ANDRÉ HORÁCIO CAMARGO CAREZIA

REDUÇÃO ADAPTATIVA DE ECO E DE
RUÍDO PARA TERMINAIS VIVA-VOZ

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia.

São Paulo
2002

ANDRÉ HORÁCIO CAMARGO CAREZIA

REDUÇÃO ADAPTATIVA DE ECO E DE
RUÍDO PARA TERMINAIS VIVA-VOZ

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Mestre em Engenharia.

Área de concentração:
Sistemas Eletrônicos

Orientador:
Prof. Max Gerken

São Paulo
2002

Carezia, André Horácio Camargo

Redução adaptativa de eco e de ruído para terminais viva-voz.

São Paulo, 2002.

140p.

Dissertação (Mestrado)—Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Telecomunicações e Controle.

1. Cancelamento de eco acústico 2. Supressão de ruído para sinais de voz 3. Algoritmos adaptativos I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Telecomunicações e Controle II. t

Para meus pais e Ana Cândida

Sumário

1	Apresentação	1
2	Aspectos Teóricos	6
2.1	Cancelamento de Eco Acústico	7
2.1.1	As diferentes abordagens	7
2.1.2	Algoritmo LMS	10
2.1.3	Algoritmo RLS	11
2.1.4	Experimento computacional	12
2.1.5	O algoritmo LSL	15
2.2	Redução de Ruído	19
2.2.1	Banco de filtros: fundamentos	20
2.2.2	Resposta em frequência	30
2.2.3	Estrutura polifásica	35
2.2.4	Subtração espectral	39
2.2.5	Estimação estatística do ruído	42
2.2.6	Ponderação psicoacústica	43
2.3	Comentários	48
3	Implementação Prática	50
3.1	Processador SHARC	51
3.2	Cancelamento de eco acústico	54
3.2.1	Núcleo	54
3.2.2	Complexidade computacional	64
3.3	Subtração espectral do ruído	64

3.3.1	Processamento	65
3.3.2	Filtros e tabelas	67
3.4	Integração	70
3.5	Comentários	71
4	Resultados	73
4.1	Medidas em campo	73
4.2	Cancelamento de eco	74
4.2.1	Resultados	75
4.2.2	Especificações	79
4.3	Redução de ruído	80
4.3.1	Resultados	80
4.3.2	Especificações	81
4.4	Sistema integrado	82
4.5	Comentários	83
5	Conclusões	87
A	Listagens Matlab	90
A.1	Algoritmo LMS convencional	91
A.2	Algoritmo RLS convencional	92
A.3	Filtro passa-baixas (janela de Kaiser)	93
A.4	Coefficientes para os filtros	95
A.5	Tabelas de senos e cossenos	96
B	Listagens SHARC	97
B.1	Redução de ruído (em linguagem C)	98
B.2	Cancelamento de eco (Assembly)	107
C	Artigo SBT 1999	112
D	Artigo ICASSP 2001	119

Lista de Figuras

1.1	Aplicação de terminal viva-voz	2
2.1	Diagrama de blocos para o cancelador de eco acústico. . .	8
2.2	Filtro transversal.	9
2.3	Resposta impulsiva (janelada) da caixa usada no experi- mento.	13
2.4	Curvas de aprendizado para o exemplo.	14
2.5	Fluxograma do filtro em treliça utilizado no algoritmo LSL com estimação de erros de predição <i>a priori</i>	17
2.6	Diagrama de subtração espectral básica.	20
2.7	Análise espectral com banco de filtros passa-baixas . . .	21
2.8	Análise espectral com banco de filtros passa-faixa . . .	24
2.9	Síntese espectral com banco de filtros passa-baixas. . .	26
2.10	Síntese espectral com banco de filtros passa-faixa. . . .	29
2.11	Cálculo de um sistema de mínima fase usando transfor- mada de Hilbert.	34
2.12	Resposta impulsiva do filtro protótipo	35
2.13	Resposta em frequência do filtro protótipo	36
2.14	Estrutura polifásica de análise.	38
2.15	Estrutura polifásica de síntese.	39
2.16	Leis para supressão de ruído.	41
2.17	Função de espalhamento.	47
3.1	Diagrama de blocos do processador SHARC	52

3.2	Diagrama de condições para funcionamento paralelo das unidades computacionais do SHARC.	53
3.3	Resposta impulsiva do filtro protótipo	69
3.4	Resposta em frequência do filtro protótipo	70
3.5	Filtros individuais e erro de reconstrução para $M = 64$ e $r = 32$	71
4.1	Resposta impulsiva para o veículo usado nos testes.	76
4.2	Influência de λ_L no AEC.	77
4.3	Influência do ruído de medida no AEC.	78
4.4	Influência de sinal de voz local no AEC.	79
4.5	Redutor de ruído para SNR alto	81
4.6	Redutor de ruído para SNR baixo	82
4.7	Redutor de ruído para SNR variável	83
4.8	Cancelamento de eco + Redução de ruído	85
4.9	Redução de ruído + Cancelamento de eco	86

Lista de Tabelas

2.1	Algoritmo LMS convencional.	11
2.2	Algoritmo RLS convencional.	12
2.3	Algoritmo LSL convencional.	18
2.4	Algoritmo EFLSL modificado.	19
2.5	Regras para supressão de ruído.	41
2.6	Bandas críticas na faixa telefônica.	46
3.1	Grupos de registradores no SHARC.	53
3.2	Núcleo do algoritmo EFLSL modificado.	55
3.3	Núcleo do algoritmo EFLSL modificado, com equações reordenadas e identificadores atrelados aos operandos. . .	56
3.4	Representação alternativa do algoritmo EFLSL	58
3.5	Quadro mostrando o conteúdo dos registradores imediatamente <i>antes</i> de cada ciclo de execução.	60
3.6	Endereçamento para o algoritmo EFLSL	61
3.7	Diagrama de transferências de memória	62
3.8	Modificadores de índice	63
4.1	Combinações de parâmetros para avaliação geral do cancelador de eco.	75
4.2	Especificações técnicas do bloco de cancelamento de eco. .	79
4.3	Combinações de parâmetros para avaliação do redutor de ruído.	80
4.4	Especificações técnicas do bloco de redução de ruído. . . .	84

Resumo

Há um grande interesse hoje em desenvolver terminais viva-voz que permitam aos participantes de uma conversa à distância contarem com um bom grau de naturalidade e inteligibilidade.

O objetivo deste trabalho é apresentar solução para dois impedimentos que surgem quando se deseja projetar um terminal viva-voz para ser utilizado em automóveis: o eco acústico resultante do acoplamento entre microfone e alto-falante do terminal; e o ruído ambiente produzido por exemplo pelo vento, pneus e motor do veículo.

A solução proposta envolve o uso de filtros adaptativos e alterações no espectro do sinal de voz para minimizar os problemas mencionados. Os aspectos teóricos são abordados de forma breve, sem deixar no entanto que nenhum detalhe importante fique de fora. Uma implementação prática e eficiente em processador digital de sinais é um dos destaques do trabalho.

Abstract

There is currently great motivation in developing hands-free devices which offer users, engaged in a telephone conversation, a good level of naturalness and intelligibility.

In this work, the goal is to present a solution for two well-known problems that occur when designing a hands-free device for use in automobile environments: (1) the acoustic echo coupling between microphone and speaker, and (2) the background noise generated for example by wind, tires and vehicle engine.

The proposed solution includes adaptive filtering techniques and modifications in the speech signal spectrum, in order to minimize the two problems above. Theoretical issues are briefly analyzed, however the author believes no relevant detail is kept out. Highlighted in the report is a practical and efficient implementation of the algorithms in a modern digital signal processor.

Contribuições deste trabalho

- Método eficiente de implementação de seqüência de operações do tipo $ab + cd$ em processador digital de sinais com *assembly* algébrico;
- Projeto de filtros de reconstrução quase-perfeita com atraso pequeno, para uso em estrutura polifásica;
- Implementação integrada do cancelador de eco e do redutor de ruído ambiente;
- Implementação de conceitos psicoacústicos no algoritmo de subtração espectral para redução de ruído;
- Implementação de técnicas de estimação do perfil do ruído sem o uso de detectores de atividade vocal;
- Testes positivos de qualidade feitos com sinais e situações reais; o sistema aprimora a experiência de estabelecer uma comunicação móvel em automóveis.

Capítulo 1

Apresentação

Em 1982, quando o FCC¹ autorizou os serviços de telefonia móvel, uma nova indústria começou seu caminho evolutivo para a situação que se apresenta hoje: milhões de telefones móveis em operação, permitindo a comunicação de voz e dados em qualquer lugar, desde a própria casa ou escritório, nas horas de lazer e até nas ruas e dentro dos automóveis.

Quando se pensa no uso de telefones celulares em automóveis, vêm à mente as preocupações com o conforto e a segurança; por isso, há hoje um grande interesse em desenvolver terminais viva-voz que permitam aos participantes de uma conversa à distância contarem com um bom grau de inteligibilidade e naturalidade. Esses terminais *avançados* tratam de resolver ou minimizar dois problemas (ver Figura 1.1 na página seguinte):

O eco acústico resultante do acoplamento entre o alto-falante, o ambiente e o microfone; e

O ruído ambiente, do próprio motor do carro, dos pneus e do vento na janela que perturbam a voz local², reduzindo a relação sinal-ruído

¹*Federal Communications Commission*: Agência do governo americano que regula o funcionamento dos sistemas de comunicação nos EUA, além de diversos aspectos dos sistemas de comunicação no mundo todo.

²*local* e *remoto* ou *distante*: termos usados no texto para diferenciar o usuário de

Figura 1.1: Terminais viva-voz são usados em automóveis para garantir segurança e conforto ao usuário.

devido à distância entre o usuário local e o microfone do terminal viva-voz.

O objetivo desta dissertação é apresentar um sistema avançado para comunicação de voz que procure corrigir os problemas antes mencionados (e representados pelas duas linhas tracejadas na Figura 1.1).

As pessoas conversam quase sempre na presença de *ecos*, que são produzidos pelas reflexões dos sinais de voz nas paredes, objetos e janelas. Normalmente o atraso entre o sinal de voz e o eco é pequeno (da ordem de alguns poucos milissegundos), e o efeito percebido soa mais como uma reverberação. Em conversas telefônicas, no entanto, o atraso pode alcançar centenas de milissegundos, devido às várias etapas de processamento do sinal de voz, passando por cabos, equipamentos de conversão analógico-digitais e enlaces via satélite nas ligações internacionais. O eco nitidamente distinto do sinal de voz irrita quem participa de uma conversa e precisa ser eliminado para torná-la mais natural e inteligível.

um equipamento viva-voz (e.g. no interior de um veículo)—local—do seu interlocutor que está no “outro lado da linha”—remoto.

Um *filtro adaptativo* serve bem a esse propósito e vem sendo utilizado extensivamente em equipamentos de viva-voz [1, 2, 3, 4, 5], para reduzir o chamado *eco acústico*. Outro tipo de eco presente nos enlaces que envolvem telefones fixos, causado pelo descasamento de impedância entre o cabo e o aparelho, tem sido resolvido com equipamentos instalados na central telefônica [6] e não é especificamente abordado neste trabalho.

Dois são os propósitos de um sistema de redução de ruído para sinais de voz: a) diminuir o incômodo causado pela presença do ruído e b) destacar o sinal de voz para aumentar sua inteligibilidade.

Ao longo da última década, diversos autores atacaram o problema de várias formas, incluindo abordagens baseadas nos conceitos de filtragem adaptativa [7, 8, 9] e que precisam de um ou mais microfones de referência longe da fonte de voz. Também alguns usaram idéias de redes neurais e separação de sinais aplicadas ao problema de melhorar a qualidade dos sinais de voz [10, 11, 12]. Outros modelaram os sinais de áudio ou voz como processos estocásticos autorregressivos de média móvel (ARMA³) e empregaram a teoria de Bayes ou dos filtros de Kalman para estimar versões mais “limpas” dos sinais de interesse [13, 14, 15].

O algoritmo empregado neste trabalho pertence a outra classe de técnicas conhecida como *subtração espectral*, a qual alcançou popularidade no desenvolvimento de terminais viva-voz graças à sua simplicidade conceitual e computacional, à capacidade de redução do ruído em ambientes hostis (fábricas, aeroportos, helicópteros e caminhões, por exemplo) e necessidade de apenas um microfone para seu uso.

A teoria, a implementação prática e os resultados obtidos aparecem assim divididos nos capítulos seguintes:

Capítulo 2: Mostra os conceitos fundamentais de filtragem adaptativa e subtração espectral, expondo os diversos algoritmos mais utilizados em cancelamento de eco e as técnicas utilizadas para

³ARMA: *Autoregressive Moving Average*

analisar e reduzir o ruído presente em sinais de voz; o capítulo inclui o projeto de banco de filtros para decomposição dos sinais no domínio da frequência e também conceitos importantes de psicoacústica;

Capítulo 3: Mostra a implementação dos algoritmos de redução de eco e ruído em linguagem *assembly* e *C*.

Capítulo 4: Apresenta e analisa algumas resultados obtidos com o sistema implementado em processador digital de sinais (DSP).

Capítulo 5: Contém as conclusões gerais sobre o trabalho e algumas idéias para futuras investigações.

Listagens e artigos técnicos diretamente relacionados a este trabalho se encontram anexados ao final do documento:

Anexo A: Listagens (em Matlab) dos programas usados no projeto de filtros e tabelas para a implementação;

Anexo B: Listagens (em C e *assembly*) dos programas implementados no processador digital de sinais;

Anexos D e C: Reprodução de artigos, deste mesmo autor, publicados nos congressos SBT'99 e ICASSP 2001 a respeito da implementação do cancelador de eco acústico mostrado aqui.

Para finalizar esta introdução, é importante salientar que embora o uso de terminais viva-voz em automóveis tenha motivado e norteado todo este estudo, essa não é a única aplicação possível para as técnicas apresentadas no texto. Eis outras possibilidades:

- Uso em multimídia: equipamentos e periféricos para computadores pessoais voltados para aplicações de audioconferência e videoconferência podem incorporar redução de eco e ruído para tornar

as conversas mais naturais; os ambientes envolvidos nessas atividades são maiores que um automóvel típico e exigem maior carga computacional, mas é possível;

- Restauração de áudio com ruído estacionário (e.g. discos velhos e fitas com “chiados”);
- Equalização de montagens com microfones para apresentação em palco, onde o efeito de “microfonia” ocorre com frequência e pode ser facilmente controlado com um cancelador adaptativo;

Capítulo 2

Aspectos Teóricos

Este capítulo é dividido em duas seções, correspondendo a primeira à teoria usada no cancelamento de eco acústico e a segunda à redução de ruído.

A primeira seção começa com uma revisão dos conceitos de *filtragem adaptativa* e suas várias abordagens. Detem-se depois no método dos mínimos quadrados para explicar o algoritmo da família RLS¹ utilizado no desenvolvimento do cancelador de eco proposto. Nenhum algoritmo é perfeito em todos os aspectos, mas esse foi escolhido por apresentar alta taxa de convergência, robustez numérica e boa capacidade de ajuste às mudanças estatísticas do ambiente modelado.

A segunda seção inicia com a apresentação de uma revisão teórica sobre análise e síntese em frequência utilizando a transformada de Fourier. Essa revisão é estendida na Seção 2.2.3 com a introdução da estrutura polifásica e o projeto de filtros para essa estrutura. O algoritmo de subtração espectral convencional, em sua forma mais simples, é explicado na Seção 2.2.4 junto com a discussão de suas propriedades mais importantes. A Seção 2.2.5 apresenta algumas técnicas de obtenção das estatísticas do ruído

¹RLS: do inglês *Recursive Least-Squares*

e como podem ser utilizadas em conjunto com o algoritmo de subtração espectral. Finalmente, são introduzidos elementos de psicoacústica que tornam o algoritmo bastante mais robusto e com menos perda de qualidade na voz.

2.1 Cancelamento de Eco Acústico

2.1.1 As diferentes abordagens

O eco acústico, que atrapalha a comunicação entre participantes de uma conversa à distância usando terminais viva-voz, pode ser eliminado ou reduzido por meio da introdução de um *filtro* no processamento dos sinais de voz. Esse filtro, para todos os efeitos práticos, precisa se *adaptar* às propriedades estatísticas do sinal e às mudanças do ambiente no qual os terminais viva-voz operam.

Nesse contexto, identificam-se pelo menos três parcelas do sinal que trafega pelo sistema de comunicação (ver Figura 2.1 na página seguinte):

- O sinal de voz do participante remoto $x(n)$, reproduzido pelo alto-falante do terminal viva-voz do participante local;
- O sinal de voz do participante local $v(n)$, captado pelo microfone de seu próprio terminal viva-voz;
- O eco do sinal de voz do participante remoto, também captado pelo microfone do terminal local

Sem a presença do filtro adaptativo, o participante remoto recebe um sinal que é a combinação do sinal de voz do participante local e sua própria voz modificada pelo ambiente reverberante (e.g. compartimento de passageiros do automóvel). De modo grosseiro, o filtro adaptativo tem como objetivo criar uma estimativa $\hat{d}(n)$ para o sinal $d(n)$

Figura 2.1: Diagrama de blocos para o cancelador de eco acústico.

captado pelo microfone local a partir do sinal do alto-falante $x(n)$. Idealmente, supondo que a contribuição $v(n)$ para o sinal do microfone seja estatisticamente independente do eco acústico, o sinal de erro

$$e(n) = d(n) - \hat{d}(n) \quad (2.1)$$

do filtro contém apenas a parcela local $v(n)$.

Solução ótima. Na abordagem estatística, certos parâmetros como média e função de correlação são assumidos conhecidos para os sinais envolvidos.

Comumente, o critério adotado na solução para esse problema de otimização é minimizar o *valor quadrático médio* do sinal de erro $e(n)$, dando origem ao que se costuma chamar de filtro ótimo de Wiener [16].

Mais precisamente, o sinal de saída do filtro é dado pela soma de convolução

$$\hat{d}(n) = \sum_{k=0}^{M-1} w_k x(n-k), \quad (2.2)$$

na qual $w_0, w_1, \dots, w_{M-1}$ são os M coeficientes do filtro, quando imple-

mentado em uma estrutura transversal como na Figura 2.2.

Figura 2.2: Filtro transversal.

Os coeficientes são escolhidos de modo a minimizar um índice de desempenho definido como

$$J(\mathbf{w}) = E \{ |e(n)|^2 \} \quad (2.3)$$

em que $E\{\cdot\}$ denota o *operador estatístico esperança*, com o erro de estimação dado pela equação (2.1) e os coeficientes do filtro representados pelo vetor $\mathbf{w}$.

Sinais não-estacionários. Quando os sinais envolvidos não são estacionários, ou suas propriedades estatísticas não são conhecidas, não é possível projetar o filtro de Wiener. É possível, no entanto, contornar essa dificuldade por dois caminhos:

1. Adotar um procedimento de dois estágios, com a estimativa dos parâmetros estatísticos ocorrendo no primeiro e o resultado sendo usado para computar, de forma não-recursiva, os coeficientes do filtro no segundo; ou
2. Usar um filtro adaptativo, no qual um algoritmo *recursivo* parte de algumas condições iniciais para convergir aos poucos para a solução ótima de Wiener. Um algoritmo bem conhecido para fazer essa adaptação é chamado *Least-Mean Square*, ou LMS.

Mínimos quadrados. Na abordagem determinística, a derivação dos algoritmos não depende do conhecimento prévio de parâmetros estatísticos dos sinais envolvidos.

Nessa abordagem, baseada no método dos mínimos quadrados, o critério é minimizar uma *soma ponderada de erros quadráticos*. Como não se assume o conhecimento das propriedades estatísticas, esses algoritmos são apropriados para uso em ambientes não-estacionários e operação em tempo real, como é o caso deste trabalho.

Em sua derivação recursiva e com implementação em uma estrutura transversal, o método dá origem ao popular algoritmo *Recursive Least-Squares*, ou simplesmente RLS.

2.1.2 Algoritmo LMS

A derivação do algoritmo LMS sai do escopo deste documento, mas um breve esboço da álgebra envolvida é fornecido aqui. Ao final desta seção, uma tabela sumariza o algoritmo e pode ser usada diretamente para implementação.

De acordo com a teoria de Wiener, a equação de Wiener-Hopf—também conhecida como *equação normal*—descreve a solução ótima w_o para o filtro

$$\mathbf{R}w_o = \mathbf{p}, \quad (2.4)$$

com índice de desempenho dado pela equação (2.3).

$\mathbf{R}$ é a matriz de autocorrelação do sinal de entrada $x(n)$, dada por:

$$\mathbf{R} = E \{ \mathbf{x}(n)\mathbf{x}^T(n) \} \quad (2.5)$$

com $\mathbf{x}^T(n) = [x(n), x(n-1), \dots, x(n-M+1)]$.

O vetor $\mathbf{p}$ é a correlação cruzada entre o sinal de entrada e o sinal desejado (a saída do microfone, por exemplo), mais precisamente

$$\mathbf{p} = E \{ \mathbf{x}(n)d(n) \}. \quad (2.6)$$

Gradiente estocástico. Para encontrar a solução da equação (2.4), o algoritmo LMS adota uma versão modificada do método de otimização “steepest descent” (também conhecido como *gradiente determinístico*) para computar recursivamente o vetor ótimo w_o . Nesse método são utilizadas a matriz R e o vetor p para calcular o gradiente exato. Já no algoritmo LMS se utiliza uma estimativa instantânea do vetor gradiente a partir das amostras disponíveis a cada passo de adaptação.

O algoritmo resultante, conhecido como *gradiente estocástico*, segue apresentado na Tabela 2.1. Nessa tabela, o *passo de adaptação* μ deve obedecer a restrição

$$0 < \mu < \frac{2}{Mr(0)} \quad (2.7)$$

para garantir a estabilidade da adaptação, e $r(0)$ é a autocorrelação da função de entrada para deslocamento nulo.

Condições iniciais: $\mathbf{w}(n) = \mathbf{0}$
Operações: $e(n) = d(n) - \mathbf{w}^T(n)\mathbf{x}(n)$ $\mathbf{w}(n+1) = \mathbf{w}(n) + \mu\mathbf{x}(n)e(n)$

Tabela 2.1: Algoritmo LMS convencional.

2.1.3 Algoritmo RLS

O método dos mínimos quadrados e o algoritmo RLS possuem uma derivação longa e tediosa. Assim como explicado na seção anterior, isso foge ao escopo deste trabalho, e o esboço exposto aqui é suficiente para a compreensão geral do algoritmo.

Com o erro de estimação dado por (2.1), o critério de otimização no método *recursivo* dos mínimos quadrados é minimizar a soma ponde-

rada

$$\mathcal{E}(n) = \sum_{k=0}^n \lambda_L^{n-k} |e(k)|^2 \quad (2.8)$$

na qual $\lambda_L < 1$ é uma constante positiva, denominada *fator de esquecimento*, empregada para garantir que as amostras passadas tenham menor peso no índice de desempenho. Assim, o filtro pode operar em ambientes não-estacionários.

Usando um resultado conhecido como *lema da inversão matricial*, e partindo da equação normal adaptada ao índice de desempenho $\mathcal{E}(n)$, obtém-se o algoritmo RLS mostrado na Tabela 2.2. Nessa tabela, a filtragem ocorre na atualização do erro de estimação *a priori*² $\alpha(n)$ e na atualização dos coeficientes $\mathbf{w}(n)$. O *vetor de ganho* $\mathbf{k}(n)$ e a inversa da matriz de correlação $\mathbf{P}$ são atualizados com as equações restantes.

<p>Condições iniciais:</p> $\mathbf{P}(0) = \frac{1}{\delta} \mathbf{I}$ $\mathbf{w}(0) = \mathbf{0}$
<p>Operações:</p> $\mathbf{k}(n) = \frac{\lambda^{-1} \mathbf{P}(n-1) \mathbf{x}(n)}{1 + \lambda^{-1} \mathbf{x}^T(n) \mathbf{P}(n-1) \mathbf{x}(n)}$ $\alpha(n) = d(n) - \mathbf{w}^T(n-1) \mathbf{x}(n)$ $\mathbf{w}(n) = \mathbf{w}(n-1) + \mathbf{k}(n) \alpha(n)$ $\mathbf{P}(n) = \lambda^{-1} \mathbf{P}(n-1) - \lambda^{-1} \mathbf{k}(n) \mathbf{x}^T(n) \mathbf{P}(n-1)$

Tabela 2.2: Algoritmo RLS convencional.

2.1.4 Experimento computacional

O desempenho do algoritmo RLS pode ser comparado com o do algoritmo LMS por meio de um exemplo aplicado ao problema do cancelamento de eco acústico.

²Note que, na Tabela 2.1 e na definição 2.1, o erro $e(n)$ pode ser denominado *a posteriori*, já que é calculado com a amostra de entrada e e os coeficientes atuais.

Para simplificar a simulação, as condições abaixo foram consideradas:

1. O ambiente é artificialmente pequeno. A resposta impulsiva, mostrada na Figura 2.3, foi obtida usando uma caixa plástica revestida internamente com espuma. O volume da caixa é de cerca de 17 litros. Uma janela de Hamming de 21 amostras foi aplicada no início da resposta impulsiva.

Figura 2.3: Resposta impulsiva (janelada) da caixa usada no experimento.

2. O sinal $x(n)$ é um ruído branco aleatório gaussiano com variância $\sigma_x = 1$.
3. No ambiente há apenas um ruído de medida branco gaussiano com variância $\sigma_v = 10^{-6}$.
4. O número de coeficientes do filtro é $M = 21$.
5. O algoritmo RLS trabalha com memória infinita, ou seja, $\lambda_L = 1$.

6. Para o algoritmo LMS, a faixa de valores permitidos para o passo de adaptação é

$$0 < \mu < \frac{2}{M\sigma_x^2} \quad (2.9)$$

e, atendendo a esse critério, dois valores foram escolhidos: $\mu_1 = 0,02$ e $\mu_2 = 0,05$.

A Figura 2.4 mostra as *curvas de aprendizado* dos filtros para os dois algoritmos, ou seja, o erro de estimação $e(n)$ ao longo do tempo. As curvas foram obtidas pela média do conjunto de 40 realizações do processo.

Figura 2.4: Curvas de aprendizado para o exemplo.

Comparação. A convergência do algoritmo RLS é muito mais rápida que a do LMS. Cerca de 50 iterações (pouco mais que o dobro do número de coeficientes) são necessárias para que o erro de estimação atinja o patamar mínimo, para o RLS. Para o LMS esse tempo é de quatro a oito vezes maior. Essa diferença se deve ao fato de o algoritmo RLS utilizar toda a informação presente no sinal, desde o instante inicial de operação.

O valor final do erro de estimação produzido com o algoritmo RLS é significativamente menor que aquele gerado com o algoritmo LMS. Em teoria, o RLS produz “desajuste zero”, ou seja, o valor médio quadrático do erro de estimação tende à variância do erro de medida σ_v^2 . No caso do exemplo, embora não seja introduzido ruído de medida no ambiente, o número de coeficientes do filtro adaptativo é pequeno para modelar perfeitamente a resposta impulsiva da Figura 2.3. O resultado, portanto, é o desajuste finito mostrado na curva de aprendizado da Figura 2.4.

Para cada iteração, o algoritmo RLS requer um número de operações proporcional a M^2 , devido à atualização da matriz $P(n)$. Em contraste, para o LMS o número de operações é proporcional a M . A superioridade na velocidade de convergência conseguida com o algoritmo RLS é alcançada às custas do aumento significativo na carga computacional.

2.1.5 O algoritmo LSL

As primeiras tentativas de aliviar a carga computacional do algoritmo RLS convencional resultaram em filtros com problemas de estabilidade numérica [17]. A idéia [16, 18, 19] desses algoritmos *rápidos* é explorar a natureza seqüencial dos dados de entrada de modo a fazer com que precisem apenas de um número de operações aritméticas proporcional a M .

Montado em uma estrutura modular de *treliça*, o algoritmo *Least-Squares Lattice*, ou LSL, é uma alternativa para implementar o filtro adaptativo de maneira eficiente e numericamente robusta.

As versões do algoritmo LSL que utilizam mecanismos de realimentação de erro (“Error Feedback”, ou EF) se destacam pela precisão e robustez numérica [20]. Embora sua estabilidade não tenha sido demonstrada, o algoritmo EFLSL baseado em erros de estimação *a priori* tem dois pontos a seu favor:

1. Sua complexidade computacional pode ser reduzida para apenas $15M$ operações aritméticas por iteração, conforme descrição mais adiante no texto; e
2. Sua estabilidade numérica pode ser tornada equivalente à dos algoritmos QRLSL, cuja estabilidade numérica é rigorosamente demonstrada e que são conhecidos pelo seu bom comportamento numérico. Para isso basta tomar alguns cuidados, detalhados no capítulo de implementação.

Treliça. Embora a derivação completa do algoritmo EFLSL seja um processo longo e fora do escopo deste trabalho, são destacados aqui alguns aspectos do algoritmo. Em contraste com os algoritmos RLS e LMS, que são implementados sobre um filtro transversal, o algoritmo LSL emprega uma estrutura modular composta de três estimadores lineares (Figura 2.5):

1. Preditor progressivo com erros de estimação $\eta_i(n)$ e coeficientes de reflexão $\Gamma_{f,i}(n-1)$.
2. Preditor regressivo com erros de estimação $\psi_i(n)$ e coeficientes de reflexão $\Gamma_{b,i}(n-1)$
3. Preditor conjunto com erros de estimação $\alpha_{i+1}(n)$ e coeficientes de regressão $\kappa_i(n-1)$

Os M estágios da treliça na porção superior da Figura 2.5 transformam, de modo eficiente, a sequência de entrada

$$x(n), x(n-1), \dots, x(n-M)$$

em um conjunto correspondente de erros de predição regressiva

$$b_0(n), b_1(n), \dots, b_M(n).$$

Uma das características desse processo é o chamado *branqueamento* da sequência de entrada, ou seja, a sequência $b_0(n), b_1(n), \dots, b_M(n)$ possui amostras descorrelacionadas, o que simplifica significativamente o

Figura 2.5: Fluxograma do filtro em treliça utilizado no algoritmo LSL com estimação de erros de predição *a priori*.

cálculo da estimativa $\hat{d}(n)$ do sinal desejado. A porção inferior da Figura 2.5 representa essa estimativa por meio de um filtro transversal.

O algoritmo assim obtido é chamado de LSL convencional e está detalhado na Tabela 2.3.

Realimentação de erro. Ao invés de computar os coeficientes de reflexão $\Gamma_{f,m}$, $\Gamma_{b,m}$ de maneira indireta, o que resulta em imprecisão numérica quando o algoritmo é implementado em processadores digitais, Ling [20] modificou-o para obter os coeficientes de reflexão de forma direta. Desse modo, os erros de estimação η_m , ψ_m , α_{m+1} *realimentam* a atualização dos coeficientes de reflexão e os coeficientes de regressão κ_m . Essa versão do algoritmo LSL com realimentação de erro exibe melhor comportamento numérico que a versão convencional, e um ligeiro aumento no custo computacional.

O número de operações aritméticas desse algoritmo EFLSL ainda pode ser reduzida se forem introduzidos [21] erros de predição *a pos-*

<p>Condições iniciais ($n = 0, m = 1, \dots, M$):</p> $\Delta_{m-1}(0) = 0$ $\xi_{f,m-1}(0) = \delta$ $\xi_{b,m-1}(0) = \delta$
<p>Predição ($n \geq 1, m = 1, \dots, M$):</p> $\eta_m(n) = \eta_{m-1}(n) - \Gamma_{f,m}(n-1)\psi_{m-1}(n-1)$ $\psi_m(n) = \psi_{m-1}(n-1) - \Gamma_{b,m}(n-1)\eta_{m-1}(n)$ $\Delta_{m-1}(n) = \lambda_L \Delta_{m-1}(n-1) + \gamma_{m-1}(n-1)\psi_{m-1}(n-1)\eta_{m-1}(n)$ $\xi_{f,m-1}(n) = \lambda_L \xi_{f,m-1}(n-1) + \gamma_{m-1}(n-1) \eta_{m-1}(n) ^2$ $\xi_{b,m-1}(n) = \lambda_L \xi_{b,m-1}(n-1) + \gamma_{m-1}(n) \psi_{m-1}(n) ^2$ $\Gamma_{f,m}(n) = \frac{\Delta_{m-1}(n)}{\xi_{b,m-1}(n-1)}$ $\Gamma_{b,m}(n) = \frac{\Delta_{m-1}(n)}{\xi_{f,m-1}(n)}$ $\gamma_m(n) = \gamma_{m-1}(n) - \frac{\gamma_{m-1}^2(n) \psi_{m-1}(n) ^2}{\xi_{b,m-1}(n)}$
<p>Estimação conjunta ($n \geq 1, m = 0, \dots, M$):</p> $\rho_m(n) = \lambda_L \rho_m(n-1) + \gamma_m(n)\psi_m(n)\alpha_m(n)$ $\alpha_{m+1}(n) = \alpha_m(n) - \kappa_m(n-1)\psi_m(n)$ $\kappa_m(n) = \frac{\rho_m(n)}{\xi_{b,m}(n)}$

Tabela 2.3: Algoritmo LSL convencional.

teriori normalizados, relacionados aos erros de predição *a priori* da seguinte forma:

$$\begin{aligned}
 \bar{f}_{m-1}(n) &= f_{m-1}(n)/\xi_{f,m-1}(n) \\
 &= \gamma_{m-1}(n-1)\eta_{m-1}(n)/\xi_{f,m-1}(n) \\
 \bar{b}_{m-1}(n) &= b_{m-1}(n)/\xi_{b,m-1}(n) \\
 &= \gamma_{m-1}(n)\psi_{m-1}(n)/\xi_{b,m-1}(n)
 \end{aligned}
 .$$

A Tabela 2.4 apresenta um resumo do algoritmo EFLSL modificado, implementado neste trabalho de acordo com detalhes fornecidos no Capítulo 3.

1	$\eta_m(n) = \eta_{m-1}(n) - \Gamma_m^f(n-1)\psi_{m-1}(n-1)$
2	$\psi_m(n) = \psi_{m-1}(n-1) - \Gamma_m^b(n-1)\eta_{m-1}(n)$
3	$b_{m-1}(n) = \gamma_{m-1}(n)\psi_{m-1}(n)$
4	$\xi_{m-1}^b(n) = \lambda_L \xi_{m-1}^b(n-1) + b_{m-1}(n)\psi_{m-1}(n) + \zeta$
5	$\bar{b}_{m-1}(n) = \frac{b_{m-1}(n)}{\xi_{m-1}^b(n)}$
6	$\gamma_m(n) = \gamma_{m-1}(n) - \bar{b}_{m-1}(n)b_{m-1}(n)$
7	$f_{m-1}(n) = \gamma_{m-1}(n-1)\eta_{m-1}(n)$
8	$\xi_{m-1}^f(n) = \lambda_L \xi_{m-1}^f(n-1) + f_{m-1}(n)\eta_{m-1}(n) + \zeta$
9	$\bar{f}_{m-1}(n) = \frac{f_{m-1}(n)}{\xi_{m-1}^f(n)}$
10	$\Gamma_m^f(n) = \Gamma_m^f(n-1) + \bar{b}_{m-1}(n-1)\eta_m(n)$
11	$\Gamma_m^b(n) = \Gamma_m^b(n-1) + \bar{f}_{m-1}(n)\psi_m(n)$
12	$\alpha_m(n) = \alpha_{m-1}(n) - \psi_{m-1}(n)\kappa_m^d(n-1)$
13	$\kappa_m^d(n) = \kappa_m^d(n-1) + \bar{b}_{m-1}(n)\alpha_m(n)$

Tabela 2.4: Algoritmo EFLSL modificado.

2.2 Redução de Ruído

Dentro do escopo deste trabalho, o sinal apresentado ao sistema de redução de ruído tem um único *canal*, isto é, somente um microfone é utilizado para captar os sinais do interior do veículo.

Considerando essa restrição, todos os algoritmos de subtração espectral se baseiam no diagrama de blocos da Figura 2.6, que destaca três etapas de processamento [22]:

- Análise espectral de curto prazo, responsável por converter o sinal de entrada (vindo do microfone) em uma representação em frequência variante no tempo;
- Processamento adaptativo para ressaltar somente a parcela vocal presente no espectro do sinal de entrada, *subtraindo* o ruído com base em alguma estimativa prévia das estatísticas do mesmo; e

- Recomposição do sinal no tempo, a partir do espectro modificado, etapa essa que faz o processo inverso da decomposição espectral.

Figura 2.6: Diagrama de subtração espectral básica.

2.2.1 Banco de filtros: fundamentos [23]

Análise: estrutura passa-baixas

Considere $x(k)$ um sinal amostrado uniformemente no tempo e servindo de entrada a uma estrutura de análise como mostra a Figura 2.7-a. Nela, o sinal passa por um conjunto de moduladores e filtros passa-baixas idênticos, com resposta impulsiva $h_0(k)$.

A cada filtro é apresentada uma cópia do sinal de entrada $x(k)$ deslocada em frequência pela modulação complexa³ correspondente $e^{-j\Omega_\mu k}$, onde

$$\Omega_\mu = \frac{2\pi\mu}{M}, \quad \mu = 0, 1, \dots, M-1 \quad (2.10)$$

são as frequências centrais das M faixas em que o espectro é dividido e $\Omega = 2\pi$ é a frequência de amostragem normalizada. Cada filtro

³Neste texto, $j = \sqrt{-1}$

deixa então passar por si uma pequena *banda* do espectro completo de frequências.

Figura 2.7: Análise espectral com banco de filtros passa-baixas: (a) diagrama de blocos; (b) exemplo gráfico para $M = 6$ e $r = 4$.

À saída dos filtros aparece a *somatória de convolução* abaixo

$$y_\mu(k) = \sum_{n=-\infty}^{+\infty} h_0(k-n)x(n)e^{-j\Omega_\mu n}, \quad (2.11)$$

que vem a ser a transformada de Fourier de curto prazo do sinal $x(k)$ na frequência Ω_μ . Na realidade o índice da somatória tem limites em

$n = 0$ e $n = k$, pois tanto o sinal $x(k)$ quanto o filtro $h_0(k)$ são causais; os limites utilizados, no entanto, facilitam o desenvolvimento matemático que vem a seguir.

Dizimação. Idealmente, $h_0(k)$ deve ter um espectro como aquele sugerido pela curva tracejada na Figura 2.7-b, ou seja,

$$H_0(\Omega) = \begin{cases} 1 & \text{se } |\Omega| < \frac{\pi}{M} \\ 0 & \text{se } |\Omega| \geq \frac{\pi}{M}, \end{cases} \quad (2.12)$$

para que cada sinal $y_\mu(k)$ não contenha informação redundante de outras bandas, e nem restem trechos sem a devida representação em frequência. Nesse caso, $y_\mu(k)$ fica limitado à faixa $-\frac{\pi}{M} \leq \Omega \leq \frac{\pi}{M}$ e pode ter sua taxa de amostragem reduzida de um fator r

$$x_\mu(k) = y_\mu(rk), \quad r \in \mathbb{N}_* \quad (2.13)$$

operação usualmente denominada de *dizimação* [24] e representada por $\downarrow r$ (ver Figura 2.7 na página precedente).

No domínio das frequências, uma expressão para a dizimação pode ser obtida utilizando-se de resultados referentes à amostragem de sinais contínuos no tempo [25]. O resultado

$$X_\mu(\Omega) = \frac{1}{r} \sum_{\ell=0}^{r-1} Y_\mu\left(\frac{\Omega - 2\pi\ell}{r}\right) \quad (2.14)$$

mostra que o sinal dizimado é composto por r cópias do sinal original escalado em frequência pelo fator r e deslocado de múltiplos inteiros de 2π (ver Figura 2.7-b). A equação (2.14) também deixa claro que o efeito de *rebatimento*—sobreposição das múltiplas cópias—pode ser evitado se

$$\frac{2\pi}{M} \leq \frac{2\pi}{r} \Rightarrow r \leq M. \quad (2.15)$$

A condição $r = M$. Aqui cabe um comentário de ordem prática: os filtros com resposta em frequência dada por (2.12) são idealizados e não podem ser implementados fisicamente, exceto de forma aproximada.

A consequência importante desse fato é que na prática os filtros não podem ser construídos tão “abruptos” quanto ilustrado na Figura 2.7-b mas deve existir uma *região de transição* finita em sua forma espectral.

Por isso, a condição $r = M$ na equação (2.15) só pode ser utilizada nos casos em que o espectro não é totalmente preenchido, como nas aplicações de conversão de sinais modulados em frequência (e.g. rádio e TV comuns, comunicação celular analógica) para sinais multiplexados no tempo (e.g. celular digital TDMA⁴). Nas aplicações onde o rebatimento é indesejável, como no processamento de áudio e voz, é preciso usar $r < M$ ou projetar os filtros para que a região de transição caia dentro da banda $|\Omega| \leq \frac{\pi}{M}$ e atenuar partes do sinal, o que também pode ser indesejável.

Após esse comentário, o leitor pode voltar sua atenção novamente para a expressão (2.11). Substituindo-a em (2.13) tem-se

$$x_\mu(k) = \sum_{n=-\infty}^{+\infty} h_0(rk - n)x(n)e^{-j\Omega_\mu n}. \quad (2.16)$$

Ou seja: a saída desse sistema de análise espectral é o conjunto das seqüências, amostras da transformada de Fourier de curto prazo (STFT⁵) nos instantes rk e nas frequências $2\pi/M\mu$.

A redução da taxa de amostragem deve ser considerada quando da necessidade de realizar de maneira computacionalmente eficiente a decomposição em bandas expressa pela equação (2.16), e a Seção 2.2.3 trata desse assunto em mais detalhes.

Análise: estrutura passa-faixa

Enquanto a discussão da seção anterior gira em torno de uma representação do tipo passa-baixas para as seqüências das bandas em frequência, a Figura 2.8-a apresenta uma alternativa onde o sinal de entrada

⁴TDMA: *Time Division Multiple Access*.

⁵STFT: *Short Time Fourier Transform*.

$x(k)$ não é modulado em frequência. Ao contrário, os filtros é que sofrem um deslocamento no espectro a partir do *protótipo* $h_0(k)$, e passam a ser escritos como

$$h_\mu(k) = h_0(k)e^{j\Omega_\mu k}. \quad (2.17)$$

(a)

(b)

Figura 2.8: Análise espectral com banco de filtros passa-faixa: (a) diagrama de blocos; (b) exemplo gráfico para $M = 6$ e $r = 4$.

Aplicando-se o sinal de entrada a esses filtros, tem-se na saída

$$\tilde{y}_\mu(k) = \sum_{n=-\infty}^{+\infty} h_\mu(k-n)x(n) \quad (2.18)$$

e substituindo (2.17) em (2.18) obtém-se

$$\begin{aligned}
 \tilde{y}_\mu(k) &= \sum_{n=-\infty}^{+\infty} h_0(k-n)e^{j\Omega_\mu(k-n)}x(n) = \\
 &= e^{j\Omega_\mu k} \sum_{n=-\infty}^{+\infty} h_0(k-n)x(n)e^{-j\Omega_\mu n} = \\
 &= y_\mu(k)e^{j\Omega_\mu k}.
 \end{aligned} \tag{2.19}$$

De acordo com a expressão desenvolvida acima, as saídas dos filtros são versões passa-faixa dos sinais definidos na equação (2.11). Veja, por exemplo, a ilustração na Figura 2.8-b, que mostra, para o caso de $M = 6$ bandas, um dos filtros passa-faixa—em linha tracejada—agindo sobre o sinal $x(k)$. Note que o espectro da saída do filtro $\tilde{Y}_2(\Omega)$ não está centralizado na origem das frequências.

Supondo então novamente o uso de filtros ideais, a taxa de amostragem pode ser reduzida de um fator r conforme discussão ocorrida na Seção 2.2.1, desde que $r \leq M$, com as restrições já discutidas também naquela seção. Assim sendo, à saída do sistema aparece a expressão

$$\begin{aligned}
 \tilde{x}_\mu(k) &= y_\mu(rk)e^{j\Omega_\mu rk} = \\
 &= e^{j\Omega_\mu rk} \sum_{n=-\infty}^{+\infty} h_0(rk-n)x(n)e^{-j\Omega_\mu n} = \\
 &= x_\mu(k)e^{j\Omega_\mu rk},
 \end{aligned} \tag{2.20}$$

que, quando comparada à equação (2.16), revela as similaridades e diferenças entre as estruturas passa-baixas e passa-faixa.

Comparação entre as estruturas. Em geral, os sinais $\tilde{x}_\mu(k)$ e $x_\mu(k)$ são diferentes, porém contêm a mesma informação: ambos representam a evolução no tempo da transformada de Fourier de curto prazo na frequência Ω_μ . Apenas, se $r < M$, o espectro de $\tilde{x}_\mu(k)$ é ligeiramente deslocado em relação à sua versão passa-baixas $x_\mu(k)$. No caso de $r = M$, o argumento da exponencial complexa

$$e^{j\Omega_\mu rk} = e^{j\frac{2\pi}{M}rk}$$

se torna múltiplo inteiro de 2π e as duas estruturas se tornam matematicamente equivalentes, ou seja,

$$r = M \rightsquigarrow \tilde{x}_\mu(k) = x_\mu(k). \quad (2.21)$$

Síntese: estrutura passa-baixas

O objetivo desta seção e da próxima é explicar como o processo de análise pode ser revertido, isto é, como, a partir dos sinais nas bandas de frequência (que representam a transformada de Fourier de curto prazo) se pode obter o correspondente sinal no tempo.

Suponha que estão disponíveis M seqüências sub-amostradas $\hat{s}_\mu(k)$, $\mu = 0, 1, \dots, M-1$, saídas de uma estrutura de análise passa-baixas como desenvolvida na Seção 2.2.1. A notação utilizada inclui possíveis modificações espectrais no sinal $x_\mu(k)$ e ficará mais clara na Seção 2.2.4.

Figura 2.9: Síntese espectral com banco de filtros passa-baixas.

Conforme ilustrado no diagrama da Figura 2.9, essas seqüências sofrem primeiramente um aumento—de um fator r —na taxa de amostragem (representado por blocos com $\uparrow r$)

$$u_\mu(k) = \begin{cases} \hat{s}_\mu\left(\frac{k}{r}\right) & \text{se } k = \rho r \\ 0 & \text{se } k \neq \rho r \end{cases}, \quad (2.22)$$

com $\rho \in \mathbb{Z}$. Essa operação, conhecida como *interpolação*, se expressa no domínio das frequências como

$$U_\mu(\Omega) = \hat{S}_\mu(r\Omega), \quad (2.23)$$

o que permite visualizar melhor seu efeito: as amostras nulas introduzidas pela interpolação causam o aparecimento de repetições periódicas (com período $\frac{2\pi}{r}$) no espectro devido à mudança na escala de frequências.

A filtragem passa-baixas subsequente

$$v_\mu(k) = \sum_{n=-\infty}^{+\infty} g_0(k-n)u_\mu(n) \quad (2.24)$$

tem por objetivo eliminar essas repetições e limitar o espectro de cada seqüência à faixa $|\Omega| \leq \frac{\pi}{M}$. Se os filtros utilizados forem ideais, nenhuma informação é perdida nesse processo; o aumento na *redundância*—mais amostras para representar a mesma informação—é temporário e apenas um passo na “montagem” da seqüência a ser obtida no final do processo de síntese.

O filtro $g_0(k)$ ideal tem o mesmo formato daquele definido para a estrutura de análise (ver equação (2.12 na página 22))

$$G_0(\Omega) = \begin{cases} r \cdot 1 & \text{se } |\Omega| < \frac{\pi}{M} \\ 0 & \text{se } |\Omega| \geq \frac{\pi}{M} \end{cases}, \quad (2.25)$$

onde o ganho adicional r serve para compensar a diminuição na *densidade de potência* que ocorre com a inserção das amostras nulas.

Modulação. Após a filtragem, os sinais $v_\mu(k)$, ainda centralizados na origem, são deslocados em frequência por meio da modulação complexa $e^{+j\Omega_\mu k}$ e somados

$$\hat{s}(k) = \sum_{\mu=0}^{M-1} e^{+j\Omega_\mu k} v_\mu(k). \quad (2.26)$$

Substituindo (2.24) em (2.26), tem-se a expressão

$$\hat{s}(k) = \sum_{\mu=0}^{M-1} \sum_{n=-\infty}^{+\infty} g_0(k-n)u_{\mu}(n)e^{+j\Omega_{\mu}k}, \quad (2.27)$$

que ainda precisa ser relacionada aos sinais de entrada $\hat{s}_{\mu}$. Para isso, é preciso reescrever a expressão da interpolação utilizando um trem de pulsos ao invés de uma definição condicional como a equação (2.22). Assim, é possível verificar que

$$u_{\mu}(k) = \sum_{\ell=-\infty}^{\infty} \hat{s}_{\mu}(\ell)\delta(k-r\ell), \quad (2.28)$$

onde $\delta(k)$ denota o pulso unitário de Dirac [25] na origem.

Com a equação (2.28) substituída em (2.27)

$$\begin{aligned} \hat{s}(k) &= \sum_{\mu=0}^{M-1} \sum_{n=-\infty}^{+\infty} g_0(k-n)u_{\mu}(n)e^{+j\Omega_{\mu}k} \\ &= \sum_{\mu=0}^{M-1} \sum_{n=-\infty}^{+\infty} \sum_{\ell=-\infty}^{\infty} g_0(k-n)\hat{s}_{\mu}(\ell)\delta(k-r\ell)e^{+j\Omega_{\mu}k} \\ &= \sum_{\mu=0}^{M-1} \sum_{\ell=-\infty}^{\infty} \underbrace{\sum_{n=-\infty}^{+\infty} g_0(k-n)\delta(k-r\ell)}_{g_0(k-r\ell)} \hat{s}_{\mu}(\ell)e^{+j\Omega_{\mu}k}, \end{aligned}$$

chega-se ao sinal reconstituído no tempo

$$\hat{s}(k) = \sum_{\mu=0}^{M-1} \left[\sum_{\ell=-\infty}^{\infty} g_0(k-r\ell)\hat{s}_{\mu}(\ell) \right] e^{+j\Omega_{\mu}k}. \quad (2.29)$$

A equação que acaba de ser derivada pode ser identificada como a fórmula da transformada inversa de Fourier de curto prazo, que permite obter exatamente o sinal $x(k)$ utilizado como entrada para análise se os filtros forem ideais e não houver nenhuma modificação no espectro.

Síntese: estrutura passa-faixa

A Figura 2.10 mostra o diagrama de blocos relativo ao processo de síntese espectral utilizando uma representação passa-faixas $\tilde{s}_\mu(k)$. O processo de interpolação é semelhante àquele descrito na seção anterior pela equação (2.22), com inserção de amostras nulas

$$\tilde{u}_\mu(k) = \begin{cases} \tilde{s}_\mu(\frac{k}{r}) & \text{se } k = \rho r \\ 0 & \text{se } k \neq \rho r \end{cases}, \quad (2.30)$$

seguida de filtragem passa-faixa

$$\tilde{v}_\mu(k) = \sum_{n=-\infty}^{+\infty} g_\mu(k - n) \tilde{u}_\mu(n). \quad (2.31)$$

Figura 2.10: Síntese espectral com banco de filtros passa-faixa.

Considerando-se ideal cada filtro $g_\mu(k)$, seu espectro é o mesmo utilizado na estrutura passa-baixas deslocado em frequência pela modulação complexa

$$g_\mu(k) = g_0(k) e^{j\Omega_\mu k} \quad (2.32)$$

e age sobre o sinal interpolado atenuando completamente as repetições periódicas (ver Seção 2.2.1) excetuando-se o espectro na região $\Omega_\mu - \frac{\pi}{M} \leq \Omega \leq \Omega_\mu + \frac{\pi}{M}$. Note que não é necessária uma etapa de modulação no sinal para prepará-lo para ser combinado com as outras bandas, ou melhor, a modulação está implícita na definição dos filtros.

Combinação das bandas. Os M sinais interpolados, reescritos pela substituição de (2.32) em (2.31), são somados na saída

$$\tilde{s}(k) = \sum_{\mu=0}^{M-1} \sum_{n=-\infty}^{+\infty} g_0(k-n) e^{j\Omega_\mu(k-n)} \tilde{u}_\mu(n). \quad (2.33)$$

Seguindo-se um procedimento análogo àquele desenvolvido para a representação passa-baixas, a equação acima pode ser relacionada às seqüências sub-amostradas. Reescrevendo a expressão da interpolação

$$\tilde{u}_\mu(k) = \sum_{\ell=-\infty}^{\infty} \tilde{s}_\mu(\ell) \delta(k - r\ell) \quad (2.34)$$

e substituindo-a em (2.33), após alguns passos algébricos, tem-se finalmente

$$\tilde{s}(k) = \sum_{\mu=0}^{M-1} \left[\sum_{\ell=-\infty}^{\infty} g_0(k - r\ell) \tilde{s}_\mu(\ell) e^{-j\Omega_\mu r\ell} \right] e^{j\Omega_\mu k}. \quad (2.35)$$

Comparação com estrutura passa-baixas. Os resultados apresentados nas equações (2.29) e (2.35) podem ser melhor interpretados na condição de redução máxima da taxa de amostragem $r = M$ e uso de filtragem ideal. Nesse caso, a exponencial complexa $e^{-j\Omega_\mu r\ell}$ reduz-se à unidade, as seqüências das bandas são idênticas (ver discussão no fim da Seção 2.2.1), e os sinais reconstruídos $\tilde{s}(k)$ e $\hat{s}(k)$ são portanto matematicamente iguais. Na verdade, mesmo para $r < M$, pode-se verificar que os sinais reconstruídos $\tilde{s}$ e $\hat{s}$ são idênticos se as estruturas utilizadas na análise e síntese forem compatíveis.

2.2.2 Resposta em frequência

O projeto dos filtros usados como protótipos, ou seja $h_0(k)$ e $g_0(k)$, deve considerar não apenas suas características individuais, mas também sua ação global no banco de filtros.

Para os propósitos deste trabalho, seria ideal obter um sinal na saída do banco de filtros idêntico àquele apresentado em sua entrada.

Na prática, entretanto, essa condição de *reconstrução perfeita* impõe restrições muito grandes ao projeto dos filtros, sendo substituída comumente pela aproximação descrita no restante desta seção.

Composição das faixas. A observação das Figuras 2.8 e 2.10, descrevendo a estrutura passa-faixa para o banco de filtros, permite a derivação simples da resposta em frequência global [23].

A convolução, no tempo, das expressões (2.17) e (2.32) fornece a resposta impulsiva efetiva para cada uma das M faixas do espectro

$$\begin{aligned}
 h_{ef,\mu}(k) &= \sum_{\ell=0}^k h_{\mu}(\ell)g_{\mu}(k-\ell) \\
 &= \sum_{\ell=0}^k h_0(\ell)e^{j\Omega_{\mu}\ell}g_0(k-\ell)e^{j\Omega_{\mu}(k-\ell)} \\
 &= \sum_{\ell=0}^k h_0(\ell)g_0(k-\ell)e^{j\Omega_{\mu}k}.
 \end{aligned} \tag{2.36}$$

A superposição das respostas acima resulta em

$$\begin{aligned}
 h_{comp}(k) &= \sum_{\mu=0}^{M-1} h_{ef,\mu}(k) = \\
 &= \sum_{\ell=0}^k h_0(\ell)g_0(k-\ell)s(k),
 \end{aligned} \tag{2.37}$$

onde $s(k)$ representa o trem de pulsos

$$s(k) = \sum_{\mu=0}^{M-1} e^{j\Omega_{\mu}k} = \begin{cases} M & \text{se } k = \pm aM \\ 0 & \text{se } k \neq \pm aM \end{cases} \tag{2.38}$$

com $a = \{1, 2, 3, \dots\}$.

“Aliasing.” Note que a expressão (2.37) é válida apenas se o rebatimento espectral (“aliasing”) causado pela redução da taxa de amostragem puder ser desconsiderado. Uma maneira de garantir essa condição, para sub-amostragem não-crítica ($r < M$), é impor uma atenuação *elevada* para a faixa de rejeição nos protótipos $h_0(k)$ e $g_0(k)$. Valores

objetivos para essa atenuação dependem do número de bandas e de critérios psicoacústicos da audição humana.

Seja então $\psi_{h,g}(k)$ a convolução dos protótipos passa-baixas

$$\psi_{h,g}(k) = \sum_{\ell=0}^k h_0(\ell)g_0(k-\ell). \quad (2.39)$$

Notando-se que

$$h_{comp}(k) = \psi_{h,g}(k)s(k) \quad (2.40)$$

é uma versão subamostrada⁶ de $\psi_{h,g}(k)$, duas condições são suficientes para garantir uma resposta impulsiva ideal (isto é, um pulso unitário) para o banco de filtros como um todo:

$$\begin{aligned} \text{i)} \quad & \psi_{h,g}(k_0) = 1/M \\ \text{ii)} \quad & \psi_{h,g}(k_0 \pm aM) = 0 \end{aligned} \quad (2.41)$$

A constante $k_0 = a_0M$ serve para contabilizar um possível atraso introduzido pelos filtros. Em implementações práticas feitas para operar em tempo real, esse atraso (que sempre existe) não pode ser compensado, apenas minimizado.

Filtro ideal. O filtro passa-baixas ideal, com frequência de corte

$$\Omega_c = \frac{\pi}{M}, \quad (2.42)$$

obviamente satisfaz as especificações (2.41) acima, mas é de pouca utilidade prática. Uma forma de atender às especificações é multiplicar a resposta impulsiva ideal

$$h_{ideal}(k) = \frac{\sin \frac{\pi}{M}k}{\pi k} \quad (2.43)$$

por uma *janela* de comprimento finito e formato apropriado para garantir a atenuação adequada na faixa de rejeição. Janelas como as de Hamming, Kaiser e Chebyshev podem ser utilizadas [26].

⁶Descontando-se o fator de escala presente em $s(k)$.

Passando-se agora a expressão (2.37) para o domínio das frequências, obtém-se

$$H_{comp}(\Omega) = \sum_{\mu=0}^{M-1} H_0(\Omega - \frac{2\pi}{M}\mu) G_0(\Omega - \frac{2\pi}{M}\mu), \quad (2.44)$$

expressão que, com o mesmo protótipo escolhido para análise e síntese, ou seja,

$$h_0 = g_0, \quad (2.45)$$

se torna simplesmente

$$H_{comp}(\Omega) = \sum_{\mu=0}^{M-1} H_0^2(\Omega - \frac{2\pi}{M}\mu). \quad (2.46)$$

A resposta em frequência do filtro protótipo $H_0(\Omega)$ pode ser obtida a partir do projeto de $H_0^2(\Omega)$ extraíndo-se a raiz quadrada. Como o método da multiplicação por janela, mencionado há pouco no texto, não garante a magnitude positiva de $H_0^2(\Omega)$, costuma-se (1) somar uma pequena constante em $H_0^2(\Omega)$ ou (2) tomar seu valor absoluto antes de extrair a raiz quadrada de seu módulo.

Atraso. A fase de $H_0(\Omega)$ é escolhida de modo a minimizar o *atraso de grupo* provocado pelo banco de filtros no sinal de voz. O protótipo assim escolhido é dito de *mínima fase*.

Uma maneira de satisfazer essa condição explora algumas relações entre a fase $\arg(H_0)$ e o logaritmo da magnitude $\log |H_0|$ da resposta em frequência do filtro protótipo. Essas relações são baseadas nas “Transformações de Hilbert” e seu detalhamento foge ao escopo deste trabalho [25].

Aqui basta lembrar que, para uma resposta impulsiva de mínima fase, o logaritmo da magnitude e a fase de sua transformada de Fourier podem ser derivados um do outro por uma transformada de Hilbert. O diagrama de blocos da Figura 2.11 na próxima página mostra o processo.

Figura 2.11: Cálculo de um sistema de mínima fase usando transformada de Hilbert.

Note que a transformação de Hilbert é realizada por três blocos ilustrados na figura, a saber: uma transformada de Fourier inversa (iDFT), uma multiplicação pela sequência

$$\ell_{min}(n) = 2u(n) - \delta(n), \quad (2.47)$$

na qual $u(n)$ representa um degrau unitário, e finalmente uma transformada de Fourier direta.

A fase de $H_{min}(\Omega)$ não é mais necessariamente linear, porém a eventual distorção introduzida não é relevante para o processamento de sinais de áudio. O ganho, em termos de atraso perceptível no sinal, é bastante significativo e mais importante que uma eventual distorção na fase.

Exemplo. O filtro projetado para $M = 32$ bandas, pode ser visto representado por sua resposta impulsiva na Figura 2.12 na página seguinte e resposta em frequência na Figura 2.13 na página 36. Em cada gráfico, duas curvas são mostradas:

- Projeto de fase linear (linha tracejada); e
- Projeto de mínima fase (linha contínua).

Para o projeto deste protótipo foi utilizada uma janela de Kaiser. Note que resposta em frequência resultante permite somente a interpenetração de filtros adjacentes, o que virtualmente elimina o efeito do rebatimento na superposição espectral.

Figura 2.12: Resposta impulsiva do filtro protótipo para $M = 32$ e especificação de 60 dB de atenuação na faixa de rejeição.

2.2.3 Estrutura polifásica

Análise. Um esquema equivalente ao da Figura 2.7 na página 21, mas requerendo menos esforço computacional, pode ser obtido como explicado a seguir [23]. A equação (2.11) pode ser dividida em duas somatórias, usando-se a substituição de variáveis

$$n = \ell M + \nu \quad (2.48)$$

na qual $\ell = \{0, 1, 2, 3, \dots\}$ e $\nu = \{0, 1, \dots, M - 1\}$. Tomando a redução da taxa de amostragem pretendida,

$$k = q \cdot r, \quad q = \{0, 1, 2, 3, \dots\} \quad (2.49)$$

obtém-se

$$y_\mu(qr) = \sum_{\ell=0}^{+\infty} \sum_{\nu=0}^{M-1} h_0(qr - \ell M - \nu) x(\ell M + \nu) e^{-j\Omega_\mu \nu}. \quad (2.50)$$

Figura 2.13: Resposta em frequência do filtro protótipo para $M = 32$ e especificação de 60 dB de atenuação na faixa de rejeição.

Assumindo que a razão

$$L = \frac{M}{r} \quad (2.51)$$

é um inteiro e definindo M diferentes subsequências de entrada (dizimação de x)

$$x_{\nu}^{(M)}(k) = \begin{cases} 0 & \text{se } k \neq \ell M + \nu \\ x(k) & \text{se } k = \ell M + \nu \end{cases} \quad (2.52)$$

além de M respostas impulsivas parciais (dizimação de h_0)

$$h_{\nu}^{(r)}(k) = \begin{cases} 0 & \text{se } k \neq qr - \nu \\ h_0(k) & \text{se } k = qr - \nu \end{cases}, \quad (2.53)$$

após algumas manipulações algébricas, chega-se à expressão abaixo

$$y_{\mu}(qr) = \sum_{\nu=0}^{M-1} [h_{\nu}^{(r)}(qr) * x_{\nu}^M(qr)] \cdot e^{-j\Omega_{\mu}\nu}, \quad (2.54)$$

em que o operador $*$ denota convolução temporal.

A equação (2.54) mostra que o deslocamento em frequência, a filtragem passa-baixas e a subsequente redução da taxa de amostragem podem ser descritos igualmente pela convolução de cada subsequência $x_\nu^{(M)}$ com a resposta impulsiva parcial $h_\nu^{(r)}$ e subsequente cálculo da DFT do resultado a cada instante $k = qr$.

Um esquema da estrutura polifásica passa-baixas do banco de filtros, para *análise espectral* de curto prazo pode ser visto na Figura 2.14, e reúne as seguintes características:

- A operação de sub-amostragem da entrada é representada por uma chave comutadora;
- As amostras x_μ das seqüências de saída são idênticas às da Figura 2.7;
- Todo o cálculo (filtragem parcial e DFT) é feito na taxa de amostragem reduzida e valores de saída indesejados (para $k \neq qr$) não são calculados;
- Note que somente r respostas impulsivas parciais são diferentes e formam L grupos idênticos;
- Se a resposta impulsiva h_0 do protótipo passa-baixas é de largura finita $n = mr$, então as respostas impulsivas parciais $h_\nu^{(r)}$ são menores, de largura m .

Devido à sub-amostragem do protótipo passa-baixas h_0 , fases diferentes estão associadas com as funções de transferência dos caminhos individuais da estrutura, razão para o termo *rede polifásica* normalmente encontrado na literatura.

Síntese. A reconstrução do sinal a partir de seu espectro, descrita pela Figura 2.9 na página 26, também pode ser feita de modo mais eficiente seguindo procedimento análogo ao descrito para a análise. Partindo da equação (2.28), usando a substituição de variáveis

$$k = \ell M + \nu, \quad (2.55)$$

Figura 2.14: Diagrama mostrando análise espectral em estrutura polifásica para $r = M/2, L = 2$.

com $\nu = \{0, 1, \dots, M - 1\}$ e definindo M respostas impulsivas parciais para os filtros de síntese (dizimação de g_0)

$$g_\nu^{(r)}(k) = \begin{cases} 0 & \text{se } k \neq \lambda r + \nu \\ g_0(k) & \text{se } k = \lambda r + \nu \end{cases}, \quad (2.56)$$

obtem-se

$$\hat{s}(\ell M + \nu) = g_\mu^{(r)}(\ell M + \nu) * \left[\sum_{\mu=0}^{M-1} \hat{s}_\mu(qr) e^{+j\Omega_\mu \nu} \right]. \quad (2.57)$$

A Figura 2.15 mostra ilustra o esquema resultante. Nela se observa que uma iDFT (transformada inversa de Fourier) gera M seqüências que são passadas pelos filtros parciais $g_\mu^{(r)}$. O comutador se encarrega de elevar a taxa de amostragem, compondo as saídas dos filtros.

Figura 2.15: Diagrama mostrando síntese espectral em estrutura polifásica para $r = M/2, L = 2$.

2.2.4 Subtração espectral

Descrito o processo de decomposição—e recomposição—do sinal de voz (ver seções anteriores), é possível agora passar às explicações sobre a modificação controlada do espectro.

O sinal de voz $s(k)$ está contaminado por ruído $r(k)$ de forma que o sinal de entrada do banco de filtros de análise é dado $x(k) = s(k) + r(k)$. Este sinal é decomposto em M sinais sub-amostrados $x_\mu(n)$.

Graças a propriedades psicoacústicas, a supressão de ruído pode ser efetuada ponderando adaptativamente os sinais x_μ da seguinte forma:

$$\hat{s}_\mu(n) = c_\mu(n)x_\mu(n), \quad (2.58)$$

sendo $0 \leq c_\mu(n) \leq 1$.

O sinal $\hat{s}_\mu(n)$ indica uma estimativa do sinal $s_\mu(n)$, componente na banda μ do sinal de voz original não contaminado e sub-amostrado. O sinal $\hat{s}(n)$ é então obtido por interpolação e superposição, operações que são efetuadas pelo banco de filtros de síntese.

Essencialmente, os coeficientes reais $c_\mu(n)$ são ajustados de forma a diminuir a magnitude do espectro do sinal contaminado nas faixas de frequência em que a potência do ruído é “alta”.

A cada instante de tempo n o conjunto de amostras

$$x_\mu(n), \mu = 0, 1, 2, \dots, M - 1$$

representa a transformada de Fourier de um bloco de amostras que inclui a amostra deste instante de tempo e as $M - 1$ anteriores. Desta forma os coeficientes reais $c_\mu(n)$ ponderam a magnitude do espectro deste bloco sem alterar a sua fase. O método faz uso da insensibilidade do ouvido humano à variações de fase dos sinais. Uma análise mais detalhada [22] mostra que para relações sinal/ruído até aproximadamente 6dB o efeito do ruído na fase não é perceptível. Relações sinal/ruído menores provocam uma degradação perceptível ao ouvido humano.

O coeficiente c_μ de cada canal é ajustado individualmente como função da potência de ruído relativa Q_μ do próprio canal:

$$c_\mu = S_i(Q_\mu) \quad (2.59)$$

$$Q_\mu = \frac{E\{|r_\mu|^2\}}{E\{|x_\mu|^2\}}. \quad (2.60)$$

Nessa última equação, $E\{\cdot\}$ indica a potência média local estimada. Algumas leis de supressão S_i mais comumente empregadas na literatura estão indicadas na Tabela 2.5 na página seguinte e esquematizadas na Figura 2.16 na próxima página. Estas leis são parcialmente baseadas em critérios matemáticos e parcialmente em critérios psicoacústicos.

A limitação fundamental deste tipo de procedimento é que ele admite que o ruído é estacionário. Como essa suposição não se verifica

Solução de Wiener aproximada	$S_1(Q_\mu) = 1 - Q_\mu$
Subtração de potência	$S_2(Q_\mu) = \sqrt{1 - Q_\mu}$
Subtração de magnitude	$S_3(Q_\mu) = 1 - \sqrt{Q_\mu}$

Tabela 2.5: Regras para supressão de ruído.

Figura 2.16: Leis para supressão de ruído.

para a maior parte dos casos práticos, o ruído efetivamente superposto ao sinal de voz tem características ligeiramente diferentes daquelas estimadas, resultando assim um ruído residual superposto ao sinal reconstruído. Experiências demonstram que este ruído residual tem muitas vezes o caráter de *tons musicais* de fundo. Além disso, deve-se observar que atenuações introduzidas nos canais em que o ruído é mais intenso também afetam as componentes do sinal de voz daqueles canais, o que no caso de baixas relações sinal/ruído provoca distorção do sinal de voz reconstruído que é tanto maior quanto maior for o grau de supressão de ruído desejado. Existe portanto um compromisso entre o nível de supressão de ruído desejável e o nível de distorção aceitável do

sinal de voz.

2.2.5 Estimação estatística do ruído

O procedimento descrito na seção anterior assume que uma estimativa do ruído está disponível para ser utilizada na equação (2.59). Obviamente essa estimativa precisa ser obtida a partir do sinal de voz já contaminado.

Uma técnica possível emprega um *detector de atividade vocal* [27, 28] e mede a potência do sinal apenas nos trechos onde não há sinal de voz presente. A dificuldade com esse método, mesmo que um esquema confiável de detecção de voz seja usado, é que as estatísticas do ruído não podem ser atualizadas nos trechos onde há atividade vocal. O *rastreamento* do nível de ruído se revela lento para as aplicações visadas por este trabalho.

Outra técnica, mais adequada ao uso em automóveis, baseia-se na observação de que a potência instantânea—suavizada—de um sinal ruidoso de voz apresenta um comportamento típico com “picos” e “vales”. A estimativa da potência do ruído baseada nesses vales é bastante precisa.

O que segue é uma descrição resumida desse segundo método, adaptada para estimar a potência do ruído em cada uma das M sub-bandas obtidas após análise espectral conforme descrito na parte inicial da Seção 2.2.1. Martin [29] apresenta o algoritmo em detalhes.

Estatísticas mínimas. Seja $P_\mu(k)$ a estimativa da potência instantânea do sinal na banda μ , ou seja,

$$P_\mu(k) = x_\mu(k) \cdot x_\mu(k). \quad (2.61)$$

O objetivo é encontrar o valor mínimo de $P_\mu(k)$ dentro de uma janela de comprimento L , decomposta em N_w sub-janelas de comprimento L_w

$$L = N_w L_w. \quad (2.62)$$

A escolha de N_w e L_w obedece aos seguintes critérios:

- L deve ser grande para que a janela acomode os picos de atividade vocal;
- L deve ser pequeno para que a estimativa acompanhe variações no nível de ruído;
- L_w é menor que L para permitir eventuais ajustes rápidos e diminuir as necessidades de armazenamento do algoritmo.

A cada L_w amostras, o valor mínimo $P_{\mu,min}(k)$ é guardado. Dois casos podem ocorrer:

1. Se o valor mínimo apresenta comportamento monotonicamente crescente nas últimas N_w sub-janelas, a estimativa do ruído é igualada a esse valor mínimo:

$$R_\mu(k) = P_{\mu,min}(k); \quad (2.63)$$

2. Se, ao contrário, o comportamento não é monotônico, a estimativa do ruído é feita igual ao menor valor mínimo das últimas N_w sub-janelas:

$$R_\mu(k) = \min \{P_{\mu,min}(k), \dots, P_{\mu,min}(k - [N_w - 1]L_w)\}. \quad (2.64)$$

Martin [29] demonstra ainda que essa estimativa possui um viés que depende do tamanho L da janela e do número de sub-bandas M , e justifica assim o uso de um *fator de correção* $1,5 < OF < 3$ para compensar o problema.

2.2.6 Ponderação psicoacústica

A subtração espectral convencional deixa no sinal processado um desagradável ruído residual que não preserva as características espectrais

do ruído original (existem [30] algumas formas tradicionais de minimizar o problema).

Recentemente, alguns trabalhos [31, 32, 33] publicados na literatura apontam para a necessidade de entender como os sinais de voz contaminados com ruído são interpretados pelo sistema auditivo humano. Indicam ainda como acrescentar aos algoritmos convencionais de subtração espectral alguns cálculos a mais de forma a incorporar certos fenômenos psicoacústicos que garantem a inaudibilidade dos artefatos conhecidos como “tons musicais”.

Percepção de ruído na presença de sinais de voz. A preocupação com o efeito psicoacústico de ruído surgiu no campo de codificação e compressão de sinais de voz para transmissão em canais de banda limitada ou diminuir as necessidades de armazenamento. Exemplo muito conhecido é o padrão MPEG de compressão de áudio [34] e suas variantes, hoje usado largamente em transmissão de TV por satélite, gravação de áudio digital (e.g. minidisc) e produtos de consumo (e.g. placas de som para computadores pessoais).

Considere (cf. Figura 2.6 na página 20) que a um sinal de voz $s(k)$ é somado um ruído ou interferência $r(k)$. O volume de ruído percebido é determinado não só por sua potência total mas também pela distribuição de energia de $r(k)$ e de $s(k)$ ao longo da cóclea⁷. O volume do ruído pode ser reduzido ou até completamente eliminado pelo sinal $s(k)$, um fenômeno conhecido como *mascaramento auditivo*, e seu cálculo pode ser feito com base em tabelas disponíveis na literatura [34, 35, 36].

Cálculo da máscara auditiva

Se o ruído presente no sinal $x(k) = s(k) + r(k)$ for atenuado abaixo de um certo limiar, o qual varia de frequência para frequência e depende

⁷Cóclea ou membrana basilar: longo canal de forma espiral, cheio de líquido, que se comunica por meio de três pequenos ossos (martelo, bigorna e estribo) com a membrana timpânica. Está situado no ouvido interno.

do espectro do sinal $s(k)$, então ele se torna inaudível. Para o cálculo desse limiar, vários passos estão envolvidos:

- Análise por bandas críticas do sinal
- Aplicação da função de espalhamento
- Cálculo do limiar auditivo
- Relacionamento desse limiar com o limiar no espectro original

Análise por bandas críticas. O sinal é analisado por quadros de N amostras sucessivas. A cada quadro, o espectro complexo é obtido com uma DFT⁸:

$$x(k) \xrightarrow{DFT} X(f) = X_{re}(f) + jX_{im}(f), \quad (2.65)$$

onde os índices re e im denotam, respectivamente, a parte real e imaginária do sinal. Esse espectro é convertido em potência

$$P_x(f) = |X_{re}(f)|^2 + |X_{im}(f)|^2 \quad (2.66)$$

e a energia contida em cada uma das bandas críticas (ver Tabela 2.6) é somada

$$B_n = \sum_{f=f_{1,n}}^{f_{2,n}} P_x(f), \quad (2.67)$$

onde $f_{1,n}$ e $f_{2,n}$ são, respectivamente, o início e fim de cada banda n .

Esse procedimento simula a transformação de frequência em espaço que ocorre na membrana basilar (esta funciona como um analisador de espectro segundo as bandas críticas), já que cada porção da membrana responde melhor a certa faixa de frequências.

Passagem pela função de espalhamento. Apesar da percepção auditiva se dar em termos das bandas críticas, porções do sinal em cada

⁸DFT: Discrete Fourier Transform. Versão discreta da transformada de Fourier, normalmente calculada por meio de um algoritmo chamado FFT (Fast Fourier Transform) devido a motivos práticos.

n (banda)	Início (Hz)	Fim (Hz)
1	0	100
2	100	200
3	200	300
4	300	400
5	400	510
6	510	630
7	630	770
8	770	920
9	920	1080
10	1080	1270
11	1270	1480
12	1480	1720
13	1720	2000
14	2000	2320
15	2320	2700
16	2700	3150
17	3150	3700
18	3700	4400

Tabela 2.6: Bandas críticas na faixa telefônica. [37] Escala de Bark.

banda excita também regiões vizinhas na cóclea, de modo que ocorre certo espalhamento de energia. Matematicamente, esse fenômeno é levado em conta por uma convolução do espectro em bandas críticas com uma função conveniente. Em unidades de dB:

$$10 \log_{10} S_n = 15,81 + 7,5(n + 0,474) - 17,5\sqrt{1 + (n + 0,474)^2}, \quad (2.68)$$

sendo que n é o *deslocamento* em número de bandas críticas e varia de -17 a $+17$ para aplicações em processamento de voz (e.g. telefone). O gráfico na Figura 2.17 na próxima página mostra o comportamento da função acima para essa faixa de valores de n .

Figura 2.17: Função de espalhamento.

Cálculo do limiar auditivo. Após a convolução de S_n com B_n , ou seja,

$$C_n = S_n * B_n, \quad (2.69)$$

o limiar de mascaramento T_n pode ser calculado por duas regras:

1. Se um ruído de banda larga for mascarado por um tom, o limiar é colocado $(14,5 + n)$ dB abaixo de C_n ;
2. Para sinal de banda larga mascarando um tom, o limiar deve ficar 5,5 dB abaixo de C_n .

Na prática os sinais se situam entre os dois extremos, e uma ponderação adequada deve ser dada para cada uma das regras. Esse peso é obtido com a relação entre a área do espectro em bandas críticas (G_m) e sua média aritmética (A_m):

$$SFM = \frac{G_m}{A_m} \quad (2.70)$$

onde SFM vem de “Spectral Flatness Measure” e seu valor está entre 0 e 1.

Relação com limiar no espectro original. Duas coisas ainda devem ser feitas: uma delas seria a deconvolução com a função de espalhamento, substituído por uma simples de-normalização. O ganho de energia provocado pelo espalhamento é aplicado inversamente a cada

banda. A segunda coisa é converter as bandas críticas para um espectro linear novamente, o que se faz repetindo valores nas faixas de frequência correspondentes a cada banda crítica. Com isso, o cálculo do valor do limiar se completa.

O redutor de ruído convencional pode ser utilizado para se obter uma estimativa do sinal de voz e partir deste calcular o limiar de mascaramento T_n para cada frequência. Suponha que r_n representa a energia local em cada banda, e λ_R é um nível residual de ruído que se deseja manter no sinal. Então a regra de atenuação espectral é a seguinte:

$$c_n = \min \left(\sqrt{\frac{T_n}{r_n}} + \lambda_R, 1, 0 \right). \quad (2.71)$$

Note que nos trechos de ausência de sinal de voz o limiar se torna baixo e o ruído é atenuado por um fator pré-determinado λ_R (0,1, por exemplo, para 20 dB). Por outro lado, se a energia do sinal de voz estiver alta em algumas bandas, o limiar de percepção se torna alto e os coeficientes $c(n)$ se aproximam da unidade (deixando passar intacto o sinal nas bandas correspondentes). O fator λ_R estabelece então um controle sobre o ruído residual perceptível, eliminando os artefatos (tons musicais) que surgem no redutor convencional devido à variação do limiar de percepção. Automaticamente, a distorção na voz é diminuída ao máximo possível para uma dada relação sinal-ruído.

2.3 Comentários

A base teórica do sistema de cancelamento de eco acústico e redução de ruído foi apresentada neste capítulo.

Crê-se que as razões para a escolha do algoritmo EFLSL foram explicadas: velocidade de convergência, robustez numérica e baixa complexidade computacional (comparado a outros algoritmos da mesma família).

Acredita-se também que o algoritmo de subtração espectral, dependente de adequadas estruturas de análise e síntese espectrais, foi justificado. Com a inclusão da técnica de estimação do ruído por estatísticas mínimas e dos critérios psicoacústicos, o subsistema de redução de ruído descrito tem condições para operar em ambientes ruidosos e com características variantes no tempo.

O próximo capítulo enfoca os aspectos práticos do sistema, sua implementação em processador digital de sinais para operação em tempo real, as escolhas feitas para alguns parâmetros introduzidos neste capítulo teórico. Referências abundantes às listagens dos programas criados para projetar e executar os algoritmos de cancelamento de eco e redução de ruído acompanham as explicações do capítulo seguinte.

Capítulo 3

Implementação Prática

As discussões do capítulo anterior foram aplicadas na construção de um protótipo funcional, capaz de demonstrar a utilidade dos filtros adaptativos da família RLS e também dos processos de análise, modificação e síntese espectrais.

Este capítulo inicia com uma exposição breve da arquitetura do processador digital de sinais escolhido para o protótipo e segue mostrando a simulação e implementação eficiente do algoritmo LSL nesse processador. Um processo semi-formal desenvolvido especialmente para este projeto, mas com potencial para uso na implementação de algoritmos adaptativos em geral, é detalhado.

A implementação do sub-sistema de análise e síntese espectrais, usando eficientes estruturas polifásicas, vem logo a seguir. Os aspectos práticos do processo de estimação de ruído a partir do sinal capturado pelo microfone são expostos, bem como as rotinas de supressão espectral de ruído, com e sem ponderação psicoacústica.

O capítulo termina com algumas observações sobre a integração do filtro adaptativo e do redutor de ruído em um único sistema. Além disso, são feitos alguns comentários sobre a montagem do protótipo funcional.

3.1 Processador SHARC

A implementação dos algoritmos usados no sistema de cancelamento de eco acústico e redução de ruído foi feita em processador digital de sinais da Analog Devices¹, modelo SHARC² ADSP-21061 (doravante denominado somente “SHARC”). Alguns detalhes de sua arquitetura seguem descritos nesta seção, de forma a facilitar a compreensão principalmente da implementação eficiente do algoritmo de cancelamento de eco acústico.

SHARC é um processador de ponto flutuante de 32 bits que pode executar qualquer instrução em um único ciclo de máquina. Outras características relevantes para a implementação em questão são (ver Figura 3.1 na página seguinte):

- Memória interna de 1 Mbit, que pode ser organizada em 32K palavras de 32 bits cada ou 16K instruções de 48 bits cada;
- Dois barramentos separados para acesso à memória interna permitem que dois operandos sejam transferidos em um mesmo ciclo;
- As operações de multiplicação (e de inversão) são realizadas por uma unidade computacional especializada, enquanto uma unidade aritmética se encarrega das adições e subtrações;
- Um conjunto de 16 registradores serve para guardar os operandos mais utilizados e minimizar a quantidade de acessos à memória;
- A programação pode ser feita usando uma sintaxe algébrica da linguagem *assembly* ou utilizando a linguagem “C”.

Operações em paralelo. Os 16 registradores, onde ficam os fatores e parcelas e também os resultados das operações, podem ser divididos em quatro grupos de quatro elementos cada um, de acordo com a Tabela 3.1 na página 53.

¹Consultar <http://www.analog.com> para mais informações.

²SHARC: Super Harvard Architecture Computer

Figura 3.1: Diagrama de blocos parcial do processador ADSP-2106x SHARC.
Adaptado de [38].

Uma adição/subtração e uma multiplicação podem ser executadas em paralelo pelo processador, desde que se atendam as seguintes condições:

1. Um dos fatores deve estar no grupo A e o outro no grupo B; e
2. Uma das parcelas deve estar no grupo C e a outra no grupo D.

A Figura 3.2 na página seguinte ilustra essas restrições e também indica que os resultados das operações paralelas podem ser colocados em qualquer um dos registradores. Para manter as unidades computa-

Grupo	Registradores
A	R0, R1, R2 e R3
B	R4, R5, R6 e R7
C	R8, R9, R10 e R11
D	R12, R13, R14 e R15

Tabela 3.1: Grupos de registradores no SHARC.

cionais operando de modo eficiente, a colocação dos fatores, parcelas e resultados intermediários deve ser cuidadosamente avaliada.

Figura 3.2: Diagrama de condições para funcionamento paralelo das unidades computacionais do SHARC.

Ponteiros. A transferência de dados entre memória e registradores pode ser feita em paralelo com as operações aritméticas, usando 16 *índices de endereçamento indireto*, identificados pelos códigos I0–I15. Metade desses índices (I0–I7) realiza transferências sobre o barramento

DM (*Data Memory*) e a outra metade (I8–I15), sobre o barramento PM (*Program Memory*).

Para realizar o incremento e decremento desses índices, há 16 *modificadores de índice*, identificados por M0–M15. Cada modificador pode ser utilizado para alterar, durante uma transferência, o valor de qualquer índice I0–I15.

Esses índices atuam portanto como ponteiros para as diversas variáveis envolvidas na execução do algoritmo. Fazendo escolhas adequadas sobre quais índices são atrelados a quais variáveis, pode-se garantir que os operandos ficam disponíveis às unidades computacionais no momento necessário.

3.2 Cancelamento de eco acústico

A implementação do algoritmo EFLSL modificado foi dividida em duas partes: um núcleo que é executado repetidamente M vezes—onde M é o número de coeficientes do filtro—e um código de inicialização e finalização, que é executado somente uma vez a cada amostra de entrada.

3.2.1 Núcleo

A Tabela 3.2, cópia parcial da Tabela 2.4 na página 19 e repetida aqui por conveniência, mostra as equações que compõem a parte principal do algoritmo. Como a eficiência na utilização do processador é um aspecto importante, um processo de otimização foi desenvolvido e será explicado a seguir.

Ordem das equações. O primeiro passo é colocar as equações em uma ordem diferente de execução, deixando as que precisam dos resultados de outra(s) equação(ões) para o final. Depois de reordenadas, as operações formam uma espécie de “hierarquia”, onde algumas do mesmo “grupo” podem ser executadas em paralelo e outras dependem de grupos anteriores.

1	$\eta_m(n) = \eta_{m-1}(n) - \Gamma_m^f(n-1)\psi_{m-1}(n-1)$
2	$\psi_m(n) = \psi_{m-1}(n-1) - \Gamma_m^b(n-1)\eta_{m-1}(n)$
3	$b_{m-1}(n) = \gamma_{m-1}(n)\psi_{m-1}(n)$
4	$\xi_{m-1}^b(n) = \lambda_L \xi_{m-1}^b(n-1) + b_{m-1}(n)\psi_{m-1}(n) + \zeta$
5	$\bar{b}_{m-1}(n) = \frac{b_{m-1}(n)}{\xi_{m-1}^b(n)}$
6	$\gamma_m(n) = \gamma_{m-1}(n) - \bar{b}_{m-1}(n)b_{m-1}(n)$
7	$f_{m-1}(n) = \gamma_{m-1}(n-1)\eta_{m-1}(n)$
8	$\xi_{m-1}^f(n) = \lambda_L \xi_{m-1}^f(n-1) + f_{m-1}(n)\eta_{m-1}(n) + \zeta$
9	$\bar{f}_{m-1}(n) = \frac{f_{m-1}(n)}{\xi_{m-1}^f(n)}$
10	$\Gamma_m^f(n) = \Gamma_m^f(n-1) + \bar{b}_{m-1}(n-1)\eta_m(n)$
11	$\Gamma_m^b(n) = \Gamma_m^b(n-1) + \bar{f}_{m-1}(n)\psi_m(n)$
12	$\alpha_m(n) = \alpha_{m-1}(n) - \psi_{m-1}(n)\kappa_m^d(n-1)$
13	$\kappa_m^d(n) = \kappa_m^d(n-1) + \bar{b}_{m-1}(n)\alpha_m(n)$

Tabela 3.2: Núcleo do algoritmo EFLSL modificado.

Com isso, há uma maior possibilidade de manter a unidade aritmética (realizando multiplicações, principalmente) em carga total, sem ter que esperar por transferências de variáveis. Assim, por exemplo, a operação da linha 4 não pode ser executada sem o resultado da linha 3, e precisa ser deslocada em direção ao final. Em seu lugar, a linha 7 pode ser colocada.

Identificadores. Para facilitar um tratamento mais formal das equações da Tabela 3.2, as variáveis são identificadas. Notando que vários operandos se referem aos mesmos símbolos com índices (ou tempos) diferentes, um esquema de identificação que agrupe os operandos por similaridade foi escolhido. A Tabela 3.3 mostra as equações reordenadas e já identificadas—por meio de caixas com letras minúsculas do alfabeto latino. A numeração das equações foi preservada para comparação.

O segundo passo é escolher uma representação das equações mos-

1	$\eta_m(n) = \eta_{m-1}(n) - \Gamma_m^f(n-1)\psi_{m-1}(n-1)$ a' a b c
7	$f_{m-1}(n) = \gamma_{m-1}(n-1)\eta_{m-1}(n)$ h d
2	$\psi_m(n) = \psi_{m-1}(n-1) - \Gamma_m^b(n-1)\eta_{m-1}(n)$ c'' e
3	$b_{m-1}(n) = \gamma_{m-1}(n)\psi_{m-1}(n)$ i d' c'
12	$\alpha_m(n) = \alpha_{m-1}(n) - \psi_{m-1}(n)\kappa_m^d(n-1)$ f' f g
10	$\Gamma_m^f(n) = \Gamma_m^f(n-1) + \bar{b}_{m-1}(n-1)\eta_m(n)$ b' j
8	$\xi_{m-1}^f(n) = \lambda_L \xi_{m-1}^f(n-1) + f_{m-1}(n)\eta_{m-1}(n) + \zeta$ l' k l m
4	$\xi_{m-1}^b(n) = \lambda_L \xi_{m-1}^b(n-1) + b_{m-1}(n)\psi_{m-1}(n) + \zeta$ n' n
9	$\bar{f}_{m-1}(n) = \frac{f_{m-1}(n)}{\xi_{m-1}^f(n)}$ o
5	$\bar{b}_{m-1}(n) = \frac{b_{m-1}(n)}{\xi_{m-1}^b(n)}$ j'
11	$\Gamma_m^b(n) = \Gamma_m^b(n-1) + \bar{f}_{m-1}(n)\psi_m(n)$ e'
13	$\kappa_m^d(n) = \kappa_m^d(n-1) + \bar{b}_{m-1}(n)\alpha_m(n)$ g'
6	$\gamma_m(n) = \gamma_{m-1}(n) - \bar{b}_{m-1}(n)b_{m-1}(n)$ d''

Tabela 3.3: Núcleo do algoritmo EFLSL modificado, com equações reordenadas e identificadores atrelados aos operandos.

tradas na Tabela 3.2 mais adequada às características do SHARC. Isso se faz necessário por dois motivos: (1) não há registradores internos (ou alguns precisam ser reservados para expansão futura) no DSP suficientes para armazenar todas as variáveis; e (2) o paralelismo das operações (ver Seção 3.1) depende da escolha dos registradores adequados

para cada operando.

A Tabela 3.4 mostra os identificadores e em quais operações está envolvido. Os fatores aparecem como dois círculos ($\circ$ ou $\bullet$) iguais na mesma coluna. Os produtos intermediários se somam³ com outros produtos intermediários ou com parcelas indicadas por círculos diferentes. O resultado de cada equação é representado por uma estrela ($\star$).

Assim, a primeira coluna representa as operações necessárias à equação 1: $\boxed{b}$ e $\boxed{c}$ são multiplicados ($\circ$) e o resultado subtraído de $\boxed{a}$ ($\bullet$), fornecendo $\boxed{j}$ ($\star$). Nota-se que $\boxed{a}$ é utilizado ainda em mais 3 equações e depois não é mais necessário. Outro exemplo: Na equação 8, $\boxed{a}$ e $\boxed{h}$ ($\bullet$) são multiplicados entre si, bem como $\boxed{k}$ e $\boxed{l}$ ($\circ$). Esses resultados parciais se somam e o resultado final ($\star$) é colocado em $\boxed{r}$.

Registradores. Dispor as variáveis nos registradores, seguindo a hierarquia de operações, é o terceiro passo no processo de otimização.

Constrói-se então uma tabela com registradores nas colunas e ciclos de execução nas linhas. Para cada ciclo do processador, os operandos são dispostos de acordo com regras bem definidas e os resultados (parciais ou totais) são colocados em registradores adequados.

Parte-se da Tabela 3.4 e, com pequeno auxílio da Tabela 3.2 para distinguir entre adições e subtrações, constrói-se a Tabela 3.5 de acordo com o seguinte método:

1. Dispor multiplicações até que (a) não haja mais fatores disponíveis ou (b) qualquer grupo de registradores já esteja 75% ocupado;
2. Fazer uma soma/subtração pendente;
3. Voltar ao passo 1.

Além disso, o processo de preenchimento da tabela segue algumas regras secundárias, abaixo relacionadas e acompanhadas de exemplos tirados da Tabela 3.5:

³A diferença entre somas e subtrações não é representada na tabela.

	Equações													
ID	1	7	2	3	12	10	8	4	9	5	11	13	6	
a	●	○	○				●							
b	○					●								
c	○		●											
d		○												
e			○								●			
c'				○	○		●							
d'				○									●	
f					●									
g					○							●		
a'	★					○								
h		★					●		○					
c''			★								○			
i				★				●		○			○	
f''					★							○		
j						○								
k							○	○						
l							○							
m							◇	◇						
n								○						
b'						★								
l'							★		○					
n'								★		○				
o									★		○			
j''										★		○	○	
e'											★			
g'												★		
d''													★	

Tabela 3.4: Representação alternativa das equações do algoritmo EFLSL.

1. Nas multiplicações, um dos fatores deve estar no grupo A e o outro no grupo B (Seção 3.1). Exemplos: a e d no ciclo 2, o e c'' no ciclo 13;
2. As somas/subtrações precisam ter uma parcelas colocada no grupo C e a outra no grupo D. Exemplos: a e bc no ciclo 2, ah e kl no ciclo 9;
3. Resultados (parciais ou não) usados posteriormente em operações de multiplicação devem ser colocados preferencialmente em registradores adequados nos grupos A ou B. Exemplos: a' é gerado no ciclo 2 e é utilizado no ciclo 6 (a'j);
4. Resultados (parciais ou não) usados posteriormente como parce-

las devem ser colocados preferencialmente em registradores adequados nos grupos C ou D. As parcelas que são posteriormente subtraídas precisam ficar no grupo D. Parcelas que participam de operação conjunta devem ser colocadas em grupos diferentes. Exemplos: $c'g$, gerado no ciclo 5, aparece no grupo D, para ser subtraído de f no ciclo 6;

5. Em cada grupo, o mínimo de registradores devem ser utilizados em um dado momento da iteração;
6. Não sobrepor conteúdo de registrador quando esse conteúdo ainda for utilizado em operação posterior no mesmo grupo;
7. A divisão é feita em duas etapas: uma inversão que não é paralelizável e uma multiplicação normal.

A Tabela 3.5 mostra, para cada ciclo (linha), o valor que cada registrador possui imediatamente *após* o ciclo anterior e imediatamente *antes* do ciclo corrente. A escrita das operações em *assembly* algébrico deriva diretamente dessa tabela.

Transferência de dados. O quarto passo é relacionar as operações de transferência de dados entre memória e registradores, todas realizadas por meio de acesso indireto.

Notando-se que algumas das variáveis da Tabela 3.3 não precisam ser guardadas de iteração para iteração (porque são constantes ou porque são intermediárias), conclui-se que as restantes podem ser representadas por *vetores*.

Os tamanhos dos vetores são escolhidos a partir de análise da Tabela 3.2. Vale lembrar que $m = 1 \dots M$, com M igual ao número de coeficientes do filtro.

Cada variável que precisa ser transferida a cada iteração é apontada por um índice I0–I15, de acordo com a Tabela 3.6.

Os índices são escolhidos de modo a possibilitar a transferência simultânea de dois operandos (um deles pelos índices I0–I7 e o outro

#	Multiplicador						Acumulador					
	A			B			C			D		
	R0	R1	R2	R4	R5	R6	R8	R9	R10	R12	R13	R14
1	b			c								
2	a			d			a			bc		
3	a		a-bc (a')	ad (h)	e							
4		c'			d'		c				ae	
5		c'			c'd' (i)	g	c-ae (c'')					
6			a'			j	f					c'g
7	a		f-c'g (f')	h			a'j			b		
8	k					l	a'j+b (b')	ah				
9		c'			i			ah		kl		
10	k					n		ah+kl	c'i	m		
11	l'=ah+ kl+m										kn	
12	1/l'			h					c'i		kn	
13	o=h/l'			c''					c'i+kn	m		
14		n'=c'i+ kn+m					oc''					
15		1/n'			i		oc''			e		
16		j'=i/n'		f''			oc''+e (e')					
17		j'			i			f'j'		g		
18							d'	f'j'+g (g')		j'i		
19							d'j'i (d'')					

Tabela 3.5: Quadro mostrando o conteúdo dos registradores imediatamente antes de cada ciclo de execução.

pelos índices I8–I15), quando necessário. Um procedimento simples facilita acomodar essa restrição:

1. Para cada linha da Tabela 3.5 onde é necessário trazer dois operandos da memória para os registradores, preencher o registrador de índice menor através do barramento DM e o outro através do barramento PM;
2. Distribuir as transferências restantes igualmente pelos dois barramentos;

Variável	Identificadores	Tamanho	Índice
η	a, a'	1	I0
Γ^f	b, b'	M	I1
ψ	c, c', c''	$2(M + 1)$	I2
$\bar{b}$	j, j'	M	I3
λ_L	k	1	I4
ξ^b	n, n'	M	I5
γ	d, d', d''	$2(M + 1)$	I8
Γ^b	e, e'	M	I9
α	f, f'	1	I10
κ^d	g, g'	M	I11
ξ^f	l, l'	M	I12
ζ	m	1	I13

Tabela 3.6: Alocação de ponteiros para as variáveis do algoritmo EFLSL.

3. Se houver identificadores atrelados a uma mesma variável sendo transferidos por dois barramentos em ciclos diferentes, optar pelo barramento atribuído no ciclo mais próximo do final.

Para cada operação de adição, subtração, ou multiplicação as variáveis precisam estar em registradores adequados para garantir o paralelismo. Após cada operação, os resultados precisam normalmente ser transferidos para a memória. A disposição utilizada para as transferências deriva portanto da observação das variáveis necessárias a cada ciclo na Tabela 3.5.

Usando os índices escolhidos, a Tabela 3.7 mostra as transferências de memória necessárias para possibilitar a execução do algoritmo. Notar que a tabela mostra tanto os operandos (registradores) copiados da memória quanto os resultados copiados de volta para a memória.

Modificadores. Com a alocação dos ponteiros definida pela Tabela 3.6 e a seqüência de transferências dada pela Tabela 3.7, os va-

Ciclo	DM	PM	Extra
0	R0 $\triangleleft$ [I1]		R4 $\triangleleft$ [I2]
1	R0 $\triangleleft$ [I0]	R4 $\triangleleft$ [I8]	R8 $\triangleleft$ [I0]
2		R5 $\triangleleft$ [I9]	
3	R1 $\triangleleft$ [I2]	R5 $\triangleleft$ [I8]	R8 $\triangleleft$ [I2]
4	R2 $\triangleright$ [I0]	R6 $\triangleleft$ [I11]	
5	R6 $\triangleleft$ [I3]	R8 $\triangleleft$ [I10]	
6	R12 $\triangleleft$ [I1]		R8 $\triangleright$ [I2]
7	R0 $\triangleleft$ [I4]	R6 $\triangleleft$ [I12]	
8	R8 $\triangleright$ [I1]	R2 $\triangleright$ [I10]	
9	R6 $\triangleleft$ [I5]	R12 $\triangleleft$ [I13]	
10			
11		R0 $\triangleright$ [I12]	
12	R4 $\triangleleft$ [I2]		
13			
14	R1 $\triangleright$ [I5]	R12 $\triangleleft$ [I9]	
15	R4 $\triangleleft$ [I10]		
16	R1 $\triangleright$ [I3]	R12 $\triangleleft$ [I11]	
17		R8 $\triangleleft$ [I8]	
18		R9 $\triangleright$ [I11]	
19		R8 $\triangleright$ [I8]	

Tabela 3.7: Diagrama de transferências entre memória e registradores para o algoritmo EFLSL.

lores dos modificadores de índice a serem adotados podem ser obtidos facilmente.

Grande parte das operações de transferência incrementam os ponteiros de um valor unitário, mas alguns dos índices são mais complexos (I2 e I8, por exemplo) e requerem saltos de valor $M + 1$ e $M + 2$. A Tabela 3.5 serve também de base para a determinação dos incrementos dos índices para cada operação de transferência.

Todas as operações de transferência utilizam endereçamento *indireto* com modificação do índice *após* a transferência. Modificadores do

conjunto M0–M7 são utilizados com os índices I0–I7, e alguns do conjunto M8–M15 são usados com os índices I8–I15. A Tabela 3.8 mostra os valores inicializados para os modificadores.

Modificador	Valor	Modificador	Valor
M0	$M + 1$	M8	$M + 1$
M1	$M + 2$	M9	$M + 2$
M5	0	M13	0
M6	1	M14	1

Tabela 3.8: Modificadores de índice

Ajustes finais. O último passo inclui alguns ajustes para eliminar a coluna “extra” da Tabela 3.7 e os ciclos desnecessários:

1. Carregar antecipadamente registradores (por exemplo: “extra” do ciclo 3 feito através de DM no ciclo 2);
2. Atrasar o salvamento de registradores (por exemplo: PM no ciclo 18 para ciclo 6 da iteração seguinte);

Alguns desses ajustes podem requerer a realocação do uso dos registradores (guardar resultados em R9 e R10 ao invés de R8, por exemplo). A listagem assembly anexada no Apêndice B.2, no bloco demarcado pela etiqueta `flt`, mostra o resultado final implementado.

Inicialização e finalização. Além do núcleo de recursão em ordem, para cada amostra que chega ao filtro as condições iniciais da Tabela 2.4 na página 19 são implementadas. Alguns ajustes (ciclos antecipados) mencionados acima também são colocados junto a essa *inicialização*. As operações atrasadas nos ciclos são colocadas na porção de *finalização*.

3.2.2 Complexidade computacional

A implementação feita compreende 17 ciclos por iteração—são M iterações, que corresponde à ordem do filtro—e está otimizada ao máximo da capacidade do DSP. Mais detalhes podem ser obtidos no código fonte construído com base no procedimento descrito. O SHARC utilizado em laboratório é um ADSP-21061, com ciclo de instruções de 25 ns, e com capacidade para 80 MFLOPS sustentado. Todas as operações de adição/subtração estão “mascaradas” sob as operações de multiplicação. Com taxa de amostragem de 8 kHz, são 5000 ciclos úteis para o processamento, e então se espera que o número de coeficientes máximo com que se possa trabalhar seja

$$M_{max} = \frac{5000}{17} \simeq 294. \quad (3.1)$$

Contando com instruções de inicialização, interface com os conversores e ciclos para atendimento da interrupção, conseguiu-se rodar o filtro em tempo real com 290 coeficientes, permitindo ao mesmo acompanhar respostas impulsivas com duração de até cerca de 36 ms. Isso se compara com o desempenho conseguido com a implementação direta em linguagem C, que foi de apenas 54 coeficientes.

3.3 Subtração espectral do ruído

O subsistema de redução de ruído é um software escrito em linguagem C rodando em um processador digital de sinais de última geração. O coração do subsistema é um bloco de subtração espectral (Seção 2.2.4) que faz a estimativa da parcela de voz presente no sinal contaminado com ruído. A subtração espectral depende em grande parte de uma eficiente estrutura de análise e síntese espectral, do cálculo de um limiar de audibilidade do ruído e de um processo contínuo de estimação do nível de ruído presente no sinal.

O leitor é convidado a se referir ao código fonte parcial listado no Apêndice B.1 desta dissertação e também à Figura 2.6 na página 20.

3.3.1 Processamento

A rotina de análise divide o sinal em $M = 64$ faixas de frequência por meio de uma estrutura polifásica (cf. Seção 2.2.3).

Cada uma dessas faixas de frequência tem uma largura 64 vezes menor que a largura do espectro original. Uma redução *crítica* da taxa de amostragem ($r = M$) dizimaria o sinal de cada faixa por 64. Para facilitar o projeto dos filtros e evitar o efeito do rebatimento (parte da energia de uma banda caindo na banda adjacente) sem rearranjos complexos da estrutura [24], optou-se por uma redução *sub-crítica* com $r = M/2$.

Na síntese o sinal no tempo é recomposto a partir das 64 bandas por um processamento inverso ao da análise: a taxa do sinal de cada faixa é aumentada de uma razão $r = M/2$ e as faixas são combinadas adequadamente.

A rotina `spr0_asserted()` (ver código-fonte) se encarrega das etapas de mudança na taxa de amostragem (implicitamente realizada pelas chaves comutadoras das Figuras 2.14 e 2.15) e filtragem.

Nessa rotina, que é na verdade o tratamento de interrupção de *hardware* acionada pela chegada de uma amostra do sinal, quatro operações de filtragem FIR (função `fir()` da biblioteca) rodam na taxa original a cada amostra recebida: dois filtros relativos à análise e dois para síntese.

Note que um dos filtros de análise recebe sempre uma amostra de valor zero, devido à dizimação não-crítica. Da mesma forma, a saída de um dos filtros de síntese não é usada. A cada amostra, os filtros são trocados (são 64 filtros distintos) por meio de duas chaves comutadoras.

A cada 32 amostras de entrada, o vetor de saída dos filtros se completa com 64 amostras. A rotina então sinaliza outra interrupção (de

software) e o processo de modulação é realizado em bloco por meio de uma DFT.

A rotina `usr0_asserted()`, tratamento da interrupção de *software* mencionada, tem em seu início (linhas 289–311) uma implementação clássica [39] da transformada de Fourier rápida para sinais reais. Note que somente metade das bandas é guardada para evitar redundância (o espectro do sinal de entrada é hermitiano⁴) e uma FFT de 32 pontos (rotina da biblioteca) é usada em conjunto com um processo necessário de “desempacotamento”.

O final da rotina `usr0_asserted()` (linhas 460–485) é responsável por fazer o inverso, isto é, de-modular de volta os sinais para serem filtrados na síntese. Novamente se nota uma implementação clássica da transformada inversa de Fourier rápida, para sinais com espectro hermitiano.

A cada dois blocos de 32 amostras, portanto, a chave comutadora (realizada com a variável `index` na rotina `spr0_asserted()`) volta ao início e recomeça sua operação.

“Double-buffering”. As operações de filtragem e reamostragem interrompem várias vezes o processamento em bloco (FFT), mas a integridade das variáveis nos dois contextos de execução é inteiramente preservada pelo processador. Também é importante para essa execução *paralela* de tarefas a inclusão de duas áreas de armazenamento temporário para as amostras filtradas na análise (`mi_active`, `mi_inact`) e outras duas áreas de armazenamento temporário para as amostras que estão aguardando filtragem na síntese (`mo_active`, `mo_inact`). Essa técnica é conhecida como “double-buffering” e consiste em algumas trocas de ponteiros apenas.

Subtração espectral. O centro da rotina `usr0_asserted()` é responsável pelo bloco de *controle adaptativo de atenuação*, dividido

⁴Hermitiano: a parte de frequências negativas do espectro é igual em magnitude à parte de frequências positivas, mas a fase tem sinal oposto.

nas seguintes etapas:

1. Eliminação das frequências abaixo de 300 Hz e acima de 3400 Hz (linhas 316–326). Essas componentes não devem estar presentes em um canal telefônico e não precisam ser processadas. Dessa forma, a carga computacional diminui e parcelas do ruído são automaticamente eliminadas;
2. Subtração espectral preliminar (linhas 329–363), pela regra “S3”. O espectro obtido é usado como estimativa do sinal *mascarante* (ver Seção 2.2.6);
3. Estimação estatística do ruído em cada faixa do espectro (linhas 365–391), seguindo o algoritmo do Martin (Seção 2.2.5). Notar a presença do fator OSF na estimativa (empiricamente determinado como 3,0), para compensar o viés do estimador;
4. Cálculo do limiar psicoacústico de audibilidade do ruído;
5. Subtração espectral definitiva (linhas 444–454), de acordo com a equação (2.71). A variável `NivelResidual` na rotina limita a redução no nível desejado.

A estimação de energia é feita utilizando uma média exponencial (filtro IIR de primeira ordem). Esse tipo de estimativa não é viciada, já que utiliza um número infinito de amostras. O peso maior para as amostras mais recentes permite que a energia acompanhe o caráter não-estacionário dos sinais envolvidos. As variáveis `SinalTC` e `RuidoTC` são os fatores de amortecimento para essas médias exponenciais (linhas 333, 351, 382 e 452).

3.3.2 Filtros e tabelas

A capacidade do banco de filtros recompor, após análise e síntese, de forma quase perfeita, o sinal apresentado à sua entrada é um aspecto

muito importante desse sistema de redução de ruído. Conforme descrição detalhada na Seção 2.2.2 no capítulo de aspectos teóricos, para $r = M/2$ não há preocupação com o rebatimento (“aliasing”) na banda j se a atenuação do filtro é suficientemente alta no centro das bandas $j - 1$ e $j + 1$.

Como se verifica acompanhando a listagem no Apêndice A.3, uma janela de Kaiser [25] foi usada para o projeto do filtro passa-baixas usado como protótipo. A convolução da janela com ela mesma (linha 65 da listagem), no tempo, fornece uma resposta em frequência garantidamente positiva. Isso é necessário para que se possa extrair a raiz quadrada após a multiplicação, no tempo, pela resposta do filtro ideal.

Como a atenuação efetiva do filtro se altera devido a esse processo, uma ligeira correção empírica da atenuação foi utilizada (linhas 46–47). O cálculo da versão de fase mínima está na linha 81.

A Figura 3.3 mostra a resposta impulsiva obtida com esse procedimento, para $M = 64$ bandas e atenuação de 60 dB no centro da banda adjacente. A forma da resposta revela um atraso de cerca de 100 amostras ($\sim 12,5$ ms com sinais amostrados a 8kHz) causado pelo filtro.

Na Figura 3.4, que mostra o comportamento da magnitude e fase da resposta em frequência do filtro passa-baixas apenas para frequências próximas às primeiras bandas, observa-se que a atenuação de 60 dB é atingida exatamente no centro da faixa adjacente ($f = \frac{2}{M} = 0,03125$) como desejado. Para frequências mais distantes da origem, a atenuação aumenta monotonicamente, o que colabora para ainda mais para que o efeito do rebatimento seja minimizado. Notar também que a fase, embora não-linear devido à assimetria temporal do filtro, é praticamente linear na faixa de passagem ($0 < f < \frac{1}{M} = 0,015625$).

Erro de reconstrução. Interessantes também são os gráficos mostrados na Figura 3.5. Para obtê-lo, o filtro passa-baixas é deslocado em frequência sucessivamente para o centro de cada banda

$$f_\mu = \frac{2\mu}{M}, \quad \mu = 0, 1, \dots, M - 1$$

Figura 3.3: Resposta impulsiva do filtro protótipo para $M = 64$ e especificação de 60 dB de atenuação na faixa de rejeição.

e sua resposta em frequência quadrática é acumulada. Após a soma das $M = 64$ contribuições, o que se obtém é a resposta em frequência efetiva do banco de filtros. O erro dessa somatória em relação ao valor unitário esperado é mostrado na parte inferior da figura. Na parte superior estão traçadas as primeiras curvas relativas aos filtros deslocados.

Uma estrutura polifásica é utilizada para implementar o banco de filtros. Conforme detalhado na Seção 2.2.3, é importante lembrar que basta apenas projetar o filtro passa-baixas h_0 e os outros são obtidos recorrendo à expressão (2.53), repetida aqui por conveniência:

$$h_{\nu}^{(r)}(k) = \begin{cases} 0 & \text{se } k \neq qr - \nu \\ h_0(k) & \text{se } k = qr - \nu \end{cases}. \quad (3.2)$$

A listagem do Apêndice A.4 gera os coeficientes dos filtros reamostrados nas linhas 31–35 e guarda o resultado em arquivos para serem incluídos na rotina principal do sistema de redução de ruído.

Tabelas para cálculo de FFT. Ainda são incluídos, durante a com-

Figura 3.4: Resposta em frequência do filtro protótipo para $M = 64$ e especificação de 60 dB de atenuação na faixa de rejeição.

pilação, no arquivo principal as tabelas de senos e cossenos usadas no cálculo da transformada rápida de Fourier na análise e síntese espectrais. Essas tabelas são geradas pelo programa cuja listagem é mostrada no Apêndice A.5.

3.4 Integração

O protótipo funcional do sistema utiliza duas placas de desenvolvimento, cada uma com um processador SHARC executando separadamente os dois sub-sistemas. Um processador é carregado com as instruções de máquina relativas ao cancelamento de eco e o outro é carregado com as instruções relativas à redução de ruído.

A comunicação entre os dois processadores se faz através de uma porta serial síncrona rápida, operando a 10 Mbits/s. O sinal pode passar pelo cancelamento de eco antes do redutor de ruído ou fazer o cami-

Figura 3.5: Filtros individuais e erro de reconstrução para $M = 64$ e $r = 32$.

nho oposto. As duas configurações são trocadas por meio de alterações em algumas poucas linhas do código-fonte. As listagens do Apêndice B não mostram o código responsável pela comunicação entre processadores.

3.5 Comentários

Neste capítulo foram apresentados os detalhes de implementação do algoritmo EFLSL para cancelamento de eco acústico e do algoritmo de subtração espectral para redução de ruído ambiente. As explicações, baseadas nos aspectos teóricos apresentados no capítulo anterior, referem-se constantemente às listagens finais dos programas implementados no moderno processador digital SHARC. Uma das utilidades desse capítulo, portanto, é completar os comentários colocados no próprio código-fonte (Apêndice B).

No próximo capítulo, essa implementação é colocada para ser exe-

cutada no processador e diversos resultados são apresentados na forma gráfica. As especificações técnicas dos dois subsistemas foram deixadas também para o capítulo seguinte.

Capítulo 4

Resultados

Os testes e medições feitas com o protótipo funcional descrito no capítulo anterior são agora mostrados em diversas tabelas e gráficos.

Iniciando com uma descrição do processo de obtenção dos sinais usados como entrada do sistema, o capítulo continua com os resultados de simulações e medidas em *tempo real*. Diversas situações de utilização do sistema de cancelamento de eco acústico e redução de ruído são avaliadas.

Uma análise detalhada dos pontos fortes e dos pontos fracos da implementação encerra o capítulo. Algumas idéias de aprimoramento futuro do sistema são registradas e comentadas.

4.1 Medidas em campo

O equipamento de aquisição de sinais, um microcomputador-padrão de mesa sem teclado e monitor, é controlado por um microcomputador *laptop*, para maior comodidade nas situações de medida e menor consumo de energia. Tal controle é realizado por um *software* executado em ambas as máquinas e utiliza uma ligação Ethernet comum.

O ambiente usado é DOS, em ambas as máquinas, devido aos se-

guintes fatores:

- Estabilidade e confiabilidade, sem perda de amostras;
- Maior velocidade de resposta aos comandos;
- Maior visibilidade no visor do Laptop, já que o modo de texto é ativado;

As medidas de ruído são feitas com até quatro canais simultâneos, com um programa criado especialmente para esse fim. As medidas de acoplamento acústico são realizadas com outro programa que consegue apenas gravar em dois canais simultaneamente, usando uma das placas para reproduzir arquivos de voz pré-gravados.

O conjunto de medidas foi realizado em um automóvel Escort 1.6. Nas medidas de eco acústico um sinal de voz é reproduzido no alto-falante e captado em dois microfones. Uma caixa acústica foi colocada no console à frente do câmbio e os dois microfones usados para as medidas fixados no quebra-sol do motorista e no espelho retrovisor interno.

Diversas condições foram usadas na aquisição:

- Carro parado, com o motor desligado e as janelas fechadas;
- Idem anterior, com as janelas abertas;
- Idem anterior, com o motor ligado;
- Idem anterior, com o carro em movimento;

4.2 Cancelamento de eco

Apresentam-se nesta seção alguns resultados obtidos com o bloco de cancelamento de eco acústico isolado. Os gráficos mostrados foram obtidos com o processador digital de sinais rodando o programa do Apêndice B.1. Apenas o parâmetro λ_L é variado em algumas situações mostradas aqui.

4.2.1 Resultados

A Tabela 4.1 apresenta um resumo das diversas condições avaliadas para o cancelador de eco.

Sinal local ausente			Ruído local		
#	λ_L	N	#	N	SNR (dB)
a	0,999	290	e	290	60
b	0,9994	"	f	"	40
c	0,999	512	g	"	20
d	0,9994	"	h	512	60
			i	"	40
			j	"	20

Tabela 4.1: Combinações de parâmetros para avaliação geral do cancelador de eco.

Influência de λ_L . O fator de esquecimento é um parâmetro importante no bom desempenho do cancelador de eco, e dois valores são usados para comparação: 0,999 e 0,9994, que fornecem janelas de esquecimento de 0,86s e 1,4s, respectivamente.

Os sinais de entrada do cancelador de eco são gerados a partir da resposta impulsiva medida (com 1024 pontos) para o ambiente do carro, mostrada na Figura 4.1.

O parâmetro N na tabela é o número de pontos da resposta impulsiva utilizados em cada caso, já que o número de coeficientes do filtro é fixo em 290 (ver listagem). Os casos de “a” a “d” estão mostrados na Figura 4.2.

É importante salientar que a medida de ERLE (*Echo Return Loss Enhancement*) feita do modo indicado na figura,

$$ERLE = \frac{d^2}{y^2},$$

reflete a capacidade de cancelamento de eco apenas quando o nível do sinal local é baixo ou ausente. Em simulações, é possível subtrair o

Figura 4.1: Resposta impulsiva para o veículo usado nos testes.

signal local após a filtragem para medir mais efetivamente a redução de eco.

Influência do ruído de medida. Outro fator importante na operação do cancelador de eco é o ruído de medida. Para avaliar esse fator, um ruído gaussiano de média nula, SNR dB abaixo do nível de pico do sinal de eco, foi somado artificialmente ao próprio sinal de eco. O resultado da ação do cancelador é mostrada na Figura 4.3 para respostas impulsivas de tamanhos diferentes.

Observa-se nas curvas apresentadas que o ruído de medida não interfere significativamente na medida de ERLE até uma relação sinal-ruído de 40dB. Para ruídos mais altos, essa medida não é a mais adequada para se avaliar o algoritmo de cancelamento, mas o protótipo funcional não foi projetado de modo a facilitar a medida do ERLE. De qualquer forma, mesmo que o ERLE efetivo (descontando-se o ruído somado) diminua como mostrado, o ruído de fundo começa a mascarar o eco residual, de forma que este não se torna necessariamente mais

Figura 4.2: Influência do valor de λ_L no cancelamento de eco. Na parte superior, $N = 290$, e na parte inferior, $N = 512$. Os gráficos mostram também o sinal desejado (fora de escala), para acompanhamento.

audível.

“Double-talk”. Se ao invés de ruído local, um outro sinal de voz, descorrelacionado com o sinal de entrada do filtro adaptativo, for somado ao sinal desejado $d(n)$ (ver Figura 2.1 na página 8), tem-se a situação conhecida como “double-talk”. Esse caso não é comum na prática, ocorrendo apenas quando os dois participantes de uma conversa à distância decidem falar simultaneamente, mas é interessante para avaliar a imunidade a interferências estatisticamente similares ao sinal

Figura 4.3: Influência do ruído de medida no cancelamento de eco. Na parte superior, $N = 290$, e na parte inferior, $N = 512$. Os gráficos mostram também o sinal desejado (fora de escala), para acompanhamento.

de entrada.

Para esse teste, $\lambda_L = 0,9994$ e uma resposta impulsiva de $N = 512$ coeficientes foi utilizada. Para o sinal de voz local foram adotados dois níveis de potência diferentes: 10 dB e 20 dB abaixo do nível do sinal de eco.

A Figura 4.4 mostra os resultados, onde se nota uma recuperação rápida do algoritmo após os trechos de atividade local. Em alguns períodos, a atividade local parece não perturbar o cancelador de eco (por

Figura 4.4: Influência de sinal de voz local no cancelamento de eco. O gráfico mostra também o sinal de entrada e o sinal de voz local (fora de escala), para acompanhamento.

exemplo, o pico de ERLE próximo à marca dos 3s).

4.2.2 Especificações

Os detalhes de especificação para o bloco de cancelamento de eco estão mostrados na Tabela 4.2, para o processador SHARC e o sistema de desenvolvimento utilizado no protótipo funcional.

Método	filtragem adaptativa RLS
Freq. de amostragem	8000 Hz
Faixa dinâmica	16 bits
Atenuação do eco	30–40 dB
Tam. máx. resposta impulsiva	36 ms
Carga computacional	100 % (80 MFLOPS)

Tabela 4.2: Especificações técnicas do bloco de cancelamento de eco.

4.3 Redução de ruído

Para avaliar o bloco de redução de ruído foram utilizados dois tipos de sinais de voz:

- Trecho de fala gravado em laboratório, com ruído gaussiano estacionário adicionado; e
- Trecho gravado em automóvel, sem processamento.

4.3.1 Resultados

A Tabela 4.3 mostra as diversas combinações testadas. Para ruído gaussiano, dois valores de relação sinal-ruído SNR (*Signal-to-Noise Ratio*) foram usados: 22 dB para o valor “alto” e 12 dB para o valor “baixo”.

Ruído gaussiano			Ruído de automóvel	
Caso	λ_R (dB)	SNR (dB)	Caso	λ_R (dB)
a	-6	alto	e	-6
b	-18	alto	f	-18
c	-6	baixo		
d	-18	baixo		

Tabela 4.3: Combinações de parâmetros para avaliação do redutor de ruído.

A Figura 4.5 apresenta os resultados para baixo ruído. Nela se observa a pequena degradação nas porções do sinal correspondentes às regiões de atividade vocal. Nota-se ainda que nos trechos mais curtos de silêncio, o ruído não é atenuado até o valor desejado. Isso demonstra a atuação do algoritmo psicoacústico incluído no bloco de redução de ruído. O sinal de voz provavelmente mascara o restante do ruído, fazendo-o ser menos audível sem que seja necessário modificar ainda mais o sinal de voz.

Figura 4.5: Saída do redutor de ruído para os casos “a” e “b”. As curvas mostram também o sinal original para comparação.

Observa-se ainda um atraso no sinal, provocado pelos filtros (cerca de 12 ms na análise mais 12 ms na síntese) e pelo processamento em bloco (cerca de 8 ms no total).

Para ruídos mais elevados, a Figura 4.6 mostra uma maior degradação na voz. O efeito audível dessa degradação não é tão pronunciado quanto se apresenta visualmente, devido ao mascaramento da distorção pelo próprio sinal.

Finalmente, a Figura 4.7 demonstra a capacidade do algoritmo em acompanhar as *variações* nas características do ruído.

As medidas são feitas em termos de potência instantânea suavizada (janela de 12 ms).

4.3.2 Especificações

Os detalhes de especificação para o bloco de redução de ruído estão mostrados na Tabela 4.4, para o processador SHARC e o sistema de

Figura 4.6: Saída do redutor de ruído para os casos “c” e “d”. As curvas mostram também o sinal original para comparação.

desenvolvimento utilizado no protótipo funcional.

4.4 Sistema integrado

O sistema completo (cancelador de eco e redutor de ruído) foi também avaliado. Esses testes partiram de um trecho de voz gravado e o eco foi calculado a partir da resposta impulsiva medida no carro. Ruído branco gaussiano foi adicionado e o resultado obtido foi usado como sinal do microfone ($d(n)$). Os subsistemas foram combinados das duas formas mais simples possíveis:

- Na configuração com supressão de ruído *antes* da filtragem adaptativa (Figura 4.8), o sinal desejado para o cancelador de eco é atrasado pelo redutor de ruído. O sinal de entrada do filtro adaptativo foi atrasado artificialmente de maneira similar.
- Na configuração com supressão de ruído *após* o cancelador de eco

Figura 4.7: Saída do redutor de ruído para os casos “e” e “f”. As curvas mostram também o sinal original para comparação.

(Figura 4.9) nenhuma modificação foi feita. O sinal foi simplesmente passado de um sistema para o outro.

Em ambas as configurações, foram usados os parâmetros abaixo:

- Fator de esquecimento: $\lambda_L = 0,9994$;
- Atenuação de ruído: $\lambda_R = -18$ dB.

4.5 Comentários

Os resultados do sistema integrado são nitidamente a favor da última configuração (redutor de ruído após cancelamento de eco). Conforme comentado por Martin e Vary [5], o eco já atenuado é amortecido ainda mais pelo algoritmo de redução de ruído.

Embora essa “colaboração” seja desejável, ela não foi explorada mais profundamente. Critérios psicoacústicos, aplicados ao redutor de ruído,

Método	subtração espectral psicoacústica
Freq. de amostragem	8000 Hz
Faixa de passagem	300–3400 Hz
Redução de ruído	> 30 dB
Análise/síntese espectral	implementação polifásica
Nro. de bandas	64
Dizimação	sub-crítica
Atenuação dos filtros	> 60 dB
Erro de reconstrução	< 0,0003 %
Atraso	~ 32 ms
Carga computacional	28 % (22 MFLOPS)

Tabela 4.4: Especificações técnicas do bloco de redução de ruído. A carga computacional foi medida com ajuda de um osciloscópio e um pino de saída do processador, mantido em estado alto durante os períodos ociosos.

poderiam ser aplicados também ao cancelador de eco. O sinal de voz, tomado como sinal mascarante para o cálculo do limiar auditivo, poderia ser utilizado para impor restrições não somente à subtração espectral, mas também ao nível de cancelamento do eco. Isso causaria ainda menores distorções no sinal de voz.

Outro aspecto que poderia ser aprimorado é a distribuição de carga entre os processadores. Um esquema de processamento paralelo mais efetivo permitiria, com o mesmo equipamento, o uso do cancelador de eco em ambientes mais reverberantes (com respostas impulsivas mais longas), sem prejudicar em nada o desempenho do redutor de ruído. Basta notar que mais de 70% da capacidade computacional do processador usado para reduzir o ruído está ociosa (Tabela 4.4).

Todas as medidas finais mostradas aqui foram tomadas em uma única tarde, na qual o processador foi carregado uma única vez com o programa e deixado em operação contínua. Na maior parte do tempo, o

Figura 4.8: Comportamento do sistema com redução de ruído realizada *após* o cancelamento de eco.

algoritmo ficou operando na ausência de sinal de entrada, sem nenhum problema de instabilidade numérica no filtro adaptativo.

De forma geral, os resultados para o cancelamento de eco acústico mostram sua robustez numérica e pouca sensibilidade ao ruído de medida ou à presença de sinal de voz local (“double-talk”).

O sistema de redução de ruído, por sua vez, é também robusto e capaz de seguir as variações estatísticas no ruído. Embora difícil de ser mostrado em gráficos, o aumento na qualidade subjetiva devido à inclusão do critério psicoacústico no algoritmo é um importante destaque na implementação. Adicionalmente, com algumas modificações na rotina, a faixa de frequências pode ser aumentada para abrigar todo o espectro audível para os seres humanos e tornar o algoritmo útil para recuperação de, por exemplo, gravações musicais antigas.

O capítulo seguinte retoma algumas dessas observações e apresenta as conclusões gerais do trabalho.

Figura 4.9: Comportamento do sistema com redução de ruído realizada *antes* do cancelamento de eco.

Capítulo 5

Conclusões

Nos capítulos anteriores foram apresentados os resultados obtidos com algoritmos de cancelamento de eco acústico e redução de ruído ambiente implementados em processador digital de sinais, utilizando de forma combinada diversas tecnologias e ferramentas matemáticas.

O enfoque prático dado ao trabalho é nítido ao longo da maior parte do texto. A começar pelo capítulo teórico, no qual se procura mostrar a necessidade de resolver o problema do atraso provocado pelo processamento espectral para redução de ruído. A técnica utilizada no projeto de filtros de reconstrução quase perfeita com mínimo atraso é bem ilustrada com um gráfico da resposta impulsiva obtida (Figura 2.13 na página 36).

Embora o projeto dos filtros tenha utilizado uma janela de Kaiser, um procedimento semelhante (e ligeiramente mais elaborado) foi desenvolvido para a janela de Chebyshev. Este não foi incluído no trabalho por apresentar resultados finais um pouco menos satisfatórios que os obtidos com a janela de Kaiser.

A adoção de uma técnica de estimação de ruído sem utilizar detectores de atividade vocal também tem uma explicação prática, já que o sistema foi pensado para operar em veículos, onde variações no nível de ruído são a regra. O algoritmo implementado se revelou muito ro-

busto e eficiente, embora algum estudo ainda seja necessário para se determinar o melhor valor para o fator OF , usado para corrigir o viés do estimador.

Destaque deve ser dado também, como já foi mencionado anteriormente no texto, para a inclusão dos critérios psicoacústicos no sistema de redução de ruído. O conhecimento das limitações do ouvido humano é de suma importância para a operação do algoritmo de subtração espectral. Se este for deixado livre para cortar qualquer sinal abaixo de um nível determinado apenas pela potência do ruído, distorções desnecessárias são introduzidas no sinal de voz. Se, por outro lado, esse nível for determinado pela potência *subjettiva* do ruído, o efeito do mascaramento auditivo é aproveitado e a qualidade final é decididamente superior.

É importante salientar que, embora a implementação feita aqui seja adaptada a um canal telefônico, a extensão do cálculo do limiar de mascaramento para todo o espectro de áudio é tarefa simples. Na verdade, esse cálculo é também usado no padrão de compressão de áudio profissional MPEG, no qual não há a mesma limitação que há para o canal telefônico. No caso de aplicação futura deste trabalho para audioconferência e restauração de gravações musicais antigas, esse aspecto é importante.

Outra contribuição importante, responsável inclusive pelos dois artigos apresentados em congressos e anexados a este documento, é o esquema desenvolvido para a implementação eficiente do algoritmo de filtragem adaptativa EFLSL. O autor acredita que, embora o método precise de uma formalização maior para ser eventualmente transformado em ferramenta automática, o esquema é relevante para qualquer algoritmo cujas operações sejam do tipo $ab+cd$. O processador foi usado, com esse método de implementação, do modo mais eficiente possível.

Finalmente, o protótipo funcional obtido foi testado em sinais capturados de situações reais, com o cancelamento de eco e a redução de

ruído operando de forma integrada. A combinação de todas as técnicas apresentadas no trabalho tem diversas razões para representar um ganho de qualidade no uso de terminais viva-voz em automóveis.

Apêndice A

Listagens Matlab

Os programas listados deste apêndice foram usados para simulação dos algoritmos mencionados no texto e projeto de tabelas. A linguagem é compatível com a versão 5.0 do Matlab e com a versão 2.1.35 do Octave.

A.1 Algoritmo LMS convencional

```
% lms.m
%
% Least-Mean-Square
% Filtro adaptativo baseado no gradiente estocastico
5 % Y=LMS(X,D,M,MU) calcula a saida do filtro
%   dado o sinal de entrada X e a saida desejada D
%   O numero de coeficientes e M e o passo de adaptacao
%   e MU (0<MU<2/(potencia de X))

10 % Octave 2.0.16
% AHCC
% Criado: set/1999
% Modificado: fev/2002

15 function y=lms(x,d,M,mu)
% inicializacao
w=zeros(1,M);
xx=zeros(1,M);
y=zeros(size(x));

20 % loop
n=1;
N=length(x);
while (n<=N)
25   xx = [x(n) xx(1:M-1)];
   y(n) = w * xx';
   erro = d(n) - y(n);
   w = w + mu*xx*erro;
   n = n + 1;
30 end
```

A.2 Algoritmo RLS convencional

```
% rls.m
%
% Recursive Least-Squares.
% Filtro adaptativo baseado no metodo dos minimos quadrados.
5 % Y=RLS(X,D,M,lambda) calcula a saida do filtro
%   dado o sinal de entrada X e a saida desejada D
%   O numero de coeficientes e M e o fator de
%   esquecimento (para tracking) e lambda (0<lambda<1).

10 % Octave 2.0.16
% AHCC
% Criado: set/1999
% Modificado: fev/2002

15 function y=rls(x,d,M,lambda)
% inicializacao
xx=zeros(1,M);
w=zeros(1,M);
y=zeros(size(x));
20 P=1000*diag(ones(1,M));

% loop
n=1;
N=length(x);
l=1/lambda;
25 while (n<=N)
    xx = [x(n) xx(1:M-1)];
    k = l*P*xx' / (1+l*xx*P*xx'); % ganho de Kalman
    erro = d(n) - w * xx';
    30 y(n) = w * xx'; % saida
    w = w + k'*erro;
    P = l*P - l*k*xx*P; % recursao da matriz inversa
    n = n + 1;
end
```


A.3 Filtro passa-baixas (janela de Kaiser)

```

## Copyright (C) 2002  André Carezia
##
## Usage:  qp_kaiser (nb, at, linear)
##
5  ## Computes a finite impulse response (FIR) filter for use with a
## quasi-perfect reconstruction polyphase-network filter bank. This
## version utilizes a Kaiser window to shape the frequency response of
## the designed filter. The number nb of bands and the desired
## attenuation at in the stop-band are given as parameters.
10 ##
## The Kaiser window is multiplied by the ideal impulse response
## h(n)=a.sinc(a.n) and converted to its minimum-phase version by means
## of a Hilbert transform.
##
15 ## By using a third non-null argument, the minimum-phase calculation is
## omitted at all.
##
## See also: qp_cheb

20 ## Author:  AHCC <acarezia@uol.com.br>

function h = qp_kaiser (nb, at, linear)

    if (nargin < 2)
        usage ("qp_kaiser (nb,at)");
    endif

    if (nargin < 3)
        linear = 0;
    endif

30    if ~(is_scalar (nb) && (nb == round(nb)) && (nb >= 0))
        error ("qp_kaiser: nb has to be a positive integer");
    endif

35    if ~(is_scalar (at) && (at == real (at)))
        error ("qp_kaiser: at has to be a real constant");
    endif

40    # Bandwidth
    bandwidth = pi/nb;

    # Attenuation correction (empirically
    # determined by M. Gerken
    # <mgk@lcs.poli.usp.br>)
45    corr = (1.4+0.6*(at-20)/80)^(20/at);
    at = corr * at;

    # size of window (rounded to next odd
    # integer)
50    N = (at - 8) / (2.285*bandwidth);
    M = fix(N/2);
    N = 2*M + 1;

55    # Kaiser window
    if (at>50)
        beta = 0.1102 * (at - 8.7);
    elseif (at>21)
        beta = 0.5842 * (at - 21)^0.4 + 0.07886 * (at - 21);
    else
60        beta = 0;
    endif
    w = kaiser(N,beta);

    # squared in freq. domain
65    wsquared = conv(w,w);

    # multiplied by ideal lowpass filter
    n = -(N-1):(N-1);
    hideal = 1/nb * sinc(n/nb);
70    hcomp = wsquared .* hideal;

    # extract square-root of response and
    # compute minimum-phase version

    Ndft = 2^15;

```

```
75  Hsqr = sqrt(abs(fft(hcomp,Ndft)));  
    if (linear)  
        h = real(ifft(Hsqr));  
        h = h(2:N);  
        h = [fliplr(h) h(1) h];  
80  else  
        Hmin = Hsqr .* exp(-j*imag(hilbert(log(Hsqr))));  
        h = real(ifft(Hmin));  
        h = h(1:N);  
    endif  
85                                     # truncate and fix amplitude scale  
                                     # (H(0)=1)  
    h = h / sum(h);  
  
endfunction
```

A.4 Coeficientes para os filtros

```

## Copyright (C) 2002  André Carezia
##
## Projeto do banco de filtros de reconstrução quase-perfeita.
## Sistema de mínima fase.
5 ## Saída em arquivos.
##
## 64 bandas (redução de taxa subcrítica: 1/2)
## Atenuação de 60dB na faixa de rejeição

10 ## Author:  AHCC <acarezia@uol.com.br>

M=64;
r=M/2;
aten=60;

15
                                # Kaiser (mínima fase)
h = qp_kaiser(M,aten);

                                # Salva resposta impulsiva em arquivo
20 out = h';
   save -ascii filter_ri.dat out

                                # Deixa o comprimento do filtro múltiplo
                                # de M
25 n = length(h);
   if n/M ~= round(n/M)
       h = [h, zeros(1, ceil(n/M)*M - n)];
       n = length(h);
   end;

30
                                # Gera arquivos de inclusão para o SHARC
taps = n/r;
h = reshape(h,r,taps);
g = h;
35 h = [h(1,:);flipud(h(2:r,:))];

fh=fopen('hfilts.h','w');
fg=fopen('gfilts.h','w');

40 for i=1:r-1
    fprintf(fh,'%8g, ',fliplr(h(i,:)));fprintf(fh,'\n');
    fprintf(fg,'%8g, ',fliplr(g(i,:)));fprintf(fg,'\n');
end
fprintf(fh,'%8g, ',fliplr(h(r,2:taps)));
45 fprintf(fg,'%8g, ',fliplr(g(r,2:taps)));
fprintf(fh,'%8g\n',h(r,1));           % last sample
fprintf(fg,'%8g\n',g(r,1));           % last sample

fclose(fh);
50 fclose(fg);

```

A.5 Tabelas de senos e cossenos

```
## Copyright (C) 2002  André Carezia
##
## trig.m
##
5  ## Gera tabelas de senos e cossenos para FFT
   ## 64 bandas (redução de taxa subcrítica: 1/2)

   ## Author:  AHCC <acarezia@uol.com.br>

10 ## nro. de canais
   M = 64;
   r = M/2;
   r2 = r/2;

15 ## calcula vetores
   fc=fopen('cosk.h','w');
   fs=fopen('sink.h','w');

   cosk=cos(pi*(0:r2-1)/r);
20  sink=sin(pi*(0:r2-1)/r);

   fprintf(fc,'%12f, ',cosk(1:(r2-1)));
   fprintf(fc,'%12f',cosk(r2));

25  fprintf(fs,'%12f, ',sink(1:(r2-1)));
   fprintf(fs,'%12f',sink(r2));

   ## fecha arquivos
   fclose(fc);
30  fclose(fs);
```

Apêndice B

Listagens SHARC

As listagens deste apêndice foram extraídas do protótipo funcional do sistema de cancelamento de eco e redução de ruído, montado em duas placas de desenvolvimento para o processador SHARC, da Analog Devices. A versão 3.3 do sistema de desenvolvimento (Compilador GNU C e Assembler) foi usada.

B.1 Redução de ruído (em linguagem C)

```

/* psiconr.c
 *
 * Copyright (C) 2002 André Carezia <acarezia@uol.com.br>
 * Versão: 1.25
5  */

/* 8 kHz
 * 64 bandas
 * Noise estimation: minimum statistics
10 */

/* ADSP-2106x System Register bit definitions */
#include <def21060.h>
#include <21060.h>
15 #include <signal.h>
#include <sport.h>
#include <macros.h>
#include <filters.h>
#include <trans.h>
20 #include <math.h>
#include <stats.h>
#include <matrix.h>

#define M 64 /* nro de bandas (canaís) */
#define R (M/2) /* subsampling factor */
25 #define FS 8000 /* sampling rate */
#define TAPS 11 /* ordem dos filtros + 1 */
#define FFT cfft32
#define IFFT ifft32
30 #define NLOW 3 /* # of zeroed bands for cutting bands
                at 300 Hz and below */
#define NHIGH 5 /* # of zeroed bands for cutting bands
                at 3400 Hz and above */

#define NCRIT 14
35 #define SFMMAX -40.0 /* max. tonality */

#define SNRMAX 50
#define SNRMIN -20
#define OSMAX 2.6
40 #define MINRED 1.0e-10

#define OSF 3.0 /* fator de correção da estimação de
                ruído */

#define PMAX 1.0e10
45 #define MM 35 /* subblock window size */
#define WW 4 /* # of subblocks */

/*****

50 /* DMA Chain pointer bit
   definitions */
#define CP_PCI 0x20000 /* Program-Controlled Interrupts
                        bit */
#define CP_MAF 0x1ffff /* Valid memory address field bits */

55 #define SetIOP(addr, val) (* (int *) addr) = (val)

/*****

60 /* DMA chaining Transfer Control Blocks */
typedef struct {
    unsigned lpath3; /* for mesh multprocessing */
    unsigned lpath2; /* for mesh multprocessing */
    unsigned lpath1; /* for mesh multprocessing */
65 unsigned db; /* General purpose register */
    unsigned gp; /* General purpose register */
    unsigned* cp; /* Chain Pointer to next TCB */
    unsigned c; /* Count register */
    int im; /* Index modifier register */
70 unsigned* ii; /* Index register */
} _tcb;

#define SZ_regs_1847 8
int regs_1847[SZ_regs_1847] = {

```

[illegible]

```

150 volatile float RuidoTC      =0.3;    /* cte de tempo para cálculo de ruído */
volatile float NivelResidual=0.125; /* -18 dB */

float saida;
int count=0;
int ganho=1;
155 int CHAVE=1;
int modo=3;
int pk_in=0;
int smp_count=0;
int blk_count=0;

160 /******
/*
/* Serial port transmit DMA complete
/*
/*
165 /******
void spt0_asserted( int sig_num )
{
    // Check if there are more commands left to transmit.
    if( xmit_count )
170 {
        // If so, put the comand into the transmit buffer and update
        // count.
        tx_buf[0] = *xmit_ptr++;
        xmit_count--;
175 }
}

/******
/*
/* Serial port receive DMA complete
/*
/*
180 /******
void spr0_asserted( int sig_num )
{
185     static int index=0;
    int sig_in;

    sig_in = rx_buf[1];
    if (abs(sig_in)>pk_in) pk_in=sig_in;
190
    /******
    /* ANALYSIS */
    /******
    // Filtro na taxa 1/R
    mi_active[index]=fir(sig_in,&h[index%R][0],&dlineh[index][0], TAPS);
195 mi_active[(index+R)%M]=fir(0.0,&h[index%R][0],&dlineh[(index+R)%M][0],TAPS);

    /******
    /* SYNTHESIS */
    /******
200 // Filtro na taxa 1/R
    saida=M*fir(mo_active[index],&g[index%R][0],&dlineg[index][0],TAPS);
    fir(mo_active[(index+R)%M],&g[index%R][0],&dlineg[(index+R)%M][0],TAPS);

205 /******
/* CONTROLE */
/******
// Testa se entrada do modulador já cheia
if (index == 0 || index == R)
210 {
    // swap input buffers
    aux=mi_active;
    mi_active=mi_inact;
    mi_inact=aux;
215
    raise(SIG_USR0);
}

index++;
220 if (index == R)
{
    // swap output buffers

```



```

        aux=mo_active;
        mo_active=mo_inact;
        mo_inact=aux;
225    }
    if (index == M)                                /* acabou segunda metade */
    {
        // swap output buffers
        aux=mo_active;
230        mo_active=mo_inact;
        mo_inact=aux;

        index=0;
235    }

    tx_buf[1] = (int)(ganho*saida+.5);
    tx_buf[2] = tx_buf[1];
}
240

/*****
/*
/* S2 switch interrupt service routine
/*
245 /*
*****/
void irq1_asserted( int sig_num )
{
    if (poll_flag_in(READ_FLAG1,RETURN_FLAG_STATE)==1)
250    {
        if (modo<3) modo++;
        else {
            modo=1;
            set_flag(SET_FLAG2,CLR_FLAG);
255            count=2000;
        }
    }
    else
260    {
        if (NivelResidual>0.03125) NivelResidual/=2;
        else
        {
            NivelResidual=0.5;
            set_flag(SET_FLAG2,CLR_FLAG);
265            count=2000;
        }
    }
}

270 /*****
/*
/* User interrupt 0 service routine
/* Polyphase Analysis
/*
275 *****/
void usr0_asserted( int sig_num)
{
    int x,y,i,j;
    float tempre[R],tempim[R],          /* armazenamento temporário */
280    A[R],F[R],                          /* armazenamento temporário */
    Re[R+1],Im[R+1];                  /* armazena espectro */
    float c;
    float Gm,Am,SFM,tonal;
    float SNR,OverSub;
285    float Offset[NCRIT],Thresh[NCRIT];
    float PLmin;

    // calcula x(2n)+j*x(2n+1)
    for (x=0;x<R;x++)
290    {
        tempre[x]=mi_inact[2*x];
        tempim[x]=mi_inact[2*x+1];
    }

295    // FFT(x(2n)+j*x(2n+1))
    FFT(tempre,tempim,A,F);

```

```

// Unpack the FFT
Re[0]=A[0] + F[0];
300 Im[0]=0;
for (x=1;x<R/2;x++)
{
    Re[x]=(A[x]+A[R-x] + cosk[x]*(F[x]+F[R-x]) + sink[x]*(A[R-x]-A[x]))/2;
    Re[R-x]=(A[x]+A[R-x] - cosk[x]*(F[x]+F[R-x]) - sink[x]*(A[R-x]-A[x]))/2;
305 Im[x]=(F[x]-F[R-x] - sink[x]*(F[x]+F[R-x]) + cosk[x]*(A[R-x]-A[x]))/2;
    Im[R-x]=(F[R-x]-F[x] - sink[x]*(F[x]+F[R-x]) + cosk[x]*(A[R-x]-A[x]))/2;
}
Re[R/2]=A[R/2];
Im[R/2]=-F[R/2];
310 Re[R]=A[0]-F[0];
Im[R]=0;

/*****
/* spectral processing */
315
/* passa-faixa telefônico */
for (x=0;x<NLOW;x++)
{
    Re[x]=0;
320 Im[x]=0;
}
for (x=0;x<NHIGH;x++)
{
    Re[R-x]=0;
325 Im[R-x]=0;
}

/* subtração espectral prévia */
for(x=NLOW;x<=R-NHIGH;x++)
330 {
    // Estimação da potência do sinal
    c = Re[x]*Re[x]+Im[x]*Im[x];
    sinal[x] = SinalTC*sinal[x] + (1.0-SinalTC)*c;

335 // atualiza valor mínimo do sinal nesta banda
    if (sinal[x] < Pxmin[x]) Pxmin[x]=sinal[x];

    // estimativa de SNR na banda
    if (sinal[x]>ruido[x])
340 SNR = 10*log10((sinal[x]-ruido[x])/ruido[x]);
    else
        SNR=SNRMIN;
    if (ruido[x]==0) SNR=SNRMAX;
    OverSub = OSMAX/2*(1+1/(1+x-NLOW))*(1-SNR/SNRMAX);

345 // Estimativa de sinal mascarante
    c=1-OverSub*sqrt(ruido[x]/sinal[x]); /* regra "H3" */
    if (modo==3)
    {
        if (c<MINRED) c=MINRED;
        350 sinalm[x] = SinalTC*sinalm[x] + (1.0-SinalTC)*(Re[x]*Re[x]+\
                                                    Im[x]*Im[x])*c*c;
    }
    else if (modo==2)
355 {
        // if (c<0) c=0;
        if (c<NivelResidual) c=NivelResidual;
        Re[x]=Re[x]*c;
        Im[x]=Im[x]*c;
        360 sinalm[x] = SinalTC*sinalm[x] + (1.0-SinalTC)*(Re[x]*Re[x]+\
                                                    Im[x]*Im[x]);
    }
}

365 // cálculo de estimativa do ruído (estatísticas mínimas)
smp_count ++;
if (smp_count==MM)
{
    for(x=NLOW;x<=R-NHIGH;x++)
370 {

```

```

        PMmin[x][blk_count] = Pxmin[x];          /* minimum of MM
                                                    samples */
        PLmin = PMmin[x][0];
        for (i=1;i<WW;i++)
375         if (PMmin[x][i] < PLmin)
            PLmin = PMmin[x][i];                /* minimum of
                                                    L=MM*WW
                                                    samples */

        // Atualiza estimativa de ruído
        if (CHAVE)
            ruido[x] = RuídoTC*ruido[x] + (1.0-RuídoTC)*OSF*PLmin;

        Pxmin[x]=PMAX;
385     }
        smp_count=0;
        blk_count++;
    }

390 // test for end of block (L=MM*WW subband samples)
    if (blk_count==WW) blk_count=0;

    /* divisão em bandas críticas (J. Johnston, 88 - modificado) */
    for(x=0;x<NCRIT;x++) CB[x]=0;
395 i=0;j=0;
    for(x=NLOW;x<=R-NHIGH;x++)
        for(y=0;y<10;y++)                        /* divide em 10 partes e soma
                                                    em escala Bark */
            {
200         if ((0.1*y+x)*FS/M >= CBLE[i+1])
            {
                CB[i]=CB[i]/j;
                i++;
                j=0;
205         }
            CB[i] += sinalm[x];
            j++;
        }
    if (j>0) CB[i]=CB[i]/j;

410

    /* função de espalhamento (membrana basilar) */
    matmul(SB,CB,spread,1,NCRIT,NCRIT);

415 /* Medida de 'planura' espectral */
    Gm = 0.0;
    for(x=NLOW;x<=R-NHIGH;x++)
        Gm += log10(sinalm[x]);
    Gm = Gm/(R-NLOW-NHIGH);
420 Am = mean(&sinalm[NLOW],R-NLOW-NHIGH);

    SFM = 10*(Gm-log10(Am)) / SFMMAX;
    tonal = 1.0;
    if (SFM < tonal) tonal = SFM;

425 /* offset e limiar em cada banda crítica */
    /* "desnormalização" (quase deconvolução...) */
    for(x=0;x<NCRIT;x++)
    {
430         Offset[x] = tonal*(15.5+x) + (1.0-tonal)*5.5;
        Thresh[x] = pow(10.0,log10(SB[x])-(Offset[x]/10)) * CB[x]/SB[x];
    }

    /* Limiar aplicado a bandas originais */
    i=0;
435 for(x=NLOW;x<=R-NHIGH;x++)
    {
        if (x*FS/M >= CBLE[i+1]) i++;
        limiar[x] = Thresh[i];
440     }

    /* subtração espectral definitiva (perceptiva) */
    if (modo==3)
        for(x=NLOW;x<=R-NHIGH;x++)

```

```

445     {
        c = sqrt(limiar[x]/ruído[x]) + NivelResidual;
        if (c < 1.0)
        {
            Re[x]=Re[x]*c;
            Im[x]=Im[x]*c;
450        }
        sinalr[x] = SinalTC*sinalr[x] + (1.0-SinalTC)*(Re[x]*Re[x]+\
                                                Im[x]*Im[x]);
    }

455    /* End of Spectral Processing */
    /*******/

    // RePack FFT
460    A[0]=(Re[0]+Re[R])/2;
    F[0]=(Re[0]-Re[R])/2;
    for (x=1;x<R/2;x++)
    {
        A[x]=(Re[x]+Re[R-x] - sink[x]*(Re[x]-Re[R-x]) - cosk[x]*(Im[x]+\
                                                Im[R-x]))/2;
465        A[R-x]=(Re[x]+Re[R-x] + sink[x]*(Re[x]-Re[R-x]) + cosk[x]*(Im[x]+\
                                                Im[R-x]))/2;
        F[x]=(Im[x]-Im[R-x] + cosk[x]*(Re[x]-Re[R-x]) - sink[x]*(Im[x]+\
                                                Im[R-x]))/2;
470        F[R-x]=(Im[R-x]-Im[x] + cosk[x]*(Re[x]-Re[R-x]) - sink[x]*(Im[x]+\
                                                Im[R-x]))/2;
    }
    A[R/2]=Re[R/2];
    F[R/2]=-Im[R/2];

475    // Calculate inverse FFT (mas não divide por R !)
    IFFT(A,F,tempre,tempim);

    // calcula y a partir de y(2n)+j*y(2n+1)
480    for (x=0;x<R;x++)
    {
        mo_inact[2*x]=tempre[x];
        mo_inact[2*x+1]=tempim[x];
    }

485 }

/*******/
void setup_sports ( void )
490 {
    /* Configure SHARC serial port SPORT0 */

    /* Multichannel communications setup */
    sport0_iop.mtcs = 0x00070007; /* transmit on words 0,1,2,16,17,18 */
    sport0_iop.mrcs = 0x00070007; /* receive on words 0,1,2,16,17,18 */
495    sport0_iop.mtccs = 0x00000000; /* no companding on transmit */
    sport0_iop.mrccs = 0x00000000; /* no companding on receive */

    /* TRANSMIT CONTROL REGISTER */
500    SetIOP(STCTL0, 0x001c00f2);

    /* RECEIVE CONTROL REGISTER */
    SetIOP(SRCTL0, 0x0f8c00f2);

505    /* Enable sport0 xmit & rcv irqs (DMA enabled) */
    interruptf(SIG_SPR0I, spr0_asserted);

    /* Set up Transmit Transfer Control Block for chained DMA */
510    tx_tcb.ii = tx_buf; /* DMA source buffer address */
    tx_tcb.cp = &tx_tcb.ii; /* define ptr to next TCB (point to self) */
    SetIOP(CP2, (((int)&tx_tcb.ii) & CP_MAF) | CP_PCI);
    /* define ptr to current TCB (kick off DMA) */
    /* (SPORT0 transmit uses DMA ch 2) */

515    /* Set up Receive Transfer Control Block for chained DMA */
    rx_tcb.ii = rx_buf; /* DMA destination buffer address */
    rx_tcb.cp = &rx_tcb.ii; /* define ptr to next TCB (point to self) */

```

```

520 SetIOP(CP0, (((int)&rx_tcb.ii) & CP_MAF) | CP_PCI);
    /* define ptr to current TCB (kick off DMA) */
    /* (SPORT0 receive uses DMA ch 0) */
}

525 /*****/
void send_1847_config_cmds( void )
{
    // Set up pointer and counter to transmit commands.
    xmit_ptr = regs_1847;
530 xmit_count = SZ_regs_1847;

    // Wait for all commands to be transmitted.
    while( xmit_count )
        idle();

535 // Wait for AD1847 autocall to start.
    while( !(rx_buf[0] & 0x0002) )
        idle();

540 // Wait for AD1847 autocall to finish.
    while( rx_buf[0] & 0x0002 )
        idle();

    return;
545 }

/*****/
void init_21k( void )
{
550 // Initialize pointer and counter to transmit commands.
    xmit_count = 0;
    xmit_ptr = regs_1847;

    // Enable interrupt nesting.
555 asm( "#include <def21060.h>" );
    asm( "bit set model NESTM;" );

    // Enable S2 switch interrupt.
    interruptf(SIG_IRQ1, irq1_asserted);

560 // Enable User interrupt 0.
    interrupt(SIG_USR0, usr0_asserted);

    // Turn flag LEDs off.
565 set_flag( SET_FLAG2, SET_FLAG );

    return;
}

570 /*****/
void main ( void )
{
    int x,y,i;

575 // Initialize some SHARC registers.
    init_21k();

    // Set up delay lines (FIR filters)
    for(x=0;x<M;x++)
580     for(y=0;y<=TAPS;y++)
        {
            dlineh[x][y]=0;
            dlineg[x][y]=0;
        }

585 // signal power initialization
    for (x=0;x<=R;x++)
    {
590         ruido[x]=0;
        sinal[x]=0;
        sinalm[x]=0;
        sinalr[x]=0;
    }
}

```

```
        Pxmin[x]=PMAX;
    }
595 // Reset the Codec.
    set_flag( SET_FLAG0, CLR_FLAG );    /* Put CODEC into RESET */
    for( x=0 ; x<0x7fff ; x++ )        /* Hold CODEC in RESET */
    {
600     set_flag( SET_FLAG0, SET_FLAG );    /* Release CODEC from RESET */

        // Configure SHARC serial port.
        setup_sports();

605     // Send setup commands to CODEC.
        send_1847_config_cmds();

        // Loop forever.
        for ( ;; )
610     {
            idle();
            if (count==0) set_flag( SET_FLAG2, SET_FLAG );
            else count--;
        };
615 }
```

B.2 Cancelamento de eco (Assembly)

```

! LING.ASM
!
! Algoritmo de filtragem adaptativa EF-LSL
!
5 ! Versão: 1.54
! Artigo ICASSP2001 (Carezia, A. et al)
!
! Modificações: 25/jan/2001 - AHCC - Correção da energia mínima
!
10 !
!           25/mar/1999 - AHCC - Otimização (17 ciclos)
!
!           19/mar/1999 - AHCC - criação
!

15 #include <def21060.h>
#define NCOEFS 290 /* nro. de coeficientes */
#define LAMBDA 0.9994 /* fator de esquecimento */
#define MINENRG 4000.0 /* teórico: 2^(12) */
#define ENRG0 4000.0 /* energia inicial (lambda * minenrg) */

20 #define IRQ1_ADDR 0x2001c
#define S0RX_ADDR 0x20028
#define S0TX_ADDR 0x20030

25 #define JUMP_OPCODE 0x063e0002 /* só funciona no EZ-KIT LITE */

/* vetor de interrupcoes */
.segment /pm seg_rth;
    nop;nop;nop;nop; /* reserved */
    nop;jump start; /* reset */
30 .endseg;

/* variaveis (dados) */
.segment /dm seg_dmda;
35 .var regs_1847[8] = /* registros do CODEC */
    0xc000, /* index 0 - left input control */
    0xc100, /* index 1 - right input control */
    0xc600, /* index 6 - left dac control */
    0xc700, /* index 7 - right dac control */
40    0xc850, /* index 8 - data format */
    0xc909, /* index 9 - interface configuration */
    0xcc0, /* index 12 - miscellaneous information */
    0x8f00;

45 .var tx_buf[3] = 0xcc0,0,0;
    .var rx_tcb[9] = 0,0,0,0,0,0,3,1,0;
    .var tx_tcb[9] = 0,0,0,0,0,0,3,1,0;
    .var rx_buf[3];
    .var xmit_count;
50 .var xmit_ptr;

    .var erro_r[2*(NCOEFS+1)];
    .var erro_p[NCOEFS+1];
    .var alfa=0;
55 .var b_bar[NCOEFS];
    .var lambda=LAMBDA;

    .var xrms=0.0;
    .endseg;

60

    .segment /pm seg_pmda;
    .var kf[NCOEFS];
    .var ftconv[2*(NCOEFS+1)];
65 .var kb[NCOEFS];
    .var kd[NCOEFS];
    .var enrg_p[NCOEFS];
    .var enrg_r[NCOEFS];
    .endseg;

70

    .segment /pm seg_pmco;
    /* rotina de tratamento da interrupcao SPORT0 TX */
    s0tx:

```

```

75      r2=dm(xmit_count);
      r2=pass r2;
      if eq jump (pc, _L$2) (DB);
      i7=dm(xmit_ptr);
      i5=i7;

80      modify(i5,m6);
      dm(xmit_ptr)=i5;
      r2=r2-1,r4=dm(i7,m5);
      dm(xmit_count)=r2;
85      dm(tx_buf)=r4;
_L$2:   rti;

/* ***** */
90 /* rot. tratamento de interrupcao (SPORT0 RX) */
/* ***** */
s0rx:   r0=dm(rx_buf+2);          /* x(n)=ref. (canal direito) */
      f0=float r0;

95      lbl:   dm(i0,m0)=f0;          /* erro_r = x(n) */
      dm(i1,m5)=f0;          /* erro_p = x(n) */

      r0=dm(rx_buf+1);          /* d(n)=des. (canal esquerdo) */
100     f0=float r0;

      dm(i2,m5)=f0;          /* alfa = d(n) */

      /* inicialização */
105     f4 =dm(i0,m5) , f0 =pm(i8,m13);
      fl0=1.0;
      fl2=0.0;
      f3 =0.0;

110     /* loop de recursão em ordem */
      lcntr=NCOEFS; do flt0 until lce;
      flt:   f12=f0*f4 , f10=f10-f12 , f0 =dm(i1,m5) , f4 =pm(i9,m8);
      f4 =f0*f4 , f10=abs f10 , f8 =dm(i1,m6) , f5 =pm(i10,m13);
      f13=f0*f5 , f2 =f8-f12 , f9 =dm(i0,m0) , pm(i9,m13)=f10;
115     f8 =f0*f4 , f6 =f9-f13 , f5 =dm(i0,m6) , f0 =pm(i9,m13);
      f0 =f0*f5 , f8 =f8+f15 , dm(i1,m5)=f2 , f1 =pm(i11,m15);
      f14=f1*f5 , f9 =dm(i2,m5) , pm(i11,m14)=f3;
      f9 =f0*f5 , f1 =f9-f14 , f5 =dm(i3,m5) , f10=pm(i8,m13);
      f12=f2*f5 , f2 =dm(i4,m5) , f5 =pm(i14,m13);
120     f13=f2*f5 , f10=f10+f12 , dm(i0,m0)=f6 , f5 =pm(i15,m13);
      f14=f2*f5 , f8 =f8+f13 , dm(i2,m5)=f1 , pm(i8,m14)=f10;
      f2 =f2*f4 , f9 =f9+f14 , pm(i14,m14)=f8;
      f8 =f2*f6 , f9 =f9+f15 , f10=pm(i9,m9);
      f4 =recips f8 , f12=pm(i10,m13);
125     f4 =recips f9 , pm(i15,m14)=f9;
      f4 =f0*f4 , f8 =f8+f12 , f12=pm(i11,m14);
      f9 =f1*f4 , dm(i3,m6)=f4 , pm(i10,m14)=f8;
      flt0: f12=f0*f4 , f3 =f9+f12 , f4 =dm(i0,m5) , f0 =pm(i8,m13);

130     /* finalização */
      f10=f10-f12 , modify(i9,m8);
      f10=abs f10 , modify(i11,m15);
      pm(i11,m14)=f3;
      pm(i9,m9)=f10;

135     /* atualização no tempo */
      modify(i1,m6);
      modify(i0,m1);

140     /* prepara amostra para conversão D/A (float -> integer) */
      r1=fix f1;

      flt_on: dm(tx_buf+1)=r1;          /* y(n)=saida (canal esquerdo) */
145     r1=dm(rx_buf+1);
      dm(tx_buf+2)=r1;          /* d(n) (canal direito) */

      rti;

```



```

150  /****** */
    /* principal */
    /****** */
    start:
155      r1=JUMP_OPCODE;
          px2=r1;
          r2=0xffff;

          r0=s0rx;
160      r0=r0 and r2;
          px1=r0;
          pm(SORX_ADDR)=px;

          r0=s0tx;
165      r0=r0 and r2;
          px1=r0;
          pm(S0TX_ADDR)=px;

          call inicio;

170      /* constantes */
          f15=MINENRG;
          m0=NCOEFS+1; m1=NCOEFS+2; m5=0; m6=1;
          m8=NCOEFS+1; m9=NCOEFS+2; m13=0; m14=1;

175      /* ponteiros fixos */
          i4=lambda;
          i2=alfa;

180      /* ponteiros circulares */
          b0=erro_r; l0=@erro_r;
          b1=erro_p; l1=@erro_p;
          b3=b_bar; l3=@b_bar;

185      b8 =kf;      l8 =@kf;
          b9 =ftconv; l9 =@ftconv;
          b10=kb;     l10=@kb;
          b11=kd;     l11=@kd;
          b14=enrg_p; l14=@enrg_p;
190      b15=enrg_r; l15=@enrg_r;

    init:  f0=ENRG0;
          f1=0.0;
          lcntr=NCOEFS; do ini0 until lce;          /* repete para M posições */
195          pm(i14,m14)=f0, dm(i0,m6)=f1;
          pm(i15,m14)=f0;
          pm(i8,m14)=f1;
          pm(i10,m14)=f1;
    ini0:  pm(i11,m14)=f1;
200          dm(i0,m6)=f1;          /* para posição M+1 */

          /* liga interrupções */
          bit clr IRPTL SPR0I;
          bit clr IMASK SPT0I | IRQ2I;
205          bit set IMASK SPR0I;

    wait:  idle;
          jump wait;

210

    /****** */
    /* inicializacao do processador (DMA, SPORT) e do CODEC */
    /****** */
215  inicio: r4=regs_1847;
          dm(xmit_count)=m5;
          dm(xmit_ptr)=r4;

          /* reset do CODEC */
220          bit clr ASTAT FLG0;
          lcntr = 0x7fff; do (pc, 1) until lce; nop;
          bit set ASTAT FLG0;

```

```

225      /* configura porta serial SPORT0 */
      r2=0x00070007;
      dm(MTCS0)=r2;
      dm(MRCS0)=r2;
      r12=0x001c00f2;
      dm(MTCCS0)=m5;
230      dm(MRCCS0)=m5;
      dm(STCTL0)=r12;
      r12=0x0f8c00f2;
      dm(SRCTL0)=r12;

235      /* truncamento em operações de ponto flutuante */
      bit set MODEL TRUNCATE;

      /* habilita 'nested interruptions' */
      bit set MODEL NESTM;

240      /* ativa interrupcoes */
      bit clr IRPTL SPT0I;
      bit set IMASK SPT0I;
      bit set MODEL IRPTEN;

245      /* configura DMA */
      i0=rx_tcb+8;
      i4=tx_tcb+8;
      r4=131071;
250      r2=i4;
      r8=131072;
      r2=r2 and r4;
      dm(i0,m5)=rx_buf;
      r2=r2 or r8;
255      dm(115)=r2;
      dm(rx_tcb+5)=i0;
      r2=i0;
      dm(tx_tcb+5)=i4;
      r2=r2 and r4;
260      dm(i4,m5)=tx_buf;
      r2=r2 or r8;
      dm(99)=r2;

      /* programação do CODEC. Ver registros no início do programa */
265      r8=regs_1847;
      dm(xmit_ptr)=r8;
      r8=8;
      dm(xmit_count)=r8;

_L$10:
270      idle;
      r2=dm(xmit_count);
      r2=pass r2;
      if ne jump (pc, _L$10) (DB);
      r2=dm(rx_buf);
275      r4=2;

      r2=r2 and r4;
      if ne jump (pc, _L$16) (DB);
      r5=r4;
280      nop;

_L$13:
      idle;
      r2=dm(rx_buf);
      r2=r2 and r5;
285      if eq jump (pc, _L$13) (DB);
      r2=dm(rx_buf);
      r4=2;

      jump (pc, _L$18) ;

290 _L$16:
      idle;
      r2=dm(rx_buf);
      r4=2;

_L$18:
295      r2=r2 and r4;
      if ne jump (pc, _L$16) ;

```

```
        rts;  
300    .endseg;
```

Apêndice C

Artigo SBT 1999

Este artigo, de autoria de A. H. C. Carezia, M. Gerken, P. M. S. Burt e M. T. M. da Silva, apareceu no XVII Simpósio Brasileiro de Telecomunicações, em setembro de 1999.

Implementação Eficiente de Algoritmo LSL para Cancelamento de Eco Acústico em Automóveis.

André H. C. Carezia, Max Gerken, Phillip M. S. Burt, Magno T. Madeira

Departamento de Engenharia Eletrônica da Escola Politécnica, Universidade de São Paulo
Av. Prof. Luciano Gualberto, trav. 3, n. 158, Cid. Universitária, 05508-900 São Paulo SP

Fone: (011) 818 5509, Fax: (011) 818 5718
{acarezia,mgk,philip,magno}@lcs.poli.usp.br

Sumário— Neste trabalho é apresentado um procedimento de otimização para a implementação eficiente de filtros adaptativos em processadores digitais de sinal. O procedimento apresentado viabilizou a utilização de um algoritmo do tipo RLS rápido e numericamente robusto com 290 coeficientes em um processador de baixo custo para cancelamento de eco acústico em automóveis.

1 INTRODUÇÃO

Existe atualmente um grande interesse no desenvolvimento de terminais telefônicos viva-voz avançados para serem utilizados em automóveis. Para esta aplicação, um dos problemas básicos que devem ser resolvidos de modo a prover uma comunicação de alta qualidade e verdadeiramente bidirecional é o problema de eco acústico. Este eco resulta do acoplamento entre o alto-falante e o microfone do terminal viva-voz, sendo que o desconforto produzido por ele cresce quando o retardo na conexão ultrapassa aproximadamente 30 ms [1]. Para superar este problema, um cancelador de eco acústico [2] ideal usaria o sinal recebido da outra extremidade da conexão para gerar uma réplica exata do sinal de eco acoplado entre o alto-falante e o microfone, e deste modo poder eliminá-lo do sinal enviado à outra extremidade da conexão. Devido ao fato de que o compartimento de passageiros de cada automóvel tem características acústicas diferentes e que um mesmo compartimento pode apresentar variações de suas características acústicas durante a conexão, é necessário que o cancelamento de eco seja realizado por um filtro adaptativo.

Os requisitos gerais de um filtro adaptativo são velocidade de convergência, capacidade de rastreamento, robustez e precisão numéricas e baixa complexidade computacional. Esses requisitos, de modo geral, são conflitantes entre si e diferentes estruturas de filtragem e algoritmos de adaptação apresentam compromissos distintos para tentar atendê-los. Os algoritmos do tipo gradiente estocástico (LMS) [1] têm uma complexidade computacional baixa (da ordem de $2M$, onde M é o número de

coeficientes do filtro adaptativo) e são de implementação relativamente fácil. Entretanto, para sinais de voz, podem apresentar convergência excessivamente lenta. Os algoritmos rápidos da família “Recursive Least Squares” (RLS), por outro lado, podem proporcionar convergência mais rápida às custas de maior complexidade computacional e de um eventual comportamento numérico instável. É necessária a utilização de um algoritmo numericamente estável, o que por sua vez tende a aumentar ainda mais a carga computacional.

Estendendo o conjunto de aplicações considerado, as características do eco acústico em cada situação podem ser bastante diversas, o que tem uma influência importante na escolha das técnicas de filtragem adaptativa. Tipicamente, o acoplamento entre alto-falante e microfone tem uma resposta impulsiva cuja duração é de aproximadamente 50 ms no caso de automóveis (o que, para uma taxa de amostragem de 8 kHz, resulta em um modelo FIR de 400 coeficientes) e de aproximadamente 500 ms para uma sala grande de conferências (o que, para uma taxa de amostragem de 16 kHz, resulta em um modelo FIR de 8000 coeficientes). A menor duração da resposta impulsiva e a taxa de amostragem mais baixa contribuem para tornar menos agudo o compromisso entre desempenho e complexidade computacional na solução do problema de eco acústico em automóveis.

Na Seção 2 é discutida brevemente a questão da robustez numérica de filtros RLS rápidos e apresentado o algoritmo RLS rápido utilizado neste trabalho. Em seguida, na Seção 3 é apresentado o procedimento de implementação deste algoritmo em um processador digital de sinais (DSP). O método de otimização da implementação, o qual permitiu utilizar com a maior eficiência possível a capacidade computacional do processador, é apresentado em detalhe. Deve-se ressaltar que, em face do compromisso já mencionado entre desempenho e complexidade computacional, esta otimização tem importância fundamental. Finalmente, na Seção 4, são apresentados resultados práticos da utilização do algoritmo implementado, para os quais foram utilizados sinais reais gravados em um automóvel.

Os arquivos de som destes sinais podem ser buscados em <http://www.lcs.poli.usp.br/sbt99>.

2 O ALGORITMO EF-LSL MODIFICADO

Algoritmos do tipo RLS são atraentes devido às propriedades de convergência e estimação consistente de parâmetros. Por outro lado, a sua complexidade computacional, mesmo das versões rápidas com complexidade computacional proporcional a M [8], é superior à dos algoritmos LMS.

As versões de algoritmos LSL utilizando realimentação de erro propostas por Ling [3, 8] são certamente das mais precisas e numericamente robustas. Mais especificamente, a precisão e a estabilidade numéricas do algoritmo LSL baseado em erros de predição *a priori* e utilizando realimentação de erro (*a priori* EF-LSL) é excelente mesmo quando comparadas com algoritmos do tipo QR-LSL que são conhecidos pelo seu bom comportamento quanto a estes aspectos [5, 7]. Embora este algoritmo não satisfaça algumas propriedades características dos algoritmos QR-LSL [7] que garantem uma propagação estável dos erros numéricos, ele é computacionalmente menos complexo e também satisfaz algumas propriedades que garantem um comportamento numericamente estável [3]. Além disso, tomando-se alguns cuidados descritos na seção seguinte, simulações extensivas mostraram que para a aplicação aqui considerada a sua estabilidade numérica é equivalente à dos algoritmos QR-LSL, sendo além disso mais preciso.

O algoritmo *a priori* EF-LSL (“Least Squares Lattice” com “Error Feedback”) convencional pode ser encontrado por exemplo em [8], página 633. Este algoritmo utiliza as seguintes variáveis: $(\xi_m^f(n), \eta_m(n), \Gamma_m^f(n))$ e $(\xi_{m-1}^b(n), \psi_m(n), \Gamma_m^b(n))$ que representam respectivamente as energias, os erros de predição *a priori* e os coeficientes de reflexão das predições progressiva (indicados por f) e regressiva (indicados por b); $(\gamma_m(n), \alpha_m(n), \kappa_m^d(n))$ que são respectivamente os fatores de conversão, os erros de estimação *a priori* e os coeficientes de regressão; e λ que é o fator de esquecimento. O índice, m or $m-1$, indica a correspondente ordem.

Na sua forma original a complexidade computacional deste algoritmo é proporcional a $22M$. Mesmo sendo considerado um algoritmo rápido, em algumas aplicações práticas como a aqui considerada esta complexidade computacional torna o seu uso proibitivo.

Para reduzir a complexidade computacional deste algoritmo foram introduzidos em [6] erros de predição *a posteriori* normalizados que estão relacionados aos erros de predição *a priori* da seguinte forma:

$$\begin{aligned}\bar{f}_{m-1}(n) &= f_{m-1}(n)/\xi_{m-1}^f(n) \\ &= \gamma_{m-1}(n-1)\eta_{m-1}(n)/\xi_{m-1}^f(n) \\ \bar{b}_{m-1}(n) &= b_{m-1}(n)/\xi_{m-1}^b(n) \\ &= \gamma_{m-1}(n)\psi_{m-1}(n)/\xi_{m-1}^b(n)\end{aligned}$$

Estes erros normalizados são utilizados como variáveis intermediárias nas equações de atualização da página 633 de [8]. Resulta o algoritmo apresentado na Tabela 1.

Em relação ao algoritmo original o número de multiplicações e de divisões é reduzido para 13 multiplicações e 2 divisões por estágio da treliça, enquanto que o número de adições, 9 por estágio da treliça, se mantém constante. As equações estão agrupadas em quatro conjuntos (1, 2, 3, 4, 5), (6, 7, 8), (9, 10) e (11, 12, 13) que são apresentados na Tabela 1 em ordem de precedência, isto é, alguns resultados do conjunto de equações precedente são necessários para os conjuntos subsequentes. Além disso foram introduzidos identificadores (ID) para as diferentes variáveis (destacados nos retângulos abaixo de cada variável) que são utilizados no desenvolvimento do programa para o DSP. Um aspecto importante é que apesar das modificações realizadas o mecanismo de **realimentação de erro** é mantido, não se afetando assim a acurácia numérica.

Para garantir a estabilidade numérica mesmo na ausência de excitação persistente faz-se necessário evitar as divisões por zero nas equações (9, 10) da Tabela 1. Para isso é suficiente forçar as variáveis $\bar{f}_{m-1}$ e $\bar{b}_{m-1}$ para zero quando as energias dos erros de predição ficam muito próximas de zero. Nota-se que nesse caso os erros de predição estão mais próximos de zero que as energias correspondentes, justificando este procedimento. Na implementação no DSP adotou-se uma outra estratégia computacionalmente mais eficiente que foi a de se somar ao valor da energia uma constante de valor baixo, no caso 2^{-60} . Com esta estratégia não se notou qualquer diferença mensurável nas simulações realizadas e nem nas implementações no DSP.

Além disso mostra-se necessário saturar os erros de predição para evitar resultados indefinidos (NaN - “Not A Number”), o que se consegue no DSP utilizado ativando o modo de **truncamento**. Nesse modo, resultados que ultrapassam o maior valor representável NORM.MAX ficam exatamente nesse valor.

3 IMPLEMENTAÇÃO NO DSP

O algoritmo EF-LSL modificado foi implementado em um processador digital de sinais (DSP) de ponto flutuante da **Analog Devices**, ADSP-21061 (SHARC). As equações mostradas na Tabela 1 foram implementadas em linguagem *assembly* e otimizadas para utilizar com a maior eficiência possível a capacidade do processador. Esta otimização foi conseguida mediante uma sequência de passos detalhada na Seção 3.2.

3.1 Arquitetura do processador[9]

No que se refere às unidades computacionais, o SHARC possui 16 **registradores internos** de 40

bits, sobre os quais são realizadas as operações de adição/subtração e multiplicação em paralelo. Os registradores são prefixados com “F” (para operações de ponto flutuante) ou “R” (para operações em ponto fixo) no código em *assembly*. Esses prefixos não afetam as transferências de dados, mas somente a forma como são realizadas as operações de soma e multiplicação sobre os dados. Agrupando-se as operações da forma correta (ver Figura 1), pode-se manter sempre as unidades computacionais trabalhando.

Figura 1: Diagrama de funcionamento paralelo das unidades computacionais do SHARC.

O SHARC possui dois **barramentos**: um dedicado à transferência de dados entre memória e registradores e outro para acesso às instruções. Usando acesso indireto (indexado) pode-se aproveitar também o barramento de programa para transferir dados entre memória e registradores. O acesso indireto à memória se faz por meio de 16 registros especiais de índice (I0 a I15) com o auxílio dos modificadores de índice (M0 a M15).

3.2 Processo de otimização

O **primeiro passo** é colocar as equações na ordem de execução, deixando as que precisam dos resultados de outra(s) equação(ões) para o final. Conforme pode-se notar nas equações da Tabela 1, as mesmas já estão ordenadas dessa forma. As equações (1-5), por exemplo, são independentes umas das outras e somente precisam de valores previamente calculados em iteração anterior. Entretanto a equação (6) precisa do resultado da equação (1) para ser feita, e por isso não pode ser deslocada para o início da iteração. O mesmo acontece com a equação (9), dependente dos resultados das equações (2) e (7). Com isso, destaca-se uma “hierarquia” de operações, com algumas podendo ser executadas em paralelo e outras dependendo de “grupos” anteriores. As variáveis são identificadas com letras para facilitar a manipulação.

O **segundo passo** é escolher uma representação das equações mostradas na Tabela 1 mais adequada às características do SHARC. Isso se faz necessário quando não há registradores internos (ou alguns precisam ser reservados para expansão futura) no DSP suficientes para armazenar todas as variáveis. Além disso, para obter paralelismo no SHARC

Inicialização ($n = 0$):	
Para $m = 1$ a M execute	
$\{\xi_{m-1}^f(n) = \xi_{m-1}^b(n) = \lambda\delta;$	
$\Gamma_m^f(n) = \Gamma_m^b(n) = \kappa_m^d(n) = \psi_{m-1}(n) = 0\}$	
Para $n = 1, 2, 3, \dots$ execute {	
Para $m = 0$ execute $\{\gamma_m(n) = 1;$	
$\eta_m(n) = u(n); \psi_m(n) = u(n); \alpha_m(n) = d(n)\}$	
Para $m = 1$ a M execute { 1) a 13) abaixo }	
1)	$\eta_m(n) = \eta_{m-1}(n) - \Gamma_m^f(n-1)\psi_{m-1}(n-1)$
	$\boxed{j} \quad \boxed{a} \quad \boxed{b} \quad \boxed{c}$
2)	$f_{m-1}(n) = \gamma_{m-1}(n-1)\eta_{m-1}(n)$
	$\boxed{k} \quad \boxed{d}$
3)	$\psi_m(n) = \psi_{m-1}(n-1) - \Gamma_m^b(n-1)\eta_{m-1}(n)$
	$\boxed{l} \quad \boxed{e}$
4)	$b_{m-1}(n) = \gamma_{m-1}(n)\psi_{m-1}(n)$
	$\boxed{m} \quad \boxed{f} \quad \boxed{g}$
5)	$\alpha_m(n) = \alpha_{m-1}(n) - \psi_{m-1}(n)\kappa_m^d(n-1)$
	$\boxed{n} \quad \boxed{h} \quad \boxed{i}$
6)	$\Gamma_m^f(n) = \Gamma_m^f(n-1) + \bar{b}_{m-1}(n-1)\eta_m(n)$
	$\boxed{s} \quad \boxed{o}$
7)	$\xi_{m-1}^f(n) = \lambda \xi_{m-1}^f(n-1) + f_{m-1}(n)\eta_{m-1}(n)$
	$\boxed{t} \quad \boxed{p} \quad \boxed{q}$
8)	$\xi_{m-1}^b(n) = \lambda \xi_{m-1}^b(n-1) + b_{m-1}(n)\psi_{m-1}(n)$
	$\boxed{u} \quad \boxed{r}$
9)	$\bar{f}_{m-1}(n) = f_{m-1}(n)/\xi_{m-1}^f(n)$
	$\boxed{v}$
10)	$\bar{b}_{m-1}(n) = b_{m-1}(n)/\xi_{m-1}^b(n)$
	$\boxed{w}$
11)	$\Gamma_m^b(n) = \Gamma_m^b(n-1) + \bar{f}_{m-1}(n)\psi_m(n)$
	$\boxed{x}$
12)	$\kappa_m^d(n) = \kappa_m^d(n-1) + \bar{b}_{m-1}(n)\alpha_m(n)$
	$\boxed{y}$
13)	$\gamma_m(n) = \gamma_{m-1}(n) - \bar{b}_{m-1}(n)b_{m-1}(n)$
	$\boxed{z}$

Tabela 1: Algoritmo EF-LSL com variáveis identificadas

(operações de multiplicação e adição/subtração no mesmo ciclo) é preciso que os operandos fiquem em posições mais restritas nos registradores. A tabela mostrada na Figura 2 evidencia os períodos (áreas hachuradas) de utilização de cada identificador (ID) e em quais operações está envolvido. Os fatores aparecem como dois **círculos** ou dois **quadrados** na mesma coluna. Os resultados parciais se somam/subtraem para gerar os resultados de cada equação (**estrelas** de quatro pontas).

Assim, a primeira coluna representa as operações necessárias à equação (1): $\boxed{b}$ e $\boxed{c}$ são multiplicados (círculos) e esse resultado subtraído de $\boxed{a}$ (quadrado), fornecendo $\boxed{j}$ (estrela). Nota-se que $\boxed{a}$ é utilizado ainda em mais 3 equações e depois não é mais necessário. Outro exemplo: Na equação (7), $\boxed{a}$ e $\boxed{k}$ (quadrados) são multiplicados entre si,

ID	Equações												
	1	2	3	4	5	6	7	8	9	10	11	12	13
a	■	●	●				■						
b	●					■							
c	●		■										
d		●											
e			●								■		
f				●									■
g				●	■			■					
h					■								
i					●							■	
j	+	+				●							
k							■		●				
l			+								●		
m				+				■		●			●
n					+							●	
o						●							
p							●	●					
q							●						
r								●					
s					+				●				
t						+				●			
u								+			●		
v									+			●	
w										+		●	●
x											+		
y												+	
z													+

Figura 2: Representação alternativa das equações básicas do algoritmo EF-LSL.

bem como $\boxed{p}$ e $\boxed{q}$ (círculos). Esses resultados parciais se somam e o resultado final é uma estrela em $\boxed{t}$.

Dispor as variáveis nos registradores, seguindo a hierarquia de operações, é o **terceiro passo** no processo de otimização. Para tanto, cria-se uma tabela como a da Figura 3 com 16 colunas (registradores) e tantas linhas (ciclos de máquina) quantas forem necessárias para completar o conjunto de equações. O método seguido é o seguinte:

1. Dispor multiplicações até que (a) não haja mais fatores disponíveis ou (b) o grupo para o qual deve ir o resultado já esteja 3/4 ocupado;
2. Fazer uma soma/subtração pendente;
3. Voltar ao passo 1.

Além disso, o processo de preenchimento da tabela segue algumas regras, abaixo relacionadas e acompanhadas de exemplos tirados da Figura 3:

1. Devido à restrição imposta pelo DSP (ver Figura 1) para executar operações em paralelo, nas multiplicações um dos fatores deve estar em F0-3 e o outro em F4-7; veja por exemplo $\boxed{b}$ e $\boxed{c}$ no ciclo 1, ou ainda $\boxed{i}$ e $\boxed{g}$ no ciclo 6, e para mais um exemplo $\boxed{v}$ e $\boxed{l}$ no ciclo 13;
2. Pelo mesmo motivo, as somas/subtrações precisam ter suas parcelas colocadas em F8-11 e F12-15, respectivamente; assim são subtraídos $\boxed{a}$ e $\boxed{bc}$ (resultado parcial) no ciclo 1, bem como se somam $\boxed{b}$ e $\boxed{jo}$ no ciclo 9;
3. Resultados (parciais ou não) que são usados posteriormente em operações de multiplicação devem ser colocados em registradores adequados no grupo F0-7; resulta da

regra 1 e tem como exemplos $k=ad$ (ciclo 3) e $m=fg$ (ciclo 6);

4. Resultados (parciais ou não) que são usados posteriormente como parcelas devem ser colocados em registradores adequados no grupo F8-15; resulta da regra 2; por exemplo: $\boxed{bc}$ (ciclo 2), $\boxed{gm}$ (ciclo 8) e $\boxed{pr}$ (ciclo 11);
5. Não sobrepor o conteúdo de registrador que ainda precisa entrar em operação posterior; usar outro registrador ou trocar a ordem dos fatores ou parcelas quando possível é a solução adotada. Exemplos: $\boxed{a}$ e $\boxed{d}$ são multiplicados no ciclo 2 e o resultado sobrepõe $\boxed{d}$, já que $\boxed{a}$ ainda é necessário nas operações seguintes; no ciclo 6, o valor $\boxed{i}$ é carregado para F1, já que $\boxed{f}$ ainda precisa ser utilizado em F0; o registrador F4 fica bloqueado do ciclo 4 até o ciclo 12, até terminar o uso do valor $\boxed{k}$;
6. A divisão é feita em duas etapas: uma inversão que não é paralelizável e uma multiplicação normal; notar nos ciclos 11 ($t \rightarrow \frac{1}{t}$) e 14 ($u \rightarrow \frac{1}{u}$) que somente inversões estão em andamento no processador.

O **quarto passo** é relacionar as operações de transferência de dados entre memória e registradores. Essa informação é toda extraída da Figura 3, com ajuda da Figura 2 para distinguir entre somas, subtrações e multiplicações. Para cada operação de adição, subtração, ou multiplicação as variáveis precisam estar em registradores adequados para garantir o paralelismo; após cada operação, os resultados precisam ser transferidos para a memória.

As transferências de dados são feitas por acesso indireto à memória, e cada variável **original** da Tabela 1 precisa ter endereços alocados na memória. No SHARC, dois acessos indiretos à memória podem ser feitos simultaneamente, desde que um deles seja feito pelo grupo I0-7 de indexadores e o outro pelo grupo I8-15.

A disposição utilizada deriva da simples observação das variáveis necessárias a cada ciclo na Figura 3 e aparece na Tabela 2, na qual também aparece o tamanho de cada vetor correspondente, relativo à ordem do filtro adaptativo (M). Usando essa alocação de vetores, a Figura 4 mostra as transferências de memória necessárias para possibilitar a execução das operações computacionais descritas na Figura 3.

O **último passo** inclui alguns ajustes, tais como:

1. Transferir os ciclos que estão hachurados (18,19) nas Figuras 3 e 4 para o início da iteração seguinte; isso é possível já que tanto o acumulador como os barramentos DM (*data memory*) e PM (*program memory*) têm alguns ciclos ociosos; isso exige que na última iteração esses ciclos sejam feitos em separado;

Ciclo	Multiplicador								Acumulador							
	X				Y				X				Y			
	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
1	b				c											
2	a				d								bc			
3	a				ad=k	e			a				bc			
4	a		a-bc=j		k					c				ae		
5	f					g	c-ae=l		ak							
6	fg=m	i				g										
7	m					g				h					gi	
8		h-gi=n	j			o				gm	b					
9			p			q					b		jo			
10			p			r			ak		b+jo=s			pq		
11			ak+pq=t												pr	
12			inv(t)		k					gm					pr	
13			k/t=v		gm+pr=u		l									
14					u				vl							
15	m				inv(u)				vl				e			
16		n			m/u=w				e+vl=x				i			
17	m				w					nw			i			
18										i+nw=y	f		mw			
19											f-mw=z					

Figura 3: Conteúdo dos registradores do SHARC imediatamente **antes** de cada ciclo.

Ponteiro	Tamanho do vetor	Identificadores
I0	2(M+1)	c,l,g
I1	M+1	a,j
I2	1	h,n
I3	M	o,w
I4	1	p
I8	M	b,s
I9	2(M+1)	d,f,z
I10	M	e,x
I11	M	i,y
I12	M	q,t
I13	M	r,u

Tabela 2: Alocação de ponteiros para as variáveis do algoritmo.

- Transferir o ciclo 0 (hachurado na Figura 4) para o fim da iteração anterior; mesmo que o barramento PM esteja ocupado no final, basta deslocar a transferência do ciclo 17 para o ciclo 12. A operação do ciclo 0 é então repetida na inicialização;

3.3 Complexidade computacional

Com o procedimento de otimização descrito, obteve-se uma implementação com **17** ciclos de máquina por iteração. A cada amostra de entrada, são executadas basicamente $17 * M$ instruções. O ciclo de instruções no processador utilizado é de 25 ns. Com uma taxa de amostragem de 8 kHz o processador pode executar $1/(8000 * 25.10^{-9}) = 5000$ instruções por amostra.

Se todas as instruções forem usadas em iterações do filtro adaptativo, o número máximo de coeficientes é dado por $M = 5000/17 \cong 294$.

Contando com instruções de inicialização, interface com os conversores e ciclos para atendimento

	DM bus	PM bus
0	F4 ← (I0)	F0 ← (I8)
1	F0 ← (I1)	F4 ← (I9)
2	F8 ← (I1)	F5 ← (I10)
3	F9 ← (I0)	
4	F5 ← (I0)	F0 ← (I9)
5	F2 → (I1)	F1 ← (I11)
6	F9 ← (I2)	
7	F5 ← (I3)	F10 ← (I8)
8	F2 ← (I4)	F5 ← (I12)
9	F6 → (I0)	F5 ← (I13)
10	F1 → (I2)	F10 → (I8)
11		F2 → (I12)
12		
13		F4 → (I13)
14		F12 ← (I10)
15		F12 ← (I11)
16	F4 → (I3)	F8 → (I10)
17		F10 ← (I9)
18		F9 → (I11)
19		F10 → (I9)

Figura 4: Diagrama de transferências entre memória e registradores no SHARC.

da interrupção, conseguiu-se rodar o filtro em tempo real com **290** coeficientes, permitindo ao mesmo acompanhar respostas impulsivas com duração de **até cerca de 36 ms**. O ganho é significativo, quando comparado à implementação em linguagem C, com a qual foi possível rodar o filtro com apenas 54 coeficientes.

4 RESULTADOS

Para ilustrar o seu funcionamento, o algoritmo implementado foi aplicado a um problema de cancelamento de eco acústico. Em um automóvel marca Escort, o acoplamento acústico entre um alto-falante localizado no console e um microfone de eletreto LeSon ML-8 localizado no espelho foi

medido. Os equipamentos e suas localizações foram escolhidos de modo a serem bastante próximos aos de uma situação real. A taxa de amostragem utilizada foi de 8 kHz e a resposta original obtida com 512 termos (ver Figura 5).

Com a resposta impulsiva obtida, o seguinte procedimento foi seguido: em MatLab, a resposta foi aplicada a um trecho de uma gravação (com o mesmo microfone utilizado no carro) de um sinal de voz $u(n)$ de modo a produzir um sinal de eco $d(n)$. Os dois sinais, voz e eco, foram então gerados usando o programa Cool Edit 96 e uma placa de som TurtleBeach Fiji e foram enviados à placa de desenvolvimento do DSP, onde o algoritmo LSL estava implementado. No algoritmo, foram usados 290 coeficientes e um fator de esquecimento $\lambda = 0,999$. O sinal de erro $\alpha_M(n)$ foi enviado para a saída da placa de DSP e foi gravado usando também o programa Cool Edit 96 e a mesma placa de som. Finalmente, em Matlab, o fator instantâneo de redução de eco ERLE (*echo return loss enhancement*) dado por

$$\text{ERLE} = 10 \log \left[\overline{d^2(n)} / \overline{\alpha_M^2(n)} \right],$$

foi calculado, estando representado no gráfico da Figura 6. As variáveis do lado direito na expressão acima foram obtidas filtrando as variáveis $d^2(n)$ e $\alpha_M^2(n)$ com uma média móvel de 64 pontos. No resultado a redução de eco está limitada pelo efeito das amostras adicionais da resposta impulsiva. Se a resposta fosse truncada em 290 amostras o eco poderia, idealmente, ser totalmente cancelado. A discrepância que restaria em relação a esse cancelamento total representaria apenas o efeito dos erros de quantização envolvidos na amostragem dos sinais e na implementação do algoritmo.

Figura 5: Resposta impulsiva original medida no carro (512 amostras).

REFERÊNCIAS

- [1] A.Gilloire, E.Moulines, D.Slock, P. Duhamel, "State of the art in acoustic echo cancel-

Figura 6: Redução de Eco (ERLE) com resposta impulsiva original (512 amostras).

- lation", pp.45-91, *Digital Signal Processing in Telecommunications*, Anibal R. Figueiras-Vidal (Ed.), Springer, 1996.
- [2] E.Hänsler, "The hands-free telephone problem - An annotated bibliography", *Signal Processing*, vol.27, no.3, Junho 1992, Elsevier, pp.259-271.
- [3] F. Ling, D. Manolakis, J.G. Proakis: "Numerically robust least-squares lattice-ladder algorithms with direct updating of the reflection coefficients", *IEEE Trans. on ASSP*, vol.34, no.4, p.837-845, Agosto 1986.
- [4] P.A. Regalia: "Numerical stability properties of a QR-based fast least squares algorithm", *IEEE Trans. on SP*, vol.41, no.6, p.2096-2109, Junho 1993.
- [5] M.D. Levin, C.F.N. Cowan: "The performance of 8 recursive least squares adaptive filtering algorithms in a limited precision environment", *Signal Processing VII. Theories and Applications*, Edinburg, Scotland, p.1261-1264, Setembro 1994.
- [6] M. Miranda, M. Gerken: "Implementação otimizada para o algoritmo LSL a priori com realimentação de erro". Anais do XV *Simpósio Brasileiro de Telecomunicações*, Recife, 8 a 11 de setembro de 1997, p. 280-284.
- [7] M.D. Miranda, M. Gerken: "Performance of the a priori and a posteriori QR-LSL algorithms in a limited precision environment", *Proceedings ICASSP97*, Munique, Abril 1997.
- [8] S. Haykin: *Adaptive filter theory*, Prentice Hall Int., 2nd. Ed., 1991.
- [9] *ADSP-2106x SHARC User's Manual*, Analog Devices, 2nd. Ed., 1996

Apêndice D

Artigo ICASSP 2001

Este artigo, cujos autores são A. H. C. Carezia, P. M. S. Burt, M. Gerken, M. Miranda e M. T. M. da Silva, foi apresentado no congresso internacional ICASSP, em maio de 2001.

A STABLE AND EFFICIENT DSP IMPLEMENTATION OF A LSL ALGORITHM FOR ACOUSTIC ECHO CANCELLING

André H.C. Carezia, Phillip M.S. Burt, Max Gerken, Maria D. Miranda[†], Magno T.M. da Silva

Laboratory of Communications and Signals
Dept. of Telecommunications and Control Engineering
Polytechnic School/University of São Paulo, Brazil

<<http://www.lcs.poli.usp.br>> {acarezia,phillip,mgk,maria,magno}@lcs.poli.usp.br

[†] Universidade Presbiteriana Mackenzie, Brazil

<<http://www.mackenzie.com.br>> mdm@mackenzie.com.br

ABSTRACT

In this paper we present an optimized DSP implementation of a modified error-feedback lattice least-square (EF-LSL) adaptive filtering algorithm. Simple measures that provide numerical stability for poor persistent excitation are also proposed. As a result of the optimization and the stability measures, an efficient and stable implementation of a fast algorithm of the RLS family was attained. We present the results of an acoustic echo cancelling experiment performed with the implemented algorithm. With a 40 MIPS SHARC DSP, up to 290 adaptive filter coefficients can be used. This represents an effective alternative to algorithms of the LMS family, while still retaining the good convergence properties of the RLS family.

1. INTRODUCTION

Acoustic Echo Cancellation (AEC) is a challenging application for adaptive filtering due to the required on-line processing through high order adaptive filters and to poor excitation characteristics of voice signals [1]. The convergence and consistent parameter estimation properties of Recursive Least Squares (RLS) algorithms make them interesting alternatives for this application. It is well known that in comparison with Least Mean Squares (LMS) algorithms the trade off for the improved performance of fast RLS algorithms is an increase in computational complexity and possible numerical instability. Moreover, the high order of the adaptive filter and poor persistent excitation make numerical instability show up more easily, stressing the importance of numerically stable algorithms. However, high implementation cost hinders the use of numerically stable fast RLS algorithms as the backward stable QR-LSL versions [2, 3].

In the described scenario, Least Squares Lattice (LSL) algorithms represent possible alternatives due to their known numerical robustness. Particularly, the *a priori* error feedback LSL algorithm (EF-LSL) proposed by Ling, Manolakis and Proakis [7] could be a good choice. It is certainly among the most numerically accurate LSL algorithms, comparing favorably with the *a priori* or a *posteriori* QR-LSL algorithms [3]. Though not sharing some theoretical properties with the QR-LSL algorithms that guarantee stable error propagation [2, 3], the *a priori* EF-LSL algorithm is computationally less complex and, as it will be shown, still presents numerically stable behavior when properly implemented.

In Section II we briefly present a modified error-feedback LSL algorithm based on the *a priori* error feedback algorithm [7, 8].

We also propose a simple numerical convention that ensures numerical stability for poor persistent excitations. Divisions may be implemented using denominators with reduced wordlength. Thus, lookup tables may be used to implement inversion of denominators, removing one of the reasons why LSL adaptive filters are usually not used in high-order real-time applications like AEC, i.e. the high computational load of divisions. These properties make the presented algorithm a good option for stringent high-order real-time adaptive filtering applications like AEC.

In Section III we present an efficient implementation of the modified EF-LSL algorithm for the floating point SHARC DSP. The implementation followed a thorough optimization procedure, which allowed the parallelism of this processor to be fully exploited. As a result, a 290 coefficient (for a 8 kHz sampling rate) EF-LSL acoustic echo canceller was made possible.

2. MODIFIED ERROR-FEEDBACK LSL ALGORITHM

The conventional *a priori* EF-LSL algorithm can be found for example in [8, p.633]. It deals with the following variables: $(\xi_m^f(n), \eta_m(n), \Gamma_m^f(n))$ and $(\xi_{m-1}^b(n), \psi_m(n), \Gamma_m^b(n))$ that represent respectively energies, *a priori* prediction errors and reflection coefficients of the forward and backward predictions, $(\gamma_m(n), \alpha_m(n), \kappa_m^d(n))$ that are respectively conversion factors, *a priori* estimation errors and regression coefficients and the forgetting factor λ . The subscript m indicates the corresponding order. In its original form, the computational complexity of this algorithm is $18M$ multiplications, $9M$ additions and $4M$ divisions, where M is the number of adaptive filter coefficients.

To reduce the computational complexity of this algorithm we introduced normalized *a posteriori* prediction errors that are related to the *a priori* prediction errors as follows [4]:

$$\begin{aligned} \bar{f}_{m-1}(n) &= f_{m-1}(n)/\xi_{m-1}^f(n) \\ &= \gamma_{m-1}(n-1)\eta_{m-1}(n)/\xi_{m-1}^f(n), \\ \bar{b}_{m-1}(n) &= b_{m-1}(n)/\xi_{m-1}^b(n) \\ &= \gamma_{m-1}(n)\psi_{m-1}(n)/\xi_{m-1}^b(n). \end{aligned}$$

This leads to the algorithm presented in Table 1 [6], which requires $13M$ multiplications, $9M$ additions and $2M$ divisions. It can be seen that equations are divided in four groups, (1, 2, 3, 4, 5), (6, 7, 8), (9, 10) and (11, 12, 13). Any equation in one given group uses only results from the preceding groups, which is a desirable feature when implementing the algorithm in a DSP. It is worth noting

that the error feedback mechanism is maintained in the modified algorithm, and therefore numerical accuracy is not affected.

Initialization ($n = 0$): For $m = 1$ to M do $\{\xi_{m-1}^f(n) = \xi_{m-1}^b(n) = \lambda\zeta$; $\Gamma_m^f(n) = \Gamma_m^b(n) = \kappa_m^d(n) = \psi_{m-1}(n) = 0\}$	
For $n = 1, 2, 3, \dots$ do { For $m = 0$ do $\{\gamma_m(n) = 1$; $\eta_m(n) = u(n)$; $\psi_m(n) = u(n)$; $\alpha_m(n) = d(n)\}$ For $m = 1$ to M do { 1) to 13) below } }	
1)	$\eta_m(n) = \eta_{m-1}(n) - \Gamma_m^f(n-1)\psi_{m-1}(n-1)$ <div style="display: flex; justify-content: space-around;">$\boxed{j}$$\boxed{a}$$\boxed{b}$$\boxed{c}$</div>
2)	$f_{m-1}(n) = \gamma_{m-1}(n-1)\eta_{m-1}(n)$ <div style="display: flex; justify-content: space-around;">$\boxed{k}$$\boxed{d}$</div>
3)	$\psi_m(n) = \psi_{m-1}(n-1) - \Gamma_m^b(n-1)\eta_{m-1}(n)$ <div style="display: flex; justify-content: space-around;">$\boxed{l}$$\boxed{e}$</div>
4)	$b_{m-1}(n) = \gamma_{m-1}(n)\psi_{m-1}(n)$ <div style="display: flex; justify-content: space-around;">$\boxed{m}$$\boxed{f}$$\boxed{g}$</div>
5)	$\alpha_m(n) = \alpha_{m-1}(n) - \psi_{m-1}(n)\kappa_m^d(n-1)$ <div style="display: flex; justify-content: space-around;">$\boxed{n}$$\boxed{h}$$\boxed{i}$</div>
6)	$\Gamma_m^f(n) = \Gamma_m^f(n-1) + \bar{b}_{m-1}(n-1)\eta_m(n)$ <div style="display: flex; justify-content: space-around;">$\boxed{s}$$\boxed{o}$</div>
7)	$\xi_{m-1}^f(n) = \lambda \xi_{m-1}^f(n-1) + f_{m-1}(n)\eta_{m-1}(n) + \zeta$ <div style="display: flex; justify-content: space-around;">$\boxed{t}$$\boxed{p}$$\boxed{q}$</div>
8)	$\xi_{m-1}^b(n) = \lambda \xi_{m-1}^b(n-1) + b_{m-1}(n)\psi_{m-1}(n) + \zeta$ <div style="display: flex; justify-content: space-around;">$\boxed{u}$$\boxed{r}$</div>
9)	$\bar{f}_{m-1}(n) = f_{m-1}(n)/\xi_{m-1}^f(n)$ <div style="display: flex; justify-content: space-around;">$\boxed{v}$</div>
10)	$\bar{b}_{m-1}(n) = b_{m-1}(n)/\xi_{m-1}^b(n)$ <div style="display: flex; justify-content: space-around;">$\boxed{w}$</div>
11)	$\Gamma_m^b(n) = \Gamma_m^b(n-1) + \bar{f}_{m-1}(n)\psi_m(n)$ <div style="display: flex; justify-content: space-around;">$\boxed{x}$</div>
12)	$\kappa_m^d(n) = \kappa_m^d(n-1) + \bar{b}_{m-1}(n)\alpha_m(n)$ <div style="display: flex; justify-content: space-around;">$\boxed{y}$</div>
13)	$\gamma_m(n) = \gamma_{m-1}(n) - \bar{b}_{m-1}(n)b_{m-1}(n)$ <div style="display: flex; justify-content: space-around;">$\boxed{z}$</div>

Table 1. EF-LSL Algorithm variables with corresponding identifiers.

To guarantee robust numerical behavior even for poor persistent excitation it is necessary to avoid divisions by values close to zero in Equations (9,10) of Table 1. To this end it is sufficient to add a positive constant ζ_0 to the denominators of these equations. Alternatively, as shown in Table 1, a constant $\zeta = \zeta_0(1 - \lambda)$ can be added to Equations (7,8) of Table 1. Considering that the input signal satisfies $-2^{k/2} \leq u(n) \leq 2^{k/2}$, with k as low as possible [5], extensive simulations have shown that $\zeta = 2^{k-b}(1 - \lambda)$ is sufficient (though not strictly necessary) to assure numerical stability. Here, b is the mantissa wordlength to which the energies are quantized. This simple numerical convention was used with excellent results in several simulations. It was also verified that divisions may be implemented using denominators with reduced mantissa

wordlength (for example, 8 bits), without producing instability.

Considering that numerical instability problems show up more easily when low forgetting factors are used, we performed acoustic echo cancellation experiments with more than one thousand coefficients and with forgetting factors as low as $\lambda = 0.1$. In these cases the algorithm remained numerically stable and did not break down, although it did not produce useful results due to the low λ values that lead to large estimation errors.

3. DSP IMPLEMENTATION

The EF-LSL algorithm was implemented in assembly language in a floating point Analog Devices 21061 DSP (SHARC). This DSP has sixteen 40-bit internal registers, which are used for parallel additions and multiplications. In the following, these registers are denoted by F0, F1, ..., F15 ("F" standing for "floating point"). For indirect memory addressing 16 pointers are used, which are denoted by I0, I1, ..., I15. Two simultaneous transfers between registers and memory can be made, as long as one transfer uses a pointer in I0-I7 and the Data Memory (DM) bus and the other transfer uses a pointer in I8-I15 and the Program Memory (PM) bus. A reduced wordlength division can be carried out in two machine cycles: an 8-bit tabulated inversion and a multiplication.

3.1. Optimization

The particular organization of the steps of the modified EF-LSL algorithm presented in Table 1 is a suitable starting point for obtaining an efficient DSP implementation, as the steps within each group can be carried out independently. Performing then a careful allocation of the required operations to the processing resources, the 13 steps of the algorithm can be fitted, initially, into 19 machine cycles. The contents of the DSP registers before each of these 19 cycles are shown in Table 2. Table 3 shows the variables that are assigned to each pointer and Table 4 shows the transfers between memory and registers carried out in each cycle. The number of required machine cycles can still be reduced to 17 by transferring steps 18 and 19 in Tables 2 and 4 to the beginning of the following iteration and step 0 in Table 4 to the end of the previous iteration. This involves making adjustments in steps 12 and 13 in Tables 2 and 4. The core of the final algorithm is shown (in assembly language [9]) in Table 5. With the 40 MIPS SHARC DSP that was employed, it is possible to use up to $M = 290$ coefficients for a 8 kHz sampling rate.

3.2. Numerical aspects

Using the definitions of Section II, in the DSP implementation $k = 30$ (due to a 16-bit A/D) and $b = 8$ (the mantissa length to which the prediction energies are quantized). This would lead to $\zeta = 2^{22}(1 - \lambda) \approx 2^{12}$ (for $\lambda = 0.999$). However, considering the high value of λ that is used, a smaller value of $\zeta = 2^7$ was adopted. As an extra safeguard against instability, the absolute value of the conversion factor γ_m was used in the implementation. For the SHARC DSP this did not represent any additional computational load. It was also necessary to saturate the prediction errors in order to avoid indefinite results (NaN), which is obtained activating the saturation mode in the DSP.

3.3. An acoustic echo cancellation experiment

Using a measured impulse response of 1024 samples (at a 8 kHz sampling rate) an echo signal was produced from a recorded voice signal. The first 512 samples of the measured impulse response are shown in Figure 1. The voice signal and the echo signal $d(n)$

Table 2: Contents of DSP registers previously to each cycle or result after each cycle

	Multiplier Unit						Accumulator Unit							
	X			Y			X				Y			
Cycle	f0	f1	f2	f4	f5	f6	f8	f9	f10	f12	f13	f14	f15	
1	b			c										
2	a			d						bc				
3	a			ad=k	e		a			bc				
4	a		a-bc=j	k				c			ae			
5	f				g	c-ae=l	ak						ζ	
6	fg=m	i			g		ak+ζ							
7	m				g			h				gi		
8		h-gi=n	j		o			gm	b					
9			p		q				b	jo				
10			p		r		ak+ζ		b+jo=s		pq			
11							ak+ζ+pq=t					pr		
12			inv(t)	k				gm				pr		
13			k/t=v			l		gm+pr=u					ζ	
14				u			vl	u+ζ		e				
15	m			inv(u+ζ)			vl			e				
16		n		m/(u+ζ)=w			e+vl=x			i				
17	m			w				nw		i				
18								i+nw=y	f	mw				
19									f-mw=z					

Table 3: Assignment of pointers to variables.

Pointer	Vector size	Identifiers
I0	2(M+1)	c,l,g
I1	M+1	a,j
I2	1	h,n
I3	M	o,w
I4	1	p
I8	M	b,s
I9	2(M+1)	d,f,z
I10	M	e,x
I11	M	i,y
I14	M	q,t
I15	M	r,u

Table 4: Transfers between registers and memory

Cycle	DM bus	PM bus
0	(i0) → f4	(i8) → f0
1	(i1) → f0	(i9) → f4
2	(i1) → f8	(i10) → f5
3	(i0) → f9	
4	(i0) → f5	(i9) → f0
5	f2 → (i1)	(i11) → f1
6	(i2) → f9	
7	(i3) → f5	(i8) → f10
8	(i4) → f2	(i14) → f5
9	f6 → (i0)	(i15) → f5
10	f1 → (i2)	f10 → (i8)
11		f8 → (i14)
12		
13		(i10) → f12
14		f9 → (i15)
15		(i11) → f12
16	f4 → (i3)	f8 → (i10)
17		(i9) → f10
18		f9 → (i11)
19		f10 → (i9)

Table 5: Core of the assembly program. The values of the modifiers are:

$m0 = m8 = M + 1$, $m1 = m9 = M + 2$, $m5 = m13 = 0$, $m6 = m14 = 1$ and $m15 = -1$

1)	flt:	f12 = f0*f4,	f10 = f10-f12,	f0 = dm(i1,m5),	f4 = pm(i9,m8);
2)		f4 = f0*f4,	f10 = abs f10,	f8 = dm(i1,m6),	f5 = pm(i10,m13);
3)		f13 = f0*f5,	f2 = f8-f12,	f9 = dm(i0,m0),	pm(i9,m13) = f10;
4)		f8 = f0*f4,	f6 = f9-f13,	f5 = dm(i0,m6),	f0 = pm(i9,m13);
5)		f0 = f0*f5,	f8 = f8+f15,	dm(i1,m5) = f2,	f1 = pm(i11,m15);
6)		f14 = f1*f5,		f9 = dm(i2,m5),	pm(i11,m14) = f3;
7)		f9 = f0*f5,	f1 = f9-f14,	f5 = dm(i3,m5),	f10 = pm(i8,m13);
8)		f12 = f2*f5,		f2 = dm(i4,m5),	f5 = pm(i14,m13);
9)		f13 = f2*f5,	f10 = f10+f12,	dm(i0,m0) = f6,	f5 = pm(i15,m13);
10)		f14 = f2*f5,	f8 = f8+f13,	dm(i2,m5) = f1,	pm(i8,m14) = f10;
11)			f2 = recip f8,		pm(i14,m14) = f8;
12)		f2 = f2*f4,	f9 = f9+f14,		f10 = pm(i9,m9);
13)		f8 = f2*f6,	f9 = f9+f15,		f12 = pm(i10,m13);
14)			f4 = recip f9,		pm(i15,m14) = f9;
15)		f4 = f0*f4,	f8 = f8+f12,		f12 = pm(i11,m14);
16)		f9 = f1*f4,		dm(i3,m6) = f4,	pm(i10,m14) = f8;
17)	flt0:	f12 = f0*f4,	f3 = f9+f12,	f4 = dm(i0,m5),	f0 = pm(i8,m13);

were then fed to the DSP system and error $\alpha_M(n)$ was recorded. A forgetting factor $\lambda = 0.999$ was used. The instantaneous echo return loss enhancement (ERLE) given by

$$\text{ERLE} = 10 \log[\overline{d^2(n)} / \alpha_M^2(n)]$$

was calculated. Figure 2 shows 2s of the echo signal prior to cancellation and the obtained ERLE. Each ERLE value shown is the mean of 64 consecutive measurements. We observe that the ERLE range is approximately 30-40 dB, a good result considering the order of the echo canceller, the quantization of denominators of Equations (9,10) of Table 1 and the usual noise levels in hands-free mobile telephony environments. No noise was superposed to the echo signal in order to show the stable behaviour of the echo canceller. It can be seen that the algorithm remains stable, without the need for additional control measures, even when the voice signal decays to almost zero.

Figure 3 is similar to Figure 2 but considering an echo canceller with 1024 coefficients. In this case the described algorithm was implemented by a C routine embedded in MatLab and using the same numerical conventions as the DSP. It can be seen that the obtained ERLE range is improved due to the higher number of coefficients and that algorithm still works well. It should be observed that in this case the maximum attainable ERLE is limited due to mantissa's quantization before division. Similar results have been obtained for higher orders and longer echo impulse responses.

Fig. 1. Measured impulse response (512 samples).

4. REFERENCES

- [1] A. Gilloire, E. Moulines, D. Slock, P. Duhamel: "State of the Art in Acoustic Echo Cancellation", in A.R. Figueiras-Vidal (Ed.), *Digital Signal Processing in Telecommunications*, Springer, 1996, pp.45-91.
- [2] P.A. Regalia: "Numerical properties of a QR-based fast least squares algorithm", *IEEE Trans. on Signal Processing*, June 1993, vol.41, no.6, pp.2096-2109.
- [3] M.D. Miranda, M. Gerken: "Performance of the a priori and a posteriori QR-LSL algorithms in a limited precision environment", *Proceedings ICASSP97*, April 1997, Munique, pp.2337-2340.
- [4] M. Miranda, M. Gerken: "Optimized implementation of the a priori LSL algorithm with error feedback" (in Portuguese), *Proceedings of the 15th Brazilian Telecommunications Symposium*, Recife, 1997, pp. 280-284.
- [5] M. Miranda, M. Gerken, M.T.M. da Silva: "Efficient implementation of error-feedback LSL algorithm", *Electronics Letters*, 1999, pp. 1308-1309.
- [6] A. H. C. Carezia, M. Gerken, P. M. S. Burt, M. T. M. da Silva: "Efficient Implementation of a LSL Algorithm for Acoustic Echo Cancelling in Automobiles" (in Portuguese), *Proceedings of the 17th Brazilian Telecommunications Symposium*, Vila Velha, 1999, pp. 372-377.
- [7] F. Ling, D. Manolakis, J.G. Proakis: "Numerically robust least-squares lattice-ladder algorithms with direct updating of the reflection coefficients", *IEEE Trans. on ASSP*, August 1986, vol.34, no.4, pp.837-845.
- [8] S. Haykin: *Adaptive filter theory*, Prentice Hall Int., 2nd Ed., 1991.
- [9] ADSP-2106x (SHARC) User's Manual, 2nd Ed., Analog Devices Inc., July 1996.

Fig. 2. Result of AEC experiment using ADSP-21061 (290 coefficients). Bottom trace: echo signal prior to cancellation.

Fig. 3. Result of AEC experiment using MatLab (1024 coefficients). Bottom trace: echo signal prior to cancellation.

Referências Bibliográficas

- [1] ANTWEILER, C. Adaptation of acoustic echo cancelers exploiting spectral characteristics of room impulse responses. In: *Proceedings of IWAENC '97*. London: [s.n.], 1997. p. 57–60.
- [2] GUSTAFSSON, S.; MARTIN, R. Combined acoustic echo control and noise reduction based on residual echo estimation. In: *Proceedings of IWAENC '97*. London: [s.n.], 1997. p. 160–163.
- [3] IHLE, M.; KROSCHER, K. Integration of noise reduction and echo attenuation for handset-free communication. In: *Proceedings of IWAENC '97*. London: [s.n.], 1997. p. 69–72.
- [4] CAPMAN, F.; BOUDY, J.; LOCKWOOD, P. Acoustic echo cancellation and noise reduction in the frequency-domain: A global optimisation. In: *Proceedings of EUSIPCO '96*. Trieste: [s.n.], 1996. p. 1–4.
- [5] MARTIN, R.; VARY, P. Combined acoustic echo control and noise reduction for hands-free telephony - state of the art and perspectives. In: *Proceedings of EUSIPCO*. Trieste: [s.n.], 1996. p. 1107–1110.
- [6] TELLABS, INC. Echo cancellation tutorial. mar. 1999. Disponível em: <<http://www.webproforum.com/tellabs/full.html>>.
- [7] GABREA, M. et al. Two microphones speech enhancement system based on a double fast recursive least squares (DFRLS) algorithm. In: *Proceedings of EUSIPCO*. Trieste: [s.n.], 1996. p. 983–986.

- [8] KIM, K.-S.; PARK, K.-J.; YOUN, D.-H. Degraded audio enhancement based on spectral envelope estimation. In: *Proceedings of ICS-PAT '97*. [S.l.: s.n.], 1997. p. 285–288.
- [9] KUO, S. M.; CHUANG, H.; MALLELA, P. P. Integrated automotive signal processing and audio system. *IEEE Transactions on Consumer Electronics*, v. 39, n. 3, p. 522–531, ago. 1993.
- [10] CHAN, D. C. B.; RAYNER, P. J. W.; GODSILL, S. J. Multi-channel signal separation. In: *Proceedings of ICASSP '96*. [S.l.: s.n.], 1996. v. 2, p. 649–652.
- [11] DOUKAS, N.; STATHAKI, T.; NAYLOR, P. Speech enhancement thorough nonlinear adaptive source separation methods. In: *Proceedings of SSAP '96*. Corfu, Greece: [s.n.], 1996.
- [12] WAN, E. A.; NELSON, A. T. Removal of noise from speech using the dual EKF algorithm. In: *Proceedings of ICASSP '98*. [S.l.: s.n.], 1998. v. 1, p. 381–384.
- [13] GODSILL, S. J. *The Restoration of Degraded Audio Signals*. Tese (Doutorado) — University of Cambridge, dez. 1993.
- [14] GODSILL, S. J. Bayesian enhancement of speech and audio signals which can be modelled as ARMA processes. *International Statistical Review*, v. 65, n. 1, p. 1–21, 1997.
- [15] GOH, Z.; TAN, K.-C.; TAN, B. Speech enhancement based on a voiced-unvoiced speech model. In: *Proceedings of ICASSP '98*. [S.l.: s.n.], 1998. v. 1, p. 401–404.
- [16] HAYKIN, S. *Adaptive Filter Theory*. 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1991. (Information and System Sciences Series).
- [17] MUELLER, M. S. Least-squares algorithms for adaptive equalizers. *Bell Syst. Tech. J.*, v. 60, p. 1906–1925, dez. 1984.

- [18] MORF, M.; KAILATH, T.; LJUNG, L. Fast algorithms for recursive identification. In: *Proc. Conf. Decision and Control*. Clearwater Beach, Fla.: [s.n.], 1976. p. 916–921.
- [19] LJUNG, L.; MORF, M.; FALCONER, D. Fast calculation of gain matrices for recursive estimation schemes. *Intern. J. Control*, v. 27, p. 1–19, 1978.
- [20] LING, F.; MANOLAKIS, D.; PROAKIS, J. G. New forms of LS lattice algorithms and an analysis of their round-off error characteristics. In: *Proceedings of ICASSP*. Tampa, Fla.: [s.n.], 1985. p. 1739–1742.
- [21] MIRANDA, M.; GERKEN, M. Implementação otimizada para o algoritmo LSL *a priori* com realimentação de erro. In: *Anais do XV simpósio brasileiro de telecomunicações*. Recife: [s.n.], 1997. p. 280–284.
- [22] VARY, P. Noise suppression by spectral magnitude estimation - mechanism and theoretical limits. *Signal Processing*, v. 8, p. 387–400, 1985.
- [23] VARY, P.; WACKERSREUTHER, G. A unified approach to digital polyphase filter banks. *AEÜ*, v. 37, n. 1/2, p. 29–34, 1983.
- [24] FLIEGE, N. J. *Multirate Digital Signal Processing*. Chichester, England: John Wiley & Sons, 1994.
- [25] OPPENHEIM, A. V.; SCHAFER, R. W. *Discrete Time Signal Processing*. [S.l.]: Prentice Hall, 1989.
- [26] HARRIS, F. J. On the use of windows for harmonic analysis with the discrete fourier transform. *Proc. IEEE*, v. 66, n. 1, p. 51–83, jan 1978.

- [27] ALKULAIBI, A.; SORAGHAN, J. J.; DURRANI, T. S. Fast 3-level binary higher order statistics for simultaneous voiced/unvoiced and pitch detection of a speech signal. *Signal Processing*, v. 63, n. 2, p. 133–140, 1997.
- [28] POLLÁK, P.; SOVKA, P. Cepstral speech/pause detectors. In: *IEEE Workshop on Nonlinear Signal and Image Processing*. Neos Marmaras, Halkidiki, Greece: [s.n.], 1995. p. 388–391.
- [29] MARTIN, R. An efficient algorithm to estimate the instantaneous SNR of speech signals. In: *Proceedings of EUROSPEECH '93*. Berlin: [s.n.], 1993. p. 1093–1096.
- [30] EPHRAIM, Y.; MALAH, D. Speech enhancement using a minimum mean-square error short-time spectral amplitude estimator. *assp*, v. 32, n. 6, p. 1109–1121, dez. 1984.
- [31] VIRAG, N. Speech enhancement based on masking properties of the auditory system. In: *Proceedings of ICASSP '95*. [S.l.: s.n.], 1995. p. 796–799.
- [32] AZIRANI, A. A.; JEANNÈS, R. L. B.; FAUCON, G. Optimizing speech enhancement by exploiting masking properties of the human ear. In: *Proceedings of ICASSP '95*. [S.l.: s.n.], 1995. p. 800–803.
- [33] GUSTAFSSON, S.; JAX, P.; VARY, P. A novel psychoacoustically motivated audio enhancement algorithm preserving background noise characteristics. In: *Proceedings of ICASSP '98*. [S.l.: s.n.], 1998. p. 397–400.
- [34] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *ISO/IEC 11172-3: Information Technology—Coding of Moving Pictures and Associated Audio for Digital Storage Media at Up to About 1.5 Mbits/S—Part 3: Audio*. [S.l.], 1993.

- [35] SCHROEDER, M. R.; ATAL, B. S.; HALL, J. L. Optimizing digital speech coders by exploiting masking properties of the human ear. *Journal of Acoustical Society of America*, v. 66, n. 6, p. 1647–1652, dez. 1979.
- [36] JOHNSTON, J. D. Transform coding of audio signals using perceptual noise criteria. *IEEE Journal on Selected Areas in Communications*, v. 6, n. 2, p. 314–323, fev. 1988.
- [37] SCHARF, B. Critical bands. In: TOBIAS, J. (Ed.). *Foundations of Modern Auditory Theory*. New York: Academic Press, 1970. v. 1, cap. 5, p. 159–202.
- [38] ANALOG DEVICES. *ADSP-2106x SHARC User's Manual*. 2nd ed. [S.l.], jul. 1996.
- [39] PRESS, W. H. et al. *Numerical Recipes in C - The Art of Scientific Computing*. 2nd ed. [S.l.]: Cambridge University Press, 1992.

Este texto foi tipografado em corpo 12 no sistema TeX, usando as fontes NewCenturySchoolbook e AvantGarde.
--