

Computação Musical

Uma gentil introdução

v.0.0.1 - Edição do Autor: Favor não distribuir!

v.0.0.1 - Edição do Autor: Favor não distribuir!

v.0.0.1 - Edição do Autor: Favor não distribuir!

v.0.0.1 - Edição do Autor: Favor não distribuir!

v.0.0.1 - Edição do Autor: Favor não distribuir!

Pre Print
V.0.0.1

Computação Musical

Uma gentil introdução

v.0.0.1 - Edição do Autor: Favor não distribuir!

v.0.0.1 - Edição do Autor: Favor não distribuir!

v.0.0.1 - Edição do Autor: Favor não distribuir!

v.0.0.1 - Edição do Autor: Favor não distribuir!

v.0.0.1 - Edição do Autor: Favor não distribuir!

Flávio Luiz Schiavoni
Marcelo Soares Pimenta



Editora da SBC
Porto Alegre - RS
Brasil

This book was typeset using L^AT_EX software.

Copyright © 2026

Flávio Luiz Schiavoni

Marcelo Soares Pimenta

License: Creative Commons Zero 1.0 Universal

Prefácio

Como surgiu este livro

A Computação musical é uma área de interesse da computação no Brasil e possui uma Comissão Especial (CE) dentro da Sociedade Brasileira de Computação (SBC)¹. Esta comissão especial se reúne na organização de um evento bienal, o Simpósio Brasileiro de Computação Musical (SBCM), onde a pesquisadores da área se reúnem para apresentar as evoluções de suas pesquisas. No entanto, há muito tempo que a CE de Computação Musical discute a popularização da área no Brasil e a necessidade de integrar novas instituições nesta área de pesquisa e, consequentemente, neste evento.

No ano de 2015, o SBCM aconteceu em Campinas - SP e a reunião da CE trouxe novamente a questão da popularização da área, ainda restrita a poucas instituições. Notou-se nesta reunião que o ensino de Computação Musical no Brasil acabava por ocorrer quase que exclusivamente em instituições de ensino superior cujos docentes tiveram a oportunidade de integrar grupos de pesquisa na área e desenvolver sua pesquisa em programas de pós graduação em instituições que possuíam pesquisa nesta área. Assim, boa parte dos cursos que ofereciam tais opções se restringiam às instituições cujos docentes realizaram sua pós graduação no exterior ou que fizeram parte de grupos de pesquisas de instituições que possuíam relações com a área como USP, UFMG, UEMG, UNB, UFPE, UFSJ, Unicamp, UNESP, UFRGS, e outras. Se por um lado esta área, seu evento bienal e a pesquisa no Brasil é antiga e já bastante sólida, por outro notamos que a área é pouco popular entre as demais instituições de ensino de computação no Brasil.

Visando mudar este quadro, a Comissão Especial apresentou nesta ocasião

¹<https://www.sbc.org.br/14-comissoes/385-computacao-musical>.

a proposta de uma ação para a popularização da área por meio do desenvolvimento de material didático para um curso de Introdução à computação Musical, de forma que docentes de cursos de computação e de outras áreas pudessem utilizar tal material como base para a propor cursos de introdução nesta área.

Foi neste cenário que em 2024 os autores deste livro se uniram para tentar criar este material e assim surgiu este livro. Certamente há muito mais para dizer sobre este assunto e por isso este material se propõe a ser apenas uma introdução, ou melhor dizendo, uma gentil introdução.

Porque estudar computação musical

Acredito que os cursos de ciência da computação, em geral, ensinam os alunos a ler dados do teclado, processar estes dados, e apresentar o resultado do processamento na tela. De certa forma, boa parte de nossos cursos se focam neste tipo de entrada, processamento e saída. Apesar de isso ser eficiente para boa parte dos processamentos de dados que precisamos, corremos o risco de formar alunos que nunca fizeram uso de outras entradas e saídas do computador. Isso implica em dizer que, muitas vezes, um bacharel em ciência da computação ou área afim nunca utilizou a caixa de som como dispositivo de saída de um processamento, ou o microfone como dispositivo de entrada de dados, ou arquivos de áudio como forma de armazenar dados no computador. Talvez esta motivação já seja o suficiente para utilizarmos este livro em sala de aula: uma computação para além do teclado e mouse.

Além disso, a utilização de prompts de saída faz com que nossos algoritmos sempre tragam resultados que conseguimos ver e atestar se estão corretos. Sempre ou quase sempre. Se temos um sistema que gera um ruído branco e fazemos um filtro de passa baixa com frequência de corte em 400Hz e largura de banda de 1 oitava, como sabemos se nosso sistema está funcionando corretamente? Este é outro ponto que acredito que seja muito interessante, em sistemas musicais não basta acreditar em nossos olhos, precisamos confiar também em nossos ouvidos. No entanto, sei que não dá para confiar assim na audição. Logo, o que nos resta para validar sistemas computacionais musicais? A resposta pode estar na matemática! As provas matemáticas estão para além dos olhos e se tornam ferramentas essenciais para quem quer entender como um determinado algoritmo de processamento de sinal funciona.

Para quem é este livro

Este livro destina-se a pesquisadores e pesquisadoras que estejam interessados em começar sua pesquisa na área de computação musical e que não possuem referências para começar. Por esta razão, fazemos questão de no

Como usar este livro

Este livro destina-se a pesquisadores e pesquisadoras que estejam interessados em começar sua pesquisa na área de computação musical e que não possuem referências para começar. Por esta razão, talvez este livro possa ser um pouco prolixo, superficial, lento ou mesmo demasiadamente explicativo para quem já conhece sobre os temas aqui abordados. Não é a intenção fazer um livro chato, mas vamos tentar abordar temas que precisam ser abordados e indicar outros materiais para quem quer deixar a introdução e se aprofundar nesta área de pesquisa.

A organização deste livro

O capítulo 1 é voltado para quem quer entender algumas questões sobre a pesquisa e o ensino da área.

O capítulo 2 é para quem quer dominar um pouco mais o conhecimento sobre música

O capítulo 3 começa a apresentar conceitos de música simbólica.

Além disso, este livro pode ser lido de várias formas. Além de ter seu conteúdo de forma sequencial, o livro também traz:

- ♪ Dicas de leitura: Livros da área de computação musical que podem auxiliar o leitor a aprofundar seus conceitos
- ♪ Dicas de software:
- ♪ Dicas de escuta:
- ♪ Trechos de código:

Convenções usadas neste livro

As imagens das capas de livros e de discos são da Amazon.

Exemplos de código

Falar sobre as APIs e coisa assim?

Como entrar em contato?

Agradecimentos

O Flávio agradece....

O Marcelo Pimenta agradece....

Contribuições

Este livro contou com a ajuda de

Lista de Figuras

1.1	Linha do Tempo das tecnologias na música	2
1.2	Representação da interdisciplinaridade da computação musical . .	16
2.1	Notação na partitura do andamento	28
2.2	Metrônomo de corda	30
2.3	Notação na partitura do andamento	30
2.4	Notação de mudanças de andamento	30
2.5	Partitura com mais de um compasso	32
2.6	Fórmulas de compasso	33
2.7	Ligadura e figura pontuada	34
2.8	Exemplo de dinâmica em uma partitura de percussão	36
2.9	Tela da ferramenta LMMS e sua representação de tempo	38
2.10	Disposição das notas no teclado do piano	40
2.11	Representação dos acidentes	41
2.12	Disposição das oitavas na partitura	46
2.13	Armaduras de Clave	47
2.14	Ciclo de quartas e quintas	48
3.1	Tela to editor de partitura MuseScore	57
3.2	Exemplo de piano roll do LMMS	58
3.3	Painel frontal da TR-808	60
3.4	Marcação rítmica no LMMS	60
3.5	Saída do código partitura em ABC	62
3.6	PDF gerado com a partitura pelo Lilypond.	63
3.7	Portas MIDI em uma interface de som	67
3.8	Notas MIDI anotadas no teclado virtual vkeybd	68
3.9	Apresentação da Partitura gerada pelo Código 3.6 na ferramenta MuseScore	73

3.10	Tela do sintetizador ZynAddSubFx	74
3.11	Ferramenta aconnectgui para conexão de portas MIDI no Linux .	75
4.1	Curva de Fletcher-Munson	89
4.2	Conversão do sinal analógico para o sinal digital	91
6.1	Exemplo de vibrações que ocorrem com cordas	130
6.2	Representação da Séria harmônica na partitura	135
A.1	Sistema ALSA MIDI em ação	147

Lista de Tabelas

1.1	Eventos na área de computação musical	6
1.2	Abordagens para o ensino de computação musical	13
2.1	Nome dos andamentos musicais e seus respectivos BPM	31
2.2	Figuras musicais, seus tempos e pausas, em um compasso 4/4	34
2.3	Representação das notas musicais com letras e possíveis frequências	39
2.4	Nome dos intervalos musicais	49
3.1	Eventos MIDI Padrão	68
3.2	Mensagens MIDI com canal	69
3.3	Outras Mensagens MIDI	70
3.4	Outras Mensagens MIDI	71
3.5	Tipos de dados OSC	78
5.1	Cálculo do filtro passa baixa IIR com uma entrada com frequência alta	111
5.2	Cálculo do filtro passa baixa IIR com uma entrada com frequência baixa	112
5.3	Cálculo dos coeficientes de filtros de primeira ordem	113
6.1	Série harmônica da frequência de 100 Hz	131
6.2	Série harmônica da frequência de 200 Hz	132
6.3	Série harmônica da frequência de 150 Hz	132
6.4	Séries harmônicas consecutivas e ajustadas	133
6.5	Frequências da série dispostas em ordem crescente	134
6.6	Série harmônica da frequência de 110 Hz	134

Lista de códigos

2.1	Exemplos de ritmos	37
2.2	Exemplo de definição rítmica para bateria de rock	37
3.1	Exemplo de código para notação musical em ABC	61
3.2	Exemplo de código para notação musical em ABC	62
3.3	Exemplo de código do Lilypond	63
3.4	Exemplo de código do Lilypond	63
3.5	Exemplo de código do MusicXML	65
3.6	Exemplo de código para criação de arquivos Midi usando Python e a biblioteca MIDIUtil	72
4.2	Geração de arquivos de áudio com silêncio	100

Lista de equações

4.1 Representação comum do áudio digital	91
4.2 Teorema de Shannon-Nyquist	92
5.1 Adição de escalar ao sinal de áudio	106
5.2 Multiplicação de escalar com sinal de áudio	106
5.3 Soma de dois sinais de áudio	107
5.4 Multiplicação de dois sinais de áudio	108
5.5 Equação matemática do filtro FIR de primeira ordem	109
5.6 Filtro passa baixa	110
5.7 Filtro passa baixa	110
5.8 Equação do filtro FIR de primeira ordem	111
5.9 Equação do filtro passa baixa FIR de primeira ordem	111
5.10 Equação canônica de filtro de primeira ordem IIR + FIR	112
5.11 Normalização Frequência de corte em relação à taxa de amostragem	113
5.12 Equação canônica de filtro FIR de segunda ordem	113
5.13 Equação canônica de filtro de segunda ordem IIR + FIR	113
5.14 Equação canônica de filtro de primeira ordem IIR + FIR	114
5.15 Equação para geração de senóides	117
5.16 Equação para geração de senóides com Taxa de amostragem definida	117
5.17 Rebatimento de frequências no Teorema de Shannon-Nyquist	119
5.18 Onda quadrada gerada com sistema de equações a partir do período	120
5.19 Onda quadrada gerada com a função chão	121
5.20 Onda quadrada gerada a partir da senoide	121
5.21 Onda dente de serra criada com a função chão	122
5.22 Onda triangular com funções seno e arco-seno	123
5.24 Onda triangular calculada a partir de uma onda dente de serra	123
5.25 Ajuste do oscilador por consulta a tabela para utilizar uma tabela de qualquer tamanho	125
5.26 Fórmula da equação linear	125

5.27 Equação linear com a saída isolada 126

5.28 Exemplo de interpolação linear 126

5.29 Cálculo da interpolação quadrática pela Fórmula de Lagrange 127

5.30 Exemplo de interpolação quadrática 127

5.31 Cálculo da interpolação cúbica pela Fórmula de Lagrange 128

6.1 Série Harmônica 131

Dicas de livros

Pre Print
V.0.0.1

Lista de Aplicações

Mixxxx	9
Sonicvisualiser	10
PureData	10
LADSPA	12
SonicPi	12
LMMS	28
Hydrogen	34
Muscore	45

Dicas de escuta

Pre Print
V.0.0.1

Sumário

1	Introdução	1
1.1	A pesquisa em Computação Musical	3
1.1.1	Congressos e eventos na área	4
1.1.2	Revistas e publicações	7
1.2	O ensino de Computação Musical	8
1.2.1	Criação e utilização de sistema musicais	8
1.2.2	Recuperação de informação musical	9
1.2.3	Novos Instrumentos musicais digitais	10
1.2.4	Efeitos de áudio digital	11
1.2.5	Composição algorítmica	12
1.3	Computação Musical e interdisciplinaridade	14
1.4	Interdisciplinaridade na música	16
1.5	Interdisciplinaridade na Computação	18
1.5.1	Sistemas operacionais e computação de alto desempenho .	18
1.5.2	Redes de computadores e as orquestras de computadores .	19
1.5.3	Interface Humano computador	19
1.5.4	Micro eletrônica e Eletrônica digital	20
1.5.5	Computação Móvel	20
1.5.6	Linguagens de programação musical	21
1.5.7	Inteligência Artificial	21
1.5.8	Web e música em HTML	22
1.5.9	Desenvolvimento de jogos	22
1.5.10	Computação Gráfica e Processamento digital de Imagens	23
1.5.11	Visão Computacional e OMR	23
2	Elementos da Música Ocidental	27
2.1	Tempo musical	28

2.1.1	Andamento	29
2.1.2	Compasso	31
2.1.3	Figuras de Tempo musical	33
2.2	Dinâmica Musical	35
2.2.1	Ritmo	37
2.3	Altura musical	38
2.3.1	As notas musicais	38
2.3.2	Escalas musicais	40
2.3.3	As Notas na partitura	45
2.3.4	Intervalos musicais	49
2.3.5	Acordes	50
2.3.6	Campo Harmônico	52
2.4	Timbre	53
3	Música simbólica	55
3.1	Escrevendo música simbólica	56
3.1.1	Notação Textual	57
3.1.2	Partitura	57
3.1.3	Piano Roll	58
3.1.4	Sequenciador de passos	59
3.2	Formatos de arquivos para notação musical	60
3.2.1	ABC	61
3.2.2	Lilypond	63
3.2.3	MusicXML	64
3.3	MIDI	66
3.3.1	Eventos	67
3.3.2	Metaeventos	70
3.3.3	SysEx	71
3.3.4	Arquivos MIDI	72
3.3.5	O protocolo MIDI em Tempo Real	73
3.4	OSC	76
3.4.1	As mensagens OSC	77
3.4.2	Um protocolo muito aberto	79
4	Áudio digital	83
4.1	No início era o som	84
4.1.1	Características do som	85

4.1.2	Da acústica para a elétrica: O sinal de áudio	89
4.2	O Sinal de Áudio Digital	90
4.2.1	Taxa de Amostragem	91
4.2.2	Quantização e Codificação	93
4.2.3	Quantidade de canais	97
4.3	Utilizando áudio no computador	98
4.3.1	Utilizando arquivos de áudio	98
	Escrevendo em um arquivo de áudio	100
4.3.2	Processamento de Áudio em tempo real	101
5	Processamento de sinais de áudio	105
5.1	Aritmética de sinais	105
5.1.1	Soma de escalar com vetor	105
5.1.2	Multiplicação de escalar com vetor	106
5.1.3	Soma de vetores ou mixagem	107
5.1.4	Multiplicação de vetores	108
5.2	Filtros de áudio digital	109
5.2.1	A teoria dos Filtros	111
5.2.2	Calculando os coeficientes de filtros	114
5.3	Extraindo propriedades do sinal de áudio	114
5.3.1	Duração	114
5.3.2	Amplitude	114
5.3.3	Frequência	114
5.4	Áudio como controle	115
5.5	Gerando sinais de áudio	115
5.5.1	Ruído	115
5.5.2	Senos e cossenos	116
5.5.3	Onda quadrada	119
5.5.4	Onda Dente de serra	121
5.5.5	Onda Triangular	122
5.5.6	Oscilador por consulta a tabela	123
6	Algoritmos de síntese sonora	129
6.1	Temperando ondas periódicas	129
6.1.1	A Série Harmônica	131
6.1.2	A afinação pitagórica	132
6.1.3	Temperamento Justo	133

6.1.4	Série harmônica e escala natural	134
6.1.5	Relações matemáticas da música	135
6.2	Envelopes	136
6.2.1	Envelope ADSR	136
6.2.2	Envelope AR	136
6.3	Síntese Aditiva	136
6.4	Síntese Subtrativa	136
6.5	Modulação de Amplitude	136
6.6	Modulação de Frequência	136
6.7	Modulação de Fase	136
6.8	Síntese Granular	136
6.9	Modelagem Física - Karplus-Strong	136
7	3 Perspectivas para continuar este trabalho	137
7.1	Music Information Retrieval	137
7.1.1	Medindo sinais de áudio	137
7.1.2	Amplitude de Sinal	137
7.2	DAFx	138
7.2.1	Dinâmicos	138
7.2.2	Filtros	138
7.2.3	Espacialização	138
7.3	NIME	138
7.3.1	Instrumentos tradicionais	138
7.3.2	O instrumento digital (tripartite)	138
7.3.3	Sintetizador	138
	Apêndices	139
A	Computação musical e Linux	141
A.1	Uma comunidade organizada	141
A.2	Distribuições de Linux para música	141
A.3	Kernel preemptivo - Um pouco de história	143
A.4	ALSA	146
A.5	Servidores de som: JACK, Pulse áudio e pipewire	148
A.6	LADSPA e LV2	148
B	Linguagens de programação Musical	149

C APIs para desenvolvimento de sistemas musicais	151
Index	153

Capítulo 1

Introdução

Este livro é focado no tema computação musical mas é possível entender que a relação entre música e tecnologia não é algo novo e já acontece há muito tempo, especialmente se entendermos que a tecnologia estava presente na criação de instrumentos como o saxofone e o violão, por exemplo. A tecnologia está presente na forma como criamos música, que produzimos, que distribuímos, que reinventamos a ideia do que é a música em si. Na história da humanidade, por exemplo, quando surge o telefone e a voz pode ser ouvida longe, certamente as pessoas cantaram neste aparelho, criando experiências musicais à distância e a música telepresente.

Se formos verificar a evolução da tecnologia e sua relação com a música iremos descobrir coisas interessantes como, por exemplo, que o primeiro gravador só gravava, não tocava mas apenas registrava o som. Este equipamento, chamado, fonautógrafo, foi inventado em 1857, 20 anos antes do fonógrafo. Também podemos entender que logo as pessoas passaram a poder gravar e formatos digitais de música surgiram. No entanto, entendendo que o violão também é tecnologia, assim como a flauta ou outros instrumentos, fazer uma revisão histórica da relação da música com a tecnologia nos leva a difícil tarefa de escolher por onde começar. A Figura 1.1 traz uma possível linha do tempo de algumas tecnologias que podem ser associadas à música.

Esta lista traz alguns instrumentos musicais interessantes como o Theremin, Ondas Martenot ou o Trautonium, o Thelharmonium, órgãos Hammond, Sintetizadores Moog e o Music I. Traz também o VODER (Voice Operating Demonstrator), primeiro dispositivo a ser considerado um sintetizador de fala,

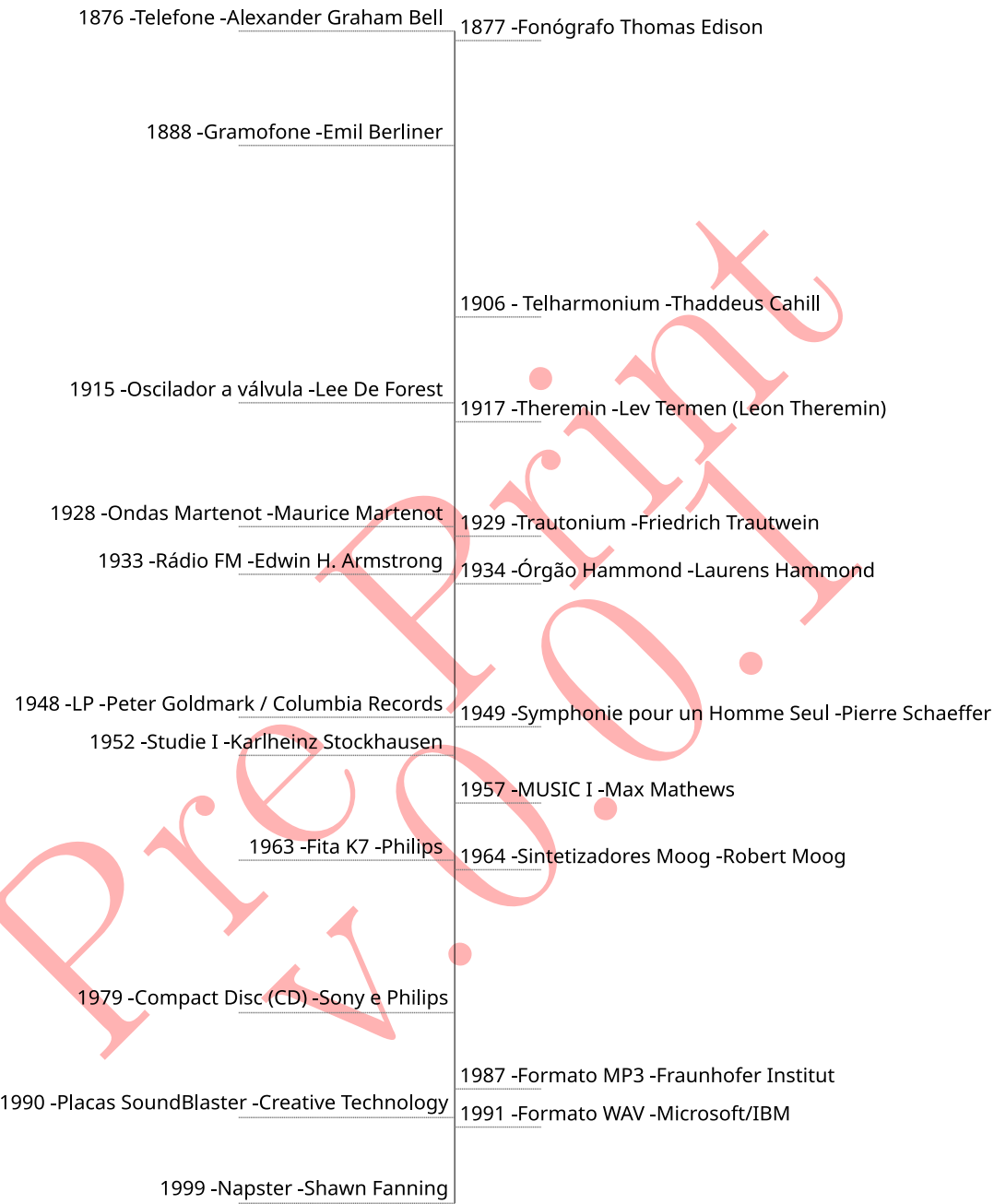


Figura 1.1: Linha do Tempo das tecnologias na música

introduzido por Homer Dudley na Feira Mundial de Nova York em 1939. Também tem algumas composições que mudaram a forma de fazer música como a de Pierre Schaeffer e Karlheinz Stockhausen. Seria possível adicionar a ela outras invenções que foram importantes e modificaram a maneira de produzir e consumir música.

De qualquer maneira, podemos entender por esta linha do tempo que a tecnologia influenciou diretamente na maneira como fazemos música e também na sua distribuição e consumo ao longo de nossa história. Siglas como LP, CD, K7, MP3, IA, DAW, e outras podem estar tão atreladas com a história da música quanto os instrumentos musicais tradicionais como violão ou violino.

1.1 A pesquisa em Computação Musical

Quando em 1957 o engenheiro eletrônico **Max Mathews** escreveu em Assembly o programa de computador *Music I* para o IBM704, ele provavelmente não sabia que estava criando uma área de pesquisa em computação chamada *Computação Musical*.

Max Mathews era doutor em Engenharia Elétrica pelo M.I.T. e trabalhava nesta época no Bell Telephone Laboratories pesquisando acústica e codificação de voz. O projeto de Max Mathews, o *Music I*, foi talvez o primeiro ou um dos primeiros programas de computador feitos para gerar / sintetizar sons. Este programa foi utilizado para criar uma composição feita totalmente em um computador, uma peça de 17 segundos chamada *The silver scale*. Max trabalhou na evolução do *Music I* e acabou criando uma série de programas de síntese sonora, chamados de família MUSIC N. Esta série de programas culminou, em 1986, na criação de uma linguagem de programação musical chamada CSound, que ainda existe e é amplamente utilizada por compositores e cientistas da computação em todo o mundo.

A evolução da família Music N não culminou apenas nesta linguagem de programação. Outras linguagens de programação também surgiram como evolução do Music N, como as linguagens CMusic (1980) e CMix (1984), que ainda possuem usuários e atualizações mas que não são tão difundidas como o **CSound**. Mais recentemente, outras linguagens de programação musical surgiram, com outros conceitos de programação, como o **FAUST**, uma linguagem de programação funcional, o **Pure Data**, uma linguagem de programação visual, e o **SuperCollider**, uma linguagem de programação orientada a objetos. O apên-

dice B traz uma revisão geral destas linguagens.

Além de culminar em diversas linguagens de programação, o projeto de Max e o interesse em computação e música foi convencendo pesquisadores em diversas universidades a trabalharem com este conceito e a pesquisa desta área se popularizou em departamentos de computação e também em departamentos de música, chegando a criar departamentos ou cursos próprios em algumas instituições, como os chamados cursos de música e tecnologia (**Musitech** em inglês).

Diversos centros de pesquisa interessados em trabalhar com computação e música surgiram desde então, como o Center for Computer Research in Music and Acoustics (**CCRMA**) em Stanford, o Institut de Recherche et Coordination Acoustique/Musique (**IRCAM**) em Paris ou o Centro di Sonologia Computazionale (**CSC**) em Padova, o Centre for Interdisciplinary Research in Music Media and Technology (**CIRMMT**) da McGill no Canadá, entre outros.

No Brasil, temos grupos de pesquisa dedicados a este tema como: **LCM** - Laboratório de Computação Musical da Universidade Federal do Rio Grande do Sul (UFRGS), **MusTIC** - Music, Technology, Interactivity, and Creativity, da Universidade Federal do Pernambuco (UFPE), **CEGeME** - Centro de Pesquisa em Gesto e Expressão Musical, da Universidade Federal de Minas Gerais (UFMG), **ALICE** - Arts Lab in Interfaces, Computers, and Everything Else, da Universidade Federal de São João del-Rei (UFSJ), **NESCoM** - Núcleo Espírito-Santense de Computação Musical, da Universidade Federal do Espírito Santo (UFES), **CompMus** e **NuSom** - Núcleo de Pesquisas em Sonologia, da Universidade de São Paulo (USP), **NICS** - Núcleo Interdisciplinar de Comunicação Sonora da Universidade Estadual de Campinas (Unicamp), **COUT-E** - Grupo itinerante de Computação Musical e Tecnologias Emergentes, da Universidade Federal do Amapá (UNIFAP), entre outros.

1.1.1 Congressos e eventos na área

Com o avanço da pesquisa na área, não tardou para que os cientistas e artistas envolvidos no campo da computação e música se reunissem para troca de experiência em congressos. Em 1974, foi fundada a ICMA (International Computer Music Association), que promove anualmente o **ICMC** — International Computer Music Conference (visite o site <https://www.computermusic.org>).

Esta conferência tem o foco artístico e composicional e envolve trabalhos

nos campos de performance ao vivo, instalações sonoras, obras eletroacústicas, música gerada por computador, novas práticas composicionais com tecnologia, com uma comunidade mais próxima da música experimental, eletroacústica, e live electronics.

Enquanto o ICMC tende a ter um foco mais forte na composição musical e na criação artística o **SMC** (Sound and Music Computing Conference) é uma conferência mais voltada à pesquisa científica e tecnológica sobre som e música, com ênfase em aspectos computacionais, perceptivos, cognitivos e engenheirísticos (visite o site <https://smcnetwork.org/>). Esta conferência, que teve seu início nos anos 2000, também aceita trabalhos artísticos mas seu foco é mais nas questões da engenharia de som, ciência da computação musical, psicoacústica e interação.

Já o público preocupado com questões de foco mais prático e técnico em áudio livre, software de código aberto e infraestrutura para produção musical digital baseada em Linux e tecnologias abertas se organizou no **LAC** - Linux Audio Conference (Visite o site <http://linuxaudio.org/lac.html>).

Já a conferência **NIME** (New Interfaces for Musical Expression) é uma das conferências mais únicas, criativas e interdisciplinares no mundo da computação musical — e provavelmente uma das mais experimentais (Visite o site <https://www.nime.org/>). Esta conferência internacional que reúne pesquisadores, artistas, engenheiros, músicos e designers interessados na criação e estudo de novos instrumentos e interfaces musicais. Foi criada em 2001 como um workshop dentro da CHI (Conference on Human Factors in Computing Systems), e hoje é um evento independente, de renome global com foco na criação e avaliação de novos instrumentos musicais digitais.

Já a área de Recuperação de Informação Musical se reúne em uma conferência chamada **ISMIR** - International Symposium on Music Information Retrieval, organizado pela ISMIR — International Society for Music Information Retrieval. Essa conferência é o principal evento mundial para quem pesquisa Recuperação de Informação Musical (MIR), ou seja: extrair, analisar e entender informações musicais a partir de dados digitais. O ISMIR é ainda responsável pelo repositório MIRtoolbox, Essentia, Music21, e outros frameworks populares na pesquisa em música + computação e organiza competições para elevar a qualidade da pesquisa na área.

Já a área de Efeitos de áudio Digital, em Inglês - Digital Audio Effects, organiza a **DAFx** (Digital Audio Effects Conference), uma conferência científica

que teve início em 1998 e que é focada especificamente em processamento digital de áudio para aplicações musicais, criativas, e tecnológicas. Esta conferência é altamente técnica, mas também ligada à criação sonora e musical e é considerada uma das principais referências no campo do processamento de áudio digital musical.

Há ainda conferência específicas para determinadas tecnologias da área de computação e música, como weaudio conference, Pure Data Convention, CSound Conference, e outras. A Tabela 1.1 traz uma visão geral destes eventos.

A questão multidisciplinar da pesquisa em Computação Musical é o foco do **CMMR** – International Symposium on Computer Music Multidisciplinary Research, um evento internacional que promove a troca de ideias e pesquisas sobre a interação entre música e tecnologia, com uma abordagem interdisciplinar (Visite o site do evento em 2023 <https://cmmr2023.gttm.jp/>). Este evento iniciou com o foco em modelagem e recuperação de informações, com o nome de Computer Music Modelling and Retrieval, e desde 2012 teve sua sigla ressignificada para incluir aspectos da interação humana na prática musical, além de perspectivas perceptuais e cognitivas relacionadas à modelagem sonora e à estrutura musical como um todo.

Conferência	Foco
LAC	Linux, áudio, e software livre para criação musical
DAFx	Processamento digital de áudio (DSP) para som e música
ISMIR	Extração e análise de informações musicais (dados + metadados)
NIME	Interfaces para performance musical (instrumentos novos)
SMC	Som, percepção, DSP, análise musical — mais amplo que DAFx
ICMC	Composição musical computacional e performance artística
CMMR	Evento multidisciplinar sobre a interação entre música e tecnologia

Tabela 1.1: Eventos na área de computação musical

No Brasil, o tema é área de pesquisa da Sociedade Brasileira de Computação e é organizado pela Comissão Especial em Computação Musical da SBC (visite o site <http://compmus.org.br>). Esta Comissão Especial reúne os pesquisadores da área em um evento bienal chamado Simpósio Brasileiro de Computação Musical (SBCM), que reúne desde 1994 pesquisadores brasileiros e estrangeiros em torno do tema. Além deste evento, o Brasil ainda possui uma área de interfaces o congresso da **ANPPOM** (Associação Nacional de Pesquisa e Pós Graduação em Música) e o **Ubmus**, Workshop de Música Ubíqua.

1.1.2 Revistas e publicações

Surgiram também revistas especializadas para publicar trabalhos desta área interdisciplinar. A **Journal of New Music Research** (JNMR - <https://www.tandfonline.com/journals/nnmr20>) é uma revista acadêmica internacional, publicada pela Routledge (Taylor & Francis) desde 1972. Esta revista, inicialmente chamada Interface, adotou o nome atual em 1994 e é publicada trimestralmente.

Também da década de 1970, a **Computer Music Journal**(CMJ - <https://direct.mit.edu/comj>), é uma revista acadêmica revisada por pares, publicada pelo MIT Press desde 1977, dedicada a pesquisas e práticas relacionadas aos temas relevantes da computação musical.

O MIT também manteve por 30 edições a revista **Leonardo Music Journal** (<https://direct.mit.edu/lmj>), uma com a Leonardo/ISAST (International Society for the Arts, Sciences and Technology), que esteve ativa entre 1991 e 2020 mas que passou a ser incorporada a revista Leonardo desde então e continua abordando temas nas interseções entre arte, ciência e tecnologia.

Publicada pela Cambridge University Press desde 1996 e mais voltada para questões musicais, a revista **Organised Sound** (<https://www.cambridge.org/core/journals/organised-sound>) também reúne pesquisas relevantes para as discussões entre arte e tecnologia.

Fazendo uma interface entre computação, música e matemática, a revista **Journal of Mathematics and Music** (<https://www.tandfonline.com/toc/tmam20/current>) é publicada pela Taylor & Francis desde 2007 e possui foco maior nos modelos matemáticos musicais.

As associações internacionais de computação, como a ACM (Association to Computer Machinery), também possuem periódicos específicos sobre este tema, como a revista **ACM Transactions on Multimedia Computing, Communications, and Applications** (<https://dl.acm.org/journal/tomm>) ou a **IEEE/ACM Transactions on Audio, Speech, and Language Processing** (<https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=6570655>).

No caso da IEEE (Institute of Electrical and Electronics Engineers), os destaques são as revistas **IEEE Multimedia Magazine**(<https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=93>), e **IEEE Signal Processing Magazine** (<https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=79>).

Outra associação, a European Association for Signal Processing (EURASIP), possui também algumas revistas acadêmicas que envolvem este tema, como a EURASIP Journal on Audio, Speech, and Music Processing (<https://asmp-urasipjournals.springeropen.com/>).

Exercícios 1.1

1. Complete a linha do tempo. Quais outros eventos históricos podem ser incluídos na nossa busca por datas entre música e tecnologia?	quisa em computação musical poderiam ser citados?
2. Quais outros grupos de pes-	3. Como soava os instrumentos Theremin, Ondas Martenot, Tratonium? Será que temos registros históricos destes sons?

1.2 O ensino de Computação Musical

O ensino na área de Computação Musical pode envolver a apresentação de técnicas computacionais com a teoria necessária para o desenvolvimento de trabalhos práticos na área. Pensamos aqui o ensino de maneira separada da pesquisa por acreditar que no contexto do ensino pode ser possível não inovar necessariamente mas criar oportunidades de ensino que, muitas vezes, apenas repetem experimentos já populares. Para pensar o ensino neste campo, vamos nos basear nos eventos citados anteriormente como possíveis grandes áreas que envolvem a área de computação musical. É importante notar que uma área possui interseção com as outras e que nem sempre é possível classificar um trabalho em apenas uma destas áreas, já que as fronteiras que as separam não são claras e tendem a ser borrosas ou indefinidas.

1.2.1 Criação e utilização de sistema musicais

A criação e utilização de sistemas musicais mais genéricos é tema de eventos como o Linux Audio Conference e Sound and Music Computing Conference e envolve o desenvolvimento de sistemas musicais. Esta área combina tecnologia e música por meio de ferramentas, softwares ou dispositivos que auxiliam na criação, performance, análise, ensino ou apreciação musical e envolve a aplicação de conhecimentos de computação, engenharia, design de interação e teoria

musical para projetar sistemas que interagem com o som e com usuários — músicos, compositores, ouvintes ou pesquisadores.

Tais práticas podem ser mais simples para o início de uma curso de computação musical por não envolver necessariamente um conhecimento mais aprofundado em música. Esta área pode incluir sistemas já existentes de composição e produção musical, sistemas para gerar ou acompanhar música automaticamente, sequenciadores, sintetizadores, plataformas de ensino musical interativas, ferramentas para análise de obras musicais, entre outras. Ferramentas para práticas musicais, como ferramentas de DJ, podem ser complexas e incluir, por exemplo, análise musical de tempo e tom, samples, efeitos e mapeamento de interfaces e podem ser uma possibilidade de prática para a introdução na área sem demandar, para isso, um alto conhecimento de teoria musical.

Dica de Software: Mixxxx



softwares pagos.

O Mixxxx é um software livre e gratuito para DJs, usado para mixar músicas ao vivo com recursos profissionais. Ele roda em Windows, macOS e Linux, e é bastante popular tanto entre DJs iniciantes quanto em setups profissionais que querem evitar

✎ Mais informações: <https://www.mixxxx.org/>

1.2.2 Recuperação de informação musical

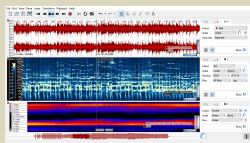
Uma grande área de pesquisa e ensino está associada ao ISMIR, que trata de **Recuperação de Informação Musical**. MIR (Music Information Retrieval) é uma área de pesquisa interdisciplinar que estuda como extrair, organizar e buscar informações relevantes a partir de dados musicais, especialmente em formatos digitais. Esta área une conhecimentos de ciência da computação, música, matemática, engenharia de som e inteligência artificial, e tem aplicações práticas tanto no mundo acadêmico quanto na indústria da música.

O objetivo principal desta área é a Escuta computacional, que permite que sistemas "entendam" e trabalhem com música de forma automática, ajudando a organizar, buscar, classificar e interagir com acervos musicais — muito além

do que apenas por metadados como o nome do artista ou da faixa.

As tarefas desta área incluem a classificação de músicas por gêneros; identificação de instrumentos em um arranjo; detecção de elementos musicais como batida (beat tracking), melodia, tons e acordes; reconhecimento automático de músicas e transcrição automática (áudio para MIDI); entre outras. Diversas bibliotecas como librosa, essentia, music21, liaubio e outras são comumente utilizadas neste campo de ensino, sendo utilizadas em ferramentas como, por exemplo, o Sonicvisualiser.

Dica de Software: Sonicvisualiser



O Sonic Visualiser é um software gratuito e open source usado para visualizar, analisar e interpretar arquivos de áudio. Ele é bastante popular em contextos de pesquisa em música, análise de som, musicologia, e ciência de dados sonoros.

➦ Mais informações: <https://www.sonicvisualiser.org/>

1.2.3 Novos Instrumentos musicais digitais

Outra grande área está associada ao **NIME** (New Interfaces for Musical Expression), ou em português Criação de Novos Instrumentos Musicais Digitais, é um campo de pesquisa e prática que investiga formas inovadoras de criar, tocar e interagir com música por meio de tecnologias digitais e une música, design, engenharia, ciência da computação, arte, psicologia, neurociência, e mais.

Esta área explorar novas formas de expressão musical que vão além dos instrumentos tradicionais — criando interfaces físicas, digitais ou híbridas que permitem performances musicais únicas, muitas vezes com gestos, sensores e algoritmos.

Dica de Software: PureData



O Pure Data (Pd) é um ambiente de programação visual voltado para criação de música, áudio interativo, vídeo e arte multimídia em tempo real. Ele foi criado por Miller Puckette no final dos anos 1990, como uma continuação de suas ideias do Max/MSP, outro ambiente semelhante, mas não aberto.

✉ Mais informações: <https://puredata.info/>

Apesar de ser possível criar instrumentos utilizando linguagens de programação de propósito geral, como Python ou C, algumas linguagens de programação musicais foram especialmente desenvolvidas para o processamento de som e por isso podem ser mais simples para desenvolver protótipos de Instrumentos musicais digitais (IMD), como Pure Data, Super Collider, Max/MSP e outras. O apêndice B traz uma revisão geral destas linguagens.

Um instrumento NIME pode responder a diferentes gestos, como a respiração, o olhar ou ondas cerebrais, e pode gerar sons que não existem na natureza, por meio de algoritmos de síntese computacionais. Também é possível partir de instrumentos existentes e adicionar, por meio de sensores e computação, novas características a estes instrumentos, os chamados **instrumentos ampliados**.

Algumas práticas com novos instrumentos incluem a criação de orquestras de notebooks, também chamadas de Laptop Orchestras, ou orquestras de dispositivos móveis, as mobile orchestras.

1.2.4 Efeitos de áudio digital

A área de pesquisa em Efeitos de Áudio Digital, conhecida como **DAFx** (Digital Audio Effects), está atrelada ao evento homônimo e estuda como manipular sinais de áudio digitalmente para alterar, aprimorar ou transformar o som, muitas vezes com fins musicais ou artísticos.

Ela envolve uma forte base em processamento de sinais digitais (**DSP**), e sua aplicação está presente em músicas, chamadas de vídeo e produções de cinema.

Os efeitos de áudio digital são algoritmos aplicados ao som para alterar características sonoras como timbre, reverberação, afinação, entre outras. Também é possível, por meio de efeitos digitais, criar novas texturas ou espacializações, simular ambientes acústicos e modificar gravações.

Dica de Software: LADSPA



O **LADSPA**[ladspa] (Linux Audio Developer's Simple Plugin API) é um padrão de plugins de áudio, criado para fornecer uma forma simples e padronizada de adicionar efeitos de áudio como reverb, delay, equalização, distorção, entre outros, a programas de áudio — especialmente no sistema Linux, mas também compatível com outros sistemas operacionais.

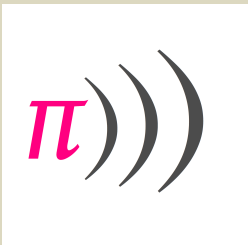
☞ Mais informações: <https://www.ladspa.org/>

Exemplos comuns de efeitos DAFX são: reverberação, delay / eco, equalização, distorções, alteração da altura (pitch shifting), modulação (flanger, chorus, phaser), Vocoder e Auto-Tune.

1.2.5 Composição algorítmica

A **Composição algorítmica** é uma área da música e da computação musical que investiga como usar algoritmos para criar música, seja automaticamente ou como apoio ao compositor humano e podemos pensar no ICMC como um evento de referência para a mesma. A composição algorítmica existe antes mesmo do computador por meio de dados, tabelas, regras, fórmulas, algoritmos ou procedimentos matemáticos/lógicos para gerar, manipular ou organizar materiais musicais — como melodias, harmonias, ritmos, formas ou texturas. No entanto, esta área ganhou novas possibilidades com a inteligência artificial e linguagens de programação voltadas à criação sonora e musical.

Dica de Software: SonicPi



O Sonic Pi é um ambiente de codificação ao vivo (live coding) voltado para criação de música com programação. Ele foi criado por Sam Aaron, com foco tanto na educação musical e de programação quanto em performances ao vivo.

↗ Mais informações: <https://sonic-pi.net/>

Estratégias comuns nesta área incluem o uso de estocástica para trabalhar com probabilidades computacional musical, o uso de gramáticas formais para modelagem musical, modelos Markovianos de probabilidades de transição entre eventos musicais, sistemas generativos, algoritmos genéticos, redes neurais, entre outras possibilidades.

Tal área também faz uso de linguagens de programação musical, como SuperCollider, Pure Data, Max/MSP e CSound, mas possuem linguagens específicas como ambiente visual de composição algorítmica OpenMusic ou a linguagem para Live Coding **Sonic Pi**.

Abordagem	Foco
Criação e utilização de sistema musicais	Introdução mais simples as ferramentas da área
Recuperação de informação musical	Ferramentas de análise computacional
Novos Instrumentos musicais digitais	Criação de novos instrumentos
Efeitos de áudio digital	Processamento de sinais e modificação de áudio
Composição algorítmica	Estratégica mais musical que envolve a composição

Tabela 1.2: Abordagens para o ensino de computação musical

Exercícios 1.2

4. As áreas de ensino aqui apresentadas trazem possíveis separações entre atividades práticas da área de computação musical mas tais separações não existem entre os conceitos envolvidos na área. Partindo disso, quais interseções podem existir entre a recuperação de informação musical (MIR) e a criação de novos instrumentos musicais digitais (NIME)?
5. Como a área de DAFx pode ser usada para a criação de novos instrumentos musicais (NIME)?
6. Como a análise da entrada do sinal (MIR) pode ser incorporada na criação de novos efeitos digitais de áudio (DAFx)?
7. Como a criação de novos instrumentos musicais pode estar associada a composição algorítmica? Apresente uma possibilidade de prática que envolva tanto a criação de novos instrumentos quando de composição.
8. Quais elementos de MIR podem ser estudados e estão presentes em ferramentas de DJ, como o Mixxx?

1.3 Computação Musical e interdisciplinaridade

A computação musical é uma área de pesquisa interdisciplinar que envolve áreas como artes, ciência da computação, matemática, acústica e engenharia de som pois os conceitos destas áreas são usados para a criação, análise, reprodução e execução de música por tecnologias digitais. A Figura 1.2 apresenta algumas possibilidades interdisciplinares desta área.

A parte de conhecimento que discute os impactos do som no ser humano e a relação deste com nossos sentidos, pode pedir conceitos da **biologia**, **psicoacústica** e também da **psicologia**, **musicoterapia**, e da **neurociência**, com a percepção de fenômenos físicos a partir dos sentidos humanos. Como o ouvido humano transforma ondas sonoras em impulsos nervosos via cóclea, ou o papel do córtex auditivo na decodificação de frequências, timbres e localização espacial do som são perguntas importantes para o estudo em computação musical mas cujas respostas estão nestas outras áreas.

Ao digitalizar um som, a preocupação inicial com sua taxa de amostragem se dá por um entendimento da **percepção** subjetiva do som, definida pelos

limiares de audição humana. Da mesma maneira, formato de áudio compactados, como o MP3, se utilizam de efeitos psicoacústivos como mascaramento e sensibilidade a consonância e dissonância de frequências.

Se o som parte do ser humano, a voz falada e cantada, é estudada pela área da **fonoaudiologia**, enquanto o entendimento de como funciona a audição humana é campo de estudo da **otorrinolaringologia**, que também se utiliza de conceitos e ferramentas computacionais para medir e analisar a audição e a voz humana.

Entendendo o som como um fenômeno físico nos leva a áreas como da **acústica**, **arquitetura**, **engenharia civil**, **engenharia mecânica**, **engenharia acústica** e também da acústica de instrumentos musicais, como a **lutheria**, ou mesmo da **medicina**, que utiliza a propagação acústica para realizar exames de ultrassonografia. Também temos uma relação muito grande com a **física** e com a **matemática** pois é necessário uma base teórica nestas áreas para explicar diversos fenômenos sonoros que não são explicáveis facilmente de forma empírica.

Já o processamento de sinais envolve áreas como **engenharia elétrica**, **telecomunicações**, e novamente **física**. A física do som (frequência, amplitude, timbre) é modelada digitalmente usando técnicas como síntese aditiva, subtraiva ou FM (Frequency Modulation). Também é utilizado para a criação de efeitos digitais como reverberações, atrasos, distorção e filtros.

O impacto da computação musical na nossa cultura também pode nos levar para discussões nos campos das **ciências humanas**, da **sociologia**, da **filosofia** e de outras áreas que podem nos ajudar a discutir e pensar a cultura como algo que hoje é indissociável da tecnologia.

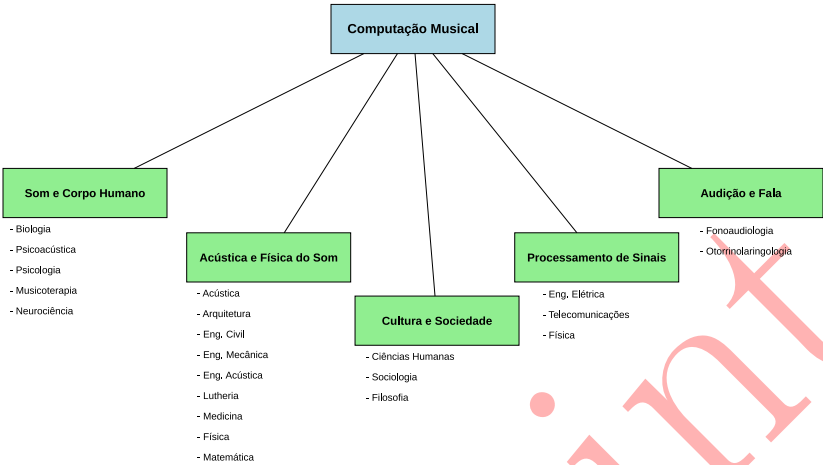


Figura 1.2: Representação da interdisciplinaridade da computação musical

Exercícios 1.3

- 9. Enquanto a acústica trata do som como um todo, a psicoacústica se preocupa com a nossa percepção do som. Como conceitos de psicoacústica podem influenciar em práticas de áudio?
- 10. Há na internet alguns varredores de frequência (freq sweep)

que serviriam para realizar testes auditivos e testar o limite da audição de uma pessoa. No entanto, tais testes podem não ser exatamente precisos. O que impede que tais sistemas sejam realmente eficientes? Quais condições seriam necessárias para que estes testes fossem realmente válidos?

1.4 Interdisciplinaridade na música

A música possui diversas áreas que podem se valer de ferramentas computacionais para sua prática. A primeira ferramenta, hoje quase imprescindível para práticas musicais são as de **Representação e Notação Musical Digital**. Formatos como **MIDI**, **MusicXML** ou **LilyPond** se tornaram ferramentas práticas para notação de música mesmo que estas músicas não sejam executadas por computadores.

Tais formatos digitais permitem a criação de grandes bases de dados (data-

set) para uma **musicologia digital** e a **análise musical computacional**, cujos elementos de análise podem ser explorados e definidos a partir de ferramentas computacionais com o uso de algoritmos para identificar padrões harmônicos, melódicos e rítmicos, por exemplo. Ferramentas como librosa ou MIRtoolbox podem ser usadas para estes fins. Ainda Musicologia Digital, Ferramentas como **MuseScore** podem ser usadas para reconstruir obras históricas e na preservação de Preservação de Acervos de partituras por meio da digitalização de obras históricas. Além deste campo, há hoje pesquisa em uma **Etnomusicologia Computacional** que permite registrar e analisar músicas tradicionais e classificar escalas e ritmos de culturas diversas e que fogem ao padrão musical ocidental.

A área de **composição** também tem se utilizado da computação há décadas e tornou o computador um novo instrumento musical. Esta área, também chamada de CAC (Computer-Assisted Composition), envolve tanto a manipulação direta do som, por meio de processamento de sinais, como a manipulação de representações musicais como partituras digitais. Com isso, é possível combinar técnicas como cadeias de Markov ou modelos baseados em inteligência artificial para criar novas músicas, tanto como áudio quanto como partituras. Além disso, sistemas computacionais baseados em **linguagens de programação musical**, como **Max/MSP** ou **Pure Data**, podem ser usados para composição algorítmica ao vivo. Diversas ferramentas computacionais para a composição foram criadas ao longo dos anos e podemos até dizer que a computação musical nasceu desta possibilidade de usar o computador como instrumento musical. Estilos musicais como a música eletroacústica têm se valido de recursos computacionais e de ferramentas para ampliar as possibilidades composicionais para além dos timbres e sons que os instrumentos convencionais possuem.

A área de **performance musical** tem utilizado de recursos computacionais para comparar duas execuções de uma mesma peça e verificar quais elementos acústicos tornam uma performance mais ou menos interessante, por meio de ferramentas de análise da performance. Novamente, ferramentas de recuperação de informação musical (MIR) podem ser usadas para este propósito e tornado possível utilizar ferramentas computacionais para medir os micro elementos da construção da performance musical como vibratos, durações de notas, alturas musicais e outras. Já a prática musical com ferramentas musicais pode explorar a criação e desenvolvimento de instrumentos musicais digitais ou mesmo de novos paradigmas de performance, como, por exemplo, a prática de criação

musical por código, também chamada de **Live Coding**.

Os formatos de representação musical, como MIDI, também auxiliam áreas como a **teoria musical**, que têm se valido de ferramentas computacionais para escrita e correção de contra-pontos, por exemplo, ou para a criação de arranjos e orquestrações. A música é baseada em estruturas matemáticas como escalas, ritmos, acordes e harmonia e a computação pode permitir a criação de algoritmos para gerar ou analisar essas estruturas.

Outra área que pode ter uma boa parceria com a computação musical é a da pedagogia musical e da **musicalização**, que pode fazer uso de ferramentas computacionais para o ensino de música. Neste campo, ferramentas para análise de ritmo e afinação em tempo real ou mesmo a criação de ambientes computacionais que gamificam o aprendizado musical se tornam bastante atrativos tanto para professores quanto para estudantes.

Exercícios 1.4

11. Além do MuseScore, quais outras ferramentas para edição de partitura você conhece?	12. Quais ferramentas computacionais os estudantes de música podem utilizar nas suas atividades diárias?
--	--

1.5 Interdisciplinaridade na Computação

A computação musical pode ser uma área relevante de pesquisa para outras áreas da computação.

1.5.1 Sistemas operacionais e computação de alto desempenho

Sistemas musicais são exemplos de sistemas em tempo real cuja latência irá definir a qualidade do serviço oferecido por uma determinada aplicação. Em música, entendemos como latência o tempo entre modificar um parâmetro musical, como tocar uma nota, e aguardar seu som ser modificado. No entanto, nem sempre é possível definir a latência de uma aplicação musical, pois a mesma irá depender de como o escalonador do sistema operacional irá definir as prioridades para a troca de contexto entre diferentes processos.

Sistemas em tempo real são exemplos comumente utilizados para pensar o escalonamento de processos em um Sistema Operacional preemptivo, e configurar um sistema musical para funcionar em tempo real pode ser um interessante projeto para estudantes de computação que querem se aventurar nesta área, mas que não conseguem encontrar exemplos de aplicação em tempo real que sejam facilmente perceptíveis.

1.5.2 Redes de computadores e as orquestras de computadores

A prática musical em conjunto por meio de dispositivos computacionais já é popular, sendo chamadas popularmente de Orquestras de computadores portáteis (“Laptop orchestra” - LOrc). Estas orquestras têm permitido a pesquisa e criação de novos instrumentos em diversas Universidades no mundo em projetos de pesquisa que transitam na fronteira interdisciplinar entre computação, engenharia, telecomunicação e música.

Estas orquestras tem trabalhado a questão da latência da comunicação entre dispositivos, por meio de protocolos de comunicação de rede, para permitir a prática musical coletiva com baixa latência.

Além das orquestras de computadores, há também uma área de pesquisa que trabalha concertos musicais geograficamente distribuídos, a chamada Música em rede. Para que tais concertos sejam possíveis, é necessário que a latência da comunicação seja extremamente baixa e isso envolve tanto conhecimentos computacionais, para diminuir a latência na comunicação, quanto conhecimentos musicais, para lidar com a latência caso ela seja inevitável.

1.5.3 Interface Humano computador

Chamamos de IHC a uma área da ciência da computação que estuda como as pessoas interagem com computadores e sistemas digitais. As pesquisas nesta área tem como o objetivo tornar essa interação mais eficiente, intuitiva e natural e envolve questões de design, usabilidade, ergonomia, psicologia cognitiva, entre outras.

A música, por ser uma forma complexa de expressão humana, oferece muitos desafios e oportunidades para a IHC, que vão desde a representação de conceitos musicais em ferramentas computacionais até a codificação criativa de novas interfaces musicais computacionais.

A criação de interfaces físicas de instrumentos musicais e a geração de som por meio de computadores nos leva para uma área de pesquisa que envolve microeletrônica, programação e música que é a área de criação de novos Instrumentos Musicais Digitais (IMD) ou novas interfaces para criação musical (NIME - New Interfaces for Musical Expression). Estes instrumentos trazem uma possibilidade quase infinita de criação sonora e de exploração e permitem pesquisar sensores, atuadores, mapeamentos e novas possibilidades de interfaces para criação de música.

A área de IHC também tem contribuído imensamente com a questão e o debate sobre a inclusão de pessoas com deficiências no contexto da arte digital.

1.5.4 Micro eletrônica e Eletrônica digital

A linha do tempo que abre este nosso capítulo mostra que a música moderna — principalmente a eletrônica, pop e experimental não existiria como conhecemos sem a microeletrônica e a eletrônica digital. Esta área está intrinsecamente amarrada com a forma que produzimos e consumimos música na atualidade.

A geração de som por meio de computadores nos leva para uma área de pesquisa que envolve microeletrônica, programação e música, novamente permitindo explorar a criação de novos Instrumentos Musicais Digitais (IMD) ou novas interfaces para criação musical (NIME - New Interfaces for Musical Expression) como oportunidade de aprender e ensinar microeletrônica. Estes instrumentos trazem uma possibilidade quase infinita de criação sonora e de exploração e permitem pesquisar sensores, atuadores, mapeamentos e novas possibilidades de interfaces para criação de música.

1.5.5 Computação Móvel

Com a evolução da tecnologia móvel, os aparelhos celulares e “smartphones” se tornaram computadores portáteis presentes na vida cotidiana de muitas pessoas. O objetivo inicial destes dispositivos era a comunicação e hoje os mesmos cumprem este papel ultrapassando a simples comunicação de voz e permitindo a comunicação por outras mídias como textos, fotos, documentos e vídeos. A expansão de tais limites comunicacionais foi possível graças à evolução da capacidade dos dispositivos principalmente no que tange à conectividade, à capacidade de processamento aliada à possibilidade de manipulação de materiais multimídia e sua capacidade de armazenamento.

A utilização de aparelhos celulares para práticas musicais é vista como a evolução das orquestras de computadores portáteis (Laptop orchestras) devido à simplicidade, capacidade e praticidade de tais dispositivos. No entanto, há uma carência de pesquisa e desenvolvimento de aplicativos que permitam a interação de músicos por meio de dispositivos móveis em tempo real.

1.5.6 Linguagens de programação musical

A programação de sistemas computacionais com música pode ser feita por meio de linguagens de propósito geral (*GPL - General Purpose Language*) que permitem o desenvolvimento de sistemas de software para este propósito desde que aumentadas com recursos específicos para esta área que proporcionam ao ambiente de execução a capacidade de acessar os dispositivos de som do computador.

No entanto, há também a possibilidade de se utilizar as linguagens específicas de domínio (*DSL - Domain Specific Language*) que são linguagens elaboradas com base nos conceitos desenvolvidos em arte digital e mais especificamente em computação musical. Este segundo grupo de linguagens possui diversas características que as tornam especialmente atraentes para o desenvolvimento de aplicações computacionais para a música. Um exemplo destas características é a capacidade de processamento em tempo real, imprescindível para a sincronização de eventos musicais e nem sempre disponível em linguagens de propósito geral.

Há na área de Computação Musical diversas linguagens de programação musical e elas podem agregar diferentes possibilidades de programar chegando a possibilidade de utilizar a programação como expressão artística em uma prática chamada de **live coding**, que trata a performance musical em tempo real por meio da programação.

1.5.7 Inteligência Artificial

A possibilidade de explorar questões musicais no computador quase sempre irá esbarrar na questão da criatividade. Neste ponto que a Inteligência Artificial se tornou uma ferramenta poderosíssima capaz de auxiliar músicos a compor e de nos auxiliar a entender como se dá a criatividade humana em tarefas musicais.

Se misturamos a capacidade de aprendizagem da IA com processamento de sinais, chegamos na área de Recuperação de informação Musical. A pesquisa nesta área tem permitido analisar parte do acervo musical de diferentes gêneros

e extrair características que nos permitem compreender questões que a análise musical não computacional pode não perceber devido a uma diferença na própria representação musical utilizada nestas análises.

1.5.8 Web e música em HTML

Desde o advento do HTML 5 que a síntese sonora em navegadores se tornou possível. A API, chamada *webaudio*, transformou os navegadores web em potentes máquinas de síntese prontas para serem exploradas por programadores que querem incluir processamentos de som em páginas de Internet. Assim, criar peças musicais e distribuí-las por meio do seu código se tornou viável com esta tecnologia. Esta versão do HTML também inclui uma API para manipulação de sinais MIDI, chamada *webmidi*, e permite criar interfaces entre páginas web e controladores musicais. Desde então, diversas ferramentas, bibliotecas e componentes surgiram para tornar o navegador um espaço para a criação musical, culminando em um evento acadêmico a respeito do tema, o chamado Web Audio Conference (<https://webaudioconf.com/>).

1.5.9 Desenvolvimento de jogos

Difícilmente podemos imaginar jogos sem sons. Nos jogos eletrônicos ou digitais o som é parte da percepção do espaço e a possibilidade de criar trilhas e paisagens sonoras únicas podem trazer para esta área o interesse em pesquisar computação musical. Isso permitiria, por exemplo, que um jogo tivesse uma trilha sonora não gravada mas criada em tempo real dependendo da situação envolvendo o personagem e o jogador. Os contextos do jogo podem influenciar a trilha sonora do mesmo os parâmetros do jogador podem servir como elementos de controle de um sintetizador que faz música totalmente reativa em tempo real. Para isso, podemos utilizar motores de síntese que aceitam parâmetros do jogo e cuja resposta sonora depende de condições estabelecidas pelo ambiente. É necessário, neste caso, um tanto de conhecimento de composição algorítmica para permitir soluções criativas nesta seara.

1.5.10 Computação Gráfica e Processamento digital de Imagens

Os algoritmos de Processamento Digital de Imagens são muito similares aos algoritmos de processamento sonoro. Algoritmos como FFT, Transformada Wavelet, Transformada discreta do cosseno, e outros processamentos são totalmente similares nestas duas áreas. Outros conceitos comuns a ambas as áreas, como filtros e síntese, também são compartilhadas entre computação musical e processamento digital de imagens.

Já a representação de imagens na computação gráfica por meio de descritores de imagens, como PDF e SVG, é muito similar a representação musical simbólica em sistemas como MIDI ou musicXML. Em ambos os casos, temos uma possibilidade de representação simbólica que depende de um software para visualizar ou ouvir o que está sendo representado.

1.5.11 Visão Computacional e OMR

Quão interessante seria poder apontar uma câmera para uma partitura e ter um software capaz de tocar a música escrita nesta pauta? A área de estudo chamada Optical Music Recognition é a área da visão computacional que trata do reconhecimento óptico de partituras e a pesquisa na área tem enfrentado há muitos anos o desafio de entender partituras. O nome remete ao uma visão musical do já conhecido reconhecedor de caracteres, ou OCR - Optical character recognition, o tipo de software que permite ler texto presente em imagens que se tornou o terror de quem desenvolve captchas para sites. Por mais incrível que pareça, ler uma partitura visualmente não é uma tarefa fácil.

A área de visão computacional está hoje também bastante associada a criação de novos instrumentos musicais, a análise da performance musical e a práticas de criação musical relacionadas com outras áreas como teatro e dança.

Exercícios 1.5

13. Além das áreas citadas, quais outras áreas da computação podem se relacionar com a computação musical?

14. A ciência da computação se co-

loca como ciência e a música é uma área das artes. No caso da computação musical, ela se encontra como ciência ou como arte?

♦ Respostas dos exercícios do capítulo 1.

1. Diversos outros elementos podem ser interessantes como lançamentos musicais. O lançamento do primeiro disco da banda Kraftwerk, ou da banda DaftPunk poderiam ser interessantes. Também pode ser interessante a data do lançamento de instrumentos como o DX7 da Yamaha ou a reactable. Poderíamos ainda incluir a data da primeira edição dos eventos da área de computação musical ou a data de lançamento das revistas acadêmicas da área.
2. Há pesquisa em computação musical em muitas outras universidades no Brasil e no mundo e possivelmente uma busca por isso pode apresentar um panorama ainda mais amplo da área.
3. A arqueologia musical permite encontrar alguns destes instrumentos ainda funcionando e também a gravação de alguns deles. Há também sintetizadores digitais que foram feitos a partir da descrição dos componentes destes instrumentos e que permitem emular o som que os mesmos faziam.
4. A área de MIR pode ser usada para reconhecimento de padrões gestuais e este reconhecimento ser utilizado como controle para um parâmetro de síntese em um IMD novo. Também é possível pensar em usar sons para controlar o som do sintetizador de um novo DMI.
5. Os efeitos de áudio são parte importante da construção das vozes dos instrumentos musicais digitais e podem ser parte de seu sintetizador.
6. Alguns efeitos de áudio, como os processadores de dinâmica (Compressor, noise gate, expandre, limiter) dependem da análise do áudio da entrada para controlar seus parâmetros.
7. Uma das possibilidades de composição está na composição que já prevê a performance ao vivo, cujos parâmetros podem ser controlador por alguma interface física. Tal interseção torna complexo de definir se tal sistema é uma composição ou um instrumento.

8. Ferramentas de DJ costumam ter algoritmos prontos para identificação de beat (BPM) e de tom, para suavizar a troca entre faixas.
9. As curvas de Fletcher-Munson tratam da percepção sonora de intensidade relacionada com diferentes frequências e podem ser utilizadas para calibrar o som de um sistema de áudio.
10. Os testes precisam levar em consideração não apenas a sua capacidade de perceber uma determinada frequência no som mas também a capacidade do sistema de reproduzir esta frequência. Não identificar uma frequência pode significar que seu fone de ouvido, por exemplo, não consegue reproduzir áudio nesta frequência e estes testes só são exatamente válidos quando possuem um sistema adequado para tal função.
11. Há diversas ferramentas para este fim, entre elas algumas proprietárias como o Finale ou o Sibelius.
12. Ferramentas como afinadores, metrônimos e leitores de partitura são comuns na vida do músico. Também é comum utilizar áudio digital, MP3, e também utilizar de canais de stream para consumir ou distribuir material musical.
13. Diversas outras áreas poderiam ser aqui abordadas como Compiladores, Grafos, Matemática discreta e outras áreas da computação que podem ser utilizadas para explicar e entender conceitos musicais.
14. Não é a intenção aqui resolver esta pergunta mas fomentar a discussão pode ser interessante. O que envolve técnica e metodologia científica é entendido como ciência. No caso da computação musical, muitas questões são técnicas e por isso científica. Há também questões que envolvem decisões estéticas e, neste caso, tais soluções envolvem o campo das artes.

Pre Print
V.0.0.1

Capítulo 2

Elementos da Música Ocidental

Lecionando a disciplina da Introdução à Computação Musical no curso de Ciência da Computação da Universidade Federal de São João del-Rei, eu sempre digo para a turma que não é necessário conhecer teoria musical para fazer música com o computador. No entanto, acredito que em todas as turmas sempre aparecia uma ou mais pessoas me pedindo ajuda com a tal teoria musical, ávidos para conhecer um tanto sobre as tais notas musicais. A razão de eu afirmar que não é necessário conhecer teoria musical é que estes conceitos presentes na teoria musical estão fortemente associados a música de tradição europeia, cuja notação musical se dá por meio de partituras. Certamente há outras músicas que não se baseiam nestes conceitos.

Como este livro é voltado para o público geral, como profissionais e estudantes de computação e outras áreas, assumiremos que parte de nossos leitores não possuem formação em música e que podem querer conhecer um pouco deste campo. Por isso, neste capítulo traremos alguns elementos da música ocidental que podem ser importantes para o entendimento das próximas seções deste livro e, mais do que isso, para fazer música. Tais conceitos são importantes para trabalhar a representação digital de músicas relacionados à partituras e notações simbólicas de música. No entanto, é importante deixar claro que este livro não deve ser usado para embasar discussões na área da música e que há uma vasta literatura que cobre Teoria musical, história da música e outros conhecimentos que serão brevemente apresentados neste capítulo.

Apesar de a teoria musical envolver toda uma seara de conhecimento, vamos aqui apresentar brevemente os elementos básicos para fazer música ocidental no computador: Tempo musical, Dinâmica musical, Altura musical e Timbre.

Quando pensamos em música ocidental, em ensino e aprendizagem de música ou em educação musical, a figura da partitura sempre aparece como um símbolo desta música. Assim, vamos iniciar este capítulo trazendo uma visão geral desta notação da música.



Figura 2.1: Notação na partitura do andamento

A partitura é uma notação gráfica da música. Esta notação gráfica inclui a notação dos elementos básicos da música: tempo, dinâmica e altura, mas

Colocar aqui uma visão geral do que será visto neste capítulo.

2.1 Tempo musical

A primeira questão que pode ser representada na música é o tempo musical. Uma maneira interessante entender a música de forma completamente genérica é assumir que a música é a arte de organizar eventos musicais no tempo.

O tempo musical é uma espécie de velocímetro que marca a velocidade ou rapidez em que os eventos musicais soam e se repetem. A humanidade tem o costume de medir e marcar o tempo e, para isso, usamos nossos relógios mecânicos, digitais ou mesmo orgânicos como a pulsação cardíaca, frequência respiratórias ou ciclos lunares e solares. No fim, tudo isso mede a mesma coisa: tempo. Entendemos que boa parte das músicas possuem uma pulsação que pode ser relacionada a estes relógios, uma marcação contínua e cíclica do tempo que permite a organização dos eventos musicais no tempo de forma relativa.

Para esta seção de nosso livro, sugerimos utilizar uma ferramenta de escrita rítmica, como um sequenciador, para acompanhar os exemplos aqui apresentados e fazer os exercícios. Nossa sugestão é utilizar o LMMS para isso mas qualquer outra ferramenta similar poderia ser utilizada.

Dica de Software: LMMS



O LMMS (Linux Multimedia Studio) é uma DAW (Digital Audio Workstation) focada na produção, criação e composição de músicas. O LMMS é open-source e funciona em Linux, Windows e MacOS. A ferramenta possui diversos sintetizadores, efeitos de áudio, e uma grande comunidade de usuários e desenvolvedores sendo comumente comparada a soluções proprietárias como o FL Studio.

➦ Mais informações: <https://lmms.io/>

2.1.1 Andamento

O **andamento** de uma música é o tempo geral dela, ou o tempo de pulsação dos eventos musicais. Se compararmos o tempo musical com nossa pulsação cardíaca podemos dizer que uma pulsação mais lenta é mais calma enquanto uma pulsação acelerada é mais agitada. Em execuções musicais, este andamento pode ser marcado por um instrumento chamado metrônomo, como apresentado na Figura 2.2.

O metrônomo marca o tempo em batidas por minuto, ou bpm, e os mais antigos funcionam como um relógio de cordas. Um metrônomo configurado para marcar 60bpm irá bater sessenta vezes por minuto, ou uma vez por segundo enquanto o metrônomo de 120bpm irá soar 2 vezes por segundo. Quanto maior o andamento, mais rápido uma música irá soar. Apesar de um metrônomo dar um tempo preciso, os andamentos musicais recebem nomes que compreendem faixas de tempo e que remetem a sentimentos. Os tempos musicais e seus respectivos BPM são apresentados na Tabela 2.1

Na partitura, o andamento é anotado por meio de seu nome, no início da pauta, como apresentado na Figura 2.3.

O andamento de uma música pode ainda mudar e acelerar ou diminuir em algum trecho. Isto é notado na partitura como uma instrução *accelerando* ou *rallentando*, como mostrado na Figura 2.4.

Note que as mudanças não indicam com precisão qual andamento a música

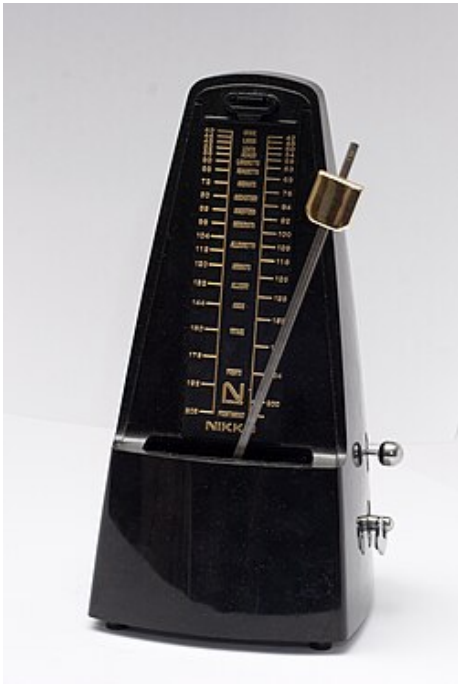


Figura 2.2: Metrônomo de corda



Figura 2.3: Notação na partitura do andamento



Figura 2.4: Notação de mudanças de andamento

terá depois de ser acelerada ou desacelerada. Este é um ponto que o compositor deixa a decisão para o músico definir em sua performance musical.

Calcular tempo como exercício logo no começo

Andamento	BPM
Gravissimo	19 ou menos
Grave	20-40
Larghissimo	40-45
Largo	45-50
Larghetto	50-55
Adagio	55-65
Adagietto	65-69
Andantino	78-83
Marcia Moderato	83-85
Andante	75-107
Andante Moderato	90-100
Moderato	108-112
Allegro Moderato	112-115
Allegro ma non troppo	116-119
Allegro	120-139
Vivace	140-159
Vivacissimo	160-169
Alegricissimo	168-177
Presto	180-200
Prestissimo	200 ou mais

Tabela 2.1: Nome dos andamentos musicais e seus respectivos BPM

Exercícios 2.1

1. Crie um metrônomo visual onde o tempo é marcado de forma visual ou sonora. Permita que o mesmo seja sincronizado com outros metrônomos em rede.
2. O metrônomo é uma ferramenta musical bastante controversa e existe até peças em sua homenagem. Recrie a peça "Poema sinfônico para 100 metrônomos", de György Ligeti.

2.1.2 Compasso

Um andamento define quando a pulsação musical irá ocorrer, como, por exemplo, cada 500ms, ou 0.5s ou 120 bpm. Musicalmente, utilizamos uma repetição cíclica destas pulsações e o agrupamento das mesmas em música é chamada de **compasso**. Um compasso pode ter um número fixo de pulsações como e alguns metrônomos permitem que seja definida a fórmula de compasso e fazem um som distinto para o início de um compasso. Na música popular, em estilos

como o rock, baião e o blues, o compasso possui sempre 4 tempos. Assim, o metrônomo pode fazer um som diferente para o primeiro pulso e outros 3 sons iguais para os outros de forma a permitir identificar mais facilmente onde está o início de cada compasso. Em valsas, contamos 3 tempos em um compasso e no samba temos um compasso de 2 tempos.

Devido a estas divisões, os tempos de um compasso podem ser binários, quando possuem dois tempos, ternários, quando possuem três tempos, ou quaternários, quando possuem quatro tempos. Compassos que possuem outra quantidade de tempos são chamados de compasso irregular, complexo ou alternado. Eles existem mas não são muito comuns na música popular, portanto, não iremos abordá-los por aqui.

As pulsações marcadas pelo metrônomo podem também possuir divisões e subdivisões, como eventos temporais que acontecem entre duas batidas do metrônomo. Assim, um compasso pode ter 4 tempos e estes tempos serem divididos em 2, 4, 8 ou 16 subtempos. Esta subdivisão é representada por uma fórmula onde o dividendo é o número de tempos do compasso e o divisor é a figura que irá ocupar um tempo do compasso. Estas figuras serão apresentadas nas próximas seções.

Na partitura, a notação do compasso da música é feita no início da pauta ou, no meio, caso ela mude sua fórmula de compasso. A Figura 2.5, mostra uma música que inicia com um compasso 4/4 e depois muda o compasso para 3/4.



Figura 2.5: Partitura com mais de um compasso

Além desta notação, é possível também utilizar o símbolo C para marcar o compasso 4/4, como ocorreu nas figuras 2.4 e 2.3. A Figura 2.6 mostra algumas fórmulas de compasso e sua relação com as figuras de tempo.

Os compassos podem ser chamados de simples, quando os tempos podem ser agrupados em grupos de dois, ou composto, quando são agrupados em grupos de três. Compassos 3/4 são compassos simples enquanto compassos 6/8, as baladas, são compassos compostos por serem agrupados em 2 grupos de 3 tempos.

Fórmulas de compasso			
$\frac{2}{4}$	$\frac{3}{4}$	$\frac{4}{4}$ ou C	$\frac{3}{8}$
Unidade de Tempo 			
Unidade de Compasso 			

Figura 2.6: Fórmulas de compasso

Exercícios 2.1

- | | |
|---|--|
| <p>3. Identifique onde no LMMS é definido a fórmula de compasso.</p> <p>4. Modifique seu compasso para que ele aceite como entrada o andamento e uma notação de figuras de tempo forte e fraco.</p> | <p>Sua saída deverá ser uma marcação de tempo que mostra elementos distintos nos tempos fortes e fracos.</p> <p>5. Encontre músicas que você conhece que usam fórmulas 4/4, 3/4 e 6/8.</p> |
|---|--|

2.1.3 Figuras de Tempo musical

Apesar de esta visão de grade utilizada em sequenciadores ser muito efetiva para escrever música no computador, a escrita musical com partitura utiliza um grupo de símbolos para representar os tempos musicais para representar a duração de uma nota ou de um silêncio. Estas figuras não possuem uma duração absoluta pois sua duração é relativa

Estas figuras possuem nomes, conforme apresentado na Tabela 2.2.

Assim, quando chamamos um compasso de 4/4 significa que o mesmo terá 4 tempos e cada tempo será ocupado por uma semínima, figura que vale 1 de tempo. Um compasso 4/8 também terá 4 tempos mas cada tempo será ocupado por uma colcheia, que vale 1/2 de tempo.

Caso seja necessário escrever um evento musical que dura 3 tempos de um compasso 4/4, podemos utilizar duas estratégias. A primeira é “ligar” duas figuras, combinando uma mínima com uma semínima, por meio de um símbolo chamado ligadura. A segunda é utilizar um ponto após a figura, que indica que ela terá a duração acrescida por 50% de sua duração original. Ambos os casos

-	Nome	Duração	Figura	Pausa
1	Semibreve	4 tempos	○	■
2	Mínima	2 tempos	♪	■
4	Semínima	1 tempo	♪	♪
8	Colcheia	1/2 tempo	♪	♪
16	Semicolcheia	1/4 tempo	♪	♪
32	Fusa	1/8 tempo	♪	♪
64	Semifusa	1/16 tempo	♪	♪

Tabela 2.2: Figuras musicais, seus tempos e pausas, em um compasso 4/4

são apresentados na Figura 2.7.

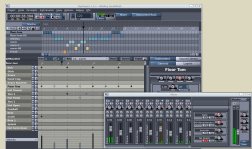


Figura 2.7: Ligadura e figura pontuada

Além de notar eventos musicais, é possível também notar a ausência dos eventos, ou os silêncios da música. Estas ausências de sons são especialmente importantes quando tratamos de instrumentos ou sons que possuem sustentação. A corda de um violão, por exemplo, irá soar por um determinado tempo após ser tocada e o músico pode interromper esta vibração abafando as cordas e impedindo a mesma de continuar soando. O som de um instrumento de sopro, como uma flauta ou saxofone, pode soar por muito tempo e a interrupção do som é feita de maneira intencional pelo musicista.

Assim, para instrumentos que soam por um curto período de tempo, como os instrumentos de percussão, a representação do silêncio pode ser menos importante do que para instrumentos que possuem sustentação, como os instrumentos de sopro ou os instrumentos de corda com arco, como o violino, que possuem sustentação.

Dica de Software: Hydrogen



O Hydrogen é um software gratuito e open source usado para sequenciar baterias e percussões. Disponível para Linux, Windows e MacOS, ele possui diversos kits de bateria que funcionam com samples de instrumentos reais e por isso trazem um som

bastante orgânico. O Hydrogen tem uma interface simples e bastante similar ao LMMS no que diz respeito ao seu editor de padrões e músicas.

✎ Mais informações: <http://hydrogen-music.org/>

Exercícios 2.1

6. Quantos milissegundos dura cada figura de tempo em uma música cujo andamento é de 120 bpm?

7. Quantos milissegundos dura cada figura de tempo em uma música cujo andamento é de 200 bpm?

2.2 Dinâmica Musical

O segundo atributo musical que iremos apresentar é a dinâmica. A dinâmica diz respeito a quão forte ou quão fraco um evento musical, como uma nota ou toque de tambor, irá soar. Da mesma forma que o tempo musical é medido em segundos e milissegundos mas não usamos tal medição, a dinâmica musical pode ser medida em decibéis, como a relação entre o “volume” do som e o silêncio, mas também não utilizamos tal medição. Em vez disso, a dinâmica musical pode ser medida de forma relativa, onde um evento é notado de forma mais acentuada do que outro. Nestes casos, adicionamos um símbolo de maior (>) em cima do evento mais forte, ou um símbolo de menor (<) sobre o evento mais fraco, como apresentado na Figura 2.8;

Além desta notação pontual, podemos ter outra notação textual na partitura para uma indicação dinâmica para um trecho musical. Para isso, são usados termos como

♪ **ppp** - Pianississimo ou Molto Pianissimo: O mais suave possível, quase



Figura 2.8: Exemplo de dinâmica em uma partitura de percussão

imperceptível.

♪ **pp** - Pianíssimo: tocado de forma muito suave.

♪ **p** - Piano: suave, o som natural do piano tocado com leveza.

♪ **mp** - Mezzo Piano: um primeiro aumento de intensidade em relação ao piano tocado com leveza.

♪ **mf** - Mezzo Forte: som mais pesado do que anterior, sendo tocado com mais força.

♪ **f** - Forte: Som efetivamente pesado.

♪ **ff** - Fortíssimo: Mais forte do que o forte.

♪ **fff** - Molto Fortissimo: a maior força possível e aplicável ao instrumento.

Além disso, a dinâmica pode variar de um trecho para outro. Neste caso, aplica-se sinais como Crescendo (cresc), Diminuendo (dim) ou Sforzando (sfz), para indicar que um determinado trecho possui um aumento ou diminuição de intensidade entre suas notas ou eventos musicais.

Exercícios 2.2

8. Como é possível modificar a dinâmica no LMMS?
9. Quantas dinâmicas diferentes será que um músico é capaz de fazer intencionalmente? Desafie seus colegas a tocar, de

forma intencional, seja batucando na mesa, ou cantando, com diferentes intensidades. Será possível repetir tais intensidades em diferentes execuções musicais?

2.2.1 Ritmo

O compasso define quantos pulsos teremos em um ciclo musical sendo o mais comum, em música popular, compassos com 4 tempos. O compasso quaternário é utilizado no Rock, Jazz, Baião, Ijexá, Xaxado, e mesmo o samba pode ser escrito em compassos quaternários. Assim, o que define um ritmo não apenas é seu compasso mas a intensidade destes pulsos. A intensidade de um tempo dentro do compasso, ele ser forte ou fraco, por exemplo, irá definir o que chamamos de ritmo. Pegamos, por exemplo, uma notação de compasso com 4 tempos, cada tempo dividido em 4. A Figura 2.1 traz alguns exemplos rítmicos possíveis em compassos 4/4.

```

1 F = Forte
2 P = Fraco (Piano)
3
4 Tempo | 1---|2---|3---|4---|
5 -----+-----+-----+-----
6 Rock | F---|p---|F-F-|p---|
7 Ijexa | p---|p---|F---|F---|
8 Baiao | F--F|F-p-|F--F|F-p-|
9 Funk | F---|p-F-|--F-|p---|
10 Hip Hop|F---|p--F|F-F-|p---|

```

Código 2.1: Exemplos de ritmos

Com esta notação, é possível, por exemplo, escrever ritmos musicais para instrumentos de percussão e definir o tempo musical de todos os instrumentos de um grupo de percussão, como uma bateria de rock ou uma escola de samba. O Código 2.2 mostra um exemplo de notação rítmica.

```

1 Tempo | 1---|2---|3---|4---|
2 -----+-----+-----+-----
3 Bumbo | F---|----|F-p-|----|

```

```
4 Caixa |----|F---|----|p---|
5 Címbal |p-p-|p-p-|p-p-|p-p-|
```

Código 2.2: Exemplo de definição rítmica para bateria de rock

Esta notação rítmica é bastante utilizada em sequenciadores e permitem uma liberdade grande para criar novos ritmos. A Figura 2.9 apresenta esta divisão de tempo no software LMMS.



Figura 2.9: Tela da ferramenta LMMS e sua representação de tempo

Exercícios 2.2

10. Modifique seu metrônomo e permita que o usuário defina ritmos diferentes para o mesmo usando uma notação textual.
11. Faça uma composição com o LMMS cujo ritmo seja de rock, baião e samba.

2.3 Altura musical

Além de representar tempo, a notação musical também traz a informação da **altura musical** de um determinado som. A altura musical diz respeito à frequência vibratória do som e esta frequência é medida em **Hertz**, ou vibrações por segundo. Uma forma de representar altura musical é dar as frequências o nome de notas musicais.

2.3.1 As notas musicais

As notas musicais são uma maneira simbólica de representar as alturas do som. No sistema ocidental chamado temperado, temos 7 notas musicais: dó, ré, mi, fá, sol, lá, si. Esta nomenclatura foi inventada pelo italiano Guido D’Arezzo (992-1050) e estes nomes correspondem à sílaba inicial do “Hino a São João Batista”, um hino religioso conhecido em seu tempo.

Ut queant laxis
 Resonare fibris
 Mira gestorum
 Famuli tuorum
 Solve polluti
 Labii reatum
 Sancte Ioannes

Na França ainda se chama a primeira nota de Ut enquanto em outros países, para simplificar o solfejo, o Ut foi substituído por Dó, de Dominus (Deus), séculos depois.

Em inglês, além destes nomes, é comum chamar as notas musicais pelas primeiras letras do alfabeto, como apresentado na Tabela 2.3.

Notação americana	Nome em português	Frequência aproximada
A	LÁ	110.000
B	SI	123.750
C	DÓ	132.000
D	RÉ	148.500
E	MI	165.000
F	FÁ	175.956
G	SOL	198.000
A	LÁ	220.000

Tabela 2.3: Representação das notas musicais com letras e possíveis frequências

Este padrão de distribuição das notas se repete. É possível notar, observando a Figura 2.10, como acontece esta repetição no piano. Depois de um si, teremos outro dó, e assim sucessivamente. Chamamos este padrão de oitava, por incluir as 8 primeiras notas musicais.

É possível imaginar que seria possível alcançar os limites da audição humana, tanto para cima (20KHz) quanto para baixo (20Hz) subindo e descendo as oitavas. Certamente, nem o piano nem nenhum outro instrumento musical chega próximo destes limites...

Muitos instrumentos musicais podem emitir notas em mais de uma oitava. Para distinguir qual a oitava de uma determinada nota, utiliza-se representar um número ao lado dela. Para servir de referência, o Dó central de um piano de 88 teclas está em sua quarta oitava e por isso é chamado C4.

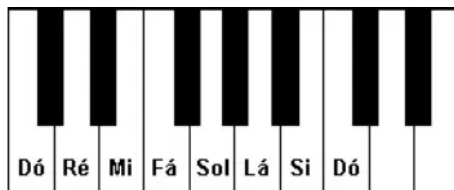


Figura 2.10: Disposição das notas no teclado do piano

Sustenidos e Bemóis

A Figura 2.10 também nos mostra que, além das notas representadas por teclas brancas, há outras notas representadas por teclas pretas. Estas notas recebem o nome de sustenidos e bemóis. Assim, entre o dó e o ré existe uma nota que pode ser chamada tanto de dó sustenido ($C\sharp$) quanto de ré bemol ($D\flat$). Desta maneira, as 7 notas musicais são 12: C, $C\sharp/Db$, D, $D\sharp/E\flat$, E, F, $F\sharp/G\flat$, G, $G\sharp/Ab$, A, $A\sharp/B\flat$, B, C e tudo se repete novamente.

= Sustenido

b = Bemol

Os acidentes, como são chamados, são representados com símbolos adicionais, antes da nota, na mesma linha. Existe também acidentes como o dobrado sustenido, que indica 2 semitons acima da nota original, e o dobrado bemol, que indicam 2 semitons abaixo. Também temos um símbolo chamado bequadro, que serve para cancelar um bemol ou sustenido e retornar a nota ao seu valor sem acidente. Estes acidentes são representados na Figura 2.11.

2.3.2 Escalas musicais

Apesar de termos 12 notas musicais, a música ocidental, em especial a música popular, é possível escolher um grupo de notas que "combinam" para criar melodias. A este grupo de notas selecionadas damos o nome de escala. Entre as diversas escalas existentes, a primeira que podemos definir é a escala cromática, que possui as 12 notas musicais.

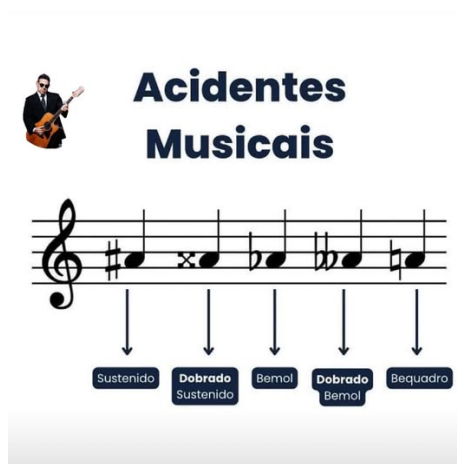


Figura 2.11: Representação dos acidentes

[illegible]

Escalas Maiores

Apesar de ser a base de todas as escalas que serão apresentadas a seguir, a escala cromática não é muito popular na música popular. Outra escala comumente utilizada na música popular de tradição europeia é a chamada escala maior, também chamada de escala diatônica, utilizada em músicas tonais. Esta escala possui apenas 7 notas que são definidas a partir de uma nota fundamental que dá o nome à escala. Assim, as notas da escala de Dó maior são as teclas brancas do piano, conforme apresentado abaixo.

+---	+---	+---	+---	+---	+---	+---	+---	+---	+---	+---	+---	+---	+---	+---
C	C#	D	D#	E	F	F#	G	G#	A	A#	B	C	...	
+---	+---	+---	+---	+---	+---	+---	+---	+---	+---	+---	+---	+---	+---	+---
T		2		3	4		5		6		7	8	...	
+---	+---	+---	+---	+---	+---	+---	+---	+---	+---	+---	+---	+---	+---	+---

Notas da escala de C maior: C D E F G A B C

Escala Menor Harmônica

A escala menor harmônica é similar a escala menor natural mas possui a sua sétima elevada em meio tom. Por conta desta sétima, a escala possui uma tensão harmônica maior em direção à tônica, sendo muito popular em canções.

♪ Estrutura: T-ST-T-T-ST-T+ST-ST

♪ Exemplo: C D Eb F G Ab B C

♪ Exemplo: A B C D E F G# A

Escala Menor Melódica

A escala menor harmônica cria um intervalo de um tom e meio entre a sexta e a sétima nota. A escala menor melódica diminui este intervalo elevando também a sexta em meio tom. Em sua forma clássica, esta modificação só acontece quando a escala é ascendente, retornando ao formato da escala menor harmônica quando tocada na descendente.

♪ Estrutura: T-ST-T-T-T-T-ST

♪ Exemplo: C D Eb F G A B C

♪ Exemplo: A B C D E F# G# A

Escala Pentatônica Maior

A escala pentatônica maior é uma escala de cinco notas (por isso seu nome) que é amplamente utilizada para improvisação. Esta escala baseia-se na escala maior com a remoção dos semitons, ou seja da quarta e da sétima nota. Outra forma de construir esta escala é pegar as 5 notas a partir da tônica no ciclo de quinta, exemplo, C G D A E.

♪ Estrutura: T-T-ST+T-T-T+ST

♪ Exemplo: C D E G A C

Escala Pentatônica Menor

A escala pentatônica menor é também uma escala de cinco notas que baseia-se na escala menor com a remoção dos semitons, ou seja da segunda e da sexta nota.

♪ Estrutura: T+ST-T-T-ST+T-T

♪ Exemplo: C Eb F G Bb C

♪ Exemplo: A C D E G A

Escala Diminuta Dominante (Dim-Dom)

A escala diminuta é uma escala simétrica de 7 notas baseada na alternância entre tom e semitom. Devido à sua estrutura, ela repete-se a cada um tom e meio, permitindo uso sobre acordes diminutos e dominantes para criar tensão.

♪ Estrutura: T-ST-T-ST-T-ST-T-ST

♪ Exemplo: C D Eb F Gb A B C

Devido a sua simetria, a escala de C, Eb, Gb e A terão as mesmas notas. Com isso, podemos dizer que há apenas 3 desenhos desta escala.

Escala Dominante Diminuta (Dom-Dim)

A escala dominante diminuta também é uma escala simétrica de 7 notas baseada na alternância entre tom e semitom, mas cuja estrutura inicia-se com um semitom. Devido à sua estrutura, ela repete-se a cada um tom e meio, permitindo uso sobre acordes diminutos e dominantes para criar tensão.

♪ Estrutura: ST-T-ST-T-ST-T-ST-T

♪ Exemplo: C Db Eb E F# G A Bb C

Devido a sua simetria, a escala de C, Eb, Gb e A terão as mesmas notas. Com isso, podemos dizer que há apenas 3 desenhos desta escala.

Escala de tons inteiros

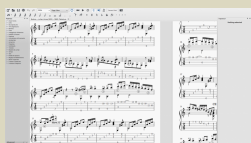
A escala de tons inteiros é uma escala simétrica hexatônica (de seis notas) composta por intervalos de um tom inteiro, sem semitons. Esta estrutura simétrica cria uma sonoridade sem centro tonal

♪ Estrutura: T-T-T-T-T-T

♪ Exemplo: C D E F# G# A# C

Esta escala não possui centro tonal pois ela é idêntica para as seis notas da escala. Pelo mesmo motivo, só existem 2 escalas, de tons inteiros, a apresentada no exemplo, e a que possui as notas que não estão na escala de exemplo.

Dica de Software: Musescore



O Musescore é uma ferramenta open-source para escrita de partitura. A ferramenta trabalha com diversos formatos intercambiáveis de música simbólica, como MIDI, musicXML e outros. Além disso, exporta para PDF e possui muitos plugins que

permitem que a mesma seja customizada e estendida para novas funcionalidades.

↗ Mais informações: <https://musescore.org/pt-br>

Exercícios 2.3

12. As notas mais altas e mais baixas de um determinado instrumento musical é chamado de tessitura. Qual a tessitura de um piano?

13. Quais instrumentos possuem a tessitura mais grave? Quais

instrumentos possuem a tessitura mais aguda?

14. Encontre a partitura de uma canção popular, como cai-cai balão ou Parabéns para você e tente entender a notação da mesma.

2.3.3 As Notas na partitura

Outra notação comum para a música é a partitura. Na partitura, a representação das notas é feita espacialmente usando cinco linhas e os espaços entre as linhas, em uma estrutura gráfica chamada de pentagrama. Esta representação segue uma sequência onde cada espaço e cada linha representa uma determinada nota. No entanto, para simplificar (ou complicar talvez) a representação de diferentes oitavas, são utilizadas diferentes Claves.

A **clave** é aquele desenho do começo do pentagrama, apresentada na Figura 2.12. Nesta Figura apresentamos as duas claves mais comuns: Clave de

Sol e Clave de Fá. A clave de Sol compreende as alturas entre o C4 e o G5, enquanto a clave de Fá compreende notas entre o E2 e o B3. Existe ainda a Clave de Dó e a possibilidade de representar claves em outras linhas, mas não vamos entrar neste assunto.

A posição de cada nota em uma partitura será sempre relativa a clave utilizada em seu início e por isso podemos dizer que esta representação é relativa pois as posições ocupadas pelas notas não são fixas, mas relativas à clave da partitura, conforme apresentado na Figura 2.12,

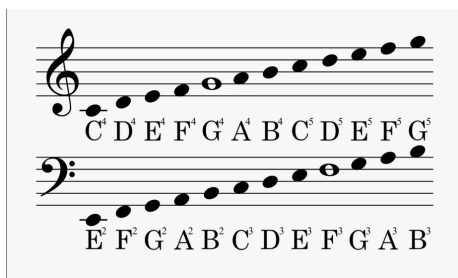


Figura 2.12: Disposição das oitavas na partitura

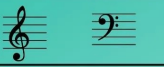
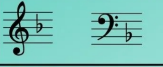
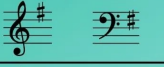
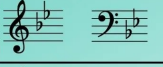
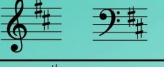
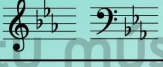
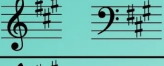
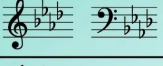
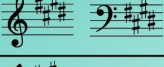
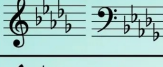
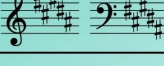
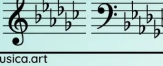
Certamente existem notas acima do G5 e também há notas abaixo deste E2 e alguns artifícios são utilizados para representar estas notas na partitura, como linhas adicionais, transposição de claves ou outras representações possíveis que não iremos tratar aqui neste livro.

Armadura de Clave

Quando escolhemos uma escala maior para escrever uma música dizemos que definimos seu tom. Uma música pode ter diversos acidentes fora de sua escala mas é comum que muitas músicas utilizem apenas as notas de uma escala. Como as escalas possuem seus intervalos fixos, podemos notar que seus acidentes serão sempre os mesmos e a escrita de uma música em um determinado tom pode demandar um certo esforço por exigir sustenidos e bemóis ao longo de toda a pauta.

Para facilitar a escrita, é utilizado um artifício chamado **armadura de clave** onde anotamos todos os sustenidos e bemóis no início da pauta e estes acidentes passam a valer para todas as notas marcadas.

A Figura 2.13 apresenta algumas armaduras de clave comuns.

Armadura	Tonalidad Mayor	Tonalidad Menor	Armadura	Tonalidad Mayor	Tonalidad Menor
	Do (C)	Lam (Am)		Fa (F)	Rem (Dm)
	Sol (G)	Mim (Em)		Sib (Bb)	Solm (Gm)
	Re (D)	Sim (Bm)		Mib (Eb)	Dom (Cm)
	La (A)	Fa#m (F#m)		Lab (Ab)	Fam (Fm)
	Mi (E)	Do#m (C#m)		Reb (Db)	Sibm (Bbm)
	Si (B)	Sol#m (G#m)		Solb (Gb)	Mibm (Ebm)

www.creatumusica.art

Figura 2.13: Armaduras de Clave

Porque tanto acidente?

Ao usar esta régua para escrever a escala de Fá maior, temos o seguinte grupo de notas: F, G, A, A#, C, D, E. É possível notar que a nota si não aparece e que temos duas notas lá: A e A#. Se formos utilizar a escala desta forma, teremos que escrever na partitura usando sustenidos e bequadros na linha da nota A enquanto a linha da nota B não será utilizada. Solução? Notamos o A# como Bb e tudo ficará mais simples para a notação da partitura.

Com isso, podemos notar também que todas as notas irão aparecer na escala maior. Todas as escalas maiores tem as 7 notas musicais: dó, ré, mi, fá, sol, lá, si, algumas delas com acidentes, como o sustenido. E é esta a razão de termos dois nomes para a mesma nota, A# ou Bb. E também é a razão para termos dobrado sustenido e dobrado bemol. Para entender, fica como exercício escrever as notas da escala de G#.

Ciclos de Quartas e Quintas

Além disso, é interessante notar que há apenas uma escala que com apenas um sustenido, a escala de Sol maior. Também teremos apenas uma escala com apenas um bemol, a escala de Fá maior.

Isso se dá devido a disposição das notas na escala ser bastante similar na

primeira metade e na segunda metade. A escala maior é definida por

Tom Tom Semitom Tom Tom Tom Semitom

Tom Tom Semitom Tom Tom Tom Semitom

Tom Tom Semitom Tom ...

Assim, uma parte grande da escala de Dó será idêntico a escala de Fá e também da escala de Sol.

Escala de Fá F G A Bb C D E F G A ...

Escala de Dó C D E F G A B C D E

Escala de Sol G A B C D E F# G A B

Escala de Ré D E F# G A B C#

Este entrelaçamento entre as escalas maiores acontece em um ciclo que chamamos de ciclo de quartas e quintas, apresentado na Figura 2.14.

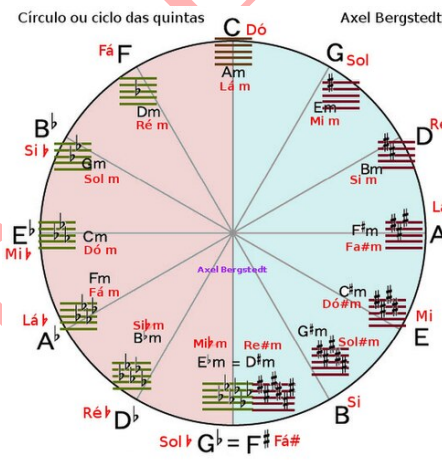


Figura 2.14: Ciclo de quartas e quintas

Exercícios 2.3

15. Faça um software que, dado a fundamental, dê como saída as notas desta escala. O software deve ser capaz de devolver es-
- calas maiores com sustenidos, bemóis e dobrados sustenidos e bemóis.

2.3.4 Intervalos musicais

Apesar de termos uma paixão pelas notas musicais, elas não são tão importantes assim para nossos ouvidos. De certa forma, o que a maioria das pessoas escuta não são notas musicais mas sim intervalos musicais ou a distância entre duas notas. Se tocamos uma melodia em uma região do braço do violão e depois repetimos os mesmos gestos em outra região do braço, iremos reconhecer como a mesma melodia, mesmo que as notas tocadas em cada execução sejam distintas. Por esta razão, arrisco aqui dizer que escutamos mais os intervalos entre as notas do que as notas em si.

Assim, mais do que uma sequência de notas, há uma sequência de intervalos relativos que nós identificamos como a escala maior, “Parabéns para você” ou aquele último sucesso da Pabllo Vittar. A estes intervalos damos nomes, dependendo da distância do intervalo relativo, conforme apresentado no Tabela 2.4.

Os intervalos ainda podem ser maiores, menores, justo, simples, justos, compostos e diminutos, dependendo da distância relativa. Intervalos simples acontecem na mesma oitava enquanto intervalos compostos possuem mais de uma oitava, ou 12 semitons, de distância. Intervalos podem ainda ser ascendentes ou descendentes mas iremos focar aqui apenas nos ascendentes para simplificar as próximas explicações.

0 semitom	uníssonos	consonante
1 semitom	Segunda menor	dissonante
2 semitons (1 tom)	Segunda maior	
3 semitons	Terça menor	consonante
4 semitons	Terça maior	consonante
5 semitons	Quarta justa	consonante
6 semitons	Quarta aumentada / Quinta diminuta	dissonante
7 semitons	Quinta justa	consonante
8 semitons	Sexta menor	consonante
9 semitons	Sexta maior	consonante
10 semitons	Sétima menor	dissonante
11 semitons	Sétima maior	dissonante
12 semitons	Oitava justa	consonante

Tabela 2.4: Nome dos intervalos musicais

Os intervalos podem ainda ser consonantes, quando geram tranquilidade e resolução, ou dissonantes, quando geram tensão e instabilidade. São considerados consonantes os intervalos uníssonos e de oitava, quartas e quintas justas, terças e sextas, maiores e menores. Já os intervalos de segundas, sétimas e

quarta aumentada, são considerados dissonantes.

Exercícios 2.3

16. Mudar a velocidade de rotação de um disco de vinil altera o tom da música, e pode ser entendido como uma forma de mudar a escala sem modificar os intervalos. Com o apoio de um software, modifique a ve-	locidade de alguns áudios para alterar o tom de músicas.
	17. Crie um software transpositor que, a partir de uma entrada, modifique o tom de músicas mantendo os intervalos.

2.3.5 Acordes

Quando mais de uma nota musical soa simultaneamente surgem o que chamamos de acordes. Na música ocidental, costumamos chamar de acordes a execução simultânea de três ou mais notas musicais de um conjunto harmônico. A tônica do acorde, ou fundamental, é a nota que costuma ser a mais grave e que dá o nome ao acorde. Além da fundamental, os acordes possuem a terça e a quinta. A terça é a nota modal que determina o caráter do acorde (maior ou menor). Acordes menores possuem uma terça menor, com 3 semitons da nota fundamental, enquanto acordes maiores possuem uma terça maior, com 4 semitons da fundamental. Os acordes possuem ainda a quinta justa, com 7 semitons da fundamental.

```

+---+---+---+---+---+---+---+---+---+---+---+---+
| C | C#| D | D#| E | F | F#| G | G#| A| A#| B | C | ...
+---+---+---+---+---+---+---+---+---+---+---+---+
| T |   |   |   | 3 |   |   | 5 |
+---+---+---+---+---+---+---+---+---+---+
| C |   |   |   | E |   |   | G |
+---+---+---+---+---+---+---+---+---+---+
| T |   |   |   | 3 |   |   | 5 |
+---+---+---+---+---+---+---+---+---+---+
| D |   |   |   | F#|   |   | A |
+---+---+---+---+---+---+---+---+---+---+

```

Acorde de C maior: C E G

Acorde de D maior: D F# A

Colocar os acordes na partitura

Com isso, temos conjuntos definidos de notas como tríades maiores ou tríades menores, dependendo da nota modal de cada tríade. Também é possível que um acorde tenha terça menor e uma quinta diminuta. Neste caso, chamamos este acorde de acorde diminuto ou tríade diminuta. Há ainda o caso de acordes com a terça maior e a quinta aumentada, chamado de acorde aumentado ou tríade aumentada.

```

+---+---+---+---+---+---+---+---+---+---+---+---+
| D | D#| E | F | F#| G | G#| A| A#| B | C | ...
+---+---+---+---+---+---+---+---+---+---+---+---+
| T |   |   | 3m|   |   |   | 5 |
+---+---+---+---+---+---+---+---+---+---+
| D |   |   | F |   |   |   | A |
+---+---+---+---+---+---+---+---+---+---+
| T |   |   | 3m|   |   |   | 5 |
+---+---+---+---+---+---+---+---+---+---+
| E |   |   | G |   |   |   | B |
+---+---+---+---+---+---+---+---+---+---+

```

Acorde de D menor: D F A

Acorde de E menor: E G B

Um acorde pode ainda ter uma quarta nota, normalmente a sétima. Se a sétima for menor, chamamos este acorde com sétima menor, enquanto se a sétima for maior, com 11 semitons da fundamental, chamamos este acorde de acorde com sétima maior. Acordes com três notas são chamados de tríades, enquanto acordes com 4 notas são chamados de tétrades.

Exercícios 2.3

18. Faça um sistema que, dado uma nota fundamental, informe suas tríades maiores e menores.	des em tempo real.
19. Faça um sistema MIDI que aceite como entrada o andamento, a fórmula de compasso e uma sequência de acordes e que gere arpeggios destes acor-	20. As tríades e tétrades são especialmente úteis para acompanhamentos musicais. Dados isso, faça um sistema que aceite como entrada um acorde e que mostra as tríades e tétrades do mesmo no braço do violão.

2.3.6 Campo Harmônico

Da mesma maneira que temos uma escala, que escolhe determinadas notas para criar melodias, há um conjunto de acordes que costumam soar bem juntos. Estes acordes, quando associados a um tom, são chamados de Campo Harmônico. O campo harmônico é uma estrutura tonal utilizada para definir os acordes de uma determinada escala. Assim, ao definir uma escala, definimos quais as notas que serão usadas e, conseqüentemente quais são os acordes que utilizam estas notas.

Se partimos da escala de dó maior, por exemplo, que não possui acidentes, temos que cada nota da escala terá sua tríade. A tríade de dó (C - E - G), será maior. Já a tríade de ré, (D - F - A), será menor pois a terça maior, no caso, o fá sustenido, não está na escala de dó maior. Assim, iremos ajustar as notas de todos os acordes e com isso podemos chegar na coleção de acordes que definem o campo harmônico.

Se atentando a estas regras, o campo harmônico de Dó maior é composto pelos seguintes acordes:

C	Dm	Em	F	G	Am	Bm(5b)
I	IIm	IIIm	IV	V	VIIm	VIIIm(5b)

Entendendo que esta formação é padrão e que pode se aplicar às demais escalas maiores, teremos uma forma padrão de montar campo harmônicos. Com isso, é possível verificar que estes acordes soam bem juntos e que a escala de dó maior irá servir para criar melodias sobre estes acordes.

Exercícios 2.3

21. Dado uma sequência de acordes em uma canção, é possível afirmar a) o campo harmônico da canção, b) o tom desta canção, e c) uma escala que sirva para improvisar melodias para

esta canção?

22. Faça um software que aceite como entrada um tom e uma escala e que retorne como saída o campo harmônico deste tom.

Cifra?

Exercícios 2.3

23. Crie um software improvisador que, a partir de um tom e uma escala, crie melodias. Para

deixar o software mais interessante utilize variações de tempos, pausas e outros recursos.

2.4 Timbre

Sons diferentes

Tessituras

Famílias de instrumentos

♦ Respostas dos exercícios do capítulo 2.

1. Existe um simulador assim...
2. Há hoje pesquisas que mostram como capturar o som por câmeras de alta resolução.

3. Os eventos estão disponíveis na internet.

Pre-Print
V.0.0.1

Capítulo 3

Música simbólica

A representação da música no computador pode ser feita a partir das informações musicais, como as apresentadas no capítulo anterior. Altura, intensidade, duração, timbre, letras, tudo isso pode ser usado para representação de música. Esta representação é o que chamamos aqui de música simbólica um sistema de representação musical que não é baseado no áudio ou no som da música, mas em informações sobre a música em si. Um sistema de música simbólica se baseia na notação de elementos musicais, que podem variar dependendo de seu propósito e funcionalidade desejada e a partitura é um exemplo de sistema de música simbólica.

A partitura serve de exemplo mas não é o único sistema de notação musical. Há diversos formatos de notação musical, que podem ser chamados de partituras, variando de acordo com os instrumentos, estilos musicais, períodos e outras possibilidades de variação em uma notação para música. Compositores contemporâneos, como Iannis Xenakis, Hans-Joachim Koellreutter, e György Sándor Ligeti criaram diversas outras representações gráficas da música em representações visuais que também são chamadas de partitura.

Além da partitura ou das partituras, há diversos outros formatos de representação musical na música, como Cifras e tablaturas, que trazem informações específicas para um determinado instrumento e que segue a mesma lógica: notar elementos musicais que permitem a reprodução de uma peça, seja por um instrumentista, seja por um software.

Neste contexto, há ferramentas computacionais que podem auxiliar no processo de escrita, reprodução, análise e edição de música simbólica. É importante

pensar nesta ações de forma separada pois é possível que uma ferramenta sirva apenas para a escrita de música simbólica, mas que a mesma ferramenta não seja capaz de reproduzir a música escrita. Ou uma ferramenta capaz de reproduzir música simbólica mas que não é capaz de editar, modificar, ou representar graficamente esta música. Por isso, podemos pensar em um ecossistema de música simbólica com ferramentas e convenções distintas e interoperáveis entre si. Este ecossistema computacional pode envolver alguns elementos como diferentes formatos de arquivos para notação musical, ferramenta de escrita e edição, sequenciadores para tocar música simbólica em tempo real, sintetizadores e instrumentos virtuais para geração de som, protocolo de comunicação para música simbólica em tempo real, entre outras. Há ferramentas que possuem todas estas características enquanto outras possuem funções específicas neste ecossistema musical. Neste capítulo, discutiremos alguns destes elementos.

Seria legal aqui uma figura com este ecossistema.

Exercícios 3.0

1. Escute a peça Pithoprakta de Iannis Xenakis e veja sua partitura. É possível imaginar um sistema computacional para executar esta peça?	tizador criado para a peça de Xenakis seria capaz de tocar a peça de Koellreuter?
2. Compare a peça de Xenakis com a peça Acronon, de HJ Koellreutter. Será que um sinte-	3. Veja a peça Artikulation, de Ligeti e descreva sucintamente o que a partitura desta peça traz.

3.1 Escrevendo música simbólica

A escrita e edição de música simbólica em ferramentas computacionais é um tópico interessante de pesquisa. Se considerarmos que as entradas de dados padrões no computador são o teclado e mouse, A escrita musical no computador demanda de formatos de entrada em formatos textuais, como Lilypond e ABC, não querem ferramentas específicas e pode acontecer usando editores de textos. No entanto, escrever arquivos MIDI ou MusicXML pode demandar ferramentas adicionais tanto para a edição quanto para a escrita musical. Tais ferramentas computacionais permitiram pensar em diferentes abordagens de escrita musi-

cal, como iremos apresentar nesta seção. Novamente, não é intenção aqui ser exaustivo na apresentação de formas de escrita e edição musical simbólica no computador, mas trazer algumas possibilidades que permitam ter uma visão geral sobre este tema.

3.1.1 Notação Textual

Uma primeira possibilidade simples de notar música é utilizar a entrada textual. Assim, podemos transformar partituras em textos como uma lista de eventos, com altura, intensidade e duração.

3.1.2 Partitura

Os editores de partitura são bastante comuns e há diversas ferramentas disponíveis entre software livres e proprietários que editam partitura. Em geral, tais ferramentas podem possuir formatos de arquivos proprietários e também importar e exportar formatos como MusicXML e MIDI. Os editores de partitura podem ser só um editor de partitura ou trazer um conjunto de ferramentas que auxiliam práticas musicais, como ferramentas automatizadas para escrita com instrumento ou solfejo, sintetizadores, arranjadores, correção de contra-ponto, entre outras.

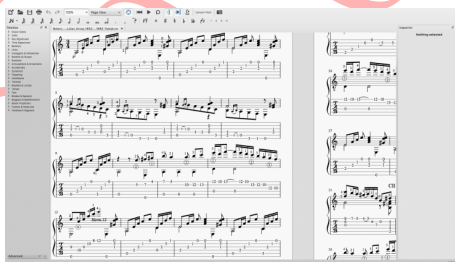


Figura 3.1: Tela do editor de partitura MuseScore

Por se tratar da maneira mais convencional de escrever música, estes editores não trazem muitas novidades na representação, ficando os destaques para as funcionalidades adicionais que uma ou outra ferramenta pode trazer sendo que estas funcionalidades muitas vezes podem ser ampliadas por meio de plugins e códigos extras desenvolvidos pela própria comunidade.

De maneira similar, há ferramentas para a escrita de outras formas, como editores de tablaturas para violão, que também permitem exportar para dife-

rentes formatos e que trazem ferramentas que permitem tratar a idiomática específica do instrumento representado.

Assim, criar representações e ferramentas para estas representações para outros instrumentos, como flauta, saxofone, gaita de boca ou berimbau, pode ser um desafio para a computação musical e também para a área de IHC.

3.1.3 Piano Roll

As caixinhas de música são possivelmente os primeiros dispositivos capazes de permitir que leigos pudessem tocar música. Estas caixinhas possuíam um pente afinado com notas musicais e um disco com pinos que, quando girados, acionavam as notas no pente e permitiam a execução de pequenas peças. Esta invenção, criada em 1796 pelo relojoeiro suíço Antoine Favre, está certamente no cerne do protocolo MIDI e da música no computador.

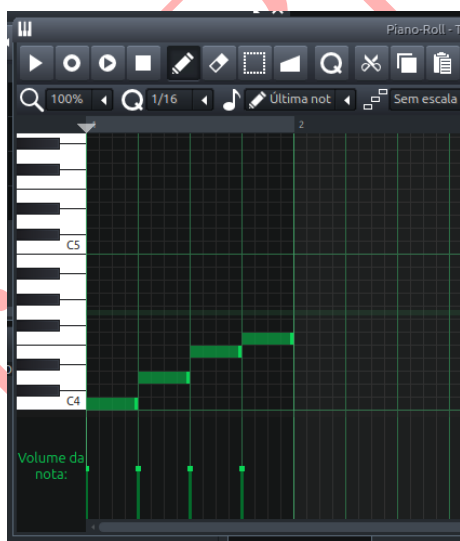


Figura 3.2: Exemplo de piano roll do LMMS

Quase um século depois das caixinhas de música, surgem as pianolas automáticas, dispositivos que tocavam música de forma autônoma. As pianolas automáticas utilizavam rolos com papéis perfurados para a escrita musical e, com isso, permitiam que diversas peças fossem executadas no mesmo dispositivo. Novamente, temos aqui um ascendente direto do protocolo MIDI e talvez até do cartão perfurado, utilizado posteriormente pela computação para a pro-

gramação de computadores.

A metáfora do cartão perfurado da pianola automática trouxe uma possibilidade de edição de música simbólica no computador que dispensa a necessidade de entender partitura. Este formato, conhecido como piano roll, representa o rolo de cartão perfurado enrolado na horizontal, o piano geralmente à esquerda, com as notas graves embaixo. O rolo se desloca para a esquerda e os furos do cartão denotam tanto a altura musical, em seu eixo y, quanto o tempo em seu eixo x.

Boa parte das ferramentas que utilizam o piano roll possuem ainda uma maneira de representar intensidade (velocidade MIDI). No caso do exemplo da Figura 3.2, a intensidade está representada por um eixo adicional, abaixo do piano roll. Cada evento recebe uma linha neste eixo e o tamanho desta linha representa a intensidade do evento.

Este formato também é relacionado com a notação de eventos musicais utilizado por jogos como o Guitar Hero.

3.1.4 Sequenciador de passos

O avanço da eletrônica na música permitiu a popularização dos órgãos eletrônicos. Estes instrumentos eletrônicos contavam com diversos sons diferentes e traziam a possibilidade de sequenciar partes da música ou mudar o timbre do instrumento durante a sua execução. Desde a década de 1960, se tornou comum que os órgãos eletrônicos trouxessem máquinas de acompanhamento rítmicos. Estas baterias eletrônicas traziam diversos padrões rítmicos pré programados que podiam ser selecionados mas que não podiam ser modificados pelos músicos.

Em 1980, a japonesa Roland decide criar uma máquina de ritmos que pudesse ser programada e surge a TR-808, a primeira máquina de ritmos que permitia ao usuário escrever seus ritmos e não apenas escolher ritmos a partir de uma lista de padrões.

A programação da TR-808 trazia botões selecionáveis (toggle) que permitiam escolher em qual tempo de um loop um determinado instrumento de percussão iria soar. A Figura 3.3 apresenta o painel frontal deste dispositivo.

Além de ter sido um sucesso de vendas e ter proporcionado o nascimento de diversos ritmos de música eletrônica, a TR-808 trouxe uma outra metáfora para a escrita musical, em especial, para faixas de ritmos. Ferramentas como FL Studio, LMMS, Hydrogen e outras, partem desta mesma interface para a escrita



Figura 3.3: Painel frontal da TR-808

de ritmos. A Figura 3.4 mostra este modelo de notação em uma ferramenta sequenciadora.



Figura 3.4: Marcação rítmica no LMMS

Vale lembrar que a Roland é uma das empresas envolvidas na criação do protocolo MIDI mas que a TR-808 foi criada antes deste protocolo.

Exercícios 3.1

4. Proponha uma outra forma de representação musical no computador, como lista textuais de eventos, grafos orientados ou círculos.
5. Crie um modelo textual de representação rítmica, como a tablatura de bateria, e faça um sequenciador capaz de tocar este seu formato de arquivo.

3.2 Formatos de arquivos para notação musical

Para representar estruturas musicais simbólicas, é necessário utilizar algum formato de dado simbólico, como formatos binários, arquivos XML, arquivos de texto ou outro formato para armazenar as características de um determinado som ou representar a ação do músico ao tocar um instrumento e gerar este mesmo som. Nesta seção traremos alguns exemplos de formatos de arquivos usados para armazenar música simbólica.

Algumas escritas de música no computador possuem por objetivo tocar a música escrita, como é o caso do formato MIDI, que será visto a seguir. Outras formas tem por objetivo apresentar visualmente a partitura da música escrita, como o formato ABC, que também será visto a seguir. Alguns destes formatos são intercambiáveis e pode ser possível transformar um formato de representação em outro.

Vale ressaltar que esta seção não traz um levantamento exaustivo de formatos de notação musical mas alguns exemplos que servem para ilustrar e exemplificar formatos existentes. A pesquisa sobre outros formatos, não listados nesta seção, fica como exercício para quem se interessa pelo assunto.

3.2.1 ABC

O ABC é um formato de notação musical comumente utilizado para representar músicas¹. Este formato utiliza caracteres no formato ASCII para escrever partituras e se propõe a permitir uma escrita de partituras rápida e descomplicada. Os arquivos no formato ABC possuem duas partes: uma espécie de cabeçalho ou preâmbulo que trazem características gerais da peça, e as notas e tempos da música.

Os elementos da partitura descritos no capítulo anterior são descritos em um arquivo de texto seguindo um formato padrão. Um exemplo de notação musical em ABC é apresentado a seguir no Código 3.1.

```
1 X:1
2 T:Notes / pitches
3 M:C
4 L:1/4
5 K:C treble
6 C, D, E, F, | G, A, B, C | D E F G | A B c d |
7 e f g a | b c' d' e' | f' g' a' b' |]
```

Código 3.1: Exemplo de código para notação musical em ABC

Ao processar este arquivo, a saída é apresentada na Figura 3.5.

Além disso, o ABC se tornou um formato bastante popular de notação de música e há neste formato mais de 750.000 músicas disponíveis, tornando-o uma notação muito interessante para quem pesquisa mineração de dados e treinamento de modelos computacionais musicais em geral.

¹Disponível em <https://abcnotation.com/>.

3.2.2 Lilypond

A necessidade de editar arquivos musicais em trabalhos acadêmicos gerou diversos trabalhos que incorporam partituras textuais em arquivos do L^AT_EX. O LilyPond (<https://lilypond.org/>) é uma ferramenta baseada em TeX, disponível para várias plataformas e que permite a escrita de partituras musicais com base numa filosofia de linguagem de marcação. Sua intenção é gerar uma partitura com a melhor qualidade possível permitindo que compositores definam detalhes específicos de cada elemento na escrita musical. Similar ao ABC, o LilyPond é textual, no entanto, seu objetivo é dar muita autonomia ao compositor para expressar detalhes da partitura que podem não são facilmente descritos no formato ABC.

O código do Lilypond permite representar o tempo musical a cada nota, como no exemplo do Código 3.3, cuja saída é apresentada na Figura 3.6.

```
1 \version "2.22.2"
2 {
3   c'1 e'2 g'2 e'2 f4 g4 a4 b8 c8
4 }
```

Código 3.3: Exemplo de código do Lilypond



Figura 3.6: PDF gerado com a partitura pelo Lilypond.

Existem também diversas ferramentas e bibliotecas de programação para tratar arquivos Lilypond e é possível a partir de arquivos lilypond gerar partituras em diversos formatos e também exportar o conteúdo para outros formatos, como MusicXML ou MIDI.

```
1 \score {
2   \new Staff {
3     \tempo "Allegro" 4 = 120
4     \time 4/4
5     \key c \major
6     \clef treble
7
8     c'4 d' e' f' |
```

```
9      g' a' b' c' ' |
10    }
11 }
```

Código 3.4: Exemplo de código do Lilypond

Exercícios 3.2

- | | |
|---|--|
| <p>12. Como funciona o arquivo no formato LilyPond?</p> <p>13. O que significa cada definição no seu arquivo?</p> <p>14. Como o Lilypond define fórmula de compasso e os tempos das notas musicais?</p> <p>15. Encontre um software Linux para:</p> | <p>♪ Converter músicas em Lilypond para PDF.</p> <p>♪ Tocar músicas no formato Lilypond.</p> <p>16. Encontre uma biblioteca Python que permita:</p> <p>♪ Ler arquivos no formato Lilypond.</p> <p>♪ Criar e modificar músicas no formato Lilypond.</p> |
|---|--|

3.2.3 MusicXML

O MusicXML <https://www.musicxml.com/> é um formato de arquivo de notação musical aberto, baseado em XML que surgiu com o intuito de ser um formato universal de notação musical para música de tradição ocidental. Muitas ferramentas de edição de partitura possuem formatos próprios de arquivos mas, em geral, há uma tendência para que estas ferramentas importem e exportem arquivos no formato MusicXML, tornando-o um formato universal e intercambiável para notação de partituras musicais. Assim, o MusicXML possui uma ampla gama de usuários e ferramentas que permitem executar, editar, criar e exportar partituras musicais neste formato.

Apesar de ser textual, como o Lilypond e o ABC, o MusicXML não tem por objetivo a edição simples e rápida de partituras, já que a utilização de tags XML torna o processo de escrita um pouco mais lento e verboso. Assim, podemos afirmar que, apesar de ser textual, este formato não é exatamente para leitura humana, apesar de ser possível. Por outro lado, a estrutura XML traz em si um modelo de documento hierárquico (DOM - Document Object Model) para a

partitura que permite modelar representações de partituras, como apresentados no Código3.5.

```

1
2 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
3 <!DOCTYPE score-partwise PUBLIC
4     "-//Recordare//DTD MusicXML 4.0 Partwise//EN"
5     "http://www.musicxml.org/dtds/partwise.dtd">
6 <score-partwise version="4.0">
7     <part-list>
8         <score-part id="P1">
9             <part-name>Music</part-name>
10        </score-part>
11    </part-list>
12    <part id="P1">
13        <measure number="1">
14            <attributes>
15                <divisions>1</divisions>
16                <key>
17                    <fifths>0</fifths>
18                </key>
19                <time>
20                    <beats>4</beats>
21                    <beat-type>4</beat-type>
22                </time>
23                <clef>
24                    <sign>G</sign>
25                    <line>2</line>
26                </clef>
27            </attributes>
28            <note>
29                <pitch>
30                    <step>C</step>
31                    <octave>4</octave>
32                </pitch>
33                <duration>4</duration>
34                <type>whole</type>
35            </note>
36        </measure>
37    </part>
38 </score-partwise>

```

Código 3.5: Exemplo de código do MusicXML

Este formato foi totalmente documentado e é adotado, recomendado e acordado pela W3C (World Wide Web Consortium), associação que regulamenta a

Web. Por isso, estes arquivos são definidos por série de DTDs e possuem um esquema de validação (XSD).

Por ser um formato XML, é possível abrir, modificar e criar arquivos no formato MusicXML utilizando uma biblioteca para tratar arquivos XML, sem utilizar uma biblioteca especial. No entanto, há diversas bibliotecas e ferramentas para a manipulação de arquivos neste formato.

Exercícios 3.2	
17. Como funciona o arquivo no formato MusicXML?	Python que permita: ♪ Ler arquivos no formato MusicXML. ♪ Criar e modificar músicas no formato MusicXML.
18. O que significa cada definição no seu arquivo?	
19. Encontre uma biblioteca	

3.3 MIDI

O MIDI (Musical Instrument Digital Interface - Interface Digital de Instrumentos Musicais) (<https://midi.org/>) é um padrão de comunicação criado por um consórcio de fabricantes de equipamentos musicais em 1982. O padrão MIDI inclui diversos padrões: uma conexão física (a chamada porta MIDI), um protocolo de comunicação em tempo real para conectar diferentes dispositivos musicais (o protocolo MIDI), um formato de arquivo (os arquivos midi) e mais. Apesar de ser bastante antigo, este padrão é ainda é bastante presente para a conexão entre diferentes dispositivos e aplicações musicais.

Quando lançado, o protocolo MIDI trouxe a especificação de uma porta de comunicação serial para a comunicação entre dispositivos MIDI. Esta porta, apresentada na Figura 3.7, utiliza o padrão DIN (geralmente de 5 pinos), sendo comum possuir 3 conectores: MIDI IN, para receber dados, MIDI OUT, para enviar dados, e MIDI THRU, para repassar dados recebidos na entrada para outro dispositivo.

Atualmente, estas portas ainda são utilizadas, em especial para comunicação com baixa latência, e é comum encontrar dispositivos MIDI que utilizam comunicação USB ou Bluetooth para sua interconexão. Com esta atualização, uma grande diversidade de dispositivos MIDI ainda se encontra no mercado e este protocolo, apesar de sua idade, continua bastante atual e amplamente



Figura 3.7: Portas MIDI em uma interface de som

utilizado, já que o mesmo segue como um protocolo padrão da indústria de dispositivos musicais. Para os interessados e as interessadas em criação de interfaces musicais físicas que usam este protocolo, há diversas bibliotecas para a comunicação MIDI em microcontroladores, como o Arduino. O sistema MIDI é baseado em alguns conceitos, que serão apresentados a seguir.

3.3.1 Eventos

Os eventos MIDI são mensagens utilizadas tanto para a comunicação entre dispositivos quanto para armazenar os dados MIDI em arquivos. Os dois principais eventos para notas são o que marca o início de uma nota musical (Note On - 0x9n) e o que marca o fim de uma nota (Note Off - 0x9n). Estes eventos são normalmente disparados quando apertamos uma tecla e quando soltamos a tecla em um controlador MIDI, como um teclado.

O formato MIDI utiliza um padrão de 128 valores para representar notas musicais, de 0 a 127 sendo que valor zero é a primeira nota Dó, o 1 é o Dó sustenido e assim sucessivamente, sendo o dó central (C4) a nota número 60 (0x3C), conforme apresentado pela Figura 3.8. Já a intensidade de uma nota musical é chamada de velocidade e também pode assumir valores de 0 a 127. Há sistemas MIDI em vez de utilizar o Note Off para marcar o fim da execução de uma nota, utiliza o Note On da nota com velocidade zero, sem som.

Alguns teclados MIDI permitem modificar o som de uma nota pressionando a tecla após ela ter sido disparada, o chamado After Touch (0xA_n). No caso deste evento ser polifônico, com mais de uma nota, ele é chamado de Polytouch ou Channel Pressure (0xD_n). Também há um evento para marcar modificação da afinação de uma nota após ela ser disparada, como o bend, feito em violões e

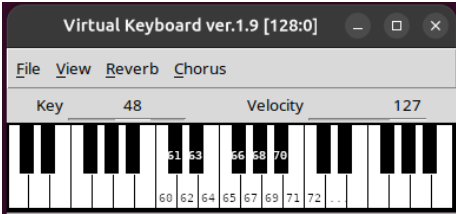


Figura 3.8: Notas MIDI anotadas no teclado virtual vkeybd

guitarras. Em teclados MIDI, isso costuma ser feito por uma roda, posicionada ao lado das teclas e é chamado de Pitch Wheel (0xEn). Estas mensagens são apresentadas na Tabela 3.1.

Mensagem	Código	Atributos
Note off	0x8n	channel, note, velocity
Note on	0x9n	channel, note, velocity
Aftertouch	0xA _n	channel, value
Control change	0xB _n	channel, control, value
Program change	0xC _n	channel, program
Polytouch	0xD _n	channel, note, value
Pitchwheel	0xE _n	channel, pitch (range: -8192 to 8191)

Tabela 3.1: Eventos MIDI Padrão

Além das mensagens que controlam notas, os arquivos MIDI podem também possuir eventos que controlam os sintetizadores MIDI. Para isso, utiliza-se a mensagem de Control (0xB_n), que modifica valores de controle. Esta mensagem é comumente utilizada por controladores MIDI em hardware ou software que possuem botões rotatórios ou deslizantes. Também há uma mensagem que modifica o programa / sintetizador de um canal MIDI, chamada Program Change (0xC_n). Esta mensagem modifica o sintetizador ou preset de um dispositivo em uma lista de valores válidos entre 0 e 127.

Canais MIDI

O protocolo MIDI foi criado para conectar diversos dispositivos em um barramento comum, seja por meio de conectores físicos ou lógicos. Para permitir endereçamentos diferentes em sua comunicação, o MIDI define uma multiplexação deste barramento em comum, por meio da definição de canais em suas mensagens. Ao todo temos 16 canais MIDI, variando entre 0 e 15. Desta ma-

neira, se temos mais de um controlador MIDI sendo endereçado para mais de um sintetizador MIDI, é possível separar as mensagens utilizando canais diferentes para cada controlador. É possível ainda que um dispositivo aceite todos os canais, como em uma comunicação broadcast.

Código	Mensagem	Canal
0x80	Note Off	0
0x91	Note On	1
0xA2	AfterTouch (ie, key pressure)	2
0xB3	Control Change	3
0xCA	Program (patch) change	10
0xDB	Channel Pressure	11
0xEC	Pitch Wheel	12

Tabela 3.2: Mensagens MIDI com canal

Assim, utilizando os canais MIDI, é possível que uma ferramenta de leitura de arquivo MIDI possa direcionar as notas de uma faixa, por exemplo, piano, para um determinado sintetizador enquanto outra faixa, por exemplo, bateria, é direcionada para outro sintetizador.

Com isso, as mensagens padrão passam a ter a informação da mensagem + a informação do canal, como apresentado na Tabela 3.2.

Sincronização de dispositivos

O sistema MIDI também traz mensagens para controlar sequenciadores e para a sincronização de dispositivos. Tais mensagens incluem mensagens como Start (0xFA), Continue (0XFB) e Stop (0xFC), para controlar o sequenciador, além de uma mensagem Song Select (0xF3), para selecionar a música a ser tocada. Também há uma mensagem de reset (0xFF), para reiniciar um dispositivo.

Para a sincronização entre dispositivos, o MIDI também pode utilizar uma mensagem de clock (0xF8) que funciona como um metrônomo, enviando 24 pulsos por nota mínima (24 PPQ - Pulses Per Quarter), para indicar o andamento de cada compasso. Há também uma mensagem para posicionar o sequenciador uma posição específica da música, chamada Song Position Pointer (0xF2). Caso de um dispositivo fique gerar eventos por muito tempo, a comunicação pode ser mantida por meio de uma mensagem de heart beat, chamada Active Sensing (0xFE), que deve ser enviada a cada 300ms.

Mensagem	Código	Atributos
SysEx (start)	0xF0	
Quarter Frame	0xF1	dados do código de tempo
songpos	0xF2	pos (0–16383)
song select	0xF3	song (0–127)
tune request	0xF6	
SysEx (end)	0xF7	
Clock	0xF8	
start	0xFA	
continue	0xFB	
stop	0xFC	
active sensing	0xFE	
reset	0xFF	

Tabela 3.3: Outras Mensagens MIDI

MIDI Time Code

A necessidade de sincronizar dispositivos musicais e de vídeo trouxe para a especificação MIDI um protocolo que permite a sincronização entre dispositivos baseado em um tempo absoluto (horas, minutos, segundos e quadros), ao invés de tempo relativo ao andamento musical. Este protocolo, chamado MIDI Time Code (MTC), permite pensar no tempo a partir da taxa de quadros por segundo, sendo que a taxa padrão em filmes é de 24fps (frames per second). Assim, o protocolo MTC envia informações de tempo em mensagens chamadas Quarter Frame (0xF1) a cada quarto de quadro, permitindo a sincronização de dispositivos quadro a quadro.

Existe ainda documentada uma mensagem chamada Tune Request (0xF6), que serve para pedir a um dispositivo para que ele se afine internamente. Esta mensagem não costuma ser utilizada em dispositivos modernos.

Note que algumas possibilidades de mensagens, como 0xF4, 0xF5, 0xF9 ou 0xFD não são especificadas pelo protocolo.

3.3.2 Metaeventos

Os eventos MIDI são utilizados em seu protocolo de comunicação para a troca de informação entre dispositivos e software. Além dos eventos MIDI padrão, o protocolo MIDI traz também uma coleção de metaeventos que não são enviados na comunicação entre dispositivos mas que são armazenados em arquivos MIDI como metadados de uma trilha. Isso permite várias Todo metaevento

MIDI inicia com o valor 0xFF, seguido do tamanho do metaevento e do valor propriamente dito. Os principais tipos de meta eventos são apresentados na Tabela :

Mensagem	Código	Atributos
Número da sequência	0x00	0x02 ssss
Texto descritivo	0x01	len text
Copyright	0x02	len text
Nome da faixa	0x03	len text
Nome do instrumento	0x04	len text
Letra de música	0x05	len text
Marcador	0x06	len text
Cues	0x07	len text
Definição de tempo	0x51	0x03 ttttt
Assinatura de tempo	0x58	0x04 nn dd cc bb
Assinatura de tonalidade	0x59	0x02 sf mi
Fim da faixa	0x2F	0x00
Específico de sequencers	0x7F	len data

Tabela 3.4: Outras Mensagens MIDI

Os meta eventos desconhecidos para uma determinada ferramenta podem ser ignorados, respeitando o comprimento definido para não corromper arquivos.

3.3.3 SysEx

Um controlador MIDI ou programa MIDI pode precisar de mensagens extras para um determinado controle que não é padrão no sistema MIDI. Um pedal diferente, um tipo de afinação, uma modulação ou qualquer comportamento não definido pelos eventos MIDI padrão podem ser incorporados no protocolo por meio de mensagens exclusivas de sistemas, as chamadas mensagens SysEx (System Exclusive). Estas mensagens iniciam com 0xF0, possuem tamanho não definido, e terminam com um status 0xF7.

As mensagens SysEx trazem informações para configurações de sistemas MIDI que podem ser exclusivas de alguns dispositivos e por isso não ser universais. Para facilitar o entendimento destas mensagens, normalmente utiliza-se o primeiro byte de dados (após o 0xF0) para definir o ID do fabricante. Assim, caso este byte não seja o esperado, o dispositivo pode ignorar todos os próximos bytes até o fim da mensagem.

Com estas mensagens, seria possível até mesmo codificar qualquer tipo de mensagem por sistemas MIDI e até mesmo codificar áudio, imagem ou vídeo e

transmitir via protocolo MIDI, entendendo, é claro, as limitações do protocolo quanto a velocidade e largura de banda.

3.3.4 Arquivos MIDI

O formato de arquivos MIDI, ou SMF (Standard MIDI Files) ou arquivos .mid, é um formato de arquivo usado para armazenar informações do protocolo MIDI. Por possuir mais de uma versão, podemos afirmar que existe mais de um formato de arquivo MIDI. Diferentemente dos formatos antes apresentados, o MIDI é um formato de arquivo binário e, por isso, depende de ferramentas para sua criação, edição, visualização ou reprodução. Por ser um formato bastante simples, aberto e amplamente aceito e utilizado, o MIDI foi e ainda é um formato de arquivo muito popular para troca de informações entre diferentes ferramentas de edição de partituras no computador, se tornando até sinônimo de música simbólica no computador.

Similar aos formatos anteriores, um arquivo MIDI também é dividido em partes: um cabeçalho e as trilhas. O cabeçalho de um arquivo MIDI traz informações sobre seu formato. As trilhas de um arquivo MIDI armazenam eventos, mensagens exclusivas do sistema (SysEx) ou metaeventos.

Diversas bibliotecas auxiliam o programador a criar e manipular arquivos MIDI. Um exemplo, no Código 3.6, utiliza a biblioteca MIDIUtil em Python para criar um arquivo midi.

```
1 # sudo apt install python3-MIDIUtil
2 from midiutil import MIDIFile
3
4 track = 0
5 time = 0 # In beats
6 channel = 0
7 seminima = 1 # In beats
8 volume = 100 # 0-127, as per the MIDI standard
9
10 mf = MIDIFile(1, file_format=1)
11 mf.addTrackName(track, time, "Escala de C Maior")
12 mf.addNote(track, channel, 60, 0, seminima, volume);
13 mf.addNote(track, channel, 62, 1, seminima, volume);
14 mf.addNote(track, channel, 64, 2, seminima, volume);
15 mf.addNote(track, channel, 65, 3, seminima, volume);
16 mf.addNote(track, channel, 67, 4, seminima, volume);
17 mf.addNote(track, channel, 69, 5, seminima, volume);
18 mf.addNote(track, channel, 71, 6, seminima, volume);
```

```
19 mf.addNote(track, channel, 72, 7, seminima, volume);
20
21 output_filename = "output.mid"
22 with open(output_filename, "wb") as output_file:
23     mf.writeFile(output_file)
```

Código 3.6: Exemplo de código para criação de arquivos Midi usando Python e a biblioteca MIDIUtil

A Figura ?? apresenta a partitura no MuseScore do arquivo MIDI gerado pelo Código 3.6.

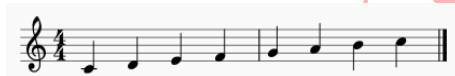


Figura 3.9: Apresentação da Partitura gerada pelo Código 3.6 na ferramenta Musescore

Além de criar arquivos MIDI, há bibliotecas que auxiliam também na leitura destes arquivos. O Código ?? mostra um código que abre um arquivo MIDI e lista as suas trilhas e a quantidade de mensagens (eventos) por trilha.

```
24 # sudo apt install python3-mido
25 import mido
26 from mido import MidiFile, MidiTrack, MetaMessage
27
28 midifile = mido.MidiFile("arquivo.mid")
29 for i, track in enumerate(midifile.tracks):
30     print(f"Track {i}: {track.name} ({len(track)} messages)")
```

Código 3.7: Exemplo de código para análise de arquivos Midi usando Python e a biblioteca Mido

3.3.5 O protocolo MIDI em Tempo Real

Além de criar arquivos MIDI, este protocolo permite também a comunicação de eventos em tempo real. Desta maneira, podemos programar um sequenciador para gerar eventos MIDI previamente agendados por uma notação musical, ou ter uma interface humana onde uma pessoa ou outro sistema poderá gerar estes eventos em tempo real usando as interfaces padrões do computador, como teclado e mouse, ou sensores e microcontroladores como o Arduino.

Um dispositivo MIDI, tanto de hardware quanto de software, possui um nome que permite identificá-lo. Além disso, um dispositivo poderá ter portas

de entrada, para receber dados, e portas de saída, para enviar dados. A criação de um dispositivo, uma porta e eventos de notas usando a API ALSA MIDI em Python é apresentada no Código ??.

```

31 # sudo pip install --user --break-system-packages package_name
    python-alsa-midi
32
33 import time
34 from alsa_midi import SequencerClient, READ_PORT, NoteOnEvent,
    NoteOffEvent
35
36 client = SequencerClient("my client")
37 port = client.create_port("output", caps=READ_PORT)
38
39 notes = [72, 74, 76, 77, 79, 81, 83, 84] # C D E F
40 while True:
41     for note in notes:
42         event = NoteOnEvent(note=note, velocity=100, channel=0)
43         client.event_output(event)
44         client.drain_output()
45         time.sleep(0.5)
46     for note in notes:
47         event = NoteOffEvent(note=note, channel=0)
48         client.event_output(event)
49         client.drain_output()

```

Código 3.8: Exemplo de código Python em tempo real para Linux

A execução do Código 3.8 irá colocar nosso sequenciador para funcionar. Para ouvir este sequenciador, é necessário executar também um sintetizador MIDI, como o ZynAddSubFx, que aceita eventos MIDI, apresentado na Figura 3.10.

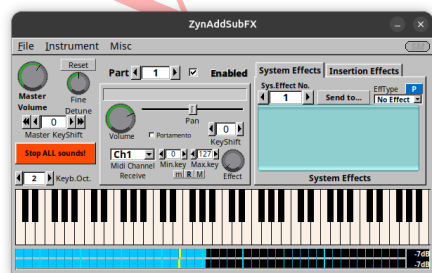


Figura 3.10: Tela do sintetizador ZynAddSubFx

Também é necessário conectar a saída do código com a entrada MIDI do sintetizador. Isso pode ser feito usando a ferramenta aconnectGUI, apresentada na Figura 3.11.

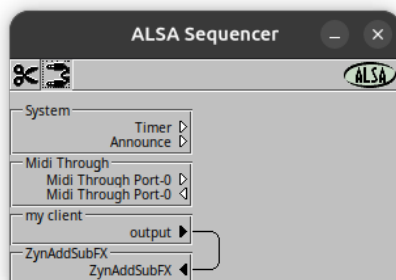


Figura 3.11: Ferramenta aconnectgui para conexão de portas MIDI no Linux

Há ainda outras bibliotecas para a criação de aplicações MIDI, como o Port-MIDI, RTMIDI e outras. Estas bibliotecas estão disponíveis em diversas linguagens de programação e possuem um funcionamento bastante similar, baseado no protocolo MIDI.

Exercícios 3.3

20. Utilizando uma biblioteca de programação, faça um programa que abre um arquivo MIDI e gere estatísticas sobre o mesmo como título, quantidade de trilhas, nome das trilhas, entre outras.
21. Utilizando uma biblioteca de programação, faça um programa que gera arquivos MIDI a partir de arquivos no formato ABC.
22. Faça um programa MIDI em tempo real que recebe mensagens MIDI em sua entrada e gere uma saída com a entrada transposta 1 oitava acima.
23. Faça um programa MIDI que receba mensagens MIDI e gere acordes para as mensagens em determinado campo harmônico.
24. Como funciona a parte de controle do protocolo MIDI? O que são as mensagens SySex e o MTC?

3.4 OSC

Quando começou a surgir instrumentos musicais elétricos ou eletrônicos, a comunicação dentro do dispositivo era feita por eletricidade. Pulsos elétricos, chaves, interruptores e moduladores de pulsos eram utilizados para controlar os sintetizadores, cuja saída era o sinal de áudio direto para a caixa de som, já que estes dispositivos eram fechados e não modulares.

Com o advento da tecnologia, surgiram os sintetizadores modulares, dispositivos para a criação musical que eram baseados em módulos onde cada módulo executava uma determinada função. A comunicação entre estes módulos utilizavam cordinhas, ou fiozinhos, também chamados de patch cables. Com isso, siglas como VCO (Oscilador Controlado por Tensão), VCF (Filtro Controlado por Tensão), VCA (Amplificador Controlado por Tensão), LFO (Oscilador de Baixa Frequência), entre outras, passaram a ser comuns e diferentes fabricantes, como EuroRack e Moog passaram a ser nomes comuns nos ambientes de música e tecnologia².

Com o advento do MIDI, em especial por ser um padrão para a indústria de equipamentos musicais, boa parte dos dispositivos passaram a aceitar e utilizar este protocolo de comunicação. Apesar da grande aceitação do MIDI, há algumas queixas quanto a este protocolo, em especial por ele ser muito fechado ou fechado demais.

Respondendo a estas limitações do protocolo MIDI, foi criado um protocolo chamado OSC (Open Sound Control - Controle Aberto de Som)³, criado em 2002, e definido como um protocolo de comunicação para sistemas computacionais e proposto para substituir o protocolo MIDI. O protocolo OSC foi desenvolvido Matt Wright e Adrian Freed no Center for New Music & Audio Technologies (CNMAT), na Universidade de Berkley, Califórnia e foi projetado para ser um método de comunicação preciso e de baixa latência para ser usado em performances musical em tempo real.

A ideia do OSC era ser um protocolo totalmente aberto e que pudesse ser definido sem o peso dos grandes nomes da indústria de equipamentos musicais, como acontece com o MIDI.

²Aos interessados em sintetizadores modulares emulados por software, o Alsa Modular Synth, ou AMS, é uma boa maneira de começar a entender estes conceitos.

³<http://opensoundcontrol.org/>

3.4.1 As mensagens OSC

O que o OSC define é o formato de suas mensagens, e isso foi o suficiente para sua adoção em diversas aplicações musicais. Uma mensagem OSC possui 3 partes: um endereço, uma string com tipos de dados de argumentos e os argumentos propriamente ditos.

Endereços OSC

As mensagens OSC possuem um endereço baseado em seu espaço de nomes, sempre iniciado com uma barra /, definido de forma hierárquica. Os endereços também são separados por barras, como o endereço de um arquivo em sistemas Unix.

Assim, podemos endereçar mensagens para um sintetizador usando o endereço /oscillator. O controle de volumes deste sintetizador poderá ser feito pelo endereço /oscillator/volume, e a frequência do oscilador pode ser enviada para o endereço /oscillator/freq. Os efeitos do oscilador podem ser endereçados como /oscillator/fx, sendo que o reverb pode ser endereçado como /oscillator/fx/reverb, e assim por diante.

Com isso, sistemas OSC podem facilmente filtrar suas mensagens e endereçá-las internamente na forma correta.

Passando parâmetros OSC

Um endereço osc (/oscillator/1/freq) pode ser seguido por uma string de tipos, chamada Data Type, e seus argumentos. Os argumentos padrões no OSC podem ser do tipo números inteiros, números decimais (ponto flutuante), textos ou dados binários. Além destes, há outros tipos de dados de argumentos que são comumente utilizados, mesmo não sendo parte do core do OSC. A Tabela 3.5 traz uma lista com os tipos mais comuns aceitos por dispositivos OSC.

A string que define os tipos de dados inicia sempre com uma vírgula, como por exemplo ",fis" para 3 argumentos sendo o primeiro do tipo float, o segundo do tipo inteiro, e o terceiro do tipo string.

Implementando clientes e servidores OSC

Existem diversas bibliotecas para a implementação de OSC para diferentes linguagens de programação. O Código 3.9 apresenta um servidor OSC em Python.

id	tipo
i	Inteiro 32 bits com sinal
f	Ponto flutuante IEEE 754 de 32 bits
s	Texto
b	Blob / Dados binários
h	Inteiro 64 bits com sinal
d	Ponto flutuante IEEE 754 64 bits (double)
t	OSC-timetag 64 bits
S	String alternativa
c	Caractere ASCII 32 bits
r	Cor RGBA 32 bits
m	mensagem MIDI 4 bytes
T	Boolean True (no data)
F	Boolean False (no data)
N	Nil/Null (no data)
I	Infinito (Utilizado para disparar eventos)

Tabela 3.5: Tipos de dados OSC

O servidor está sendo executado localmente, usando o protocolo UDP na porta 5005, e tem um método definido para tratar as mensagens do tipo /filter.

```
50 # pip3 install python-osc
51
52 from pythonosc.dispatcher import Dispatcher
53 from pythonosc.osc_server import BlockingOSCUDPServer
54
55 # Funcao para tratar as mensagens enviadas para o endereco "/filter
56 "
57 def print_handler(address, *args):
58     print(f"[{address}] {args}")
59
60 dispatcher = Dispatcher()
61 dispatcher.map("/filter", print_handler)
62
63 IP = "127.0.0.1"
64 PORT = 9000
65 server = BlockingOSCUDPServer((IP, PORT), dispatcher)
66
67 print(f"Servidor no {IP}:{PORT}")
68 try:
69     server.serve_forever()
70 except KeyboardInterrupt:
71     server.server_close()
```

```
71 print("Server stopped.")
```

Código 3.9: Exemplo de código servidor OSC em Python

Já o cliente tem um código mais simples, com o endereço do servidor e a criação de uma mensagem para o endereço /filter.

```
73 # pip3 install python-osc
74 from pythonosc import udp_client
75 import time
76
77 # Configura o cliente para enviar para localhost na porta 5005
78 client = udp_client.SimpleUDPClient("127.0.0.1", 5005)
79
80 # Envia uma mensagem para o endereço "/filter" com o valor 0.5
81 volume = 0.1
82 while True:
83     client.send_message("/volume", volume)
84     volume = volume + 0.1
85     time.sleep(0.5)
```

Código 3.10: Exemplo de código servidor OSC em Python

3.4.2 Um protocolo muito aberto

Apesar de ser definido como um protocolo, o OSC é aberto e flexível e não define diversas convenções que estão definidas no protocolo MIDI. Por esta razão, o nome protocolo aqui não segue o que costumamos ter como definição de um protocolo na pilha de protocolos TCP/IP, por exemplo. No OSC não existem papéis, mensagens, ou conexões, também não existem portas, eventos ou espaços de nomes definidos. Nem mesmo protocolos de comunicação são definidos no OSC.

• Protocolo de transporte não definido

Para garantir baixa latência, grande interoperabilidade e facilidade de comunicação, muitos dispositivos OSC usam o protocolo de transporte UDP e comunicação multicast sobre TCP/IP. NO entanto, isso não é obrigatório e outros protocolos, como TCP, ou endereçamentos, como unicast, podem ser usados. O protocolo OSC não implica em comunicação com UDP e pode ser possível usar OSC com TCP ou mesmo com conexão em redes que não usam o endereçamento com IP, como a comunicação Bluetooth ou XBee.

Papéis de rede não definidos

Em redes de computadores, temos 2 papéis bem definidos: cliente e servidor. Estes papéis são importantes para definir o início de uma comunicação pois espera-se que um cliente conheça o IP do servidor, seu protocolo de transporte (TCP / UDP) e sua porta de rede para acessar o servidor. Mesmo para comunicação UDP Multicast ou Broadcast, há a necessidade de separar tais papéis, já que o servidor é iniciado em uma determinada porta e fica aguardando o recebimento de mensagens nesta porta. Estes papéis estão associados apenas ao início de uma conexão e não dizem respeito ao fluxo de informação pois, após a comunicação ser iniciada, ela pode ocorrer em ambos os sentidos (full-duplex). No entanto, não é possível iniciar uma comunicação sem ser pelo cliente e sem conhecer estes dados: IP, protocolo de transporte e porta.

Na comunicação OSC não há nada definido sobre tais papéis e a escolha entre implementações de clientes ou servidores vão depender do desenvolvedor.

Indefinição das mensagens

O OSC também não possui um padrão definido de mensagens, como Note On ou Note Off, ou os atributos destas mensagens. Tudo é aberto neste protocolo e o programador pode definir suas próprias mensagens. Esta definição, que será apresentada a seguir, traz muita flexibilidade mas também pode ser um problema para conectar dispositivos distintos, já que é necessário conhecer o espaço de nomes e os atributos para garantir a comunicação entre dispositivos OSC.

Indefinições e não comunicação

O protocolo OSC é aberto e por isso, tudo fica tudo a critério do programador. Podemos então dizer que há a chance de dois dispositivos OSC não conseguirem se conectar por

- ♪ utilizar protocolos diferentes, como um ser UDP e o outro usar TCP.
- ♪ usando o mesmo protocolo de transporte, por exemplo, UDP, não se conectarem por usar portas de rede diferentes, como um cliente que tenta acessar o servidor na porta 5000 e o servidor estar sendo executado na porta 5001.

- ♪ ambos terem sido implementados usando o mesmo papel, ambos serem clientes ou ambos serem servidores, já que o protocolo não define tais papéis, e novamente a comunicação não irá acontecer.
- ♪ dois dispositivos podem conseguir estabelecer a conexão de rede entre si mas isso não garante a comunicação devido a possibilidade de os mesmos usarem endereços diferentes para realizar a mesma ação.

Com isso, entendemos que a abertura do protocolo traz uma flexibilidade enorme por não ter estas padronizações mas que isso pode ser um problema gerado pela falta destes padrões.

A necessidade de adaptar mensagens e endereços para cada caso faz com que o protocolo acabe sendo tão complexo quanto as mensagens exclusivas do MIDI (SysEx).

No entanto, apesar de tais dificuldades, o OSC é um protocolo amplamente utilizado em aplicações musicais muitas aplicações, como o ambiente da linguagem de programação musical Sonic Pi, são totalmente baseados em mensagens OSC.

Exercícios 3.4

25. Encontre um sintetizador que possui suporte ao protocolo OSC e faça um controlador que altere os parâmetros deste sin-

tetizador. Se achar adequado, utilize a biblioteca Light OSC (liblo) para esta implementação.

◆ Respostas dos exercícios do capítulo 3.

1. Existe um simulador assim...
2. Há hoje pesquisas que mostram como capturar o som por câmeras de alta resolução.
3. Os eventos estão disponíveis na internet.

Pre Print
V.O.O.

Capítulo 4

Áudio digital

No capítulo anterior vimos como representar computacionalmente informações sobre música simbólica. A representação simbólica da música pode até ser bastante eficiente para representar música em instrumentos musicais por trazer informação de como tocar estes instrumentos, seja por uma pessoa ou por um sistema computacional. No entanto, por ser uma representação limitada a normalmente trazer informações apenas para representar altura, duração e intensidade, dificilmente consegue representar outros elementos utilizados para construir a expressividade de uma performance musical. A representação simbólica da música, também costuma ser ineficiente para representar instrumentos cuja afinação foge da divisão da escala ocidental e para a representação de sons mais complexos, como o da voz humana. No caso particular da voz, há um desafio de criar um algoritmo de síntese eficiente que aceite tanto dados musicais, como altura e intensidade, quanto dados relacionados a voz humana, como articulação, tessitura, ou diferentes registros. Neste caso, o melhor a fazer é não utilizar síntese, mas gravar a voz ou o instrumento a partir do seu som original.

Neste capítulo iremos discutir o que se passa neste processo de gravação do som no computador, o chamado Áudio Digital. Esta gravação passa por diversas etapas e possui uma infinidade de detalhes como a ligação de cabos, tipos de microfones, formatos de arquivo para armazenamento dos dados gravados, entre outros.

Além disso, uma vez que podemos estabelecer os parâmetros do áudio digital para a gravação e armazenamento de sons, podemos pensar também em criar sons digitalmente, por meio de algoritmos de síntese sonora e geradores

de som, como osciladores e geradores de sinais. Também existe uma infinidade de algoritmos que permitem o processamento e a modificação do áudio digital.. Também existem maneiras de analisar este sinal de áudio digital e extrair informações e características do mesmo.

Neste capítulo iremos verificar alguns destes pontos, iniciando pelo entendimento do som, passando por formatos de arquivos de som até chegar em geradores digitais de som.

4.1 No início era o som

Antes de começar a tratar do áudio digital, podemos começar a pensar um pouco sobre o som. O som é a propagação acústica de uma onda mecânica, sendo assim, parte do estudo desta área da física. Por onda, entende-se a transferência de energia sem deslocamento de massa, como acontece também com ondas de calor ou as ondas do mar. Por esta razão, aprendemos desde muito cedo que o som precisa de um ambiente elástico de matéria para se deslocar e, conseqüentemente, não existe propagação de som no vácuo. A onda sonora, quando propagada no ar, irá comprimir e descomprimir o ar em seu caminho, deslocando a energia sonora sem, com isso, deslocar o ar em si. Dentro da acústica temos a possibilidade de tratar a onda sonora de maneira a melhorar a acústica de um ambiente, com os chamados tratamentos acústicos. Difusores acústicos, abafadores, armadilhas de graves e de agudos, refletores e amplificadores sonoros acústicos, como os ressonadores de Helmholtz são estruturas que permitem de tratar a acústicas de ambientes. O som pode ser barrado por meio destes tratamentos acústicos e existe toda uma área de pesquisa que trabalha com isolamentos e tratamentos acústicos mas, certamente, esta não é uma discussão para este livro.

Comumente desenhamos o som como uma espécie de onda direcional mas não é assim que o evento físico acontece. Na verdade, podemos imaginar a onda sonora da maneira similar à perturbação da água na superfície de um lago quando o mesmo está em repouso e atiramos uma pedra ao lago. Veremos na superfície a perturbação das águas, mas podemos imaginar que esta perturbação também irá ocorrer dentro da água, como se fosse uma metade de esfera. No caso do som, esta propagação também irá acontecer acima, como em uma esfera completa, deslocando a onda sonora em todas as direções possíveis por meio da variação da pressão atmosférica do ambiente.

4.1.1 Características do som

O som possui algumas características físicas que torna possível classificar e comparar dois sons distintos a partir destas características. Uma onda sonora possui uma determinada intensidade, e também terá uma duração. Além disso, a onda sonora pode ser ou não periódica. Caso ela seja periódica, ela irá se repetir em uma determinada frequência, gerando uma altura.

Intensidade

O repouso do ambiente traz a não existência do som, também chamado de **silêncio**. Uma perturbação um pouco maior, como uma onda sonora pequena, gera um som de volume baixo, de menos intensidade. Aumentar o tamanho desta onda significa aumentar a intensidade da mesma, ou aumentar seu volume. Esta intensidade do som é medida em decibéis.

194 dB	- Bomba Atômica
184 dB	- Maior recorde som automotivo
180 dB	- Lançamento de foguete a 30m
130 dB	- Armas de fogo
120 dB	- Decolagem de avião a jato
120 dB	- Bateria acustica
116 dB	- Saxofone
115 dB	- Show de rock (2-3 metros de distância)
110 dB	- Bar Lotado
105 dB	- Buzina (8 metros de distância)
100 dB	- Latido de cães
99 dB	- Martetelete perfurador
92 dB	- Guitarra ligada no amplificador
90 dB	- Piano acustico
80 dB	- Despertador de campainha
80 dB	- Aspirador de pó
78 dB	- Liquidificador
75 dB	- Carro Ligado (dentro de um carro com motor ligado)
65 dB	- Ronco humano
60 dB	- Conversação alta
50 dB	- Conversação normal
40 dB	- Conversação silenciosa (quase sussurrando.)
30 dB	- Biblioteca (Em uma sala de espera, em total silêncio.)
20 dB	- Superfície da Terra.

Enquanto a física mede as intensidade em decibéis, a percepção desta característica pode ser relacionada a potência do som, ou se o mesmo é forte ou fraco. Vale lembrar que a intensidade de um determinado som pode variar ao longo do tempo, iniciando forte e terminando fraco, como, por exemplo, as badaladas de um sino.

Duração

Além da intensidade, um som também tem uma duração específica. Esta característica, na física, é medida em minutos, segundos ou milissegundos, mas a percepção da mesma pode classificar os sons entre sons curtos ou longos. Além

disso, um som pode ser composto de vários outros sons, de diferentes durações e intensidades. Isso é o que costumamos chamar de música: a organização de eventos sonoros no tempo.

Em música, a organização dos sons no tempo é o que define ritmo e duração dos eventos sonoros, permitindo os contrastes entre sons e silêncio ao longo do tempo musical.

Periodicidade

Um som pode também ser periódico ou não periódicos. Sons periódicos são ondas sonoras que se repetem em um determinado intervalo de tempo, sendo este intervalo regular e constante. As ondas periódicas possuem algumas características como:

- ♪ **Crista e Vale:** A crista é o ponto mais alto da onda e o vale é o ponto mais baixo. Estes pontos servem para entendermos a repetição periódica destas ondas.
- ♪ **Amplitude (A):** É a altura máxima da onda, medida do eixo central até a crista ou até o vale.
- ♪ **Comprimento de onda (λ):** É a distância de um ciclo completo da onda, ou a distância entre duas cristas consecutivas ou dois vales consecutivos.
- ♪ **Período (T):** É o tempo que a onda leva para completar um ciclo completo, ou o tempo que leva para uma crista passar por um ponto e a próxima crista chegar nesse mesmo ponto.
- ♪ **Frequência (f):** É o número de ciclos que a onda completa em um segundo. É calculada como o inverso do período: $f = \frac{1}{T}$. A unidade oficial é o Hertz (Hz).
- ♪ **Velocidade (v):** É a velocidade com que a onda se propaga pelo meio, calculada multiplicando o comprimento de onda pela frequência: $v = \lambda \times f$

Pensando em termos físicos, estes sons possuem uma forma de onda que oscila com uma frequência fixa, resultando em um som com uma altura definida que pode ser percebido como notas musicais. Sons periódicos possuem a sensação de altura musical relacionada ao período

O contrário de um som periódico seria um som não-periódico ou aperiódico, ondas cujo comportamento não possui repetições. Estes sons não periódicos são entendidos como ruídos por serem desordenados e irregulares.

Matematicamente, uma senóide é um bom exemplo de um som periódico por possuir ciclos idênticos e repetitivos. Outros exemplos seriam ondas quadradas, triangulares e dentes-de-serra. Estas ondas serão melhores explicadas em breve. Apesar de ser muito úteis para entender o som, tais ondas não existem na natureza.

Altura

A altura é a propriedade do som que permite classificar o mesmo por parâmetros como grave, médio ou agudo. Um som agudo é chamado de alto, enquanto um som grave é chamado de baixo. Cotidianamente, há uma confusão pois chamamos sons com muita intensidade / volume, de alto, e sons com pouca intensidade / volume, de sons baixos. Apesar de estes termos se confundirem no dia a dia, no entendimento das características do som eles são distintos e bem definidos. A altura diz respeito ao som ser grave, médio ou agudo, enquanto a intensidade ou volume diz respeito ao som ser forte ou fraco.

Desta maneira, enquanto a intensidade em decibel mede o tamanho das ondas sonoras, a altura será medida pela frequência ou o número de ciclos por segundo de uma onda sonora periódica. Pela física, altura é medida em hertz. Uma frequência alta possui mais ciclos por segundo e sons que chamamos de agudos, como assovios, sons sibilantes ou a voz feminina. Já as frequências baixas possuem menos ciclos por segundo e sons que chamamos de graves, como motores, um contrabaixo ou a voz masculina.

O entendimento da altura do som é importante para entender a nossa percepção do som, já que a audição humana possui limitações. As pesquisas, como a de Fletcher-Munson, nos apresentam que as pessoas são capazes de ouvir e distinguir, em média, sons com frequências entre 20Hz e 20.000Hz. Existem sons que soam abaixo desta frequência, e são chamados de infrassônicos, enquanto os sons acima desta frequência são chamados de ultrassônicos. Alguns animais, como cães, golfinhos e morcegos, são capazes de captar sons muito acima do limite da audição humana. Além disso, a audição humana também capta cada frequência com uma intensidade diferente, como mostrado nas curvas de Fletcher-Munson, apresentadas na Figura 4.1.

Musicalmente falando, a altura é a da característica que nos permite iden-

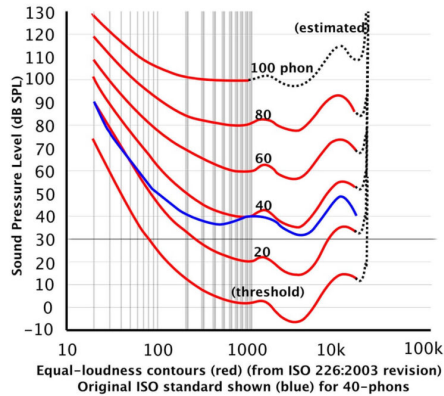


Figura 4.1: Curva de Fletcher-Munson

tificar notas musicais em sons periódicos.

4.1.2 Da acústica para a elétrica: O sinal de áudio

Quando colocamos um microfone diante de uma onda sonora, a mesma irá se transformar em uma onda elétrica. Neste momento, mudamos de área na física, abandonando a acústica e adentrando na área da **elétrica/áudio** por deixar de lado a energia mecânica da onda do som e passar para o domínio da onda no sinal elétrico. Agora, o som deixa de ser som e passa a ser chamado de áudio, ou de sinal de áudio, um campo de estudo da **engenharia elétrica** e da **engenharia de áudio**. Neste momento, toda uma gama de Processamento de sinais passa a ser possível com o som.

O dispositivo utilizado para captar o som é o microfone, um dispositivo transdutor que converte um tipo de energia em outro. No caso do microfone, a transdução é mudar de onda sonora para onda elétrica¹.

Existem vários tipos distintos de microfones, com diferentes impedâncias, direcionalidades e sensibilidades. Existem microfones que são diferentes em sua construção, como microfones dinâmicos e condensadores, além de modelos de microfones que atuam melhor em determinadas faixas de frequências, como microfones que captam melhores sons graves ou que captam melhor sons agudos.

¹ Temos outros transdutores que atuam de maneira similar como resistências que transformam energia elétrica em calor, motores que transformam energia elétrica em energia mecânica, dínamos que transformam energia mecânica em elétrica e altifalantes que transformam energia elétrica em onda sonora.

Assim, podemos entender que as propriedades do som como intensidade ou faixa de frequências também serão propriedades do sinal de áudio.

Assim como o som, o sinal de áudio também pode ser processado e modificado. Dispositivos de processamento de sinais de áudio como filtros, compressores, equalizadores, distorcedores, moduladores e toda uma gama de efeitos que habitam espaços de palcos perto dos pés de guitarristas mundo afora são exemplos destas modificações. Também temos toda uma gama de equipamentos de áudio como mesas misturadoras, gravadores, reprodutores, divisores de sinal, entre outros.

Da mesma forma que temos fontes sonoras acústicas, no áudio também podemos ter fontes de sinal que não passam por um microfone mas que são criadas de forma totalmente elétrica. Geradores de sinais elétricos permitiram que os primeiros instrumentos totalmente elétricos fossem inventados como o Theremin ou o Ondas Martenot envolvendo a **eletrônica** e a **microeletrônica** em nosso arcabouço de áreas envolvidas.

Também é possível analisar o áudio para extrair informações sobre o mesmo. Multímetros, osciloscópios e decibelímetros fazem parte do rol de equipamentos de podem ser utilizados como ferramentas para medir o som a partir de sua representação como sinal de áudio.

Exercícios 4.1

1. Encontrar um simulador que apresenta o som na física.	2. Será possível pensar em digitalizar o som sem passar primeiramente pelo áudio?
--	---

4.2 O Sinal de Áudio Digital

Quando o som já está representado como sinal de áudio, digitalizá-lo e transformar o mesmo em um sinal digital é um passo que nos permite utilizar este sinal nos computadores. A digitalização do áudio passa por um elemento chamado conversor analógico-digital (AD), presente na entrada de microfone de nossas interfaces de áudio. E é aqui que o som entra a computação musical propriamente dita e o encontro da música com o processamento digital de sinal (DSP - Digital Signal Processing).

O sinal de entrada do conversor AD é o áudio, um sinal contínuo, e este dispositivo coleta amostras deste sinal em uma determinada frequência, trans-

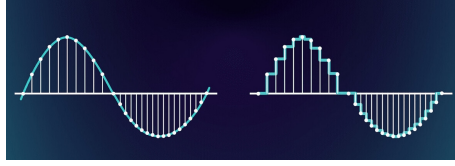


Figura 4.2: Conversão do sinal analógico para o sinal digital

formando o sinal contínuo do áudio analógico para um sinal discreto do áudio digital, conforme mostrado na Figura 4.2.

Quando o áudio é digitalizado, ele se torna um sinal discreto que pode ser representado computacionalmente por um array de amostras. Desta maneira, podemos pensar que o sinal de áudio digital é uma sequência de números onde cada valor é chamado de amostra ou sample e representa uma amostra do áudio. O áudio em si é a coleção destas amostras, normalmente armazenada em um array ou vetor. Entendemos computacionalmente que este array possui uma quantidade de amostra e é finito. Por esta razão, a representação matemática comum do áudio digital é a forma:

$$..., x[n-1], x[n], x[n+1], ... \quad (4.1)$$

Assim, o sinal de áudio digital depende de algumas configurações de sua amostragem como Taxa de amostragem, Codificação e Quantização e da sua quantidade de canais, como será visto a seguir.

4.2.1 Taxa de Amostragem

A taxa de amostragem no áudio digital diz respeito a quantas amostras por segundo do sinal contínuo do áudio da entrada serão feitas no processo de discretização do sinal. A taxa de amostragem (Sample Rate) é representada por uma grandeza em Hertz, que, neste caso, representa a quantidade de amostras por segundo. Um sinal amostrado com uma amostra por segundo tem a taxa de amostragem de 1 Hz, um sinal amostrado com 1000 amostras por segundo tem a taxa de amostragem de 1K Hz.

A taxa de amostragem servirá como uma espécie de régua para calcular a duração do áudio em amostras. Se temos uma taxa de 48.000 amostras por segundo, podemos dizer que, para gerar 1 segundo de áudio teremos um array

com 48.000 amostras, que serão tocadas em 1 segundo. Logo, se queremos 5 segundos de áudio, precisamos de um array com $48.000 * 5 = 240.000$ amostras para representar este som, e 1 milissegundo do áudio terá 48 amostras de áudio nesta taxa de amostragem.

A taxa de amostragem irá definir também um parâmetro importante no sinal digitalizado que é a frequência máxima representada. Sabendo que uma onda precisa completar seu período, ou seja, "subir e descer" para ser ouvida, precisamos de ao menos 2 amostras para representar tal sinal, um para cima e um para baixo, em uma representação limítrofe do sinal de áudio. Por isso, a menor onda representada em um áudio digital é a metade de sua taxa de amostragem. Chamamos esta relação de teorema da amostragem de Shannon-Nyquist, ou simplesmente como teorema de Nyquist.

$$FreqMax = SR/2 \quad (4.2)$$

Com isso, podemos entender que frequência altas, também chamados de sons agudos, precisam de uma taxa de amostragem maior para serem representadas, enquanto frequências baixas, também chamadas de sons graves, podem ser representadas por taxas de amostragens menores.

O que acontece com as frequências sub-amostradas e que não podem ser representadas por determinada taxa de amostragem? Elas serão erroneamente representadas como uma frequência mais baixa, causando um fenômeno chamado **rebatimento** ou **aliasing**. Por esta razão, a conversão de um sinal analógico para um sinal digital utiliza um filtro de passa baixa que elimina todas as frequências acima da frequência de Nyquist para evitar que tais frequências sejam rebatidas e representadas erroneamente no sinal amostrado.

Este filtro de passa baixa é uma parte importante do conversor AD e o mesmo pode ser construído com um circuito simples, composto de capacitor e resistor, já que o áudio antes da conversão é um sinal elétrico. Vale dizer que filtros de passa baixa não são "cirurgicos" e não cortam exatamente na frequência desejada. O comportamento dos filtros é atenuar o sinal a partir de uma frequência de corte, o que faz com que o mesmo tenha um declive, também chamado de slope.

Desta maneira, como a audição humana costuma ser limitada a 20kHz, a taxa de amostragem mínima para representar todo o espectro audível é, teori-

camente, 40kHz. No entanto, devido ao slope do Filtro de passa baixa utilizado na conversão AD, utilizar a frequência de 40.000Hz atenua ou mesmo elimina as frequências mais altas. Por esta razão, a taxa de amostragem do CD ser de 44.100Hz, e a amostragem do DVD ser de 48KHz.

Apesar de existir este limite da audição humana, existe a possibilidade de o áudio digital ser amostrado com taxas de maiores, como 96kHz ou 192kHz. No entanto, vale ressaltar que, musicalmente falando, dobrar a frequência significa aumentar 1 oitava musical. Assim o áudio com 96kHz representaria 1 oitava a mais que o áudio amostrado a 48kHz, sendo que esta oitava não é audível para uma grande parte da população.

4.2.2 Quantização e Codificação

Após a separação das amostras em um determinado intervalo de tempo, cada amostra digital precisa ser colocada em um formato específico de representação onde a amplitude do sinal de áudio será convertida em valores numéricos discretos. Este processo é chamado de **Quantização** e define quantos bits serão usados para representar cada amostra de áudio.

A taxa de amostragem é uma frequência, medida em Hz, e influencia as frequências representadas no sinal digital. Já a quantização é relacionada a amplitude e por isso influencia nas amplitudes do sinal digital. Desta maneira, aumentar a quantidade de bits na amostragem não modifica em nada a capacidade de representar mais ou menos frequências. A faixa do espectro representado é relacionada apenas a taxa de amostragem. Aumentar a quantidade de bits irá alterar a faixa dinâmica representada, e isso diz respeito apenas a amplitude do sinal.

Relação sinal-ruído

Quanto mais bits temos para representar a curva do sinal, mais próximo o sinal digital será do sinal analógico. E o que acontece quando usamos poucos bits? Vamos pensar em situações extremas: representar o sinal com 1 bit. Neste caso, temos apenas um pulso, que varia entre zero e um, o que torna impossível qualquer variação dinâmica. Musicalmente falando, isso significa que não tem como mudar de forte para médio, ou para fraco, ou pianíssimo. Temos apenas volume/amplitude máxima ou silêncio.

Para quantificar o sinal de áudio, medimos a amplitude do sinal de entrada e

definimos o valor das amostras. Ainda pensando no sinal com 1 bit, utilizamos um threshold para medir a amplitude do sinal de entrada onde qualquer valor acima do threshold assume o valor 1 e qualquer coisa abaixo disso terá o valor 0. Logo, a definição deste threshold irá implicar diretamente na amostragem do sinal pois se o threshold for colocado em um valor muito alto, os sons baixos serão todos considerados silêncio. Para não correr o risco de eliminar estes sons, pode-se assumir uma estratégia de quantização de colocar o threshold em uma amplitude bem baixa, permitindo assim, representar também sons baixos. No entanto, isso fará, com que qualquer valor baixo seja representado com amplitude máxima, inserindo ruídos que antes seriam imperceptíveis no sinal. Por isso, a quantidade de bits na quantização irá definir o que chamamos de relação sinal/ruído (em inglês: SNR - Signal-to-Noise Ratio).

Adicionando mais um bit em nosso exemplo, temos um sinal com 2 bits, ou 4 valores, e 4 thresholds para amostrar nosso sinal. Isso já é muito melhor pois permite ter silêncio (0), fraco (1), médio (2), e forte (3). Cada bit acrescentado na quantização irá adicionar 6 *dB* na relação sinal/ruído).

Exemplos de quantizações comuns são a do CD, que usa amostras de 16 bits, com uma relação sinal/ruído de aproximadamente 96 *dB*. Já o áudio em DVD usa amostras de 24 bits, com SNR de aproximadamente 145.5 *dB*, uma relação acima dos limites da percepção humana.

Ruídos de quantização

No processo de quantização, cada intervalo de amplitude do sinal analógico será aproximado para o nível mais próximo representável no digital. Estes valores são mapeado em um único código de N bits que normalmente representa o valor médio do intervalo e pontos na fronteira de dois intervalos devem ser “arredondados” para cima ou para baixo. Por isso, pequenos ruídos de quantização poderão surgir no processo de conversão AD, especialmente em representações com poucos bits. Estes ruídos de quantização podem distorcer completamente o som original.

Tomando novamente como exemplo um caso extremo, com apenas 1 bit, e um sinal senoidal. A representação de uma senóide com apenas um bit será uma onda quadrada, que mantém a mesma frequência original mas cujo som é bastante distorcido. Ainda usando este exemplo, podemos notar que o ruído de quantização pode não ser o mesmo para qualquer sinal e depende do sinal de entrada. A amostragem de uma onda quadrada, por exemplo, com apenas 1

bit não irá adicionar necessariamente ruído ao sinal original. Por isso, o ruído de quantização depende das características espectrais do sinal de entrada, pois estas características irão tornar o ruído de quantização mais ou menos perceptível.

Uma solução para eliminar o ruído de quantização é adicionar ruídos aleatórios muito graves em um nível quase imperceptível para tornar o sinal mais variado. Esta técnica é chamada de *dithering*).

Codificação

Além do tamanho das amostras de áudio, outro fator importante na representação digital é o formato destas amostras. Esta etapa é chamada de codificação e pode incluir formatos como

- ♪ inteiros sem sinal,
- ♪ inteiros com sinal,
- ♪ inteiros com sinal em complemento de 2,
- ♪ com bytes ordenados na sequência natural (*little-endian*) ou inversa (*big-endian*),
- ♪ ponto flutuante

Estas variações podem trazer diferentes entendimentos do que seria, por exemplo, o silêncio. Se tomamos por exemplo a representação com inteiro sem sinal com 16 bit, o valor do sinal irá variar entre 0 e 65.535, sendo que o estado de silêncio ou repouso seria o valor 32.768. Já em um sinal representado por inteiro com sinal, o sinal de áudio digital irá variar entre 32.767 e -32.768 sendo que o silêncio será o valor 0.

Quantização em ponto-flutuante

Uma solução para melhorar a relação sinal/ruído para sons de menor amplitude é usar técnicas de quantização *não-linear*. Isso é, usar regiões mais finas para amplitudes mais baixas, em detrimento de regiões mais grosseiras nas amplitudes maiores. Para isso, utiliza-se abordagens como a quantização em ponto flutuante e quantização logarítmica.

De maneira muito simplista, podemos imaginar a codificação com ponto flutuante como um número com 8 dígitos sendo que um deles é a vírgula. Nesta

codificação, podemos representar valores grandes com pouca precisão deixando a vírgula à direita, como o valor 9999999,. Valores grandes que usam 7 casa à esquerda da vírgula não possuem nenhuma casa para sua parte fracionária, o que não é relevante já que 0,1 é um arredondamento muito bom para um valor tão grande como 9999999. Colocando a vírgula totalmente à esquerda podemos representar valores muito pequenos com muita precisão, como ,00000001. Pensando nesta representação, um erro de quantização com maior amplitude tem mais chance de passar despercebido se o sinal é muito mais forte, resultando em uma melhor relação sinal/ruído para vários níveis de amplitude simultaneamente.

O exemplo anterior é uma tentativa de explicar o ponto flutuante mas, na prática, não é assim que esta representação funciona. A representação destes valores no computador utiliza conceitos como mantissa e expoente e é chamado *float*. Vamos pular aqui a explicação de como se dá a construção de um float no computador mas podemos afirmar que nem todo valor pode ser representado por um float, o que é chamado de erro de arredondamento. Apesar desse erro de arredondamento adicionar ruído de quantização na quantização em ponto flutuante, este erro é muito menor do que os erros da codificação linear.

Na codificação em ponto flutuante, assumimos que 0 é valor para silêncio e que o valor do sinal varia entre -1 e 1, utilizando a região com mais precisão decimal da representação (como se fosse a vírgula à esquerda). Esta escolha de valores traz outra vantagem, pois torna mais simples a parte matemática da representação do áudio, já que estes valores são os mesmos utilizados matematicamente para representar seno e cosseno.

Contudo, limitar a representação do áudio em ponto flutuante entre os valores -1 e 1 não impede que valores acima disso também sejam utilizados, o que não é possível para valores representados por inteiros, por exemplo. Na representação em inteiros, valores acima do limite de 32.767 são automaticamente assumidos como este valor, adicionando muita distorção caso o ganho do sinal de entrada esteja muito alto. Já o sinal representado em ponto flutuante permite representar valores muito acima dos limites (-1 e 1), mesmo que com menos precisão, permitindo que o sinal seja posteriormente corrigido e adequado ao limite da representação. Esta característica é chamada em áudio de *headroom infinito*.

4.2.3 Quantidade de canais

Quando tratamos o sinal de áudio até aqui, pensamos em apenas um sinal de áudio, ou uma única fonte sonora. No entanto, um sistema de áudio analógico pode ter diferentes fontes sonoras. Um show pode ter no palco diversos instrumentistas, cada um com um microfone ou instrumento elétrico, gerando uma fonte sonora distinta. Normalmente, em uma situação assim, usamos um mixer, também chamado de mesa de som, para misturar estes sinais todos, somando-os e gerando um ou mais sinais de saída.

Quando esta mixagem gera um único canal de saída, chamamos este sinal de mono. Em gravações musicais, a mixagem costuma gerar um sinal de saída com 2 canais, chamado de estéreo. O sinal estéreo é composto de dois sinais independentes, um chamado canal da direita e outro esquerda. Tal sinal é assim mixado para ser ouvido por sistemas estereofônicos, como fones de ouvido ou aparelhos de som, onde um som será direcionado para o ouvido esquerdo e o outro para o ouvido direito do espectador.

Apesar destes sons serem normalmente relacionados um com o outro, um sinal estéreo é, na verdade, composto de 2 sinais totalmente independentes. É possível gravar um som no canal direito que é totalmente distinto do som do canal esquerdo.

Em áudio digital, quando temos uma fonte de entrada estéreo, como a entrada de microfone do computador, o sistema irá reconhecer esta entrada como 2 canais que poderão ser gravados juntos ou separados. O mesmo acontece com a saída do sistema. Uma saída de fone de ouvido é normalmente uma saída estéreo, que pode ter 2 canais, chamados de esquerdo e direito.

Apesar de as interfaces de som padrão terem apenas 1 entrada e uma saída estéreo, é comum termos hardware com mais entradas ou com mais saídas. Sistemas de sonorização com surround, por exemplo, podem ter 6 canais distintos, como é o caso de um sistema 5.1. Para gravar uma banda e ter sinais independentes para cada músico pode ser necessário utilizar uma interface com mais canais de entrada. É comum que estúdios de gravação possuam interfaces de áudio com 8, 12, 24, 36 ou 48 canais independentes para realizar estas gravações.

Esta questão também está presente em arquivos de áudio, que poderão ter mais de um canal. Um DVD que possui saída para surround, por exemplo, pode possuir diferentes canais de áudio, um para cada saída.

Exercícios 4.2

- | | |
|---|--|
| 3. Encontrar um simulador que apresenta o som na física. | 5. Sabendo que a audição humana ouve até 20KHz, qual a taxa de amostragem que representa tal frequência? |
| 4. Será possível pensar em digitalizar o som sem passar primeiramente pelo áudio? | 6. Ouça áudios em outras resoluções. São mesmo ruins? |

4.3 Utilizando áudio no computador

Uma vez que entendemos como o áudio é representado no computador, podemos começar a utilizar este áudio digital. Para isso, usaremos duas abordagens distintas: a utilização de arquivos de áudio e o Processamento de áudio em tempo real.

4.3.1 Utilizando arquivos de áudio

Uma vez que entendemos o chamado áudio digital, passamos a então verificar o formato de armazenamento de arquivo de áudio. Há diversos formatos possíveis para armazenar o áudio, que respondem por nomes que usam siglas como PCM, DPCM, PAM, PWM, DM, ADPCM, entre outros. Também há formatos de arquivos que não guardam o áudio amostra por amostra mas compactam o áudio, como o MP3, FLAC ou OGG. Neste trabalho focaremos no formato PCM e usaremos o mesmo para trabalhar com arquivos de áudio.

O formato PCM (Pulse Code Modulation) é um formato muito comum de arquivo que armazena as amostras de áudio em um vetor onde cada x_n representa o valor instantâneo de uma amostra. O formato PCM é caracterizado pela sua *taxa de amostragem* e pelo *formato da amostra* incluindo número de bits e tipo de codificação.

O formato de arquivos WAV (Waveform Audio File Format) é um formato muito comum para armazenar áudio no formato PCM. Este formato de arquivo contém um cabeçalho com as informações sobre as amostras de áudio do arquivo, como taxa de amostragem, profundidade de bits e número de canais (mono ou estéreo), seguido pela seção de dados, que armazena os valores PCM brutos.

Colocar o cabeçalho do arquivo wav aqui.

Apesar de ser simples criar um arquivo WAV manualmente, definindo seu cabeçalho e seus dados, há diversas bibliotecas que permitem trabalhar com este formato de arquivo, como a *libsoundfile*, que será usada nos próximos exemplos. A biblioteca *libsoundfile* possui funções para a escrita de dados em diversos formatos, como inteiro, short ou float.

Lendo os metadados do arquivo

Uma primeira experiência é ler os metadados de um arquivo de áudio e verificar as informações que estão no cabeçalho do arquivo. Utilizando a biblioteca *libsndfile*, esta operação requer apenas um pouco de conhecimento sobre a API desta biblioteca, como apresentado no Código 4.1.

```
86 #include <stdio.h>
87 #include <sndfile.h>
88
89 int main() {
90     SNDFILE *arquivo_audio;
91     SF_INFO info_audio;
92
93     // Zera a estrutura e define o formato como 0 para leitura
94     info_audio.format = 0;
95
96     // 1. Abre o arquivo em modo de leitura
97     arquivo_audio = sf_open("seu_audio.wav", SFM_READ, &info_audio)
98     ;
99
100     if (!arquivo_audio) {
101         printf("Erro ao abrir o arquivo: %s\n", sf_strerror(NULL));
102         return 1;
103     }
104
105     // Exibe algumas informacoes sobre o arquivo
106     printf("Arquivo aberto com sucesso!\n");
107     printf("Taxa de amostragem: %d Hz\n", info_audio.samplerate);
108     printf("Canais: %d\n", info_audio.channels);
109     printf("Total de frames: %lld\n", info_audio.frames);
110
111     // 2. Fecha o arquivo corretamente
112     sf_close(arquivo_audio);
113     return 0;
```

113 }

Código 4.1: Extrair informações de um arquivo de áudio

Escrevendo em um arquivo de áudio

A primeira tarefa que iremos fazer, para criar arquivos de áudio é criar um arquivo de áudio com silêncio. Na prática, se pensarmos que o altifalante de seu computador faz som sendo “chacoalhado” para frente e para trás, sendo que o “pra frente” será o 1, e o “pra trás” será o -1. Com isso, assumimos que a posição de repouso será o 0, o que nos permite assumir que nossas amostras, caso sejam ponto flutuante, irão gerar silêncio quando o valor zero é atribuído para todas as amostras.

Cuidado! Array com outros valores constantes (e.g. 1 ou -1) também produzem silêncio mas o resultado pode não ser o esperado! Isso significa uma constante, que chamamos de DC Offset, ou o deslocamento do sinal para cima ou para baixo do que seria o zero ou silêncio. Colocar 1 em todas as amostras também gera silêncio, mas faz o altifalante ficar na posição “pra frente” em vez de em repouso, o que pode levar a danos no dispositivo.

Para gerar nosso arquivo de áudio com silêncio, começamos definindo o formato PCM, com os valores para sample rate, quantidade de canais e quantidade de bits. Uma vez definido o formato e criado o arquivo, basta inserir zero em todas as amostras, como apresentado no Código 4.2. Note que o arquivo gerado possui 3 segundos, já que o tamanho dos dados armazenados é 3 vezes maior que a taxa de amostragem do arquivo.

```
1 // sudo apt-get install libsndfile1 libsndfile1-dev
2 //gcc exemplo1.c -o exemplo1 -lsndfile
3
4 #include <stdio.h>
5 #include <sndfile.h>
6 #define SAMPLE_RATE 48000
7
8 int gerar_sinal(float * samples){
9     for(int i = 0 ; i < SAMPLE_RATE * 3; i++)
10         samples[i] = 0.0f;
11 }
12
13 int main(void) {
14     SF_INFO sfinfo;
15     sfinfo.channels = 1;           // Mono
```

```
16     sinfo.samplerate = SAMPLE_RATE;    // 48 kHz
17     sinfo.format = SF_FORMAT_WAV | SF_FORMAT_PCM_16; // WAV file
    with 16-bit PCM encoding
18
19     SNDFILE *sndfile = sf_open("output.wav", SFM_WRITE, &sinfo);
20     if (!sndfile) {
21         printf("Erro: %s\n", sf_strerror(NULL));
22         return 1;
23     }
24     float samples[SAMPLE_RATE * 3];
25     gerar_sinal(samples);
26
27     sf_count_t frames_written = sf_writef_float(sndfile, samples,
    SAMPLE_RATE * 3);
28     if (frames_written < SAMPLE_RATE * 3) {
29         printf("Warning: Not all frames were written.\n");
30     }
31     sf_close(sndfile);
32     return 0;
33 }
```

Código 4.2: Geração de arquivos de áudio com silêncio

4.3.2 Processamento de Áudio em tempo real

O processamento de áudio em tempo real implica em trabalhar com as entradas e saídas de som do computador. Como o gerenciamento de entradas e saídas é parte da tarefa do sistema operacional, a primeira questão que temos é qual o sistema operacional que será utilizado em nossa atividade.

O sistema operacional irá fornecer uma ou mais APIs para o processamento de áudio em tempo real. Nos exemplos aqui presentes usaremos o Linux e a API do ALSA (Advanced Linux Sound Architecture) para as nossas implementações. Outras APIs estão disponíveis para este e outros sistemas operacionais. O Anexo 4 apresenta algumas possibilidades distintas para isso.

Identificando os dispositivos de som

A primeira tarefa é identificar os dispositivos de som disponíveis no computador. A API irá nos fornecer uma lista com estes dispositivos e trará também informações sobre o áudio neste dispositivo, como a taxa de amostragem e formato das amostras aceitas pelo dispositivo. Este código é mostrado no Código 4.3.

```
34 // gcc alsa-list.c -o alsa-list -lm -lasound
35 #include <stdio.h>
36 #include <alsa/asoundlib.h>
37
38 int main() {
39     int card_num = -1;
40     int err;
41     snd_ctl_t *card_handle;
42     snd_ctl_card_info_t *card_info;
43     snd_ctl_card_info_alloca(&card_info);
44
45     printf("--- Listando Placas de Som Ativas ---\n");
46     // Loop para encontrar todas as placas de som disponiveis
47     while (1) {
48         if ((err = snd_card_next(&card_num)) < 0) {
49             fprintf(stderr, "Erro: %s\n", snd_strerror(err));
50             break;
51         }
52         if (card_num < 0) break;
53
54         char card_name[16];
55         sprintf(card_name, "hw:%d", card_num);
56
57         // Abre o controle da placa de som especifica
58         if ((err = snd_ctl_open(&card_handle, card_name, 0)) < 0) {
59             fprintf(stderr, "Erro: %s\n", snd_strerror(err));
60             continue;
61         }
62
63         // Recupera as informacoes detalhadas da placa aberta
64         if ((err = snd_ctl_card_info(card_handle, card_info)) < 0)
65         {
66             fprintf(stderr, "Erro ao ler info da placa %s: %s\n",
67                 card_name, snd_strerror(err));
68             snd_ctl_close(card_handle);
69             continue;
70         }
71         printf("\n[Placa %d]\n", card_num);
72         printf("    ID:          %s\n", snd_ctl_card_info_get_id(
73             card_info));
74         printf("    Nome:         %s\n", snd_ctl_card_info_get_name(
75             card_info));
76         printf("    Componente: %s\n",
77             snd_ctl_card_info_get_components(card_info));
```

```
73     printf("  Driver:    %s\n", snd_ctl_card_info_get_driver(  
    card_info));  
74     snd_ctl_close(card_handle);  
75 }  
76 return 0;  
77 }
```

Código 4.3: Metadados da

A documentação da biblioteca ALSA e de outras APIs para processamento de som em tempo real pode auxiliar a entender como se dá a leitura e escrita de som nas entradas e saídas de sua placa de som.

Exercícios 4.3

- | | |
|---|--|
| 7. Faça um programa que combina as bibliotecas ALSA e libsound file e que toque arquivos na saída padrão do computador. | 8. Utilizando as mesmas bibliotecas, faça um programa que leia a entrada de som (microfone) e armazene seu conteúdo em um arquivo WAV. |
|---|--|

Colocar waveform

Pre Print
V.0.0.1

Capítulo 5

Processamento de sinais de áudio

O nome “processamento de sinais de áudio” é um bom nome para este capítulo, mas é também o título de outros livros. Por esta razão, vou me contentar neste capítulo a introduzir o tema e trazer uma base de aritmética de sinais, filtros e geração de ondas de som. No entanto, o tema Filtros digitais também é um título de alguns livros e pode ser muito mais aprofundado do que veremos por aqui.

5.1 Aritmética de sinais

Uma vez que entendemos como utilizar o áudio digital no computador, é possível verificar as operações operações aritméticas em sinais que permitem alterar o valor deste sinal.

5.1.1 Soma de escalar com vetor

A primeira operação simples que podemos fazer em um sinal é somar um valor escalar ao sinal. A soma de um escalar com um vetor significa adicionar um número fixo, o escalar, a cada uma das amostras individuais do vetor de áudio. Esta soma irá modificar a posição vertical da onda sonora em relação à sua linha de zero, ou silêncio. Assim, se um sinal de áudio está codificado para variar entre -1 e 1, ao somar o escalar 1 ao mesmo ele passa a variar entre 0 e

2. O resultado desta operação é chamado de Deslocamento DC, ou DC Offset.

A representação matemática desta operação é:

$$y[n] = x[n] + c, \quad \text{para } n = 0, 1, 2, \dots, N - 1 \quad (5.1)$$

Claro que aqui precisa de uma figura

Certamente há casos onde ocorreram erros na amostragem, o sinal ficou fora de seu centro e a soma de um escalar ao sinal pode vir a resolver este problema. No entanto, em geral, somar um escalar ao sinal de áudio é considerado uma operação indesejada e até arriscada. Os altifalantes foram feitos para oscilar para frente e para trás em uma determinada faixa de energia, cujos valores deverão estar entre os valores -1 e 1. Se mudamos isso e fazemos o altifalante oscilar entre 2 e 4, por exemplo, corremos o risco de levar o equipamento à fadiga, para além do que ele foi feito, e há risco de danificá-lo por forçá-lo a se mover fora do seu eixo.

5.1.2 Multiplicação de escalar com vetor

A multiplicação de um escalar por um vetor de áudio consiste em multiplicar cada amostra deste vetor pelo valor escalar. Diferente da soma, que causa o deslocamento da onda em relação ao seu centro, a multiplicação altera a amplitude do sinal. Por esta razão, na prática, essa é a operação matemática usada para mudar o volume ou o ganho de um som.

$$y[n] = a \cdot x[n], \quad \text{para } n = 0, 1, 2, \dots, N - 1 \quad (5.2)$$

Podemos verificar que o resultado sonoro final vai depender do valor do escalar usado nesta multiplicação. Assumindo que o sinal varia entre -1 e 1:

- ♪ Se o escalar for o valor 0, a multiplicação irá gerar um sinal com o valor 0 em todas amostras, o que irá silenciar totalmente o áudio.
- ♪ Se o escalar for um valor entre 0 e 1, o sinal será atenuado e o volume será reduzido.
- ♪ Se o escalar for o valor 1, o sinal será mantido intacto, exatamente igual ao original.

- ♪ Se o escalar for um valor maior que 1, o sinal será amplificado, aumentando o volume do sinal.
- ♪ Se o escalar for um valor negativo, o resultado será similar ao apresentado até aqui mas a polaridade ou fase do sinal será invertida.

Claro que aqui precisa de uma figura

Aqui vale lembrar que, aumentar o volume do áudio para além do que esperado, como valores fora da escala entre -1 e 1, pode causar o efeito que chamamos de clipagem, ou clipping. Este efeito irá gerar distorções harmônicas no som e pode aproximar o sinal da onda quadrada.

5.1.3 Soma de vetores ou mixagem

A soma de dois vetores de áudio representa a mistura, ou mixagem, de dois sinais sonoros diferentes em um único sinal. Matematicamente, essa operação baseia-se no Princípio da Superposição Linear, onde as pressões acústicas (ou tensões elétricas/valores digitais) se combinam ponto a ponto no tempo. Isso implica em dizer que a soma de dois vetores de áudio é a soma amostra por amostra destes sinais.

$$y[n] = x_1[n] + x_2[n], \quad \text{para } n = 0, 1, 2, \dots, N - 1 \quad (5.3)$$

O resultado sonoro vai variar de acordo com o conteúdo destes sinais. No caso de os dois vetores serem exatamente iguais, terem a mesma frequência e estarem em fase, a soma será construtiva e o resultado será um vetor com a mesma frequência mas com o dobro da amplitude, ou +6dB. Neste caso, o resultado será similar ao de multiplicar o vetor pelo escalar 2 e poderá ocorrer clipping, amplificando o sinal para além do teto da amostragem.

Figura

No caso de dois vetores com a mesma frequência, mas com fase invertida, ou defasagem de 180° , a soma será destrutiva por conta do cancelamento de fase e esta soma irá resultar em silêncio absoluto. Isso porque a parte positiva da onda irá cancelar totalmente a parte negativa, retornando o valor 0 para todas as amostras. Desta maneira, o resultado será similar ao de multiplicar o sinal pelo escalar 0.

Figura

Se os sinais somados tiverem sons independentes, como uma voz e um violão, a soma das amostras irá combinar as fases de maneira aleatória ao longo do tempo, cancelando algumas frequências e amplificando outras. Esta soma não irá necessariamente dobrar a amplitude do sinal e o aumento médio será de +3dB no volume geral.

Figura

5.1.4 Multiplicação de vetores

De maneira análoga a soma de dois vetores de áudio, a multiplicação de dois vetores de áudio consiste em multiplicar as amostras correspondentes de cada sinal ponto a ponto no tempo, como na equação abaixo.

$$y[n] = x_1[n] \cdot x_2[n], \quad \text{para } n = 0, 1, 2, \dots, N-1 \quad (5.4)$$

Na prática, a multiplicação é bastante utilizada quando associada a um sinal de áudio e um sinal de controle. O sinal de controle é um sinal de áudio com propriedades específicas, como, por exemplo, variar apenas entre 0 e 1. Assim, se este sinal for uma senoide, ela irá funcionar como uma modulação de volume, entre 0 e 1, que varia no tempo, comumente chamado de LFO (Low Frequency Oscillator). Com isso, conseguimos obter um efeito comum, chamado de trêmolo.

Sinais de controle também são usados para criar rampas de volume, chamados de fade in e fade out, para que a troca entre 2 sons seja suavizada.

No caso da multiplicação de 2 sinais de áudio, o resultado de tal multiplicação irá, novamente, depender do conteúdo de cada vetor. Se ambos os sinais forem iguais, a multiplicação dos mesmos irá gerar um sinal totalmente positivo, se eles estiverem totalmente em fase, ou totalmente negativo, no caso de eles terem as fases invertidas.

A multiplicação de dois sinais é ainda utilizada para a criação de sintetizadores com modulação de amplitude e o efeito de modulação em anel. Estes temas serão apresentados mais adiante.

Exercícios 5.1

- | | |
|---|---|
| 9. Crie samples de áudio que inicie com fade id e termine com fade out. | 11. Concatene 2 arquivos de áudio de forma que os últimos 10 segundos do primeiro arquivo coincida com os 10 primeiros segundos do segundo arquivo. |
| 10. Mixe 2 arquivos, somando o conteúdo deles. | |

5.2 Filtros de áudio digital

A questão da fase na soma de um sinal com ele mesmo pode nos levar a uma situação bastante interessante. Vimos que, se somamos dois sinais iguais em fase, o resultado será um sinal com o dobro da intensidade, e se somamos dois sinais com fase invertida, o resultado será silêncio. No entanto, o que acontece se somamos 2 sinais com a defasagem de uma amostra?

Filtros FIR

$$y[n] = x[n] + x[n - 1] \quad (5.5)$$

O resultado desta soma vai depender do conteúdo deste sinal. Imaginemos que este sinal é a frequência mais alta que pode ser representada nesta taxa de amostragem. Neste caso, o sinal será algo como $x = [1, -1, 1, -1, 1, -1, \dots]$. A soma deste sinal com ele mesmo, defasado em 1 amostra, irá gerar o cancelamento total do sinal, resultando em silêncio.

E se o sinal for uma frequência muito baixa, algo muito grave, que demora muitas amostras para variar entre -1 e 1. Neste caso, o sinal será algo como $x = [1, 1, 1, 1, 1, 1, \dots]$ por muitas amostras até que ele varie. A soma deste sinal com ele mesmo, defasado em 1 amostra irá amplificar o sinal, gerando como resultado praticamente o sinal original com o dobro da amplitude.

Por esta razão, entendemos esta equação como um filtro que amplifica as frequências baixas e corta frequências agudas, também chamado de **filtro passa baixa**. Caso a amplificação das frequências baixas seja indesejável, podemos multiplicar ambos os sinais de entrada pelo escalar 0.5 de maneira a atenuar o sinal grave, mantendo-o com a mesma intensidade.

$$y[n] = 0.5 \cdot x[n] + 0.5 \cdots x[n-1] \quad (5.6)$$

Desta maneira, esta soma do sinal irá cancelar totalmente as frequências altas, deixando passar apenas as frequências baixas. Por conta de seus coeficientes, este filtro também é chamado de filtro das médias, por ser uma média aritmética entre uma amostra e a amostra anterior.

E o que acontece se, em vez de somarmos, subtrairmos um sinal dele mesmo com uma amostra de atraso?

$$y[n] = 0.5 \cdot x[n] - 0.5 \cdots x[n-1] \quad (5.7)$$

Novamente, o resultado da subtração vai depender do conteúdo do sinal. Se este sinal é a frequência mais alta que pode ser representada nesta taxa de amostragem, algo como $x = [1, -1, 1, -1, 1, -1, \dots]$, a subtração irá gerar na saída um sinal praticamente idêntico ao sinal de entrada. Se o sinal for uma frequência muito baixa, algo como $x = [1, 1, 1, 1, 1, 1, \dots]$, a subtração irá cancelar totalmente o sinal, resultando em silêncio.

Por esta razão, chamamos este filtro de **filtro passa alta**, pois ele deixa passar apenas as frequências altas e elimina totalmente frequências baixas.

Para pensar no filtro de forma genérica, podemos assumir que o mesmo terá 2 coeficientes, b_0 e b_1 , que servem para determinar o comportamento deste filtro. Se ambos forem positivos ou negativos, este será um filtro passa alta, se um for positivo e o outro negativo, este será um filtro passa baixa. O quanto queremos filtrar vai depender da escolha destes coeficientes sendo que, para evitar que a saída tenha uma intensidade maior que as entradas, é ideal que $|b_0| + |b_1| = 1$.

Filtros IIR

Tanto o filtro de passa baixa quanto o filtro de passa alta aqui apresentados são chamados de filtros FIR (Finite Impulse Response - Resposta ao Impulso Finita) por tratar apenas de dados de entrada. Existem também a possibilidade de somar saída atrasada ao sinal de entrada, como apresentado na equação abaixo:

$$y[n] = b_0 \cdot x[n] + a_1 \cdot y[n-1] \text{ onde } a_1 = 1 - b_0 \quad (5.8)$$

$x[n]$	$y[n-1]$	$y[n]$
1	0	0.5
-1	0.5	-0.25
1	-0.25	0.375
-1	0.375	-0.3125
1	-0.3125	0.34375
-1	0.34375	-0.328125
1	-0.328125	0.3359375
-1	0.3359375	-0.33203125
1	-0.33203125	0.333984375
-1	0.333984375	-0.3330078125
1	-0.3330078125	0.33349609375
-1	0.33349609375	-0.333251953125

Tabela 5.1: Cálculo do filtro passa baixa IIR com uma entrada com frequência alta

Estes filtros, chamados filtros IIR, são realimentados, ou recursivos, e por isso podem ser mais instáveis. No entanto, seu custo computacional é mais baixo, e por isso eles são mais adequados para aplicações em tempo real.

Vamos pegar o exemplo a seguir:

$$y[n] = 0.5 \cdot x[n] + 0.5 \cdot y[n-1] \quad (5.9)$$

Sabendo que a saída do filtro irá depender da entrada, vamos analisar o que acontece em nosso filtro FIR se temos na entrada um sinal de frequência alta, como $x = [1, -1, 1, -1, 1, -1, \dots]$. Desta vez, devido a recursividade do sinal, teremos que calcular a saída ponto a ponto. Assim, como $y[-1]$ é indefinido, e assumiremos que o mesmo vale 0.

Adotaremos novamente a frequência grave como sendo uma entrada que tem pouca variação, como $x = [1, 1, 1, 1, 1, 1, \dots]$. O que ele faria com frequências graves?

Logo, tal filtro é um filtro passa baixa.

5.2.1 A teoria dos Filtros

Estas somas com o sinal atrasado permitem filtrar sinais e isso significa escolher o que queremos deixar passar e o que queremos reter em nosso filtro. Na

$x[n]$	$y[n-1]$	$y[n]$
1	0	0.5
1	0.5	0.75
1	0.75	0.875
1	0.875	0.9375
1	0.9375	0.96875
1	0.96875	0.984375
1	0.984375	0.9921875
1	0.9921875	0.99609375
1	0.99609375	0.998046875
1	0.998046875	0.9990234375
1	0.9990234375	0.99951171875
1	0.99951171875	0.999755859375

Tabela 5.2: Cálculo do filtro passa baixa IIR com uma entrada com frequência baixa

prática, filtros são utilizados em equalizadores e permitem tanto cancelar quanto amplificar apenas determinadas frequências ou faixas de frequência do sinal.

Os filtros FIR possuem muito cancelamento e pouco acoplamento, enquanto os filtros IIR geram pouco cancelamento e muito acoplamento. A equação canônica mistura os dois e permite amplificar ou cortar frequências, dependendo da escolha dos coeficientes.

$$y[n] = b_0 \cdot x[n] + b_1 \cdot x[n-1] - a_1 \cdot y[n-1] \quad (5.10)$$

Calculando os coeficientes dos filtros

Até este momento, vimos como os filtros funcionam mas não citamos em nenhum momento qual a taxa de amostragem do áudio aqui representado. Isso porque, até agora vimos que os filtros podem filtrar frequências graves, representadas por longo trechos com o mesmo valor, ou frequências altas, representadas por uma grande alternância de valores.

Para calcular os coeficientes de um filtro, precisamos levar em consideração a taxa de amostragem, além da frequência de corte, aqui chamada de f_c . A proporção da frequência de corte (f_c) e a taxa de amostragem (f_s) irá nos gerar uma constante K , onde:

$$K = \tan(\pi \cdot f_c / f_s) \quad (5.11)$$

Com esta constante calculada, podemos calcular os coeficientes usando a seguinte tabela:

	b_0	b_1	a_1
Lowpass	$K/(K+1)$	$K/(K+1)$	$(K-1)/(K+1)$
Highpass	$1/(K+1)$	$-1/(K+1)$	$(K-1)/(K+1)$
All pass	$(K-1)/(K+1)$	1	$(K-1)/(K+1)$

Tabela 5.3: Cálculo dos coeficientes de filtros de primeira ordem

Filtros de segunda ou mais ordens

Por utilizar o sinal original e o mesmo atrasado em 1 amostra, estes filtros são chamados de filtros de primeira ordem. Há ainda filtros IIR e FIR de segunda ordem, que usam 2, 3 ou mais amostras atrasadas.

$$y[n] = b_0 \cdot x[n] + b_1 \cdot x[n-1] + b_2 \cdot x[n-2] \quad (5.12)$$

$$y[n] = b_0 \cdot x[n] + b_1 \cdot x[n-1] + b_2 \cdot x[n-2] - a_1 \cdot y[n-1] - a_2 \cdot y[n-2] \quad (5.13)$$

Os filtros de segunda ordem atual atuam de forma similar a combinação de dois filtros de primeira ordem. Assim, se combinamos 2 filtros passa baixa de primeira ordem, teremos um filtro passa baixa cujo corte é mais íngreme. Este corte é chamado de slope. O mesmo acontece se combinamos 2 filtros passa alta.

Se combinamos um filtro passa alta e um filtro passa baixa, podemos ter um filtro passa banda, ou um filtro de rejeita banda, cujo comportamento permite filtrar ou amplificar uma faixa de frequências.

Mais sobre filtros em geral pode ser encontrado em materiais específicos.

5.2.2 Calculando os coeficientes de filtros

Sabemos que os filtros apresentados irão deixar passar frequências altas ou frequências baixas, mas não sabemos qual será a frequência de corte destes filtros. Frequência de Corte, ou f_c , é a frequência a partir da qual as frequências serão atenuadas em um filtro. Assim, como fazemos para definir o ponto ao qual um determinado filtro irá começar a agir?

Tomando a equação canônica de um filtro de primeira ordem, sabemos que os coeficientes b_0 , b_1 e a_0 irão modificar o funcionamento do filtro, tornando-o passa alta ou passa baixa.

Além disso, a escolha de tais coeficientes irão nos dizer qual será a frequência

$$y[n] = b_0 \cdot x[n] + b_1 \cdot x[n - 1] - a_1 \cdot y[n - 1]$$

(5.14)

Exercícios 5.2

- | | |
|---|------------|
| 12. Crie um filtro de passa baixa que funciona em tempo real e que retira simbilâncias da fala (letras s). Utilize a entrada de | microfone. |
| 13. Crie um filtro passa baixa capaz de retirar os sons graves de um arquivo de áudio. | |

5.3 Extraíndo propriedades do sinal de áudio

5.3.1 Duração

A duração de um sinal de áudio dependerá do tamanho do vetor de amostras e da taxa de amostragem do mesmo. Se um sinal tem taxa de amostragem de 48.000 amostras por segundo, cada amosta irá ter 1/48.000 segundos, ou 0.02ms.

5.3.2 Amplitude

5.3.3 Frequência

Banco de filtros

Exercícios 5.3

- | | |
|---|------------|
| 14. Crie um filtro de passa baixa que funciona em tempo real e que retira simbilâncias da fala (letras s). Utilize a entrada de | microfone. |
| 15. Crie um filtro passa baixa capaz de retirar os sons graves de um arquivo de áudio. | |

5.4 Áudio como controle

5.5 Gerando sinais de áudio

5.5.1 Ruído

O ruído é considerado um sinal indesejado e há diversos ruídos conhecidos no áudio como o ruído branco, ruído rosa, ruído cinza, ou outros. O ruído branco é um sinal muito importante pois ele possui todas as frequências representadas no áudio com a mesma energia ou intensidade. Por isso, ele pode ser usado em diversas medições, como a medição da relação sinal-ruído, onde queremos saber qual frequência está ou presente em um determinado experimento. Outros ruídos possuem distribuições diferentes de frequências e energias. O ruído rosa, por exemplo, possui energia balanceada por faixa de frequência baseada na percepção humana.

A geração de ruído é feita gerando valores randômicos para cada amostra do sinal de áudio, conforme apresentado no Código 5.1. O valor de cada amostra é gerado aleatoriamente pela função `rand()` e ajustado para ficar entre -1 e 1. Este código utiliza a estrutura anterior e apenas modifica o gerador de sinal para a geração de ruído.

```
78 #include <stdlib.h>
79 #include <time.h>
80
81 int gerar_sinal(float * samples){
82     srand(time(NULL));
83     float sample = 0.0f;
84     for(int i = 0 ; i < SAMPLE_RATE * 3; i++){
85         sample = (float)rand() / (float)RAND_MAX;
86         samples[i] = sample * 2.0f - 1.0f;
```

Código 5.1: Geração de ruído branco para arquivo

5.5.2 Senos e cossenos

As ondas seno e cosseno possuem oscilações repetitivas suaves e, graças a estas repetições, possuem uma frequência definida. Por isso, diferentemente do silêncio e do ruído, a geração deste sinal irá depender da taxa de amostragem do áudio pois há a limitação da taxa de Shannon-Nyquist que diz que a maior frequência representada é a metade da taxa de amostragem (SR).

Antes de começar a geração de uma onda, sugiro um exercício de intuição, gerando o cosseno de 1Hz com uma taxa de amostragem de 48kHz. Para a geração de 1 segundo de áudio iremos gerar 48.000 amostras sendo que cada amostra poderá ter um valor de cosseno de um ângulo diferente no círculo trigonométrico. Como nossa taxa de amostragem é de 48kHz, iremos dividir o círculo trigonométrico em 48.000 partes iguais, e chamaremos este ângulo de ω . Teremos então um primeiro ângulo ω , que vai gerar o valor da primeira amostra como $\cos(\omega)$. Depois, teremos um segundo ângulo 2ω , cujo valor do $\cos(2 * \omega)$ irá gerar o valor da segunda amostra e assim por diante até completar a volta no círculo trigonométrico, gerando as 48.000 amostras, onde a amostra t será gerada pelo valor de $\cos(t * \omega)$.

Ainda continuando nosso exercício de intuição, é possível não iniciar a geração de amostras na posição zero, mas assumir que poderíamos ter uma fase inicial ϕ que pode ser o deslocamento para a geração da primeira amostra e, conseqüentemente, de todas as demais. Assim, se queremos começar nosso oscilador com o valor 1, podemos ter uma fase inicial de 90 graus, ou $\pi/2$, se queremos começar com -1, podemos ter uma fase inicial de 270 graus, ou $3\pi/4$.

Além disso, podemos não querer que as amostras fiquem variando entre 1 e -1, já que isso significa a amplitude máxima do sinal de áudio para esta representação. Podemos querer que todas as amostras sejam amplificadas para um valor acima disso, ou terem seu volume reduzido para um valor abaixo disso. Então, é possível assumir que teríamos um fator escalar A que serve para controlar a amplitude deste sinal.

Com isso, a geração de amostras em uma senoide pode ser representada por meio da seguinte equação:

$$x[t] = A \cos(\omega t + \phi) \quad (5.15)$$

Onde:

- ♪ A é a amplitude da senóide
- ♪ ω é a velocidade angular da frequência a ser gerada
- ♪ t é o número sequencial da amostra
- ♪ ϕ é a fase inicial

No entanto, esta equação não nos diz nada sobre a taxa de amostragem pois ela está subentendida na velocidade angular ω . Podemos entender a taxa de amostragem serviu para fracionar o círculo trigonométrico. Para entender como a taxa de amostragem está nesta equação, vamos continuar com nosso exercício de intuição.

Como seria gerar uma frequência mais alta, como 2Hz? Neste caso, precisaríamos dar 2 voltas no círculo trigonométrico em 1 segundo, o que nos leva a andar com um passo maior, implicando em uma velocidade angular ω duas vezes maior. E como seria para gerar uma senoide de 440hz? Daremos 440 voltas no círculo em 1 segundo aumentando o passo, ou a velocidade angular, em 10 x. Com isso, podemos entender como gerar qualquer frequência se assumirmos que a velocidade angular pode ser calcular a partir da frequência que queremos representar (f_0) e a taxa de amostragem (f_s). Isso pode ser representado pela expressão:

$$x[t] = A \sin\left(2\pi \frac{f_0}{f_s} t + \phi\right) \quad (5.16)$$

Onde:

- ♪ A é a amplitude da senóide
- ♪ f_0 é a frequência representada
- ♪ f_s é a taxa de amostragem
- ♪ t é o número sequencial da amostra

♪ ϕ é a fase inicial

Assim, se a frequência de 1Hz tem velocidade angular $\omega = 2\pi * 1/f_s$, a velocidade angular de 2Hz será um ângulo maior e terá $\omega = 2\pi * 2/f_s$, enquanto a frequência de 440 hertz terá uma velocidade angular ainda maior, $\omega = 2\pi * 440/f_s$.

Revendo o teorema de Shannon-Nyquist

E o que acontece quando o valor da frequência aumenta a ponto de a velocidade angular ser igual a 360° ou 2π ? Neste caso, todas as amostras terão o mesmo valor, $\cos(2\pi) = 1$ e o sinal será uma constante (sinal DC). Assim, podemos verificar que um sinal com uma taxa de amostragem SR não é capaz de representar a frequência SR.

Também podemos verificar o que acontece quando o valor da frequência for maior do que 180° ou π , algo como $\pi + x$. Como a função cosseno é “espelhada” no eixo Y, o valor do cosseno do ângulo de $\pi + x$ é o mesmo valor que o cosseno de $\pi - x$.

Colocar uma figura aqui

Assim, quando a frequência é maior do que π , ou $\pi + x$ para algum valor positivo de x , ela passa a ser representada por uma frequência menor do que ela, ou, exatamente, $\pi - x$, até que este valor seja menor ou igual a zero, ou $0 - x$, ou ainda $2\pi - x$. Neste ponto, a frequência volta a crescer e representar um valor maior, ou $0 + x$, ou ainda $2\pi + x$. Neste ponto, a velocidade angular é maior do que o próprio círculo, ou maior de π , e podemos dizer que $x > \pi$.

Como assumimos inicialmente que 1 volta no círculo significa a taxa de amostragem, ou $SR = 2\pi$, entendemos que meia volta no círculo é metade desta taxa, ou $\pi = SR/2$, e que a maior frequência representável em um sinal de áudio digital é a metade da taxa de amostragem.

Este exercício permite nos permite entender a razão do Teorema de Shannon-Nyquist e o porquê de uma taxa de amostragem limitar a frequência máxima representada. Podemos ainda calcular qual será a frequência representada no caso de tentarmos representar uma frequência maior do que metade da taxa de amostragem, também chamada de frequência de rebatimento ou aliasing. Sinais com frequência (f) acima de Nyquist (f_n) e taxa de amostragem f_s irão ser amostradas para a frequência rebatida (f_r) de acordo com a expressão:

$$f_r = |((f + f_n) \% f_s) - f_n| \quad (5.17)$$

Gerando senóides na prática

Tendo entendido como gerar sinusóides, como senos e cossenos, podemos partir para um exemplo de código onde podemos ver isso na prática. O código 5.2 apresenta a geração de uma senóide se 1Hz.

```
88 (...)
89 #include <math.h>
90 #define SAMPLE_RATE 48000.0
91 #define PI 3.141592653
92 #define FREQ 1.0
93
94 // gcc soundfile1.c -o soundfile -lsndfile -lm
95 int gerar_sinal(float * samples){
96     for(int i = 0 ; i < SAMPLE_RATE * 3; i++){
97         samples[i] = cos(i * 2 * PI * FREQ / SAMPLE_RATE);
98     }
99 (...)

```

Código 5.2: Geração de senoide de 1Hz para arquivo

Apesar de esta frequência ser inaudível, é possível abrir este arquivo em um software como o Audacity para verificar a onda gerada.

audacity]

Exercícios 5.5

- | | |
|--|---|
| 16. Calcule a frequência de rebatimento para um sinal com taxa de amostragem em 44.100 Hz e frequências de 200Hz, 25 kHz, 50 kHz e -230 kHz. | 17. Faça um software que recebe como entrada o tempo e a frequência desejada e que gere arquivos de áudio com a entrada especificada. |
|--|---|

5.5.3 Onda quadrada

Outra onda comumente utilizada em sinais elétricos e em contextos musicais é chamada de onda quadrada. Similar a senoide, esta onda as ondas quadradas possuem repetições periódicas mas, diferentemente das senoides, não são suaves

pois possuem uma mudança abrupta entre a sua fase positiva e sua fase negativa. A onda quadrada é caracterizada por assumir o maior valor representado em toda fase positiva da onda e o menor valor para toda a fase negativa, como, por exemplo, os valores 1 e -1 .

Assim, para calcular uma onda quadrada assumimos que metade dela, ou metade do seu período (de 0 até $T/2$), terá o valor máximo A , e a outra metade (de $T/2$ até T), terá o valor $-A$. Tal intuição pode ser representada pelo sistema de equações abaixo:

$$x[t] = \begin{cases} A, & \text{se } 0 \leq t < \frac{T}{2} \\ -A, & \text{se } \frac{T}{2} \leq t < T \end{cases} \quad (5.18)$$

Aplicando um pouco de intuição, se queremos uma frequência de 1Hz, amplitude máxima (-1 a 1), e taxa de amostragem 48kHz, para gerar 1 segundo de áudio temos que as amostras de 0 a 24.000 terão o valor 1, e as amostras de 24.000 até 48.000 terão o valor -1 .

E como ficaria a frequência a frequência de 2Hz? Ela tem metade do período, logo, as amostras de 0 a 12.000 terão valor 1, as amostras de 12.000 a 24.000 terão o valor -1 , as amostras de 24.000 a 36.000 terão o valor 1, e as amostras de 36.000 a 48.000 terão valor -1 .

Se tomamos a equação acima, precisamos saber qual será o período da onda e, para isso, precisamos saber quanto tempo tem cada amostra t . Este tempo é proporcional a taxa de amostragem f_s . Assim, podemos entender que o período de uma onda de 1Hz é de 1 segundo, ou 48.000 amostras, enquanto o período de uma onda de 2Hz é de 0.5 segundos, ou 24.000 amostras. Com isso, podemos entender que o período de uma frequência f_0 será de $T = f_s/f_0$. A implementação desta função pode ser vista no Código 5.4.

```

100 int gerar_sinal(float * samples, int size){
101     int range = 0;
102     int T = (SAMPLE_RATE / freq);
103     for(int i = 0 ; i < size; i++){
104         range = i % (T);
105         samples[i] = (range >= 0 && range < (T / 2.0)) ? 1.0 : -1.0;
106     }
107 }
```

Código 5.3: Geração de onda quadrada para arquivo

Outra forma de gerar esta onda é utilizar a potência de um número negativo, como $(-1)^n$. Quando a potência n é um número par, o resultado é 1. Quando a potência n é ímpar, o resultado é -1. Desta maneira, para gerar uma frequência qualquer com período T precisamos de uma função na qual todos os valores entre 0 e $T/2$ seja par, e todos os valores entre $T/2$ e T sejam ímpares.

Isso pode ser feito pela função chão, onde f representa a frequência.

$$(x(t) = (-1)^{\lfloor 2ft/f_s \rfloor} \quad (5.19)$$

A implementação desta função pode ser vista no Código ??.

```

108 int gerar_sinal(float * samples, int size){
109     for(int i = 0 ; i < size; i++){
110         samples[i] = pow(-1, floor(i * 2.0 * FREQ/ SAMPLE_RATE ));
111     }
112 }
```

Código 5.4: Geração de onda quadrada para arquivo

Também podemos pensar que a onda quadrada pode ser feita a partir do sinal do seno da frequência, como apresentada na equação abaixo. Isso pode ser vantajoso, quando já temos a senóide calculada mas pode ser mais caro computacionalmente por precisar calcular todos os valores da senoide. Tal representação pode ser vista na equação abaixo, e a implementação dela deixamos como exercício para o leitor.

$$x(t) = \text{sign}(\sin(\omega t)) \quad (5.20)$$

Há ainda uma maneira de calcular ondas quadradas pela soma de senóides, mas isso iremos apresentar mais adiante.

5.5.4 Onda Dente de serra

A onda dente de serra é outra onda periódica clássica da área de sinal e também da computação musical. Esta onda parte de um valor inicial, como 0, sobe linearmente até um valor de pico, como 1, e depois cai até o valor de vale, como -1. Esta queda abrupta lembra a mudança da onda quadrada, e também seu som encorpado, mas como a subida é linear, o som acaba sendo menos áspero

que o da onda quadrada.

A forma mais simples e direta de criar onda dente de serra com período $T = 1$ é descrita pela seguinte fórmula:

$$x(t) = t - \lfloor t \rfloor \quad (5.21)$$

Onde t é o tempo, e $\lfloor t \rfloor$ é a função “pisso” (maior inteiro menor ou igual a t). Essa equação gera uma rampa linear que sobe de 0 a 1 e retorna para 0 imediatamente, repetindo o ciclo. Se quisermos que esta onda varie entre -1 e 1, podemos ajustar estes valores, multiplicando este sinal por 2 (mudando a variação para valores entre 0 e 2) e depois subtraindo 1. Podemos ainda adicionar a frequência desejada (f_0) e a taxa de amostragem (f_s).

$$f(t) = 2 \left(\frac{t * f_0 / f_s}{2\pi} - \left\lfloor \frac{t * f_0 / f_s}{2\pi} \right\rfloor \right) - 1 \quad (5.22)$$

A implementação desta onda dente de serra pode ser vista no Código 5.5

```

113 int gerar_sinal(float * samples, int size){
114     for(int i = 0 ; i < size; i++){
115         float z = i * FREQ / SAMPLE_RATE;
116         samples[i] = 2.0 * (z - floor(z)) - 1.0;
117     }
118 }
```

Código 5.5: Geração de onda triangular para arquivo

5.5.5 Onda Triangular

A onda triangular é também uma onda periódica não senoidal que apresenta um formato geométrico semelhante ao de um triângulo. Esta onda é composta por rampas lineares alternadas, de subida e de descida, e possui uma sonoridade intermediária entre uma onda senoidal e uma onda quadrada.

Uma maneira de gerar computacionalmente uma onda triangular é utilizar a função arco-seno. A função arco-seno é o inverso do seno e aceita como entrada qualquer valor válido de seno, entre -1 e 1, e tem como saída ângulos entre $[-\frac{\pi}{2}, \frac{\pi}{2}]$. Assim, para ajustar sua saída em uma amplitude entre A e $-A$, podemos multiplicar o valor de saída desta função por $\frac{\pi}{2}$, já que $\pm \frac{\pi}{2} \cdot \frac{2A}{\pi} = \pm A$. A entrada desta função pode ser a velocidade angular $\omega == \frac{2\pi}{T}$, onde T é o período da onda em segundos. A representação matemática da onda triangular:

$$x(t) = \frac{2A}{\pi} \arcsin \left(\sin \left(\frac{2\pi t}{T} \right) \right) \quad (5.23)$$

Para ajustar este período para samples, novamente iremos utilizar a taxa de amostragem, dividindo a taxa de amostragem pela frequência desejada. O resultado pode ser apresentado no Código 5.6.

```

119 int gerar_sinal(float * samples, int size){
120     for(int i = 0 ; i < size; i++){
121         float z = i * FREQ / SAMPLE_RATE;
122         samples[i] = 1.0 - 2.0 * fabs(2.0 * (z - floor(z + 0.5)));
123     }
124 }
```

Código 5.6: Geração de onda triangular para arquivo

Outra maneira de gerar ondas triangulares é utilizar a função módulo e a função chão, partindo de uma onda dente de serra. Utilizamos a função geradora da onda dente de serra e aplicamos o módulo a mesma. Com isso, “invertemos” a parte negativa da onda dente de serra e conseguimos uma onda triangular. No entanto, esta onda triangular estará compreendida entre os valores 0 e 1. Para ajustar a mesma entre 1 e -1 basta multiplicar sua saída por 2 e subtrair 1, como apresentado na equação abaixo.

$$f(t) = 1 - \left| 2 \left(\frac{t}{2\pi} - \left\lfloor \frac{t}{2\pi} + 0.5 \right\rfloor \right) \right| \quad (5.24)$$

A implementação desta técnica pode ser vista no Código 5.7.

```

125 int gerar_sinal(float * samples, int size){
126     for(int i = 0 ; i < size; i++){
127         float z = i / (1.0 * SAMPLE_RATE / FREQ);
128         samples[i] = 1.0 - 2.0 * fabs(2.0 * (z - floor(z + 0.5)));
129     }
130 }
```

Código 5.7: Geração de onda triangular a partir da onda dente de serra

5.5.6 Oscilador por consulta a tabela

A criação destas ondas periódicas usando as fórmulas apresentadas anteriormente é bastante simples, mas pode trazer algumas desvantagens computaci-

onais. A primeira delas é a possibilidade de causar um estouro na variável n em laços muito grandes. Para gerar minutos ou horas de áudio teríamos que calcular muitos valores e a variável deste laço de repetição pode não ser grande o suficiente para isso, por maior que a mesma seja. Uma segunda desvantagem é o custo computacional para calcular pisos, cossenos, arcos cossenos, tetos e afins. Este custo pode ser proibitivo para que tais algoritmos sejam implementados em dispositivos com capacidade computacional reduzida, como microcontroladores. Além disso, cada forma de onda apresentada possui uma ou mais fórmulas para a sua geração mas não é intuitivo gerar outras formas de onda com tais representações e sim, é possível gerar ondas periódicas com qualquer formato.

Uma solução comumente utilizada em computação musical é a utilização de osciladores por consulta a tabela. Neste solução, uma tabela auxiliar irá ser utilizada para armazenar uma onda de referência, como uma senoide, ou qualquer outro formato. Depois, esta tabela será consultada para calcular todos os valores da onda desejada. Com isso a) o custo computacional com o cálculo da senoide só vai existir na criação da tabela, b) não vai haver estouro de variável de laço, e c) esta tabela poderá ser utilizada para armazenar outras formas de onda para além das ondas clássicas (senóide, quadrada, triangular ou dente de serra).

Vamos a um entendimento intuitivo. Queremos uma senoide de 1 minuto com a frequência de 1Hz e SR de 48kHz. Isso iria demandar calcular $60 * 48.000$ valores com a função seno. Ao invés disso, iremos criar uma tabela com 48.000 amostras e simplesmente copiar o valor desta tabela para o vetor de saída 60 vezes, como se a mesma fosse circular. Toda vez que chegamos ao fim, retornamos ao começo. A criação da tabela irá utilizar as fórmulas matemáticas apresentadas anteriormente e só será necessário calcular os pontos deste áudio uma vez. Com isso, reduzimos drasticamente (60 vezes menos) o custo computacional e podemos gerar horas de senoides sem precisar recalcular os valores da função seno.

Agora, será que conseguimos utilizar esta mesma tabela para gerar uma senoide de 2Hz? A resposta é sim. Basta dobrar o passo da leitura que será possível calcular esta frequência. Com isso o valor de saída $x[t] = tabela[t * 2]$ e novamente faremos a leitura da tabela como se a mesma fosse circular, retornando ao começo toda vez que o tamanho da mesma for ultrapassado. Se queremos a frequência de 4Hz, basta consultar o índice $tabela[t*4]$, e se queremos

um valor de frequência f_0 qualquer, iremos consultar os índices $tabela[t * f_0]$. Será que podemos utilizar esta tabela para calcular qualquer frequência?

A próxima pergunta é, será que precisamos de uma tabela com 48.000 amostras? A resposta é não. Uma tabela com 64, 128, 256 ou 1024 amostras pode ser o suficiente. Só precisamos, para isso, ajustar a busca por amostras nesta tabela para a) garantir que a mesma seja circular, e b) incluir o tamanho da tabela no cálculo dos seus índices. Tal cálculo pode ser apresentado na equação a seguir.

$$x[i] = tabela[mod((i * tamanho * f_0 / f_s), tamanho)] \quad (5.25)$$

Note que este cálculo irá arredondar (ou truncar) os valores fracionários, buscando os valores inteiros mais próximos. Isso pode ser feito de forma explícita, por meio de funções como arredondamento, chão e teto. Apesar de ser computacionalmente muito barato, este arredondamento pode adicionar distorções indesejáveis ao sinal de saída. Uma solução mais eficiente para isso é utilizar algumas técnicas de interpolação.

A interpolação é uma técnica utilizada para estimar valores desconhecidos em um intervalo de dados conhecidos. Os métodos de interpolar usam dados existentes para construir linhas ou curvas contínuas que preenche as lacunas de forma lógica. Há diferentes técnicas de interpolação, como a interpolação linear, interpolação quadrática e interpolação cúbica.

Interpolação linear

A técnica de interpolação linear consiste em ajustar uma equação de primeiro grau ($y = ax + b$) para encontrar um valor y correspondente a um x desejado, utilizando dois pontos conhecidos: (x_0, y_0) e (x_1, y_1) sendo que o valor x está entre os dois valores, $x_0 < x < x_1$.

A fórmula da interpolação linear é expressa como:

$$\frac{y - y_0}{x - x_0} = \frac{y_1 - y_0}{x_1 - x_0} \quad (5.26)$$

Isolando o y , chegamos à equação prática:

$$y = y_0 + \frac{(x - x_0) \cdot (y_1 - y_0)}{x_1 - x_0} \quad (5.27)$$

Para exemplificar, imagine que temos um array que na posição 1 tem o valor 8 e na posição 2 tem o valor 15. Imagine agora que precisamos do valor na posição 1.4. Para calcular, assumiremos que:

$$\begin{aligned} x_0 &= 1, y_0 = 8, (x[1] = 8) \\ x_1 &= 2, y_1 = 15, (x[2] = 15) \\ x &= 1.4, y = ?, \\ y &= 8 + \frac{(1.4 - 1) \cdot (15 - 8)}{2 - 1} \\ y &= 8 + \frac{(0.4) \cdot (7)}{1} \\ y &= 8 + \frac{2.8}{1} \\ y &= 10.8 \end{aligned} \quad (5.28)$$

A interpolação linear é computacionalmente barata e simples de implementar, mas irá deixar rampas lineares no sinal de saída. Isso pode ser eficiente para ondas quadradas, triangulares e dentes de serra mas poderá gerar distorções harmônicas em ondas sinusoidais. Para ondas curvas, uma solução melhor é utilizar a interpolação quadrática.

Interpolação quadrática

A interpolação quadrática traz a possibilidade de usarmos uma curva em formato de parábola em vez de uma reta, para calcular valores dentro de um determinado intervalo. Esta parábola passa por estes três pontos e irá prever o ponto x usando a curvatura básica dos dados fornecidos se baseando em uma equação de segundo grau $y = ax^2 + bx + c$). A interpolação quadrática irá utilizar os pontos (x_0, y_0) , (x_1, y_1) e (x_2, y_2) para calcular um ponto intermediário (x, y) , sendo que o ponto $x_0 < x < x_1 < x_2$.

Existem algumas maneiras de calcular a interpolação quadrática e uma ma-

neira simples de calcular esta interpolação é utilizar a fórmula de Lagrange:

$$y = y_0 \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} + y_1 \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} + y_2 \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} \quad (5.29)$$

Para exemplificar, imagine que temos o mesmo array que na posição 1 tem o valor 8 e na posição 2 tem o valor 15 mas agora adicionamos na posição 3 o valor -1. Imagine agora que precisamos novamente do valor na posição 1.4. Para calcular, assumiremos que:

$$x_0 = 1, y_0 = 8, (x[1] = 8)$$

$$x_1 = 2, y_1 = 15, (x[2] = 15)$$

$$x_2 = 3, y_2 = -1, (x[3] = -1)$$

$$x = 1.4, y = ?,$$

$$y = 8 \frac{(1.4 - 2)(1.4 - 3)}{(1 - 2)(1 - 3)} + 15 \frac{(1.4 - 1)(1.4 - 3)}{(2 - 1)(2 - 3)} + -1 \frac{(1.4 - 1)(1.4 - 2)}{(3 - 1)(3 - 2)} \quad (5.30)$$

$$y = 8 \frac{(-0.6)(-1.6)}{(-1)(-2)} + 15 \frac{(0.4)(-1.6)}{(1)(-1)} + -1 \frac{(0.4)(-0.6)}{(2)(1)}$$

$$y = 8 \frac{0.96}{2} + 15 \frac{-0.64}{-1} + -1 \frac{-0.24}{2}$$

$$y = 8 * 0.48 + 15 * 0.64 + -1 * -0.12$$

$$y = 3.84 + 9.60 + 0.12$$

$$y = 13.56$$

Certamente, por envolver mais contas, o custo computacional da interpolação quadrática é maior que o custo computacional da interpolação linear.

Interpolação cúbica

Continuando com as interpolações, a próxima possibilidade é utilizar um polinômio de grau 3, da forma $f(x) = ax^3 + bx^2 + cx + d$. Para isso, esta interpolação irá utilizar 4 pontos para definir a curva e definir o ponto intermediário cujo valor é desconhecido. Assim, partimos dos pontos $(x_0, y_0), (x_1, y_1), (x_2, y_2), (x_3, y_3)$

para identificar o valor da curva no ponto (x, y) sendo que $x_0 < x_1 < x < x_2 < x_3$.

Temos mais de uma maneira de calcular este polinômio sendo um método comum a fórmula de Lagrange. De maneira similar à interpolação quadrática, esta fórmula parte da equação:

$$\begin{aligned}
 P_3(x) &= y_0 L_0(x) + y_1 L_1(x) + y_2 L_2(x) + y_3 L_3(x) \\
 L_0(x) &= \frac{(x - x_1)(x - x_2)(x - x_3)}{(x_0 - x_1)(x_0 - x_2)(x_0 - x_3)} \\
 L_1(x) &= \frac{(x - x_0)(x - x_2)(x - x_3)}{(x_1 - x_0)(x_1 - x_2)(x_1 - x_3)} \\
 L_2(x) &= \frac{(x - x_0)(x - x_1)(x - x_3)}{(x_2 - x_0)(x_2 - x_1)(x_2 - x_3)} \\
 L_3(x) &= \frac{(x - x_0)(x - x_1)(x - x_2)}{(x_3 - x_0)(x_3 - x_1)(x_3 - x_2)}
 \end{aligned} \tag{5.31}$$

Esta interpolação irá ser ainda mais suave mas adiciona ainda mais custo computacional para calcular um determinado ponto.

Exercícios 5.5

18. Criar composições concatenando arquivos

19. Gerar arquivos com ruído e silêncio e frequências.

Capítulo 6

Algoritmos de síntese sonora

- Timbre e características timbrísticas - Espectrograma - Visualização de envelopes - Envelopes

6.1 Temperando ondas periódicas

Nem todo som possui uma frequência definida pois nem todo som possui um padrão de vibração ou padrão de repetição da vibração. Estes sons, que não possuem frequência de vibração definida, podem ser chamados de ruídos e são comuns entre os instrumentos de percussão, por exemplo. Não iremos nos aprofundar por aqui na questão do ruído e iremos focar, neste momento, nos sons que possuem vibração definida, ou altura musical.

Uma vez que entendemos as notas musicais, podemos voltar a pensar sobre frequências e alturas. Ao esticar uma corda e tocá-la, teremos uma frequência de vibração, por exemplo, de 100 vibrações por segundo, ou 100 Hz. Esta frequência de vibração é relacionada ao tamanho da corda e a tensão da mesma. No entanto, a física nos mostra que um corpo vibrante não vibra em apenas uma frequência, ele dispara uma série de frequências que chamamos de **série harmônica**, e que isso acontece com diversos materiais vibrantes como cordas, placas, sinos, peles, entre outros.

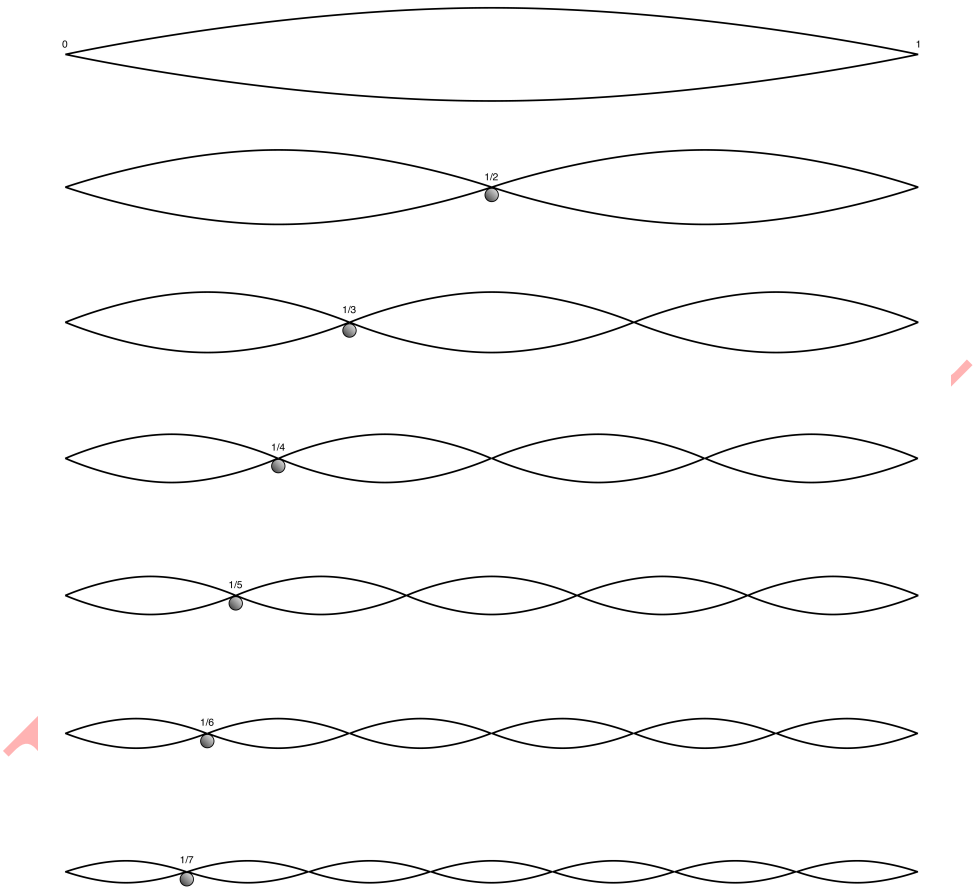


Figura 6.1: Exemplo de vibrações que ocorrem com cordas

#	Frequência
F1	100Hz
F2	200Hz
F3	300Hz
F4	400Hz
F5	500Hz
F6	600Hz
F7	700Hz
F8	800Hz
F9	900Hz
...	...

Tabela 6.1: Série harmônica da frequência de 100 Hz

6.1.1 A Série Harmônica

A série harmônica é um conhecimento antigo da humanidade e está relacionada aos estudos do filósofo e matemático grego **Pitágoras**. Os estudos de Pitágoras nos mostram que, uma corda, ao ser vibrada dispara uma primeira frequência, que chamamos de fundamental, e também diversas outras frequências harmônicas que são relativas a esta fundamental, chamadas de harmônicos. Os harmônicos podem ter uma intensidade menor do que a fundamental e por isso são menos perceptíveis que a fundamental ou muitas vezes inaudíveis.

$$\sum_{n=1}^{\infty} \frac{1}{n} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots \quad (6.1)$$

O segundo harmônico (F2) desta série é equivalente a vibração de uma corda com metade do tamanho da corda inicial e tem o dobro da frequência inicial (F1). O terceiro harmônico (F3) tem a vibração de uma corda com 1/3 do tamanho, e terá o triplo de sua frequência. O quarto harmônico (F4) tem 4 vezes a frequência inicial e o quinto harmônico (F5) terá 5 vezes a frequência inicial e assim por diante.

Usando como exemplo nossa corda que vibra a 100 Hz, seus harmônicos vibrarão como apresentado na Tabela 6.6.

Isso implica em dizer que, se fizermos instrumentos musicais que possuem estas frequências, eles sempre soarão de forma afinada pois

#	Frequência
F1	200Hz
F2	400Hz
F3	600Hz
F4	800Hz
F5	1600Hz
...	...

Tabela 6.2: Série harmônica da frequência de 200 Hz

#	Frequência
F1	150Hz
F2	300Hz
F3	450Hz
F4	600Hz
F5	7500Hz
...	...

Tabela 6.3: Série harmônica da frequência de 150 Hz

6.1.2 A afinação pitagórica

Para aumentar nossas possibilidades, podemos verificar o que acontece se dispararmos a mesma série para o segundo harmônico. Como ele é o dobro do primeiro harmônico, teremos a mesma série, mas com todos as frequências dobradas e com isso alcançaremos logo nos primeiros harmônicos frequências muito altas e veremos que todas as frequências desta série estão presentes na primeira.

Podemos fazer isso a partir do segundo harmônico, 300Hz, no entanto, também é possível verificar que este harmônico, 300Hz, está presente na série harmônica que tem como fundamental a frequência de 150Hz.

Partindo da série harmônica, podemos continuar calculando harmônicos que estão presentes em mais de uma série e fazendo novas séries harmônicas a partir de um determinado harmônico para compor uma tabela de frequências que são harmônicas entre si. Note que estamos sempre usando uma fundamental que está entre o primeiro harmônico da série original (100Hz) e seu segundo harmônico (200Hz). O resultado pode ser visto na Tabela 6.4.

Note que estamos truncando os valores na segunda casa decimal para que os mesmos caibam na página do livro. Certamente, na natureza, quando uma corda vibra, a mesma não precisa ter esta preocupação.

A última frequência desta série, 202.7Hz, é muito próxima do segundo harmônico da série original. Por isso, convencionou-se dividir as frequências

#	F1	F2	F3	Ajuste
1	100	200	300	(300 / 2 = 150)
2	150	300	450	(450 / 4 = 112,5)
3	112,5	225	337,5	(337,5 / 2 = 168,75)
4	168,75	337,5	506,25	(506,25 / 4 = 126,56)
5	126,56	253,12	379,68	(379,68 / 2 = 189,84)
6	189,84	379,68	569,52	(569,52 / 4 = 142,38)
7	142,38	284,76	427,14	(427,14 / 4 = 106,78)
8	106,78	213,56	320,34	(320,34 / 2 = 160,17)
9	160,17	320,34	480,51	(480,51 / 4 = 120,13)
10	120,13	240,26	360,39	(360,39 / 2 = 180,19)
11	180,19	360,38	540,57	(540,57 / 4 = 135,14)
12	135,14	270,28	405,42	(405,42 / 4 = 202,71)
13	202,71	..	..	..

Tabela 6.4: Séries harmônicas consecutivas e ajustadas

desta série em um grupo de 12, “ajustando” a última frequência para 200Hz.

6.1.3 Temperamento Justo

Outra forma de chegar nestas 12 frequências seria ignorar a proporção do terceiro harmônico e ajustar todos os valores para os valores de uma Progressão Geométrica, onde a razão seria $\sqrt[12]{2}$. Podemos agora, ordenar estas frequências, como apresentado na Tabela 6.5.

Chegamos assim em dois sistemas de afinação. Chamamos o primeiro sistema de “**Afinação Justa**” ou “**Afinação Pitagórica**”, onde as frequências são encontradas a partir de uma relações matemáticas simples, como, por exemplo, $3/2$. Seria possível utilizar outras relações matemáticas, como $5/4$, $9/8$, $19/11$, ou outros. O segundo sistema, que utiliza uma proporção igual para dividir as notas, no caso $\sqrt[12]{2}$, é chamado de “**Afinação Temperada**”. Podemos observar que poderíamos mudar este “temperamento” e criar outros sistemas de afinação temperado que divide a escala em 10, 11, 15 ou qualquer outra quantidade de notas, simplesmente mudando a razão desta Progressão Geométrica.

Cravo bem temperado

História do lá 440

#	Freq da série	Freq da PG
1	100	100
2	106,78	105,95
3	112,5	112,25
4	120,13	118,92
5	126,56	125,99
6	135,14	133,48
7	142,38	141,42
8	150	149,83
9	160,17	158,74
10	168,75	168,18
11	180,19	178,18
12	189,84	188,77
13	202,71	200

Tabela 6.5: Frequências da série dispostas em ordem crescente

#	Frequência	Nota	Relação
F1	110Hz	Lá 1	Frequência fundamental
F2	220Hz	Lá 2	Oitava acima da fundamental
F3	330Hz	Mi 3	Quinta justa acima do segundo harmônico
F4	440Hz	Lá 3	Duas oitavas acima da fundamental
F5	550Hz	Dó # 4	Terça maior acima do quarto harmônico
F6	660Hz	Mi 4	Quinta justa acima do quarto harmônico

Tabela 6.6: Série harmônica da frequência de 110 Hz

6.1.4 Série harmônica e escala natural

Faremos agora um exercício que consta como uma possibilidade de ter servido para o surgimento do que chamamos de escala natural. Vamos tomar a série harmônica da nota Dó, cuja frequência é de 131 Hz. Com um pouco de cálculo, podemos verificar que o Peguemos também a série harmônica de sua quint

Esta série nos dá algumas linhas para pensar a relação entre as alturas musicais. A primeira dela é entender que alturas que possuem o dobro da frequência estão presentes em uma frequência fundamental. Esta relação, que logo em breve chamaremos de oitava, nos traz possibilidades de criar sons que sempre irão soar de forma consonante, como se fossem uma coisa só. O mesmo acontece com os próximos harmônicos, cuja relação estamos chamando de terça maior e quinta justa.

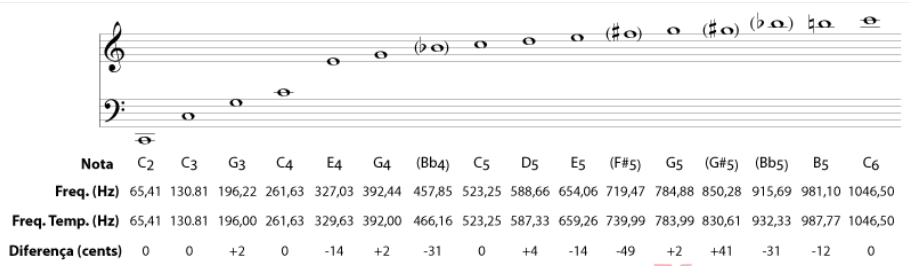


Figura 6.2: Representação da Série harmônica na partitura

6.1.5 Relações matemáticas da música

Continuando com as relações matemática da música, podemos pensar que um sistema pode ser afinado a partir de qualquer frequência e que a frequência fundamental pode ser mudada desde que exista uma combinação (ou afinação) entre as pessoas que irão tocar juntas. Uma possibilidade de pensar no sistema de afinação é calcular as frequências a partir de uma determinada frequência fundamental.

Ciclo de quinta

Uma primeira relação parte da série harmônica, onde um

Ciclo de quarta

Ciclo de terças maiores

Ciclo de terças menores

Assim, é possível usar uma escala linear para representar as notas musicais.

A divisão das notas em 12 se baseia no conceito de oitava musical, ou da repetição das notas a cada ciclo de 8 notas e isso está relacionado a Série Harmônica apresentada anteriormente. Apesar de isso ser uma questão cultural, há um pouco de física e de matemática nesta divisão da escala.

Além disso, cada nota musical possui uma frequência aproximada. Aproximada pois há várias possibilidades de frequências que podem ser utilizadas como frequência fundamental na conta abaixo. Para calcular as frequências de cada nota, vamos utilizar o que chamamos de ciclo de quinta.

6.2 Envelopes

Multiplicação e volume

6.2.1 Envelope ADSR

* ADSR

6.2.2 Envelope AR

* AR * Outros

6.3 Síntese Aditiva

Operação sobre áudio DC OFFSET e soma

6.4 Síntese Subtrativa

- Teoria rápida de Filtros

6.5 Modulação de Amplitude

* Modulação em Anel

6.6 Modulação de Frequência

6.7 Modulação de Fase

6.8 Síntese Granular

6.9 Modelagem Física - Karplus-Strong

Capítulo 7

3 Perspectivas para continuar este trabalho

7.1 Music Information Retrieval

7.1.1 Medindo sinais de áudio

7.1.2 Amplitude de Sinal

♪ Amplitude de pico:

$$A_p(x, I) = \max\{x(t) \mid t \in I\}$$

♪ Amplitude média:

$$A_m(x, I) = \frac{\int_I |x(t)| dt}{|I|}$$

♪ Amplitude RMS (Root-Mean-Square):

$$A_{rms}(x, I) = \sqrt{\frac{\int_I x(t)^2 dt}{|I|}}$$

Exemplos:

♪ senoide ($A_p = 1$, $A_m = \frac{2}{\pi} \approx 0.637$, $A_{rms} = \frac{\sqrt{2}}{2} \approx 0.707$)

♪ quadrada ($A_p = A_m = A_{rms} = 1$)

♪ dente-de-serra ($A_p = 1, A_m = \frac{1}{2}, A_{rms} = \frac{1}{3}$)

♪ 110 V AC possui pico de $\frac{110}{0.707} \approx 155.56$ V (equivalente DC).

7.2 DAFx

7.2.1 Dinâmicos

- Compressor - Noise Gate - Expander - Ducker

7.2.2 Filtros

- Equalizadores - Passa alta - Passa baixa - Notch

7.2.3 Espacialização

7.3 NIME

7.3.1 Instrumentos tradicionais

7.3.2 O instrumento digital (tripartite)

7.3.3 Sintetizador

Apêndices

Pre Print
V.0.0.1

Apêndice A

Computação musical e Linux

[McCartney:2002]

A necessidade de diminuir a latência em um sistema computacional para trabalhar com música nos leva a possibilidade de modificar o sistema operacional para que o mesmo funcione em tempo real. Com o surgimento do Linux, muitos pesquisadores da área de computação musical adotaram este sistema operacional como base para seus sistemas computacionais musicais.

A.1 Uma comunidade organizada

LAD e LAC

A.2 Distribuições de Linux para música

Hoje, o custo de um estúdio é metade hardware, metade software. Os plugins de produção musical são caros e eu acho que os programadores que fazem estas ferramentas maravilhosas devem ser bem remunerados. Por isso, não é errado usar ferramentas fechadas e proprietárias. No entanto, eu também não acho certo usar estas ferramentas de forma pirata ou ilegal. Assim, para ter a chance de aprender computação musical, criar coisas musicais no computador e não violar o direito autoral de outros programadores, nada melhor do que usar

ferramentas livres, que além de estarem disponíveis gratuitamente ainda nos permite acessar seu código-fonte para aprender como ela foi desenvolvida e talvez aprimorar o seu funcionamento.

Então, neste livro iremos divulgar ferramentas livres para criação musical. E com isso, surge uma ideia, e se juntarmos as ferramentas livres para Linux em uma distribuição voltada para este fim? Bem, isso já existe. Há diversas iniciativas, ainda ativas ou descontinuadas de distribuições de Linux para áudio e Música como 64 Studio, Musix, Dyne:Bolic, ArtistX, Agnula, Demudi, Audio Slack, KX Studio, AV Linux, entre outras. Destaco aqui duas em especial:

- ♪ Planet CCRMA: Desenvolvido pela Universidade de Stanford, o Planet CCRMA (lê-se "Karma") não é propriamente uma distribuição mas um conjunto de pacotes RPM para ser utilizados em distribuições baseadas no Red Hat como o Fedora ou o CentOS. Entre estes pacotes há um Kernel de baixa latência, drivers para dispositivos de áudio e várias aplicações para esta finalidade. Site oficial: <http://ccrma.stanford.edu/planetccrma/software/>
- ♪ Ubuntu Studio: Baseado na distribuição Ubuntu, o Ubuntu Studio acompanha as versões desta distribuição e possui pacotes adicionais em um repositório próprio para Música, Vídeo e Imagem. Site oficial: <http://ubuntustudio.org/>

Tais distribuições costumam estar recheadas de aplicações mas também trazem configurações específicas do sistema operacional, como um kernel preemptivo, configurações de prioridades de escalonamento e alocação de memória, além de plugins de áudio, o servidor JACK, drivers ALSA e um visual super maneiro!

Precisa usar estas distros? Não precisa. Pode instalar os pacotes na sua distro de linux favorita e ser feliz. No entanto, estas distros costumam ter vários pacotes específicos para trabalhar música e áudio no Linux e isso é muito legal. Além disso, algumas destas distros rodam em Live pen drive. Assim, se você quiser usar uma distro Linux apenas para estudar computação musical, é possível fazer isso em um dispositivo USB.

Exercícios A.2

20. Veja no site [distrowatch](http://distrowatch.net) outras distribuições de Linux para música.

21. Liste algumas figuras legais de áudio para Linux.

A.3 Kernel preemptivo - Um pouco de história

Durante muito tempo os sistemas operacionais não suportavam a execução de vários programas simultaneamente. Estes sistemas operacionais, como o MS-DOS, eram chamados de Mono-tarefa. Boa parte dos sistemas operacionais atuais permitem a execução de vários programas ao mesmo tempo. É bastante comum que, por exemplo, você esteja navegando na Internet e ouvindo música ao mesmo tempo. Estes sistemas operacionais, como o Linux ou o Windows, são chamados Multi-tarefas. Bem, sabendo que o seu computador possui apenas um processador, como ele é capaz de executar vários programas ao mesmo tempo? A resposta é simples: Ele não é. Sim, seu computador executa apenas um programa de cada vez, porém, alterna entre os vários programas que estão executando deixando uma fatia de tempo (time slice) para cada um deles. É como o cinema, várias fotos piscando rapidamente dão a impressão de movimento. Vários programas alternando rapidamente dão a impressão de multi-tarefa. A realidade é que apenas um deles irá executar por vez. O responsável por alternar entre as várias tarefas é o sistema operacional, ou mais especificamente, seu escalonador.

O **escalonador** é uma parte do sistema operacional que irá cuidar da divisão das fatias de tempo (time slice) do processamento entre os vários programas em execução. Há uma série de algoritmos diferentes para escalonamento como, por exemplo, SJF ou FIFO e nem todos trabalham com fatias de tempo. Além de usar fatias de tempo, alguns destes algoritmos de escalonamento tratam os programas em execução com prioridades diferentes. O Linux utiliza esta abordagem. Assim, se um programa for mais importante, ele poderá executar por mais tempo e, se ele for menos importante, ele executará por menos tempo. Esta importância é definida por uma ordem de **prioridades**.

Para garantir que processos (ou tarefas, ou programas) mais importantes executem sempre que for necessário, há a possibilidade de atribuir-lhes uma prioridade maior. Os processos do Sistema operacional, por exemplo, costumam

possuir maior prioridade que os processos do usuário. No Linux, os processos mais importantes possuem prioridade 0 (isto mesmo, zero) e conforme este valor cresce a importância do processo diminui. Os programas que não estão executando ficam esperando em uma fila. Os algoritmos de escalonamento fazem uma conta para tentar dividir de uma maneira justa o processador para todos os processos. Quem está esperando a mais tempo, por exemplo, pode ser chamado primeiro que alguém com maior prioridade. Para alterar entre um processo e outro, o escalonador utiliza um recurso chamado **preempção**.

A preempção é a alternância entre os vários processos da fila causados por um evento externo, como um relógio. Partimos da idéia de que apenas um processo está no processador. O escalonador pode preemptar um processo que já executou por tempo bastante, tirá-lo do processador antes de ele terminar sua execução, colocar outro processo para executar, e garantir com isto a sensação de multi-tarefa. Porém, além dos processos da fila, o usuário também pode requisitar algum processamento. Isto é feito, por exemplo, digitando no teclado ou movendo o mouse pela tela. Isto gera o que chamamos de interrupção de Hardware. Cada dispositivo tem um endereço (chamado IRQ) que identifica se esta interrupção está vindo do teclado, da placa de rede, do mouse ou da placa de som, por exemplo. O tempo entre você pressionar uma tecla e o computador responder por isto, ou seja, atender a esta interrupção, é o que chamamos de **latência**.

Latência, é o tempo que o computador demora para responder quando, por exemplo, uma tecla é pressionada no teclado. Nem todos computadores querem trabalhar com latência baixa. Imagine um servidor da Internet. Normalmente estes computadores não possui nem teclado nem mouse. Usuários não operam estas máquinas por meios de seus dispositivos de entrada e saída e, logicamente, elas não trabalham com preempção forçada. Isto significa que ninguém "fura fila" de execução. Seria o mesmo que dizer: Computador, faça isto. E ele responder: Agora não posso, estou gravando um arquivo. Para um servidor, é mais importante gravar o arquivo. A Latência, neste caso, pode ser bastante alta.

Computadores Desktop trabalham com preempção voluntária do Kernel. Isto significa que, quando há uma interrupção dos usuário, o programa voluntariamente cede seu espaço no processador para que a interrupção do usuário seja atendida. Talvez ele termine de fazer o que estava fazendo se julgar que isto seja importante. A Latência, neste caso, é menor.

Para conseguir uma resposta com baixa latência de um sistema operacional pode demandar que o mesmo seja configurado para isso em seu núcleo, que chamamos de kernel. Alguns sistemas operacionais podem não permitir alterações em seu Kernel. O Linux permite. Então, em sistema Linux, temos a chance de configurar um Kernel preemptível de baixa latência que, como seu próprio nome diz, possui uma latência ainda menor devido ao mesmo permitir preempção. Foi com a intenção de diminuir este tempo de latência em sistemas Linux que um Húngaro chamado Ingo Molnar, desenvolvedor da Red Hat americana, iniciou o projeto de um Kernel com preempção completa, nos idos dos também chamado de Kernel RT, na versão Linux-2.6.23 do kernel.

O Kernel RT do Linux foi modificado para permitir preempção completa e acelerar assim o tempo de resposta do processador as interrupções. Com isso, o usuário não corre o risco de ouvir do computador "Espera um pouco" quando pedir ao mesmo para executar uma tarefa. Só para exemplificar, um dos parâmetros que o escalonador utiliza para a preempção é o System Timer Resolution. Existe um intervalo de tempo (time slice) para cada tarefa que o processador recebe para fazer. Cada sistema utiliza uns milésimos de segundos para gerenciar estes intervalos de tempo. Assim um sistema gasta $1/n$ segundos por intervalo; algo como $1/33$ para o W95, $1/100$ para a série o Windows NT, $1/250$ para o Linux padrão e $1/1000$ para o Linux com Kernel RT.

O tempo de resposta do processador afeta especialmente as aplicações gráficas e multimídias, não apenas em velocidade, mas também em qualidade. Com a utilização de um Kernel RT são menores as chances de um gravador multipista não conseguir executar suas 18 faixas "simultaneamente". Com o tempo de 0,001 segundos de preempção entre uma faixa e outra fará o resultado final, com certeza, soar como um único som. O mesmo se aplicaria para gravações. É normal em um ambiente de produção musical gravar uma faixa enquanto se ouve outras 5 ou 6.

Atualmente, os sistemas computacionais trabalham com um escalonamento muito rápido e não há mais a necessidade de instalar um kernel RT para garantir uma baixa latência no sistema operacional Linux. No entanto, há a possibilidades de configurar melhor o sistema para alcançar tempos de latência menores. O servidor de som JACK, por exemplo, faz esta configuração nos sistemas Linux atuais sem precisar configurar um Kernel RT. No entanto, com o avanço dos microcontroladores e de sistemas embarcados, a discussão de latência sempre retorna quando o assunto é processamento de áudio em tempo real e por

isso, vamos deixar esta história aqui registrada pois nunca sabemos quando isso poderá ser útil em nossas vidas.

A.4 ALSA

Outra peça fundamental dos sistemas Linux para o processamento de som é o que chamamos de ALSA. Atualmente existem placas de som com diversas configurações distintas. Várias entradas, várias saídas, entrada / saída para controlador MIDI, full-duplex (capacidade de gravar e ouvir ao mesmo tempo), com conexão USB, PCI, Firewire, HDMI, e por aí vai. A função de deixar sua placa funcionando é do sistema operacional. O sistema operacional, com certeza, não conhece todas as placas de som do universo e por isto eles contam com uma ajudinha do fabricante para lhe explicar como conversar com sua placa de som. Esta "ajudinha do fabricante" é o tal driver de dispositivo. Quando o Sistema operacional consegue identificar sua placa a mesma se torna disponível para todas as aplicações que irão utilizá-la.

Bem, se o fabricante não tem Driver para Linux, o que fazer? É neste ponto que entra o ALSA. O ALSA é a parte do Kernel do Linux responsável por conectar seu sistema a sua placa de som. Então, sempre vale a pena dar uma olhadinha na base de dados do ALSA para verificar se existem drivers disponíveis para um periférico antes de comprar uma placa de som.

Além disso, o ALSA também é uma API de programação para áudio e MIDI que inclui:

- ♪ Síntese de MIDI baseada em hardware;
- ♪ Mixagem em hardware de múltiplos canais;
- ♪ Operação Full-duplex;

Além da API de programação, o ALSA inclui também uma série de comandos do sistema operacional para testar e configurar dispositivos de áudio e MIDI.

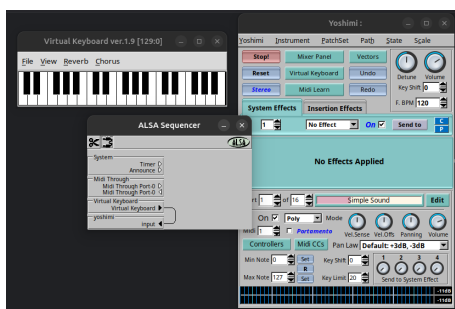


Figura A.1: Sistema ALSA MIDI em ação

- ♪ **speaker-test**: Gera um ruído rosa para testar falantes quando a placa de som está configurada via ALSA.
- ♪ **aplay -l**: lista os dispositivos play back do hardware ou, de maneira mais simples, a(s) saída(s) de sua(s) placa(s) de som.
- ♪ **arecord -l**: lista os dispositivos de gravação do hardware ou, de maneira mais simples, a(s) entrada(s) de sua(s) placa(s) de som.
- ♪ **alsamixer**: Permite ajustar o volume das entradas/saídas do Hardware diretamente.
- ♪ **alsa reload**: recarrega os módulos de som para o ALSA.
- ♪ **aconnect**: Permite conectar dispositivos MIDI.
- ♪ **aconnectgui**: Conectar dispositivos MIDI com GUI.

Dentro desta API, vale a pena dar uma olhada na parte do ALSA MIDI. Nos sistemas Linux, os dispositivos MIDI possuem portas de entrada e portas de saída e é possível criar diferentes conexões entre diferentes dispositivos. Assim, uma aplicação como o vkeybd, é um teclado MIDI que simplesmente não faz nenhum som. No entanto, este teclado virtual pode ser conectado em um sintetizador, como o Yoshimi, por meio da aplicação aconnectgui, conforme mostrado na Figura A.1.

A.5 Servidores de som: JACK, Pulse áudio e pipewire

O Linux tem uma característica que é tratar todos os seus dispositivos de Entrada e Saída como arquivos. Assim, o dispositivo de áudio passa a ser mapeado em um arquivo como `/dev/audio` e tocar um som no computador significa escrever o conteúdo sonoro em um arquivo. Acontece que normalmente não é possível que duas aplicações escrevam em um mesmo arquivo ao mesmo tempo. Logo, apesar de esta ser uma solução muito boa para muitas coisas, tal abordagem impede a utilização do som por mais de uma aplicação ao mesmo tempo.

Com isso, por muito tempo, foi impossível escutar música no navegador e usar um software de video chamadas, por exemplo. Para resolver esta questão, surgiram servidores de som como o pulseaudio, o JACK e o pipewire. Tais servidores de som permitem que mais de uma aplicação utilize a placa de som ao mesmo tempo sendo que o JACK é considerado um servidor de som para usuários avançados e o pulseaudio um servidor de som para usuários leigos. Já o pipewire promete integrar JACK e pulseaudio e é uma promessa que veio para simplificar a configuração do áudio no Linux.

A.6 LADSPA e LV2

LADSPA

LV2

DSSI

VAMP

Apêndice B

Linguagens de programação Musical

[McCartney:2002]

Baseado no artigo: <https://alice.ufsj.edu.br//papers/2018araujoSandySchia.pdf>

Pre Print
V.0.0.1

Apêndice C

APIs para desenvolvimento de sistemas musicais

[McCartney:2002]

Baseado no artigo: <https://alice.ufsj.edu.br//papers/2012SchiavoniSMCT.pdf>

Pre Print
V.0.0.1

Índice Remissivo

- Abordagem, 13
- ACM Transactions on Multimedia Computing, Communications, and Applications, 7
- aconect, 147
- aconectgui, 147
- acústica, 15
- Afinação Justa, 133
- Afinação Pitagórica, 133
- aliasing, 92
- ALICE, 4
- alsa reload, 147
- alsamixer, 147
- altura musical, 38
- Amplitude, 87
- andamento, 29
- ANPPOM, 6
- análise musical computacional, 17
- aplay -l, 147
- arecord -l, 147
- armadura de clave, 46
- arquitetura, 15
- biologia, 14
- CCRMA, 4
- CEGeME, 4
- CIRMMT, 4
- ciências humanas, 15
- clave, 45
- CMMR, 6
- compasso, 31
- CompMus, 4
- composição, 17
- Composição algorítmica, 12
- Comprimento de onda, 87
- Computer Music Journal, 7
- COUT-E, 4
- Crista e Vale, 87
- CSC, 4
- CSound, 3
- DAFx, 5, 11
- DSP, 11
- Duração, 34
- eletrônica, 90
- elétrica/áudio, 89
- engenharia acústica, 15
- engenharia civil, 15
- engenharia de áudio, 89
- engenharia elétrica, 15, 89
- engenharia mecânica, 15
- escalonador, 143
- Etnomusicologia Computacional, 17
- f, 36
- Fa, 39
- FAUST, 3

- ff, 36
- fff, 36
- Figura, 34
- filosofia, 15
- filtro passa alta, 110
- filtro passa baixa, 109
- Foco, 13
- fonoaudiologia, 15
- Frequência, 87
- física, 15

- Hertz, 38
- Hydrogen, 34

- I, 39
- ICMC, 4
- IEEE Multimedia Magazine, 7
- IEEE Signal Processing Magazine, 7
- IEEE/ACM Transactions on Audio, Speech, and Language Processing, 7
- instrumentos ampliados, 11
- IRCAM, 4
- ISMIR, 5
- Journal of Mathematics and Music, 7
- Journal of New Music Research, 7

- Lab, 39
- LAC, 5
- LADSPA, 12
- latência, 144
- LCM, 4
- Leonardo Music Journal, 7
- LilyPond, 16
- linguagens de programação musical, 17

- Live Coding, 18
- live coding, 21
- LMMS, 28
- lutheria, 15

- matemática, 15
- Max Mathews, 3
- Max/MSP, 17
- medicina, 15
- mf, 36
- Mi, 39
- microeletrônica, 90
- MIDI, 16
- Mixxx, 9
- mp, 36
- MuseScore, 17
- Musescore, 45
- musicalização, 18
- musicologia digital, 17
- musicoterapia, 14
- MusicXML, 16
- Musitech, 4
- MustIC, 4

- NESCoM, 4
- neurociência, 14
- NICS, 4
- NIME, 5, 10
- Nome, 34
- NuSom, 4

- Organised Sound, 7
- otorrinolaringologia, 15

- p, 36
- Pausa, 34
- percepção, 14
- performance musical, 17

Período, 87

Pitágoras, 131

pp, 36

ppp, 35

preempção, 144

prioridades, 143

psicoacústica, 14

psicologia, 14

Pure Data, 3, 17

PureData, 10

Quantização, 93

Re, 39

rebatimaento, 92

Recuperação de Informação Musical,
9

Representação e Notação Musical Di-
gital, 16

S, 39

silêncio, 85

SMC, 5

sociologia, 15

Sol, 39

Sonic Pi, 13

SonicPi, 12

Sonicvisualiser, 10

speaker-test, 147

SuperCollider, 3

série harmônica, 129

telecomunicações, 15

teoria musical, 18

Ubimus, 6

Ut, 39

v.0.0.1 - Edição do Autor: Favor não
distribuir, i

Velocidade, 87