) e o primeiro evento s_0 da estrutura S_j original. Observe que δ é o deslocamento no tempo.
- A dilatação ou contração temporal é a alteração da duração de cada arco por um fator μ : $s_j'^{\Delta} = s_j^{\mu j \cdot \Delta}$. Possivelmente, $\mu_j = \mu$ constante.
- Reversão temporal consiste em gerar uma sequência com os elementos em ordem invertida da sequência original S_j , assim: $S'_j = \{s'_j\}_0^{H-1} = \{s_{(H-j-1)}\}_0^{H-1}$.
- A translação em altura é o deslocamento τ do material para uma altura diferente $\Xi' = \Xi + \tau$ da música. É uma variação com modificação na localização em altura do material: $\{s'_j\} = \{s_j^{\Xi'}\} = \{s_j^{\Xi+\tau}\}$ onde Ξ é a altura do trecho S_j ou do primeiro evento s_0 da estrutura S_j original. Observe que τ é o deslocamento em altura. Caso τ seja dado em semitons, o deslocamento em frequência é $\tau_f = f_0 \cdot 2^{\frac{\tau}{12}}$ onde f_0 é resultado da adoção de alguma referência: $f_0 = \Xi_{f_0} Hz \sim \Xi_0$ valor absoluto de altura. Para a frequência para qualquer valor de altura: $\Xi_f = \Xi_{f_0} \cdot 2^{\frac{\Xi - \Xi_0}{12}}$. Para altura de qualquer valor de frequência: $\Xi = \Xi_0 + 12 \cdot \log_2\left(\frac{\Xi_f}{\Xi_{f_0}}\right)$. No protocolo MIDI, $\Xi_{f_0} = 55 Hz$ enquanto $\Xi_0 = 33$ de altura marca o lá 1, outro ponto de referência é o $\Xi_{f_0} = 440 Hz$ que corresponde a $\Xi_0 = 69$. A diferença da unidade equivale ao semitom. Não se pode dizer que a unidade é o semitom pois $\Xi = 1$ não é um semitom, é uma nota de frequência audível como ritmo, com menos de 9 ocorrências por segundo (veja a tabela 2.25).
- A inversão intervalar é a inversão do sentido dos intervalos percorridos pelo material. A inversão é estrita se o número de semitons esteja sendo usado como referência para a

operação: $S'_j = \{s'_j\}_0^{H-1} = \left\{s_j^{-\varepsilon_j \cdot f_0}\right\}$, onde ε_j é o fator entre a frequência do evento s_j e a frequência de s_0 . A inversão é tonal caso as distâncias sejam consideradas em termos de números de graus da escala E_k : $S'_j = \{s'_j\}_0^{H-1} = \left\{s_j^{-\varepsilon^{(e_{(j_e)})} \cdot f_0}\right\}_0^{H-1}$ onde $j_e = e_j^{inv}$ é o índice $k = j_e$ em E_k da nota do evento s_j .

- Rotação de elementos musicais é a transferência do último elemento para a posição do primeiro e o deslocamento deste ao penúltimo uma posição para frente. Pode-se definir a rotação de \tilde{n} posições por $S'_n = S_{(n+\tilde{n})\%H}$. Caso $\tilde{n} < 0$, basta usar $\tilde{n}' = H - \tilde{n}$. É razoável associar $\tilde{n} > 0$ com a rotação horária e $\tilde{n} < 0$ com a rotação anti-horária. Mais sobre rotações na subseção 2.3.7.
- A inserção e remoção de materiais na estrutura S_j pode ser ornamental ou estruturante: $S'_j = \{s'_j\} = \{s_j \text{ se condição A, caso contrário } r_j\}$, para qualquer material musical r_j , incluindo o instante vazio. Elementos de R_j podem ser inseridos no começo, como um prefixo de S_j , ao final, como um sufixo, ou no meio, dividindo S_j ou fazendo dele o prefixo e o sufixo. Os dois materiais podem se misturar das formas mais diversas.
- Modificações de articulação, instrumentação e espacialização $s'_j = s_j^{*j}$, com $*j$ a nova característica incorporada pelo elemento s'_j .
- Acompanhamento. Tanto a instrumentação quanto as linhas melódicas presentes na ocorrência de S_j podem sofrer modificações e ser considerada uma variação de S_j em si.

Com estes procedimentos, outros são derivados: o retrógrado do inverso, uma contração temporal com um sufixo externo, etc. As estruturas musicais ressoam no sistema cognitivo devido à própria natureza do pensamento. Em suas várias facetas, uma ideia lida com o mesmo número de elementos e aspectos conectivos entre eles. A música, através de sintonia com estas estruturas mentais, suscita impressões. É, assim, desencadeado todo um processo de ressonância mental e neurológica responsável pelos sentimentos, lembranças e imaginações típicas de uma

audição musical atenta. Esta atividade cortical contribui para a terapia musical, conhecida pela utilidade em casos de depressão e dano neurológico. Considera-se que as regiões do cérebro responsáveis pelo processamento auditivo são também usadas para outras atividades, incluindo linguísticas e matemáticas.(3, 55)

As estruturas paradigmáticas orientam a criação de novos materiais musicais. Uma delas, central, é o dipolo tensão/relaxamento. Relaciona-se com este dipolo, cada outro dipolo tradicional: tônica/dominante, repetição/variação, consonância/dissonância, coerência/rompimento, simetria/assimetria, igualdade/diferença, chegada/saída, perto/longe, parado/em movimento, etc. Já as relações ternárias tendem a se relacionar com o circular e a unificação. A lúcida comunhão ternária, 'modus perfectus', se opõe ao dicotômico passional, 'modus imperfectus'. A seguir está uma exposição dedicada aos arcos direcionais e cíclicos.

2.3.6 Estruturas direcionais

Os arcos podem ser decompostos em duas sequências convergentes: uma que atinge o ápice, e outra que, de forma paradigmática, volta do ápice à região de partida. Este ápice é chamado de clímax pela teoria musical tradicional. Distingue-se entre arcos cujos clímax estão: no começo, no meio, no final, na primeira metade e na segunda metade da duração considerada. Estas estruturas estão na figura 2.27. O parâmetro que varia pode não existir, no caso o arco consiste somente em uma estrutura de referência.(13)

Seja $S_i = \{s_i\}_0^{H-1}$ uma sequência crescente. A sequência $R_i = \{r_i\}_0^{2H-2} = \{s_{(H-1-|H-1-i|)}\}_0^{2H-2}$ é uma sequência que apresenta simetria especular perfeita, i.e. a segunda metade é uma versão espelhada da primeira. Segundo conceitos musicais, o clímax está exatamente no meio da sequência. Pode-se modificar isso com o uso de sequências de tamanhos diferentes. Toda a teoria matemática de sequências, já estabelecida e ensinada corriqueiramente em cursos de cálculo III, pode ser utilizada para geração destes arcos.(13, 56) Teoricamente, estas sequências,

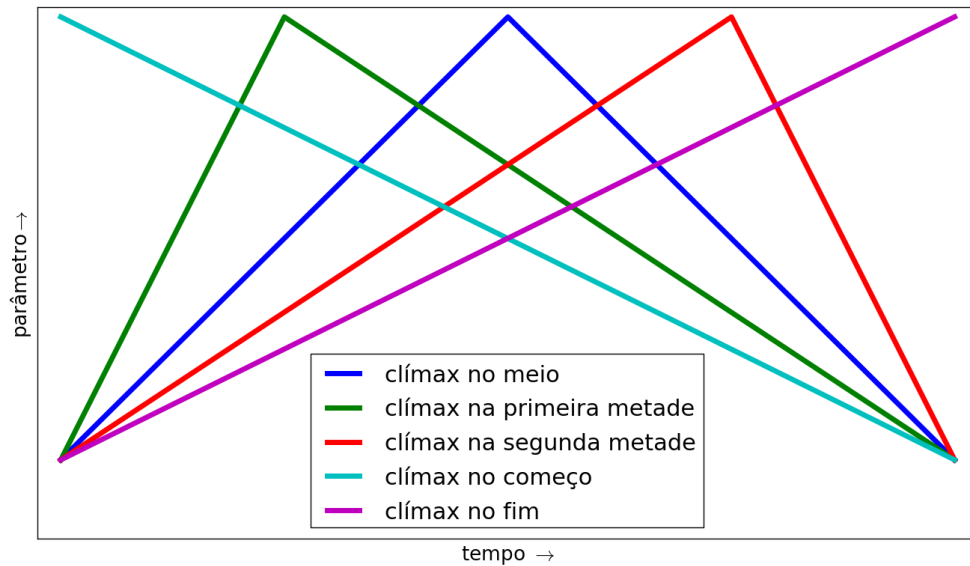


Figura 2.27 – Distinções canônicas do clímax musical em uma melodia e outros domínios. As possibilidades diferenciadas são: clímax no começo, clímax na primeira metade, clímax no meio, clímax na segunda metade, clímax no fim. Não está especificado o eixo das ordenadas pois pode não haver variação paramétrica real, neste caso a estrutura é uma referência.

aplicadas desta forma a qualquer característica dos eventos musicais, produzem arcos, pois implicam no afastamento e retorno de uma parametrização inicial. Assim, é possível para uma mesma sequência de eventos possuir um número de arcos distintos, com tamanhos e clímax diferentes. Este é um recurso interessante e útil e a correlação dos arcos resulta na coerência da escuta.(48)

Na prática, e historicamente, tem especial importância a razão áurea e, por isso, a sequência de Fibonacci. A sequência de Lucas permite uma generalização da sequência de Fibonacci que pode ser compreendida facilmente. Dados dois números quaisquer x_0 e x_1 , obtém-se a sequência de Lucas: $x_n = x_{n-1} + x_{n-2}$. Quanto maior for n , mais a razão $\frac{x_n}{x_{n+1}}$ se aproxima da razão áurea: 1.61803398875.... A sequência converge rápido mesmo para valores iniciais bem discrepantes. Seja $x_0 = 1$ e $x_1 = 100$, e $y_n = \frac{x_n}{x_{n+1}}$ uma sequência auxiliar. O erro em percentagem dos primeiros valores desta sequência com relação à proporção áurea é, aproximadamente, $\{e_n\} =$

$$\left\{100 \frac{y_n}{1.61803398875} - 100\right\}_1^{10} = \{6080.33, -37.57, 23, -7.14, 2.937, -1.09, 0.42, -0.1601, 0.06125, -0.02338\}.$$

A sequência de Fibonacci apresenta aproximadamente a mesma progressão de erros, mas começa já no segundo passo pois $\frac{1}{1} \approx \frac{101}{100}$.

A montagem sonora *Dirracional* expõe estes arcos em estruturas direcionais. Seu código está no Apêndice B.3.7 e disponível online como parte da MASSA.(2)

2.3.7 Estruturas cíclicas

O entendimento filosófico de que o pensamento humano é fundamentado na percepção de semelhanças e diferenças, dentre os estímulos e objetos, coloca as simetrias no cerne do processo cognitivo.(57) Matematicamente, as simetrias são grupos algébricos e um grupo finito é sempre isomorfo a um grupo de permutações. Pode-se dizer que as permutações representam quaisquer simetrias em um sistema finito. Na música, as permutações são ubíquas e estão presentes em técnicas, o que confirma seu papel central. A aplicação sucessiva das permutações gera arcos cíclicos.(12, 58, 59) A esta abordagem foram dedicados dois trabalhos acadêmicos para geração de estruturas musicais.(60, 61)

Qualquer conjunto de permutações pode ser utilizado como gerador de grupos algébricos.(59)

As propriedades que definem um grupo G são:

$$\begin{aligned} \forall p_1, p_2 \in G &\Rightarrow p_1 \bullet p_2 = p_3 \in G && \text{(propriedade de fechamento)} \\ \forall p_1, p_2, p_3 \in G &\Rightarrow (p_1 \bullet p_2) \bullet p_3 = p_1 \bullet (p_2 \bullet p_3) && \text{(propriedade da associatividade)} \\ \exists e \in G : & p \bullet e = e \bullet p \quad \forall p \in G && \text{(existência do elemento neutro)} \\ \forall p \in G, \exists p^{-1} : & p \bullet p^{-1} = p^{-1} \bullet p = e && \text{(existência do inverso)} \end{aligned} \quad (2.88)$$

Da primeira propriedade conclui-se que toda permutação pode ser operada com outra permutação. De fato, pode-se aplicar uma permutação p_1 , depois outra p_2 e, se comparadas as ordena-

ções inicial e final, há uma permutação p_3 .

Todo elemento p operado consigo mesmo um número suficiente de vezes n atinge o elemento neutro $p^n = e$, caso contrário o grupo seria infinito (gerado por p). O menor $n : p^n = e$ é chamado de ordem do elemento. Assim, uma permutação finita p aplicada sucessivamente atinge a ordenação inicial dos elementos, formando um ciclo. Este ciclo, se utilizado para parâmetros de notas musicais, implica em um arco musical cíclico.

Estes arcos podem ser efetuados pelo uso conjunto de diferentes permutações. Como exemplo histórico, a tradição chamada *change ringing* concebe música através de sinos tocados um após o outro e então tocados novamente, mas em uma ordem diferente. Este processo se repete até que se atinja a ordenação inicial. O conjunto de ordenações diferentes percorridas é um *peal*. A tabela 2.28 representa um *peal* tradicional de 3 sinos (1, 2 e 3) que explora todas as suas ordenações. Cada linha apresenta uma ordenação dos sinos a ser tocada. As permutações estão entre as linhas. Neste caso, a estrutura musical consiste em permutações propriamente ditas e algumas permutações diferentes operam para o comportamento cíclico.

Tabela 2.28 – *Change Ringing: Peal (padrão) com 3 sinos. As permutações estão entre as ordenações. Cada linha é uma ordenação dos sinos, cada ordenação é tocada, uma linha por vez.*

1	2	3
2	1	3
2	3	1
3	2	1
3	1	2
1	3	2
1	2	3

A utilização de permutações na música pode ser resumida da seguinte forma: seja $S_i = \{s_i\}$ uma sequência de eventos musicais s_i (e.g. notas) e p uma permutação. $S'_i = p(S_i)$ consiste nos mesmos elementos de S_i mas em ordem diferente. As permutações podem ser escritas em duas notações: cíclica ou natural. A notação natural consiste na ordem dos índices resultante da permutação. Assim, convencionada a ordenação original dada pela sequência de seus índices

[0 1 2 3 4 5 ...] a permutação é notada pela sequência que produz (ex. [1 3 7 0 ...]). Na notação cíclica, a permutação é a troca de um elemento pelo da frente, e o último pelo primeiro.

Não é necessário permutar os elementos de S_i , mas somente alguma ou algumas de suas características. Assim, seja p^f uma permutação nas frequências e S_i uma sequência de notas básicas como expostas ao final de 2.1.6. A nova sequência $S'_i = p^f(S_i) = \{s_i^{p(f)}\}$ consiste nas mesmas notas musicais, na mesma ordem e com as mesmas características, com as frequências fundamentais permutadas segundo o padrão que p^f apresenta.

Duas sutilezas deste procedimento. 1) A permutação p não precisa envolver todos os elementos de S_i , i.e. ela pode operar em subconjuntos de S_i . 2) Nem todos os elementos s_i precisam ser executados a cada consulta de estado realizada. Para exemplificar, seja S_i uma sequência de notas musicais s_i . Se i vai de 0 a n , e $n > 4$, a cada compasso de 4 notas pode-se executar as primeiras 4 notas. As outras notas de S_i podem incidir nos compassos em que as permutações aloquem estas notas para as primeiras quatro notas de S_i .

A cada uma destas permutações p_i , segundo a exposição acima, relaciona-se: dimensões das notas em que opera (frequência, duração, *fades*, intensidade, etc) e período de incidência (a cada quantas consultas é aplicada a permutação). Na realização das notas de S_i , uma forma fácil e coerente é executar as primeiras n notas²⁹.

No Apêndice C está a implementação computacional disponibilizada em.(2, 60, 61)

2.3.8 Idioma musical?

Existem diversas empreitadas que se propõem a modelar e explorar entendimentos sobre a 'linguagem musical', a 'linguística aplicada à música' ou ainda para discernimento entre o que seriam diferentes 'idiomas musicais'.(9, 30, 48, 53) De forma simples, um idioma musical é fruto da escolha de materiais básicos e repetição de elementos e da repetição de relações en-

²⁹ A execução de notas disjuntas de S_i equivale a modificar a permutação e executar as primeiras notas.

tre elementos presentes no decorrer da música. Nestas questões, as dicotomias são salientes, como explicadas na subseção 2.3.5: repetição e variação, relaxamento e tensão, equilíbrio e desequilíbrio, consonância e dissonância, etc.

2.3.9 Usos musicais

Primeiro, a nota básica foi definida e caracterizada em termos claros e quantitativos (seção 2.1). Em seguida, a composição interna da nota foi abordada, e compreendidas as transições internas e tratamentos imediatos (seção 2.2). Por fim, esta presente seção dedica-se a organizar estas notas em música. A gama de recursos e consequente infinidade de possibilidades de resultados é situação típica e cara às artes.(8, 30)

Existem estudos para cada recurso apresentado. Por exemplo, pode-se obter as harmonias triádicas 'suja's' (com notas não pertencentes à tríade) através de sobreposições de quartas justas. Outro exemplo interessante é a presença simultânea de ritmos em diferentes métricas, constituindo o que chama-se de *polirritmia*. A montagem musical *Poli-hit mia* explora estas métricas simultâneas através de trem de impulsos convoluídos com as notas que compõem cada linha. Seu código está no Apêndice B.3.6 e disponível online como parte da MASSA.

As escalas microtonais são importantes na música do século XX (44) e possuem resultados muito marcantes, como os quartos de tom na música indiana. A sequência musical *MicroTom* explora estes recursos, incluindo melodias microtonais e harmonias microtonais com várias notas em um âmbito de alturas bastante reduzido. Seu código está no Apêndice B.3.3 e disponível online como parte da MASSA.

Como também apontado na subseção 2.2.6, os vínculos entre parâmetros são formas poderosas de se obter peças e montagens musicais. O número de notas permutadas pode variar no decorrer da música, revelando vínculo com a duração da peça. As harmonias podem constituir-se triádicas (eqs. 2.86) com notas replicadas em várias oitavas e mais numerosas quanto menor a

profundidade e frequência de vibratos (eqs. 2.56, 2.57, 2.58, 2.59, 2.60), dentre outras incontáveis possibilidades.

As simetrias apresentadas nas divisões da oitava (eqs. 2.82) e as simetrias apresentadas através das permutações (tabela 2.28 e eqs. 2.88) podem ser usadas em conjunto. Nas peças *3 trios* esta associação é feita de forma sistemática para possibilitar uma audição a ela dedicada. Esta é uma peça instrumental e não consta dentre os códigos dos Apêndices e da MASSA.(62)

O *PPEPPS* (Pure Python EP: Projeto Solvente) é um EP sintetizado com os recursos apresentados neste trabalho. Com pouca parametrização, o programa gera músicas inteiras, permitindo a composição de músicas e conjuntos de músicas com facilidade. Através de poucas linhas de código e, pela execução, pode-se obter uma pasta com as músicas. Esta facilidade e entrega tecnológica abre possibilidades estéticas, de compartilhamento e educacionais.

3 *Conclusões e trabalhos futuros*

No capítulo anterior está um sistema conciso que relaciona elementos musicais ao som digital. *Scripts* implementam estas relações, e em conjunto foram nomeados MASSA (Música e Áudio em Sequências e Séries Amostrais). A exposição didática destes desenvolvimentos no capítulo anterior destina-se a facilitar a utilização do arcabouço.

As possibilidades abertas por estes resultados envolvem a criação de interfaces de geração de ruídos e outros sons em alta fidelidade (*hi-fi*), experimentos psicoacústicos e a utilização destes resultados para fins artísticos e didáticos. A incorporação de conhecimentos em programação é bastante facilitada através de recursos audiovisuais, o que já realizamos por práticas de *livecoding* e cursos focados em ferramentas especializadas, como o Puredata e o ChuckK. Está prevista a utilização destes resultados com métodos de inteligência artificial para geração de materiais artísticos.

A disposição online destes conteúdos na forma de hipertexto junto aos códigos e exemplos sonoros, todos em licenças livres, facilita colaborações e geração de subprodutos em co-autoria, e com isso a expansão da MASSA com novas implementações e desenvolvimentos das montagens musicais. Explorações sistemáticas de parametrizações (dos tremolos, da ADSR, etc) em alta fidelidade tem utilidade artística e é possibilitada por este trabalho com controle amostral. Tal descrição analítica precisa, junto às implementações computacionais, não foi atingida anteriormente, como mostra o Apêndice G com uma visita aos trabalhos relacionados.

Este trabalho também teve resultados não previstos, como a formação de grupos de interesse em torno da questão criativa aliada à computação. Neste contexto, destaca-se o grupo labMacambira.sf.net, que reúne colaboradores de todo o Brasil e alguns fora do país. Este grupo já apresentou contribuições relevantes em diferentes áreas como Democracia Direta Digital, ferramentas de georreferenciamento e atividades artísticas e educacionais, como cursos, workshops e apresentações artísticas. Vários destes resultados estão no Apêndice F e no acervo online, que ultrapassa 700 vídeos, documentações escritas, diversos software originais e contribuições em software externos utilizados no mundo todo, como o Firefox, Scilab, LibreOffice, GEM/Puredata, para citar somente alguns exemplos.(63–65)

Há um aumento no número de pesquisas relacionadas à música em andamento no campus de São Carlos da USP, o que sugere facilidade para estabelecer parcerias. As publicações acadêmicas efetivadas durante este mestrado também apontam para uma multidisciplinaridade, tratando diretamente de questões humanas como artes, filosofia, humor e linguagem falada e escrita, através de artifícios lineares e estatísticos.(66–69) Os desdobramentos estão alcançando redes sociais e teorias epistemológicas com base em pesquisas prévias, dos orientadores deste trabalho, com forte presença de redes complexas e processamento de linguagem natural.

REFERÊNCIAS

1 ZAMPRONHA, E. S. *Notação, representação e composição: um novo paradigma da escritura musical*. São Paulo: Annablume, 2000.

2 FABBRI, R.; COSTA, L. F.; OLIVEIRA JUNIOR, O. N. MASSA: música e áudio em sequências e séries amostrais. Disponível em:
<<http://labmacambira.git.sourceforge.net/git/gitweb.cgi?p=labmacambira/massa;a=blob;f=README.txt>>. Acesso em: 18 nov. 2012.

3 ROEDERER, J. G. *The physics and psychophysics of music: an introduction*. 4th ed. New York: Springer Verlag, 2009. ISBN: 9780387094700.

4 MENEZES, F. *A acústica musical em palavras e sons*. Cotia: Ateliê Editorial, 2004. ISBN:8574802166.

5 EVEREST, F. A. *Master handbook of acoustics*. New York: McGraw-Hill, 2001. ISBN:0071360972.

6 ALTEN, S. R. *Audio in media*. 9th ed. Boston: Wadsworth Publishing, 2010. ISBN: 9780495572398.

7 WISNIK, J. M. *O som e o sentido*. São Paulo: Companhia das Letras, 1999. ISBN: 8571640424.

8 WEBERN, A. *O caminho para a música nova*. São Paulo: Novas Metas, 1984.

9 LERDAHL, F.; JACKENDOFF, R. *A generative theory of tonal music*. Cambridge: MITPress, 1983. ISBN: 026262107X.

10 COOK, P. R. *Real sound synthesis for interactive applications*. Natick: A K Peters, 2002. ISBN: 1568811683.

11 LACERDA, O. *Compêndio de teoria elementar da música*. 9a ed. São Paulo: Ricordi Brasileira, 1966.

12 ZAMACOIS, J. *Curso de formas musicales: con numerosos ejemplos musicales*. Barcelona: Idea Books, 2002. ISBN: 8482362356.

- 13 SCHOENBERG, A.; STRANG, G.; STEIN, L.; SEINCMAN, E. *Fundamentos da composição musical*. São Paulo: EDUSP, 1991. ISBN: 8531400457.
- 14 ROADS, C. *Microsound*. Cambridge: MIT Press, 2004. ISBN: 9780262681544.
- 15 CHACON, S.; HAMANO, J.; PEARCE, S. *Pro Git*. Berkeley: Apress, 2009. ISBN: 9781430218333.
- 16 OLIPHANT, T. E. *A guide to numPy*. Spanish Fork: Trelgol Publishing, 2006.
- 17 COURNAPEAU, D. *Audiolab, a python package to make noise with numpy arrays*. Disponível em: <<http://www.ar.media.kyotou.ac.jp/members/david/software/audiolab>>. Acesso em: 18 jan. 2013.
- 18 VAN ROSSUM, G.; DRAKE JR, F. L. *Python tutorial*. S. l.: Odense Universitet, Institut for Matematik og Datalogi, 1995.
- 19 VAN ROSSUM, G. V.; DRAKE JR, F. *Python reference manual*. S. l.: Iuniverse, 1995. ISBN: 9780595136681.
- 20 RAYMOND, E. S. *The art of Unix programming*. Boston: Addison-Wesley, 2004. ISBN:0131429019.
- 21 LESSIG, L. *Free culture: how big media uses technology and the law to lock down culture and coltrol creativity*. New York: Penguin Press, 2004. ISBN: 9781594200069.
- 22 EMMERSON, S. *Living electronic music*. Aldershot: Ashgate, 2007. ISBN: 9780754655480.
- 23 LOVELOCK, W.; HOLLOWAY, E. *A concise history of music*. London: Bell, 1953. ISBN: 9780713506785.
- 24 CHOWNING, J. M. Digital sound synthesis, acoustics and perception: a rich intersection. In: COST G-6 CONFERENCE ON DIGITAL AUDIO EFFECTS (DAFX-00), 2000, Verona. *Proceedings...* Verona: University of Verona, 2000. p. DAFX-1-DAFX-6.
- 25 OPPENHEIM, A. V.; SCHAFER, R. W. *Discrete-time signal processing*. Upper Saddle River: Prentice-Hall, 1989. ISBN: 9780131988422.

26 CHENG, C. I.; WAKEFIELD, G. H. Introduction to head-related transfer functions (HRTFs): representations of HRTFs in time, frequency, and space. *Journal of the Audio Engineering Society*, v. 49, n. 4, p. 231-249, 2001.

27 HEEGER, D. Perception lecture notes: auditory pathways and sound localization. Disponível em:

<<http://www.cns.nyu.edu/~david/courses/perception/lecturenotes/localization/localization.html>>. Acesso em: 16 out. 2012.

28 ALGAZI, V.; DUDA, R. O.; THOMPSON, D. M.; AVENDANO, C. The cipic hrtf database. In: IEEE WORKSHOP ON APPLICATIONS OF SIGNAL PROCESSING TO AUDIO AND ACOUSTICS, 2001, New York. *Proceedings...* Piscataway: Institute of Electrical and Electronics Engineers, 2001. p. W2001-1- W2001-4.

29 CARTY, B.; LAZZARINI, V. Binaural HRTF based spatialisation: new approaches and implementation. In: INTERNATIONAL CONFERENCE ON DIGITAL AUDIO EFFECTS (DAFX-09), 12., 2009, Como. *Proceedings...* London: Department of Electronic Engineering- Queen Mary University of London, 2009. p. 49-54.

30 SCHOENBERG, A. *Harmonia*. São Paulo: Ed. UNESP, 1999.

31 GEIGER, G. Table lookup oscillators using generic integrated wavetables. In: INTERNATIONAL CONFERENCE ON DIGITAL AUDIO EFFECTS (DAFX-06), 9., 2006, Montreal. *Proceedings...* Montreal: McGill University, 2006. p. DAFX-169-DAFX-172.

32 BRISTOW-JOHNSON, R. Wavetable synthesis 101, a fundamental perspective. In: AES CONVENTION, 101., 1996, Los Angeles. *Proceedings....* Disponível em: <<http://www.musicdsp.org/files/Wavetable-101.pdf>>. Acesso em: 22 nov. 2012.

33 DEHAENE, S. The neural basis of the Weber–Fechner law: a logarithmic mental number line. *Trends in Cognitive Sciences*, v. 7, n. 4, p. 145–147, 2003.

34 GUILLAUME, P. *Music and acoustics: from instrument to computer*. London: ISTE, 2006. ISBN: 9781905209262.

35 SMITH, S. W. *The scientist and engineer's guide to digital signal processing*. 2nd. ed. San Diego: California Technical Publishing, 1999.

- 36 SMITH III, J. O. *Mathematics of the discrete fourier transform (DFT) with audio applications*. 2nd ed. 2007. ISBN: 9780974560748. Disponível em: <<https://ccrma.stanford.edu/~jos/log>>. Acesso em: 16 out. 2012.
- 37 MUSIC-DSP source code archive. Disponível em: <musicdsp.org>. Acesso em: 23 jan. 2013.
- 38 PORRES, A. T.; FURLANETE, F.; MANZOLLI, J. Análise de dissonância sensorial de espectros sonoros. In: CONGRESSO DA ANPPOM, 16., 2006, Brasília. *Anais...* Brasília:UnB, 2006.
- 39 PORRES, A.; PIRES, A. Um external de aspereza para puredata & MAX/MSP. In: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO MUSICAL, 12., 2009, Recife. *Anais...* Recife: Sociedade Brasileira de Computação (SBC), 2009.
- 40 SCHOTTSTAEDT, B. *An introduction to FM*. Disponível em: <<https://ccrma.stanford.edu/software/snd/snd/fm.html>>. Acesso em: 12 ago. 2012.
- 41 GRAMANI, J. E. *Rítmica viva: a consciência musical do ritmo*. Campinas: UNICAMP, 1996.
- 42 SMITH III, J. O. *Physical audio signal processing : for virtual musical instruments and audio effects*. Berkeley: Center for New Music and Audio Technologies, 2006.
- 43 ROADS, C. *The computer music tutorial*. Cambridge: MIT Press, 1996.
- 44 WILKINSON, S. R. *Tuning in: microtonality in electronic music: a basic guide to alternate scales, temperaments, and microtuning using synthesizers*. Milwaukee: H. Leonard Books, 1988.
- 45 CLOUGH, J.; ENGEBRETSEN, N.; KOCHAVI, J. Scales, sets, and interval cycles: a taxonomy. *Music Theory Spectrum*, v.21, n. 1, p. 74–104, 1999.
- 46 KOSTKA, S. et al. *Tonal harmony: with an introduction to twentieth*. New York: McGraw-Hill, 1995.
- 47 KOELLREUTTER, H. *Harmonia funcional*. São Paulo: Ricordi, 1986.

48 SALZER, F. *Structural hearing: tonal coherence in music*. New York: Dover Publications, 1962.

49 ALDWELL, E.; SCHACHTER, C.; CADWALLADER, A. *Harmony & voice leading*. Boston: Wadsworth Publishing Company, 2010.

50 FUX, J.; MANN, A. *The study of counterpoint: from Johann Joseph Fux's gradus ad Parnassum*. New York: WW Norton & Company, 1965. ISBN:-10: 0393002772.

51 TRAGTENBERG, L. *Contraponto: uma arte de compor*. 2a ed. São Paulo: Edusp, 2002.

52 SCHOENBERG, A.; STEIN, L. *Preliminary exercises in counterpoint*. London: Faber & Faber, 1963.

52 ASSIS, G. O. A. *Em busca do som*. São Paulo: Ed. UNESP. ISBN: 8539302079.

53 BOULEZ, P. *A música hoje*. 3a ed. São Paulo: Perspectiva, 2005. ISBN: 8527302896.

54 GUIDORIZZI, H. L. *Um curso de cálculo*. 5a ed. Rio de Janeiro: Livros Técnicos e Científicos Editora, 2001. v. 3.

55 DELEUZE, G.; GUATTARI, F. *What is philosophy?* New York: Columbia University Press, 1996. ISBN: 9780231079891.

56 DUCKWORTH, R.; STEDMAN, F. *Tintinnalogia, or, the art of ringing*. Teddington: Echo Library, 2007. ISBN: 9781406826203

57 BUDDEN, F. J. *The fascination of groups*. Cambridge: Cambridge University Press, 1972.

58 FABBRI, R.; MAIA JUNIOR, A. Applications of group theory on granular synthesis. In: SIMPÓSIO BRASILEIRO DE COMPUTAÇÃO MUSICAL, 11., 2007, São Paulo. *Anais...* São Paulo: Sociedade Brasileira de Computação (SBC), 2007.

59 FABBRI, R.; MAIA JUNIOR, A. Applications of group theory on sequencing and spatialization of granular sounds. In: AES Brazil Conference, 6., 2008, São Paulo. *Proceedings...* São Paulo: Sociedade Brasileira de Computação (SBC), 2008. v. 1, p. 1-3.

60 FABBRI, R.; MAIA JUNIOR, A. Applications of group theory on granular synthesis. In: BRAZILIAN SYMPOSIUM ON COMPUTER MUSIC, 11, 2007, São Paulo. *Proceedings...* São Paulo: SBCM, 2007.

61 FABBRI, R.; MAIA JUNIOR, A. Applications of group theory on sequencing and spatialization of granular sounds. In: AES-BRAZIL CONFERENCE, 6, 2008, São Paulo. *Proceedings ...* São Paulo: DMA, 2008.

62 FABBRI, R. 3 trios para oboé, flauta e fagote. 2007. Disponível em: <http://www.estudiolivres.org/el-gallery_view.php?arquivoId=7962>. Acesso em: 10 jan. 2013.

63 LABORATÓRIO Macambira. #labmacambira @ Freenode. Disponível em: <<http://labmacambira.sourceforge.net>>. Acesso em: 18 jan. 2013.

64 WIKI do Lab Macambira. #labmacambira @ Freenode. Disponível em: <http://wiki.nosdigitais.teia.org.br/Lab_Macambira>. Acesso em: 18 jan. 2013.

65 CANAL Vimeo do Lab Macambira (mais de 700 videos). #labmacambira @ Freenode. Disponível em: <<https://vimeo.com/channels/labmacambira>>. Acesso em: 18 jan. 2013.

66 VIEIRA, V.; FABBRI, R.; TRAVIESO, G.; OLIVEIRA JUNIOR, O. N.; COSTA, L. F. A quantitative approach to evolution of music and philosophy. *Journal of Statistical Mechanics: theory and experiment*, v. 2012, n. 8, p. P08010, 2012.

67 AMANCIO, D. R.; FABBRI, R.; OLIVEIRA JUNIOR, O. N.; NUNES, M. G. V.; COSTA, L. F. Distinguishing between positive and negative opinions with complex network features. In: ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS (ACL), 48.; WORKSHOP ON GRAPH-BASED METHODS FOR NATURAL LANGUAGE PROCESSING, 2010, Uppsala. *Proceedings*. Stroudsburg: Association for Computational Linguistics (ACL), 2010. p. 83–87.

68 FABBRI, R. et al. Speech polarity detection using complex networks measures: first explorations. In: INTERNATIONAL WORKSHOP ON COMPLEX NETWORKS (CompleNet), 2., 2010, Rio de Janeiro. *Anais...* Rio de Janeiro: Universidade Federal do Rio de Janeiro (UFRJ), 2010. p. 9.

69 AMANCIO, D. R.; FABBRI, R.; OLIVEIRA JUNIOR, O. N.; NUNES, M. G. V.; COSTA, L. F. Opinion discrimination using complex network features. *Communications in Computer and Information Science*, v. 116, p. 154–162, 2011.

70 FABBRI, R. Repositório da Dissertação. Disponível em: <<http://labmacambira.git.sourceforge.net/git/gitweb.cgi?p=labmacambira/dissertacao;a=tree;h=refs/heads/msc>>. Acesso em: 19 jan. 2013.

71 FABBRI, R.; TAVARES, T. Audioexperiments. Disponível em: <<https://www.assembla.com/spaces/audioexperiments/wiki>>. Acesso em: 12 jan. 2013.

72 FABBRI, R.; MENDES, R. S. Compressão de Áudio via wavelets, aproximações polinomiais e permutações. In: ENCONTRO DOS ALUNOS E DOCENTES DO DEPARTAMENTO DE COMPUTAÇÃO E AUTOMAÇÃO INDUSTRIAL (EADCA), 2., 2009, Campinas. *Anais...* Campinas: Faculdade de Engenharia Elétrica e de Computação - Universidade Estadual de Campinas, 2009. p. 155-158.

73 SMITH III, J. O. Audio signal processing in FAUST. 2012. Disponível em: <<https://ccrma.stanford.edu/jos/aspf/>>. Acesso em: 15 abr. 2013.

74 SMITH III, J. O. Spectral audio signal processing. 2011. Disponível em: <<https://ccrma.stanford.edu/jos/sasp/>>. Acesso em: 15 abr. 2013.

75 SMITH III, J. O. Introduction to digital filters with audio applications. 2007. Disponível em: <<https://ccrma.stanford.edu/jos/filters/>>. Acesso em 10 abr. 2013.

APÊNDICE A – Código computacional dos procedimentos expostos no capítulo 2

A.1 Código Python das relações descritas na seção 2.1

Todas as equações da seção 2.1 estão implementadas abaixo em Python e disponibilizadas como parte da toolbox MASSA.(2)

equações da seção 2.1 em Python

```

1  #-- coding: utf8 --
2  import numpy as n
3  import scikits.audiolab as a
4
5  ##### 2.1.1 Duração
6  # a equação relaciona o número de amostras à duração do som
7  f_a = 44100. # frequência de amostragem
8  Delta = 3.7 # duração de Delta segundos = >
9
10 Lambda = int(f_a*Delta) # número de amostras
11 # 2.1
12 T_i = n.zeros(Lambda) # silêncio com ~ Delta segundos
13
14 # escrita em disco como arquivo PCM (WAV no caso)
15 a.wavwrite(T_i, "silencio.wav", f_a)
16
17 ##### 2.1.2 Volume
18 Lambda = 100. # 100 amostras
19 T_i = n.random.random(Lambda) # 100 amostras quaisquer
20
21 # 2.2 Potência

```

```

22 pot = (T_i**2.).sum()/Lambda
23
24 T2_i = n.random.normal(size=Lambda)
25 pot2 = (T2_i**2.).sum()/Lambda # potência 2
26 # 2.3 Diferença de volume em decibels, dadas as potências
27 V_dB = 10.*n.log10(pot2/pot)
28
29 # 2.4 dobra amplitude => ganha 6 dB
30 T2_i = 2.*T_i
31 pot = (T_i**2.).sum()/Lambda # potência
32 pot2 = (T2_i**2.).sum()/Lambda # potência 2
33 V_dB = 10.*n.log10(pot2/pot)
34 Mm6 = abs(V_dB - 6) < .5 # Mm6 é True
35
36 # 2.5 dobra potência => ganha 3 dB
37 pot2 = 2.*pot
38 V_dB = 10.*n.log10(pot2/pot)
39 Mm3 = abs(V_dB - 3) < .5 # Mm3 é True
40
41 # 2.7 dobra volume => + 10 dB => amplitude * 3.16
42 V_dB = 10.
43 A = 10.**((V_dB/20.))
44 T2_i = A*T_i # A ~ 3.1622776601
45
46 # 2.8 Conversão de decibels em amplificação
47 A = 10.**((V_dB/20.))
48
49
50 ##### 2.1.3 Altura
51 f_0 = 441.
52 lambda_0 = f_a/f_0
53 periodo = n.arcsin(n.random.random(lambda_0)) # amostras quaisquer
54 # 2.9 Som com frequência fundamental f_0
55 Tf_i = n.array(list(periodo)*1000) # 1000 períodos
56
57 # normalizando para convenção no intervalo [-1,1]
58 Tf_i = ((Tf_i-Tf_i.min())/((Tf_i.max()-Tf_i.min())))*2.-1.
59 a.wavwrite(Tf_i, "f_0.wav", f_a) # escrita em disco
60
61
62 ##### 2.1.4 Timbre
63 T = 100000. # número de amostras das sequencias
64 ii = n.arange(T)
65 f = 220.5
66 lambda_f = f_a/f
67 # 2.10 Senoide
68 Sf_i = n.sin(2.*n.pi*f*ii/f_a)
69 # 2.11 Dente de serra

```



```

70 Df_i = (2./lambda_f)*(ii % lambda_f)-1
71 # 2.12 Triangular
72 Tf_i = 1.-n.abs(2.-(4./lambda_f)*(ii % lambda_f))
73 # 2.13 Onda quadrada
74 Qf_i = ((ii % lambda_f) < (lambda_f/2))*2-1
75
76 Rf_i = a.wavread("22686__acclivity__oboe-a-440_periodo.wav")[0]
77 # 2.14 Período amostrado
78 Tf_i = Rf_i[n.int64(ii) % len(Rf_i)]
79
80
81 ##### 2.1.5 O espectro no som amostrado
82 Lambda = 50.
83 T_i = n.random.random(Lambda)*2.-1.
84 C_k = n.fft.fft(T_i)
85 A_k = n.real(C_k)
86 B_K = n.imag(C_k)
87 w_k = 2.*n.pi*n.arange(Lambda)/Lambda
88
89
90 # 2.15 Recomposição do espectro no tempo
91 def t(i):
92     return (1./Lambda)*n.sum(C_k*n.e**(1j*w_k*i))
93
94
95 # 2.16 Recomposição real
96 def tR(i):
97     return (1./Lambda)*n.sum(n.abs(C_k)*n.cos(w_k*i-n.angle(C_k)))
98
99 # 2.17 Número de coeficientes espectrais pareados
100 tau = (Lambda - Lambda % 2)/2 + Lambda % 2-1
101 # 2.18 Coeficientes equivalentes
102 kk = n.arange(tau)
103 F_k = C_k[1:tau+1]
104 F2_k = C_k[Lambda-tau:Lambda-1][::-1]
105
106 # 2.19 Coeficientes equivalentes: módulos
107 ab = n.abs(F_k)
108 ab2 = n.abs(F2_k)
109 MIN = n.abs(ab-ab2).sum() # MIN ~ 0.0
110 # 2.20 Coeficientes equivalentes: fases
111 an = n.angle(F_k)
112 an2 = n.angle(F2_k)
113 MIN = n.abs(an+an2).sum() # MIN ~ 0.0
114
115 # 2.21 Combinação das componentes em cada amostra
116 w_k = 2*n.pi*n.arange(Lambda)/Lambda
117

```

```

118
119 def t_(i):
120     return (1./Lambda)*(A_k[0]+2.*n.sum(n.abs(C_k[1:tau+1]) *
121                                     n.cos(w_k*i-n.angle(C_k)) + A_k[Lambda/2] *
122                                     (1-Lambda % 2)))
123
124
125 ##### 2.1.6 A nota básica
126 f = 220.5 # Herz
127 Delta = 2.5 # segundos
128 Lambda = int(2.5*f_a)
129 ii = n.arange(Lambda)
130 Lf_i = Df_i # Já fizemos Df_i acima
131 # 2.24 Nota Básica
132 TfD_i = Lf_i[ii % len(Lf_i)]
133
134
135 ##### 2.1.7 Localização espacial
136 zeta = 0.215 # metros
137 # tomemos localizacao (x,y) qualquer
138 x = 1.5 # metros
139 y = 1. # metros
140 # 2.25 distâncias de cada ouvido
141 d = n.sqrt((x-zeta/2)**2+y**2)
142 d2 = n.sqrt((x+zeta/2)**2+y**2)
143 # 2.26 Distância de Tempo Interaural
144 DTI = (d2-d)/343.2 # segundos
145 # 2.27 Distância de Intensidade Interaural
146 DII = 20*n.log10(d/d2) # dBs
147
148 # 2.28 aplicação de DTI e DII em T_i
149 Lambda_DTI = int(DTI*f_a)
150 DII_a = d/d2
151 T_i = 1-n.abs(2-(4./lambda_f)*(ii % lambda_f)) # triangular
152 T2_i = n.hstack((n.zeros(Lambda_DTI), DII_a*T_i))
153 T_i = n.hstack((T_i, n.zeros(Lambda_DTI)))
154
155 som = n.vstack((T2_i, T_i)).T
156 a.wavwrite(som, "estereo.wav", f_a)
157 # espelhando
158 som = n.vstack((T_i, T2_i)).T
159 a.wavwrite(som, "estereo2.wav", f_a)
160
161 # 2.29 ângulo do objeto
162 theta = n.arctan(y/x)
163
164
165 ##### 2.1.8 Usos musicais

```

```

166 Delta = 3. # 3 segundos
167 Lambda = int(Delta*f_a)
168 f1 = 200. # Hz
169 foo = n.linspace(0., Delta*f1*2.*n.pi, Lambda, endpoint=False)
170 T1_i = n.sin(foo) # senoide de Delta segundos e freq = f1
171
172 f2 = 245. # Hz
173 lambda_f2 = int(f_a/f2)
174 T2_i = (n.arange(Lambda) % lambda_f2 < (lambda_f2/2))*2-1 # quadrada
175
176 f3 = 252. # Hz
177 lambda_f3 = f_a/f3
178 T3_i = n.arange(Lambda) % lambda_f3 # Dente de serra
179 T3_i = (T3_i/T3_i.max())*2-1
180
181 # 2.30 mixagem
182 T_i = T1_i+T2_i+T3_i
183 # normalização
184 T_i = ((T_i-T_i.min())/(T_i.max()-T_i.min()))*2-1
185 # escrita em disco
186 a.wavwrite(T_i, "mixados.wav", f_a)
187
188 # 2.31 concatenação
189 T_i = n.hstack((T1_i, T2_i, T3_i))
190 # escrita em disco
191 a.wavwrite(T_i, "concatenados.wav", f_a)

```

A.2 Código Python das relações descritas na seção 2.2

Todas as equações da seção 2.2 estão implementadas abaixo em Python e disponibilizadas como parte da toolbox MASSA.(2)

equações da seção 2.2 em Python

```

1  -*- coding: utf8 -*-
2  import numpy as n
3  import scikits.audiolab as a
4
5  f_a = 44100 # Hz, frequência de amostragem
6
7  ##### 2.2.1 Tabela de busca (LUT)
8  # tamanho da tabela: use par para não conflitar abaixo
9  # e ao menos 1024
10 Lambda_tilde = Lt = 1024
11
12 # Senoide
13 foo = n.linspace(0, 2*n.pi, Lt, endpoint=False)
14 S_i = n.sin(foo) # um período da senoide com T amostras
15
16 # Quadrada:
17 Q_i = n.hstack((n.ones(Lt/2)*-1, n.ones(Lt/2)))
18
19 # Triangular:
20 foo = n.linspace(-1, 1, Lt/2, endpoint=False)
21 Tr_i = n.hstack((foo, foo*-1))
22
23 # Dente de Serra:
24 D_i = n.linspace(-1, 1, Lt)
25
26 # som real, importar período e
27 # usar T correto: o número de amostras do período
28 Rf_i = a.wavread("22686__acclivity__oboe-a-440-periodo.wav")[0]
29
30 f = 110. # Hz
31 Delta = 3.4 # segundos
32 Lambda = int(Delta*f_a)
33
34 # Amostras:
35 ii = n.arange(Lambda)
36
37 # 2.32 LUT
38 Gamma_i = n.array(ii*f*Lt/f_a, dtype=n.int)
39 # Pode-se usar S_i, Q_i, D_i ou qualquer período de som real

```

```

40 # suficientemente grande
41 L_i = Tr_i
42 TfD_i = L_i[Gamma_i % Lt]
43
44
45 ##### 2.2.2 Variações incrementais de frequência e amplitude
46 # VARIAÇÕES DE FREQUÊNCIA
47 f_0 = 100. # freq inicial em Hz
48 f_f = 300. # freq final em Hz
49 Delta = 2.4 # duração
50
51 Lambda = int(f_a*Delta)
52 ii = n.arange(Lambda)
53 # 2.33 – variação linear
54 f_i = f_0+(f_f-f_0)*ii/(float(Lambda)-1)
55 # 2.34 coeficientes para a LUT
56 D_gamma_i = f_i*Lt/f_a
57 Gamma_i = n.cumsum(D_gamma_i)
58 Gamma_i = n.array(Gamma_i, dtype=n.int)
59 # 2.35 som resultante
60 Tf0ff_i = L_i[Gamma_i % Lt]
61
62 ## 2.36 – variação exponencial
63 f_i = f_0*(f_f/f_0)**(ii/(float(Lambda)-1))
64 # 2.37 coeficientes para a LUT
65 D_gamma_i = f_i*Lt/f_a
66 Gamma_i = n.cumsum(D_gamma_i)
67 Gamma_i = n.array(Gamma_i, dtype=n.int)
68 # 2.38 som resultante
69 Tf0ff_i = L_i[Gamma_i % Lt]
70
71
72 # VARIAÇÕES DE AMPLITUDE
73 # sintetizando um som qualquer para
74 # a variação de amplitude
75 f = 220. # Hz
76 Delta = 3.9 # segundos
77 Lambda = int(Delta*f_a)
78
79 # Amostras:
80 ii = n.arange(Lambda)
81
82 # (como em 2.30)
83 Gamma_i = n.array(ii*f*Lt/f_a, dtype=n.int)
84 L_i = Tr_i # pode-se usar igualmente S_i, Q_i, D_i ou
85 # qualquer período de som real suficientemente grande
86 T_i = TfD_i = L_i[Gamma_i % Lt]
87

```

```

88 a_0 = 1. # razão da amplitude em que é iniciada a sequência
89 a_f = 12. # razão da amplitude em que é finalizada
90 alpha = 1. # índice de suavidade da transição
91 # 2.39 envoltória exponencial para transição de amplitude
92 A_i = a_0*(a_f/a_0)**((ii/float(Lambda))**alpha)
93 # 2.40 aplicação da envoltória no som T_i
94 T2_i = A_i*T_i
95
96 # 2.41 envoltória linear de amplitude
97 A_i = a_0+(a_f-a_0)*(ii/float(Lambda))
98
99 # 2.42 transição exponencial de V_dB
100 V_dB = 31.
101 T2_i = T_i*((10*(V_dB/20.))**((ii/float(Lambda))**alpha))
102
103
104 ##### 2.2.3 Aplicação de filtros digitais
105 # VEJA iir.py para a geração da figura 2.17
106 # T_i herdado
107 # resposta ao impulso sintética (reverb)
108 H_i = (n.random.random(10)*2-1)*n.e**(-n.arange(10))
109
110 # 2.43 Convolução
111 T2_i = n.convolve(T_i, H_i)
112
113 # 2.44 veja linhas seguintes para aplicação da
114 # equação a diferenças :- )
115
116 fc = .1
117 # 2.45 passa baixas de polo simples
118 x = n.e**(-2*n.pi*fc) # fc = > freq de corte em 3dB
119 # coeficientes
120 a0 = 1-x
121 b1 = x
122 # aplicação do filtro
123 T2_i = [T_i[0]]
124 for t_i in T_i[1:]:
125     T2_i.append(t_i*a_0+T2_i[-1]*b1)
126
127 # 2.46 passa altas de polo simples
128 x = n.e**(-2*n.pi*fc) # fc = > freq de corte em 3dB
129 a0 = (1+x)/2
130 a1 = -(1+x)/2
131 b1 = x
132
133 # aplicação do filtro
134 T2_i = [a0*T_i[0]]
135 last = T_i[0]

```

```

136 for t_i in T_i[1:]:
137     T2_i += [a0*t_i + a1*last + b1*T2_i[-1]]
138     last = n.copy(t_i)
139
140
141 fc = .1
142 bw = .05
143 # 2.47 Variáveis auxiliares para os filtros nó
144 r = 1-3*bw
145 k = (1-2*r*n.cos(2*n.pi*fc)+r**2)/(2-2*n.cos(2*n.pi*fc))
146
147 # 2.48 passa banda
148 # coefs passa banda
149 a0 = 1-k
150 a1 = -2*(k-r)*n.cos(2*n.pi*fc)
151 a2 = r**2 - k
152 b1 = 2*r*n.cos(2*n.pi*fc)
153 b2 = -r**2
154
155 # aplicacao do filtro em T_i resultando em T2_i
156 T2_i = [a0*T_i[0]]
157 T2_i += [a0*T_i[1]+a1*T_i[0]+b1*T2_i[-1]]
158 last1 = T_i[1]
159 last2 = T_i[0]
160 for t_i in T_i[2:]:
161     T2_i += [a0*t_i+a1*last1+a2*last2+b1*T2_i[-1]+b2*T2_i[-2]]
162     last2 = n.copy(last1)
163     last1 = n.copy(t_i)
164
165 # 2.49 rejeita banda
166 # coeficientes
167 a0 = k
168 a1 = -2*k*n.cos(2*n.pi*fc)
169 a2 = k
170 b1 = 2*r*n.cos(2*n.pi*fc)
171 b2 = -r**2
172
173 # aplicacao do filtro em T_i resultando em T2_i
174 T2_i = [a0*T_i[0]]
175 T2_i += [a0*T_i[1]+a1*T_i[0]+b1*T2_i[-1]]
176 last1 = T_i[1]
177 last2 = T_i[0]
178 for t_i in T_i[2:]:
179     T2_i += [a0*t_i+a1*last1+a2*last2+b1*T2_i[-1]+b2*T2_i[-2]]
180     last2 = n.copy(last1)
181     last1 = n.copy(t_i)
182
183
184 ##### 2.2.4 Ruídos

```

```

185 # VEJA ruidos.py para o script que gerou a figura 2.18
186 Lambda = 100000 # Lambda sempre par
187 # diferença das frequências entre coeficientes vizinhos:
188 df = f_a/float(Lambda)
189
190 # 2.50 Ruído branco
191 # geração de espectro com módulo 1 uniforme
192 # e fase aleatória
193 coefs = n.exp(1j*n.random.uniform(0, 2*n.pi, Lambda))
194 # real par, imaginaria impar
195 coefs[Lambda/2+1:] = n.real(coefs[1:Lambda/2])[::-1] - 1j * \
196     n.imag(coefs[1:Lambda/2])[::-1]
197 coefs[0] = 0. # sem bias
198 coefs[Lambda/2] = 1. # freq max eh real simplesmente
199
200 # as frequências relativas a cada coeficiente
201 # acima de Lambda/2 nao vale
202 fi = n.arange(coefs.shape[0])*df
203 f0 = 15. # iniciamos o ruido em 15 Hz
204 i0 = n.floor(f0/df) # primeiro coef a valer
205 coefs[i0] = n.zeros(i0)
206 f0 = fi[i0]
207
208 # obtenção do ruído em suas amostras temporais
209 ruido = n.fft.ifft(coefs)
210 r = n.real(ruido)
211 r = ((r-r.min())/(r.max()-r.min()))*2-1
212 a.wavwrite(r, 'branco.wav', f_a)
213
214
215 # 2.51 Ruído rosa
216 # a cada oitava, perde-se 3dB
217 fator = 10.**(-3/20.)
218 alphai = fator**(n.log2(fi[i0:]/f0))
219
220 c = n.copy(coefs)
221 c[i0:] = coefs[i0:]*alphai
222 # real par, imaginaria impar
223 c[Lambda/2+1:] = n.real(c[1:Lambda/2])[::-1] - 1j * \
224     n.imag(c[1:Lambda/2])[::-1]
225
226 ruido = n.fft.ifft(c)
227 r = n.real(ruido)
228 r = ((r-r.min())/(r.max()-r.min()))*2-1
229 a.wavwrite(r, 'rosa.wav', f_a)
230
231
232 # 2.52 Ruído marrom
233 # a cada oitava, perde-se 6dB

```



```

234 fator = 10.**(-6/20.)
235 alphai = fator**(n.log2(fi[i0:]/f0))
236 c = n.copy(coefs)
237 c[i0:] = c[i0:]*alphai
238
239 # real par, imaginaria impar
240 c[Lambda/2+1:] = n.real(c[1:Lambda/2])[::-1] - 1j * \
241     n.imag(c[1:Lambda/2])[::-1]
242
243 # realizando amostras temporais do ruído marrom
244 ruido = n.fft.ifft(c)
245 r = n.real(ruido)
246 r = ((r-r.min())/(r.max()-r.min()))*2-1
247 a.wavwrite(r, 'marrom.wav', f_a)
248
249 ruido_marrom=n.copy(r) # será usado para a reverberação
250
251
252 # 2.53 Ruído azul
253 # para cada oitava, ganhamos 3dB
254 fator = 10.**(3/20.)
255 alphai = fator**(n.log2(fi[i0:]/f0))
256 c = n.copy(coefs)
257 c[i0:] = c[i0:]*alphai
258
259 # real par, imaginaria impar
260 c[Lambda/2+1:] = n.real(c[1:Lambda/2])[::-1] - 1j * \
261     n.imag(c[1:Lambda/2])[::-1]
262
263 # realizando amostras temporais do ruído azul
264 ruido = n.fft.ifft(c)
265 r = n.real(ruido)
266 r = ((r-r.min())/(r.max()-r.min()))*2-1
267 a.wavwrite(r, 'azul.wav', f_a)
268
269
270 # 2.54 Ruído violeta
271 # a cada oitava, ganhamos 6dB
272 fator = 10.**(6/20.)
273 alphai = fator**(n.log2(fi[i0:]/f0))
274 c = n.copy(coefs)
275 c[i0:] = c[i0:]*alphai
276
277 # real par, imaginaria impar
278 c[Lambda/2+1:] = n.real(c[1:Lambda/2])[::-1] - 1j * \
279     n.imag(c[1:Lambda/2])[::-1]
280
281 ruido = n.fft.ifft(c)
282 r = n.real(ruido)

```

```

283 r = ((r-r.min())/ (r.max()-r.min()))*2-1
284 a.wavwrite(r, 'violeta.wav', f_a)
285
286 # 2.55 Ruído preto
287 # a cada oitava, perdemos mais que 6dB
288 fator = 10.**(-12/20.)
289 alphai = fator**(n.log2(fi[i0:]/f0))
290 c = n.copy(coefs)
291 c[i0:] = c[i0:]*alphai
292
293 # real par, imaginaria impar
294 c[Lambda/2+1:] = n.real(c[1:Lambda/2])[::-1] - 1j * \
295     n.imag(c[1:Lambda/2])[::-1]
296
297 ruido = n.fft.ifft(c)
298 r = n.real(ruido)
299 r = ((r-r.min())/ (r.max()-r.min()))*2-1
300 a.wavwrite(r, 'preto.wav', f_a)
301
302
303 ##### 2.2.5 Tremolo e vibrato, AM e FM
304 # VEJA: vibrato.py e tremolo.py para as figuras 2.19 e 2.20
305 f = 220.
306 Lv = 2048 # tamanho da tabela do vibrato
307 fv = 1.5 # frequência do vibrato
308 nu = 1.6 # desvio maximo em semitons do vibrato (profundidade)
309 Delta = 5.2 # duração do som
310 Lambda = int(Delta*f_a)
311
312 # tabela do vibrato
313 x = n.linspace(0, 2*n.pi, Lv, endpoint=False)
314 tabv = n.sin(x) # o vibrato será senoidal
315
316 ii = n.arange(Lambda) # índices
317 # 2.56 índices da LUT para o vibrato
318 Gammav_i = n.array(ii*f_v*float(Lv)/f_a, n.int) # índices para a LUT
319 # 2.57 padrão de oscilação do vibrato para cada amostra
320 Tv_i = tabv[Gammav_i % Lv]
321 # 2.58 frequência em cada amostra
322 F_i = f*(2.** (Tv_i*nu/12.))
323 # 2.59 índices para LUT do som
324 D_gamma_i = F_i*(Lt/float(f_a)) # movimentação na tabela por amostra
325 Gamma_i = n.cumsum(D_gamma_i) # a movimentação na tabela total
326 Gamma_i = n.array(Gamma_i, dtype=n.int) # já os índices
327 # 2.60 som em si
328 T_i = Tr_i[Gamma_i % Lt] # busca dos índices na tabela
329
330 a.wavwrite(T_i, "vibrato.wav", f_a) # escrita do som

```

```

331
332
333 Tt_i = n.copy(Tv_i)
334 # 2.61 Envoltória do tremolo
335 V_dB = 12. # decibels envolvidos na variação
336 A_i = 10**((V_dB/20)*Tt_i)
337 # 2.62 Aplicação na sequência T_i
338 Gamma_i = n.array(ii*f*Lt/f_a, dtype=n.int)
339 T_i = Tr_i[Gamma_i % Lt]
340 T_i = T_i*A_i
341 a.wavwrite(T_i, "tremolo.wav", f_a) # escrita do som
342
343
344 # 2.63 – Espectro da FM, implementada em 2.66–70
345 # 2.64 – Função de Bessel, foge ao escopo
346 # 2.65 – Espectro da AM, implementada em 2.70,71 abaixo
347
348 fv = 60. # > 20Hz
349 # 2.66 índices para a LUT da moduladora da FM
350 Gammav_i = n.array(ii*f_v*float(Lv)/f_a, n.int)
351 # 2.67 padrão de oscilação da moduladora
352 Tfm_i = tabv[Gammav_i % Lv]
353 f = 330.
354 mu = 40.
355 # 2.68 Frequência em cada amostra na FM
356 f_i = f+Tfm_i*mu
357 # 2.69 índices da LUT para síntese do som
358 D_gamma_i = f_i*(Lt/float(f_a)) # movimentação na tabela por amostra
359 Gamma_i = n.cumsum(D_gamma_i) # a movimentação na tabela total
360 Gamma_i = n.array(Gamma_i, dtype=n.int) # já os índices
361 # 2.70 FM
362 T_i = S_i[Gamma_i % Lt] # busca dos índices na tabela
363
364 a.wavwrite(T_i, "fm.wav", f_a) # escrita do som
365
366
367 Tam_i = n.copy(Tfm_i)
368 V_dB = 12.
369 alpha = 10**((V_dB/20.)) # profundidade da AM
370 # 2.71 Envoltória para AM
371 A_i = 1+alpha*Tam_i
372 Gamma_i = n.array(ii*f*Lt/f_a, dtype=n.int)
373 # 2.70 AM
374 T_i = Tr_i[Gamma_i % Lt]*(A_i)
375 a.wavwrite(T_i, "am.wav", f_a) # escrita do som
376
377

```

```

378 ##### 2.2.5 Usos musicais
379 # 2.73 Veja peça Tremolos, Vibratos e a Frequência
380
381 # Efeito Doppler
382 v_r= 10 # receptor se move em direção à fonte com v_r m/s
383 v_s=-80. # emissor se move em direção ao receptor com -v_s m/s
384 v_som=343.2
385 f_0=1000 # frequência do emissor
386 # frequência com o efeito Doppler:
387 # 2.74 Frequência por efeito Doppler
388 f=((v_som + v_r) / (v_som + v_s)) * f_0
389 # a partir do cruzamento entre o emissor e o receptor:
390 f_=(((v_som - v_r) / (v_som - v_s)) * f_0
391
392 # distâncias iniciais:
393 x_0=0 # emissor à frente
394 y_0=200 # distante y_0 metros
395
396 Delta=5. # duração em segundos
397 Lambda=Delta*f_a # número de amostras
398 # posições ao longo do tempo, X_i=n.zeros(Lambda)
399 Y_i=y_0 - ((v_r-v_s)*Delta) * n.linspace(0,1,Lambda)
400
401 # A cada amostra, é preciso calcular a DTI e a DII com X_i e Y_i
402 # No caso, DTI e DII são == 0 pois a fonte está no meio.
403 # 2.75 Amplitude relativa ao efeito Doppler
404 # Assumindo z_0 metros acima da cabeça:
405 z_0=2.
406 D_i=( z_0**2+Y_i**2 )**0.5 # distância a cada amostra
407 # Amplitude relativa do som em cada amostra devido à distância:
408 A_i=z_0/D_i
409 # Alteração da amplitude devido ao efeito Doppler:
410 A_DP_i=( (v_r-v_s)/343.2+1 )**0.5
411 A_DP_i=( (-v_r+v_s)/343.2+1 )**0.5
412 A_DP_i=(Y_i>0)*A_DP_i+(Y_i<0)*A_DP_i
413 A_i=A_i * A_DP_i
414
415 # Os sinais das velocidades se invertem
416 # no caso da fonte passar o receptor.
417 # Portanto:
418 # 2.76 Progressão de frequência do efeito Doppler
419 coseno_i=(Y_i)/((Y_i**2+z_0**2)**0.5)
420 F_i=( ( 343.2+v_r*coseno_i ) / ( 343.2+v_s*coseno_i ) )*f_0
421 # coeficientes para a LUT
422 D_gamma_i = F_i*Lt/f_a
423 Gamma_i = n.cumsum(D_gamma_i)
424 Gamma_i = n.array(Gamma_i, dtype=n.int)
425
426 L_i = Tr_i # Onda triangular

```

```

427 # Som:
428 Tdoppler_i = L_i[Gamma_i % Lt]
429 Tdoppler_i*=A_i
430
431 # Normalizando e gravando:
432 Tdoppler_i=((Tdoppler_i-Tdoppler_i.min()) /i \
433            (Tdoppler_i.max()-Tdoppler_i.min()))*2.-1
434 a.wavwrite(Tdoppler_i, 'doppler.wav', f_a)
435
436
437 ##### Reverberação
438 # 0 primeiro período da reverberação:
439 Delta1 = 0.15 # tipicamente E [0.1,0.2]
440 Lambda1= int(Delta1*f_a)
441 Delta = 1.9 # duração total da reverberação
442 Lambda=int(Delta*f_a)
443
444 # Probabilidades de reincidência do som no primeiro período:
445 ii=n.arange(Lambda)
446 P_i = (ii[:Lambda1]/float(Lambda1))*2.
447 # incidências:
448 R1_i=n.random.random(Lambda1)<P_i
449 A_i=10.**((-50./20)*(ii/Lambda))
450 # 2.77 – Primeiro período da reverberação:
451 R1_i=R1_i*A_i[:Lambda1]*ruído_marrom[:Lambda1] # Primeiras incidências
452
453 # Ruído marrom em decaimento exponencial para o segundo período:
454 # -120dB até o final:
455 # 2.78 – Segundo período da reverberação:
456 Rm_i=ruído_marrom[Lambda1:Lambda]
457 R2_i=Rm_i*A_i[Lambda1:Lambda]
458 # 2.79 – Resposta ao impulso da reverberação
459 R_i=n.hstack((R1_i,R2_i))
460 R_i[0]=1. # resposta ao impulso está pronta
461
462 # Realização de um som para aplicar a reverberação:
463 f_0 = 100. # freq inicial em Hz
464 f_f = 700. # freq final em Hz
465 Delta = 2.4 # duração
466 Lambda = int(f_a*Delta)
467 ii = n.arange(Lambda)
468
469 # (usando 2.36 – variação exponencial)
470 f_i = f_0*(f_f/f_0)**(ii/(float(Lambda)-1))
471 # (usando 2.37 coeficientes para a LUT)
472 D_gamma_i = f_i*Lt/f_a
473 Gamma_i = n.cumsum(D_gamma_i)
474 Gamma_i = n.array(Gamma_i, dtype=n.int)
475 # (usando 2.38 som resultante)

```

```

476 Tf0ff_i = L_i[Gamma_i % Lt]
477
478 # Aplicação da reverberação
479 T_i=Tf0ff_i
480 T_i=n.convolve(T_i,R_i)
481 T_i=(T_i-T_i.min())/(T_i.max()-T_i.min())
482 a.wavwrite(T_i, 'reverb.wav', f_a)
483 a.wavwrite(R_i, 'RI_reverb.wav', f_a)
484
485
486 # 2.80 ADSR – variação Linear
487 Delta = 5. # duração total em segundos
488 Delta_A = 0.1 # Ataque
489 Delta_D = .3 # Decay
490 Delta_R = .2 # Release
491 a_S = .1 # nível de sustentação
492
493 Lambda = int(f_a*Delta)
494 Lambda_A = int(f_a*Delta_A)
495 Lambda_D = int(f_a*Delta_D)
496 Lambda_R = int(f_a*Delta_R)
497
498 # Realização da envoltória ADSR: A_i
499 ii = n.arange(Lambda_A, dtype=n.float)
500 A = ii/(Lambda_A-1)
501 A_i = A
502 ii = n.arange(Lambda_A, Lambda_D+Lambda_A, dtype=n.float)
503 D = 1-(1-a_S)*((ii-Lambda_A)/(Lambda_D-1))
504 A_i = n.hstack((A_i, D))
505 S = a_S*n.ones(Lambda-Lambda_R-(Lambda_A+Lambda_D), dtype=n.float)
506 A_i = n.hstack((A_i, S))
507 ii = n.arange(Lambda-Lambda_R, Lambda, dtype=n.float)
508 R = a_S-a_S*((ii-(Lambda-Lambda_R))/(Lambda_R-1))
509 A_i = n.hstack((A_i, R))
510
511 # 2.81 Realização do som com a envoltória
512 ii = n.arange(Lambda, dtype=n.float)
513 Gamma_i = n.array(ii*f*LT/f_a, dtype=n.int)
514 T_i = Tr_i[Gamma_i % Lt]*(A_i)
515
516 a.wavwrite(T_i, "adsr.wav", f_a) # escrita do som em disco
517
518
519 # 2.80 ADSR – variação Exponencial
520 xi = 1e-2 # -180dB para iniciar o fade in e finalizar o fade out
521 De = 2*100. # duracao total (\Delta)
522 DA = 2*20. # duracao do ataque \Delta_A
523 DD = 2*20. # duracao do decay \Delta_D
524 DR = 2*20. # duracao do release \Delta_R

```

```

525 SS = .4 # fração da amplitude em que ocorre o sustain
526
527 Lambda = int(f_a*De)
528 Lambda_A = int(f_a*DA)
529 Lambda_D = int(f_a*DD)
530 Lambda_R = int(f_a*DR)
531
532 A = xi*(1./xi)**(n.arange(Lambda_A)/(Lambda_A-1)) # amostras do ataque
533 A_i = n.copy(A)
534 # amostras do decay
535 D = a_S**((n.arange(Lambda_A, Lambda_A+Lambda_D)-Lambda_A)/(Lambda_D-1))
536 A_i = n.hstack((A_i, D))
537 S = a_S*n.ones(Lambda-Lambda_R-(Lambda_A+Lambda_D)) # amostras do sustain
538 A_i = n.hstack((A_i, S))
539 R = (SS)*(xi/SS)**((n.arange(Lambda-Lambda_R, Lambda)+Lambda_R-Lambda)/(Lambda_R-1))
540 # release
541 A_i = n.hstack((A_i, R))
542
543 # 2.81 Realização do som com a envoltória
544 ii = n.arange(Lambda, dtype=n.float)
545 Gamma_i = n.array(ii*f*Lt/f_a, dtype=n.int)
546 T_i = Tr_i[Gamma_i % Lt]*(A_i)
547
548 a.wavwrite(T_i, "adsr_exp.wav", f_a) # escrita do som em disco

```

A.3 Código Python das relações descritas na seção 2.3

Todas as equações da seção 2.3 estão implementadas abaixo em Python e disponibilizadas como parte da toolbox MASSA.(2)

equações da seção 2.3 em Python

```

1  #-*- coding: utf8 -*-
2  import numpy as n
3
4  f_a = 44100.  # Hz, frequência de amostragem
5  Lambda_tilde = Lt = 1024.
6  foo = n.linspace(0, 2*n.pi, Lt, endpoint=False)
7  S_i = n.sin(foo)  # um período da senóide com T amostras
8
9
10 def v(f=200, d=2., tab=S_i, fv=2., nu=2., tabv=S_i):
11     Lambda = n.floor(f_a*d)
12     ii = n.arange(Lambda)
13     Lv = float(len(S_i))
14
15     Gammav_i = n.floor(ii*fv*Lv/f_a)  # índices para a LUT
16     Gammav_i = n.array(Gammav_i, n.int)
17     # padrão de variação do vibrato para cada amostra
18     Tv_i = tabv[Gammav_i % int(Lv)]
19
20     # frequência em Hz em cada amostra
21     F_i = f*(2.**((Tv_i*nu/12.)))
22     # a movimentação na tabela por amostra
23     D_gamma_i = F_i*(Lt/float(f_a))
24     Gamma_i = n.cumsum(D_gamma_i)  # a movimentação na tabela total
25     Gamma_i = n.floor(Gamma_i)  # já os índices
26     Gamma_i = n.array(Gamma_i, dtype=n.int)  # já os índices
27     return tab[Gamma_i % int(Lt)]  # busca dos índices na tabela
28
29
30 ##### 2.3.1 Afinação, intervalos, escalas e acordes
31
32 # Microtonalidade de quartos de tom
33 # e sétimos de oitava
34 # com
35 epsilon = 2**(1/12.)
36 s1 = [0., 0.25, 1.75, 2., 2.25, 4., 5., 5.25]
37 #ou
38 epsilon = 2**(1/7.)
39 s2 = [0., 1., 2., 3., 4., 5., 6.]

```



```

40
41 # Tabela 2.22: Intervalos
42 I1j = 0.
43 I2m = 1.
44 I2M = 2.
45 I3m = 3.
46 I3M = 4.
47 I4J = 5.
48 ITR = 6.
49 I5J = 7.
50 I6m = 8.
51 I6M = 9.
52 I7m = 10.
53 I7M = 11.
54 I8J = 12.
55 I_i = n.arange(13.)
56
57
58 # o intervalo soma nove nomenclatura da inversão
59 # mas soma sempre 12 na inversão de semitons
60 def inv(I):
61     """retorna intervalo inverso de I: 0< = I < = 12"""
62     return 12-I
63
64
65 # intervalo harmônico
66 def intervaloHarmonico(f, I):
67     return (v(f)+v(f*2.**((I/12.)))*0.5
68
69
70 # intervalo melódico
71 def intervaloMelodico(f, I):
72     return n.hstack((v(f), v(f*2.**((I/12.))))
73
74 # 2.82 Escalas simétricas
75 Ec_i = [0., 1., 2., 3., 4., 5., 6., 7., 8., 9., 10., 11.]
76 Et_i = [0., 2., 4., 6., 8., 10.]
77 Etm_i = [0., 3., 6., 9.]
78 EtM_i = [0., 4., 8.]
79 Ett_i = [0., 6.]
80
81 # 2.83 Escalas diatônicas
82 Em_i = [0., 2., 3., 5., 7., 8., 10.]
83 Emlo_i = [1., 3., 5., 6., 8., 10.]
84 EM_i = [0., 2., 4., 5., 7., 9., 11.]
85 Emd_i = [0., 2., 3., 5., 7., 9., 10.]
86 Emf_i = [0., 1., 3., 5., 7., 8., 10.]
87 Eml_i = [0., 2., 4., 6., 7., 9., 11.]
88 Emmi_i = [0., 2., 4., 5., 7., 8., 10.]

```

```

89
90 # 2.84 Padrão diatônico
91 E_i_ = n.roll(n.array([2.,2.,1.,2.,2.,2.,1.]), n.random.randint(7.))
92 E_i = n.cumsum(E_i_)-E_i_[0.]
93
94
95 # 2.85 Escalas menores harmônica e melódica
96 Em_i = [0., 2., 3., 5., 7., 8., 10.]
97 Emh_i = [0., 2., 3., 5., 7., 8., 11.]
98 Emm_i = [0.,2.,3.,5.,7.,9.,11.,12.,10.,8.,7.,5.,3.,2.,0.]
99
100
101 # 2.86 Tríades
102 AM_i = [0., 4., 7.]
103 Am_i = [0., 3., 7.]
104 Ad_i = [0., 3., 6.]
105 Aa_i = [0., 4., 8.]
106
107 def comSetimam(A): return A+[10.]
108 def comSetimaM(A): return A+[11.]
109
110
111
112 ##### 2.3.2 Harmonias atonal, tonal, expansão e modulação
113 # Tabela 2.23
114 def relativa(TT):
115     """TT é tríade maior ou menor em posição fechada e fundamental"""
116     T = n.copy(TT)
117     if T[1]-T[0] == 4.: # ac maior
118         T[2] = 9. # retorna acorde menor abaixo
119     elif T[1]-T[0] == 3.: # ac menor
120         T[0] = 10. # retorna acorde maior acima
121     else:
122         print("send me only minor or major perfect triads")
123     return T
124
125
126 def antiRelativa(TT):
127     T = n.copy(TT)
128     if T[1]-T[0] == 4.: # maior
129         T[0] = 11. # retorna menor acima
130     if T[1]-T[0] == 3.: # menor
131         T[2] = 8. # retorna maior abaixo
132     return T
133
134
135 class Mediana:
136     def sup(self, TT):
137         T = n.copy(TT)

```

```

138         if T[1]-T[0] == 4.: # maior
139             T[0] = 11.
140             T[2] = 8. # retorna maior
141         if T[1]-T[0] == 3.: # menor
142             T[0] = 10.
143             T[2] -= 1. # retorna menor
144         return T
145
146     def inf(self, TT):
147         T = n.copy(TT)
148         if T[1]-T[0] == 4.: # maior
149             T[2] = 9
150             T[0] = 1. # retorna maior
151         if T[1]-T[0] == 3.: # menor
152             T[2] = 8.
153             T[0] = 11. # retorna menor
154         return T
155
156     def supD(self, TT):
157         """Preserva a quinta da primeira tríade na terça """
158         T = n.copy(TT)
159         if T[1]-T[0] == 4.: # maior
160             T[0] = 10.
161             T[1] = 3. # retorna maior
162         if T[1]-T[0] == 3.: # menor
163             T[0] = 11.
164             T[1] = 4. # retorna menor
165         return T
166
167     def infD(self, TT):
168         T = n.copy(TT)
169         if T[1]-T[0] == 4.: # maior
170             T[1] = 3.
171             T[2] = 8. # retorna maior
172         if T[1]-T[0] == 3.: # menor
173             T[1] = 4.
174             T[2] = 9. # retorna menor
175         return T
176
177     # Funções principais
178     tonicaM = [0., 4., 7.]
179     tonicam = [0., 3., 7.]
180     subM = [0., 5., 9.]
181     subm = [0., 5., 8.]
182     dom = [2., 7., 11.]
183     Vm = [2., 7., 10.] # quinto grau menor nao eh dominante
184
185
186     ##### 2.3.3 Contraponto
187     def contraNotaNotaSup(alturas=[0,2,4,5,5,0,2,0,2,2,2,0,7,\

```

```

188                                     5,4,4,4,0,2,4,5,5,5]):
189     """Realiza rotina de independência das vozes
190
191     Limitado em 1 oitava acima da nota"""
192     primeiraNota=alturas[0]+(7,12)[n.random.randint(2)]
193     contra=[primeiraNota]
194
195     i=0
196     cont=0 # contador de paralelas
197     reg=0 # registrador de intervalo em que se fez a paralela
198     for al in alturas[:-1]:
199         mov_cf=alturas[i:i+2]
200         atual_cf,seguinte_cf=mov_cf
201         if seguinte_cf-atual_cf>0:
202             mov="asc"
203         elif seguinte_cf-atual_cf<0:
204             mov="asc"
205         else:
206             mov="obl"
207
208         # possibilidades por consonancia
209         possiveis=[seguinte_cf+interval for interval in\
210                 [0,3,4,5,7,8,9,12]]
211
212         movs=[]
213         for pos in possiveis:
214             if pos -contra[i] < 0:
215                 movs.append("desc")
216             if pos - contra[i] > 0:
217                 movs.append("asc")
218             else:
219                 movs.append("obl")
220
221         movt=[]
222         for m in movs:
223             if 'obl' in (m,mov):
224                 movt.append("obl")
225             elif m==mov:
226                 movt.append("direto")
227             else:
228                 movt.append("contrario")
229         blacklist=[]
230         for nota,mt in zip(possiveis,movt):
231
232             if mt == "direto": # mov direto
233                 # n aceita intervalo perfeito
234                 if nota-seguinte_cf in (0,7,8,12):
235                     possiveis.remove(nota)
236
237         ok=0
238         while not ok:
239             nnota=possiveis[n.random.randint(len(possiveis))]

```

```

238         if nnota-seguinte_cf==contra[i]-atual_cf: # paralelo
239             intervalo=contra[i]-atual_cf
240             novo_intervalo=nnota-seguinte_cf
241             if abs(intervalo-novo_intervalo)==1: # do mesmo tipo 3 ou 6
242                 if cont==2: # se já teve 2 paralelas
243                     pass # outro intervalo
244                 else:
245                     cont+=1
246                     ok=1
247             else: # mov obl ou contrario
248                 cont=0 # zera paralelos
249                 ok=1
250             contra.append(nnota)
251             i+=1
252     return contra

```

255 **##### 2.3.4 Ritmo**

256 **# Veja peça Poli Hit Mia no Apêndice B**

259 **##### 2.3.5 Repetição e variação: motivos e unidades maiores**

260 **# Conceitos ubíquos**

263 **##### 2.3.6 Estruturas direcionais**

264 **# Veja peça Dirracional no Apêndice B**

267 **##### 2.3.7 Estruturas cíclicas**

268 **# Veja as peças 3 Trios no Apêndice B**

269 **# e o PPEPPS no Apêndice C**

APÊNDICE B – Código Computacional das Peças Musicais

Todas as peças a seguir foram feitas para exemplificar as relações apresentadas no capítulo 2 e são disponibilizadas online junto ao *toolbox* MASSA.(2)

B.1 Peças referentes à seção 2.1

B.1.1 Quadros sonoros

A montagem musical ‘quadros sonoros’ é dedicada para demonstração da mixagem pela soma direta de sequências amostrais (subseção 2.1.8). Esta rotina sintetiza 5 pequenas peças de sonoridades estáticas. Peça demonstrativa dos conceitos apresentados na seção 2.1. Houve relatos em listas de *emails* de que estas sonoridades causaram alteração do estado de consciência. Em especial, o quadro 5 causou em diferentes pessoas o mesmo efeito: a sensação de amplificação e distorção dos sons da mandimbula.

Quadros sonoros 1-5

```

1  #-*- coding: utf8 -*-
2  import numpy as n
3  import scikits.audiolab as a
4
5  # Peça musical baseada em sons estáticos
6  # são mixagens de sons básicos apresentados na seção 2.1
7
8  f_a = 44100 # 44.1kHz, frequência de amostragem de CDs
9  Delta = 180. # cada quadro terá Delta segundos
10 Lambda = int(Delta*f_a) # número de amostras
11
12 ii = n.linspace(0, Delta*2*n.pi, Lambda, endpoint=False)
13
14 # frequências que dividem f_a
15 fs = []
16 for i in xrange(1, f_a/2+1):
17     if f_a/float(i) == int(f_a/i):
18         fs.append(i)
19
20 # Quadro 1: senóides no grave em batimento e
21 # no agudo uma dente de serra bem suave
22 f1 = 100
23 f2 = 100.5
24
25 som = n.sin(ii*f1)+n.sin(ii*f2)
26 dente_aguda = (n.arange(Lambda) % 4-2)
27 som += dente_aguda/80
28
29 # normalização no intervalo [-1,1]

```



```

30 som = ((som - som.min())/(som.max()-som.min()))*2-1
31
32 a.wavwrite(som, "quadro1.wav", f_a)
33 print(u"quadro 1 feito em quadro1.wav (mono), são %i amostras \
34 em uma frequência de amostragem de %iHz" % (len(som), f_a))
35
36
37 # Quadro 2: 3 conjuntos separados de triangulares
38 fs2 = fs[20:21]+fs[65:70]+fs[77:]
39 som = n.zeros(Lambda)
40 ii = n.arange(Lambda)
41 for f in fs2:
42     lambda_f = f_a/f
43     som += (1-n.abs(2-(4./lambda_f)*(ii % lambda_f)))*(1./f**1.2)
44
45 # normalizando no intervalo [-1,1]
46 som = ((som - som.min())/(som.max()-som.min()))*2-1
47
48 a.wavwrite(som, "quadro2.wav", f_a)
49 print("quadro 2 feito em quadro2.wav (mono), são %i amostras \
50 em uma frequência de amostragem de %iHz" % (len(som), f_a))
51
52
53 # Quadro 3: estereofonia alternada no espectro harmônico
54 f = 50.
55 fs3 = [f*i for i in xrange(1, 7)] # 6 harmônicos
56 som_d = n.zeros(Lambda)
57 som_e = n.zeros(Lambda)
58 ii = n.linspace(0, Delta*2*n.pi, Lambda, endpoint=False)
59 i = 0
60 for f in fs3:
61     if i % 2 == 0:
62         som_d += n.sin(f*ii)*(1./f)
63     else:
64         som_e += n.sin(f*ii)*(1./f)
65     i += 1
66 som = n.vstack((som_d, som_e)).T
67 som = ((som - som.min())/(som.max()-som.min()))*2-1
68
69 a.wavwrite(som, "quadro3.wav", f_a)
70 print("quadro 3 feito em quadro3.wav (estéreo), são %i amostras \
71 em uma frequência de amostragem de %iHz" % (len(som), f_a))
72
73
74 # Quadro 4: batimentos intercalados por ouvido e com defasagens
75 fs4 = n.array([50, 51.01, 52.01, 53])
76 som_d = n.zeros(Lambda)
77 som_e = n.zeros(Lambda)
78 ii = n.linspace(0, Delta*2*n.pi, Lambda, endpoint=False)

```

```

79 i = 0
80 for f in fs4:
81     if i % 2 == 0:
82         som_d += n.sin(f*ii)*(1./f)
83     else:
84         som_e += n.sin(f*ii)*(1./f)
85     i += 1
86 som = n.vstack((som_d, som_e)).T
87
88 fs4 = n.array([500, 501.01, 502.01, 503])
89 som_d = n.zeros(Lambda)
90 som_e = n.zeros(Lambda)
91 ii = n.linspace(0, Delta*2*n.pi, Lambda, endpoint=False)
92 i = 0
93 for f in fs4:
94     if i % 2 == 0:
95         som_d += n.sin(f*ii)*(1./f)
96     else:
97         som_e += n.sin(f*ii)*(1./f)
98     i += 1
99 som += n.vstack((som_d, som_e)).T/60
100 som = ((som - som.min())/(som.max()-som.min()))*2-1
101
102 a.wavwrite(som, "quadro4.wav", f_a)
103 print("quadro 4 feito em quadro4.wav (estéreo), são %i amostras \
104 em uma frequência de amostragem de %iHz" % (len(som), f_a))
105
106
107 # Quadro 5: Dente de serra grave bate com harmônico em cada lado
108 f = 42. # Hz, freq da dente
109 lambda_f = 44100/f
110 dente = ((n.arange(float(Lambda)) % lambda_f)/lambda_f)*2-1
111
112 ii = n.linspace(0, Delta*2*n.pi, Lambda, endpoint=False)
113 som_d = dente+n.sin(ii*43)+n.sin(ii*84)
114 som_e = dente+n.sin(ii*43) + n.sin(ii*126.3)
115
116 som = n.vstack((som_d, som_e)).T
117 som = ((som - som.min())/(som.max()-som.min()))*2-1
118
119 a.wavwrite(som, "quadro5.wav", f_a)
120 print("quadro 5 feito em quadro5.wav (estéreo), são %i amostras \
121 em uma frequência de amostragem de %iHz" % (len(som), f_a))

```

B.1.2 Reduced-fi

Pequena peça musical para a concatenação de sequências amostrais como notas musicais (subseção 2.1.8). Sintetiza uma pequena peça de 25 segundos em Python puro, i.e. sem utilizar bibliotecas externas como *Numpy* ou *Scikits/Audiolab*. Peça demonstrativa dos conceitos apresentados na seção 2.1 e disponibilizada online junto ao *toolbox* MASSA.(2)

reduced-fi

```

1  #!/usr/bin/python
2
3  # Rode com:
4  # ./reduced-fi.py
5  # ou
6  # python reduced-fi.py
7
8  import math as m
9  import random as r
10
11 def p(freq):
12     return 2*m.pi*freq/44100
13
14 def d(dur):
15     return int(dur*44100)
16
17 def s(freq,dur):
18     return [m.sin(p(freq)*i) for i in xrange(d(dur))]
19
20 ss=(s(100,.1)+s(200,.1)+s(400,.1)+s(800,.1))*4
21 ss+=(s(100,.1)+s(200,.1)+s(400,.1)+s(800,.1))*4
22 ss+=(s(200,.05)+s(100,.05)+s(3200,.05)+s(800,.05))*8
23 ss+=(s(200,.05)+s(100,.05)+s(3200,.05)+s(800,.05))*8
24
25 ss+=(s(100,.01)+s(200,.01)+s(400,.01)+s(800,.01))*4
26 ss+=(s(100,.01)+s(200,.01)+s(400,.01)+s(800,.01))*4
27 ss+=(s(200,.005)+s(100,.005)+s(3200,.005)+s(800,.005))*8
28 ss+=(s(200,.005)+s(100,.005)+s(3200,.005)+s(800,.005))*8
29 ss+=s(300,2)
30
31 ss+=(s(100,.01)+s(200,.01)+s(400,.01)+s(800,.01))*4
32 ss+=(s(100,.01)+s(200,.01)+s(4000,.01)+s(800,.01))*4
33 ss+=(s(200,.005)+s(100,.005)+s(6400,.005)+s(800,.005))*8
34 ss+=(s(200,.005)+s(100,.005)+s(3200,.005)+s(800,.005))*8
35 ss+=s(300,2)
36

```

```

37  foo=ss[:int(44100*1)]; r.shuffle(foo); ss+=[i*.1 for i in foo]
38  ss+=s(300,2)
39
40
41  foo=ss[:int(44100*.5)]; r.shuffle(foo); ss+=[i*.1 for i in foo]
42  ss+=(s(100,.01)+s(200,.01)+s(400,.01)+s(800,.01))*4
43  ss+=(s(100,.01)+s(200,.01)+s(400,.01)+s(800,.01))*4
44  ss+=(s(200,.005)+s(100,.005)+s(3200,.005)+s(800,.005))*8
45  ss+=(s(200,.005)+s(100,.005)+s(3200,.005)+s(800,.005))*8
46
47  ss+=(s(100,.01)+s(200,.01)+s(400,.01)+s(800,.01))*4
48  ss+=(s(100,.01)+s(200,.01)+s(4000,.01)+s(800,.01))*4
49  ss+=(s(200,.005)+s(100,.005)+s(6400,.005)+s(800,.005))*8
50  ss+=(s(2000,.005)+s(100,.005)+s(10000,.005)+s(800,.005))*8
51  ss+=s(300,2)
52
53  ss+=(s(50,.1)+s(200,.1)+s(400,.1)+s(800,.1))*1
54  ss+=(s(50,1)+s(200,.1)+s(400,.1)+s(800,.1))*1
55  ss+=(s(100,.5)+s(100,.05)+s(3200,.05)+s(800,.05))*1
56  ss+=(s(200,.05)+s(100,.05)+s(3200,.05)+s(400,.5))*1
57  foo=ss[:int(44100*.25)]; r.shuffle(foo); ss+=[i*.1 for i in foo]
58
59  ss+=s(400,.2)
60  ss+=s(200,.8)
61  ss+=s(150,.5)
62  ss+=s(100,1.5)
63
64
65  #####
66  seq=ss
67  ## Gravando em disco
68  import wave, struct
69  smin=min(seq); smax=max(seq)
70  seq= [(-.5+(i-smin)/(smax-smin)) for i in seq] # normalizando
71  sound = wave.open('musica.wav','w')
72  sound.setframerate(44100)
73  sound.setsampwidth(2) # sempre 16bit/sample (2 bytes)
74  sound.setnchannels(1) # mono
75  sonic_vector=[i*(2**15-1) for i in seq] # 16 bit com sinal
76  sound.writeframes(struct.pack('h'*len(sonic_vector),\
77                               *[int(i) for i in sonic_vector]))
78  sound.close()
79  print "arquivo musica.wav gravado"

```

B.2 Peças referentes à seção 2.2

B.2.1 Transita para metro

Sintetiza em 1 pequena peça de 49 segundos com varreduras de frequência, chamados *chirps*. Também é proposta para explorações de transições de intensidade. Peça demonstrativa dos conceitos apresentados na subseção 2.2.2 e disponibilizada junto à MASSA.(2)

Transita para metro

```

1  #-- coding: utf8 --
2  import numpy as n, scikits.audiolab as aa
3
4  # peça dedicada e expor as diferentes transições
5  # de intensidade e altura
6
7  # partes
8  # 1) Transições de altura log e lin e com alpha
9  # 2) Transições de intensidade log e in e com alpha
10 # 3) Usos combinados de ambos
11
12 f_a = 44100. # Hz, frequência de amostragem
13 Lambda_tilde=Lt=1024 # tamanho da tabela para LUT
14
15 # Senoide
16 foo=n.linspace(0,2*n.pi,Lt,endpoint=False)
17 S_i=n.sin(foo) # um período da senóide com T amostras
18
19 # Quadrada:
20 Q_i=n.hstack( ( n.ones(Lt/2)*-1 , n.ones(Lt/2) ) )
21
22 # Siangular:
23 foo=n.linspace(-1,1,Lt/2,endpoint=False)
24 Tr_i=n.hstack( ( foo , foo*-1 ) )
25
26 # Dente de Serra:
27 D_i=n.linspace(-1,1,Lt)
28
29 def T(f1,f2,dur,ttype="exp",tab=S_i,alpha=1.):
30     Lambda=n.floor(dur*f_a)
31     ii=n.arange(Lambda)
32
33     if ttype=="exp":
34         f_i=f1*(f2/f1)**( (ii/(float(Lambda)-1))*alpha ) # exponencial
35     else:

```

```

36         f_i=f1+(f2-f1)*ii/(Lambda-1) # linear
37
38     Lt=len(tab)
39     D_gamma_i=f_i*Lt/f_a
40     Gamma_i=n.cumsum(D_gamma_i)
41     Gamma_i=n.array(Gamma_i,dtype=n.int)
42     return tab[Gamma_i%Lt]
43
44     #####
45     # PARTE 1)
46
47     # 1a: comparando variação linear e logarítmica
48     #class
49     """parametros pré-estabelecidos para maior coerência da exploração"""
50
51     f=[50.,100.,150.,200.,250.,300.,350.,400.,450.,500.,550., 600.]
52     # 01 11 25 31 43 55 67 71 82 83 94# 105
53     d=[0.1,0.2,0.3,0.5,1.0,1.5,2.0,3.0,5.0,7.0,10.0]
54     # 0 1 2 3 4 5 6 7 8 9 10
55     a=[0.01,0.1,1.0,10.,100.]
56     # 0 1 2 3 4
57
58
59     #intro (8 segundos)
60     intr=n.hstack((T(f[9],f[0],d[7]),T(f[0],f[10],d[8], 'lin',Tr_i)))
61
62     #entrada (8 segundos)
63     entr= n.hstack(( T(f[10],f[10],d[8], 'lin',Tr_i),n.zeros(f_a*d[7]))+ \
64         n.hstack(( T(f[3],f[0],d[3]),
65             T(f[1],f[0],d[3]),T(f[1],f[0],d[3]),T(f[1],f[0],d[3]),
66             T(f[2],f[0],d[3]),
67             T(f[1],f[0],d[3]),T(f[1],f[0],d[3]),T(f[1],f[0],d[3]),
68             T(f[3],f[0],d[3]),
69             T(f[1],f[0],d[3]),T(f[1],f[0],d[3]),T(f[1],f[0],d[3]),
70             T(f[5],f[0],d[3]),
71             T(f[3],f[0],d[3]),T(f[2],f[0],d[3]),T(f[1],f[0],d[3], 'lin') ))
72
73     #desenvolvimento da entrada (8 segundos)
74     devEntr=n.hstack(( T(f[0],f[0],d[8]),T(f[0],f[0],d[4]),
75         T(f[0],f[0],d[3], 'lin',Tr_i),
76         T(f[0],f[0],d[3], 'lin',Tr_i),T(f[0],f[0],d[3], 'lin',Q_i),\
77         T(f[0],f[0],d[3], 'lin',Tr_i) )) + n.hstack(( T(f[5],f[2],d[3]),
78         T(f[5],f[8],d[3]),T(f[5],f[2],d[3]),T(f[8],f[5],d[3]),
79         T(f[5],f[2],d[3]),
80         T(f[5],f[1],d[3]),T(f[5],f[0],d[3]),T(f[1],f[2],d[3]),
81         T(f[5],f[2],d[3]),
82         T(f[10],f[2],d[3]),T(f[10],f[0],d[3]),T(f[10],f[9],d[3]),
83         T(f[5],f[2],d[3]),
84         T(f[7],f[5],d[3]),T(f[7],f[2],d[3]),T(f[0],f[2],d[3]) ))
85

```

```

86 #sublimada 1 / passagem para outros funcionamentos e estereofonia 8s
87 sub_e=n.hstack(( T(f[1],f[0],d[3], 'log',Q_i),
88     T(f[1],f[0],d[3]),T(f[1],f[0],d[3], 'lin'),T(f[1],f[0],d[3]),
89     T(f[1],f[0],d[3]),
90     T(f[1],f[0],d[3]),T(f[1],f[0],d[3], 'lin'),T(f[1],f[0],d[3]),
91     T(f[1],f[0],d[3], 'log',Q_i),
92     T(f[1],f[0],d[3]),T(f[1],f[0],d[3], 'lin'),T(f[1],f[0],d[3]),
93     T(f[1],f[0],d[3]),
94     T(f[1],f[0],d[3]),T(f[1],f[0],d[3], 'lin'),T(f[1],f[0],d[3]) )) + \
95     n.hstack(( T(f[0],f[10],d[9])+\\
96     T(f[0],f[10],d[9], 'lin',Tr_i),n.zeros(f_a*d[4]) ))
97
98 sub_d=n.hstack(( T(f[1],f[0],d[3], 'log',Q_i),
99     T(f[1],f[0],d[3]),T(f[1],f[0],d[3], 'lin'),T(f[1],f[0],d[3]),
100     T(f[1],f[0],d[3]),
101     T(f[1],f[0],d[3]),T(f[1],f[0],d[3], 'lin'),T(f[1],f[0],d[3]),
102     T(f[1],f[0],d[3], 'log',Q_i),
103     T(f[1],f[0],d[3]),T(f[1],f[0],d[3], 'lin'),T(f[1],f[0],d[3]),
104     T(f[1],f[0],d[3]),
105     T(f[1],f[0],d[3]),T(f[1],f[0],d[3], 'lin'),T(f[1],f[0],d[3]) )) + \
106     n.hstack(( T(f[0],f[10],d[9], 'log',Tr_i)+\\
107     T(f[0],f[10],d[9], 'lin'),n.zeros(f_a*d[4]) ))
108
109 # dev da sublimada +8s
110 sub2_e=n.hstack(( T(f[10],f[9],d[3]),
111     T(f[10],f[7],d[3]),T(f[10],f[9],d[3]),T(f[10],f[6],d[3]),
112     T(f[10],f[9],d[3]),
113     T(f[10],f[7],d[3]),T(f[10],f[9],d[3]),T(f[10],f[6],d[3]), )) + \
114     n.hstack(( T(f[10],f[8],d[3],tab=Tr_i),
115     T(f[10],f[5],d[3],tab=Tr_i),T(f[10],f[4],d[3],tab=Tr_i),
116     T(f[10],f[6],d[3],tab=Tr_i),
117     T(f[10],f[10],d[3],tab=Tr_i),
118     T(f[10],f[5],d[3],tab=Tr_i),T(f[10],f[0],d[3],tab=Tr_i),
119     T(f[10],f[1],d[3],tab=Tr_i) ))
120
121
122 sub2_d=n.hstack(( T(f[10],f[9],d[3],tab=Tr_i),
123     T(f[10],f[7],d[3],tab=Tr_i),T(f[10],f[9],d[3],tab=Tr_i),
124     T(f[10],f[6],d[3],tab=Tr_i),
125     T(f[10],f[9],d[3],tab=Tr_i),
126     T(f[10],f[7],d[3],tab=Tr_i),T(f[10],f[9],d[3],tab=Tr_i),
127     T(f[10],f[6],d[3],tab=Tr_i) )) + n.hstack(( T(f[10],f[8],d[3]),
128     T(f[10],f[5],d[3]),T(f[10],f[4],d[3]),T(f[10],f[6],d[3]),
129     T(f[10],f[10],d[3]),
130     T(f[10],f[5],d[3]),T(f[10],f[0],d[3]),T(f[10],f[1],d[3]) ))
131
132 # rarefação e repetições preparando para término
133 rar_e=n.hstack(( T(f[7],f[9],d[3]),
134     n.zeros(d[3]), T(f[10],f[9],d[3]),T(f[0],f[9],d[3]),
135     n.zeros(d[3]),T(f[0],f[9],d[5]),

```

```

136     n.zeros(d[6]),
137     T(f[1],f[9],d[6], 'lin')    ))
138
139 rar_d=n.hstack(( T(f[7],f[9],d[3], 'lin'),
140     n.zeros(d[3]), n.zeros(d[3]),T(f[1],f[9],d[3]),
141     T(f[0],f[10],d[4], 'lin',Tr_i),
142     T(f[9],f[9],d[4]),
143     T(f[1],f[9],d[6], 'log')    ))
144
145
146 # finalização com elementos dos outros momentos
147 intr=n.hstack((T(f[9],f[0],d[7]),T(f[0],f[10],d[8], 'lin',Tr_i)))
148
149 fin_e=n.copy(intr)
150 fin_d=n.copy(intr)
151 fin_e[d[4]*f_a:d[5]*f_a]=n.zeros(d[3]*f_a)
152 fin_d[d[5]*f_a:d[6]*f_a]=n.zeros(d[3]*f_a)
153
154 fin_e[7*f_a:7.5*f_a]=n.zeros(d[3]*f_a)
155 fin_d[7.25*f_a:7.75*f_a]=n.zeros(d[3]*f_a)
156
157 _e=n.hstack(( intr,entr,devEntr,sub_e,sub2_e,rar_e,fin_e    ))
158 _d=n.hstack(( intr,entr,devEntr,sub_d,sub2_d,rar_d,fin_d,n.zeros(2)    ))
159 s=n.vstack((_e,_d)).T
160
161 s=((s-s.min())/(s.max()-s.min()))*2-1
162
163 aa.wavwrite(s, 'trans1.wav', f_a)
164
165
166 # Exploracao sistemática dos tremolos:
167 #first=n.array(())
168 #for d in d[:5]:
169 #    prinT(d)
170 #    for f in f:
171 #        for f2 in f:
172 #            first=n.hstack((first,
173 #                trans(f,f2,d),
174 #                trans(f,f2,d, 'lin'),
175 #                trans(f,f2,d,tab=Tr_i),
176 #                trans(f,f2,d, 'lin',tab=Tr_i),
177 #                trans(f,f2,d,tab=D_i),
178 #                trans(f,f2,d, 'lin',tab=D_i),
179 #                (trans(f,f2,d)+trans(f,f2,d, 'lin'))*.5,
180 #                (trans(f,f2,d,tab=Tr_i)+trans(f,f2,d, 'lin',tab=Tr_i))*0.5,
181 #                (trans(f,f2,d,tab=Q_i)+trans(f,f2,d, 'lin',tab=Q_i))*0.5    ))

```


B.2.2 Vibra e treme

Para explorações de tremolos e vibratos como expostos na subseção 2.2.5, sintetiza 17 pequenas montagens de 8 a 24 segundos cada. Script disponibilizado online junto ao toolbox MASSA.(2)

B.2.3 Tremolos, vibratos e a frequência

Pequena montagem musical para demonstração de vínculo entre os parâmetros dos tremolos e vibratos com a frequência fundamental central da nota. Resulta em 19 pequenas montagens de 4-32 segundos. Peça demonstrativa dos conceitos apresentados na subseção 2.2.6 e disponibilizada online junto ao toolbox MASSA.(2)

B.2.4 Trenzinho de caipiras impulsivos

Pequena peça musical para demonstração do deslocamento causado pela convolução com o impulso. Sintetiza 11 pequenas montagens cada uma com duração de 4-240 segundos. Peça demonstrativa dos conceitos apresentados na subseção 2.2.3 e disponibilizada online junto ao toolbox MASSA.

B.2.5 Ruidosa faixa

Pequena peça musical de 240 segundo para demonstração de filtragens diversas em ruídos e para reverberação, conceitos apresentados na subseção 2.2.3 e disponibilizada online junto ao toolbox MASSA.(2)

ruidosa faixa

```

1 #-- coding: utf8 --
2 import numpy as n, scikits.audiolab as a
3 H=n.hstack
4 V=n.vstack
5

```

```

6  f_a = 44100. # Hz, frequência de amostragem
7
8  ##### 2.2.1 Tabela de busca (LUT)
9  Lambda_tilde=Lt=1024.
10
11 # Senoide
12 foo=n.linspace(0,2*n.pi,Lt,endpoint=False)
13 S_i=n.sin(foo) # um período da senóide com T amostras
14
15 # Quadrada:
16 Q_i=n.hstack( ( n.ones(Lt/2)*-1 , n.ones(Lt/2) ) )
17
18 # Triangular:
19 foo=n.linspace(-1,1,Lt/2,endpoint=False)
20 Tr_i=n.hstack( ( foo , foo*-1 ) )
21
22 # Dente de Serra:
23 D_i=n.linspace(-1,1,Lt)
24
25 def v(f=200,d=2.,tab=S_i,fv=2.,nu=2.,tabv=S_i):
26     Lambda=n.floor(f_a*d)
27     ii=n.arange(Lambda)
28     Lv=float(len(S_i))
29
30     Gammav_i=n.floor(ii*fv*Lv/f_a) # índices para a LUT
31     Gammav_i=n.array(Gammav_i,n.int)
32     # padrão de variação do vibrato para cada amostra
33     Tv_i=tabv[Gammav_i%int(Lv)]
34
35     # frequência em Hz em cada amostra
36     F_i=f*( 2.** ( Tv_i*nu/12. ) )
37     # a movimentação na tabela por amostra
38     D_gamma_i=F_i*(Lt/float(f_a))
39     Gamma_i=n.cumsum(D_gamma_i) # a movimentação na tabela total
40     Gamma_i=n.floor( Gamma_i ) # já os índices
41     Gamma_i=n.array( Gamma_i, dtype=n.int) # já os índices
42     return tab[Gamma_i%int(Lt)] # busca dos índices na tabela
43
44 def A(fa=2.,V_dB=10.,d=2.,taba=S_i):
45     Lambda=n.floor(f_a*d)
46     ii=n.arange(Lambda)
47     Lt=float(len(tab))
48     Gammaa_i=n.floor(ii*fa*Lt/f_a) # índices para a LUT
49     Gammaa_i=n.array(Gammaa_i,n.int)
50     # variação da amplitude em cada amostra
51     A_i=taba[Gammaa_i%int(Lt)]
52     A_i=A_i*10.** (V_dB/20.)
53     return A_i
54
55

```

```

56 def adsr(som,A=10.,D=20.,S=-20.,R=100.,xi=1e-2):
57     a_S=10**(S/20.)
58     Lambda=len(som)
59     Lambda_A=int(A*f_a*0.001)
60     Lambda_D=int(D*f_a*0.001)
61     Lambda_R=int(R*f_a*0.001)
62
63     ii=n.arange(Lambda_A,dtype=n.float)
64     A=ii/(Lambda_A-1)
65     A_i=A
66     ii=n.arange(Lambda_A,Lambda_D+Lambda_A,dtype=n.float)
67     D=1-(1-a_S)*((ii-Lambda_A)/(Lambda_D-1))
68     A_i=n.hstack((A_i, D))
69     S=n.ones(Lambda-Lambda_R-(Lambda_A+Lambda_D),dtype=n.float)*a_S
70     A_i=n.hstack((A_i, S))
71     ii=n.arange(Lambda-Lambda_R,Lambda,dtype=n.float)
72     R=a_S-a_S*((ii-(Lambda-Lambda_R))/(Lambda_R-1))
73     A_i=n.hstack((A_i,R))
74
75     return som*A_i
76
77
78 BPM=60. # 80 batidas por minuto
79 DELTA=BPM/60 # duração da batida
80 LAMBDA=DELTA*f_a # número de samples da batida
81 LAMBDA_=int(LAMBDA) # inteiro para operação com índices
82
83 #cabeca=[1]+[0]*(LAMBDA-1)
84 #contra=[0]*Lambda/2+[1]+[0]*(Lambda/2-1)
85 tempo=n.zeros(LAMBDA)
86 cabeca=n.copy(tempo); cabeca[0]=1.
87 contra=n.copy(tempo); contra[LAMBDA_/2]=1.
88
89
90 # tempo de musica
91 Delta=4*DELTA # segundos
92 Lambda=Delta*f_a
93 Lambda_=int(Lambda)
94 ii=n.arange(Lambda_)
95 linha_cabeca=cabeca[ii%LAMBDA_]
96 linha_contra=contra[ii%LAMBDA_]
97
98 som1=adsr(v(tabv=Tr_i ,d=.3,fv=3.,nu=7.0,f=300.),10,10,-10.)
99 som2=adsr(v(tabv=Tr_i ,d=.2,fv=2.,nu=1.),10,10,-10.)
100 som3=adsr(v(tabv=Tr_i ,d=.2,fv=10.,nu=7.),10,10,-10.)
101 som4=adsr(v(tabv=Tr_i ,d=.2,fv=3.,nu=7.,f=1800.),1.,100.,-60.,80.)
102 som5=adsr(v(tabv=Tr_i ,d=.2,fv=3.,nu=7.,f=1800.)*A(d=.2,fa=100.),
103           1.,100.,-60.,80.)
104 som6=adsr(v(tabv=Tr_i ,d=.2,fv=30.,nu=7.,f=1800.)*A(d=.2),
105           1.,100.,-60.,80.)

```

```

106
107 em3=n.copy(tempo);em3[[0,LAMBDA_/3,2*LAMBDA_/3]]=1.
108
109 linha_em3=em3[ii%LAMBDA_]
110
111 #####
112 #RUIDOS
113
114 Lambda = 100000 # Lambda sempre par
115 # diferença das frequências entre coeficientes vizinhos:
116 df=f_a/float(Lambda)
117
118 # e fase aleatoria
119 coefs=n.exp(1j*n.random.uniform(0, 2*n.pi, Lambda))
120 # real par, imaginaria impar
121 coefs[Lambda/2+1:]=n.real(coefs[1:Lambda/2])[::-1] \
122                      - 1j*n.imag(coefs[1:Lambda/2])[::-1]
123 coefs[0]=0. # sem bias
124 coefs[Lambda/2]=1. # freq max eh real simplesmente
125
126 # as frequências relativas a cada coeficiente
127 # acima de Lambda/2 nao vale
128 fi=n.arange(coefs.shape[0])*df
129 f0=15. # iniciamos o ruido em 15 Hz
130 i0=n.floor(f0/df) # primeiro coeff a valer
131 coefs[:i0]=n.zeros(i0)
132 f0=fi[i0]
133
134 # realizando o ruído em suas amostras temporais
135 ruido=n.fft.ifft(coefs)
136 r=n.real(ruido)
137 rb=((r-r.min())/((r.max()-r.min()))*2-1 # ruido branco
138
139 # fazendo ruido preto
140 fator=10.**(-7/20.)
141 alphai=fator**(n.log2(fi[i0:]/f0))
142 c=n.copy(coefs)
143 c[i0:]=c[i0:]*alphai
144
145 # real par, imaginaria impar
146 c[Lambda/2+1:]=n.real(c[1:Lambda/2])[::-1] -\
147                1j*n.imag(c[1:Lambda/2])[::-1]
148
149 ruido=n.fft.ifft(c)
150 r=n.real(ruido)
151 rp=((r-r.min())/((r.max()-r.min()))*2-1
152
153 LR=rb[n.arange(int(len(linha_em3)*2.5))%len(rb)]*\
154     A(d=int(len(linha_em3)*2.5)/f_a,fa=.2,V_dB=50.)*10**(-60/20.)
155

```

```

156 LR2=rb[n.arange(int(len(linha_em3)*4.5))%len(rb)]*\
157     A(d=int(len(linha_em3)*4.5)/f_a)*.05
158
159 LR3=6.*rp[n.arange(int(len(linha_em3)*6.5))%len(rp)]*\
160     A(d=int(len(linha_em3)*6.5)/f_a)*.05
161
162
163 obj1=rb[:int(.4*f_a)]*A(d=.4,fa=15.)
164 obj2=rp[:int(.4*f_a)]*A(d=.4,fa=10.)
165
166 obj3=adsr(rb[:int(.4*f_a)]*A(d=.4,fa=15.))
167 obj4=adsr(rp[:int(.4*f_a)]*A(d=.4,fa=10.),S=-5)
168
169 obj5=adsr(rp[:int(1.4*f_a)]*A(d=1.4,fa=10.),5.,500.,-20,200)
170
171 #####
172
173 l1=n.convolve(obj1,linha_em3)[:len(linha_em3)]
174 l2=n.convolve(obj2,linha_em3)[:len(linha_em3)]
175 l3=n.convolve(obj3,linha_em3)[:len(linha_em3)]
176 l4=n.convolve(obj4,linha_em3)[:len(linha_em3)]
177 l6=n.convolve(obj5,linha_em3)[:len(linha_em3)]
178
179 l1_=n.convolve(obj1,linha_cabeca)[:len(linha_em3)]
180 l2_=n.convolve(obj2,linha_contra)[:len(linha_em3)]
181 l3_=n.convolve(obj3,linha_cabeca)[:len(linha_em3)]
182 l4_=n.convolve(obj4,linha_contra)[:len(linha_em3)]
183 l6_=n.convolve(obj5,linha_cabeca)[:len(linha_em3)]
184
185
186 linha1=n.convolve(som2,linha_cabeca)[:len(linha_cabeca)]
187 linha2=n.convolve(som4,linha_em3)[:len(linha_em3)]
188 linha4=n.convolve(som5,linha_em3)[:len(linha_em3)]
189 linha6=n.convolve(som6,linha_em3)[:len(linha_em3)]
190 linha3=n.convolve(som2,linha_contra)[:len(linha_contra)]
191
192
193 H_i=(n.random.random(int(f_a*1.2))*2-1)*n.e**(-n.arange(int(f_a*1.2)))
194 def r(l):
195     return n.convolve(H_i,l)[:len(linha_em3)]
196
197 som_e=n.hstack((r(linha2)+l1,linha3+r(l2),linha1+l3,
198                 r(linha1)+linha2+linha3,r(l6)))
199 som_e=n.hstack((som_e,r(linha4)+l1_,r(l2_),l3_+linha3+linha1,
200                 r(l4_)+linha1+linha3,r(l6_)+linha2))
201
202 som_d=n.hstack((linha1+r(linha2),linha2+linha3,r(linha3)+linha1,
203                 linha2+linha3+l4,linha2))
204 som_d=n.hstack((som_d,r(linha4)+l1_,r(l2_),l3_+linha4+linha1,
205                 r(l4_)+linha2+linha3,r(linha6)+linha2))

```

```
206
207 som=n.vstack((som_e,som_d))
208
209 som[:,len(LR)]+=LR
210
211 som[:,len(l1)*3:len(l1)*3+len(LR2)]+=LR2
212 som[:,int(len(l1)*3.5):int(len(l1)*3.5)+len(LR3)]+=LR3
213 som[:,len(l1)*7:len(l1)*7+len(LR)]+=LR
214
215 T_i=som
216
217 T_i=(T_i-T_i.min())/(T_i.max()-T_i.min())
218
219 a.wavwrite(n.hstack((T_i,T_i,T_i,T_i,T_i,T_i)).T,
220               "ruidosaFaixa4.wav",f_a)
```

B.2.6 Bela Rugosi

Pequena peça musical de 96 segundos para demonstração da rugosidade causada por oscilações com frequências $\approx 13 - 30\text{Hz}$, conceitos apresentados na subseção 2.2.5. A peça é também disponibilizada online junto à toolbox MASSA.(2)

bela rugosi

```

1  #-- coding: utf8 --
2  import numpy as n, scikits.audiolab as a
3  H=n.hstack
4  V=n.vstack
5
6  f_a = 44100. # Hz, frequência de amostragem
7
8  # Tamanho da LUT > 2**10 para usar também em oscilacoes
9  # de comprimento de onda grandes (LF0)
10 Lambda_tilde=Lt=(2.**5)*1024.
11 # Senoide
12 foo=n.linspace(0,2*n.pi,Lt,endpoint=False)
13 S_i=n.sin(foo) # um período da senóide com T amostras
14
15 # Quadrada:
16 Q_i=n.hstack( ( n.ones(Lt/2)*-1 , n.ones(Lt/2) ) )
17
18 # Triangular:
19 foo=n.linspace(-1,1,Lt/2,endpoint=False)
20 Tr_i=n.hstack( ( foo , foo*-1 ) )
21
22 # Dente de Serra:
23 D_i=n.linspace(-1,1,Lt)
24
25
26 def v(f=200,d=2.,tab=S_i,fv=2.,nu=2.,tabv=S_i):
27     Lambda=n.floor(f_a*d)
28     ii=n.arange(Lambda)
29     Lv=float(len(tabv))
30
31     Gammav_i=n.floor(ii*fv*Lv/f_a) # índices para a LUT
32     Gammav_i=n.array(Gammav_i,n.int)
33     # padrão de variação do vibrato para cada amostra
34     Tv_i=tabv[Gammav_i%int(Lv)]
35
36     # frequência em Hz em cada amostra
37     F_i=f*( 2.**( Tv_i*nu/12. ) )
38     # a movimentação na tabela por amostra

```

```

39     D_gamma_i=F_i*(Lt/float(f_a))
40     Gamma_i=n.cumsum(D_gamma_i) # a movimentação na tabela total
41     Gamma_i=n.floor( Gamma_i) # já os índices
42     Gamma_i=n.array( Gamma_i, dtype=n.int) # já os índices
43     return tab[Gamma_i%int(Lt)] # busca dos índices na tabela
44
45
46 dd=v(tabv=Tr_i ,d=2,fv=35.,nu=7.0)
47 dd2=v(tabv=D_i ,d=2,fv=35.,nu=7.0)
48
49 dd3=v(tabv=Tr_i,d=2,fv=35.,nu=7.0)
50 dd4=v(tabv=S_i ,d=2,fv=35.,nu=7.0)
51
52 dd5=v(tabv=Q_i ,d=2,fv=35.,nu=7.0)
53 dd6=v(tabv=S_i ,d=2,fv=35.,nu=7.0)
54
55 dd7=v(tabv=Q_i ,d=2,fv=35.,nu=7.0)
56 dd8=v(tabv=D_i ,d=2,fv=35.,nu=7.0)
57
58 zz=V((H((dd,dd3,dd5,dd7)), H((dd2,dd4,dd6,dd8))))
59 aa1=n.array(list(v(tabv=Q_i,fv=.5,f=200,nu=9))*4)
60 aa2=n.array(v(tabv=Tr_i,fv=.25/2.,f=200,nu=9,d=8.))
61 aa3=n.array(v(fv=.25/2.,f=200,nu=9,d=8.))
62
63 aa=n.vstack((( zz+aa1).T*.5, (zz+aa2).T*.5,(zz+aa3).T*.5))
64
65 dd= v(tab=Tr_i,tabv=Tr_i ,d=2,fv=35.,nu=7.0)
66 dd2=v(tab=Tr_i,tabv=D_i ,d=2,fv=35.,nu=7.0)
67
68 dd3=v(tab=Tr_i,tabv=Tr_i,d=2,fv=35.,nu=7.0)
69 dd4=v(tab=Tr_i,tabv=S_i ,d=2,fv=35.,nu=7.0)
70
71 dd5=v(tab=Tr_i,tabv=Q_i ,d=2,fv=35.,nu=7.0)
72 dd6=v(tab=Tr_i,tabv=S_i ,d=2,fv=35.,nu=7.0)
73
74 dd7=v(tab=Tr_i,tabv=Q_i ,d=2,fv=35.,nu=7.0)
75 dd8=v(tab=Tr_i,tabv=D_i ,d=2,fv=35.,nu=7.0)
76
77 zz=V((H((dd,dd3,dd5,dd7)), H((dd2,dd4,dd6,dd8))))
78 aa1=n.array(list(v(tabv=Q_i,fv=.5,f=200,nu=9))*4)
79 aa2=n.array(v(tabv=Tr_i,fv=.25/2.,f=200,nu=9,d=8.))
80 aa3=n.array(v(fv=.25/2.,f=200,nu=9,d=8.))
81
82 aa=n.vstack(( aa, (zz+aa1).T*.5, (zz+aa2).T*.5,(zz+aa3).T*.5))
83
84 dd= v(tab=Q_i,tabv=Tr_i ,d=2,fv=35.,nu=7.0)
85 dd2=v(tab=Q_i,tabv=D_i ,d=2,fv=35.,nu=7.0)
86
87 dd3=v(tab=Q_i,tabv=Tr_i,d=2,fv=35.,nu=7.0)
88 dd4=v(tab=Q_i,tabv=S_i ,d=2,fv=35.,nu=7.0)

```



```

89
90 dd5=v(tab=Q_i,tabv=Q_i ,d=2,fv=35.,nu=7.0)
91 dd6=v(tab=Q_i,tabv=S_i ,d=2,fv=35.,nu=7.0)
92
93 dd7=v(tab=Q_i,tabv=Q_i ,d=2,fv=35.,nu=7.0)
94 dd8=v(tab=Q_i,tabv=D_i ,d=2,fv=35.,nu=7.0)
95
96 zz=V((H((dd,dd3,dd5,dd7)), H((dd2,dd4,dd6,dd8))))
97 aa1=n.array(list(v(tabv=Q_i,fv=.5,f=200,nu=9))*4)
98 aa2=n.array(v(tabv=Tr_i,fv=.25/2.,f=200,nu=9,d=8.))
99 aa3=n.array(v(fv=.25/2.,f=200,nu=9,d=8.))
100
101 aa=n.vstack(( aa, (zz+aa1).T*.5, (zz+aa2).T*.5,(zz+aa3).T*.5))
102
103 dd= v(tab=D_i,tabv=Tr_i ,d=2,fv=35.,nu=7.0)
104 dd2=v(tab=D_i,tabv=D_i ,d=2,fv=35.,nu=7.0)
105
106 dd3=v(tab=D_i,tabv=Tr_i,d=2,fv=35.,nu=7.0)
107 dd4=v(tab=D_i,tabv=S_i ,d=2,fv=35.,nu=7.0)
108
109 dd5=v(tab=D_i,tabv=Q_i ,d=2,fv=35.,nu=7.0)
110 dd6=v(tab=D_i,tabv=S_i ,d=2,fv=35.,nu=7.0)
111
112 dd7=v(tab=D_i,tabv=Q_i ,d=2,fv=35.,nu=7.0)
113 dd8=v(tab=D_i,tabv=D_i ,d=2,fv=35.,nu=7.0)
114
115 zz=V((H((dd,dd3,dd5,dd7)), H((dd2,dd4,dd6,dd8))))
116 aa1=n.array(list(v(tabv=Q_i,fv=.5,f=200,nu=9))*4)
117 aa2=n.array(v(tabv=Tr_i,fv=.25/2.,f=200,nu=9,d=8.))
118 aa3=n.array(v(fv=.25/2.,f=200,nu=9,d=8.))
119
120 aa=n.vstack(( aa, (zz+aa1).T*.5, (zz+aa2).T*.5,(zz+aa3).T*.5))
121
122 print("BellaRugosiSdadE.wav escrito")
123 a.wavwrite(aa,"BellaRugosiSdadE.wav",f_a)

```

B.2.7 Chorus infantil

Script para demonstração do efeito *chorus* (inspirado em coro de cantores) . Sintetiza 4 pequenas montagens de 8-32 segundos com os conceitos apresentados na subseção 2.2.6 e disponibilizada online junto ao toolbox MASSA.(2)

B.2.8 ADa e SaRa

Pequenas peças musicais para demonstração da envoltória ADSR. Sintetiza 7 peças de 5-17 segundos, cetradas nos conceitos apresentados na subseção 2.2.6 e disponibilizada online junto à MASSA.(2)

B.3 Peças referentes à seção 2.3

B.3.1 Intervalos entre alturas

Pequena peça musical de 45 segundos para exploração do sistema de intervalos, demonstrativa dos conceitos apresentados na subseção 2.3.1, e disponibilizada online junto ao toolbox MASSA.

B.3.2 Cristais

Pequena peça musical de 64 segundos para demonstração das escalas simétricas, conceitos apresentados na subseção 2.3.1, e disponibilizada online junto à MASSA.(2)

cristais

```

1  #-- coding: utf8 --
2  import numpy as n, scikits.audiolab as a
3  H=n.hstack
4  V=n.vstack
5
6  f_a = 44100. # Hz, frequência de amostragem
7
8  Lambda_tilde=Lt=1024.*16
9
10 # Senoide
11 foo=n.linspace(0,2*n.pi,Lt,endpoint=False)
12 S_i=n.sin(foo) # um período da senóide com T amostras
13
14 # Quadrada:
15 Q_i=n.hstack( ( n.ones(Lt/2)*-1 , n.ones(Lt/2) ) )
16
17 # Triangular:
18 foo=n.linspace(-1,1,Lt/2,endpoint=False)
19 Tr_i=n.hstack( ( foo , foo*-1 ) )
20
21 # Dente de Serra:
22 D_i=n.linspace(-1,1,Lt)
23
24 def v(f=200,d=2.,tab=S_i,fv=2.,nu=2.,tabv=S_i):
25     Lambda=n.floor(f_a*d)
26     ii=n.arange(Lambda)
27     Lv=float(len(tabv))
28
29     Gammav_i=n.floor(ii*fv*Lv/f_a) # índices para a LUT

```

```

30     Gammav_i=n.array(Gammav_i,n.int)
31     # padrão de variação do vibrato para cada amostra
32     Tv_i=tabv[Gammav_i%int(Lv)]
33
34     # frequência em Hz em cada amostra
35     F_i=f*( 2.** ( Tv_i*nu/12. ) )
36     # a movimentação na tabela por amostra
37     D_gamma_i=F_i*(Lt/float(f_a))
38     Gamma_i=n.cumsum(D_gamma_i) # a movimentação na tabela total
39     Gamma_i=n.floor( Gamma_i) # já os índices
40     Gamma_i=n.array( Gamma_i, dtype=n.int) # já os índices
41     return tab[Gamma_i%int(Lt)] # busca dos índices na tabela
42
43 def A(fa=2.,V_dB=10.,d=2.,taba=S_i):
44     Lambda=n.floor(f_a*d)
45     ii=n.arange(Lambda)
46     Lt=float(len(taba))
47     Gammaa_i=n.floor(ii*fa*Lt/f_a) # índices para a LUT
48     Gammaa_i=n.array(Gammaa_i,n.int)
49     # variação da amplitude em cada amostra
50     A_i=taba[Gammaa_i%int(Lt)]
51     A_i=A_i*10.** (V_dB/20.)
52     return A_i
53
54 def adsr(som,A=10.,D=20.,S=-20.,R=100.,xi=1e-2):
55     a_S=10** (S/20.)
56     Lambda=len(som)
57     Lambda_A=int(A*f_a*0.001)
58     Lambda_D=int(D*f_a*0.001)
59     Lambda_R=int(R*f_a*0.001)
60
61     ii=n.arange(Lambda_A,dtype=n.float)
62     A=ii/(Lambda_A-1)
63     A_i=A
64     ii=n.arange(Lambda_A,Lambda_D+Lambda_A,dtype=n.float)
65     D=1-(1-a_S)* ( ( ii-Lambda_A )/( Lambda_D-1) )
66     A_i=n.hstack( ( A_i, D ) )
67     S=n.ones(Lambda-Lambda_R-(Lambda_A+Lambda_D),dtype=n.float)*a_S
68     A_i=n.hstack( ( A_i, S ) )
69     ii=n.arange(Lambda-Lambda_R,Lambda,dtype=n.float)
70     R=a_S-a_S* ( ( ii-(Lambda-Lambda_R))/(Lambda_R-1) )
71     A_i=n.hstack( ( A_i,R ) )
72
73     return som*A_i
74
75
76 # 2.7.5. Escalas simétricas
77 Ec_i=[0.,1.,2.,3.,4.,5.,6.,7.,8.,9.,10.,11.]
78 Et_i=[0.,2.,4.,6.,8.,10.]
79 Etm_i=[0.,3.,6.,9.]

```

```

80 EtM_i=[0.,4.,8.]
81 Ett_i=[0.,6.]
82
83 notast_A=[adsr( v(200*2**((i/12.)),d=.2 ) ) for i in Et_i]
84
85 notast_D=[adsr( v(400*2**((-i/12.)),d=.2 ) ) for i in Et_i]
86 s=H( notast_D +notast_A+notast_D)
87 sl=n.copy(s)
88
89 notastm_A=[adsr( v(200*2**((i/12.)),d=.2 ) ) for i in Etm_i]
90
91 notastm_D=[adsr( v(400*2**((-i/12.)),d=.2 ) ) for i in Etm_i]
92 s=H(( notastm_D +notastm_A+notastm_D))
93
94 notastM_A=[adsr( v(200*2**((i/12.)),d=.2 ) ) for i in EtM_i]
95
96 notastM_D=[adsr( v(400*2**((-i/12.)),d=.2 ) ) for i in EtM_i]
97 s=H(( [list(s)]+ notastM_D +notastM_A+notastM_D))
98
99 notasttt_A=[adsr( v(200*2**((i/12.)),d=.2 ) ) for i in Ett_i]
100
101 notasttt_D=[adsr( v(400*2**((-i/12.)),d=.2 ) ) for i in Ett_i]
102 s=H(( [list(s)]+ notasttt_D +notasttt_A+notasttt_D))
103
104 s=H((s,s[:::-1]))
105 s=H((s,s[::2],s[:::-2]))
106 s=H(( s,s+H((s[::2],s[:::-2])),
107         s+H((s[::4],s[:::-4],s[::4],s[:::-8],s[:::-8])) ))
108
109 s=((s-s.min())/(s.max()-s.min()))*2.-1.
110 a.wavwrite(s,"cristais.wav",f_a) # escrita do som

```

B.3.3 Micro tom

Pequena peça musical para demonstração da microtonalidade. Usados de quartos de tom e de sétimos de oitava, conceitos apresentados na subseção 2.3.1. Disponibilizada *online* junto à MASSA.(2)

micro tom

```

1  #-*- coding: utf8 -*-
2  import numpy as n, scikits.audiolab as a
3  import random as r
4  H=n.hstack
5  V=n.vstack
6
7  f_a = 44100. # Hz, frequência de amostragem
8
9  ##### 2.2.1 Tabela de busca (LUT)
10 Lambda_tilde=Lt=1024.*16
11
12 # Senoide
13 foo=n.linspace(0,2*n.pi,Lt,endpoint=False)
14 S_i=n.sin(foo) # um período da senóide com T amostras
15
16 # Quadrada:
17 Q_i=n.hstack( ( n.ones(Lt/2)*-1 , n.ones(Lt/2) ) )
18
19 # Triangular:
20 foo=n.linspace(-1,1,Lt/2,endpoint=False)
21 Tr_i=n.hstack( ( foo , foo*-1 ) )
22
23 # Dente de Serra:
24 D_i=n.linspace(-1,1,Lt)
25
26
27 def v(f=200,d=2.,tab=S_i,fv=2.,nu=2.,tabv=S_i):
28     Lambda=n.floor(f_a*d)
29     ii=n.arange(Lambda)
30     Lv=float(len(tabv))
31
32     Gammav_i=n.floor(ii*fv*Lv/f_a) # índices para a LUT
33     Gammav_i=n.array(Gammav_i,n.int)
34     # padrão de variação do vibrato para cada amostra
35     Tv_i=tabv[Gammav_i%int(Lv)]
36
37     # frequência em Hz em cada amostra
38     F_i=f*( 2.** ( Tv_i*nu/12. ) )

```

```

39     # a movimentação na tabela por amostra
40     D_gamma_i=F_i*(Lt/float(f_a))
41     Gamma_i=n.cumsum(D_gamma_i) # a movimentação na tabela total
42     Gamma_i=n.floor( Gamma_i) # já os índices
43     Gamma_i=n.array( Gamma_i, dtype=n.int) # já os índices
44     return tab[Gamma_i%int(Lt)] # busca dos índices na tabela
45
46
47 def A(fa=2.,V_dB=10.,d=2.,taba=S_i):
48     Lambda=n.floor(f_a*d)
49     ii=n.arange(Lambda)
50     Lt=float(len(tab))
51     Gammaa_i=n.floor(ii*fa*Lt/f_a) # índices para a LUT
52     Gammaa_i=n.array(Gammaa_i,n.int)
53     A_i=taba[Gammaa_i%int(Lt)]
54     A_i=A_i*10.**((V_dB/20.))
55     return A_i
56
57
58 def adsr(som,A=10.,D=20.,S=-20.,R=100.,xi=1e-2):
59     a_S=10**((S/20.))
60     Lambda=len(som)
61     Lambda_A=int(A*f_a*0.001)
62     Lambda_D=int(D*f_a*0.001)
63     Lambda_R=int(R*f_a*0.001)
64
65     ii=n.arange(Lambda_A,dtype=n.float)
66     A=ii/(Lambda_A-1)
67     A_i=A
68     ii=n.arange(Lambda_A,Lambda_D+Lambda_A,dtype=n.float)
69     D=1-(1-a_S)*((ii-Lambda_A)/(Lambda_D-1))
70     A_i=n.hstack((A_i, D))
71     S=n.ones(Lambda-Lambda_R-(Lambda_A+Lambda_D),dtype=n.float)*a_S
72     A_i=n.hstack((A_i, S))
73     ii=n.arange(Lambda-Lambda_R,Lambda,dtype=n.float)
74     R=a_S-a_S*((ii-(Lambda-Lambda_R))/(Lambda_R-1))
75     A_i=n.hstack((A_i,R))
76
77     return som*A_i
78
79
80 def ac(f=200.,notas=[0.,4.,7.,12.],tab=S_i):
81     acorde=adsr(v(tab=tab,f=f*2.**((notas[-1]/12.)),nu=0))
82     for na in notas[:-1]:
83         acorde+=adsr(v(tab=tab,f=f*2.**((na/12.)),nu=0))
84
85     return acorde
86
87
88 # Microtonalidade

```

```

89  epsilon=2** (1/12.) # divisão da oitava em 12 semitons
90  # quartos de tom sao metade do semitom
91  ff=[0.,0.5,3.5,4.,4.5,5. , 7.,7.5]
92  dd=[.5,.25, .25,.5/2,.25, .25,.5/4,.5/4]
93  tt=[adsr(v(tab=Tr_i,f=200*2**(f/12),d=d,nu=0),R=30.) for
94      f,d in zip(ff,dd)]
95  s=H((tt+tt[::-1]))
96
97  ff=[3.5,4.,4.5, 6.5, 7.,7.5, 7. , 6.5]
98  dd=[.5,.25, .25, .5/2, .25, .25,.5/4,.5/4]
99  tt=[adsr(v(tab=Tr_i,f=200*2**(f/12),d=d,nu=0),R=30.) for
100     f,d in zip(ff,dd)]
101  so=H((tt+tt[::-1]))
102  s=H((s)+tt+tt[::-1]))
103
104  # ou em 7 grados na oitava com um estilo tradicional tailandês
105  epsilon=2** (1/7.)
106  ff=[0.,1.,2.,3.,4.,5.,6.,7.]
107  dd=[.8,0.4,0.2,.1,0.4]
108
109  ff=[200.*2.** (r.choice(ff)/7.) for i in xrange(15*3)] # 15 notas
110  dd=[r.choice(dd) for i in xrange(15*3)] # 15 notas
111
112  tt=[adsr(v(tab=Tr_i,f=f,d=d,nu=0),R=50.) for f,d in zip(ff,dd)]
113  s=H((s)+tt+tt[::-1]))
114
115  sa=H((s[::2],s[:::-2],s[::2],s[:::-2]))
116
117  ff=[0.,1.,2.,3.,4.,5.,6.,7.]
118  ff=[50.*2.** (r.choice(ff)/7.) for i in xrange(15*3)] # 15 notas
119  dd=[.8,0.4,0.2,.1,0.4]
120  dd=[r.choice(dd) for i in xrange(15*3)] # 15 notas
121
122  tt=[adsr(v(tab=Tr_i,f=f,d=d,nu=7.,fv=12),R=50.) for
123      f,d in zip(ff,dd)]
124  sg=H((tt+tt[::-1]))*2.
125
126  m=min(len(so),len(sa),len(sg))
127  foo=H((so[:m]+sa[:m]+sg[:m]))
128
129  s=H((s,foo,foo,foo,foo,foo,foo,foo))
130
131  s=((s-s.min())/(s.max()-s.min()))*2.-1.
132  a.wavwrite(s,"microTom.wav",f_a)

```


B.3.4 Acorde cedo

Pequena peça musical de 40 segundos para demonstração dos encadeamentos tonais básicos, conceitos apresentados na subseção 2.3.2, e disponibilizada online como parte da MASSA.(2)

B.3.5 Conta ponto

Pequena montagem musical de 17 segundos para demonstração de vozes conduzidas por regras simples para manterem independência, conceitos apresentados na subseção 2.3.3, e disponibilizada online junto à MASSA.(2)

B.3.6 Poli Hit Mia

Pequena montagem musical de 99 segundos para demonstração da métrica musical e das diferentes contagens, conceitos apresentados na seção 2.3.4, e disponibilizada online junto à MASSA.(2)

B.3.7 Dirracional

Pequena peça musical de 22 segundos para demonstração das estruturas direcionais e posicionamentos dos clímax, conceitos apresentados na seção 2.3, e disponibilizada online junto à MASSA.(2)

dirracional

```

1  -*- coding: utf8 -*-
2  import numpy as n, scikits.audiolab as a
3  import random as r
4  H=n.hstack
5  V=n.vstack
6
7  f_a = 44100. # Hz, frequência de amostragem
8
9  ##### 2.2.1 Tabela de busca (LUT)
```

```

10 Lambda_tilde=Lt=1024.*16
11
12 # Senoide
13 foo=n.linspace(0,2*n.pi,Lt,endpoint=False)
14 S_i=n.sin(foo) # um período da senóide com T amostras
15
16 # Quadrada:
17 Q_i=n.hstack( ( n.ones(Lt/2)*-1 , n.ones(Lt/2) ) )
18
19 # Triangular:
20 foo=n.linspace(-1,1,Lt/2,endpoint=False)
21 Tr_i=n.hstack( ( foo , foo*-1 ) )
22
23 # Dente de Serra:
24 D_i=n.linspace(-1,1,Lt)
25
26
27 def v(f=200,d=2.,tab=S_i,fv=2.,nu=2.,tabv=S_i):
28     Lambda=n.floor(f_a*d)
29     ii=n.arange(Lambda)
30     Lv=float(len(tabv))
31
32     Gammav_i=n.floor(ii*fv*Lv/f_a) # índices para a LUT
33     Gammav_i=n.array(Gammav_i,n.int)
34     # padrão de variação do vibrato para cada amostra
35     Tv_i=tabv[Gammav_i%int(Lv)]
36
37     # frequência em Hz em cada amostra
38     F_i=f*( 2.** ( Tv_i*nu/12. ) )
39     # a movimentação na tabela por amostra
40     D_gamma_i=F_i*(Lt/float(f_a))
41     Gamma_i=n.cumsum(D_gamma_i) # a movimentação na tabela total
42     Gamma_i=n.floor( Gamma_i ) # já os índices
43     Gamma_i=n.array( Gamma_i, dtype=n.int) # já os índices
44     return tab[Gamma_i%int(Lt)] # busca dos índices na tabela
45
46 def A(fa=2.,V_dB=10.,d=2.,taba=S_i):
47     Lambda=n.floor(f_a*d)
48     ii=n.arange(Lambda)
49     Lt=float(len(taba))
50     Gammaa_i=n.floor(ii*fa*Lt/f_a) # índices para a LUT
51     Gammaa_i=n.array(Gammaa_i,n.int)
52     # variação da amplitude em cada amostra
53     A_i=taba[Gammaa_i%int(Lt)]
54     A_i=A_i*10.** (V_dB/20.)
55     return A_i
56
57 def adsr(som,A=10.,D=20.,S=-20.,R=100.,xi=1e-2):
58     a_S=10** (S/20.)
59     Lambda=len(som)

```

```

60     Lambda_A=int(A*f_a*0.001)
61     Lambda_D=int(D*f_a*0.001)
62     Lambda_R=int(R*f_a*0.001)
63
64     ii=n.arange(Lambda_A,dtype=n.float)
65     A=ii/(Lambda_A-1)
66     A_i=A
67     ii=n.arange(Lambda_A,Lambda_D+Lambda_A,dtype=n.float)
68     D=1-(1-a_S)*((ii-Lambda_A)/(Lambda_D-1))
69     A_i=n.hstack((A_i, D))
70     S=n.ones(Lambda-Lambda_R-(Lambda_A+Lambda_D),dtype=n.float)*a_S
71     A_i=n.hstack((A_i, S))
72     ii=n.arange(Lambda-Lambda_R,Lambda,dtype=n.float)
73     R=a_S-a_S*((ii-(Lambda-Lambda_R))/(Lambda_R-1))
74     A_i=n.hstack((A_i,R))
75
76     return som*A_i
77
78     seq=((i-1.)/i for i in range(60))[1::2]
79     # particionamento d oitava em 17 grados, 18 com a oitava
80     seq=[i for i in n.linspace(0,1,14)]
81     s1=seq[::2]+seq[:::-2] # simetrica
82     s2=seq[::4]+seq[:::-2] # na primeira metade
83     s3=seq[::2]+seq[:::-4] # na segunda metade
84     s4=seq # no comec
85     s5=seq[:::-1] # no fim
86     pausa=p=[0.]*8
87     s_=s1+p+s2+p+s3+p+s4+p+s5+p+p
88
89     s=H(( [adsr(v(tab=Tr_i,f=800.*2.**((ss),nu=0.5,d=.2,fv=20),
90                                     70.,100.,R=20.) for ss in s_] ))
91     s=((s-s.min())/((s.max()-s.min())))*2-1
92
93     a.wavwrite(s,"dirracional.wav",f_a)

```


APÊNDICE C – Finite Groups in Granular and Unit Synthesis e a síntese de um EP

C.1 FIGGUS

Escrito como um módulo python, o FIGGUS sintetiza estruturas musicais através de permutações, como apresentadas na subseção 2.3.7. Utiliza os princípios da MASSA e os códigos foram reescritos para python nativo, i.e. sem a utilização de bibliotecas externas como o Numpy e o Audiolab, para facilitar o uso de terceiros. O FIGGUS é parte da toolbox MASSA como exemplo de implementação dos princípios da *toolbox* com as bibliotecas padrão da linguagem Python.(2)

C.1.1 FIGGUS.py

Arquivo principal, possui todas as rotinas.

C.1.2 tables.py

Arquivo auxiliar para tratar as tabelas separadamente.

C.1.3 __init__.py

Inicialização do módulo.

C.2 PPEPPS: músicas de um EP solvente

O PPEPPS (Pure Python EP: Projeto Solvente) usa o FIGGUS para sintetizar um EP inteiro. As músicas estão abaixo, junto com o arquivo que executa cada uma.

C.2.1 RUNME make EP MUSIC.py

Arquivo que executa os outros um por um para sintetizar as músicas do EP.

RUNME_make_now_an_EP_MUSIC.py

```

1  import os
2
3  os.mkdir('ep-Projeto_Solvente')
4
5  execfile("examples/triangulo3_B.py")
6  os.system('mv sound.wav ep-Projeto_Solvente/01-Eter.wav')
7  print("gravado o arquivo 'ep-Projeto_Solvente/01-Eter.wav'")
8
9  execfile("examples/try4_B.py")
10 os.system('mv sound.wav ep-Projeto_Solvente/04-Butano.wav')
11 print("gravado o arquivo 'ep-Projeto_Solvente/04-Butano.wav'")
12
13 execfile("examples/try5_B.py")
14 os.system('mv sound.wav ep-Projeto_Solvente/05-Thinner.wav')
15 print("gravado o arquivo 'ep-Projeto_Solvente/05-Thinner.wav'")
16
17 execfile("examples/try6_B.py")
18 os.system('mv sound.wav ep-Projeto_Solvente/06-Tolueno.wav')
19 print("gravado o arquivo 'ep-Projeto_Solvente/06-Tolueno.wav'")
20
21 execfile("examples/try7.py")
22 os.system('mv both.wav ep-Projeto_Solvente/07-Benzina.wav')
23 print("gravado o arquivo 'ep-Projeto_Solvente/07-Benzina.wav'")
24
25 execfile("examples/try3_B.py")
26 os.system('mv sound.wav ep-Projeto_Solvente/08-LSA.wav')
27 print("gravado o arquivo 'ep-Projeto_Solvente/08-LSA.wav'")
28
29 execfile("examples/try2.py")
30 os.system('mv sound.wav ep-Projeto_Solvente/09-Cloroformio.wav')
31 print("gravado o arquivo 'ep-Projeto_Solvente/09-Cloroformio.wav'")
32
33 execfile("examples/try5.py")

```

```

34 os.system('mv sound.wav ep-Projeto_Solvente/10-Agua.wav')
35 print("gravado o arquivo 'ep-Projeto_Solvente/10-Agua.wav'")

```

C.2.2 Éter

```

1  import figgus as f
2
3  n=3
4  grains=[f.UnitGrain(),f.UnitGrain(),f.UnitGrain()]
5  s=f.Sequence(grains)
6
7  def doCompass(durs,freqs,perms,ncomps):
8      for i in xrange(n):
9          s.ordered_units[i].freq=eval(freqs)
10         s.ordered_units[i].duration=eval(durs)
11
12         p=f.Pattern(s,f.PermutationPattern(perms),ncomps)
13         p.synthesizeSonicVectors()
14         return p.sonic_vector
15
16 semitom = "2**((1/12.))" # 12 semitons sums to an octave
17 dur=".5"
18 sound=[]
19
20 #####
21
22
23 sounda=doCompass(dur,"200*("+semitom+"）**((3*i))", (3,1,2),3*4)
24 soundb=doCompass(dur,"200*2*("+semitom+"）**((3*i))", (2,3,1),3*4)
25 sound = sounda+[(i+j)*.5 for i,j in zip(sounda,soundb)]
26
27 dur=".3"
28 sounda1=doCompass(dur,"100*("+semitom+"）**((3*i))", (3,1,2),3*4)
29 soundb1=doCompass(dur,"200*(2/3.)*("+semitom+"）**((3*i))", (2,3,1),3*4)
30 stemp = [(i+j)*.5 for i,j in zip(sounda1,soundb1)]
31 sound += stemp
32
33 sound += sounda1+soundb1
34
35 sounda=doCompass(dur,"200*("+semitom+"）**((3*i))", (3,1,2),3*4)
36 soundb=doCompass(dur,"200*("+semitom+"）**((3*i))", (2,3,1),3*4)
37
38 stemp2 = [(i+j)*.5 for i,j in zip(sounda,soundb)]
39 stemp3 = [(i+j)*.5 for i,j in zip(stemp,stemp2)]
40 sound += 2*stemp3
41
42 io=f.IOUtils()
43 io.recordFile(sound) #records a mono 16 bit
44

```

```
45 print("\n== triangulo gira em crescente ==\n")
```


C.2.3 Benzina

```

1 import figgus as f
2
3 n=4
4 grains=[f.UnitGrain(),f.UnitGrain(),f.UnitGrain(),f.UnitGrain()]
5 s=f.Sequence(grains)
6
7 def doCompass(durs,freqs,perms,ncomps):
8     for i in xrange(n):
9         s.ordered_units[i].freq=eval(freqs)
10        s.ordered_units[i].duration=eval(durs)
11        #print 1*(i+1)
12
13        p=f.Pattern(s,f.PermutationPattern(perms),ncomps)
14        p.synthesizeSonicVectors()
15        return p.sonic_vector
16
17 semitom = "2**(1/12.)"
18 dur=".2"
19 sound=[]
20
21 #####
22 sound1=doCompass(dur,"200*("+semitom+"**(3*i))",(2,3,1),3)
23 sound1B=doCompass(dur,"4*200*("+semitom+"**(3*i))",(2,3,1),3)
24 sound1C=doCompass(dur,".5*200*("+semitom+"**(3*i))",(2,3,1),3)
25 # Inverting quirkality
26 sound2=doCompass(dur,"200*("+semitom+"**(3*i))",(3,1,2),3)
27 sound2B=doCompass(dur,"4*200*("+semitom+"**(3*i))",(3,1,2),3)
28 sound2C=doCompass(dur,".5*200*("+semitom+"**(3*i))",(3,1,2),3)
29 silencio=doCompass(dur,"0",(2,3,1),1)
30
31
32
33 left=[(i+j+k)*.32 for i,j,k in zip(sound1,sound2B,sound1C,)]
34 left1=left[:]
35 right=[(i+j+k)*.32 for i,j,k in zip(sound2,sound1B,sound2C,)]
36 right1=right[:]
37 left += right1
38 right += left1
39 left*=2
40 right*=2
41
42 left2=[(i+j+k)*.32 for i,j,k in zip(sound1,sound1B,sound2C,)]
43 left+=left2
44 right2=[(i+j+k)*.32 for i,j,k in zip(sound2,sound2B,sound1C,)]
45 right+=right2
46 left+=right2
47 right+=left2

```

```

48
49 left30=[(i+j+k)*.32 for i,j,k in zip(sound1,sound2B,sound2C)]
50 left+=left30
51 right30=[(i+j+k)*.32 for i,j,k in zip(sound2,sound2B,sound1C)]
52 right+=right30
53 left+=right30
54 right+=left30
55
56
57 left3=[(i+j+k)*.32 for i,j,k in zip(sound2,sound2B,sound2C)]
58 left+=left3
59 right3=[(i+j+k)*.32 for i,j,k in zip(sound2,sound2B,sound1C)]
60 right+=right3
61 left+=right3
62 right+=left3
63
64 ###=====
65
66 L=len(sound2)
67 left33= sound2[:L/3]+sound2B[L/3:2*L/3]+sound2C[2*L/3:] # Sobe
68 right33=sound2C[:L/3]+sound2[L/3:2*L/3]+sound2B[2*L/3:] # Desce
69 left+=left33+right33
70 right+=right33+left33
71
72 left4= sound1[:L/3]+sound1C[L/3:2*L/3]+sound1B[2*L/3:] # Sobe
73 right4=sound1B[:L/3]+sound1[L/3:2*L/3]+sound1C[2*L/3:] # Sobe
74 left+=(left4+right4) * 2
75 right+=(right4+left4) * 2
76
77 left5= sound1[:L/3]+sound1B[L/3:2*L/3]+sound1C[2*L/3:] # Desce
78 right5=sound1B[:L/3]+sound1C[L/3:2*L/3]+sound1[2*L/3:] # Desce
79 left+=(left5+right5) * 2
80 right+=(right5+left5) * 2
81
82 left6= sound2[:L/3]+sound2B[L/3:2*L/3]+sound2C[2*L/3:] # Desce
83 right6=sound2C[:L/3]+sound2B[L/3:2*L/3]+sound2[2*L/3:] # Sobe
84 left+=left6+right6
85 right+=right6+left6
86
87 ###=====
88
89
90 left+=left3
91 right+=right3
92 left+=right3
93 right+=left3
94
95 #####
96

```

```
97
98 left*=2
99 right*=2
100
101 io=f.IOUtils()
102 #io.recordFile(left,[],"left.wav") #records a mono 16 bit
103 #io.recordFile(right,[],"right.wav") #records a mono 16 bit
104 io.recordFile(left,right,"both.wav") #records a mono 16 bit
```

4 Clorofórmio

[illegible]

```

48 p=f.Pattern(s,f.PermutationPattern(),8)
49 p.synthesizeSonicVectors()
50 sound+=p.sonic_vector
51
52 # COMPASS 9–12 ..... 3
53 for i in xrange(n):
54     s.ordered_units[i].freq=10000 - 1000*(i+1)
55     s.ordered_units[i].duration=.05*i#.2# + 0.05*i
56
57 s.ordered_units[1].freq=100
58 s.ordered_units[3].freq=500
59
60 #pp=PermutationPattern()
61 p=f.Pattern(s,f.PermutationPattern(),8)
62 p.synthesizeSonicVectors()
63 sound+=p.sonic_vector
64
65 #####
66
67 # COMPASS 13–16 ..... 4
68 for i in xrange(n):
69     s.ordered_units[i].freq=10000 - 1000*(i+1)
70     s.ordered_units[i].duration=.1#.2# + 0.05*i
71
72 s.ordered_units[1].freq=100
73 s.ordered_units[3].freq=500
74
75 #pp=PermutationPattern()
76 p=f.Pattern(s,f.PermutationPattern(),8)
77 p.synthesizeSonicVectors()
78 sound+=p.sonic_vector
79
80 ##### 4 compassos de 4 1
81 # COMPASS 1–4 ..... 1
82 for i in xrange(n):
83     s.ordered_units[i].freq=10000 - 1000*(i+1)
84     s.ordered_units[i].duration=.2 + 0.05*i
85
86 s.ordered_units[1].freq=100
87 s.ordered_units[3].freq=500*3./2 # Uma quinta a cima o pedal/bola
88
89 #pp=PermutationPattern()s
90 p=f.Pattern(s,f.PermutationPattern(),8)
91 p.synthesizeSonicVectors()
92 sound+=p.sonic_vector
93
94
95 # COMPASS 5–8 ..... 2
96 for i in xrange(n):

```

```

97     s.ordered_units[i].freq=10000 - 1000*(i+1)
98     s.ordered_units[i].duration=.2# + 0.05*i
99
100 s.ordered_units[1].freq=100
101 s.ordered_units[3].freq=500*3./2 # Uma quinta a cima o pedal/bola
102
103 p=f.Pattern(s,f.PermutationPattern(),8)
104 p.synthesizeSonicVectors()
105 sound+=p.sonic_vector
106
107 # COMPASS 9-12 ..... 3
108 for i in xrange(n):
109     s.ordered_units[i].freq=10000 - 1000*(i+1)
110     s.ordered_units[i].duration=.01*i**2#.2# + 0.05*i
111
112 s.ordered_units[1].freq=100
113 s.ordered_units[3].freq=500*3./2 # Uma quinta a cima o pedal/bola
114
115 p=f.Pattern(s,f.PermutationPattern(),8)
116 p.synthesizeSonicVectors()
117 sound+=p.sonic_vector
118
119 #####
120 # COMPASS 13-16 ..... 4
121 for i in xrange(n):
122     s.ordered_units[i].freq=10000 - 1000*(i+1)
123     s.ordered_units[i].duration=.1#.2# + 0.05*i
124
125 s.ordered_units[1].freq=100
126 s.ordered_units[3].freq=500*3./2 # Uma quinta a cima o pedal/bola
127
128 p=f.Pattern(s,f.PermutationPattern(),4)
129 p.synthesizeSonicVectors()
130 sound+=p.sonic_vector
131
132 ##### 4 compass de 4 ::: 2
133 # COMPASS 1-4 ..... 1
134 for i in xrange(n):
135     s.ordered_units[i].freq=10000 - 1000*(i+1)
136     s.ordered_units[i].duration=.2 + 0.05*i
137     #print 100*(i+1)
138
139 s.ordered_units[1].freq=100
140 s.ordered_units[3].freq=500/2
141
142 #pp=PermutationPattern()
143 p=f.Pattern(s,f.PermutationPattern(),8)
144 p.synthesizeSonicVectors()
145 sound+=p.sonic_vector

```

```

146
147
148 # COMPASS 5-8 ..... 2
149 for i in xrange(n):
150     s.ordered_units[i].freq=10000 - 1000*(i+1)
151     s.ordered_units[i].duration=.2# + 0.05*i
152     #print 100*(i+1)
153
154 s.ordered_units[1].freq=100
155 s.ordered_units[3].freq=500/2
156
157 p=f.Pattern(s,f.PermutationPattern(),8)
158 p.synthesizeSonicVectors()
159 sound+=p.sonic_vector
160
161 # COMPASS 9-12 ..... 3
162 for i in xrange(n):
163     s.ordered_units[i].freq=10000 - 1000*(i+1)
164     s.ordered_units[i].duration=.2*i#.2# + 0.05*i
165
166 s.ordered_units[1].freq=100
167 s.ordered_units[3].freq=500/2
168
169 p=f.Pattern(s,f.PermutationPattern(),8)
170 p.synthesizeSonicVectors()
171 sound+=p.sonic_vector
172
173 #####
174 # COMPASS 13-16 ..... 4
175 for i in xrange(n):
176     s.ordered_units[i].freq=10000 - 1000*(i+1)
177     s.ordered_units[i].duration=.1*i#.2# + 0.05*i
178
179 s.ordered_units[1].freq=100
180 s.ordered_units[3].freq=500/2
181
182 #pp=PermutationPattern()
183 p=f.Pattern(s,f.PermutationPattern(),8)
184 p.synthesizeSonicVectors()
185 sound+=p.sonic_vector
186 ##### Acabou 3 x 4 compass de 4 == tonica dominante tonica
187
188 io=f.IOUtils()
189 io.recordFile(sound)
190 print("== Ouca que 'o triangulo gira \
191     e existe uma bola do lado' (nome da peca) ==")

```


APÊNDICE D – Síntese FM e AM em escala logarítmica

$$\begin{aligned}
 \{t'_i\}_0^{\Lambda-1} &= \left\{ \cos \left[f_i \cdot 2\pi \frac{i}{f_a - 1} \right] \right\}_0^{\Lambda-1} = \left\{ \cos \left[f \left(\frac{f + \mu}{f} \right)^{\text{sen} \left(f' \cdot 2\pi \frac{i}{f_a - 1} \right)} \cdot 2\pi \frac{i}{f_a - 1} \right] \right\}_0^{\Lambda-1} = \\
 &= \left\{ \cos \left[f \left(2^{\frac{\gamma}{12}} \right)^{\text{sen} \left(f' \cdot 2\pi \frac{i}{f_a - 1} \right)} \cdot 2\pi \frac{i}{f_a - 1} \right] \right\}_0^{\Lambda-1} = \left\{ \cos \left[f \left(2^{\frac{\gamma}{12} \sin \left(f' \cdot 2\pi \frac{i}{f_a - 1} \right)} \right) \cdot 2\pi \frac{i}{f_a - 1} \right] \right\}_0^{\Lambda-1} \Rightarrow \\
 &\Rightarrow \text{usando: } \left(2^x = e^{x \ln 2}, \quad e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}, \quad \sin^n x = y_1(n) \sum_k y_2(n, k) \cdot \sin[y_3(n, k, x)] \right) \Rightarrow \\
 \Rightarrow \{t'_i\}_0^{\Lambda-1} &= \left\{ \sum_{k=-\infty}^{+\infty} J_k(\mu, f) \cos[y(f, f', f_a)] \right\}_0^{\Lambda-1}
 \end{aligned}
 \tag{D.1}$$

Por brevidade, as identidades úteis para a obtenção da equação final, que explicita o conteúdo em frequências puras. Portanto, se utilizada a progressão exponencial de frequência, o espectro da FM perde a simplicidade. A função de Bessel dependerá não somente da profundidade de oscilação, mas também da frequência original do som. Além disso, as frequências dependem do produto das frequências da portadora e da moduladora ($f \cdot f'$), o que também introduz complexidade.

No caso da modulação de amplitude na escala logarítmica:

$$\{t'_i\}_0^{\Lambda-1} = \{(a_{\text{máx}})^{ti'} \cdot t_i\}_0^{\Lambda-1} = \left\{ a_{\text{máx}}^{\sin \left(f' \cdot 2\pi \frac{i}{f_a - 1} \right)} \cdot P \cdot \sin \left(f \cdot 2\pi \frac{i}{f_a - 1} \right) \right\}_0^{\Lambda-1}
 \tag{D.2}$$

através das mesmas três identidades, observa-se um espectro com complexidade similar.

APÊNDICE E – Análise sonora e musical

Este capítulo foi escrito para complementar a didática da dissertação pois há forte interesse em análise musical assistida por computador, tanto para a indústria, como para pesquisas acadêmicas e musicológicas. Assim, o texto a seguir é uma introdução rápida e deve ser apreciado como um conjunto despretencioso de apontamentos informais a conceitos básicos de análise de música em evidência com os recursos computacionais.

E.1 Espectro em frequência

Muitas vezes, se está preocupado com as componentes em frequência de um dado som ou estilo. Para tal, a transformada de Fourier é a solução de uso mais comum. A localização de energia no tempo e em frequência, por exemplo para observar as componentes do espectro no decorrer de uma sonoridade ou uma música, é geralmente feita com a transformada de Fourier janelada ou com a transformada *wavelet* (transformada de ondeleta). Uma figura bidimensional da evolução do espectro no decorrer do tempo é o espectrograma (veja figura 2.19). A transformada wavelet é bastante simples e computacionalmente barata, consistindo em um banco de filtros FIR. Por outro lado, o espectro apresentado não é senoidal, tão caro à cognição e à música. Além disso, a transformada wavelet é mais rígida com relação à resolução temporal e frequencial do que a transformada de Fourier janelada.

Com a observação espectral, verifica-se a presença de formantes, detecta-se frequências funda-

mentais, calcula-se centróides, observa-se decaimento e aparecimento de harmônicas e ruído, para citar somente alguns exemplos. O vínculo direto do timbre com o espectro de um som, mesmo não se sabendo os mecanismos cognitivos envolvidos, é compreensão básica.

E.2 Eventos no tempo

Com o quadrado das amostras temporais é obtida uma medida da energia no trecho considerado. Variações abruptas de energia indicam ataques, o que pode ser informativo do ponto de vista rítmico e até como indicação de estilos ou técnicas. Os decaimentos são importantes musicalmente, mas suas detecções são secundárias para a análise. Dados os ataques ao longo da duração, é capital e capcioso (difícil) adequá-los a alguma métrica. Para detecção de ataques de instrumentos diferentes no mesmo áudio, pode-se, por exemplo, realizar a separação do áudio por faixas do espectro ou procurar um espectro já conhecido.

E.3 Análise de partituras

As partituras são registros musicais que usam símbolos consagrados. A extração de características musicais a partir das partituras, é de uso musicológico e classificatório. Estão em uso diferentes formatos para a utilização computacional destas informações (como o Kern) e, como exemplo de uso, pode-se obter a sequência de intervalos de altura de linhas melódicas que percorrem toda a música ou as durações relativas de suas notas.

APÊNDICE F – Música digital em domínios não digitais

As tecnologias expostas neste trabalho tiveram origem em usos reais, advindas de práticas culturais e voltadas para repercussões sociais. Este material humano e artístico é o assunto deste capítulo. Pode-se dividir estes resultados em: experimentos abertos em áudio, música em tempo real, música em tempo diferido, música na matéria e repercussões no tecido social. Estes tópicos são tratados separadamente abaixo em termos do que é mantido *online*. Estas atividades extrapolam os usos musicais, especialmente devido às finalidades pedagógicas e sociais das práticas audiovisuais. Assim, por completude, alguns materiais didáticos e ferramentas *web* são descritas. Todos os desenvolvimentos desta dissertação estão em repositórios abertos.(70)

F.1 Experimentos abertos em áudio: LADSPAs, Wavelets e Redes Complexas

O código computacional feito para a arte sonora, incluindo a musical, manifesta-se como cultura pois é fruto de práticas espontâneas, diárias e coletivas.¹ A seguir estão alguns exemplos de resultados destas incidências culturais, especificamente em áudio e música.

¹ As chamadas culturas biopunk, ciberpunk, cipherpunk, hacker, digital e outras mais, possibilitadas pelos desenvolvimentos em telecomunicação, dizem respeito em menor ou maior grau à produção de código computacional como uma manifestação cultural. As elaborações de conceitos e ferramentas voltados para o compartilhamento nestas culturas estão na gênese do que se entende como 'Cultura Livre'.

Plugins LADSPA e lv2

LADSPA (Linux Audio Developers Simple Plugin API) é a API livre de *plugins* de áudio até a presente data. A última versão é a 1.1, em uso corrente até hoje e é de 2002 segundo os arquivos de cabeçalho dos códigos relacionados. O LADSPA *version 2*, abreviado lv2, é uma segunda versão do protocolo também estabelecida e estável, mas extensível, motivo pelo qual está sempre em desenvolvimento.

Na síntese, para maior qualidade, pode-se fazer a recomposição a partir do espectro desejado, como na seção 2.2.4. Menos purista porém mais eficiente é usar uma amostra curta do ruído e reproduzi-la indefinidamente através de *cross-fades*. Esta eficiência é desejada em um *plugin*, pois seus usos comuns incluem aplicações em tempo real.

Através deste trabalho, foi disponibilizado um pacote de plugins lv2 que sintetizam os ruídos branco, azul, violeta, rosa e marrom. Os códigos em C e a implementação como plugin lv2 estão no repositório git como pode ser visto em *online*².

Já na remoção de ruídos as abordagens são as mais variadas e extremamente dependentes da aplicação. A complexidade dos algoritmos atinge níveis de especialidade em processamento de sinais que já mereceram trabalhos dedicados. Adiante está a implementação de um remove-dor de ruído 'Hum'. Este ruído é geralmente causado pela corrente alternada que alimenta os equipamentos utilizados.

A remoção de ruído *Hum* é baseada em uma sequência de filtros nó rejeita banda (veja subseção 2.2.3) idealmente dispostos nos harmônicos da frequência de oscilação da corrente elétrica. Cada um destes filtros deve permitir ajustes finos no fator de qualidade e também na frequência central pois trata-se de um sistema real passível dos mais diversos efeitos de distorção do modelo ideal. O código C/C++ e a implementação como *plugin* está no mesmo repositório

² Este link é o *README* do repositório *git* com os códigos dos plugins: <http://labmacambira.git.sourceforge.net/git/gitweb.cgi?p=labmacambira/LV2;a=summary>.

disponibilizado para a síntese de ruídos.

Wavelets

Alguns dos experimentos em *wavelets* praticados estão em Audioexperiments.(71) Uma destas investidas superou em muito um experimento e constitui um protocolo de compactação de áudio desenvolvido com o Prof. Rafael Santos Mendes na FEEC/UNICAMP como consequência do mestrado realizado por André Luvizzoto sob orientação do mesmo professor. O método está bem descrito no artigo desenvolvido. Em resumo, o método consiste em particionar o áudio com sobreposições para contemplar aplicações reais sem as distorções de borda da decomposição e recomposição. Em seguida, o áudio é decomposto em árvore *Wavelet Packet*. Cada folha é então ordenada, os coeficientes com baixa energia eliminados e um polinômio encontrado pelo método dos mínimos quadrados é usado para representar os coeficientes restantes. Detalhes de escrita e leitura dos polinômios e das permutações completam o protocolo. Aplicações para *stream* de voz em tempo real foram vislumbradas e testes adicionais propostos.(72)

Redes Complexas

Com o fim de detectar nuances emocionais no uso da linguagem natural, foram feitos alguns trabalhos em análise de fala e de texto, ambos focados no uso de redes complexas. O primeiro trabalho de processamento de texto foi apresentado na ACL de 2010, em Upsala, na Suécia, pelo Prof. Thiago Pardo, atual presidente da ACL.(67–69)

Os trabalhos focaram em distinguir polaridade em excertos de textos e de fala. Nos textos, a rede é formada a partir da sequência de palavras, em que cada palavra é um vértice e a sucessão imediata cria uma aresta. O pré-processamento dos textos consiste na lematização e na retiradas das *stop-words* e da pontuação. Nas falas, as bandas de frequência utilizadas são vértices e a transição entre uma banda e outra é uma aresta. Ambas as redes possuem arestas com peso e

podem ser utilizadas como direcionais ou não. Os trabalhos consistiam em formar estas redes com os bancos de dados correlatos, extrair medidas e aplicar reconhecimento de padrão. Ambos os estudos não fazem uso do conteúdo semântico, mas sim topológico dos sinais recebidos, e chegam a mais 80% de cobertura ou precisão em alguns casos, podendo contribuir com métodos tradicionais que fazem uso de redes semânticas, por exemplo.

F.2 Áudio e música

Este capítulo está dividido em 4 partes: música em tempo diferido (ou seja, que não é realizada em tempo real), música em tempo real, música na matéria (suporte físico em *hardware*) e música no tecido social (mobilizações humanas). O texto se baseia em exemplos reais da prática musical através do código, com aplicações e em utilização pelo autor, membros do LabMacambira.sf.net, parceiros, colaboradores e por usuários eventuais das naturezas mais diversas.

F.2.1 Música em tempo diferido: *minimum-fi* e FIGGUS

'I also find that intelligent people always respect the intelligence needed to construct a simple structure in a clear way that really works.'

Tom Johnson

A realização musical em tempo diferido é o paradigma inicial da música computacional, iniciada com o *Music V*, depois com o *CSound*. Pode-se dizer que até hoje é a forma como compositores usualmente pensam a música: concebendo e escrevendo as estruturas que depois são executadas por instrumentistas ou aparelhos eletrônicos. Assim como a composição instrumental permite alguns refinamentos estruturais não presentes na improvisação instrumental, a realização musical em tempo diferido permite um detalhamento maior dos procedimentos do que a realização em tempo real.

O som musical pode ser caracterizado fisicamente, tornando possível a sua síntese digital. Estes sons e a organização deles são objeto do Capítulo 2 da presente dissertação. A utilização destes sons de forma direta, utilizando somente os recursos básicos de justaposição e sobreposição, está na pequena peça *minimum-fi*. Já o FIGGUS (FInite Groups in Granular and Unit Synthesis), utiliza os princípios do *minimum-fi* e utiliza simetrias através de permutações e de Teoria de Grupos, para a composição de músicas. O FIGGUS gera um EP³ com um único comando. Este é o *PPEPPS*⁴, como veremos a seguir.

Desta forma, esta seção exemplifica e explicita - através de dois exemplos reais - o uso de código para a síntese *musical*, desde as amostras relativas a uma nota com dada frequência, amplitude e timbre, até a confecção de uma ferramenta derivada, que já incorpora propostas musicais e estruturas mais elaboradas.

Minimum-fi

Existe uma perene matiz estética e também tecnológica dedicada a realizar uma dada tarefa com *o mínimo* necessário. Na música, esta matiz é igualmente presente e decorre em grande parte do princípio de unidade e coerência⁵. Em código computacional, a empreitada para manifestar este princípio ele próprio de forma mínima resultou no *minimum-fi.py*, código Python em um único arquivo curto que sintetiza a música segundo as estruturas especificadas em linha. Na versão atual, de 2012 mas adiantada em 2011, os algoritmos em Python propriamente ditos somam 53 linhas e incluem apenas 5 funções. Com estas funções, estruturas musicais são criadas padrão a padrão, nota a nota, amostra por amostra. Na prática, e em linguagem cotidiana, as notas formam blocos e estruturas hierarquicamente superiores.

³ *Extended Play*, um álbum musical maior que um single mas menor que um LP (*Long Play*) inteiro.

⁴ Pure Python EP: Projeto Solvente.

⁵ Este princípio tanto é fundamental que as escolas musicais possuem técnicas específicas, músicas possuem suas próprias convenções mantidas por toda a sua duração, os arcos mantêm características, enfim, podemos até mesmo concluir que as simetrias definem o escopo musical.

Os princípios, bastante simples, são⁶:

- Deve-se ter um mecanismo de síntese sonora que possibilite a geração de unidades sonoras com diferentes timbres, controle sobre a frequência fundamental, duração e volume, como especificado na seção 2.1.
- Deve-se ser capaz de construir séries de unidades, sejam sobrepostas (e.g. acordes) ou justapostas (e.g. melodias).

Para o primeiro item, presta-se o procedimento de busca em tabelas/vetores com formas de ondas em alta resolução, chamado *lookup table*. O procedimento é barato computacionalmente, com resultados diversificados e tidos como de alta qualidade pois não acrescenta ruídos relevantes ao sinal. Uma descrição do procedimento está na subseção 2.2.1.

Através da utilização do *lookup* sucessivo (procura-se um valor na primeira tabela e este valor indica o valor na segunda tabela a ser utilizado), executa-se um *waveshaping*. Este procedimento é bastante apreciado pela simplicidade e eficácia na síntese de timbres diversos e ricos em harmônicos e evolução temporal. Embora uma explicação exaustiva do *waveshaping* fuja ao escopo deste trabalho, este método se caracteriza pela aplicação de uma função não linear ao sinal de entrada. Neste caso o sinal de entrada é gerado pela primeira busca, a função não linear aplicada é a segunda busca, na outra tabela. Existem muitas formas de se executar um *waveshaping*, a seguir segue o que usamos no *minimum-fi*, que se sustenta principalmente por ser leve e simples:

Waveshaping com consultas sucessivas a tabelas

```

1 # T, incremento, n_amostras e dur definidos como no script anterior
2 ap=0.0
3 amostras=[0] * n_amostras
4 for i in xrange(n_amostras): # n_de_amostras = dur * 44100

```

⁶ Osvaldo Lacerda, em seu livro *Compêndio de Teoria Elementar da Música* fala das propriedades do som musical e sua organização de forma condizente.

```

5     foo=func[int(ap)] # foo entre [-1,1]
6     bar=( bar + (T/2) )*(T/2) # centrando no meio da tabela e ampliando o âmbito
7     amostras[i] = func2[int(bar)]
8     ap = (SI + ap)%T

```

O segundo item - dos dois princípios expostos sobre o *minimum-fi* - presta-se à discretização do espaço musical. Unidades como batidas e notas tornam mais eficiente a comunicação pois a quantidade de estruturas sugeridas é maior e as estruturas são mais explícitas no discreto do que no contínuo. Roederer aponta que as próprias notas dos instrumentos musicais são um reflexo de que é mais eficiente o uso do discreto do que do contínuo para a geração de estruturas musicais.(3)

De fato, unidades bem definidas se mostram úteis na prática musical para fazer sequências de unidades. Quando as unidades são notas com frequência definida, as sequências de unidades justapostas no tempo tendem a ser compreendidas como melodias ou linhas melódicas. As sequências sobrepostas no tempo são comumente compreendidas como acordes, mas podem ser tidas simplesmente como sobreposições circunstanciais de duas ou mais linhas melódicas⁷.(11)

As duas construções básicas explicitadas - baseadas na dicotomia melodia/harmonia, horizontalidade/verticalidade, justaposição/sobreposição - são realizadas através das funções *fazSequencia* e *fazAcorde* no *minimum-fi*. Vale notar elas são absolutamente equivalentes em uma análise puramente conceitual, i.e. uma delas pode ser omitida sem perda das possibilidades musicais. Isso fica particularmente óbvio quando se nota que os procedimentos de mixagem e concatenação são plenamente capazes de realizar o que estas funções realizam. Aliás, as funções nada mais são do que usos típicos e quase caricatos destes procedimentos: no *fazAcorde* a mixagem sobrepõe no tempo todas as unidades, no *fazSequencia* as unidades são todas justapostas no tempo.

Como pode-se notar a seguir, as sequências de notas e os acordes, em última instância, são usos

⁷ A música do século XX apresentou diversos modelos teóricos que quebram com este entendimento simplificado sobre a música, suas unidades básicas e estruturas relacionadas

específicos das 2 funções de síntese sonora implementadas: *lookup* e *lookupcruz*.

Realização de Sequências de Notas

```

1  # Note as variáveis que são fixadas de antemão
2  def fazSequencia(notas, f0, fator, dur_nota, dur_silencio,\
3                      funcao1, funcao2=0):
4      seq=[] # a sequência é iniciada vazia
5      for nota in notas: # para cada nota
6          # a frequência é a fundamental vezes 2
7          # elevado ao número de semitons:
8          freq=f0*fator**nota
9          if funcao2: # caso a funcao2 seja especificada
10             # fazer o waveshaping:
11             amostras=lookupcruz(funcao1,funcao2,dur_nota,freq)
12         else:
13             # ou faz com a forma de onda simples:
14             amostras=lookup(funcao1,dur_nota,freq)
15             # soma o silêncio
16             amostras += [0]*int(taxa_amostragem*dur_silencio)
17             seq += amostras # concatena na sequência principal
18     # normalização em 2 passos:
19     smin=min(seq); smax=max(seq)
20     seq= [(-.5+(i-smin)/(smax-smin)) for i in seq]
21     return seq # retorna a sequência pronta

```

Realização de Acordes de Notas

```

1  def fazAcorde(notas, f0, fator, dur_nota, dur_silencio,\
2                      funcao1, funcao2=0):
3      seq=[]
4      for nota in notas:
5          freq=f0*fator**nota
6          if funcao2:
7              amostras=lookupcruz(funcao1,funcao2,dur_nota,freq)
8          else:
9              amostras=lookup(funcao1,dur_nota,freq)
10             amostras=amostras+[0]*int(dur_silencio*taxa_amostragem)
11             # para sobrepor unidades de tamanhos diferentes
12             # é conveniente uma função auxiliar
13             seq=somador(seq,amostras)
14     # normalização em duas linhas:
15     smin=min(seq); smax=max(seq)
16     seq=[(-.5+(i-smin)/(smax-smin)) for i in seq]
17     return seq

```

A última das cinco funções utilizadas é uma soma amostra a amostra de dois sons. Para isso, é necessário completar com zeros a sequência com o menor número de amostras para somar rapidamente:

Somador (função auxiliar)

```

1 def somador(seq1,seq2):
2     # Basicamente adiciona zeros na sequencia menor para somar rapidamente
3     diff=len(seq1)-len(seq2)
4     if diff > 0:
5         seq2+=[0]*diff
6     elif diff<0:
7         seq1+=[0]*-diff
8
9     seq=[(i+j) for i,j in zip(seq1,seq2)]
10    return seq

```

Com isso o básico está coberto e o foco vai para a criação de estruturas. Por exemplo: pode-se criar as escalas completamente simétricas na oitava cromática, escalas diatônicas ou até microtonais, como as descritas na subseção 2.3.1.

O estabelecimento de pequenas sequências é bastante útil para reutilizações, variações e geração de materiais derivados, por exemplo:

Sequências diversas

```

1 # padrão rítmico e variações
2 b1=[0]*4
3 b2=[0]*7+[7]
4 b3=[0,7,0,1]
5 b3_2=[0,7,8,7]
6 b4=[0,7,7,7]
7 b5=[0,7]
8 b6=[0,19,7,12]
9 b7=[0,-1,0,-1]
10 b8=[0,-1,7,-1]
11 b9=[0,1,0,-1]
12 b10=[0,7,6,7]
13
14 # Série utilizada por Anton Webern
15 # em suas Variações Op.30
16 c=[0,1,4,3,2,5,6,9,8,7,10,11]
17
18 # Sequência em mixolídio comum, por exemplo,
19 # Em algumas manifestações do folclore nordestino:
20 d=[0,4,7,10,9,7,10]

```

O passo seguinte é sintetizar, mixar e concatenar para a obtenção de sequências musicais. A síntese de sequências e acordes pode ser feita desta forma:

Sintetizando sequências e acordes

```

1  # inicializamos as variáveis:
2  notas=[1,4,5,6,7]
3  fundamental=200
4  fator=2**(1/12.)
5  dur_nota=0.5
6  dur_silencio=0.1
7  onda1=senoide
8  onda2=dente_de_serra
9  # realização da sequência com senóides:
10 sequencia = fazSequencia( notas,
11                             fundamental,
12                             fator,
13                             dur_nota,
14                             dur_silencio,
15                             onda1 )
16 # realização do acorde com waveshaping
17 acorde = fazAcorde( notas
18                     fundamental,
19                     fator,
20                     dur_nota,
21                     dur_silencio,
22                     onda1,
23                     onda2 )

```

Já a síntese de estruturas compostas (e.g. sequências de acordes e sobreposição de linhas melódicas), é feita com os recursos usuais da linguagem: listas em Python e vetores Numpy (mais eficientes em tempo de execução e também na simplicidade do código). Estes procedimentos são praticamente os mesmos se implementados em Scilab, C/C++, Javascript, PHP, etc.

A seguir, para fins didáticos, está a construção de acordes periódicos em python puro⁸:

Acordes periódicos

```

1  # aproveitando a variavel
2  # acorde do script anterior
3  dur = dur_nota + dur_silencio
4  silencio_1_tempo = [0]*int((dur)*taxa_amostragem)
5
6  # adicionando o acorde a cada 2 tempos 8 vezes
7  # ou seja, por 16 tempos, durante 4 compassos de 4
8  # a música terá o acorde sedo tocado a cada 2 tempos:
9  ac += (acorde+silencio_1_tempo)*8

```

⁸ Uma implementação não didática destas três linhas de código pode ser feita em uma só linha.

Em posse destas sequências, acordes e recursos da linguagem, são formadas estruturas hierarquicamente superiores através da concatenação de estruturas, da mixagem de estruturas, e da amplificação (ou atenuação) seletiva das mesmas:

Amplificação e mixagem

```

1  #-# De posse das notas, fundamental, fator, durações e ondas #-#
2  # sequência do restrogrado das notas com waveshapping:
3  sequencial = fazSequencia(nt[::-1], f, fator, d, s, senoide, dente)
4  # sequência original das notas uma quinta justa acima,
5  # com dente de serra:
6  sequencia2 = fazSequencia(nt, f*(3/2.), fator, d, s, dente)
7
8  #-# Procedimentos básicos #-#
9  # concatenação
10 seq = sequencial+sequencia2
11 # amplificação e concatenação
12 seq += [1.5*i for i in sequencial]
13 # mixagem e concatenação
14 seq += [(i+j) for i,j in zip(sequencial,sequencia2)]
15 # mixando igual acima, mas quando os comprimentos
16 # dos vetores não são os mesmos
17 seq += somador(seq,sequencia2)

```

Neste ponto, basta criar músicas e sequências de interesse estético ou para pesquisa. Os encaideamentos dependem de intensões estéticas, entendimentos musicais e estruturas abstratas que mantêm a coerência e o interesse em uma peça musical. Para o leitor mais interessado, recomendamos uma visita ao capítulo 2 e à *toolbox* MASSA, onde pode-se encontrar o próprio script *minimum-fi.py* e outros experimentos. A seguir utilizamos esta base apresentada para sintetizar estruturas musicais e então um EP.

FIGGUS: FInite Groups in Granular and Unit Synthesis

Nesta subseção há um foco nas estruturas musicais. São elas, inseridas em um momento histórico e executadas por instrumentos específicos com as técnicas de época, que constituem em grande parte uma linguagem musical e músicas propriamente ditas. O FIGGUS constitui uma técnica composicional manifesta em software como ferramenta de síntese de estruturas musi-

cais. Esta técnica consiste na utilização de estruturas matemáticas que representam simetrias para a organização de materiais musicais.

O FIGGUS foi iniciado em 2006 com o físico-matemático Prof. Adolfo Maia Junior (do IMECC e NICS, ambos da UNICAMP) - bem anterior ao nascimento do *minimum-fi* - para tratar de simetrias na música com vistas à composição musical através de métodos matemáticos⁹. Mais especificamente, a proposta gerou um programa de síntese granular e de estruturas musicais através de Grupos Algébricos. O nome dado foi FIGGUS, sigla de FInite Groups in Granular and Unit Synthesis¹⁰.

Na atual reescrita, embora ainda bastante atrás do FIGGUS original quanto à interface gráfica, a ferramenta opera diretamente em Python puro, com as bibliotecas imbutidas por padrão. Isso permite com que o FIGGUS sintetize todo um EP usando somente os comandos:

Utilizando o FIGGUS para Sintetizar um EP

```
1 $ git clone git://labmacambira.git.sourceforge.net/gitroot\
2                               /labmacambira/FIGGUS
3 $ cd FIGGUS
4 $ python RUNME_make_now_an_EP_MUSIC.py
```

Desta forma, a ferramenta é simples de ser usada para experimentações e implementações adicionais. Outro uso planejado para o FIGGUS é a síntese de timbres e amálgamas sonoros através da Síntese Granular. Embora o foco atual seja outro, cabe algumas breves palavras sobre o assunto.

A Síntese Granular é uma área bem estabelecida tanto na acústica quanto na Computação Musical e se caracteriza pela geração de sons bastante curtos e em quantidade massiva. Tipicamente,

⁹ Duas iniciações científicas trataram do assunto.

¹⁰ Também foi usado o nome FIGGS (FInite Group in Granular Synthesis) dado que o termo *unit synthesis* não é usual na literatura. Posteriormente o primeiro autor deste trabalho recorreu novamente ao uso do nome FIGGUS. Isso foi motivado pela utilidade da técnica para síntese de estruturas musicais, que, nos usos que o FIGGUS teve, foi maior do que a utilidade para síntese de amálgamas sonoros tipicamente resultantes da síntese granular

os sons possuem entre 5 e 40 milissegundos e a quantidade destes *microsons*¹¹ pode chegar a milhares por segundo. O tratamento específico da síntese granular foge ao escopo deste trabalho. O leitor interessado pode consultar os artigos produzidos sobre Síntese Granular e Teoria de Grupos.(60, 61) Desta forma, o texto a seguir concentra-se em Grupos Finitos para a síntese de estruturas musicais.

Nas artes é de comum conhecimento o papel absolutamente central que as simetrias possuem. Na música, para citar somente alguns exemplos simples, há os numerosos estudos de simetrias na música de J. S. Bach, os jogos de dados de Mozart e os usos recorrentes da proporção áurea na música de Béla Bartók. Matematicamente, as *simetrias* são descritas por Grupos, e estes são definidos como um conjunto (seja G) munido de uma operação (seja \bullet), formando um grupo (G, \bullet) satisfazendo as propriedades descritas na subseção 2.3.7. No FIGGUS, o grão ou unidade sonora é uma classe que possui apenas os seguintes atributos: duração (segundos), frequência (Hz), timbre (identificador para usar mediante implementações convenientes), intensidade (pico $\in [0, 1]$), e duração dos fades (in e out em segundos). O FIGGUS funciona com base em uma sequência de grãos especificada inicialmente e na qual operam as permutações. Aos grãos em sequência são aplicadas permutações. Para isso é bastante conveniente representar as permutações em classes próprias. A classe do padrão de permutação possui também um período de aplicação da permutação, ou seja, de quantas em quantas leituras da sequência a permutação é aplicada.

Em posse dos grãos, da sequência, das permutações e do padrão de permutações, pode-se realizar a estrutura musical em si. Basta adicionalmente especificar o número desejado de iterações da sequência. Com isso, a sequência de grãos é lida um número de vezes, aplicando as permutações na sequência de grãos segundo o padrão especificado, de forma a resultar em uma sequência musical. Note que se a permutação usar menos elementos que a sequência possui,

¹¹ Grãos sonoros e microsons são jargões equivalentes típicos da síntese granular. São usados para indicar sons com durações bastante curtas, como assinalado no texto.

alguns destes elementos ficarão estáticos nas iterações da sequência no padrão sonoro.

A partir das representações abstratas do padrão musical, são feitos os vetores sonoros da representação digital da música a ser realizada e pode-se escrever um arquivo de áudio propriamente dito. O mais conveniente, neste caso, é escrever um arquivo PCM (Pulse Code Modulation) em algum padrão amplamente utilizado e reconhecido. Ambos WAV e AIFF satisfazem estes requisitos. Mais especificamente, o padrão de CDs é WAV com 44100 amostras por segundo e 16 bits por amostra. As amostras dos vetores sonoros são calculados, por conveniência e convenção, no âmbito $[-1, 1]$ e precisam ser normalizados para o âmbito $[-32767, 32768]$ e truncados em números inteiros. Depois disso devem ser escritos em um arquivo com os *bits* como na convenção da linguagem C/C++. A biblioteca *struct* cuida dessa escrita do inteiro no formato correto, e a biblioteca *wave* escreve o cabeçalho no formato WAV adequado. Assim, a classe de escrita do vetor sonoro em arquivo comum fica assim:

Escrita do Vetor Sonoro em Arquivo WAV

```

1 import wave, struct
2 class IOUtils:
3     def recordPattern(self, filename, pattern):
4         self.recordFile(pattern.sonic_vector, [], "sound.wav", pattern.SR)
5
6     def recordFile(self, sonic_vector=[], sonic_vector2=[],
7                   filename="sound.wav", samplerate=44100):
8         sound = wave.open(filename, 'w')
9         sound.setframerate(samplerate)
10
11        sound.setsampwidth(2) # Always 16bit/sample (2 bytes)
12        if not sonic_vector2:
13            sound.setnchannels(1) # Mono
14            sonic_vector=self.boundVector(sonic_vector)
15
16            sonic_vector=[i*(2**15-1) for i in sonic_vector]
17            sound.writeframes(struct.pack('h'*len(sonic_vector),
18                                         *[int(i) for i in sonic_vector]))
19        else:
20            sound.setnchannels(2) # stereo
21            sonic_vector=self.boundVector(sonic_vector)
22            sonic_vector2=self.boundVector(sonic_vector2)
23
24            sonic_vector=[i*(2**15-1) for i in sonic_vector]
25            sonic_vector2=[i*(2**15-1) for i in sonic_vector2]

```

```

26         SV=[]
27         for i,j in zip(sonic_vector,sonic_vector2):
28             SV.extend((i,j))
29         sound.writeframes(struct.pack('h'*len(SV),*SV))
30     sound.close()
31
32     def boundVector(self,vector):
33         """Bound vector in the [-1,1] interval"""
34         svmin=min(vector)
35         svmax=max(vector)
36         ambit=svmax-svmin
37         if svmax>1 or svmin<-1:
38             if svmax-svmin > 2:
39                 i=0
40                 for sample in vector:
41                     new_sample=(sample-svmin)/ambit # results in [0,1]
42                     new_sample=new_sample*2-1 # results in [-1,1]
43                     vector[i]=new_sample
44                     i+=1
45             elif svmax > 1:
46                 offset=svmax-ambit/2
47                 i=0
48                 for sample in vector:
49                     new_sample=sample - offset
50                     vector[i]=new_sample
51                     i+=1
52             elif svmin < -1:
53                 offset=-svmin -ambit/2
54                 i=0
55                 for sample in vector:
56                     new_sample=sample + offset
57                     vector[i] = new_sample
58                     i+=1
59         return vector

```

F.2.2 Música em tempo real: *Livecoding* e ABeatTracker (ABT)

Tornou-se usual a síntese sonora em tempo real mesmo em *laptops* populares. Assim, surgiram linguagens de domínio específico, para o áudio e a música, em sua maioria dedicadas - ou ao menos capacitadas - para interação em tempo de execução como *Puredata*, *SuperCollider*, *ChucK* e *ixilang*. Todos estes são exemplos de linguagens largamente utilizadas para a composição musical e síntese sonora em tempo real. Em outras palavras, estas linguagens possibilitam

que o usuário ouça o resultante sonoro do código utilizado e altere o código com resultados imediatos no processamento e sonoridades produzidas.

A exploração estética destas ferramentas em performances musicais é a proposta do *Livecoding*. Esta prática de *performance* se dedica especialmente à escrita de código em tempo real com vistas à projeção visual dos códigos enquanto são escritos, junto à projeção sonora que produzem. O ABeatTracker ainda pode ser considerado livecoding, mas já é um intermediário, com traços de um aplicativo ou ferramenta, embora os comandos sejam acessados pela escrita. É uma linguagem com macros para execução de ritmos através de *samples* em conjunto com instrumentos tradicionais e outras fontes sonoras/musicais externas.

Livecoding

Recentemente, grupos de ponta em música experimental estão desenvolvendo apresentações musicais públicas baseadas na escrita de código ao vivo. Este é um fenômeno cultural e estético, do qual vale ressaltar os grupos pioneiros *Slub* e *Benoît and the Mandelbrots* assim como as 'orquestras de *laptops*' *PLOrk*, *SLOrk* e *DubLOrk*¹². Usualmente, o código é projetado para que a audiência possa ver o que está sendo escrito, no ritmo em que se escreve, e se projeta também o resultante sonoro por autôfalantes.

Estas são motivações presentes em diferentes grupos¹³, embora não necessariamente ou da mesmíssima forma:

- A apresentação musical, com o uso do computador como instrumento, carece de recursos performáticos visuais dos instrumentos tradicionais. Os gestos são por demais discretos e a concentração do *performer*/instrumentista é bastante focada na tela do computador. Recursos performáticos são, portanto, preciosíssimos.

¹² Para maiores detalhes, veja: <http://toplap.org>.

¹³ Vide 'Manifesto Live coding' em <http://toplap.org/wiki/ManifestoDraft>.

- O *feedback* auditivo do código projetado permite que o espectador infira significados dos códigos. Este recurso do *livecoding* é usado para desmistificar a programação de computadores e sua aplicação em computação musical, comumente considerada intangível.
- O código em si é um recurso poderosíssimo que permite ao usuário controlar os sons produzidos amostra por amostra ou em escalas maiores de tempo, como notas, compassos, fraseados inteiros ou mesmo em escalas maiores de tempo, como minutos, horas, dias e semanas.
- O compartilhamento do código é usual, leve e eficiente como entrega de tecnologia valiosa para a proposta estética. Isso motiva não só o programador a aplicar seus conhecimentos na música, mas também o músico a adentrar o uso de linguagens de programação para expressão de suas ideias musicais.

Desta forma, foi iniciada em 2011 uma linha de atuação em *livecoding* com a criação do duo *FooBar* composto por Vilson Vieira e o autor deste escrito. Este duo se desenvolveu no trio *FooBarBaz/Variáveis Metasintáticas* composto pelo duo e por Gilson Beck, este atuante na mesa de som e usando detecção de cor no *laptop* para incrementar a apresentação. Este trio realizou uma *performance* no *V Festival Contato*. É interessante ressaltar que, até onde se sabe, essa foi a primeira apresentação de *livecoding* no Brasil, e a maior apresentação internacional em tamanho de platéia, estima-se que entre 3 e 5 mil pessoas estavam presentes.

Foi usada a linguagem *ChucK* por apresentar os recursos que o duo inicial considerou mais apropriados, embora de forma alguma isso seja consensual na prática atual de *livecoding*. Além disso, a apresentação contou com recursos adicionais para agregar interesse, como a utilização do *cowsay* para enviar mensagens enquanto se desenrolava a música. Em especial, as trajetórias de um ponto branco no fundo do código sugeria o sono REM como uma experiência coletiva de alteração do estado de consciência. Estes recursos podem ser vistos em uso nos vídeos

demonstrativos¹⁴ As linhas do *cowsay* e o *script* em linguagem Processing relacionados, assim como maiores detalhes sobre o duo e fotos da apresentação podem ser vistos em <http://wiki.nosdigitais.teia.org.br/FooBarBaz>. Estavam presentes no palco também outros membros do labMacambira.sourceforge.net, davam suporte em gravações e outros auxílios: Ricardo Fabbri, Alexandre 'Bzum', Daniel Penalva e Ivan Marin.

Especificamente sobre a prática do *livecoding*, duas pessoas executaram *scripts* em *Chuck* simultaneamente. A saber, Vilson Vieira executou ritmos, batidas bastante marcadas que serviam como base. Renato Fabbri, autor deste texto, executava linhas fluídas quasi-melódicas que formavam arcos maiores. Havia também interlúdios em que ambos se revejavam com músicas curtas e inusitadas, como em um duelo. A mixagem e espacialização dos dois canais de áudio foram controladas por um *patch* em Puredata que permitia o *cross-fading* entre os canais através da detecção do movimento das mãos. Isso foi possível através do objeto em Puredata/GEM chamado *color classify*, criado por Ricardo Fabbri e posto em uso por Gilson Beck durante a apresentação. Esse mesmo objeto foi utilizado no instrumento AirHackTable, discutido na seção F.2.3.

A prática de *livecoding* requer o uso de instruções curtas, que incentivem o improviso musical através do código. Ambos os *livecoders* usaram bons editores de texto para a escrita dos *scripts* em *Chuck*: Vim e Emacs. Renato utilizou aspectos estruturais de alto nível, arcos longos e estruturas melódicas e minimalistas, como os que seguem:

Interface para controle de parâmetros em tempo real

```

1 // Em Chuck, soltar estas duas linhas em alguma musica
2 // bem ritmada e deixar os padroes aa vontade com este
3 // arco maior:
4
5 // crescimento continuo
6 SinOsc s1 => Gain g => g => s1 => dac;
7 2 => s1.sync;
```

¹⁴ Disponíveis em <http://vimeo.com/33012735>, <http://vimeo.com/33018740>, <http://vimeo.com/33019291>, <http://vimeo.com/33025717> e <http://vimeo.com/33025913>.

```

8 while(second => now);
9
10 // vento
11 Noise s1 => Gain g1 => g1 => SinOsc s2 => dac;
12 2 => s2.sync;
13 while(second => now);

```

Ao mesmo tempo, Vilson fez uso de scripts para facilmente especificar a posição do arquivo de áudio em que deve ser iniciada a execução, assim como a velocidade de reprodução do áudio, duração e amplitude. Uma interface simples permitiu a alteração rápida desses parâmetros em tempo de execução.

Interface para controle de parâmetros em tempo real

```

1 ["samples/fx/s20.wav"] @=> Foo.name;
2 [0.] @=> Foo.prop;
3 [.25, .15] @=> Foo.rate;
4 [2., 1., 1., 4.] @=> Foo.du;
5 [.8] @=> Foo.gain;

```

Classe para *sampling* de arquivos de áudio

```

1 public class Foo {
2     static string name[];
3     static float prop[];
4     static float rate[];
5     static float du[];
6     static float gain[];
7 }
8
9 ["samples/fx/s22.wav"] @=> Foo.name;
10 [0.] @=> Foo.prop;
11 [1.] @=> Foo.rate;
12 [4.] @=> Foo.du;
13 [0.] @=> Foo.gain;
14
15 TimeGrid tg;
16 tg.set(1::minute/60/2, 8, 10);
17 tg.sync();
18
19 SndBuf buf => JCRev j => dac;
20 .5 => j.gain;
21 .2 => j.mix;
22
23 0 => int i;
24
25 while (true) {
26     Foo.name[0] => buf.read;

```

```

27     Math.trunc(buf.samples()*Foo.prop[i%Foo.prop.size()]) $ int => buf.pos;
28     Foo.gain[i%Foo.gain.size()] => j.gain;
29     Foo.rate[i%Foo.rate.size()] => buf.rate;
30     tg.beat*Foo.du[i%Foo.du.size()] => now;
31     i++;
32 }

```

Estes desenvolvimentos inspiraram a criação de uma linguagem específica para *livecoding*, chamada *Vivace*, que está em constante desenvolvimento e tem seu código e maiores detalhes disponíveis online de forma livre¹⁵. Já foram realizadas novas apresentações utilizando essa linguagem, a notar: Hacklab do Velho, em São Carlos (SP), Semana da Comunicação FAAP (SP), Palco UFSCar (SP) e Semana Nacional da Ciência e Tecnologia (SP). Nelas houve a predominância do apelo às mídias populares, em uma reinauguração do gênero da *pop art*: trechos de vídeos de uma novela brasileira: 'Avenida Brasil' eram rearranjados em tempo real em sincronia com linhas rítmicas e melódicas¹⁶.

Catalizadas pela aproximação de "Gera" Magela, Caleb Mascarenhas Luporini e Guilherme Lunhani as apresentações e amadurecimentos de 2012 desembocaram na escrita coletiva de códigos (acesso e controle da platéia no código que está sendo escrito), utilização de monstros e *ascii art* e sonoridades bizarras. Com isso, está sendo observado e proposto um gênero de *livecoding* batizado de '*Freakcoding*', possivelmente o primeiro subgênero de *livecoding*. Foi escrito um *manifesto* sobre o *freakcoding* e um artigo acadêmico. Estão sendo planejados mais apresentações e desenvolvimentos tecnológicos, todos motivados pelo *livecoding* e este subgênero brasileiro que está despontando.

¹⁵ Veja: <http://automata.github.com/vivace>.

¹⁶ Uma amostra está disponível *online* para ser utilizada por qualquer em qualquer computador com o navegador *Google-Chrome*: <http://pulapirata.com/skills/vivace>.

ABeatTracker (ABT)

O ABT é uma linguagem que dispara linhas rítmicas através de macros. Nestas, especifica-se as células rítmicas, amostras sonoras que são utilizadas como conteúdo sonoro destas linhas, modos de leitura destas amostras sonoras e variáveis randômicas utilizadas para execução da linha. Além disso, o ABT dispõe de variáveis globais que podem ser alteradas a qualquer momento pelo usuário também com macros, como BPM, volume, velocidade de leitura das amostras e variáveis randômicas globais (que se somam às individuais). As macros pré-estabelecidas constituem um conjunto de recursos pré-estabelecidos bem definido, o que contrasta com a ideia de uma 'linguagem de programação' que tenha capacidades mais amplas. De qualquer forma, linguagens com domínios específicos não são raras e por vezes o ABT foi descrito como uma linguagem por usuários e interessados.

O ABT é um instrumento computacional essencialmente rítmico. Junto a esta proposta, está a necessidade da utilização em conjunto com outros instrumentos, externos ao ABT, ao computador em que o ABT está sendo executado e possivelmente externo com relação a qualquer computador. Para isso foi elaborado o ABD (ABeatDetector), no qual o usuário tamborila no teclado do computador os ritmos que estiver ouvindo ou imaginando para que o ABT sincronize o pulso e utilize células rítmicas relacionadas. A análise feita pelo ABD resulta em uma série de ritmos explicitados por sequências de compassos que encapsulem durações regulares do ritmo tamborilado. Estes ritmos relacionados ao tamborilar do usuário são chamados de harmônicos e podem ser selecionados prontamente para o disparo de linhas melódicas no ABT. De fato, atualmente o ABD é parte do ABT. Todo o código do ABT está disponível online junto à documentação para uso¹⁷.

¹⁷ Veja o repositório: git://labmacambira.git.sourceforge.net/labmacambira/audioArt/ABT

F.2.3 Música na matéria: EKP e AHT

Embora o foco deste trabalho seja a exploração musical através de códigos computacionais, nesta seção está uma explicação simples, clara e factual de como estas investidas transcendem o código e até mesmo a música em si. Por vezes, mais relevantes que os próprios desenvolvimentos são as mobilizações criadas nos entornos e os engajamentos. A seguir estão dois trabalhos que geraram alguma mobilização, resultando em trocas, reuniões, desenvolvimentos, pesquisas e apresentações propriamente ditas. O primeiro utiliza o estado do hardware como entrada, o segundo consiste na flutuação de origamis para a obtenção de um instrumento musical lúdico.

Emotional Kernel Panic (EKP)

Em 2008, em colaboração intensa com o CDTL (Centro de Desenvolvimento de Tecnologias Livres)¹⁸ foi lançada a ideia de utilizar o estado do sistema operacional - especialmente o kernel linux - para gerar de sons. Surge então o '*Emotional Kernel Panic*' (EKP) na parceria de Felipe Machado (então coordenador do CDTL), Ricardo Brasileiro (artista conhecedor de tecnologias livres) e o primeiro autor do presente trabalho.

Foram reconhecidas três finalidades para esta exploração do sistema operacional:

- Pedagógica, através da utilização de um sentido que não o visual para a tarefa de analisar o sistema.
- Artística, para apresentações musicais/audiovisuais.
- Monitoramento do sistema operacional, pela emissão de sons periódicos relativos à carga de processamento, memória, uso de rede, etc.

Esta empreitada se desdobrou em apresentações na Conferência Internacional de PureData

¹⁸ Uma associação civil formada e desmembrada em 2008, sediada em Recife, PE.

(2009), SESC (2009), e no III Festival Contato (2009). No Festival Internacional de Software Livre de 2010 foi apresentada uma pesquisa sobre o EKP com diversos patches feitos por Ricardo Brasileiro e Felipe Machado, assim como amadurecimentos da proposta. Uma implementação conceitual do EKP está aqui: <http://trac.assembla.com/audioexperiments/browser/ekp-base>.

AirHackTable

A AirHackTable (AHT) é um instrumento musical eletrônico controlado por origamis (dobraduras de papel), construída na forma de uma mesa. Nela, uma rede de *coolers* reciclados faz flutuar origamis de geometria e cores variadas. Os movimentos dos origamis são captados por *webcam* e interpretados em tempo real por software de processamento de imagens, gerando padrões que controlam a transformação sonora da música. Desta forma, pode-se dizer que os sons gerados refletem o vôo dos origamis de acordo com suas geometrias (que geram trajetórias de vôo características). Dito de maneira mais teórica, as estruturas fora do tempo (geometrias dos origamis) estão sendo mapeadas em estruturas temporais (trajetórias dos origamis em seus vôos), o que é um macete da música erudita desde tempos antigos.

Na versão atual da AirHacktable, cada cor (vermelho, amarelo, azul, verde, preto, ou branco) controla uma voz. Já a posição do origami na mesa controla aspectos do som. Como um exemplo, pode-se configurar a AHT de forma que, se o origami está flutuando para a esquerda, o som também se move para esquerda, se o origami está mais próximo da câmera, o volume é maior, e se está mais afastando do operador, o som se torna mais agudo.

A AHT segue a filosofia de desenvolvimento contínuo, comum ao *software* livre, que atualmente foi incorporado ao movimento chamado *hardware* livre ou aberto. A concepção inicial foi de Chico Simões (mestre de maracatu e conhecedor de tecnologias livres) e o autor deste texto, através do amadurecimento de possíveis instrumentos musicais de papel. Logo neste estágio

inicial, foram aproximados Vilson Vieira, Ricardo Fabbri, Fábio Simões e Daniel Penalva em reuniões presenciais. Destas reuniões saíram ideias diversas, escritas em um *pad* aberto¹⁹, e, por fim, a AHT em si.

A primeira realização da mesa se deu no V Festival Contato, no *Espaço Macambira*, com apresentação na Teia Casa de Criação, durante o mesmo festival. A estrutura da mesa é também de papel (papelão), e foi feita por Francisco Simões por pesquisa própria. O uso do reticulado de ventoinhas foi fruto de discussões coletivas e de experimentos do coletivo labMacambira.sourceforge.net, com especial empenho do Francisco e uso de sua luthieria de percussão: *Tora Tambores*.

Em 2012 houve a aproximação de Caleb Mascarenhas Luporini e "Gera" Magela nesta frente de atuação. Foram feitas oficinas no SESC Pinheiros e SESC Belenzinho e apresentação no AVAV²⁰, todas com o foco na construção e uso da AHT.

F.2.4 Música no tecido social: Sabrina Kawahara, Audioexperiments, EstudioLivre.org, CDTL, juntaDados.org, Devolts.org, MSST, Lab-Macambira.sf.net

A singularidade do cruzamento artístico, cultural e tecnológico dos trabalhos expostos também é desenvolvida por diferentes grupos. Os propósitos de compartilhamento e apropriação tecnológica estão no cerne destas comunidades, de forma que as investidas naturalmente tomam rumos engajados socialmente. Em especial, o empoderamento civil e a criação de um patrimônio tecnológico da humanidade são consequências imediatas das posturas de compartilhamento e apropriação citados. Segue uma lista parcial dos grupos mais relevantes para o presente trabalho, tanto pela influência que estes grupos/redes tiveram na formação e atuação do primeiro autor quanto pela ressonância que os trabalhos aqui expostos encontram.

¹⁹ Veja: <http://pontaopad.me/origami-sensores>.

²⁰ Evento mensal que ocorre na cidade São Paulo: AudioVisual Ao Vivo.

- Sabrina Kawahara: formado pelo autor deste texto, Guilherme Lunhani, Guilherme Rebecchi, Israel Laurindo, Rodrigo Felício e Otávio Martigli, este grupo dedicou-se à realização de apresentações públicas com peças musicais, bem como à escrita de peças e textos através de dinâmicas coletivas.
- Audioexperiments (Æ): iniciado por Tiago Tavares, o autor do texto e aproximações de Daniel Pastore. O grupo se dedicou à publicação de textos didáticos e produção de código em repositórios públicos. Em destaque, o repositório no *Assembla* possui códigos em Python e C/C++ que realizam diversos experimentos em áudio, música e inteligência artificial²¹.
- Estudios Livres: são ambientes de estudo, produção e distribuição de mídias livres. Formam, em conjunto, um movimento e uma rede social brasileira, tida como única no mundo, embora o conceito seja conhecido internacionalmente²². Tanto a comunidade quanto a lista operacional e plataforma da comunidade foram essenciais nos amadurecimentos de questões relacionadas às mídias livres e ao uso de Python para áudio e música, como pode ser visto na Carta Mídias Livres e no Tutorial de Python para Áudio e Música, abordados abaixo em F.3.1 deste mesmo Apêndice.
- CDTL (Centro de Desenvolvimento de Tecnologias Livres): foi uma associação civil com base em Recife/PE, dedicada ao desenvolvimento e difusão de tecnologias livres. Esta associação proporcionou suporte para os primeiros materiais didáticos e investidas em plugins de áudio através do convênio estabelecido com o Ministério da Cultura de Pontão de Cultura Digital.
- JuntaDados.org: é um grupo com membros em todo o território brasileiro e ainda ativo. Embora tenha havido um momento de maior institucionalidade, durante o convênio de

²¹ Veja a página principal do espaço: <https://www.assembla.com/spaces/audioexperiments>.

²² Há uma plataforma aberta com capacidades de *wiki*, blog e midiateca em <http://estudiolivre.org>.

Pontão de Cultura Digital com o Ministério da Cultura e parceria com a Universidade Estadual da Bahia (UNEB), o grupo é marcado pela descentralização e autonomia dos envolvidos. O autor é membro fundador e ativo do *juntaDados*, em conjunto, especialmente, com Fabiana "Goa" Sherine e Marcelo Soares Souza. Neste grupo foram desenvolvidas ideias como o EKP, o ABT, os tutoriais de Python para Áudio e escritos socialmente engajados.

- Devolts.org: responsável por Esporos de Pesquisa e Experimentação do programa Cultura Viva, os "Jardins Devolts" cultivam listas nacionais de trocas de conhecimentos livres e cultivo de informações voltadas para música com usos especiais de *PureData*, *Arduinos* e *Python*. Este é um meio de troca rico e alternativo a alguns meios mais politizados, como a lista do *estudiolivre.org*.
- MSST (Movimento dos Sem Satélite): com um manifesto e propostas coerentes, mas nada óbvias, como a demanda específica por satélites civis, abordagens tecnomágicas, o grupo é uma instância iniciática em termos tecnológicos e conceituais.
- LabMacambira.sf.net: espaço virtual de programadores dedicados ao audiovisual e propósitos socialmente engajados. Fundado em parceria do autor deste trabalho e de Vilson Vieira, Daniel Marostegan e Carneiro e Ricardo Fabbri e de parte dos investimentos da segunda etapa do convênio de Pontão de Cultura Digital Nós Digitais, da Teia Casa de Criação. Em um primeiro momento, membros foram remunerados como mentores e como aprendizes²³. Em um segundo momento, aproximações e parcerias diversas ocorreram. Detalhes estão nas subseções seguintes de *Web* e materiais didáticos.
- Submidialogia: grupo bastante relacionado aos encontros de *'submidialogias*, ao Descen-

²³ Renato Fabbri, Ricardo Fabbri, Vilson Vieira, Marcos Mendonça e Gilson Beck mentoraram; Alexandre "Bzum" Negrão, Lucas Zambianchi, Larissa Arruda, Nivaldo Henrique Bondança, Fernando Gorodscy, Andres Martano desenvolveram como bolsistas; Danilo Shiga e Marcos Murad foram parceiros colaboradores; Francisco Simões, Fábio Simões, Caleb Mascarenhas Luporini, "Gera" Magela, Edson "Presto" Correia, Daniel Penalva se aproximaram para atividades específicas, cruciais em todo ano de 2012 já sem os investimentos do convênio.

tro e às ideias de submídia. A lista de emails do grupo possui membros de diversos dos outros grupos citados.

- Outros grupos relacionados: o MuSa foi um grupo de Joinville importante neste trabalho por ter sido fundado por Vilson Vieira, que possui presença forte no LabMacambira.sf.net e nos trabalhos aqui descritos. A Metareciclagem é uma rede engajada em apropriação tecnológica e empoderamento civil com bastante presença dos outros grupos aqui citados e muito movimentada. A lista de emails ligada ao movimento Transparência Hacker foi também de importância central como espaço focado em questões de transparência e dados abertos. A Casa de Cultura Tainã é uma entidade cultural de Campinas/SP, referência no uso de tecnologias livres e atividades socialmente engajadas, e onde muito do que está aqui descrito foi inspirado. O Hacklab do Velho foi fundado por pessoas ligadas ao labMacambira.sf.net e por moradores do alojamento velho da USP, campus de São Carlos, e acolheu diversas experimentações e articulações. Outros grupos com alguma importância foram a Casa Fora do Eixo de São Carlos, a Teia Casa de Criação, a Nuvem estação rural de experimentação tecnológica e o grupo que organiza o AVAV Audiovisual Ao Vivo, ligado ao Epicentro Cultural.

F.3 Materiais didáticos

F.3.1 Tutoriais em texto e código: python, filtros e nyquist, plugins lv2, metrics, carta mídias livres, contra-cultura digital

- **Tutorial de python para áudio e som**

Este tutorial foi apresentado parcialmente em Berlim no LAC 2007 e, desde então, foi melhorado algumas vezes. Esta primeira versão ficou resumida em forma de texto na plataforma Estúdio Livre²⁴. Um agradecimento especial para Fábio Furlanete e Marília

²⁴ <http://estudiolivre.org/python-e-som-tutorial>.

Chiozo pelas contribuições. Em 2010 a Associação Python Brasil escolheu este trabalho, então já mais maduro, para ser apresentado no Festival Internacional de *Software Livre*, em Porto Alegre. Como consequência, foi feita uma série de video-tutoriais²⁵. Os vídeos são tidos como iniciáticos em Python por diferentes pessoas ligadas ao movimento de *software livre*.

- **Tutoriais de filtros e amostragem via python**

Voltados para explicitar fundamentos de áudio digital, estes tutoriais são baseados em pequenos *scripts* escritos em Python que exploram conceitos pontuais. Pequenas explicações são dadas com o intuito de orientar a exploração inteligente destes *snippets*. *Teorema de Amostragem*: estes *scripts* executam experimentações ilustrativas com o Teorema de Nyquist e figuras. *Filtros*: além de filtros FIR e IIR simples, duas utilizações clássicas destes filtros estão implementadas de forma didática: Wavelets (FIR) e Quad (IIR). Estes códigos didáticos podem ser baixados do repositório *audioArt*²⁶.

- **Tutorial de plugins lv2**

Dadas as dificuldades que o desenvolvimento dos *plugins* de áudio apresenta, foi feito um tutorial passo a passo com *plugins* operantes em todas as etapas. Os códigos e os textos estão todos em repositório²⁷.

- **Microtutoriais Django**²⁸

Estes 'microtutoriais' são baseados nos conceitos de *scripts mínimos* e *alterações pontuais*. O primeiro conjunto de microtutoriais é dedicado a reconstruir o tutorial oficial do django de forma condensada e não prolixa. O segundo destes conjuntos é dedicado a instrumentalizar de fato o leitor com o entendimento do funcionamento dos princípios funda-

²⁵ <http://estudiolivres.org/tiki-index.php?page=Video+Tutoriais>.

²⁶ Veja <http://labmacambira.git.sourceforge.net/git/gitweb.cgi?p=labmacambira/audioArt;>.

²⁷ Veja <http://labmacambira.git.sourceforge.net/git/gitweb.cgi?p=labmacambira/lv2Tut;>.

²⁸ Veja: <https://vimeo.com/channels/labmacambira>.

mentais deste *framework*. A importância deste material reside no fato de ser uma das primeiras incidências dos *scripts* mínimos, muito usados no labMacambira.sourceforge.net para passagens de tecnologias de forma precisa.

- **Figusdevpack (FDP)**

Idealizado como um meio de interação da comunidade de Python e Música para compartilhamento de *scripts* inteiros e excertos, o FDP foi Baseado principalmente em documentação organizada sobre as práticas e as bibliotecas existentes para Python. Parte desta documentação é proposta como *scripts* que fazem uso de objetos e módulos de forma isolada, puntual. Este trabalho foi aceito na maior conferência de áudio em linux, a Linux Audio Conference de 2008 (LAC2008) e foi reativado algumas vezes pelo autor deste texto em conjunto com Vilson Vieira, Ivan Marin e outros desenvolvedores. Este projeto está documentado na plataforma do Estúdio Livre²⁹.

- **Carta mídias livres**

Texto criado em decorrência da participação do autor do presente escrito na comissão de seleção no “Prêmio Mídias Livres”, a convite do Ministério da Cultura por “notório saber”. A participação consistiu em avaliar os inscritos no Prêmio Mídias Livres e distribuir 4 milhões de reais dentre categorias regionais e nacionais. Esta carta é um documento único, deixando às claras o conceito de Mídias Livres como não aprisionadas pelo conceito de propriedade, ou seja, que priorizam a sua livre circulação e a possibilidade de geração de materiais derivados. Há o viés de priorizar processos colaborativos, comunitários e setores da sociedade menos contemplados na geração e circulação midiática³⁰.

- **Philosometrics**

²⁹ Veja a página do PDF <http://estudiolivre.org/tiki-index.php?page=fdp&highlight=fdp>, o artigo aceito no LAC2008 http://www.estudiolivre.org/el-gallery_view.php?arquivoId=8221 e o repositório do FDP <http://sourceforge.net/projects/fdpack/developfdpsf>.

³⁰ Veja: <http://www.estudiolivre.org/carta-pts-midias-livres>

Em decorrência deste trabalho, surgiu o Musimetrics, o Cinemetrics e o Literametrics. Uma publicação no *Journal of Statistical Mechanics: Theory and Experiment* condensa e compara as análises de filosofia e música.⁽⁶⁶⁾ Além disso, é uma utilização das ciências duras para assunto mais incidente em ciências humanas.

- **Textos socialmente engajados**

O uso de pseudônimos é um costume apreciado em diversos meios. As pesquisas informais confirmam vantagens desta prática³¹. Em especial utiliza-se pseudônimos para auxiliar a despersonalização, gerando textos menos presos à satisfação da auto-imagem. Como resultado, além de aumento de produtividade, costuma-se conseguir também compreensões diferentes. Destes textos, dois foram publicados em duas publicações relevantes: O Contracultura Digital e o Peixe Morto, este último relacionado aos submidialogias³².

F.3.2 *Screencasts e outros materiais em video*

- **Python para áudio e música:** Relacionado ao tutorial em texto citado acima, foram feitos videos sobre a utilização da linguagem Python, em grande parte através de usos em *IPython*, para leitura e escrita de arquivos de áudio com bibliotecas padrão e externas, assim como uso básico de wavelets, OSC e outros recursos.
- **Canal Macambira no Vimeo** Atualmente, são mais de 700 videos sobre sessões de *hacking* e experiências de arte e tecnologia. Feitos pelo LabMacambira.sourceforge.net, o canal tem videos do autor deste escrito, e de: Larissa Arruda, Alexandre Negrão, Lucas Zambianchi, Andres Martano, Fernando Gorodski, Chico Simões, Daniel Marcicano, Daniel Pizetta, Daniel Penalva, Nivaldo Bondança, Danilo Shiga, Marcos Murad, Ricardo Fabbri e Vilson Vieira³³. Exemplos especiais dos videos disponibilizados no canal

³¹ Veja: <http://disqus.com/research/pseudonyms/>

³² Veja: <http://mutgamb.org/blog/Submidialogias-Peixe-Morto-para-Baixar> e <http://culturadigital.br/contraculturadigital/2012/02/01/publicacao-contraculturadigital/>.

³³ O canal pode ser acessado em: <http://vimeo.com/channels/labmacambira>.

são:

- *Livcoding*: foram expostos os processos utilizados na apresentação de *livcoding* com ChuckK, vacas e sono REM³⁴.
- Raspagem de dados: menos artístico e mais engajado, é uma breve explicação sobre o processo de raspagem de dados em páginas HTML³⁵.
- Outros videos incluem gambiarras artísticas, uso de scripts/ferramentas para audio-visual, etc.

F.4 Web

Esta dissertação é focada em uma descrição física do áudio digital enquanto música e usos em código desta abordagem. Como pode-se observar acima, neste mesmo Apêndice, este trabalho, disponibilizado e concebido como tecnologias livres, tem dimensões bastante consideráveis em aspectos socialmente engajados e artísticos. Esta seção exhibe os feitos que extrapolam a música por completo.

F.4.1 Tecnologias sociais: Sítios, Conteúdos e Articulação

Sítios

Alguns *sites* foram criados, como parte de trabalhos engajados e tentativa de manutenção das estruturas do labMacambira.sourceforge.net. O primeiro site com alguma relevância seja talvez o site para a Casa dos Meninos / LIDAS, em que foi feita uma estrutura para acolher as conferências setoriais e municipais dos direitos da criança e do adolescente da cidade de São Paulo. Outros exemplos, mais recentes, são as páginas para a Rede Nacional de Cultura Ambiental

³⁴ Veja: <https://vimeo.com/33012735>, <https://vimeo.com/33018740>, <https://vimeo.com/33019291>, <https://vimeo.com/33025717>, <https://vimeo.com/33025913>.

³⁵ Veja: vimeo.com/channels/labmacambira/2681879.

Afrobrasileira e os blogs para a Rio+20 das secretarias do Ministério da Cultura.

Nest meio tempo também foram criadas várias ferramentas *web*, que rodam em navegadores comuns, como o *Firefox*. Destes, vale citar³⁶:

- SOS: sistema para coleta e difusão de conhecimentos populares e étnicos ligados à saúde.
- Ágora Communs, Novo Ágora: sistema de deliberações *online*.
- Economia Criativa: rascunho de sistema para facilitar circulação de bens me 4 eixos: venda de produtos, aluguel de equipamentos, aluguel de lugares e prestação de serviços.
- Catálogo de Ideias: ferramenta simples para catálogo de ideias sobre temas diferentes, junto a palestras, etc.
- Mapeamento dos 301 pontos de SP: mapeamento dos 301 pontos de cultura estaduais.
- Mapas Coletivos: ferramenta para criação e publicação de mapas com possibilidades colaborativas. Bastante usada por pessoas de meios diferentes pela usabilidade e proposta pertinente.
- Leitura de *shapefiles* no Mapas de Vista: implementação de leitura de arquivos do tipo *shapefile* com dados geográficos em um plugin e tema de *Wordpress*.
- Mapper: atualmente, um rascunho de uma ferramenta de publicação de Mapas com Vistas a publicações no facebook e outros formatos.

³⁶ Os arquivos fonte estão nos repositórios em <http://labmacambira.git.sourceforge.net/git/gitweb-index.cgi>.

Conteúdos

Os conteúdos estão em diversas páginas da *wiki* do Nós Digitais e do Estúdio Livre.³⁷ Tratam principalmente de documentações de *software* e receitas para acesso a repositórios ou instalações. Nestas documentações também estão apontamentos de recursos artísticos e apresentações realizadas com a AHT, *livecoding* e outras propostas. Chama a atenção a quantidade de instâncias de *etherpads* usadas, o labMacambira.sf.net chega a criar vários destes epads por dia, para escrita simultânea a várias mãos³⁸.

Articulação, disponibilização e desenvolvimento conjunto

As articulações se dão principalmente através do IRC, do AA e de emails, além de conversas presenciais. O IRC é uma forma de comunicação leve e focada na comunicação coletiva. O AA está explicado abaixo. Não menos importante são os *Etherpads* e as páginas *wiki* criadas, permitindo proposições e escritas rápidas por diferentes pessoas.

AA

Uma das tecnologias desenvolvidas nesta empreitada se propõe a estabelecer formas de compartilhar andamentos e processos. As motivações incluem transparência civil e organização assíncrona de grupos descentralizados. O sistema se baseia no envio de mensagens periódicas sobre as atividades que estão sendo desempenhadas, evitando foco em produtos e propaganda. A ferramenta pode ser usada no IRC ou em linha de comando. Existem algumas interfaces *web* para exibir as mensagens e dados do AA³⁹. Estão planejados desenvolvimentos do AA para que seja utilizável através do *chat* em diferentes redes sociais. Também está sendo considerada a utilização do AA para distribuição de verba de acordo com as dedicações comprovadas pelos

³⁷ Veja: http://wiki.nosdigitais.teia.org.br/Lab_Macambira, http://www.estudiolivre.org/el-user.php?view_user=gk.

³⁸ Veja: <http://pontaopad.me/epads>

³⁹ Veja: <http://wiki.nosdigitais.teia.org.br/AA>.

mecanismos do AA, tornando possível que se trabalhe de forma remunerada pelo bem comum.

F.5 Momento atual e previsões

Está sendo escrito sobre *livecoding*, especialmente ligado ao Vivace e ao subgênero *Freakcoding*, que recebeu um manifesto recentemente. Um artigo explicando os vínculos entre a proposta estética e o programa Vivace está sendo escrito por várias mãos. O georeferenciamento, perene na atuação do labMacambira.sf.net, tomou agora a direção de arrumar animações de fluxos em arestas de grafos dispostos em mapas. Apresentações artísticas estão previstas para uso da AHT e experimentações de *livecoding*. Um *crowdfunding* foi feito para experimentação de difusão na rede e respostas, com estudos em andamento sobre comportamentos no *Facebook* e listas de *emails*. O Cultura Viva, programa do governo federal que apoiou parte destas iniciativas, indica fortalecimento, tornando-se política de estado, não somente do governo que a implantou. As aproximações federais são observadas também por parte do MEC (Ministério da Educação), com a prestação de uma acessoria em 2012, e do MinC, devido à Lei Cultura Viva. Comunicações mais moderadas continuam com o MCTi, MC e MMA. Os canais de IRC e emails do labMacambira.sf.net estão bastante ativos e focados em execução de tarefas, e, dentre os desenvolvimentos em código previstos, está o aproveitamento desta dissertação para implementações em JavaScript, experimentações, apresentações, publicações, usos didáticos e estudo propriamente dito. O alto rendimento do labmacambira.sourceforge.net no GSoC de 2012, e o forte empenho de alunos do IPRJ/UERJ, de Nova Friburgo, através do suporte oferecido pelo prof. Ricardo Fabbri, da mesma instituição, apontam para uma vida própria que estes andamentos tomaram. Em São Paulo, a atuação de Geraldo Magela e Caleb Luporini se propõe a difundir estas práticas em circuitos artísticos e de ensino. Por emails privados, listas ou IRC, os participantes do grupo estabelecem trocas com pessoas em diversas localidades no Brasil e no mundo para fins de desenvolvimento tecnológico, trabalhos, elaborações didáticas e convivên-

cias, mesmo que brandas, por protocolos de comunicação. A proposta no momento é focar em algo que contemple estas coisas todas da melhor forma, aliando cobertura dos interesses e precisão no aprofundamento, o que está despontado nos estudos de *logs* de IRC e listas de *emails*, com bancos de dados massivos e públicos. As possibilidades artísticas, engajadas, científicas e de instrução e interesse pessoal são confluentes através da modelagem por redes complexas, o que sugere a unificação do conhecimento, traço marcante da área e observado com frequência.

APÊNDICE G – Trabalhos relacionados e caracterização das contribuições deste trabalho

Os trabalhos relacionados a esta dissertação são numerosos. Dentre as causas disso, pode-se apontar:

- Natureza interdisciplinar entre música, computação e física.
- Há um interesse generalizado em música por parte das pessoas que compõem a sociedade.
- A programação de computadores está se difundindo notavelmente.
- As rotinas descritas neste trabalho são essenciais para boa parte dos *software* voltados para áudio e música.

Embora as rotinas, apresentadas nesta dissertação, estejam presentes em diversas implementações livres e proprietárias, suas descrições precisas se encontram somente em código computacional. A maior contribuição desta dissertação é exatamente a descrição analítica das qualidades básicas que compõem elementos musicais no áudio digital. A apresentação didática dos fenômenos envolvidos também não foi encontrada na literatura visitada, o que, junto com as implementações em código Python destas relações e de peças musicais que as exemplifiquem, forma uma contribuição simples e convidativa embora inédita e multidisciplinar. No início da

escrita desta dissertação, a caixa de ferramentas não estava prevista, ela foi fruto das equações e descrições precisas, o que tornou imediata a escrita dos scripts que compõem a *toolbox* MASSA.

Este capítulo é dedicado aos trabalhos similares ou relacionados. Os livros mais próximos são descritos. Na sequência, são apontadas as implementações computacionais proprietárias e livres. Por fim, esta dissertação é posicionada com relação aos trabalhos relacionados.

G.1 Livros

1. *Music For Geeks And Nerds: learn more about music with Python and a little bit of math*

- **Descrição:** com exemplos em código computacional e sonoros, este excelente livro de Pedro Kroeger aborda conceitos de notas, afinações, especificação da nota Midi e conversão entre nomenclaturas latina (dó-ré-mi) e anglo-saxã (C-D-E). Lida também com operações musicais fundamentais como transposição, inversão e afins, combinações randômicas, por Fibonacci. Explora estas organizações tanto para acordes quanto para combinações horizontais (melódicas). Apresenta o básico sobre a constituição dos sons, batimentos, série harmônica, e um aprofundamento sobre as afinações. Por fim, aponta os recursos de ampliação temporal e de tessitura (alturas) de um dado conjunto de notas. Com isso, aponta o dipolo repetição/variação. Faz vínculos de apreensão de estruturas com peças de Josquin des Prez, Bach, Rachmaninoff e Steve Reich.
- **Aspecto complementar:** as formalizações de operações dentro da notação tradicional são preciosos adendos às questões naturais abordadas na presente dissertação. A notação em si capta aspectos estruturais do sistema tonal e de 12 notas. Além disso, com a notação abre-se uma ponte com as tradições musicais eruditas.

- **Aspecto diferencial:** o livro não desenvolve descrição precisa de aspectos musicais do som em si e não há foco em relacionar qualidades psicofísicas aos elementos musicais.

2. *The Theory and Technique of Electronic Music*

- **Descrição:** um livro de Miller Puckette, de reconhecida complexidade, se define, nas palavras de Max Mathews (Prefácio) "The Theory and Technique of Electronic Music is a uniquely complete source of information for the computer synthesis of rich and interesting musical timbres". O livro começa com medições do som e controle de parâmetros, cai em síntese, modulações, métodos espectrais, atrasos e reverberações e termina com filtros.
- **Aspecto complementar:** o livro apresenta diversos procedimentos valiosos para síntese, tratamento e análise. O texto busca ser de computação musical em geral e todo o texto é acompanhado de exemplos em Puredata, que é uma excelente linguagem de programação por patches voltada para audiovisual. Puredata é a linguagem de programação mais difundida na música acadêmica e tecnológica em geral.
- **Aspecto diferencial:** salvo raras exceções, o livro não apresenta uma descrição analítica das amostras sonoras com relação aos procedimentos, assim, não relaciona de forma precisa as qualidades físicas do som. Tampouco se aprofunda em aspectos formais da teoria musical tradicional.
- **Contribuições diretas:** na página 92, há uma solução para a o *fade-in* e o *fade-out* que, se feitos em progressão geométrica, demora a cair ao inaudível. A curva "quártica" atinge o zero e se é bastante próxima da progressão exponencial, especialmente nas intensidades maiores: $a_n = \left\{ \left(\frac{n}{\Lambda-1} \right)^4 \right\}_0^{\Lambda-1}$. Outra contribuição é a descrição prática do uso ideal de 1000 ou mais linhas de atrasos por segundo para simular a reverberação. Também deixa claro que há uma equalização na atenuação do som

refletido, e que esta equalização tende ser mais atenuante nos agudos.

3. *Real Sound Synthesis for Interactive Applications*

- **Descrição:** livro do Perry Cook, de 2002, discorre sobre fundamentos de áudio digital e modelagem física. Possui preciosas dicas de modelagens de sons reais com características de instrumentos tradicionais e efeitos com origem nos instrumentos analógicos.
- **Aspecto complementar:** implementa diversos instrumentos musicais e efeitos sonoros. Trata de uma biblioteca em C para síntese sonora que contempla boa parte das funcionalidades da MASSA.
- **Aspecto diferencial:** O trabalho escapa à descrição das amostras sonoras em si e não possui uma sistematização de elementos musicais básicos em termos das características sonoras.
- **Contribuições diretas:** a parte "1.3 Quantização" descreve de forma breve e suficiente o ruído de quantização e pode complementar à dissertação em "1.1 Som em áudio digital". O escrito sugere uma melhora na relação sinal/ruído de 6 decibels por bit utilizado na quantização de cada amostra do áudio PCM. O que indica diretamente uma relação sinal/ruído de 96dB para 16 bits/amostra (padrão de CD) e 48dB para 8 bits/amostra (comum em sistemas de voz).

4. *Interação Tímbrica na Música Eletroacústica Mista*

- **Descrição:** dissertação de Ignacio de Campos, possui diversas discussões cognitivas e musicais.
- **Aspecto complementar:** desenvolvem os apontamentos da dissertação e do Apêndice F com relação à *performance* musical.

- **Aspecto diferencial:** o trabalho não lida de forma sistemática com as características do sinal digital de elementos musicais.
- **Contribuições diretas:** a parte "2.3.2 A Sensação de Identidade Tímbrica" pode completar a exposição sobre timbre em "2.1.4 Timbre" com os apontamentos de envoltória, inarmonicidade, jitter e shimmer.

5. *Music, Cognition, and Computerized Sound: An Introduction to Psychoacoustics*

- **Descrição:** outro livro do Perry Cook, desta vez são artigos de vários autores, ele cuidou da edição. Um valioso apanhado pertinente para a psicofísica de elementos musicais no áudio digital, com textos focados em aspectos cognitivos e físicos do som além de eventuais descrições de procedimentos elétricos e digitais envolvidos. O foco é qualitativo embora com bastante embasamento quantitativo.
- **Aspecto complementar:** o livro traz artigos de vários autores. As temáticas são pertinentes para a dissertação como cultura a respeito dos assuntos.
- **Aspecto diferencial:** o trabalho não é uma descrição de elementos musicais com relação às características do som digital.
- **Contribuições diretas:** talvez as partes mais interessantes para a dissertação sejam a subseção "23.10 Special Considerations in Psychoacoustic Research" e o capítulo "Appendix A: Suggested Lab Exercises".

6. *Modelos Psicoacústicos de Dissonância para Eletrônica ao Vivo*

- **Descrição:** tese de Alexandre Porres com considerações pertinentes sobre modelos psicoacústicos, dissonância e aspereza/rugosidade. Outros trabalhos, do mesmo autor, incluem um livro sobre computação musical em PD e apresentam, de forma acessível, procedimentos-chave de computação musical junto às implementações em Puredata.

- **Aspecto complementar:** a tese foca em teorias psicoacústicas e em descrições minuciosas de rugosidade e dissonâncias que podem acrescentar bastante ao que foi apresentado na dissertação. Além disso, há um viés prático com a utilização do Puredata.
- **Aspecto diferencial:** a trabalho não é uma descrição psicofísica de sequências amostrais relacionadas a elementos musicais.

7. 5 livros do Julious O. Smith III(36, 42, 73–75)

- **Descrição:** este autor possui diversos escritos, mais de 200 artigos publicados. Ao menos 5 livros de interesse para a dissertação: *Mathematics of the Discrete Fourier Transform (DFT)* descreve a DFT minuciosamente, incluindo FFT. O livro *Introduction to Digital Filters (with audio applications)* trata de filtros de diversos tipos e descreve transformadas, técnicas de design de filtros e análises de frequência e por polos e zeros. Além de abordar filtros não convencionais, o livro termina com implementações em matlab, Faust e PD e possibilidades de confecção de plugins. O livro *Physical Audio Signal Processing (for virtual musical instruments and audio effects)* trata de modelagem física. O livro *Spectral Audio Signal Processing* adentra STFT em detalhes das janelas utilizadas e aplicações da STFT. Também aborda Wavelets de forma superficial porém consistente. O livro *Audio Signal Processing in Faust* trata da linguagem Faust e suas facilidades para o processamento de áudio e música.
- **Aspecto complementar:** a seção "Perceptual Aspects of Reverberation", do livro de filtros, pode fechar o assunto da espacialização junto à subseção "2.1.7 Localização espacial". As partes de modelagem de voz e de fundamentos físicos podem também servir de bons complementos à dissertação. o livro sobre Faust complementa a dissertação por explicitar uma linguagem de domínio específico para áudio e música.

O livro também contempla usos integrados com outros programas e linguagens via, por exemplo, o padrão de plugin LADSPA ou o protocolo OSC de comunicação entre programas para manipulação multimídia.

- **Aspecto diferencial:** estes trabalhos apresentam teorias fundamentais para o que se faz hoje em termos de programação para música, e não uma descrição de elementos musicais em termos das amostras digitais.
- **Contribuições diretas:** a parte de espacialização tem como referencia forte o livro de modelagem física. A clareza sobre a natureza da reverberação e sobre suas características foi toda disparada por este livro.

8. *Music: A Mathematical Offering*

- **Descrição:** este livro, escrito por Dave Benson, se ocupa dos fundamentos matemáticos do fenômeno e tradições musicais.
- **Aspecto complementar:** algumas afinações, escalas e explicações dos capítulos 5 e 6 (sobre afinações e escalas) podem facilitar algumas compreensões apontadas em subseção 2.3.1. O capítulo "9- Symmetry in Music" pode acrescentar nas estruturas cíclicas do capítulo 2.3 da dissertação, especialmente a parte de grupos, Cayley e órbitas.
- **Aspecto diferencial:** o livro não trata de áudio digital. Além disso, o trabalho é uma coleção de modelagens matemáticas, não uma descrição do fenômeno sonoro-musical.

9. *The Topos of Music: Geometric Logic of Concepts, Theory, and Performance*

- **Descrição:** este impressionante livro de Guerino Mazzola possui quase 1500 páginas e explora diversos formalismos matemáticos para estruturas musicais. É um livro fascinante que atinge questões filosóficas.

- **Aspecto complementar:** podem servir de ilustração ou aprofundamento: a subseções "6.5 Regular and Circular Forms", "22.4 Paradigmatic Groups" e "22.5 Pseudometrics on Orbits" e as seções "8 Symmetries and Morphisms" e "11 Orbits" fundamentam e ampliam a parte de estruturas cíclicas de 2.3. A subsubseção "7.2.2 Local Meters and Local Rhythms" e a seção "21 Metrics and Rhythms" podem aprofundar na parte de "2.3.4 Ritmo". A Subsubseção 9.4.2. pode dar uma contribuição na abordagem artística do código computacional. A subseções "9.4 Understanding Fine and Other Arts" e "10.1 Paradigmata in Musicology, Linguistics, and Mathematics" e as seções "47 Unfolding Geometry and Logic in Time" e "48 Local and Global Strategies in Composition" podem fundamentar melhor a parte 2.3.7. Idioma Musical. Os tomos "VI Harmony" e "VII Counterpoint" podem aprofundar as partes "2.3.2 Rudimentos de harmonia" e "2.3.3 Contraponto". O tomo "XV Appendix: Sound" é bastante interessante, com algumas partes que lembram bastante a dissertação: "A Common Parameter Spaces", "A.1 Physical Spaces", "A.2.1 Onset and Duration", "A.2.2 Amplitude and Crescendo", "A.2.3 Frequency and Glissando" e "B Auditory Physiology and Psychology". No tomo "XVI Appendix: Mathematical Basics", interessa a parte "C.3 Groups" em que trata de homomorfismos, produtos direto, semidireto e 'Wreath', teoremas de Sylow, classificação, grupos afins gerais e grupos de permutação, a subseção "E.3 Categories of Modules and Affine Transformations", "F.2 Spectra of Commutative Rings". O tomo "XVII Appendix: Tables" pode ter algumas afinações para constar em "2.3.1 Afinação, intervalos, escalas e acordes".
- **Aspecto diferencial:** o livro não se ocupa da descrição de elementos musicais em termos do áudio digital.

- **Descrição:** livro Larry J. Salomon, de 1973 e revisado em 2002, é uma leitura de procedimentos composicionais tradicionais tendo como base comum a simetria.
- **Aspecto complementar:** é um escrito bastante pertinente para toda a seção "2.3 Organização de notas em música", principalmente para a parte "2.3.6 Estruturas cíclicas" com as rotações, reflexões, translados e incidências tradicionais.
- **Aspecto diferencial:** o trabalho não relaciona as características musicais com o som como fenômeno físico.
- **Contribuições diretas:** foi um livro importante no amadurecimento das questões tratadas nesta dissertação por ser um livro focado em simetrias na música.

11. *The CSound Book*

- **Descrição:** editado por Richard Boulanger, 2000, possui contribuições de diversos autores. Além de abordar princípios da linguagem, seus funcionamentos internos e paradigmas musicais, é um apanhado de técnicas de síntese musical do som e estruturas de mais alto nível.
- **Aspecto complementar:** apresenta a paradigmática linguagem de programação voltada para música: CSound. Junto a esta valiosa empreitada, consagrados recursos musicais eletrônicos são expostos.
- **Aspecto diferencial:** é focado no CSound e não se propõe a ser um livro de fundamentos com descrição formal das características do som relacionado às estruturas musicais.

12. *The Audio Programming Book*

- **Descrição:** de Richard Boulanger e Victor Lazzarini, 2011, um livro interessantíssimo que trata de programação em C/C++ voltado para áudio. Possui várias receitas

tradicionais e fundamentais, todas implementadas em código C. Aborda CSound ao final.

- **Aspecto complementar:** aborda programação em C/C++, apresenta aspectos cruciais das implementações.
- **Aspecto diferencial:** não é uma descrição do comportamento amostral do sinal sonoro com relação aos elementos musicais.

13. *Computer Music Tutorial*

- **Descrição:** de Curtis Roads, 1996, é já um clássico com algumas partes escritas por outros autores, como colaborações.
- **Aspecto complementar:** o capítulo "VII - Psychoacoustics" pode ser complementar à dissertação em algum detalhes e para aprofundamento.
- **Aspecto diferencial:** Os textos do livro são de alto nível, sem preocupação central com o rigor, mas sim com apontamentos qualitativos e históricos das subáreas da música computacional.

14. *Music and Acoustics: From Instrument to Computer*

- **Descrição:** um curto e preciso livro com os fundamentos acústicos, de modelagem de instrumentos musicais, de síntese e tratamento digitais.
- **Aspecto complementar:** apresenta técnicas de modelagem física, além de aspectos musicais e computacionais.
- **Aspecto diferencial:** não é uma descrição das amostras do sinal sonoro com relação às características musicais.
- **Contribuição direta:** a proposta de modelar a reverberação como um ruído com decaimento exponencial foi importante para a elaboração da exposição sobre reverberação na subsubseção 2.2.6

G.2 Bibliotecas, linguagens e conjuntos de implementações computacionais voltados para música

Implementações fechadas. As seguintes implementações fogem ao escopo do trabalho dada a inexistência de descrição amostral ou código que assegure uma implementação precisa e limpa. Como estas implementações são comerciais e o compromisso é com a satisfação do usuário, podem possuir pequenas adições de reverbs, delays e outros artifícios que, conhecidamente, fazem o resultado final soar mais agradável (chamado de *'catchup'* no som ou na música). Observe que os plugins com funções adicionais às propostas pela classificação podem ser considerados adulterados. Exemplos desta adulteração são, em funções díspares do efeito especificado: aplicar um compressor de dinâmica para fazer o som ser ouvido como *'mais volumoso'*, aplicar reverb, dar ganho de equalização nos agudos ou nos graves, etc.

1. *Plugins da Waves*: vários plugins que simulam equipamentos consagrados de estúdio analógico, além de montagem e configurações típicas de músicos e produtores famosos. Há plugins concebidos/desenhados por artistas/produtores com notória competência. Há uma série de plugins de um só parâmetro. Há uma linha de compressores, *gates* e *de-essers*. Uma linha especial de *plugins* de *limiters*. Uma extensa linha de excelentes *plugins* de EQs. Uma linha de plugins de *reverbs* e *delays*. Outra de efeitos como Doppler, *flanger*, *morpher*, etc. Uma linha focada em *plugins* de *surround*. Uma linha bastante diferenciada de *plugins* de remoção de ruído. Diversos *plugins* para vocais. Alguns *plugins* para alteração e modificação de altura. Outros analisadores. Por fim, alguns *plugins* para criação de imagem estéreo e maximização da sensação do grave.
2. *GRM Tools* é uma coleção de plugins para transformação sonora que atinge, na prática cotidiana, resultados únicos e poderosos. É um conjunto de efeitos bastante considerado na música eletroacústica. Pode interessar à dissertação como exemplo de implementações

modelo.

3. A *Steinberg* possui uma série de programas considerados da melhor qualidade, incluindo os sequenciadores Cubase/Nuendo e o editor de áudio Wavelab. Há também diversos plugins VST (que é o padrão de plugin de áudio mais difundido e é da Steinberg) para síntese e tratamento sonoro. A Steinberg está também fornecendo bancos de sons.
4. *Synths* (i.e. sintetizadores) mais conhecidos: Reason e ReCycle são produtos da Propellerheads com bastante ênfase em música de pista e simulação computacional de equipamentos e configurações de canais de áudio em hardware. O Tassman e outros instrumentos da Applied Acoustics Systems proporcionam sons de qualidade impressionante. Os instrumentos da Native Instruments são consagrados, em especial o sintetizador ABSynth, o Traktor para VJing e o Maschine para fazer batidas; além destes, o sintetizador modular Reaktor e o pacote Komplete são também bastante conhecidos.

Implementações livres. Podem ser consideradas documentações pois os códigos computacionais estão disponíveis e podem ser considerados especificações dos procedimentos envolvidos.

1. *SndObj* é uma biblioteca em C usada também como um módulo Python. O manual está claro: "o SndObj é uma unidade de programação que pode gerar sinais com especificações de áudio e parâmetros de controle". A biblioteca tem anos de existência, foi escrita e é mantida por Victor Lazzarini com contribuições da FFTW e do Frank Barknecht. A documentação, incluindo sítio, manual e códigos, lida com especificações técnicas da biblioteca como um todo e com as entradas e saídas de cada objeto. É complementar com relação às implementações dos Apêndices. Não possui descrição analítica dos procedimentos.
2. *The MusicKit* é uma biblioteca em ObjC com mais de 10 anos de existência. Descreve-se como um sistema de *software* orientado a objetos para criação musical.

3. *Abjad* é uma biblioteca Python que se define como um "sistema de *software* interativo para ajudar compositores a escreverem complexas peças de notação musical de forma incremental e iterativa". O *Abjad* utiliza o *Lilypond* para renderizar as estruturas em notação musical.
4. *music21*: esta biblioteca do MIT se descreve como um conjunto de ferramentas para musicologia auxiliada por computadores. Em grande parte, lida com a entrada e saída de elementos musicais em bancos de dados com obras dos compositores mais relevantes da música erudita. Trata principalmente de notação musical e análise de dados. Complementar à dissertação é esta própria ênfase em análise e notação musical. Diferencial que não trata de aspectos físicos do som ou sua representação amostrada no tempo, mas sim de aspectos que estão descritos pelo viés natural em "2.3 Notas em música".
5. Embora ainda possa ser usado como uma biblioteca, o *AthenaCL* tem já características de uso que fogem à linguagem Python, com terminal próprio e outras interfaces. O autor, Christopher Arizza, também co-responsável pelo *Music21*, o define com um "sistema que é uma ferramenta computacional orientada a objetos e em código aberto". O manual possui diversas das ferramentas, os detalhes de uso e implementações. É complementar quanto ao trabalho o aspecto de ferramenta computacional com linguagem própria e ferramentas encapsuladas à disposição. Diferencial é que não tem um tratamento analítico e não se atém às amostras em si.
6. *PD* e *Max/MSP*: são linguagens de domínio específico (para música) que aproveitam a interface das linguagem de patches. As caixinhas cujo texto especifica o objeto e cujas entradas e saídas são ligadas por cordinhas são tanto uma interface amigável quanto uma interface gráfica para utilização do programa. Ambas as linguagens foram feitas em grande parte por Miller Puckette, que cuida atualmente somente da versão livre, que é o *PD*. O *Max/MSP* é mantido pela empresa Cycling 74. Os manuais do *Max/MSP* são

ótimas e até históricas introduções qualitativas e práticas. O livro do Puckette, citado em na lista de livros acima, é um dos materiais excelentes que utilizam o PD. Complementar à dissertação há esta aproximação de um programa/sistema, com interface, e mais engessado do que uma linguagem de programação. Diferencial é que não há descrição analítica do que ocorre com as amostras. O código computacional, neste caso, é a documentação e Puckette explicita que o código do PD deva sempre ser completo se impresso em uma folha de papel.

7. *Faust, Chuck, Supercollider, Nyquist, Impromptu*: linguagens de domínio específico como p PD e o Max/MSP, voltadas para a síntese de música em tempo real, mas escritas normalmente em linhas, não em patches. Complementar ao trabalho é que são implementações computacionais excelentes para uso em tempo real. Diferencial é que há perda da desenvoltura para manipulações de áudio com controle amostral de precisão arbitrária. A descrição analítica também é inexistente.
8. *CSound*: linguagem de programação voltada para música mais antiga que está em uso. As constantes produções acadêmicas em torno do CSound podem servir de descrições mais rigorosas do que as encontradas nas soluções anteriores. De qualquer forma, não há uma documentação clara e concisa sobre os procedimentos básicos como o capítulo 2 da dissertação.
9. *tuneR*: um pacote em R para análise de música. A documentação é um bom manual dos objetos e as características que extraem.
10. *Gwibber* e *Vivace*: linguagens de programação voltadas para música (e video no caso do Vivace) utilizáveis via *browser*.

G.3 Aprofundamento sobre esta dissertação com base nos trabalhos visitados nas subseções G.1 e G.2

Há livros de técnicas de música eletrônica, como o do Puckette. Já materiais como o livro do Pedro Koeger e a biblioteca music21 não tratam das amostras sonoras em momento algum, mas sim da representação e manipulação de estruturas musicais, assunto abordado na seção 2.3 do ponto de vista natural. Há também trabalhos focados em modelagens matemáticas de técnicas e princípios musicais, cuja consideração permeia toda a dissertação e aprofundamentos são bem vindos. Estes trabalhos, quando muito, apresentam princípios de sinais discretos, mas não se ocupam da descrição do comportamento amostral. Os trabalhos de Cook, Smith III e algumas implementações possuem descrições de fenômenos vibratórios em corpos rígidos e outras modelagens físicas. Pode-se propor expansões para usos conjuntos com alguns dos trabalhos, visitados ou não. Vale assinalar: com base na experiência prévia dos autores, nos escassos artigos encontrados e nos trabalhos visitados em G.1 e G.2, a descrição analítica dos elementos musicais básicos em termos das amostras do áudio digital parece ser inédita. Como há também a implementação em código livre das relações descritas nesta dissertação na *toolbox* MASSA, abre-se um leque para experimentações científicas rigorosas assim como para usos artísticos com alta fidelidade e precisão arbitrária.(2)