MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE GRADUAÇÃO EM ENGENHARIA ELETRÔNICA

Al. JOSÉ ROSA KUIASKI

CONFORMAÇÃO ADAPTATIVA DE FEIXES EM UM ARRANJO DE SENSORES

Rio de Janeiro
2008

INSTITUTO MILITAR DE ENGENHARIA

Al. JOSÉ ROSA KUIASKI

CONFORMAÇÃO ADAPTATIVA DE FEIXES EM UM ARRANJO DE SENSORES

Projeto de Final de Curso apresentada ao Curso de Graduação em Engenharia Eletrônica do Instituto Militar de Engenharia, como requisito parcial para a obtençãoo de Graduação em Engenharia Eletrônica.

Orientador: Prof. José Antonio Apolinário Jr. Dr. Sc.

Rio de Janeiro

2008

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 - Praia Vermelha
Rio de Janeiro - RJ CEP:22290-270

INSTITUTO MILITAR DE ENGENHARIA

Al. JOSÉ ROSA KUIASKI

CONFORMAÇÃO ADAPTATIVA DE FEIXES EM UM ARRANJO DE SENSORES

Projeto de Final de Curso apresentada ao Curso de Graduação em Engenharia Eletrônica do Instituto Militar de Engenharia, como requisito parcial para a obtenção de Graduação em Engenharia Eletrônica.

Orientador: Prof. José Antonio Apolinário Jr. - Dr. Sc.

Aprovada em 25 de agosto de 2008 pela seguinte Banca Examinadora:

Prof. José Antonio Apolinário Jr. - Dr. Sc., IME

Maj. Alberto Gaspar Guimarães - Dr. PUC, IME

Maj. Maurício Carneiro - XXXXXXX?

Rio de Janeiro, 2008

AGRADECIMENTOS

- X1
- X2
- X3
- X4
- X5

Conteúdo

Lista de Figuras

Lista de Tabelas

RESUMO

Este trabalho compõe-se de um estudo das técnicas de conformação adaptativa de feixes baseadas em algoritmos adaptativos com restrições lineares empregadas em arranjos de sensores e da implementação das técnicas mais usuais, visando a aplicação em um arranjo de sensores capaz de maximizar a relação sinal-ruído em uma determinada direção (alvo amigo) e colocar um nulo em uma outra direção, considerada ruído natural ou proposital (interferidor). É feita uma avaliação das técnicas de algoritmo na forma direta (CLMS, ou de *Frost*), na forma decomposta do conhecido *Generalized Sidelobe Canceller* (GSC-LMS e GSC-RLS) e na forma decomposta *Household Transform* LMS.

É feita uma avaliação dos elementos da implementação física do dispositivo necessário para o trabalho com os algoritmos estudados e implementados. Uma sugestão de sistema, equipamentos, *softwares* e literaturas é feita.

ABSTRACT

This work is based on the studies of the techniques of adaptive beamforming based upon linearly constrained adaptive algorithms employed in a sensor array and on the implementation of the most usual techniques, looking forward the application in a sensor array capable of raising the signal-to-noise ratio upon a certain direction (friend target) and to set a null in another direction, considered a natural or proposital noise (jammer). An evaluation of the techniques in algorithms in a direct form (CLMS or Frost's algorithm), in the so-called Generalized Sidelobe Canceller decomposed form (GSC-LMS and GSC-RLS) and in the Household Transform LMS form is carried out.

The analisis of the elements employed at the implementation of the necessary setup for working with those techniques is also carried out. As well a suggestion about the system, the equipments, softwares and theoretical literature.

Capítulo 1

Introdução

Conformação de feixes é um conjunto de técnicas de processamento digital de sinais que visa o processamento espacial e direcional para a transmissão ou recepção de um sinal. Para tanto, vale-se da utilização de um conjunto de sensores ou transmissores, a partir de agora nominados arranjo, e do processamento combinado dos sinais em cada receptor / transmissor.

De uma forma geral, as técnicas de conformação de feixes alteram eletronicamente o padrão polar do arranjo, através da combinação ponderada através de coeficientes fixos ou adaptativos dos sinais e do controle coordenado da fase e amplitudes relativas. Por exemplo, quando deseja-se transmitir um sinal, cria-se um padrão de interferência construtiva na frente de onda na direção de propagação desejada, configurando-se um ganho direcional, assim como pode-se criar interferências destrutivas em outras direções, configurando rejeição (ou perda) espacial. A figura 1.1 ilustra um arranjo genérico com ganho e rejeição direcionais.

O objeto-base deste estudo são as técnicas de conformação de feixes em sensores atuando como receptores. Neste caso, os sinais recebidos em cada um dos sensores são ponderados e somados, de forma que um desejado padrão de radiação seja observado. Por essa razão, as técnicas de conformação de feixes agem como filtros digitais que realizam filtragem espacial. Nesse caso, o interesse maior é a eliminação de fontes de ruido e uma melhor resolução de fontes de sinal, para aumentar a relação sinal-ruído. Tais beamformers, ou antenas inteligentes, podem ser tanto convencionais (fixos), como adaptativos. Beamformers convencionais utilizam um conjunto fixo de coeficientes e um padrão de

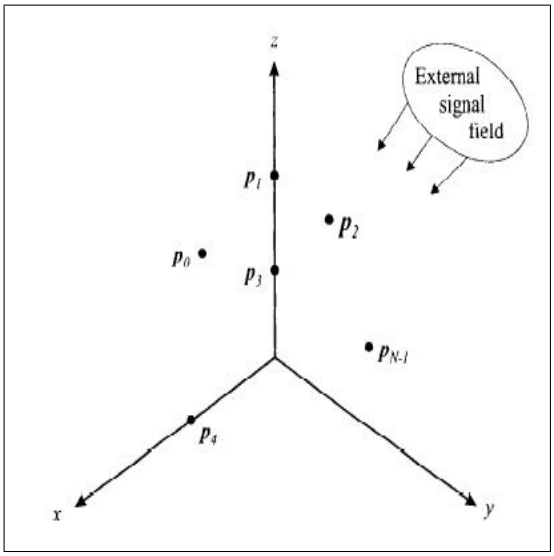

Figura 1.1: **Arranjo genérico de sensores.**

deslocamento de fase, segundo informações da localização e geometria do arranjo e da direção de chegada do sinal. Já beamformers adaptativos combinam essas informações com propriedades do próprio sinal para adaptar sua resposta ao sinal, alterando o valor de seus coeficientes segundo uma equação de update.

Segundo Chern, a idéia inicial das técnicas de conformação de feixes surgiu como objeto de estudo das áreas de sonar e de radares. Nesse meio, aplicam-se especialmente para o uso militar: as chamadas antenas inteligentes, para detecção e monitoramento de alvos, amigos ou inimigos. Hoje em dia, porém, os beamformer apresentam larga utilidade comercial. Por exemplo, podem-se usar técnicas de conformação de feixes em videoconferências, para direcionar a captação para um falante, por exemplo, para contornar a necessidade de uma grande quantidade de microfones; pode-se também utilizá-las no processamento de hidrofones nos oceanos, captando sinais acústicos, assim como em geofones instalados no solo, em busca de sinais sísmicos.O advento da telefonia celular trouxe um novo ramo de aplicação para tais técnicas. Técnicas de conformação de feixes adaptativo são empregadas em estações base, para uplink e downlink em CDMA.

A primeira e mais simples forma de antena adaptativa foi o cancelador de lobo lateral, implementado por Howells em 1950 e posteriormente foi desenvolvido pelo mesmo com participação de Applebaum, no que resultou no diagrama de Howells & Applebaum, como mostra a figura 1.2.

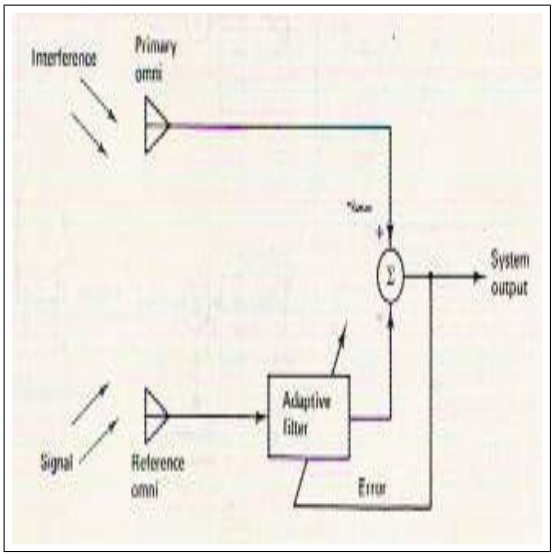

Figura 1.2: **Esquema de Howells & Applebaum**

Posteriormente, em 1960, Widrows e Hoff desenvolveram os algoritmos LMS, propiciando o maior desenvolvimento do estudo das técnicas de conformação de feixes.

O presente estudo aborda técnicas de conformação adaptativa de feixes, menos formalmente conformação de feixes adaptativo. Os principais objetivos deste projeto são:

1 estudar os princípios das técnicas de conformação de feixes e de filtragem adaptativa e adequá-las ao projeto;

2 estudar algoritmos computacionais que realizem essas técnicas e implementá-los;

3 gerar e coletar sinais para utilizar os algoritmos e analisar os resultados.

Para tanto,

- No capítulo 2, apresenta-se a configuração de hardware necessária para o trabalho prático com as técnicas de conformação de feixes. Inicialmente, comenta-se sobre a escolha dos sensores e da geometria do arranjo. Faz-se uma análise do setup utilizado e dos equipamentos que o compõe; as suas configurações e especificações, em especial da aquisição dos dados. Apresenta-se a interface de gravação e o seu funcionamento. Por fim, apresentam-se os softwares e bibliotecas utilizados para gravação e processamento; introduz-se a biblioteca pa_wav do Matlab¨.

- No capítulo 3, faz-se uma introdução à filtragem adaptativa e suas aplicações. Inicia-se pela definição da filtragem ótima e suas características; é apresentada a definição da função objetiva e dos métodos de minimização da mesma, enfocando os utilizados ao longo deste projeto; faz-e uma ilustração da superfície MSE de forma a explicitar a movimentação dos coeficientes e a necessidade de se minimizar uma função objetiva. Alguns algoritmos mais comuns de filtros adaptativos são mostrados, caso do algoritmo LMS e do algoritmo RLS. Por fim, dada a instabilidade dos algoritmos RLS convencionais, são apresentadas formas alternativas de algoritmos RLS, mais estáveis.

- O capítulo 4 introduz a teoria de conformação de feixes, suas características e aplicações. Aborda, em primeiro momento, a teoria clássica do conformação de feixes convencional ou de atraso-e-soma. Em seguida, faz-se uma breve introdução à teoria das restrições lineares e as implicações que tais restrições lineares acarretam na estrutura / performance de um beamformer. As estruturas mais comuns de beamformers são mostradas, primeiramente a estrutura direta na figura do CLMS e do CRLS. Então, trata-se das formas decompostas: o GSC (GSC-LMS e GSC-RLS) e o Household transform.

- A descrição do experimento de uma forma mais detalhada e suas considerações é mostrada no capítulo 5. A montagem do setup, a instalação e utilização dos softwares, bibliotecas e drivers, de acordo com as especificações apresentadas no capítulo 2, e os procedimentos são explicitados. É feita uma análise dos sinais utilizados e do seu tratamento, modulação e codificação utilizadas. São mostrados todos os algoritmos envolvidos implementados em ambiente Matlab¨, bem como seus resultados.

- O capítulo 6 conclui o trabalho, analisando todo o projeto em si, seu cronograma, objetivos e resultados.

Capítulo 2

A montagem do sistema

Para se utilizar as técnicas de conformação de feixes, é necessária a utilização de um arranjo de sensores (array). Um array configura-se por um conjunto de sensores com disposição espacial conhecida, à qual se dá o nome de geometria. A figura 2.1 mostra diferentes arranjos de sensores com geometrias comumente utilizadas.

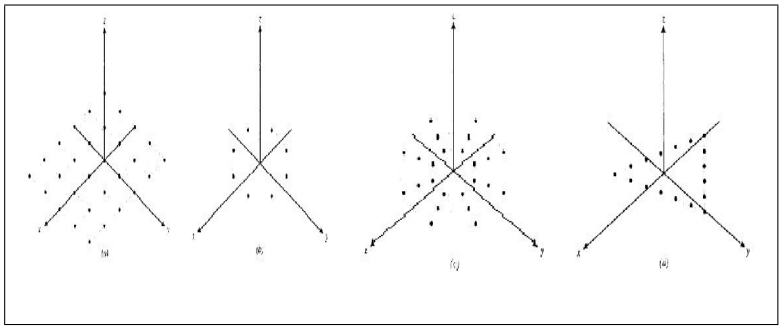

Figura 2.1: **Arranjos de sensores em diferentes geometrias**

Uma descrição mais detalhada da importância e da necessidade da utilização de arrays será abordada no capítulo 3.

2.1 O arranjo de microfones

Por maior facilidade de implementação do setup e menor custo, considerou-se a utilização de microfones como sensores. Os microfones são a peça mais importante na captação de sons, pois é o equipamento que age como transdutor da energia sonora em sinais elétricos que devem corresponder o mais próximo possível das ondas sonoras originais.

Existem vários tipos de microfones presentes no mercado. Os mais comuns são os dinâmicos, os condensados e os de zona de pressão. O funcionamento deles, no entanto é o mesmo. Uma onda sonora é a expansão de regiões de alta e baixa pressão que se formam em um meio, comumente o ar. Esta onda de pressão faz vibrar uma membrana flexível encapsulada. O que difere, no entanto, é a forma como é feita a conversão de sinal sonoro em elétrico.

Um microfone é composto basicamente de uma cápsula na qual se encontra a membrana e o dispositivo transdutor, de uma tela metálica forrada por uma camada de espuma que age como filtro redutor do ruído do vento (Wind screen), de um suporte e de um cabo para carregar o sinal elétrico. O que caracteriza um microfone é seu diagrama polar e sua resposta em freqüência, parâmetros amplamente influenciados pelo mecanismo de transdução.

Nos microfones dinâmicos, por exemplo, a membrana flexível é fixa em uma bobina de cobre que se movimenta livremente ao redor de um ímã ou magneto permanente, como pode ser visto na figura 2.2a. Variações de pressão na membrana geram movimento relativo da bobina ao ímã. A variação de campo magnético na bobina gera uma corrente elétrica segundo Lei de Lorentz. No entanto, o peso da bobina restringe a atuação de um microfone dinâmico em altas freqüências. Para baixas e médias freqüências, os microfones dinâmicos apresentam resposta em freqüência plana (figura 2.2b), são resistentes e dispensam alimentação externa por fontes ou baterias.

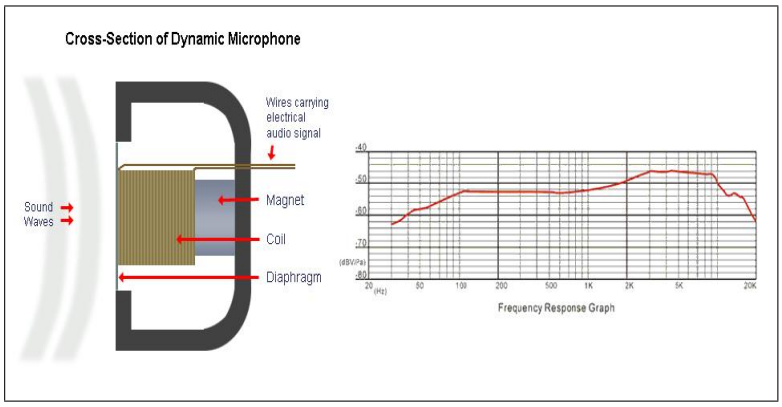

Figura 2.2: **Microfone dinâmico e resposta em freqüência**

Já microfones condensadores, a membrana vibratória é uma fina camada de ”mylar” banhada a ouro e suspensa sobre uma placa base condutiva. De forma mais grosseira, é

um capacitor chamado de condensador, como vê-se na figura 2.3a. Por isso, para que a vibração da membrana crie uma corrente elétrica desejada, é necessário que se alimente esse capacitor com uma tensão constante. Padronizou-se a tensão de 48V, vinda diretamente da interface de gravação, sob o nome de Phantom Power. Ainda, o sinal gerado por esses microfones é baixo, necessitando de pré-amplificação antes de entrar na interface de gravação. Tanto a membrana, quanto a necessidade de alimentação e de pré-amplificação, encarecem o microfone condensador em relação ao dinâmico. Sua resposta em freqüência, porém, alcança freqüências mais altas e é mais plana (figura 2.3b).

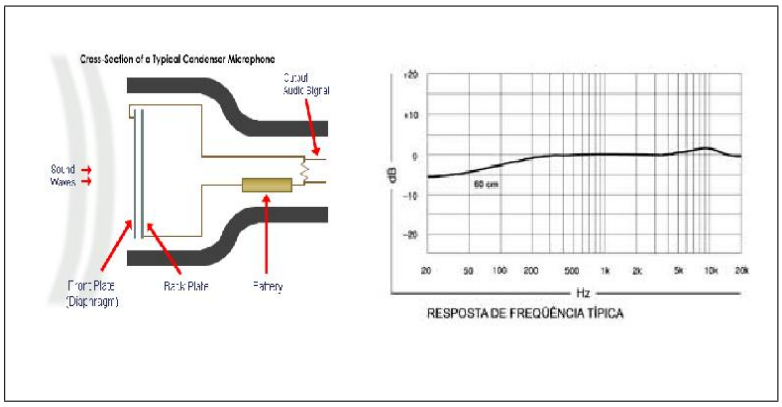

Figura 2.3: **Microfone condensador e resposta em freqüência**

Os microfones de zona de pressão é uma forma alternativa de microfone condensador, no qual uma cápsula de eletreto é montada sobre uma placa metálica e alimentado por uma tensão de 9V e são mais recomendados para a captação de som ambiente.

Ainda, os microfones podem ser classificados segundo o seu diagrama polar. O diagrama polar de um microfone é a sensibilidade que a membrana tem de reagir a estímulos vindos de diferentes direções. A figura 2.4 mostra os diferentes tipos de microfones, segundo os seus diagramas polares.

A figura 2.4a mostra o diagrama polar de um microfone omnidirecional. Esse é o mais abrangente dos microfones, pois possui boa captação em todas as direções. é ideal para a captação de ruído ambiente. O microfone cardióide da figura 2.4b apresenta captação somente em um hemisfério. é o mais comercial dos microfones, mais robusto e conseqüentemente mais utilizado por cantores. Microfones hipercardióides, como na figura 2.4c, apresentam captação em um dos hemisférios e uma pequena captação no outro. Microfones condensadores costumam ter um diagrama polar como o da figura 2.4d, em formato

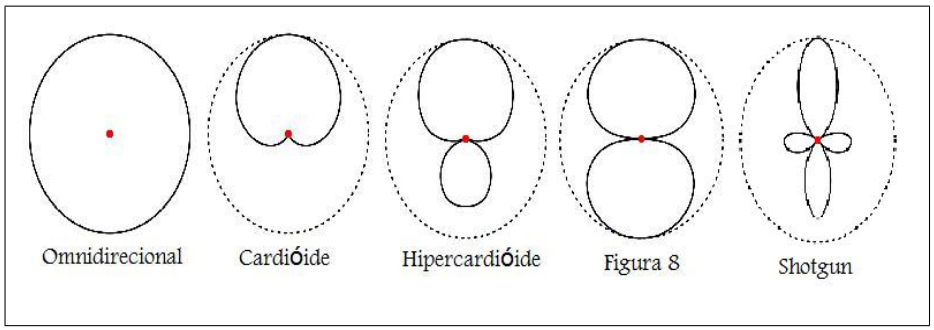

Figura 2.4: **Diagramas polares**

de 8. Microfones shotgun, figura 2.4e, são os mais direcionais.

Dadas as relações custo/benefício, utilizou-se no projeto microfones dinâmicos. Ao todo, 8 microfones da marca Behringer, modelo Ultravoice XM8500, cuja resposta em freqüência e diagrama polar podem ser visualizados no apêndice 1. A utilização de microfones cardióides fez-se conveniente, pois a direção de chegada restringe-se a um dos hemisférios. A pouca captação em sua parte traseira age como um eliminador de ruído ambiente.

2.2 Gravação

O segundo estágio do setup é a interface de gravação. é este o estágio responsável por amostrar os sinais elétricos gerados pelos microfones, convertê-los em dados digitais e disponibilizar uma stream de bits. Uma interface de gravação consiste de uma placa de som digitalizadora ou um hardware específico e dependendo da aplicação ou do sinal em estudo, pode-se fazer necessário o uso de pré-amplificadores nas entradas de seus canais.

Uma interface de gravação pode amostrar múltiplos canais simultaneamente ou de forma multiplexada. Em aplicações comerciais de áudio, uma imprecisão gerada pela multiplexação dos canais não compromete o áudio final. Para a utilização de algoritmos de técnicas de conformação de feixes, cujas relações de fase e amplitude relativas são críticas, no entanto, faz-se necessário que a interface de gravação realize a amostragem simultânea a fim de que tais características se mantenham. Neste projeto, utilizou-se a interface de gravação firewire Firepod¨ da PreSonus, com 10 canais, 2 estéreo balanceados e 8 mono. As especificações e características da interface Firepod¨ encontram-se no apêndice

2. Note-se que a escolha pela utilização de uma interface externa de gravação e não por uma placa de som digitalizadora deu-se, também, pela portabilidade possibilitada.

Um requisito para o trabalho com uma interface multi-canal é a presença de um driver ASIO e de um software de gravação compatível com essa tecnologia. O driver ASIO (do inglês, Audio Stream Input/Output), é um protocolo para resolução de áudio digital especificado pela empresa alemã Steinberg, que possibilita o interfaceamento com baixa latência entre interface de gravação e software.

O software de gravação deve ter a capacidade de comunicação com a interface, para que a mesma possa ser configurada e para que a stream de áudio possa ser lida e convertida em áudio. Quando os sinais são amostrados pela interface em freqüência determinada, as amostras são quantizadas e convertidas em palavras de bits. Estas palavras são organizadas em cadeias de bits, ou streams, que são enviadas para buffers e ficam disponíveis para o software. Alguns parâmetros, como o número de amostras em cada buffer, podem ser determinados no próprio software. No caso do Firepod¨, as streams são enviadas para o computador via porta firewire. Os aplicativos comerciais utilizam a própria stream para atualizar as pistas, ou tracks, relativas a cada canal.

Para os algoritmos implementados neste estudo, o áudio das streams precisou ser convertido em um vetor numérico em ambiente Matlab¨. Isso gerou a necessidade de converter streams em arquivos wave, através da adição de um cabeçalho específico. Na maioria dos aplicativos, isso se faz exportando as pistas como arquivos Ô.wavÕ. Utilizou-se, no entanto, o próprio Matlab¨ como software de gravação, pois o mesmo já carregava as streams como vetores numéricos que podem ser imediatamente tratados pelos algoritmos e a adição de cabeçalhos para a formação de arquivos Ò.wavÓ pode ser incluída no próprio código a ser executado.

Em ambiente Matlab¨, necessita-se habilitar a opção de se trabalhar com o protocolo ASIO. Existe uma biblioteca disponível chamada PA_WAV que possui funções específicas para o trabalho com interfaces de gravação multi-pista. A declaração e parâmetros das funções da biblioteca PA_WAV encontram-se no apêndice 3.

2.3 Reprodução e geração de sinais

Para a geração e reprodução de sinais, a própria biblioteca PA_WAV dispõe de função específica para reproduzir e gravar sinais. Trabalhou-se sempre com uma fonte de ruído e uma fonte de informação, que eram geradas no próprio Matlab¨ e reproduzidas por caixas de som Yamaha MSP3X.

Uma caixa de som é composta por uma caixa ressonante que pode ser tanto de madeira como de resina, de mid-rangers, twitters e woofers, dependendo do range de freqüências trabalhado, de uma tela anti-wind e de um sistema elétrico de amplificação e alimentação. Mid-rangers atuam em freqüências médio-graves. Twitters ou cornetas atuam em freqüência de médio-altas a altas. Woofers atuam em freqüências mais baixas. As caixas de som comerciais utilizam o princípio inverso dos microfones dinâmicos para a reprodução de sons. Uma corrente alimenta um eletroímã permanente que está circulado por uma bobina. Como enunciado pela Lei de Lorentz, a variação de campo magnético gera movimento da espira. Este movimento faz movimentar um diafragma flexível, no caso um cone, que irá gerar pressão nas partículas de ar proporcional à corrente entrante, portanto, ondas sonoras. A figura 2.5 ilustra o funcionamento de uma caixa de som.

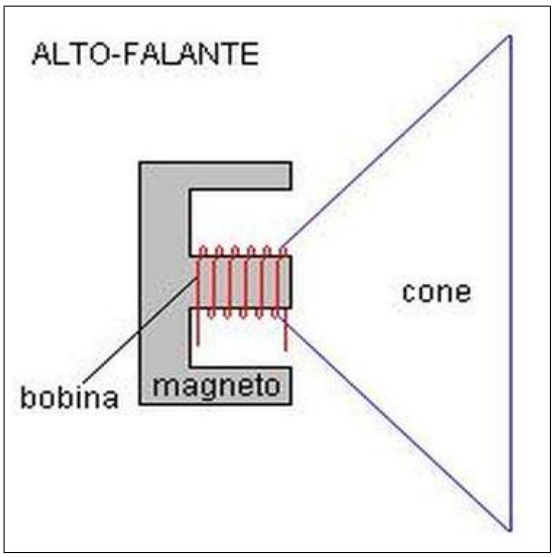

Figura 2.5: **Funcionamento de uma caixa de som**

As caixas de som devem reproduzir com fidelidade o som; o seu volume deve ser ajustado de forma a evitar a saturação dos auto-falantes.

Capítulo 3

Introdução à filtragem adaptativa

Filtros adaptativos possuem parâmetros que variam constantemente no tempo, de forma a alcançar um determinado desempenho. Isso se faz através de um algoritmo de atualização de seus coeficientes e doutras informações, usualmente vindas de um sinal de referência.

Um filtro adaptativo é necessário sempre que especificações fixas sejam desconhecidas ou que as especificações não possam ser atendidas por filtros invariantes no tempo [Diniz]. Não existe uma definição quando a linearidade dos filtros adaptativos. Alguns autores apresentam filtros adaptativos como não lineares, pois a dependência dos sinais de entrada faz com que as condições de aditividade e homogeneidade não sejam satisfeitas. Outros consideram filtros adaptativos lineares, pois sua saída a cada instante é a combinação linear das entradas. Considerar-se-ão os filtros adaptativos lineares, segundo a segunda definição.

A figura 3.1 apresenta a estrutura básica de um filtro adaptativo.

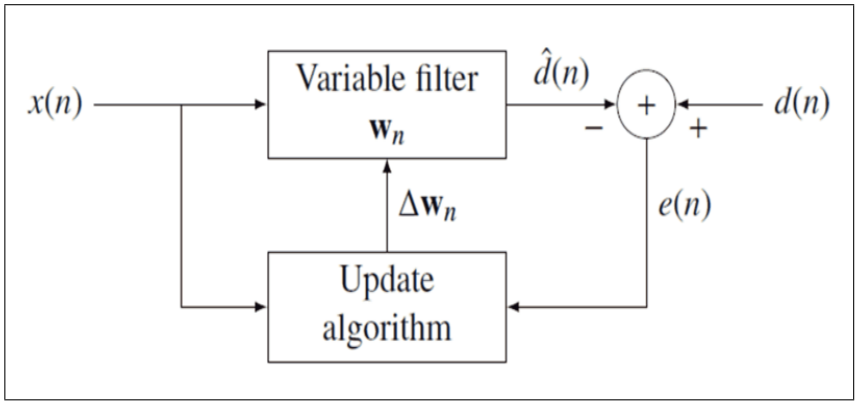

Figura 3.1: **Estrutura de um filtro adaptativo**

Considera-se que existe uma função $\xi(k)$, chamada função objetiva, que é costumeiramente função do sinal de referência (desejado) e do sinal de saída. A escolha da função objetiva é crítica para a estrutura do filtro adaptativo. Deve obedecer a dois critérios:

- Não negatividade: $\xi(k) \geq 0$
- Condição ótima: $\xi(k) = 0$

Isso quer dizer que o objetivo de um filtro adaptativo é ajustar os seus parâmetros (coeficientes) $\omega(k)$, de forma a minimizar um determinada função objetiva $\xi(k)$, de tal forma que o sinal na saída do filtro, $y(k)$, se iguale ao sinal de referência, $d(k)$. Uma forma de se analisá-la é considerando-a uma função direta de um sinal de erro, diga-se:

$$\xi(k) = F[e(k)] = F[d(k)y(k)] \tag{3.1}$$

onde e(k) é definida como a diferença entre o sinal desejado e a saída do filtro. Para isso, deve-se considerar tanto a escolha da função objetiva, como do algoritmo que a irá minimizar.

A definição do algoritmo de minimização da função objetiva é o principal fator responsável pela complexidade computacional do processo adaptativo. A forma de minimização utilizada neste projeto foi o método de *steepest-descent*. Consiste na atualização de coeficientes seguindo a variação negativa do vetor gradiente da função objetiva:

$$\omega(k+1) = \omega(k) - \mu.\nabla\xi(k) \tag{3.2}$$

onde μ é o fator de convergência ou, mais informalmente, o passo da função. Uma visão mais ampla do método de *steepest descent* será abordada no item 3.1.2.

Pode-se determinar a função objetiva de diversas formas, contanto que as condições de não-negatividade e de condição ótima sejam satisfeitas. As formas mais comuns são

- MSE Ð Mean Square Error: $F[e(k)] = E[|e(k)|^2]$
- LS Ð Least Square: $F[e(k)] = \frac{1}{k+1}\sum_{i=0}^{k}|e(k-i)|^2$
- WLS Ð Weighted Least Squares: $F[e(k)] = \sum_{i=0}^{k}\delta^2|e(k-1)|^2$

3.1 Filtro ótimo

Definindo-se a função objetiva como MSE, obtém-se:

$$F[e(k)] = E[|e(k)|^2] = E[d^2(k)2d(k)y(k) + y^2(k)] \tag{3.3}$$

Esta é uma solução teórica, uma vez que demanda uma quantidade infinita de amostras para ser computada. Seja o combinador linear da figura 3.2.

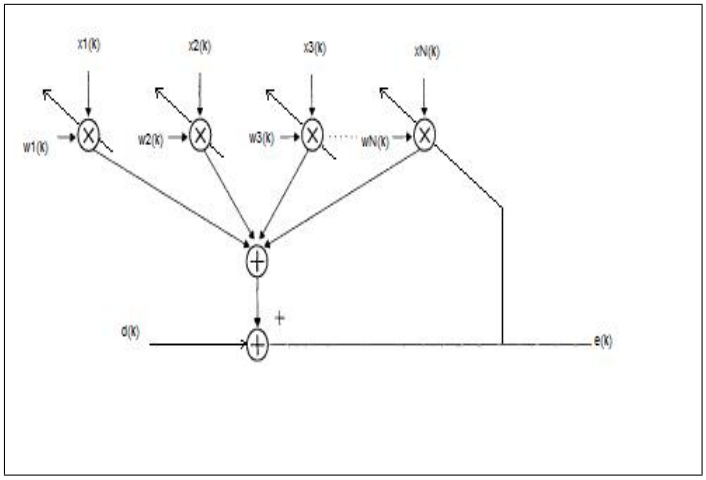

Figura 3.2: **Combinador linear**

Suponha as entradas $x_i(k)$ e um filtro com ordem N de tal forma que

$$y(k) = \sum_{i=0}^{N} \omega(k) x_i(k) = \omega^T(k)\mathbf{x}(k) \tag{3.4}$$

onde $\mathbf{x}(k) = [x_0(k) \quad x_1(k) \quad ... \quad x_N(k)]^T$ é o vetor de entrada e $\omega(k) = [\omega_0(k) \quad \omega_1(k) \quad ... \quad \omega_N(k)]$ é o vetor dos coeficientes. No entanto, para um filtro adaptativo, cada termo xi(k) representa um atraso de i amostras do sinal $\mathbf{x}(k)$, ou seja, $x_i(k) = x(k-i)$. Por conseqüência, a estrutura do filtro adaptativo é mostrada na figura 3.3 e dada por:

$$y(k) = \sum_{i=0}^{N} \omega(k) x(k-i) = \omega^T(k)\mathbf{x}(k) \tag{3.5}$$

com entrada $\mathbf{x}(k) = [x(k) \quad x(k-1) \quad ... \quad x(k-N)]^T$ e com o respectivo vetor de coeficientes $\omega(k) = [\omega_0(k) \quad \omega_1(k) \quad ... \quad \omega_N(k)]^T$

Comentou-se que

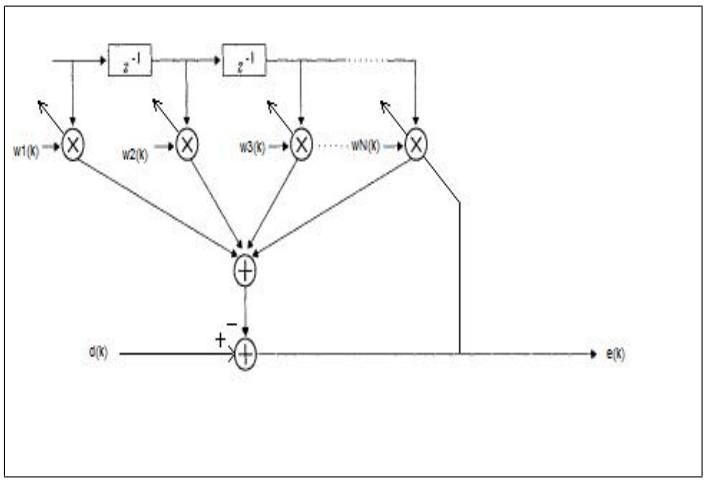

Figura 3.3: **Atrasos em um filtro adaptativo**

$$\xi(k) = E[|e(k)|^2] = E[d^2(k)2d(k)y(k) + y^2(k)] \tag{3.6}$$

Logo,

$$\begin{aligned} \xi(k) &= E[|d^2(k)|]2E[d(k)y(k)] + E[y^2(k)] \\ &= E[d^2(k)] - 2\omega^T E[d(k)\mathbf{x}(k)] + \omega^T E[\mathbf{x}(k)\mathbf{x}^T(k)]\omega \end{aligned} \tag{3.7}$$

Dado que $E[d(k)\mathbf{x}(k)]$ é o vetor correlação cruzada entre o sinal desejado e o sinal de entrada e que $E[\mathbf{x}(k)\mathbf{x}^T(k)]$ é a matriz de correlação do sinal de entrada, respectivamente $\mathbf{p}$ e $\mathbf{R}$, podemos reescrever a fórmula 3.3 como

$$\xi(k) = E[d^2(k)] - 2\omega^T\mathbf{p} + \omega^T\mathbf{R}\omega \tag{3.8}$$

Quer-se o vetor de coeficientes para o qual seja mínimo o valor de $\xi(k)$. Logo, quer-se ω_{opt} para o qual o gradiente de $\xi(k)$ seja nulo.

$$\begin{aligned} \nabla\xi(k) = \frac{\partial\xi(k)}{\partial\omega} &= [\frac{\partial\xi(k)}{\partial\omega_0}\frac{\partial\xi(k)}{\partial\omega_1}...] = 0 \\ &= -2\mathbf{p} + 2\mathbf{R}\omega = 0 \end{aligned} \tag{3.9}$$

O que leva ao valor ótimo

$$\omega_{opt} = \mathbf{R}^{-1}\mathbf{p} \tag{3.10}$$

também conhecida como solução de Wiener. Convem observar que a solução ótima é um conjunto fixo de coeficientes, portanto, invariante no tempo. Como já se citou anteriormente, é uma solução teórica, visto que a definição da matriz de correlação $\mathbf{R}$ e do vetor de correlação cruzada $\mathbf{p}$ necessitam de infinitas amostras.

3.1.1 Superfície MSE

Da equação 3.8, vê-se que a função erro é precisamente uma função quadrática do vetor de coeficientes ω, quando as componentes de entrada e o sinal desejado são variáveis estocásticas estacionárias. Assumindo que $E[d^2(k)]$ é a variância do sinal desejado $d(k)$ com valor médio nulo, representada por σ_d^2, podemos reescrevê-la como

$$\xi(k) = \sigma_d^2 - 2\omega^T T\mathbf{p} + \omega^T \mathbf{R}\omega \tag{3.11}$$

Ou seja, a função objetiva, nesse caso, forma uma superfície hiperparabolóide [Diniz]. De uma forma mais simplificada, para apenas dois coeficientes, a superfície se torna um parabolóide que, segundo Widrows, deve ser côncava para cima e positiva.

A figura 3.4 representa a superfície de performance para dois coeficientes. Note-se que existe apenas um ponto mínimo global, o Òfundo da tigelaÓ [Stearns], que corresponde a ω_{opt}. Com uma função quadrática, não existem mínimos locais.

3.1.2 O método de steepest-descent

Suponha-se que exista apenas um coeficiente. A superfície de performance reduzir-se-ia a uma parábola, como representada na figura 3.5 e pode ser representada por

$$\xi = \xi_{min} + \lambda(\omega - \omega*)^2 \tag{3.12}$$

onde $\omega*$ é o valor ótimo.

Computacionalmente, escolhe-se um valor de ω, diga-se ω_0 e incrementa-se um valor para se achar ω_1, de acordo com o valor negativo do declive da função nesse ponto. Da mesma forma acha-se ω_2. O valor $\omega*$ é alcançado pela repetição desse procedimento.

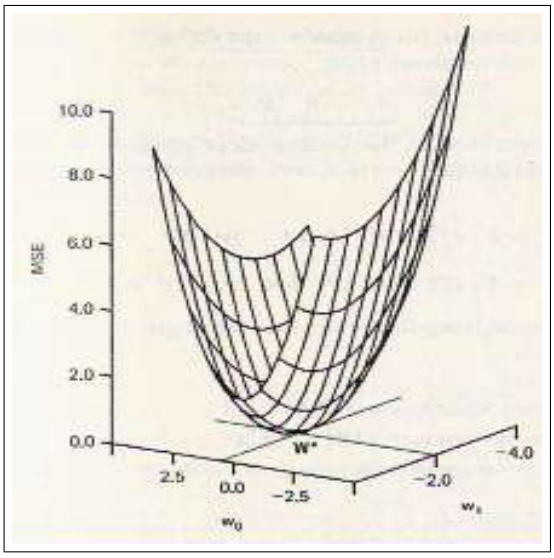

Figura 3.4: **Porção de uma superfície de performance bidimensional**

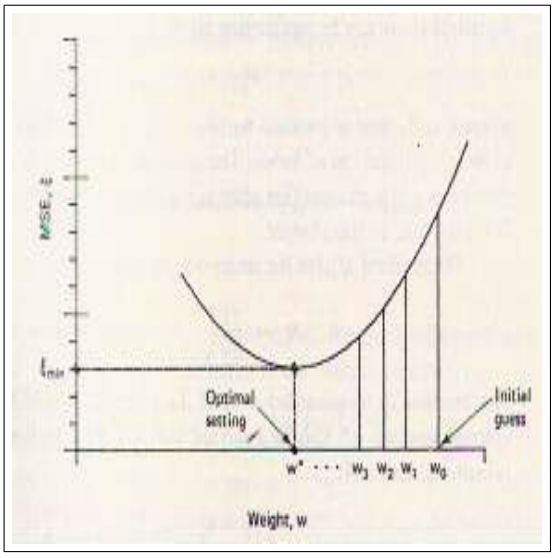

Figura 3.5: **Superfície MSE para um coeficiente único**

Assim, pode-se escrever uma função para representar a estimação pelo gradiente

$$\omega_{k+1} = \omega_k - \mu(\omega_k \omega*) \tag{3.13}$$

de modo que μ é um parâmetro que governa a taxa de estabilidade e convergência. Avaliando a função objetiva para cada iteração, ou seja, para cada valor de ω_k, observa-se que o seu gráfico decai de um valor inicial para o valor de ξ_{min}, no que se chama "curva de aprendizado". A figura 3.6 mostra o gráfico de uma curva de aprendizagem e a figura 3.7 mostra a projeção de uma superfície MSE para dois coeficientes, com a convergência dos coeficientes para o vetor ótimo.

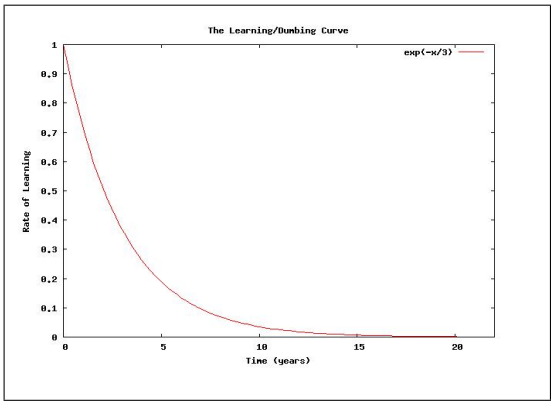

Figura 3.6: **Curvas de aprendizagem**

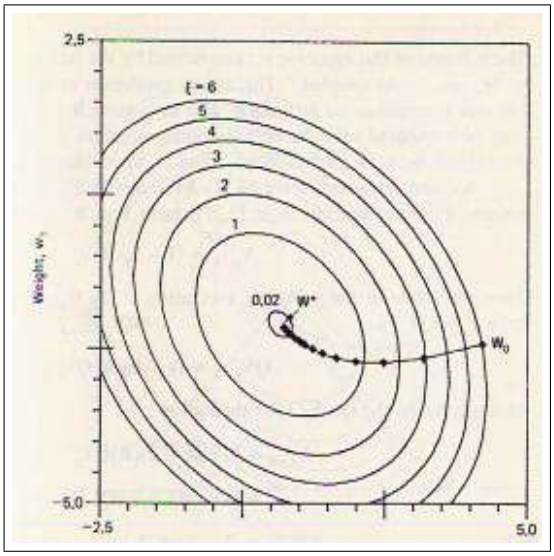

Figura 3.7: **Projeção da superfície MSE. Curvas de nível.**

O valor μ define, por exemplo, a velocidade com a qual a curva de aprendizado converge para o valor de ξ_{min}. Segundo [Diniz], uma boa escolha para garantir a convergência da função objetiva é considerar μ dentro do limite

$$0 < \mu < \frac{1}{\lambda_{max}} \tag{3.14}$$

onde λ_{max} é o maior autovalor da matriz de autocorrelação $\mathbf{R}$.

3.2 Algoritmo Least-Mean-Square (LMS)

Modificando-se a função objetiva, pode-se conseguir uma simplificação significante no cômputo do vetor gradiente. Essa simplificação leva a uma outra definição de um algoritmo para percorrer a superfície de performance. Isso é conseguido, pois os algoritmos LMS não requerem estimação off-line do gradiente [Widrows], ou seja, não requer infinitas amostras, mas sim, uma janela finita, ou repetição de dados.

De modo geral, os algoritmos LMS são de baixa complexidade e convergem, em um meio estacionário. Por isso, são amplamente utilizados em sistemas adaptativos.

Essa imprecisão faz com que a matriz de autocorrelação $\mathbf{R}$ e o vetor de correlação cruzada $\mathbf{p}$ tornem-se estimados, ou seja, $\dot{\mathbf{R}}$ e $\dot{\mathbf{p}}$, respectivamente. Logo, a representação do gradiente $\nabla\xi(k)$ também é uma estimada e pode ser dada por

$$\nabla\xi(k) = -2\dot{\mathbf{p}}(k) + 2\dot{\mathbf{R}}(k)\omega \tag{3.15}$$

mas ainda

$$\begin{aligned} \dot{\mathbf{R}}(k) &= \mathbf{x}(k)\mathbf{x}^T(k) \\ \dot{\mathbf{p}}(k) &= d(k)\mathbf{x}(k) \qquad e \\ y(k) &= \mathbf{x}^T(k)\omega(k) \end{aligned} \tag{3.16}$$

então, $\nabla\xi(k) = 2\mathbf{x}(k)[-d(k) + \mathbf{x}^T(k)\omega(k)] = -2\mathbf{x}(k)e(k)$ pois considera-se o erro $e(k) = d(k)y(k)$. Uma vez que é válida a fórmula 3.13, a equação de atualização dos coeficientes torna-se

$$\omega(k+1) = \omega(k) + 2\mu e(k)\mathbf{x}(k) \tag{3.17}$$

A forma genérica do algoritmo LMS básico é mostrada no quadro 1:

Algoritmo LMS

$X(0) = \omega(0) = [0 \quad 0 \quad 0 \quad ... \quad 0]^T$

Para k ≥ 0

$e(k) = d(k) - y(k)$

$\omega(k+1) = \omega(k) + 2\mu e(k)$

e o seu diagrama de blocos é mostrado na figura 3.8:

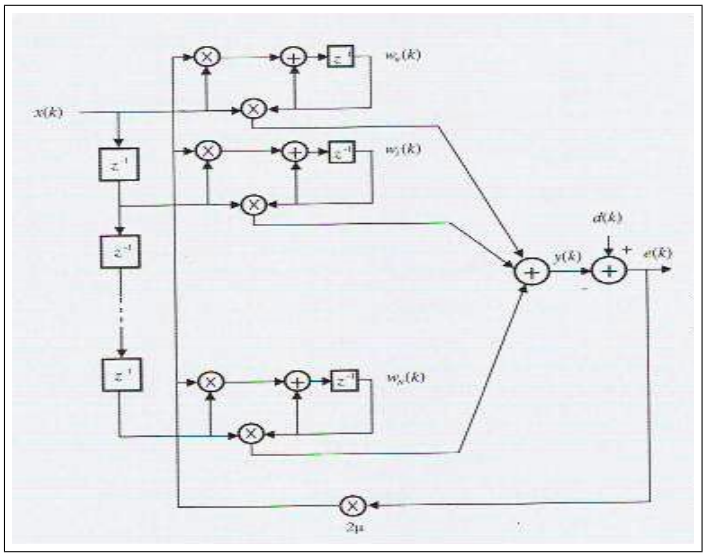

Figura 3.8: **Diagrama de blocos do LMS**

Em um meio não estacionário, no entanto, as medidas de estimação da matriz R e do vetor p são variantes no tempo. Essa variação faz com que a solução ótima também varie no tempo, no entanto, a convergência de algoritmos LMS não é rápida o suficiente para acompanhar essas mudanças, o que compromete a sua performance.

3.3 Algoritmos Recursive-Least-Square (RLS)

O objetivo dos algoritmos RLS é escolher os coeficientes do algoritmo adaptativo de forma a tornar a saída $y(k)$ o mais parecida possível com o sinal desejado dentro de um intervalo de tempo. Nesse caso, requer-se informação *a priori* do sinal de entrada e, ainda, a função objetiva $\xi(k)$ é determinística.

Para os algoritmos *least-squares*, a função objetiva é dada por:

$$\begin{aligned} \xi(k) &= \sum_{i=0}^{k} \lambda^{k-i} e^2(i) \\ &= \sum_{i=0}^{k} \lambda^{k-i} [d(i) - \mathbf{x}^T(i)\omega(k)]^2 \end{aligned} \tag{3.18}$$

onde $e(i)$ é o erro *a posteriori* e λ é o conhecido fator de esquecimento. O fator de esquecimento consiste em um peso exponencial que consideram maiores os efeitos de amostras recentes na equação de *update*. O valor de λ deve ser escolhido entre $0 \leq \lambda < 1$. Observe-se que na definição dos algoritmos LMS, usou-se o erro *a priori*, enquanto nos algoritmos RLS usa-se o erro *a posteriori*.

Para essa função objetiva $\xi(k)$, a aproximação por gradiente em respeito a ω fica na forma:

$$\nabla\xi(k) = \frac{\partial\xi(k)}{\partial\omega} = -2\sum_{i=0}^{k} \lambda^{k-i}\mathbf{x}(i)[d(i) - \mathbf{x}^T(k)\omega(k)] \tag{3.19}$$

Igualando-se a zeros, atrav'es da relação

$$-\sum_{i=0}^{k} \lambda^{k-i}\mathbf{x}^T(i)\omega(k) + \sum_{i=0}^{k} \lambda^{k-i}\mathbf{x}(i)d(k) = [0 \quad 0 \quad ... \quad 0]^T \tag{3.20}$$

resultando na expressão do coeficiente ótimo ω_{opt}

$$\begin{aligned} \omega(k) &= \left[\sum_{i=0}^{k} \lambda^{k-i}\mathbf{x}(i)\mathbf{x}^T(i)\right]^{-1} \sum_{i=0}^{k} \lambda^{k-i}\mathbf{x}(i)d(i) \\ &= \mathbf{R}_D^{-1}(k)\mathbf{p}_D(k) \end{aligned} \tag{3.21}$$

onde $\mathbf{R}_D$ e $\mathbf{p}_D$ são a matriz de autocorrelação e o vetor de correlação cruzada determinísticos.

Computacionalmente, o cálculo da inversa de $\mathbf{R}_D(k)$ é custoso, e tende a ser evitado. Para o algoritmo RLS convencional, pode-se calculá-la por

$$\mathbf{S}_D(k) = \mathbf{R}_D^{-1}(k) = \frac{1}{\lambda}\left[\mathbf{S}_D(k-1) - \frac{\mathbf{S}_D(k-1)\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{S}_D(k-1)}{\lambda + \mathbf{x}^T(k)\mathbf{S}_D(k-1)\mathbf{x}(k)}\right] \tag{3.22}$$

Uma versão do algorítmo RLS convencional é apresentada no quadro 2:

Algoritmo RLS

$\mathbf{S}_D(-1) = \delta\mathbf{I}$, com δ pequeno

$\mathbf{p}_D(-1) = \mathbf{x}(-1) = [0 \quad 0 \quad ... \quad 0]^T$

Para cada $k \geq 0$

$\mathbf{S}_D(k) = \mathbf{R}_D^{-1}(k) = \frac{1}{\lambda}\left[\mathbf{S}_D(k-1) - \frac{\mathbf{S}_D(k-1)\mathbf{x}(k)\mathbf{x}^T(k)\mathbf{S}_D(k-1)}{\lambda+\mathbf{x}^T(k)\mathbf{S}_D(k-1)\mathbf{x}(k)}\right]$

$\mathbf{p}_D(k) = \lambda\mathbf{p}_D(k-1) + d(k)\mathbf{x}(k)$

$\omega(k) = \mathbf{S}_D(k)\mathbf{P}_D(k)$

$y(k) = \omega^T(k)\mathbf{x}(k)$

$e(k) = d(k) - y(k)$

Capítulo 4

Conformação de feixes

Um Beamformer, ou também chamado Combinador Linear, é um sistema MISO (Multiple inputs, single output), que recebe vários sinais entrantes e gera um único sinal sainte. A figura 4.1 ilustra a estrutura de um beamformer.

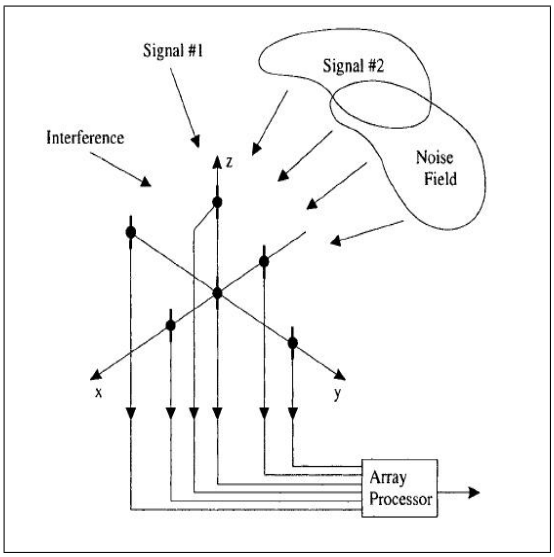

Figura 4.1: **Estrutura de um beamformer**

Utiliza-se um array para as técnicas de conformação de feixes para explorar as características espaciais dos sinais, através das relações de fase e de ângulo de chegada. Para tanto, assume-se que existe um ou vários sinais localizados em alguma região do espaço, bem como se pode haver interferidores em outras regiões. A geometria escolhida é, talvez, o ponto de partida crítico, pois pode estabelecer restrições físicas e operacionais sobre o desempenho de um combinador linear. No caso deste estudo, por exemplo, a utilização

de um array linear considera somente uma direção angular, não levando em consideração, portanto, a altura da fonte sonora, não sendo aplicáveis a radares, por exemplo. Outras geometrias, como circulares e em forma de cruz, possuem diferentes padrões que podem limitar a escolha e usabilidade dos mesmos.

Como já se citou anteriormente, um beamformer apresenta características de um filtro, portanto, é conhecido como filtro espacial. Um dos aspectos que determina o desempenho do filtro é justamente a escolha dos pesos relativos a cada sinal, ou melhor, dos coeficientes multiplicadores dos dados na saída dos sensores.

Outras características, tais como número de sensores, relação sinal-ruído e relação sinal-interferência, ajudam a caracterizar o filtro espacial.

De uma forma mais genérica e didática, todas essas características serão agrupadas em três medidas que definirão o combinador linear: posicionamento dos sensores, ganho dos sensores e direção de chegada.

4.1 Conformação de feixes de atraso-e-soma ou conformação de feixes convencional

A técnica de conformação de feixes mais primitiva consiste em aplicar atrasos relativos em cada sensor, de acordo com a geometria do array e da direção de chegada, e somar os sinais. O resultado é a amplificação do som vindo de determinada direção. No entanto, essa forma de conformação de feixes é pouco efetiva, visto que não considera eliminação de ruído. A figura 4.2 ilustra a estrutura de um beamformer de atraso-e-soma com geometria linear. Observe-se que parte-se do princípio de que os sinais entrantes são todos provenientes de uma única fonte, ou seja, são o mesmo sinal atrasados um em relação ao resto.

é necessário que a fonte sonora esteja a uma distância considerável do sensor para que se possa considerar a frente de onda plana. Uma vez com a fonte sonora no infinito, sabendo-se sua direção de chegada e a velocidade do som no meio, o atraso relativo que deve ser aplicado a cada microfone é

$$atraso = \frac{dcos(\theta)}{v_{som}} \tag{4.1}$$

Onde d é a distância entre os microfones e θ é o ângulo de chegada, em relação à

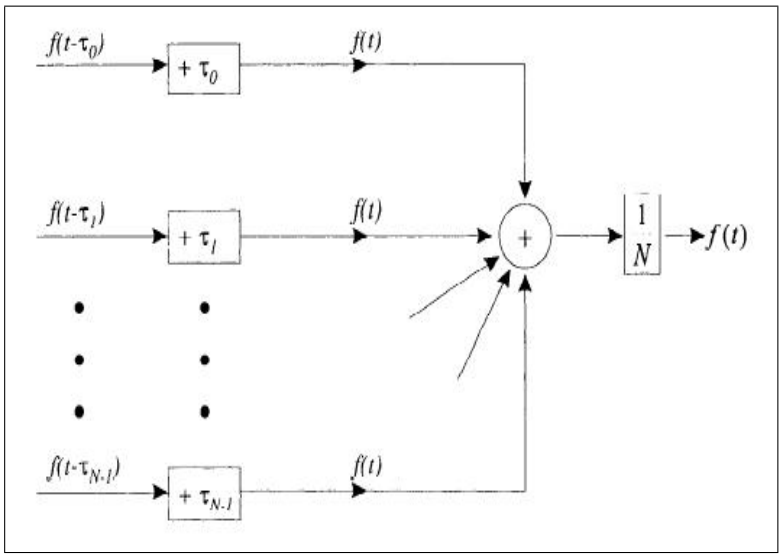

Figura 4.2: **Beamformer de atraso-e-soma**

normal.

4.2 Conformação de feixes adaptativo

Beamformers adaptativos são particularmente úteis em áreas de detecção de sinais, como no processamento de sinais de radares, sonares, hidrofones e geofones, bem como na determinação da direção de chegada de um sinal. Ainda, possui importância em áreas não afins, como na área médica. Pode-se citar, por exemplo, na aplicação com tomógrafos. Um objeto é iluminado de diferentes direções e as reflexões sofridas no objeto são captadas por um conjunto de sensores. As técnicas de conformação de feixes adaptativo auxiliam na reconstrução da imagem do objeto.

Um beamformer adaptativo, também pode ser chamado de antena inteligente ou antena adaptativa, e consiste basicamente de um conjunto de sensores dispostos no espaço, conectados a um processador adaptativo multicanal. O que o diferencia de um beamformer clássico é que as suas ponderações ou, mais formalmente, os seus coeficientes são variáveis e se adequam às variações do meio, através de algoritmos adaptativos.

4.2.1 Restrições lineares

Em diversas situações, não existe um sinal de treinamento para o algoritmo adaptativo. Para contornar a falta desse sinal, pode-se aplicar uma restrição linear ao filtro. Uma restrição linear consiste em um vetor **s**, contendo informações espaciais relativas a um sinal.

A figura 4.3 mostra a definição de uma restrição linear.

Figura 4.3: **Restrições lineares**

Nesse caso, uma frente de onda vinda de uma direção θ em relação à normal. O sinal chega com um atraso relativo entre cada sensor / microfone. Considera-se um sinal modulado, na forma $x(t) = X(t)e^{-j\omega t}$ incidente sobre um arranjo linear de sensores.

O sinal que chega ao primeiro sensor é o próprio $x(t) = X(t)e^{j}\omega t$. Após τ segundos, o sinal chega ao segundo sensor como $x(t+\tau) = X(t+\tau)e^{-j\omega(t+\tau)}$. Como o sinal $X(t)$ varia muito pouco dentro do intervalo τ em relação à portadora $e^{-j\omega t}$, pode-se dizer que $X(t+\tau) \approx X(t)$ e, portanto,

$$x(t+\tau) = X(t)e^{-j\omega(t+\tau)} = X(t)e^{-j\omega t}e^{-j\omega\tau} = x(t)e^{-j\omega\tau} \tag{4.2}$$

que corresponde ao sinal $x(t)$ atrasado por um um fator $e^{-j\omega\tau}$.

Então, toma-se por referência o primeiro microfone a receber o sinal. O atraso relativo em cada microfone é

$$\Delta_i = e^{-j\omega\tau_i} \tag{4.3}$$

onde i é a ordem do microfone.

O atraso τ_i pode ser calculado para cada sensor conforme a figura 4.4:

Figura 4.4: **Atraso** τ

Tem-se que

$$\tau_i = \frac{(i-1)\omega dsen(\theta)}{c} \tag{4.4}$$

sendo c a velocidade da frente de onda, d a separação entre os sensores e i a ordem do sensor.

Logo, compõe-se o vetor **s**, para M sensores linearmente dispostos.

$$\mathbf{s} = [1 \quad e^{-j\frac{\omega dsen(\theta)}{c}} \qquad e^{-j(M-1)\frac{\omega dsen(\theta)}{c}}]^T \tag{4.5}$$

Para cada restrição linear, associa-se uma função de ganho f. Tal função indica o ganho associado à direção imposta pela restrição linear.

A utilização de várias restrições lineares dá origem à matriz **C**, tal que

$$\mathbf{C} = [s_1 \quad s_2 \quad s_3 \quad ... \quad s_N]^T \tag{4.6}$$

Da mesma forma, compõe-se uma matriz ganho $\mathbf{f} = [f_1 \quad f_2 \quad f_3 \quad ... \quad f_N]^T$ com as respectivas funções ganho f.

A utilização de restrições implica impor um hiperplano à superfície parabolóide do item 3.1.1, definido por $\mathbf{C}^H\omega = \mathbf{f}$, de tal forma que os coeficientes sempre pertençam a esse hiperplano, tal como mostra a figura 4.5. $\mathbf{C}^H$ é a matriz hermitiana de $\mathbf{C}$.

Figura 4.5: **Superfície MSE com restrições lineares**

Então, o objetivo de um algoritmo com restrições lineares é minimizar a função objetiva de tal forma que os coeficientes estejam contidos dentro do hiperplano imposto.

$$\omega = \omega\, min[\xi_\omega(k)] \qquad sujeito\ a\ \mathbf{C}^H\omega = \mathbf{f} \tag{4.7}$$

4.3 Conformação de feixes adaptativo na forma direta

4.3.1 Conformação de feixes ótima com restrições lineares

A figura 4.6 mostra a configuração de um beamformer com restrições lineares em um arranjo de M sensores,p restrições lineares e ordem N.

Para esse beamformer, $y(k) = \omega^H(k)\mathbf{x}(k)$ é a saída, onde

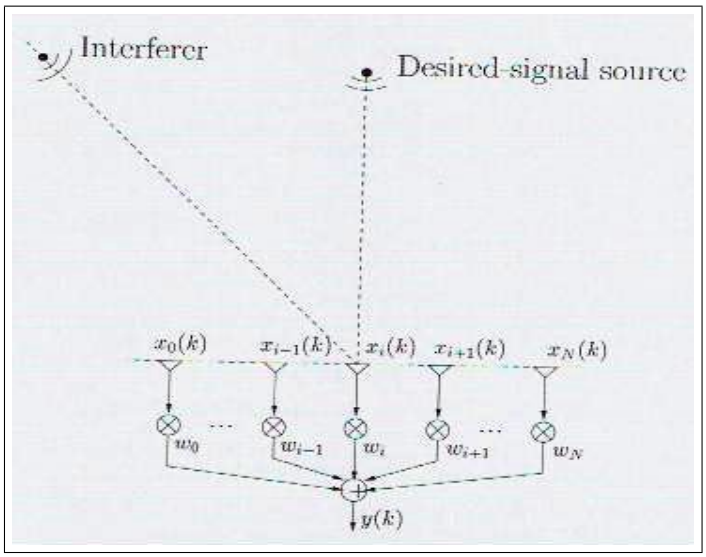

Figura 4.6: **Visão geral de um beamformer com restrições lineares**

$$\begin{aligned} \omega(k) &= [\omega_1^T(k) \quad \omega_2^T(k) \quad ... \quad \omega_M^T(k)]^T \\ \mathbf{x}(k) &= [\mathbf{x}_1^T(k) \quad \mathbf{x}_2^T(k) \quad ... \quad \mathbf{x}_M^T(k)]^T \\ \mathbf{x}_i^T(k) &= [\mathbf{x}_i(k) \quad \mathbf{x}_i(k-1) \qquad \mathbf{x}_i(k-N+1)]^T \end{aligned} \tag{4.8}$$

e $\mathbf{C}$ é uma matriz $MNxp$ e $\mathbf{f}$ é um vetor $px1$, assumido que o sinal é estocástico com média nula

De forma análoga à definição do item 3.2, utilizando a função objetiva relacionada ao sinal de erro e$(k) = d(k)y(k)$ e $\xi_\omega(k) = E[|e(k)|^2]$, obtém-se um resultado ótimo

$$\omega_{opt} = \mathbf{R}^{-1}\mathbf{p} + \mathbf{R}^{-1}\mathbf{C}(\mathbf{C}^H\mathbf{R}^{-1}\mathbf{C})^{-1}(\mathbf{f}\mathbf{C}^H\mathbf{R}^{-1}\mathbf{p}) \tag{4.9}$$

com $\mathbf{R} = E[\mathbf{x}(k)\mathbf{x}^H(k)]$ e $\mathbf{p} = E[d*(k)\mathbf{x}(k)]$. No entanto, na ausência de um sinal de treinamento $d(k)$, objeto deste estudo, $d(k) = 0$ e $\mathbf{p} = \mathbf{0}$, então, a solução ótima se reduz para

$$\omega_{opt} = \mathbf{R}^{-1}\mathbf{C}(\mathbf{C}^H\mathbf{R}^{-1}\mathbf{C})^{-1}\mathbf{f} \tag{4.10}$$

Esta solução é conhecida por LCMV (Linearly-constrained minimum variance), onde a função objetiva apresenta menor energia de saída, $\xi_\omega(k) = \omega^H\mathbf{R}\omega$.

4.3.2 Algoritmo Constrained Least-Mean-Square (CLMS)

Para se implementar 4.9, a forma mais robusta, de baixa complexidade computacional, de estabilidade comprovada e por conseqüência a mais utilizada é a forma proposta por Frost em 1972, o algoritmo conhecido por CLMS ou algoritmos de Frost. Tal algoritmo re quer tão somente informações sobre o ângulo de chegada da frente de onda e a banda de freqüências do sinal de interesse.

Através do método de aproximação por gradiente e utilizando multiplicadores de Lagrange para minimizar a função $\xi_\omega(k) = |e(k)|^2$, sujeita a $\mathbf{C}^H\omega = \mathbf{f}$, chega-se à equação de update

$$\omega(k+1) = \mathbf{P}[\omega(k) + \mu e^*(k)\mathbf{x}(k)] + \mathbf{F} \tag{4.11}$$

onde $\mathbf{P}$ é uma matriz que projeta $\omega(k)$ sobre o hiperplano definido por $\mathbf{C}^H\omega = 0$, conhecida como matriz de projeção e definida por

$$\mathbf{P} = \mathbf{I} - \mathbf{C}(\mathbf{C}^H\mathbf{C})^{-1}\mathbf{C}^H \tag{4.12}$$

μ é o passo, responsável pelo controle da estabilidade e da velocidade de convergência e $\mathbf{F}$ é um vetor tal que traz que traz a solução projetada por $\mathbf{P}$ novamente para sobre o hiperplano $\mathbf{C}^H\omega = \mathbf{f}$, dado por

$$\mathbf{F} = \mathbf{C}(\mathbf{C}^H\mathbf{C})^{-1}\mathbf{f} \tag{4.13}$$

Ainda, o vetor $\mathbf{F}$ pode ser interpretado como o vetor inicial de coeficientes, ω_0, ou vetor quiescente [Chern].

A figura 4.7 ilustra a atuação de $\mathbf{P}$ e $\mathbf{F}$ sobre o hiperplano imposto pelas restrições.

Nessa figura, vê-se claramente que as soluções para $\omega(k)$ permanecem sobre a reta $\mathbf{C}^H\omega = \mathbf{f}$.

Durante o processo adaptativo, os coeficientes de ajustam de modo a progressivamente incorporar as estatísticas do ruído vindo da direção de chegada para o qual ele está especificado, dentro da banda de freqüências de interesse.

Além disso, sua estabilidade computacional é garantida, podendo ser aplicável para implementações cujos processamentos são demorados, sem que o resultado se desvie dos

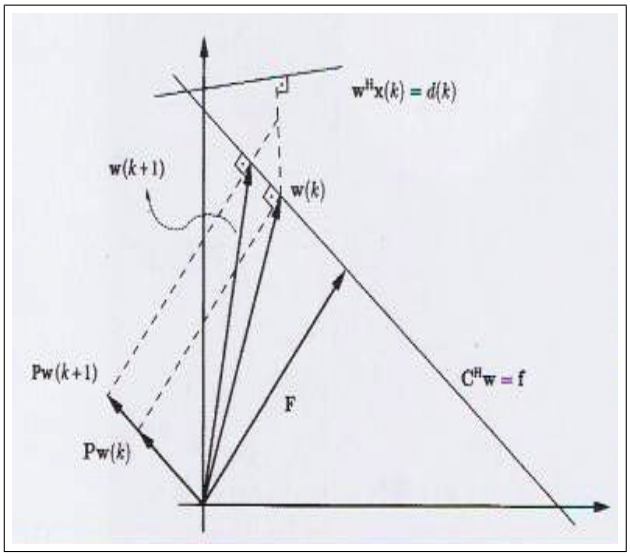

Figura 4.7: **Hiperplanos para o CLMS**

impostos pelas restrições devido a erros de truncamento. Então, o algoritmo de Frost encontra aplicações significantes em processos de medidas e processamento de sinais em sonares, arrays de antenas eletromagnéticas e nas geociências.

Em contrapartida, a estabilidade é conseguida desde que o passo esteja dentro da faixa ótima

$$0 < \mu < \frac{2}{3tr(\mathbf{R})} \tag{4.14}$$

e ainda sob o custo de velocidades de convergência mais baixas para sinais altamente correlacionados. Seu desempenho não é garantido em ambientes não estacionários.

De forma resumida, o algoritmo CLMS pode ser descrito de acordo com o quadro 2:

Quadro 3: Algoritmo de *Frost*

Algoritmo CLMS

$\omega(0) = F$

Para k ≥ 0

$e(k) = d(k) - \omega^T(k)\mathbf{x}(k)$

$\omega(k+1) = \mathbf{P}[\omega(k) + \mu e^*(k)\mathbf{x}(k)] + \mathbf{F}$

4.3.3 Algoritmo Constrained Recursive Least-Squares (CRLS)

Comparativamente ao algoritmo de *Frost*, o CRLS apresenta uma convergência rápida, independentemente da matriz de autocorrelação do sinal de entrada. Para compensar a necessidade de inversão da matriz $\mathbf{R}$, o valor de $\mathbf{R}^{-1}(k)$ é computado recursivamente. Entretando, este algoritmo pode divergir para um grande número de repetições, devido a erros e aproximações numéricas e a complexidade computacional é maior em cada iteração. A utilização de uma forma decomposta, tal como a GSC-RLS é uma alternativa que visa estabilizar o algoritmo RLS.

4.4 Formas decompostas de conformação de feixes

A idéia de utilizar formas alternativas para a estruturação de um beamformer é melhorar a performance computacional, aproveitando a performance do algoritmo. Decompor, neste caso, significa transformar um problema com restrições em outro sem restrições e com menos parâmetros, através de matrizes de transformação.

4.4.1 Generalized Sidelobe Canceller - GSC

Uma estrutura GSC, representada na figura 4.8 resolve o problema apresentado em 4.9 dividindo o vetor de coeficientes ω em duas componentes ortogonais, uma adaptativa, ω_a, e outra não adaptativa, ω_c.

Nesse caso,

$$\omega(k) = \omega_c - \mathbf{B}\omega_a(k) \tag{4.15}$$

Convém falar que ω_c é o vetor quiescente, $\mathbf{F}$, e $\mathbf{B}$ é a *blocking matrix*. Da definição de $\mathbf{F}$, no item 4.3.2, pode-se concluir que

$$\omega_c = \mathbf{C}(\mathbf{C}^H\mathbf{C})^{-1}\mathbf{f} \tag{4.16}$$

o que implica que ω_c simplesmente se assegura de que as equações das restrições são impostas.

$\mathbf{B}$, no entanto, pode ser qualquer matriz de posto cheio que satisfaça

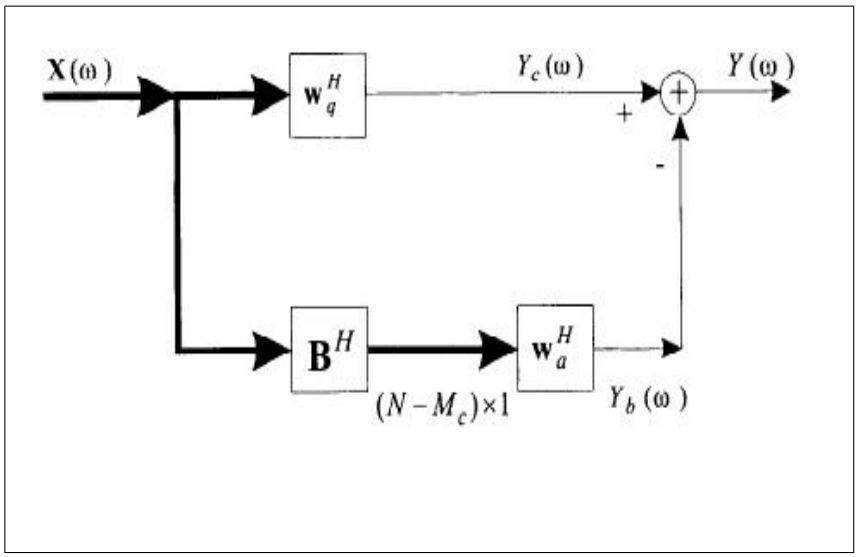

Figura 4.8: **Estrutura GSC**

$$\mathbf{B}^H\mathbf{C} = 0 \tag{4.17}$$

de modo que $\mathbf{C}^H\omega = \mathbf{C}^H(\omega_c - \mathbf{B}\omega_a) = \mathbf{C}^H\omega_c - \mathbf{C}^H\mathbf{B}\omega_a = \mathbf{f}$.

A desvantagem da estrutura GSC é que a mesma necessita da determinação a priori da matriz **B**. A matriz **B** influencia diretamente sobre a complexidade computacional do GSC. Comumente, é implementada via decomposição da matriz **C**, via *Single Value Decomposition* (SVD) ou decomposição QR. O método de SVD é uma fatoração de uma matriz retangular em N matrizes elementares; já a decomposição QR consiste na decomposição de uma matriz em outras duas, uma ortogonal e outra triangular, ou seja, $\mathbf{A} = \mathbf{QR}$, com **Q** uma matriz ortogonal e **R** uma matriz retangular.

Os comandos em Matlab¨ para a decomposição da matriz C encontram-se no apêndice 3.

A solução para o problema da parte adaptativa, ω_a, pode ser resolvido tanto por um algoritmo LMS quanto por um algoritmo RLS. A diferença, no entanto é que para um algoritmo LMS, os transientes da abordagem direta e o da abordagem por estrutura GSC somente serão iguais na condição de **B** ser unitária, ou seja,$\mathbf{B}^H\mathbf{B} = \mathbf{I}$. A escolha de uma matriz unitária para**B**, nesse caso, não assegura nenhuma forma de redução da complexidade computacional da multiplicação $\mathbf{Bx}(k)$, o que pode tornar a magnitude da filtragem da mesma ordem, ou maior, do algoritmo. Entretanto, para um algoritmo RLS,

a complexidade computacional do update dos coeficientes é pelo menos da mesma ordem da multiplicação $\mathbf{Bx}(k)$. Isso é um indício de que a implementação de algoritmos RLS sobre estruturas GSC é desejável sobre algoritmos LMS.

Os quadros 4 e 5 apresentam, respectivamente, a estrutura GSC sobre algoritmo LMS (GSC-LMS) e sobre algoritmo RLS (GSC-RLS).

Quadro 4: Algoritmo GSC-LMS

Algoritmo GSC-LMS

$\omega_a(0) = [0 \quad 0 \quad ... \quad 0]^T$

Para k ≥ 1

$y_c(k) = \omega_c \mathbf{C}^H \mathbf{u}(k)$

$\mathbf{u}(k) = \mathbf{B}^H \mathbf{x}(k)$

$y_a(k, k-1) = \omega_a^H(k-1)\mathbf{u}(k)$

$e(k, k-1) = (\omega_\mathbf{c} - \mathbf{B}\omega_\mathbf{a}(k-1))\mathbf{x}^H(k)$

$\omega_a(k+1) = \omega_a(k) + \mu \mathbf{u}(k) e^*(k, k-1)$, *com* μ *pequeno*

Quadro 5: Algoritmo GSC-RLS

Algoritmo GSC-RLS

$\mathbf{R}^{-1}(0) = \delta^{-1}\mathbf{I}$, δ *uma constante positiva e pequena*

$0 \leq \lambda < 1$, λ *e o fator de esquecimento*

$\omega_a(0) = [0 \quad 0 \quad ... \quad 0]^T$

Para k ≥ 1

$y_c(k) = \omega_c \mathbf{C}^H \mathbf{u}(k)$

$\mathbf{u}(k) = \mathbf{B}^H \mathbf{x}(k)$

$y_a(k, k-1) = \omega_a^H(k-1)\mathbf{u}(k)$

$\kappa(k) = \frac{\mathbf{R}^{-1}(k-1)\mathbf{u}(k)}{\lambda + \mathbf{u}^H(k)\mathbf{R}^{-1}(k-1)\mathbf{u}(k)}$

$e(k, k-1) = (\omega_\mathbf{c} - \mathbf{B}\omega_\mathbf{a}(k-1))\mathbf{x}^H(k)$

$\omega_a(k) = \omega_a(k-1) + \kappa(k) e^*(k, k-1)$

$\mathbf{R}^{-1}(k) = \lambda^{-1}\mathbf{R}^{-1}(k-1) - \lambda^{-1}\kappa(k)\mathbf{u}^H(k)\mathbf{R}^{-1}(k-1)$

4.4.2 Household-Transform

O Household-transform é uma forma alternativa à estrutura do Generalized Sidelobe Canceller, podendo ser analisada como uma implementação eficiente de uma estrutura GSC com matriz B unitária. A idéia básica é aplicar uma transformação unitária no vetor de entrada de forma a transformá-lo em duas componentes, uma superior não adaptativa e outra inferior adaptativa, como pode ser visto na figura 4.9

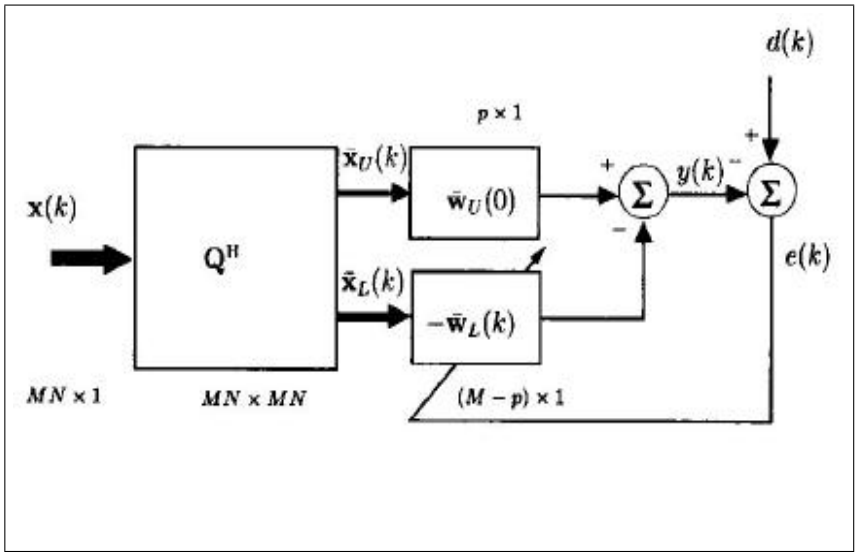

Figura 4.9: **Estrutura do HT**

Sendo $\mathbf{Q}$ uma matriz de transformação unitária de modo a gerar um vetor $\dot{\omega}(k) = \mathbf{Q}\omega(k)$, se aplicada ao vetor de entrada $\mathbf{x}(k)$, gerando $\dot{\mathbf{x}}(k) = \mathbf{Q}\mathbf{x}(k)$, a saída $y(k)$

$$y(k) = \dot{\omega}^H(k)\dot{\mathbf{x}}(k) = \omega^H(k)\mathbf{Q}^H\mathbf{Q}\mathbf{x}(k) = \omega^H(k)\mathbf{x}(k) \tag{4.18}$$

continua a mesma. Uma proposta para $\mathbf{Q}$ é uma matriz tal que triangulize $\mathbf{C}(\mathbf{C}^H\mathbf{C})^{-\frac{1}{2}}$ através de uma séria de transformações Householder, de forma que o vetor transformado $\dot{\omega}(k)$ se constitua de duas partes: uma superior $\omega_U(k)$, px1, não adaptativa e uma inferior, $\omega_L(k)$, (MN-p)x1, que pode ser atualizada por qualquer algoritmo adaptativo.

Essa transformação é interessante no sentido de melhorar a performance de implementacão da multiplicação $\mathbf{Q}\mathbf{x}(k)$ sobre a multiplicação matricial da estrutura GSC, através de uma série recursiva de multiplicações matriciais

$$\dot{\mathbf{x}}(k) = \mathbf{Q}_1\mathbf{Q}_2\mathbf{Q}_3...\mathbf{Q}_p\mathbf{x}(k) \tag{4.19}$$

onde cada Q_i é uma matriz MNxMN.

O quadro 4 mostra o algoritmo básico do HT-LMS.

Quadro 6: Algoritmo básico HTLMS

Algoritmo HTLMS

$\omega_U(0) =$ *primeiros elementos de* **QF**

Para k ≥ 1

$\dot{\mathbf{x}}(k) = \mathbf{Q}\mathbf{x}(k) = [\mathbf{X}_U^T(k) \quad \mathbf{X}_L^T(k)]^T$

$\dot{\omega}(k) = [\omega_U^T \quad \omega_L^T]^T$

$e(k) = d(k) - \dot{\omega}^H(k)\dot{\mathbf{x}}(k)$

$\omega_L(k+1) = \omega_L(k) + \mu e^*(k)\dot{\mathbf{x}}(k)$

Capítulo 5

Resultados experimentais

Neste capítulo, apresentar-se-ão os resultados obtidos. No item 5.1, faz-se uma descrição detalhada do experimento: a montagem do setup, os objetivos e um passo-a-passo de sua realização. Os itens 5.2 e 5.3 apresentam características complementares ao processamento dos sinais. é mostrado o tratamento do sinal e a codificação utilizada, bem como o tipo de sinal utilizado e seus porquês. Por fim, é feita uma análise dos resultados obtidos, no item 5.4.

5.1 Descrição do experimento

A figura 5.1 mostra a configuração do setup utilizado.

Os 8 microfones foram linearmente dispostos sobre um suporte acrílico (figura 5.2).

A saída de cada microfone é conectada a um cabo com conector XLR, cuja outra extremidade é conectada à interface firewire. Abaixo, na figura 5.3, mostra-se o painel frontal do Firepod¨ com os relativos cabos e respectivas nomeclaturas. Por conveniência de organização, os cabos foram nomeados de 1 a 8, pois a ordem dos microfones influencia no processamento dos sinais.

Observe-se que, devido ao fato dos microfones serem dinâmicos, é totalmente dispensável o uso de Phantom Power, devendo o mesmo estar desligado do Firepod¨. O uso indevido do mesmo pode corromper a cápsula de algum microfone.

As caixas de som são conectadas a um adaptador, de forma a ficar uma caixa como left e outra como right de um sinal estéreo. Isso possibilita que 2 sinais diferentes sejam

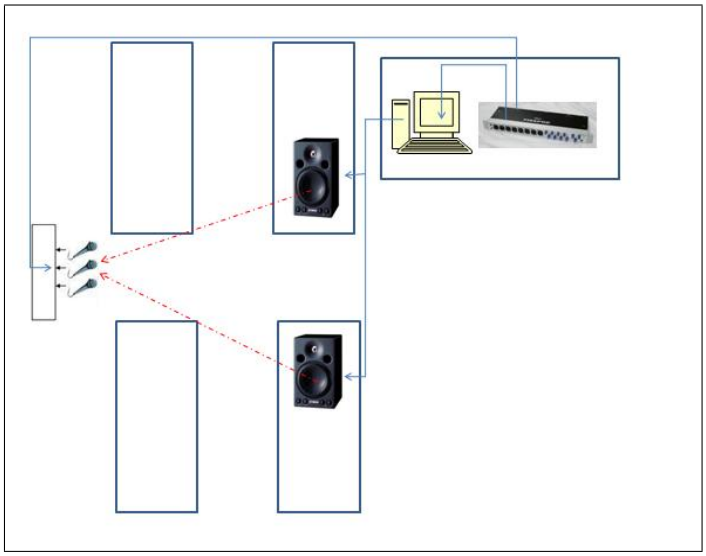

Figura 5.1: **Diagrama do setup**

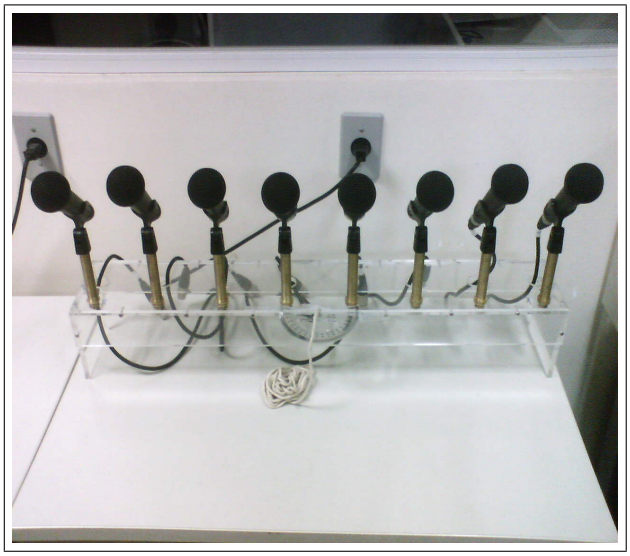

Figura 5.2: **Arranjo de microfones sobre o suporte(foto)**

Figura 5.3: **Painel frontal do Firepod**

gerados utilizando-se apenas um computador, um para um sinal de informação, outro como fonte de ruído.

O Firepod¨ é ligado ao computador via sua entrada firewire. Para que o computador possa reconhecer o dispositivo, no entanto, faz-se necessária a instalação de seu driver.

Como o processamento dos sinais realizou-se em ambiente Matlab¨, foi necessário habilitá-lo a utilizar dispositivos ASIO. A biblioteca PA_WAV foi baixada do próprio site da MathWorks (http://www.mathworks.com/) e adicionada ao Set-Path do Matlab¨. Com isso, três funções ficam disponíveis para gravar e reproduzir sons via dispositivos multicanais: pa_wavplayrecord, pa_wavplay e pa_wavrecord.

A função pa_wavplayrecord é uma função que posibilita que um determinado som seja tocado ao mesmo tempo que é gravado pela interface. É definida por

inputbuffer = pa_ wavplayrecord(playbuffer,[playdevice],[samplerate],[recnsamples], [recfirstchannel], [reclastchannel],[recdevice], [devicetype])

onde

- playbuffer : matriz a ser tocada
- playdevice: Dispositivo a ser usado para reprodução. Default: 0
- samplerate: Freqüência de amostragem. Default: 44100
- recnsamples: Número de amostras a serem gravadas. Default: 0 (caso 0, será gravado o mesmo tamanho do *playbuffer*)

- recfirstchannel: O primeiro canal a gravar. Default: 1
- reclastchannel O último canal a gravar. Default: 1
- recdevice: O dispositivo usado para a gravação. Default: 0
- devicetype: Qual driver de som usar:

    'win' Windows Multimedia Device

    'dx' DirectX DirectSound driver

    'asio' ASIO Driver (default)

Já a função pa_wavplay apenas utiliza um dispositivo para reproduzir um som. Está definida como

pa_wavplay(buffer, [samplerate], [deviceid], [devicetype])

com

- buffer: Matriz a ser tocada
- samplerate: Freqüência de amostragem. Default: 44100
- deviceid: Identificação do dispositivo de saída Default: 0
- devicetype: Qual driver de som usar

    'win' Windows Multimedia Device

    'dx' DirectX DirectSound driver

    'asio' ASIO Driver (default)

Por fim, a função pa_wavrecord utiliza um dispositivo multicanal para gravação.

inputbuffer = pa_wavrecord(firstchannel, lastchannel, nsamples, [samplerate], [deviceid], [devicetype])

onde

- firstchannel: O primeiro canal a ser gravado
- lastchannel: O último canal a ser gravado
- nsamples: O número de amostras a ravar de cada canal

- samplerate: A freqüência de amostragem. Default: 44100
- deviceid: Dispositivo de entrada Default: 0
- informat: O tipo de informação desejada no inputbuffer.
- devicetype: Qual driver de som usa

  'win' Windows Multimedia Device

  'dx' DirectX DirectSound driver

  'asio' ASIO Driver (default)

- inputbuffer: Variável que irá conter o áudio gravado. Cada canal do inputbuffer corresponde a um canal de gravação.

Montado o setup e configurado o computador, bem como com os algoritmos implementados, o array de microfones é disposto a uma distância considerável das caixas de som, digamos, 3 metros, idealmente. As caixas de som devem ser dispostas sobre um círculo com centro no centro do array e os seus respectivos ângulos em relação à normal devem ser medidos, tal como mostrado na figura 5.4.

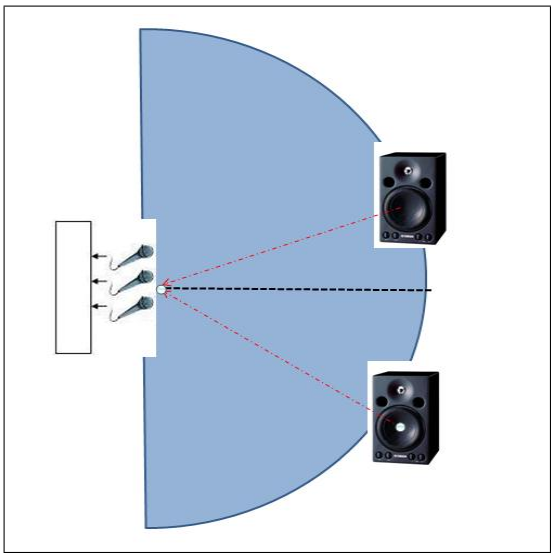

Figura 5.4: **Disposição das caixas de som**

A biblioteca PA_WAV possibilita que se gere um som ao mesmo tempo em que se grava. Isso facilita a aquisição dos sinais. Utilizando-se a função pa_wavplayrecord, gera-se tanto o sinal como o ruído ao mesmo tempo em que se grava. O apêndice 3 mostra

o algoritmo utilizado para a geração / gravação dos sinais. A partir daí, tem-se o sinais necessários para o processamento pelos algoritmos.

O objetivo deste experimento é analisar a implementação de algoritmos adaptativos com restrições lineares. Portanto, é importante conhecer a direção de chegada do som, fato que potencializa a importância da localização das caixas.

Está-se trabalhando na faixa de áudio, com algoritmos para sinais de banda estreita. Tal consideração restringe os tipos de sinais que se podem utilizar, bem como possíveis modulações. Por exemplo, a voz humana encontra-se numa faixa que vai de 300Hz a 4000Hz. é, portanto, impossível usar uma portadora com menos de 8000Hz para a modulação sem que haja perda de valores. Um sinal de música possui um range de freqüências bem maior.

Por isso, resolveu-se utilizar um tom de 1000Hz, modulado apenas em amplitude, através da modulação BASK Ð binary amplitude shift keying.

5.2 Tratamento do sinal

Assumiu-se um sinal em banda estreita modulando uma portadora com 1000Hz, de tal forma que se origina-se um sinal de áudio. Deseja-se, no entanto, que a multiplicação de um fator exponencial complexo $e^{-j\omega_c\tau}$ gere um atraso correspondente de τ. Isso nos permite utilizar o steering vector como vetor de atrasos, pois se pode aproximar $x(t) = s(t)e^{-j\omega_c(t-\tau)}$ de $x(t-\tau) = s(t-\tau)e^{-j\omega_c(t-\tau)}$, uma vez que $s(t)$ não varia consideravelmente dentro do intervalo τ.

5.2.1 Modulação BASK Ð Binary Amplitude Shift Keying

A modulação BASK consiste na utilização de um sinal $s(t)$, correspondente a um trem de pulsos que modulará uma portadora, neste caso, um tom de 1kHz, formando um sinal $x(t) = s(t)cos(2\pi 1000t)$. A figura 13 mostra o sinal $s(t)$ e seu respectivo espectro e a figura 14 mostra a portadora com o seu respectivo espectro.

A modulação $x(t) = s(t)cos(2\pi 1000t)$ leva ao seguinte espectro e forma de onda, de acordo com a figura 15:

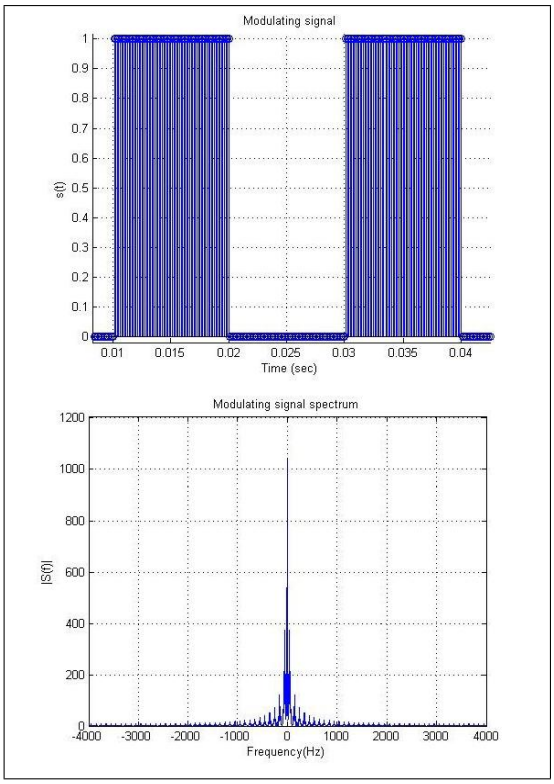

Figura 5.5: **S(t) e espectro**

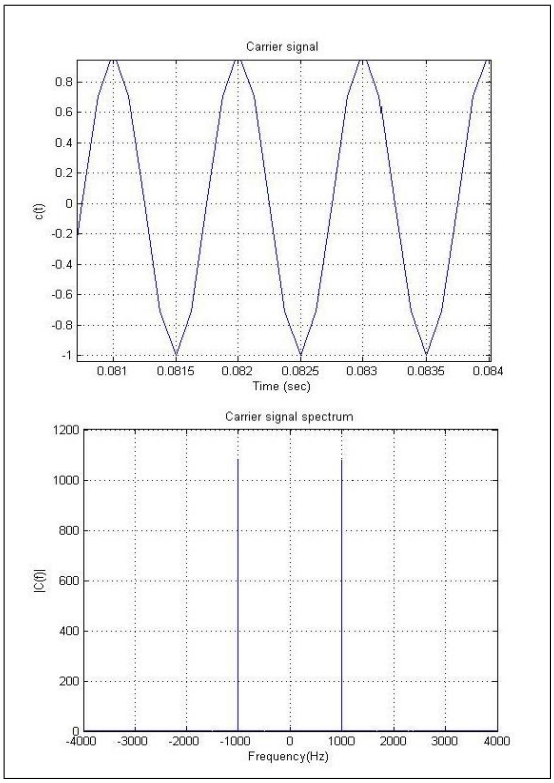

Figura 5.6: **Portadora e Espectro**

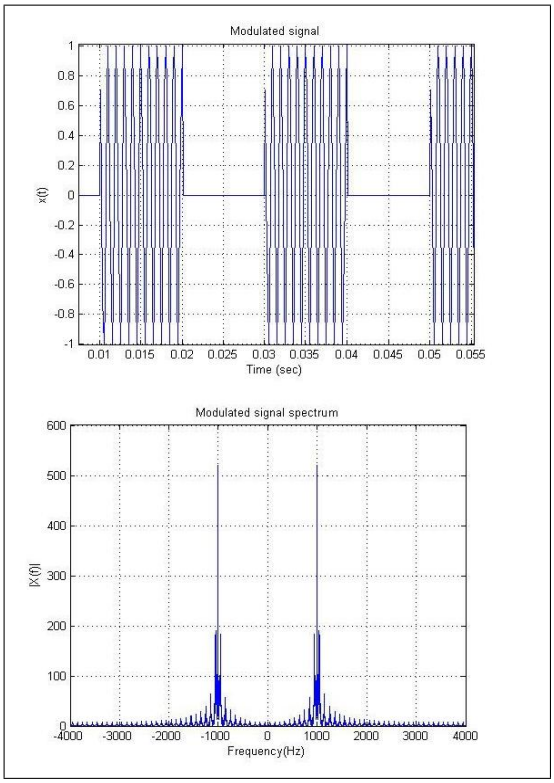

Figura 5.7: **Sinal modulado e espectro**

5.2.2 Filtragem do sinal

O sinal empregado é um sinal de áudio, portanto, um sinal real. Por isso, o seu espectro, como pode ser visto na figura 14, é simétrico em relação à origem. Da prerrogativa de que deseja-se um sinal que sofra um atraso quando multiplicado por uma exponencial, deve-se eliminar uma das bandas do sinal $x(t)$ a fim de adicionar uma componente imaginária ao sinal.

Então, após gravados os sinais, eles passaram por uma filtragem passa-banda, de acordo com a máscara ilustrada na figura 16.

Com isso, o novo espectro do sinal é o ilustrado na figura 17:

Isso valida a possibilidade de uso das restrições lineares.

5.2.3 Codificação empregada

A utilização da modulação BASK sobre uma portadora de 1kHz gera uma seqüência de bursts na freqüência da portadora, que assemelham-se ao código Morse. Dessa forma,

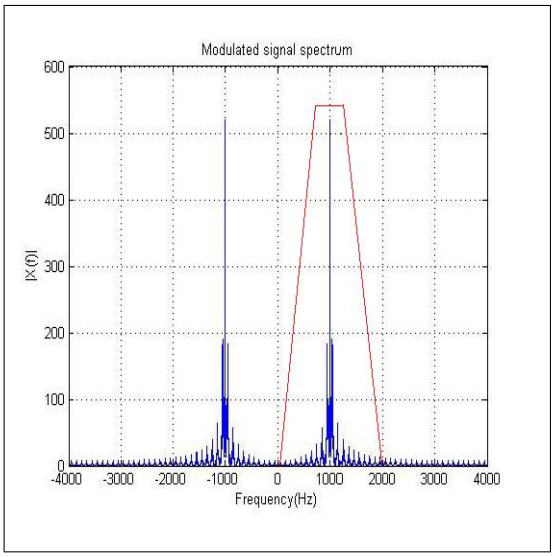

Figura 5.8: **Filtro passa banda**

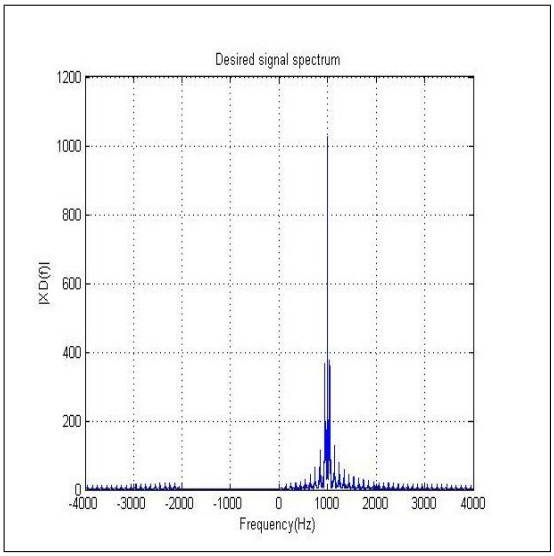

Figura 5.9: **Espectro do sinal filtrado**

decidiu-se utilizar uma codificação de forma a transmitir alguma informação. Empregou-se a seguinte codificação:

Para cada algarismo de uma frase, incluindo espaços e pontuação, associou-se uma palavra de 5 bits, de tal forma que não houvesse 2 palavras idênticas. Cada bit 1 corresponderia a um pulso de 10 comprimentos de onda com amplitude unitária, assim como cada bit 0 corresponderia a uma pausa dos mesmos 10 comprimentos de onda.

Os pulsos foram alocados serialmente e transmitidos. A tabela com as correspondências encontra-se nos apêndices.

5.2.4 Algoritmos utilizados

Após a montagem do sistema, o processo de filtragem, propriamente dito, foi dividido em 4 partes:

1 Codificação e modulação do sinal

2 Geração e aquisição

3 Tratamento

4 Processamento

A idéia era a de criar um código com interface e com indicações de etapas. De início, o código solicita uma mensagem de até 100 caracteres para ser codificada e servir de sinal de informação. A mensagem utilizada para testes foi 'mensagem em bask '.

Codificação e modulação do sinal Para a codificação, decidiu-se utilizar letras minúsculas e pontuação. Para tanto, necessitou-se a utilização de 5 bits para cada caractere. A mensagem codificada gera um formato de onda modulante de acordo com a figura 18:

Com esse sinal modulante, criou-se uma portadora senoidal de 1000Hz para ser modulada.

Geração e aquisição O sinal contendo a informação é gerado pelo próprio Matlab através do comando 'soundsc' e tem duração aproximada de 4 segundos.

Ao mesmo tempo, utiliza-se a função *pa_wavrecord* para se gravar o sinal. Esse comando é configurado para gravar 8 canais através do dispositivo ' asio'. Após gravados,

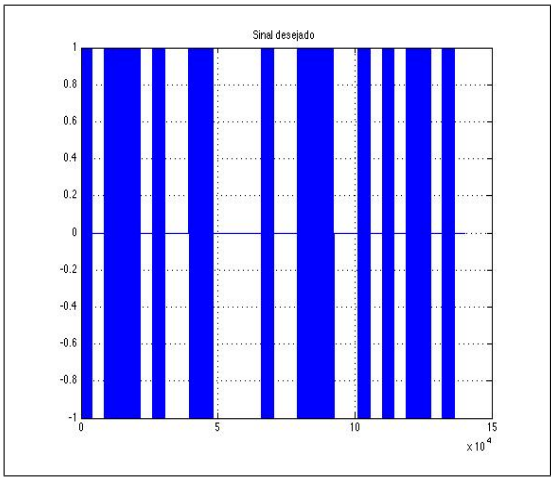

Figura 5.10: **Mensagem codificada**

os sinais são separados do *inputbuffer*, instanciando-se 8 vetores para receber cada um dos 8 canais bufferizados.

Nesse momento, pode-se tanto gravar os sinais como arquivos .wav, bem como utilizá-los de imediato para tratamento e processamento.

A tabela 5.1 mostra os comandos usados para a geração e gravação do sinal.

Tratamento O tratamento do sinal , abordado na seção 5.2 é realizado através da função 'filter(b, [1], sinal de entrada)', onde b são coeficientes de um filtro FIR centrado na origem e deslocado para a freqüência de 1000Hz, de acordo com a tabela 5.2.

Após filtrados os sinais, eles são transpostos e concatenados numa matriz
$\mathbf{X} = [x1; \quad x2; \quad x3; \quad x4; \quad x5; \quad x6; \quad x7; \quad x8];$

Processamento A iniciação do processamento se dá pela definição dos parâmetros que serão utilizados.

- Freqüência de amostragem: fs = 44100Hz
- Freqüência da portadora: 1000Hz
- Velocidade do som no ar: v = 340.29 m/s
- Distância entre os microfones: d = 0.1 m
- Ângulo da fonte do sinal de informação: θ_{info} = 17 graus

Tabela 5.1: Geração e aquisição

```
% Gravando os sinais
disp('Gravando os sinais...')
disp('...ON AIR...')
disp(' ')
tempo = length(modulado)*ts;
nsamples=fs*tempo; % Number os samples
% Gerando os sinais
soundsc(modulado,fs)
inputbuffer = pa_wavrecord(1, 8, nsamples, fs, 2, 'asio');
u1=inputbuffer(:,1);
u2=inputbuffer(:,2);
u3=inputbuffer(:,3);
u4=inputbuffer(:,4);
u5=inputbuffer(:,5);
u6=inputbuffer(:,6);
u7=inputbuffer(:,7);
u8=inputbuffer(:,8);
disp('Gravação ok!')
disp(' ')
% Salvando os sinais
wavwrite(u1,fs,'u1')
wavwrite(u2,fs,'u2')
wavwrite(u3,fs,'u3')
wavwrite(u4,fs,'u4')
wavwrite(u5,fs,'u5')
wavwrite(u6,fs,'u6')
wavwrite(u7,fs,'u7')
wavwrite(u8,fs,'u8')
disp('Sinais salvos na pasta corrente.')
disp(' ')
```

Tabela 5.2: Geração e aquisição

```
% Filtering
b1=fir1(100,0.5);
b1=b1.';
n=(0:(length(b1)-1)).';
b=b1.*exp(j*0.5*pi*n);
x1 = 2*filter(b,[1],u1);
x2 = 2*filter(b,[1],u2);
x3 = 2*filter(b,[1],u3);
x4 = 2*filter(b,[1],u4);
x5 = 2*filter(b,[1],u5);
x6 = 2*filter(b,[1],u6);
x7 = 2*filter(b,[1],u7);
x8 = 2*filter(b,[1],u8);
x1 = transpose(x1);
x2 = transpose(x2);
x3 = transpose(x3);
x4 = transpose(x4);
x5 = transpose(x5);
x6 = transpose(x6);
x7 = transpose(x7);
x8 = transpose(x8);
X = [x1; x2; x3; x4; x5; x6; x7; x8]; % Input matrix
```

- Ângulo da fonte do sinal de ruído: θ_{noise} = -33 graus
- Comprimento de onda: λ = 0.34029 m
- Passo: μ = 0.1
- Fator de esquecimento: λ_{RLS} = 0.999

A inicialização é feita com os comandos de Matlab descritos na tabela 5.3.

Tabela 5.3: Parâmetros

```
fs = 44100; ts = inv(fs);
fc = 1000; wc = 2*pi*fc; tc = inv(fc);
vsnd = 340.29; % Velocity of sound at sea level in m/sec
d = 0.1; % Distance between the microphones in meters
anglex = 17; % Angle of the signal in degrees
angley = -32; % Angle of the noise signal in degrees
len = length(x1);
disp(['Serão realizadas ' num2str(len) ' iterações'])
mi = 0.1;
lambda = 0.34029;
fix = 2*pi*d*sin(2*pi*anglex/360)/lambda;
fiy = 2*pi*d*sin(2*pi*angley/360)/lambda;
for i=1:8
    sx(i) = exp(-j*(i-1)*fix);
    sy(i) = exp(-j*(i-1)*fiy);
end
sx = transpose(sx);
sy = transpose(sy);
C = horzcat(sx,sy);
f = [1; 0];
P = eye(8) - C*inv(C'*C)*C';
F = C*inv(C'*C)*f;
tvet = 0:tc:len*tc-tc;
```

Observe-se que são especificados os parâmetros comuns a todos os algoritmos. Observe-se também que sx e sy são as restrições lineares relativas ao sinal de informação e a ruído, respectivamente. Contatenadas, elas formam a matriz **C**, com a qual se chegará na matriz **P** e na matriz **F**.

Os sinal são então submetidos a filtragem nos algoritmos de *Frost*, GSC-LMS, HTLMS e GSC-RLS e os resultados são plotados em gráficos.

5.3 Resultados obtidos

Deve-se considerar o ambiente em que o experimento foi realizado para se avaliar a performance dos algoritmos implementados. O meio acústico introduz distorções e dispersão dos pulsos do sinal de informação. Paredes não aborventes geram reflexões consecutivas que atuam como interferidores parasitas. A presença de outros elementos no meio também interferem nas características do ambiente, tornando-o não estacionário.

Este experimento foi realizado em um ambiente não estacionário, com forte presença de elementos interferidores. É de se esperar que o algoritmo GSC-RLS tenha um melhor desempenho que os demais algoritmos (da família RLS).

A idéia da escolha do processamento via tais algoritmos é, primeiramente, mostrar a similaridade dos algoritmos LMS com estruturas diferentes. Em segunda instância, mostrar a diferença de performance entre algoritmos LMS e RLS.

CLMS ou Algoritmo de *Frost*

O algoritmo de *Frost* é a forma mais convencional e direta de algoritmos LMS. A tabela 5.4 mostra os comandos em Matlab para a execução de um combinador linear com algoritmo CLMS e cujo resultado encontra-se na figura 5.11.

Tabela 5.4: Algoritmo CLMS

```
w = F;
for k=1:len
    eCLMS(k) = -w'*X(:,k);
    w = P*(w + mi*conj(eCLMS(k))*X(:,k)) + F;
end
```

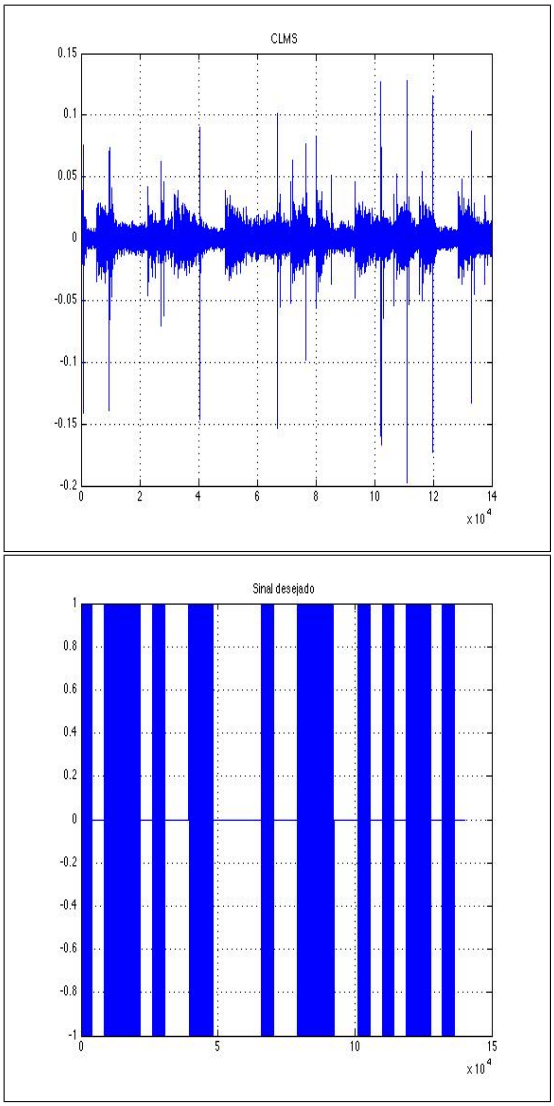

Figura 5.11: **Resultado do CLMS - 8 canais**

Como pode-se observar, existe uma grande diferença entre o sinal desejado e o sinal obtido pela filtragem. Isso era de se esperar, devido a variação constante do meio, ponto fraco dos algoritmos da família LMS.

GSC-LMS

A estrutura GSC-LMS visa ser uma forma alternativa da forma direta do algoritmo de *Frost*. Com isso, espera-se menor custo computacional e menor tempo de processamento.A sua implementação encontra-se na tabela 5.5 e seu resultado na figura 5.12.

Em um tempo de processamento de XXXXXX segundos.

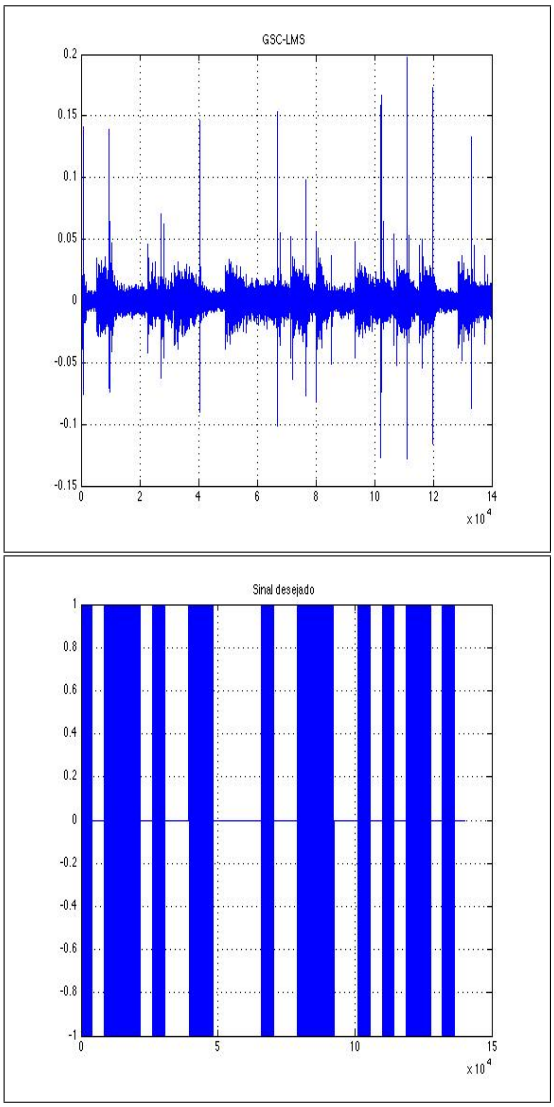

Figura 5.12: **Resultado GSC-LMS - Gravação**

Tabela 5.5: Algoritmo GSC-LMS

```
wc = F;
[Q,R] = qr(C);
B = Q(:,3:8);
wa = zeros(6,1);
for k=1: len
    yc = wc'*X(:,k);
    xa = B'*X(:,k);
    ya = wa'*xa;
    z(k) = yc - ya;
    wa = wa + mi*xa*conj(z(k));
end
```

A função qr(C) é a funcção que faz a decomposição QR da matriz C e a atribuição B = Q(:,3:8) assegura que a matriz B seja de posto cheio.

Note-se que a performance do GSC-LMS é a mesma do CLMS, dado que são algoritmos similares, alterando-se somente o tempo de processamento e a complexidade computacional.

HTLMS

O HTLMS é uma segunda forma alternativa ao CLMS. A utilização de matrizes de transformação visam reduzir ainda mais a complexidade computacional, garantir a estabilidade própria dos algoritmos LMS e mantendo a mesma performance.

A tabela 5.6 e a figura 5.13 mostram a implementação e o resultado da filtragem, respectivamente.

Como pode-se esperar, a performance foi similar ao dos algoritmos CLMS e GSC-LMS, mas num tempo de processamento de XXXXX segundos.

GSC-RLS

A utilização de um algoritmo RLS melhora a performance em ambientes não estacionários. A tabela 5.7 mostra a implementação do algoritmo GSC-RLS e a figura 5.14 mostra o seu

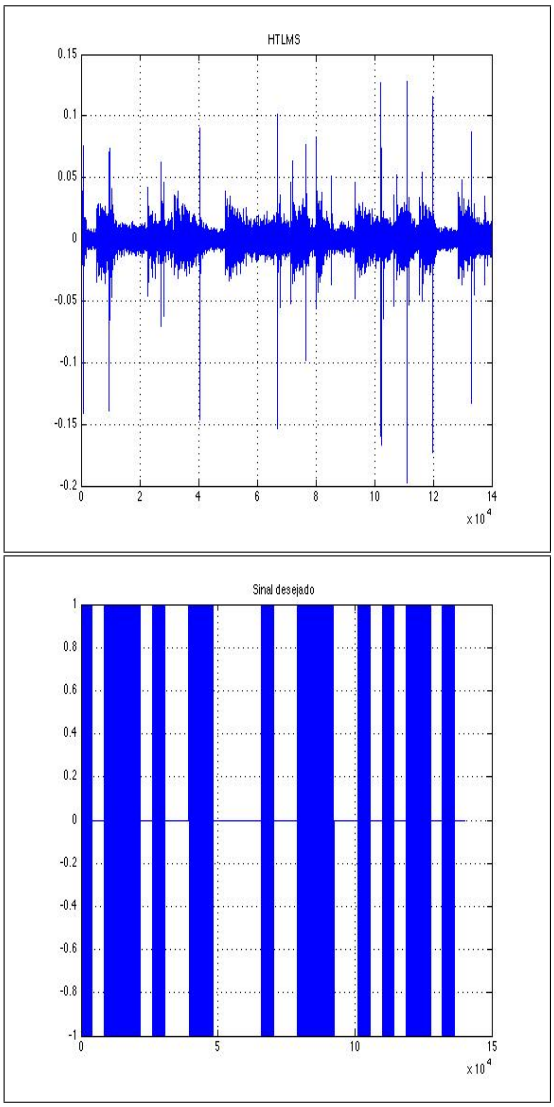

Figura 5.13: **Resultado do HTLMS - Gravação**

Tabela 5.6: Algoritmo HTLMS

```
A = X*inv(sqrt(C'*C));
[V,Q] = housvec(A);
wu = Q*F;
wu = wu(1:2);
wl = zeros(6,1);
for k=1: len
    xb = Q*X(:,k);
    xu = xb(1:2);
    xl = xb(3:8);
    wb = [wu; wl];
    eHT(k) = -wb‘*xb;
    wl = wl + mi*conj(eHT(k))*xl;
end
```

resultado.

Como pode-se observar, o sinal obtido assemelha-se ao sinal de informação e está menos corrompido pelas variações no meio.

em um tempo de processamento de XXXX segundos.

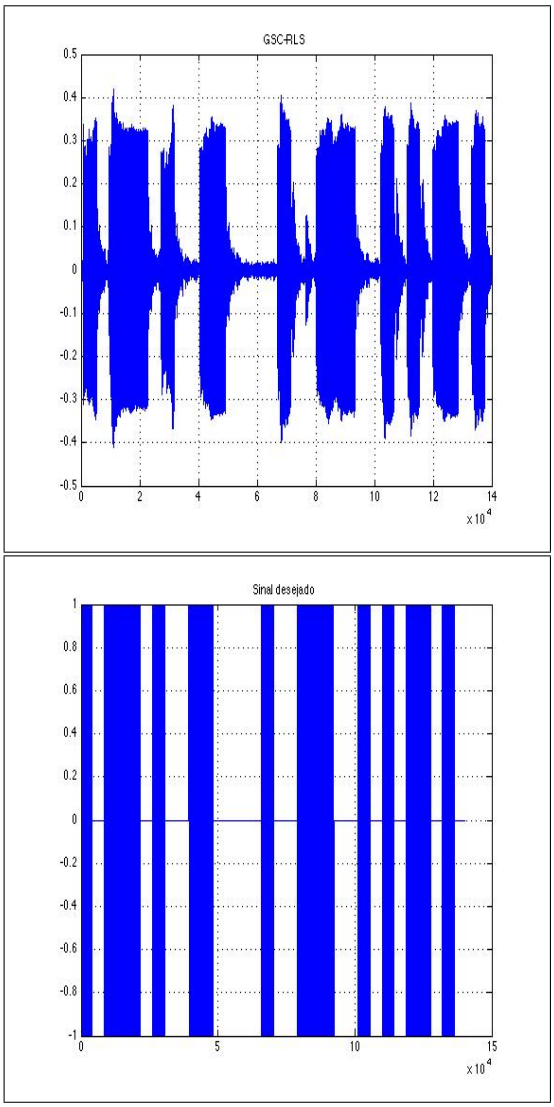

Figura 5.14: **Resultado do GSC-RLS - Gravação**

Tabela 5.7: Algoritmo GSC-RLS

```
delta = 0.000001;
lam = 0.99;
R = inv(delta)*eye(6);
wcRLS = F;
[QRLS,RRLS] = qr(c);
BRLS = QRLS(:,3:8);
waRLS = zeros(6,1);
for k=1: len
    ycRLS = wcRLS'*X(:,k);
    xaRLS = BRLS'*X(:,k);
    yaRLS = waRLS'*xaRLS;
    ka = R*xaRLS/(lam + xaRLS*R*xaRLS);
    R = lam*R - inv(lam)*ka*xaRLS'*R;
end
```

Capítulo 6

Conclusão

6.1 Organização do projeto

O trabalho neste projeto estava planejado em quatro etapas, divididas em nove atividades principais.

A primeira etapa consistia da fase de estudo do conteúdo necessário para a implementação dos algoritmos, bem como dos equipamentos, *softwares*, bibliotecas e literatura disponível. Faziam parte dessa etapa as seguintes atividades:

- Atividade 1: Pesquisa bibliográfica;
- Atividade 2: Estudo dos equipamentos;
- Atividade 3: Estudo e concepção dos algoritmos;
- Atividade 5: Estudo para o caso banda-larga.

A segunda etapa era a etapa de implementação dos algoritmos concebidos e estudados nos estudos, resumindo-se somente a essa atividade.

A terceira etapa consistia na aquisição de dados e em teste dos dados com os algoritmos implementados. Estavam previstas para essa etapa as seguintes atividades

- Atividade 6: Aquisição de dados;
- Atividade 7: Teste com os dados obtidos;
- Atividade 8: Teste em outros ambientes.

Por fim, a quarta etapa consistia da conclusão do projeto, o que incluia a preparação de apresentações e relatório, bem como desta monografia.

O planejamento mensal inicialmente proposto está apresentado na tabela 6.1.

Tabela 6.1: Planejamento

AT	AGO	SET	OUT	NOV	DEZ	JAN	FEV	MAR	ABR	MAI	JUN	JUL
AT1	X	X	X	X	X							
AT2	X	X										
AT3				X	X	X	X	X				
AT4					X	X	X	X				
AT5					X	X	X	X	X	X		
AT6								X				
AT7									X			
AT8										X		
AT9											X	X

Entretanto, observou-se que o estudo e a implementação dos algoritmos seriam atividades concorrentes e paralelas e mesmo durante os períodos de teste, deveriam ser atividades presentes. Da mesma forma, foi observado que o estudo de algoritmos para o caso de banda-larga deveria ficar em segundo plano, dado o objetivo inicial do projeto.

Das atividades propostas, tanto o estudo do caso banda-larga (atividade 5), quanto os testes em outros ambientes (atividade 8) não foram realizados. O caso banda-larga foi visto apenas *en passant*, a título de curiosidade e como uma visão geral.

Houve uma restrição grande quanto à aquisição dos sinais para testes. A necessidade de trabalhar em banda estreita com sinal de áudio levou às considerações citadas na seção 5.1. A utilização do sinal descrito anteriormente restringiu, inclusive o ambiente de teste. Não podendo ser testados com sinais tais como voz humana, músicas etc, os testes em outros ambientes se inviabilizam.

As atividades foram realizadas, então, de acordo com a tabela 6.2.

Tabela 6.2: Andamento do projeto

AT	AGO	SET	OUT	NOV	DEZ	JAN	FEV	MAR	ABR	MAI	JUN	JUL
AT1	X	X	X	X	X							
AT2	X	X										
AT3				X	X	X	X	X				
AT4					X	X	X	X	X	X	X	
AT6								X	X	X	X	
AT7									X	X	X	
AT9											X	X

6.2 Dos resultados obtidos

Em *latu senso* os objetivos sumários propostos foram alcançados. Foram implementados algoritmos com restrições lineares, em estruturas e famílias variadas e suas performances e resultados foram obtidos e comparados. Dada a situação de testes, pode-se comprovar a melhor eficácia de algoritmos RLS frente algoritmos LMS (meio não estacionário), bem como pode-se ver a melhor performance de estruturas decompostas frente a estruturas diretas, de uma mesma família, no caso o LMS.

Este trabalho está disposto de forma a ser usado como referência para futuros projetos finais e/ou trabalhos na área de Combinação linear adaptativa, pois reune em um mesmo papel tanto a base da teoria, bem como a parte prática, bem como uma literatura sugerida e uma ordem lógida de estudo. A implementação física do sistema necessário para testes é um grande obstáculo a ser contornado e cuja literatura encontra-se esparsa por diversos meios de comunicação. Ainda, os equipamentos aqui mostrados e citados são sugeridos para eventuais pesquisas na área, pois apresentaram bom desempenho e relação custo/benefício.

6.3 Propostas de futuros trabalhos

Vista a teoria de conformação de feixes para o caso banda estreita em ambiente Matlab, sugerem-se dois passos futuros, para desenvolvimendo:

- Melhoria de performance dos algoritmos implementados. Partindo-se do caminho de

sair do ambiente Matlab e procurando o desenvolvimento em plataforma específica, mesmo em uma FPGA. Para tanto, seria necessária a interdisciplinariedade com a cadeira de arquitetura de computadores e eletrônica digital, bem como com a cadeira de programação. Ainda, pode-se seguir pelo caminho do estudo de combinadores lineares adaptativos robustos.

- Desenvolvimento de algoritmos para o caso banda-larga. A divisão de um sinal de banda larga em muitos sinais de banda estreita e o processamento no domínio da freqüência iriam requerer uma excelente performance computacional, maior entrave na implementação desses algoritmos.

Desenvolvidas essas duas atividades, os trabalhos tomaria lugar em muitas aplicações comerciais, como video-conferências, análises sísmicas, perícia forense e segurança.