UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

ESCOLA POLITÉCNICA
DEPARTAMENTO DE ELETRÔNICA E DE COMPUTAÇÃO

Adaptação de Codificador CELP à Transmissão de Voz sobre IP

Autor: Eduardo Fonseca Brasil

Orientador: Sérgio Lima Netto

Examinador: Eduardo Antônio Barros da Silva

Examinador: Marcelo Luiz Drumond Lanza

DEL
Agosto/2006

Aos meus pais...

Agradecimentos

Agradeço:

- Ao professor Sérgio Lima Netto, pela apoio e orientação de iniciação científica e de projeto final. Obrigado por todo o apoio, incentivo e, por último, mas não por isso menos importante, pela paciência;
- Aos professores Eduardo Antônio Barros da Silva e Marcelo Luiz Drumond Lanza, por terem examinado o projeto, e participado da banca de avaliação;
- Aos meus professores, que se tornaram amigos a serem guardados e modelos a serem seguidos;
- Aos meus colegas de turma, com quem tanto aprendi e me diverti;
- Ao Departamento de Engenharia Eletrônica e de Computação (DEL/UFRJ) e ao Laboratório de Processamento de Sinais (LPS/COPPE), por oferecerem a infra-estrutura necessária para o desenvolvimento do trabalho;
- À Natasha, por seu companheirismo e por seu apoio;
- E aos meus pais, por todo o incentivo recebido.

Palavras-chave

- Voz sobre IP
- VoIP
- Codificação de Voz
- Codificação CELP
- Filtragem Adaptativa

Siglas e Abreviaturas

- ADPCM - Adaptive Differential Pulse Code Modulation
- ALSA - Advanced Linux Sound Architecture
- CCITT - Comite Consultatif Internationale de Telegraphie et Telephonie
- CELP - Code Excited Linear Prediction
- DPCM - Differential Pulse Code Modulation
- FLAC - Free Lossless Audio Codec
- IEEE - Institute of Electrical and Electronics Engineers
- IETF - Internet Engineering Task Force
- IP - Internet Protocol
- ISO - International Standards Organization
- LPC - Linear Predictive Coding
- LSF - Line Spectral Frequencies
- LZW - Lempel-Ziv-Welch
- MPEG - Moving Picture Experts Group
- OSI - Open Systems Interconnection
- PCM - Pulse Code Modulation
- RFC - Request for Comments
- TCP - Transmission Control Protocol
- UDP - User Datagram Protocol
- VoIP - Voice over IP

Resumo

Adaptação de um codificador CELP para a transmissão de voz pela Internet. Implementou-se um sistema de comunicação TCP e UDP entre dois computadores ligados à Internet, pelo qual é possível enviar uma sequência de bits contendo informações de voz codificada. Adicionalmente, realizou-se o estudo, a implementação e testes de um sistema de cancelamento de eco por meio de filtragem adaptativa, utilizando-se diferentes algoritmos. O estudo teórico sobre filtragem adaptativa foi acrescentado ao projeto, assim como a comparação entre sete tipos de algoritmos de adaptação.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Objetivo do Trabalho

O trabalho desenvolvido nesse projeto final buscou adaptar um codificador de voz CELP (*Code Excited Linear Prediction*) para a transmissão de voz pela Internet. A codificação CELP é utilizada para reduzir o volume necessário de dados a ser trafegado, de forma a adequar essa transmissão às limitações impostas pela Internet, onde geralmente encontra-se problemas de latência e de limitação de banda.

1.2 Motivação

A principal motivação do trabalho foi o alto custo de ligações telefônicas de longa distância, realizadas pelo sistema telefônico convencional. Os sistemas de voz sobre IP (*Internet Protocol*) permitem que esses altos custos sejam evitados, uma vez que as transmissões de longa distância são realizadas pela Internet.

1.3 Organização do Trabalho

O trabalho foi dividido em quatro capítulos de conteúdo, além da introdução, conclusão e apêndices. O conteúdo discutido em cada capítulo foi:

1.3.1 A Codificação CELP

No Capítulo 2, a teoria por trás da codificação CELP será apresentada. Nesse capítulo, todo o algoritmo utilizado pelo codificador será apresentado e discutido.

O codificador CELP utilizado pelo trabalho foi inicialmente desenvolvido por alunos dos Profs. Sérgio Lima Netto e Fernando Gil Vianna Resende Júnior, no Departamento de Egenharia Eletrônica e de Computação (DEL/UFRJ) e no Laboratório de Processamento de Sinais (LPS/COPPE).

1.3.2 Transmissão de Voz sobre IP

Após detalhar o funcionamento da codificação CELP, a infra-estrutura utilizada para transmitir o seu resultado pela Internet foi explicada. O Capítulo 3 apresenta a forma escolhida para transferir, pela Internet, os dados do codificador, e justifica as escolhas feitas.

Esse capítulo também apresenta uma segunda possibilidade de uso do sistema, onde a transmissão de voz é feita entre Internet e o sistema telefônico tradicional. Nesse caso, um servidor auxiliar é utilizado, e é responsável por realizar essa interconexão.

1.3.3 Filtros Adaptativos

Os testes do sistema de transmissão de voz sobre IP montados apresentaram um problema de eco, que inviabilizava a comunicação. Uma possível solução para esse problema é utilizar filtros adaptativos.

O capítulo 4 apresenta a teoria por trás da filtragem adaptativa, e também como eles podem ser utilizados para realizar o cancelamento de eco.

Nesse capítulo, 7 tipos diferentes de filtros adaptativos são apresentados, e suas propriedades comparadas.

1.3.4 Filtros Adaptativos - Resultados

No capítulo 5, os 7 tipos de filtros apresentados no capítulo anterior foram testados, variando-se as suas propriedades, de forma a determinar a influência sobre o resultado dos principais parâmetros que o projetista utiliza.

1.4 Status atual da Voz sobre IP

O sistema implementado nesse projeto é semelhante a outros sistemas de Voz sobre IP existentes. Em especial, pode-se citar o sistema Skype e a sua funcionalidade chamada de SkypeOut [19], que permite realizar chamadas da Internet para linhas telefônicas convencionais, com baixas tarifas e alta qualidade de voz.

Também existe um sistema de Voz sobre IP no Núcleo de Computação Eletrônica (NCE) da UFRJ [6]. Esse sistema permite chamadas gratuitas da Internet para telefones localizados no Rio de Janeiro.

Capítulo 2

A Codificação CELP

2.1 Introdução

Nesse capítulo serão apresentados os principais conceitos de transmissão e armazenamento analógico e digital de dados. Atenção especial será dada à transmissão digital de voz, que é o foco desse trabalho. A codificação CELP será apresentada, e seus diversos passos serão detalhados.

Na seção 2.2, e nas suas sub-seções, serão apresentados os principais conceitos referentes à compressão de dados. Na sub-seção 2.2.1 as propriedades da transmissão digital de dados serão discutidas, enquanto a relação entre codificação e transmissão será discutida na sub-seção 2.2.2.

Na seção 2.3, os conceitos de compressão de dados serão aplicadas a sinais de voz. A codificação de voz CELP, na qual se baseia o trabalho, será discutida na seção 2.4, enquanto a decodificação será apresentada na seção 2.5.

2.2 Compressão de Dados

Podemos dividir as formas de transmissão e armazenamento de sinais em dois grandes grupos, a transmissão analógica e a digital. Na primeira, mais antiga, os sinais são armazenados e/ou transmitidos, de forma geral, como valores contínuos. Para isso, utiliza propriedades contínuas do meio onde é feita a transmissão e/ou armazenamento.

2.2.1 Transmissão Digital

Com o surgimento da eletrônica digital, novos sistemas de transmissão foram criados, para contornar as limitações do meio com maior eficiência. Nesses sistemas, a informação é transmitida como uma seqüência de valores discretos, e não mais contínuos. Em especial, são utilizados valores binários, bits, que são lidos pelo receptor, e transformados novamente em um sinal analógico.

Em algum momento da transmissão digital, os dados ainda passarão por um meio analógico. Os bits de um sinal digital de voz precisam ser transformados, por exemplo, em um sinal de tensão, para serem transmitidos por uma fio de cobre. Entretanto, diferentemente da transmissão analógica, os bits são resistentes ao ruído. Como exemplo, podemos imaginar um sistema onde um bit zero é representado por sinal de $0V$, e um bit 1 é representado por um sinal de $5V$. Ainda que exista ruído, e desde que ele seja pequeno comparado ao valor de $5V$, o receptor conseguirá entender o que o transmissor enviou, sem erros. Podemos imaginar que o receptor "aceite" sinais entre $-1V$ e $+1V$ como sendo um bit 0, e sinais entre $+4V$ e $+6V$ como sendo um bit 1. Dessa forma, ainda que exista um ruído aleatório, e desde que ele tenha módulo menor do que $1V$, o receptor conseguirá transcrever, sem erros, o que foi enviado. Um sinal de $0,7V$ seria corretamente interpretado como um bit 0, da mesma forma que um sinal de $4,6V$ seria corretamente interpretado como um bit 1. Nos dois casos, a diferença entre os valores nominais ($0V$ e $5V$) seria resultado do ruído, que, nesse caso, não causaria nenhum erro de transmissão.

A transmissão digital é, portanto, uma forma eficiente de transmitir informação, pois contorna as limitações impostas pela presença de ruído. As formas como esses bits são transformados em sinais analógicos, para a transmissão, não são parte desse trabalho, e podem ser econtrados em [10]. Inicialmente, é necessário definir como os sinais que nos interessam (voz, vídeo, música, etc) são transformados em bits. Manteremos o foco na voz, que é a proposta desse trabalho.

2.2.2 Codificação e Transmissão de Dados

O sinal de voz, analógico, pode ser medido pela variação de pressão do ar, durante um determinado período de tempo. Ele pode ser digitalizado realizando-se uma amostragem dessa pressão em intervalos regulares (período de amostragem). Dessa amostragem

resultará uma seqüência de valores não-discretos. O sinal que, originalmente, era uma função variando no tempo, $f(t)$, é transformado em uma seqüência, $x[n]$. O passo final é transformar cada um dos valores dessa seqüência em bits.

Individualmente, um bit pode representar apenas dois valores, 0 ou 1. Para representar valores maiores, é necessário utilizar um seqüência de bits. Sequências de oito bits, por exemplo, podem ser utilizadas para representar 256 valores diferentes. Para realizar o passo final da digitalização podemos, portanto, mapear cada valor da seqüência $x[n]$ em um número discreto de 0 a 255, no caso de utilizar-se oito bits. Para realizar essa transformação, pode-se definir o menor valor de pressão como uma seqüência de oito bits 0, e o maior valor de pressão como uma seqüência de oito bits 1. Os valores intermediários seriam mapeados dentro desse intervalo.

Na codificação básica de voz telefônica, a PCM [10], realiza-se a digitalização utilizando uma freqüência de amostragem de 8 kHz, e cada amostra é transformada em 8 bits. Dessa forma são gerados, ao total, 64 mil bits por segundo de voz digitalizada. É razoável esperar que o custo da transmissão cresça junto com o número de bits a ser transmitida. Dessa forma, torna-se interessante reduzir o número de bits transmitidos. As técnicas de compressão de dados nos permitem fazê-lo.

As formas de compressão de dados dividem-se em dois grandes grupos: compressão sem perdas e compresão com perdas. Na compressão sem perdas, o sinal pode ser recuperado perfeitamente, sem erros. Um exemplo de compressão sem erros é o sistema de compressão de áudio FLAC [5]. Ele retira algumas redundâncias do sinal de voz, garantindo que a partir do sinal comprimido possa ser extraído uma transcrição perfeita do sinal original, bit a bit.

Já os sistemas de compressão com perda levam em consideração a capacidade de percepção do usuário do sinal. Certos erros em um sinal de música, por exemplo, não são perceptíveis pelo "usuário" do sinal, o ouvido humano. Dessa forma, os algoritmos de compressão com perda, como o MP3 [7], não permitem que seja realizada a transcrição perfeita do sinal original mas gera erros imperceptíveis para o ouvido humano. Esses erros são compensados pela diminuição do número de bits necessários para a transmissão ou armazenamento, e, portanto, do seu custo.

2.3 Compressão de Voz

Como visto, transmitir a voz sem qualquer tipo de compressão/codificação resulta em um esforço maior do que o necessário. O sinal de voz apresenta muitas redundâncias que podem ser retiradas antes da transmissão, diminuindo, assim, a quantidade total de dados a ser transmitidos. É possível utilizar um algoritmo genérico de compressão de dados, como o LZW [8], para comprimir a voz. Intuitivamente, pode-se acreditar que essa abordagem não seja a mais eficiente. Um algoritmo específico para compressão de voz deve ser melhor, ou pelo menos tão bom quanto um algoritmo de compressão genérico. Além disso, os algoritmos de compressão de dados genéricos são, em geral, sem perdas, o que não é necessário para a transmissão de voz. Para a voz, uma transmissão com perdas, desde que controladas, pode ser suficiente. Também é intuitivo que aceitando algumas perdas pode-se obter ganhos na eficiência do algoritmo, que pode requerer menor poder de processamento ou menor largura de banda.

Assim, é natural que um algoritmo de codificação/compressão específico para a voz seja procurado. Para obtê-lo, precisamos primeiro entender as características do sinal de voz, para sabermos o que devemos considerar ao projetar o algoritmo.

2.3.1 Propriedades da Voz

A produção natural da voz pode ser dividida em, fundalmentamente, dois tipos de processos. No primeiro, os pulmões geram uma excitação, de características aleatórias. No segundo, essa excitação passa por diversos "tubos" do sistema da fala e pelas cordas vocais. O conjuto de tubos e cordas vocais compõe o segundo processo da produção da fala.

Nos sons sonoros, as cordas vocais vibram de forma quase periódica, alterando as características da excitação aleatória enviada pelos pulmões. Já nos sons surdos, as cordas vocais permanecem abertas, não alterando a excitação produzida pelos pulmões. Sons surdos apresentam características ruídosas.

Os diversos "tubos" do sistema fonador (o trato vocal) também alteram a excitação gerada pelos pulmões. Eles modificam a distribuição de energia no espectro, introduzindo ressonâncias e antiressonâncias [2].

Podemos representar o sistema fonador como um filtro digital, conforme mostrado na

figura 2.1.

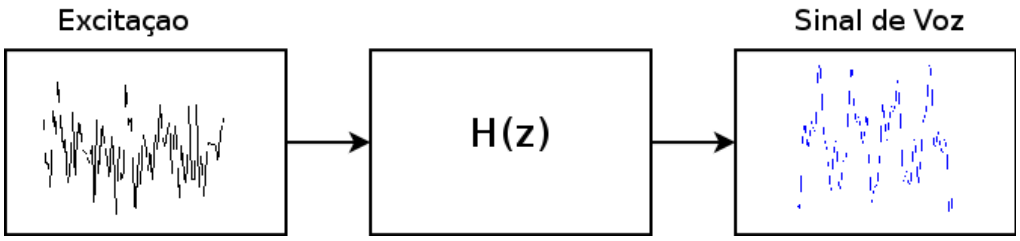

Figura 2.1: Sistema fonador representado como filtro digital.

O "filtro" formado pelo trato vocal não é constante. Ele varia conforme falamos. Entretanto, para efeitos práticos, podemos considerá-lo invariável para pequenos períodos, como intervalos de 10 a 30 ms [1].

Esse modelo de produção de voz será utilizado na construção dos codificadores que serão estudados.

2.3.2 Tipos de Codificadores de Voz

Podemos dividir os codificadores de voz em três grandes famílias:

- Codificadores de forma de onda;
- Codificadores paramétricos;
- Codificadores híbridos;

Os da primeira família, como o PCM, se limitam a transcrever o sinal analógico para forma digital. Os valores digitais podem ser diretamente convertidos para valores analógicos, e vão gerar o sinal de voz novamente. Além do PCM, também fazem parte dessa família o DPCM e o ADPCM [10].

Os codificadores paramétricos não tentam transmitir o sinal em si, e sim informações necessárias para re-criar o sistema que gerou o sinal. Vimos na subseção 2.3.1, que podemos descrever a criação do sinal de voz como uma filtragem digital de um sinal aleatório, e que esse filtro pode ser considerado constante por pequenos intervalos de tempo. A codificação paramétrica, ao invés de transmir o sinal em si, tenta, a partir do sinal, gerar um modelo de filtro que, ao ser excitado por um sinal aleatório, gere novamente o sinal de voz de entrada, ou um sinal suficientemente semelhante. Gerando esse modelo de filtro, o codificador paramétrico transmite os coeficientes do modelo, e o filtro é reconstituído

pelo receptor, que gera o sinal de voz. Utilizando intervalos de, por exemplo, 20 ms, um novo modelo de filtro teria que ser gerado e transmitido a cada 20 ms, para que o receptor pudesse ser atualizado adequadamente. O principal representante dessa família é o LPC. [9]

A codificação híbrida une as duas famílias anteriores. Como a codificação CELP, objetivo desse estudo, é um exemplar de codificação híbrida, maiores detalhes serão apresentados sobre essa família.

Os codificadores de forma de onda apresentam, em geral, alta qualidade de voz. O sinal reconstituído é fiel ao original. Entretanto, essa família requer altas taxas de transmissão. Já os codificadores paramétricos apresentam taxas baixas, e fidelidade igualmente baixa. Os codificadores híbridos, por se apropriarem do que as duas outras famílias apresentam de melhor, conseguem obter um compromisso razoável entre fidelidade e taxa de transmissão, conforme mostra a figura 2.2:

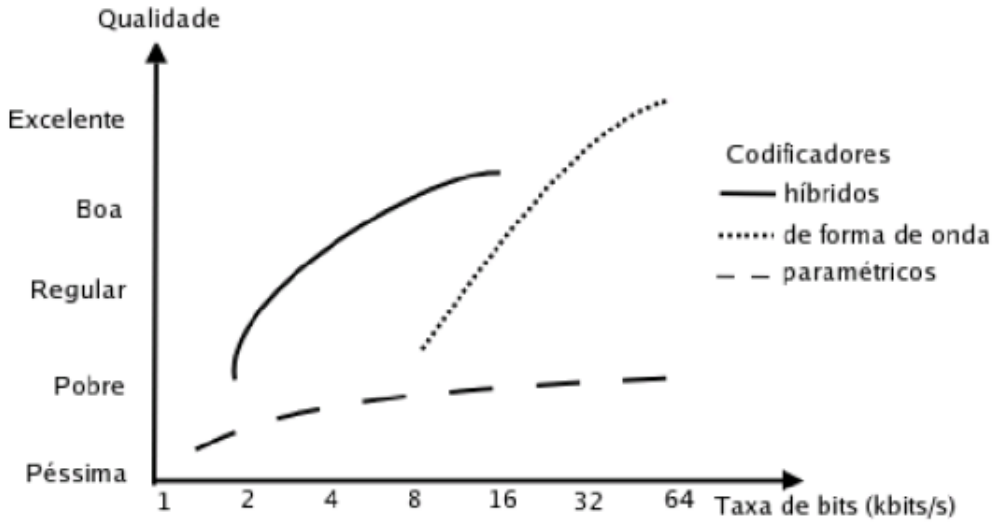

Figura 2.2: Comparação dos três principais tipos de codificadores, conforme visto em [2].

2.4 A codificação CELP

Iniciaremos, agora, uma discussão completa sobre o algoritmo de codificação utilizado nesse trabalho.

A figura 2.3 mostra as principais etapas da codificação. Posteriormente, cada passo será detalhado.

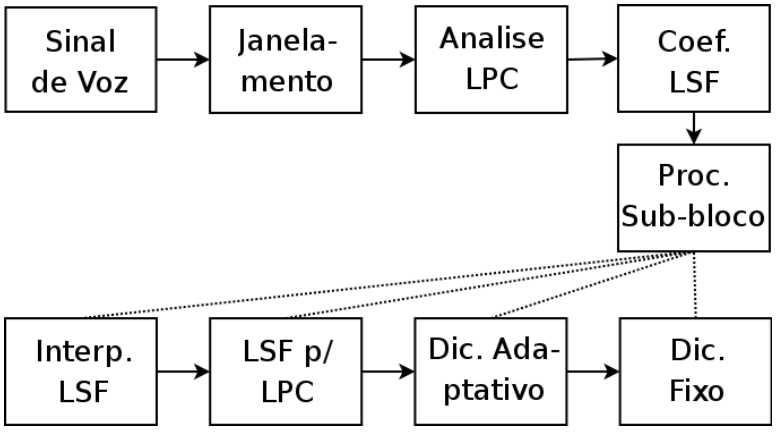

Figura 2.3: Principais etapas da codificação CELP.

2.4.1 Obtenção da Voz Digitalizada

O primeiro passo da codificação é obter o sinal de voz que desejamos codificar. Esse sinal é capturado de forma analógica, e convertido para um sinal digital, com freqüência de amostragem de 8 kHz, utilizando 8 bits por amostra. Esse sinal será chamado de sinal-alvo. O objetivo da codificação é que o decodificador consiga reproduzir um sinal suficientemente semelhante ao sinal-alvo.

Como visto, só podemos considerar o trato vocal como um sistema estacionário para pequenos intervalos. Por isso, na codificação CELP, trataremos a voz em janelas de 20 ms, o que corresponde a 160 amostras. Para cada uma dessas janelas, o trato vocal poderá ser modelado por um filtro estacionário.

2.4.2 Janelamento

Em sistemas de processamento digital de sinais, é comum aplicar uma função-janela ao sinal a ser processado. Isso evita que o processamento "enxergue" altas-freqüências nas bordas do sinal [4].

Para esse codificador, uma janela de Hamming é utilizada. Uma seqüência de janela de Hamming pode ser gerada da seguinte forma:

$$w_H[n] = \begin{cases} \alpha + (1-\alpha)cos(\frac{2\pi n}{M}) & \text{se } |n| \leq \frac{M}{2}; \\ 0 & \text{se } |n| > \frac{M}{2}. \end{cases} \tag{2.1}$$

onde $\alpha = 0,54$, $M = (N-1)$ e $N = 160$ (número de amostras em uma janela).

Dado um sinal digital $s[n]$, o sinal janelado é obtido por $s_W[n] = w_H[n]s[n]$.

2.4.3 Filtro de Predição Linear

O filtro de predição linear, originário da codificação LPC, busca achar os coeficientes necessários para relacionar uma amostra com um determinado número de amostras anteriores (correlações de curto-termo).

Inicialmente, supomos que uma amostra $x[n]$ de um sinal de voz possa ser calculada a partir das amostras anteriores da seguinte forma:

$$x[n] = \sum_{i=1}^{p} a_i x[n-i] \tag{2.2}$$

Essa relação pode ser representada por um filtro digital, só-pólos:

$$H(z) = \frac{1}{A(z)} = \frac{1}{1 - \sum_{i=1}^{p} a_i z^{-i}} \tag{2.3}$$

Dado um sinal de voz janelado, $s_W[n]$, calculamos os coeficientes a_i que minimizariam o erro médio quadrático desse modelo. O erro médio quadrático para um determinado conjunto de coeficientes pode ser calculado por:

$$E_p = E\left[e_w^2[n]\right] = E\left[\left(s_w[n] - \sum_{i=1}^{p} a_i s_w[n-i]\right)^2\right] \tag{2.4}$$

Para determinar os valores a_i que minimizam o erro E_p, faz-se:

$$\frac{\partial E_p}{\partial a_i} = 0 \tag{2.5}$$

Resolvendo essa derivada parcial, obtemos:

$$\sum_{i=1}^{N} a_i r_s[k-i] = r_s[k] \tag{2.6}$$

onde $r_s[k]$ é a autocorrelação do sinal de voz janelado, $s_W[n]$.

Podemos representar a equação anterior em notação matricial, da seguinte forma:

$$\begin{bmatrix} r_s[0] & r_s[1] & \cdots & r_s[p-1] \\ r_s[1] & r_s[0] & \cdots & r_s[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ r_s[p-1] & r_s[p-2] & \cdots & r_s[0] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = \begin{bmatrix} r_s[1] \\ r_s[2] \\ \vdots \\ r_s[p] \end{bmatrix} \tag{2.7}$$

Ela pode ser representada de forma simplificada por $\boldsymbol{R_s a} = \boldsymbol{r_s}$, e pode ser resolvida por $\boldsymbol{a} = \boldsymbol{R}_s^{-1}\boldsymbol{r}_s$. O vetor $\boldsymbol{r_s}$, de dimensão $p \times 1$, representa os valores de autocorrelação do sinal de voz janelado e o vetor $\boldsymbol{a}$, também de dimensão $p \times 1$, representa os coeficientes de predição linear que minimizam o erro médio quadrático.

A estrutura da matriz $\boldsymbol{R}$, $p \times p$, é conhecida como matriz de *Toeplitz*. Nesse tipo de matriz, os elementos de cada diagonal são iguais. Para o cálculo do vetor de coeficientes $\boldsymbol{a}$ é necessário inverter essa matrix. Como esse cálculo precisa ser feito para cada conjunto de 160 amostras, optou-se pela utilização de um algoritmo rápido de inversão de matrizes, específico para matrizes de *Toeplitz*. O método usado é chamado de *Levinson-Durbin* [25].

Um fator importante para definir o filtro de predição linear é a sua ordem. Quanto mais amostras anteriores utilizarmos para a predição da amostra atual, menor será o erro. Entretanto, precisaremos transmitir mais um coeficiente, já que o filtro linear utiliza uma amostra por coeficiente. Devido às características do sinal de voz, optou-se por utilizar $p = 10$ coeficientes para o filtro de predição linear.

2.4.4 Coeficientes LSF

Os coeficientes LPC não são os mais apropriados para a transmissão de voz, já que são muito sensíveis à quantização. A alternativa utilizada no codificador apresentado são os coeficientes LSF (*Line Spectral Frequencies*).

Os coeficientes LSF (representados pelos polinômios $P(z)$ e $Q(z)$) relacionam-se com os coeficientes LPC (representados pelo polinômio $A(z)$) pelas seguintes expressões:

$$P(z) = A(z) + z^{-(p+1)}A(z^{-1}) \tag{2.8}$$

$$Q(z) = A(z) - z^{-(p+1)}A(z^{-1}) \tag{2.9}$$

A partir das duas equações obtemos:

$$A(z) = \frac{1}{2}\left[P\left(z\right) + Q\left(z\right)\right] \tag{2.10}$$

Os coeficientes LPC são representados pelos zeros do polinômio $A(z)$, enquanto os coeficientes LSF que precisam ser transmitidos são 5 raízes do polinômio $P(z)$ e outras 5 do polinômio $Q(z)$. Esses polinômios possuem outras raízes, os conjugados complexos, que não precisam ser transmitidas.

Ao final desse passo possuímos um conjunto de 10 coeficientes LSF que podem ser transmitidos e substituem os coeficientes LPC.

Até aqui todos os coeficientes foram tratados como números reais, com precisão infinita. Ainda que seja possível transmitir números de ponto flutuante pela Internet, utilizando os padrões IEEE, essa transmissão seria de baixíssima eficiência, já que as representações de números reais utilizam grande quantidades de bits, o que aumentaria bruscamente a quantidade de dados transmitidos. Para resolver esse problema, o próximo passo é a quantização dos coeficientes LSF.

2.4.5 Quantização dos Coeficientes LSF

Para viabilizar a transmissão dos coeficientes LSF é necessário representar os coeficientes com poucos bits. Intuitivamente, podemos imaginar que quanto maior for o número de bits melhor deverá ser a fidelidade da voz reconstruída pelo receptor da transmissão. Dessa forma, é necessário achar um ponto de equilíbrio entre quantidade de bits transmitida e fidelidade da reconstrução.

A quantização consiste em processar uma grande quantidade de amostras de voz e, para cada coeficiente determinar, baseado na sua distribuição de valores, alguns pontos representantes. Com quatro bits podemos representar dezesseis valores distintos. Se quisermos quantizar uma variável que tenha valores de 0 até 150, podemos determinar que o valor quantizado 0 (quatro bits 0) corresponde ao valor não-quantizado 0. O 1 (três bits 0 e um bit 1) corresponde ao valor 10. O 2 corresponde ao 20, e assim sucessivamente. Ao realizar a quantização, um valor 9 seria aproximado para o seu representante mais próximo, nesse caso, 10, e seria quantizado para o valor 1. O transmissor, ao receber o valor 1 não teria como reconstruir o valor original, 9, mas obteria o aceitável 10.

Para o nosso codificador CELP um grande estudo foi realizado ([1]), processando uma grande seqüência de voz e analisando o impacto da variação do número de bits por coeficiente sobre a qualidade da reprodução. A distribuição utilizada de bits por coeficiente encontrada está resumida na Tabela 2.1.

Tabela 2.1: Bits utilizados na quantização dos coeficientes LSF.

Coeficiente	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10
Bits	4	4	3	3	3	3	3	3	3	3

Os dois primeiros coeficientes LSF, que são os mais significativos, são quantizados com 4 bits (podendo assumir, portanto, 16 valores diferentes), enquanto os demais são quantizados com 3 bits (podendo assumir apenas 8 valores).

Agora, possuímos um conjunto de dez coeficientes LSF quantizados. Esses coeficientes são os que efetivamente serão transmitidos para o receptor.

Como esses valores quantizados são os únicos que o receptor terá para reconstruir o sinal original, é importante que o transmissor realize todos os cálculos que ainda serão feitos utilizando esses valores, e não os originais, que nunca chegarão ao receptor. Dessa forma, sempre que os coeficientes LSF forem mencionados até o fim do capítulo, deve-se entender os valores quantizados dos coeficientes LSF, que serão utilizados nos cálculos no lugar dos valores originais.

2.4.6 Processamento dos sub-blocos

Até esse passo, todo o processamento foi realizado sobre a janela de 160 amostras da voz. Como foi dito, pode-se considerar que os tubos do sistema fonador formam um sistema estacionário durante o período necessário para gerar essas 160 amostras. Entretanto, a excitação gerada pelos pulmões não pode sofrer a mesma aproximação. Para manter a qualidade da transmissão é necessário que a excitação seja modelada em períodos menores, de 5 ms, ou 40 amostras, numa taxa de amostragem de 8 kHz. Assim, a janela inicial de 160 amostras será quebrada em quatro sub-blocos de 40 amostras, e cada um desses sub-blocos passará por uma mesma seqüência de processamento, que gerará um modelo para a excitação.

Para os passos a seguir é assumido que exista um sinal de entrada de 40 amostras, e o conjunto de coeficientes LSF, que foi gerado para a janela completa, de 160 amostras.

Interpolação dos Coeficientes LSF

Para evitar uma transição abrupta dos coeficientes LSF entre cada janela, optou-se por realizar uma interpolação desses coeficientes antes de iniciar o processamento de cada sub-bloco. A interpolação é feita utilizando-se os valores dos coeficientes LSF da janela anterior e os calculados para a janela atual.

No primeiro sub-bloco de uma janela utiliza-se como coeficientes LSF os valores de LSF calculados para a janela anterior. No segundo sub-bloco utiliza-se os valores calculados para a janela anterior multiplicados por $0,75$ somados aos da atual multiplicados por $0,25$. No terceiro sub-bloco soma-se os coeficientes LSF da janela anterior e o da atual, multiplicados por $0,50$. No quarto sub-bloco usa-se o inverso do segundo sub-bloco: multiplica-se os coeficientes da janela anterior por $0,25$ e os da atual por $0,75$.

Chamando-se de w_i^n o n-ésimo coeficiente LSF calculado para a i-ésima janela, podemos expressar a interpolação da seguinte forma:

$$w_{i_b}^n = (1 - 0,25b)w_{i-1}^n + 0,25bw_i \tag{2.11}$$

onde $w_{i_b}^n$ representa o valor interpolado do n-ésimo coeficiente LSF calculado para o b-ésimo sub-bloco da i-ésima janela. Os valores de b variam de 0 a 3.

O objetivo da interpolação é fazer com que no processamento de cada sub-bloco sejam utilizados como coeficientes LSF valores intermediários entre o calculado para a janela anterior e os calculados para a janela atual. Quanto mais próximo o sub-bloco estiver da janela anterior, maior será o seu efeito.

É importante lembrar que os coeficientes LSF utilizados na interpolação são os que já passaram pela quantização. Então o mais correto é dizer que w_i^n corresponde ao valor quantizado do n-ésimo coeficiente LSF calculado para a i-ésima janela. Também é importante ressaltar que não será necessário transmitir os valores interpolados, já que eles podem ser gerados a partir dos valores quantizados. Dessa forma, não é necessário quantizar novamente os coeficientes interpolados. Eles podem ser armazenados localmente, e utilizados no processamento, como números reais.

Obtenção dos Coeficientes LPC para o sub-bloco

Embora os coeficientes LSF sejam mais apropriados à transmissão, já que são menos sensíveis aos efeitos da quantização, o processamento que ainda precisa ser feito utiliza os coeficientes LPC. Tendo calculado os coeficientes LSF que deverão ser utilizados para o sub-bloco atual, podemos aplicar a função inversa que foi utilizada para gerar os coeficientes LSF, e obter os coeficientes LPC referentes a esse bloco.

Esses são os coeficientes que serão utilizados nos próximos passos.

Dicionários de excitação

Uma vez encontrado o filtro digital que melhor modela o trato vocal para um sub-bloco de 5 ms, é necessário determinar qual excitação deve-se passar por esse filtro para que o resultado seja suficientemente próximo ao sinal original. Essa excitação é um sinal de 40 amostras, que o receptor utilizará para reconstruir o sinal original.

O método utilizado para obter essa excitação utiliza dicionários de excitações, e é conhecido como Análise por Síntese. Esses dicionários contêm um conjunto de seqüências de sinais, cada uma delas com 40 amostras, que são testadas, passando-as pelo filtro LPC, e comparando o resultado com o sinal original. O sinal de excitação que possuir o menor erro médio quadrático em relação ao sinal original será o escolhido.

Nessa implementação da codificação CELP optou-se por utilizar dois dicionários. O primeiro deles, o dicionário fixo, contém um determinado número de sinais, cada um com 40 amostras, que foram gerados antes da transmissão, e são conhecidos tanto pelo transmissor quanto pelo receptor. O método utilizado para gerar esse dicionário pode ser encontrado em [3].

O outro dicionário, o adaptativo, possui algumas peculiaridades. Ele não possui uma conjunto de sinais de excitação isolados, como o dicionário fixo. O dicionário adaptativo é uma grande seqüência de amostras, na qual os possíveis sinais de excitação estão sobrepostos. O primeiro possível sinal de excitação é composto pelas 40 primeiras amostras do dicionário adaptativo. O segundo possível sinal de excitação, que também tem 40 amostras, se inicia a partir de alguma amostra do primeiro sinal, como mostrado na figura 2.4.

Além disso, o dicionário adaptativo é modificado ao final do processamento de cada sub-bloco. A forma como essa atualização é feita, e a sua justificativa, serão apresentadas

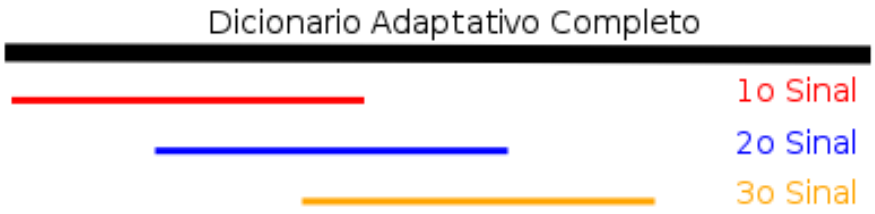

Figura 2.4: Demonstração da sobreposição existente entre os sinais do dicionário adaptativo.

posteriormente.

Os dois dicionários de excitações serão utilizados nos próximos passos. Para esses passos é necessário assumir que, no início do processamento de cada sub-bloco, os dicionários do transmissor e do receptor sejam idênticos. Isso é trivial para o dicionário fixo, e será garantido também para o adaptativo, pelo algoritmo utilizado para atualizá-lo.

Busca no Dicionário Adaptativo

O dicionário adaptativo contém AC_SIZE possíveis excitações sobrepostas, onde uma excitação começa AC_STEP amostras após a excitação anterior. A primeira possível excitação é formada pelas amostras de $ac[0]$ até $ac[39]$. A segunda possível excitação é composta pelas amostras de $ac[AC\_STEP]$ até $ac[39 + AC\_STEP]$, e assim sucessivamente, onde $ac[i]$ representa a i-ésima amostra do dicionário adaptavivo.

O tamanho total de amostras do dicionário adaptativo é, portanto:

$$40 + AC\_STEP(AC\_SIZE - 1)$$

Para manter a coerência com a implementação, a primeira amostra da seqüência ac terá o índice 0.

Também será utilizada a seqüência AC, onde $AC[i]$ representa a i-ésima possível excitação do dicionário adaptativo. É importante observar que cada elemento da seqüência AC corresponde a um conjunto de 40 amostras. As seqüências AC e ac relacionam-se da seguinte forma:

$$AC[i] = \begin{bmatrix} ac\,[AC\_STEP \times (i-1)] \\ ac\,[1 + AC\_STEP \times (i-1)] \\ ac\,[2 + AC\_STEP \times (i-1)] \\ \vdots \\ ac\,[39 + AC\_STEP \times (i-1)] \end{bmatrix}^t \tag{2.12}$$

A busca no dicionário adaptavivo é feita usando as diversas excitações $AC[i]$ como entrada do filtro LPC determinado para o atual sub-bloco. Calcula-se a potência do resultado obtido e a potência do sinal original, multiplicando-se a excitação pela razão entre as duas potências, chamada de ganho do dicionário adaptativo, de forma a gerar uma excitação $AC'[i]$, que, quando filtrada, gera um sinal de mesma potência do sinal original.

Esse sinal, gerado pela filtragem da excitação $AC'[i]$ pelo filtro LPC do sub-bloco atual, é comparado ao sinal original, extraindo-se o erro médio quadrático entre os dois sinais. O valor do erro e o ganho aplicado à excitação são armazenados em uma tabela temporária. O mesmo processo é aplicado às outras AC_SIZE possíveis excitações, $AC[0]$ até $AC[AC\_SIZE - 1]$, gerando um valor de erro e um ganho para cada uma das excitações.

Tendo processado todas as excitações do dicionário adapativo, escolhe-se como a melhor excitação para esse sub-bloco aquela que tiver gerado o menor valor de erro.

O valor temporário para a excitação que será utilizada pelo receptor é, portanto, a excitação selecionada, que será de $AC[i_b]$, multiplicada pelo ganho associado a essa excitação, que será chamado de $G_a[i_b]$. O índice i_b corresponde ao índice que gerou o menor erro no processo de análise por síntese.

É importante observar que não é necessário transmitir as 40 amostras que compõem a excitação $AC[i_b]$. Como o receptor possui o mesmo dicionário adaptativo que o transmissor, é suficiente transmitir o índice i_b. O ganho, por ser um número fracionário, precisa ser quantizado. Uma tabela para a quantização do ganho também é utilizada, e ao invés de transmitir-se o valor $G_a[i_b]$, transmite-se um índice da tabela de quantização, que será chamado de I_{Ga}, que pode ser transformado pelo receptor em G_{a_Q}, utilizando a mesma tabela de quantização. A quantização precisa ser feita de forma que o valor quantizado do ganho, G_{a_Q} seja suficientemente próximo do valor original, $G_a[i_b]$.

Conforme explicado na quantização de coeficientes LSF, o transmissor precisa trabalhar com os mesmos valores que o receptor terá a sua disposição e, dessa forma, deverá utilizar o valor quantizado do ganho, G_{a_Q}. Portanto, para o transmissor, a parte da excitação referente ao dicionário adaptativo pode ser representada por $G_{a_Q} \times AC[i_b]$.

O sinal-alvo, $s_w[n]$ é "atualizado", subtraindo-se desse valor o resultado da filtragem da excitação referente ao dicionário adaptativo, para gerar o novo sinal alvo:

$$s_w[n] = s_{w_{antigo}}[n] - G_{a_Q} \times AC[i_b][n] \tag{2.13}$$

onde $AC[i_b][n]$ representa a n-ésima amostra do sinal de excitação selecionado do dicionário adaptativo.

O sinal $s_w[n]$ pode ser interpretado como a parte do sinal original que não foi possível reproduzir a partir do dicionário adaptavivo. Para reproduzir esse "resíduo" do sinal original, utilizamos o dicionário fixo.

Busca no Dicionário Fixo

O dicionário fixo apresenta duas diferenças em relação ao adaptativo. A primeira diferença diz respeito à atualização que o dicionário adaptativo sofre no fim do processamento, o que não acontece com o dicionário fixo. A segunda é que no dicionário fixo não existe sobreposição entre os sinais de excitação.

O dicionário fixo possui FC_SIZE possíveis sinais de excitação, cada um com 40 amostras, não existindo sobreposição entre os sinais. O tamanho total do dicionário fixo é, portanto, $40 * FC\_SIZE$.

Repete-se o mesmo processo aplicado ao dicionário adaptativo. Cada um dos sinais de excitação é filtrado, calcula-se o ganho necessário para igualar a potência do resultado à potência do sinal alvo, atualizado após a busca no dicionário fixo.

Após processar todas as possíveis excitações do dicionário fixo, aquela que gerou o menor erro é escolhida. O ganho referente a esta excitação é quantizado, e o índice da tabela de quantização do ganho referente ao dicionário fixo é armazenado.

A parte da excitação referente ao dicionário fixo pode ser representada por $G_{f_Q} \times FC[i_{b_f}]$, onde G_{f_Q} representa o ganho quantizado referente ao dicionário fixo e $FC[i_{b_f}]$ representa o sinal selecionado do dicionário fixo, de índice i_{b_f}.

É necessário enviar para o receptor o índice referente ao sinal selecionado do dicionário fixo, i_{b_f}, e o índice da tabela de quantização do ganho do dicionário fixo, I_{Gf}.

Cálculo da Excitação Completa

Como visto, o sinal completo de excitação é composto por duas partes, uma referente ao dicionário adaptativo e outra ao dicionário fixo, e pode ser representado por:

$$G_{a_Q} \times AC[i_b] + G_{f_Q} \times FC[i_{b_f}] \tag{2.14}$$

O receptor pode obter G_{a_Q} e G_{f_Q} a partir de I_{Gf}, e I_{Ga}, respectivamente. Dessa forma, para que o receptor possa reconstruir a mensagem, é suficiente transmitir, para cada sub-bloco: i_{Gf} (índice na tabela de quantização do ganho dicionário fixo), i_{Ga} (índice na tabela de quantização do ganho do dicionário adaptativo, i_b (índice referente ao melhor sinal de excitação do dicionário adaptativo) e i_{b_f} (índice referente ao melhor sinal de excitação do dicionário fixo).

Atualização do Dicionário Adaptativo

Com o dicionário adaptativo busca-se incluir dinamicamente neste dicionário as excitações referentes aos últimos sub-blocos, já que é provável que a excitação referente a um sub-bloco seja semelhante às excitações observadas nos sub-blocos adjacentes. A adaptação é feita após o cálculo da excitação completa. As 40 primeiras amostras do dicionário adaptativo são descartadas, e as 40 amostras da excitação completa são adicionadas ao final do dicionário.

Ao processar o primeiro sub-bloco tanto o transmissor quanto o receptor possuem o dicionário adaptativo com o valor inicial, que é igual nos dois lados. Após processar o primeiro sub-bloco, o transmissor atualiza o dicionário adaptativo, utilizando a excitação completa que foi gerada. Quando os dados referentes aos índices dos sinais e dos ganhos são transmitidos, o receptor pode obter a mesma excitação completa, e, portanto, consegue atualizar a dicionário adaptativo com os mesmos valores utilizados pelo transmissor. Dessa forma, os dois lados conseguem atualizar o dicionário da mesma forma. Isso permite que, no início do processamento do próximo sub-bloco, os dicionários do transmissor e do receptor sejam idênticos, o que é fundamental para que a reconstrução do sinal ocorra

com sucesso.

A atualização do dicionário adaptativo encerra o processamento do sub-bloco. Os quatro sub-blocos de uma janela são processados de acordo com o procedimento detalhado nesse item.

2.4.7 Transmissão de Dados

A transmissão dos dados codificados será discutida no próximo capítulo. Por enquanto é suficiente definir que dados precisam ser transmitidos.

Para cada janela são gerados 10 coeficientes LSF. Cada janela se divide em quatro sub-blocos e para cada sub-bloco é gerado um índice e um ganho para cada um dos dois dicionários. Ao total, para cada janela, são gerados dez coeficientes LSF, quatro índices do dicionário fixo, quatro índices do dicionário adaptativo, quatro ganhos do dicionário fixo e quatro ganhos do dicionário adaptativo. Esses são os valores que deverão ser transmitidos.

É importante lembrar que os coeficientes LSF e os ganhos não são transmitidos como números de ponto flutuante, e sim como índices referentes às devidas tabela de quantização.

2.5 Decodificação

Ao receptor da mensagem são entregues, para cada janela, 10 coeficientes LSF, 4 índices do dicionário adaptativo, 4 do fixo, 4 índices do ganho do dicionário adaptativo e 4 do fixo.

O procedimento utilizado para transformar essas informações em um sinal de voz pode ser resumido por:

- Realizar a interpolação dos coeficientes LSF para o primeiro sub-bloco, utilizando-se os coeficientes LSF da janela atual e da anterior
- Transformar os coeficientes LSF em coeficientes LPC
- Obter a excitação referente ao dicionário adaptativo, multiplicando-se as amostras pelo ganho

- Obter a excitação referente ao dicionário fixo, multiplicando-se as amostras pelo ganho
- Somar as duas excitações para obter a excitação total
- Utilizar a excitação total como entrada no filtro formado pelos coeficientes LPC referentes a esse sub-bloco. A saída do filtro será formada por 40 amostras de voz.
- Repetir o processo para os três sub-blocos restantes, armazenando a saída do filtro

Esse procedimento, aplicado aos quatro sub-blocos, irá gerar um sinal de voz de 160 amostras que deverá ser suficientemente próximo ao sinal original.

2.6 Conclusão

Nesse capítulo foi apresentada uma versão simplificada do funcionamento de um codificador CELP. Os principais passos da codificação e da decodificação foram detalhados. Alguns passos, como o filtro perceptual, ou a remoção da resposta a entrada zero, que não foram apresentados, podem ser encontrados em [1] e [3]. Esses passos são melhorias aplicadas ao codificador, que não são necessárias para a compreensão dos próximos capítulos.

Ao final da codificação obtemos um conjunto de 26 valores que precisam ser transmitidos para o receptor, para que esse possa obter o sinal de voz. No próximo capítulo será apresentada a forma como esses dados são continuamente transmitidos pela Internet.

Capítulo 3

Transmissão de Voz sobre IP

3.1 Introdução

Nesse capítulo serão apresentados os conceitos básicos sobre transmissão de dados pela Internet. No seção 3.2 será descrito o protocolo IP. Os protocolos UDP e TCP, que são os efetivamente utilizados para transmissão de dados, serão apresentados, respectivamente, nas subseções 3.2.1 e 3.2.2. Na seção 3.2.3 eles serão comparados, apresentando as vantagens e as deficiências de cada um deles. As vantagens e desvantagens serão detalhadas na seção 3.5, onde a estratégia de transmissão de dados será vista.

Na seção 3.3 o codificador CELP será retomado. Os tamanhos dos dicionários de excitações e das tabelas de quantização serão apresentados nas subseções 3.3.1 e 3.3.2, respectivamente. Com essas informações, o *bitstream* completo será apresentado, na seção 3.4.

Uma utilização alternativa do sistema, como interface entre um computador ligado à Internet e a rede telefônica convencional será apresentada na seção 3.6.

Durante o capítulo diversas referências serão feitas a documentos que definem os protocolos de comunicação da Internet. Além desses documentos, deve-se consultar [11], uma referência sobre todos eles.

3.2 Protocolo IP

Toda transmissão de dados pela Internet é feita por troca de "pacotes". Um computador conectado à Internet envia seguidos pacotes de dados para outro computador, que agrupa os dados contidos nesses pacotes para obter a mensagem enviada pelo transmissor.

Uma estrutura de dados pode conter outros tipos de estruturas, processo conhecido como encapsulamento. O tipo mais básico de pacote transmitido pela Internet é o pacote IP, de *Internet Protocol*. Pode-se dizer que os pacotes IP formam a plataforma básica de transmissão de dados pela Internet.

O formato de dados dos diversos protocolos da Internet são definidos por um órgão aberto, formado por voluntários, chamado de *Internet Engineering Task Force* (IETF). A IETF [14] é formada por diversos grupos de trabalho que divulgam documentos chamados de *Request for Comments* (RFCs), que são os padrões *de-facto* utilizados pelas aplicações criadas para a Internet. Esses padrões são fundamentais para que aplicações de diferentes origens consigam operar conjuntamente.

Os pacotes IP são definidos por um desses documentos, a RFC 791 [15], como mostrado pela figura 3.1.

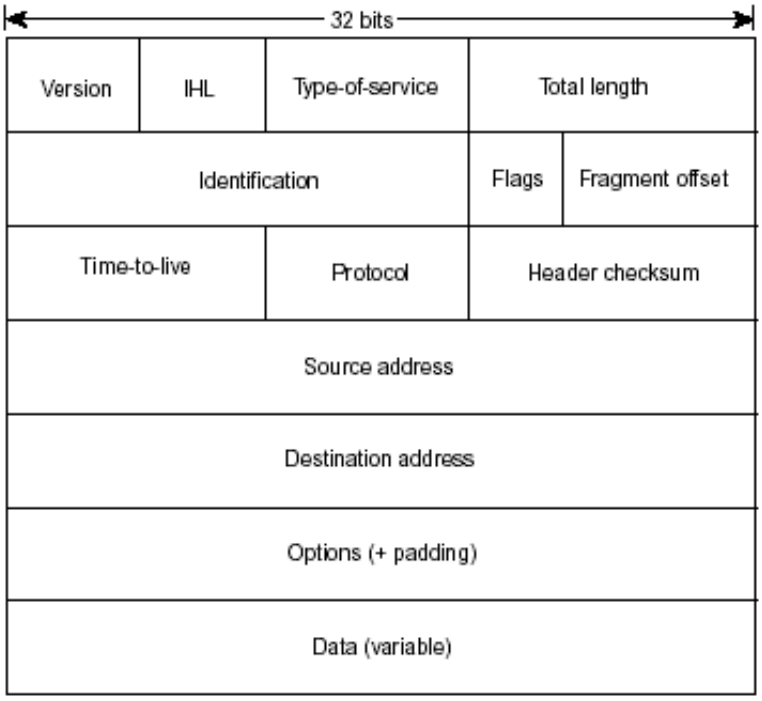

Figura 3.1: Formato de um pacote IP, conforme [15].

Um pacote IP contém, entre outros dados, três informações fundamentais: endereço de

origem, endereço de destino, e dados. Os endereços de origem e de destino são utilizados pelos roteadores que compõem a infra-estrutura da Internet para levar os pacotes de um lado para o outro. Os dados, de tamanho variável, podem ser preenchidos pelo "cliente" do protocolo IP com qualquer tipo de informação.

Entretanto, as aplicações em geral não utilizam os protocolos IP diretamente para efetuar a transmissão de dados. Por ser a plataforma básica da Internet, esse protocolo precisa ser tão simples quanto possível. Ele não possui, por exemplo, o conceito de conexão: cada pacote é isolado dos outros pacotes. Também não possui nenhuma garantia de confiabilidade: o transmissor não percebe o erro caso um pacote não seja entregue para o destinatário. Além disso, o protocolo IP não oferece nenhuma forma de multiplexar conexões entre dois computadores. Não existe como determinar o destinatário de um pacote caso existam duas aplicações em um computador falando com duas aplicações em um outro computador. Isso ocorre pois o pacote IP só define o endereço do emissor e o endereço do receptor, sem oferecer nenhuma forma de determinar para quem, dentro do receptor, deve ser entregue a mensagem.

A opção feita pelos grupos padronizadores da Internet, como a IETF, foi deixar esse tipo de inteligência para outros protocolos que seriam encapsulados dentro de um pacote IP.

Dessa forma, esses outros protocolos, como o TCP e UDP, são os "clientes" dos pacotes IP, e as aplicações finais são "clientes" dos segmentos TCP ou UDP. Esses pacotes, mais complexos, são enviados dentro do campo de dados do pacote IP, que por sua vez são enviados pela Internet.

Os procolos TCP e UDP não são os únicos usados para adicionar inteligência aos pacotes IP, mas são os principais, e por isso serão estudados. Outros protocolos podem ser encontrados em [11].

3.2.1 UDP

O *User Datagram Protocol* (UDP) é definido pela RFC 768 [16]. Ele contém uma característica fundamental para a troca de dados pela Internet: o conceito de portas.

Os pacotes IP, como vistos no item 3.2, possuem apenas o endereço de origem e o de destino. Com eles não é possível distinguir para qual aplicação dentro de um mesmo computador uma mensagem se dirige. O conceito de portas resolve esse problema. Uma

aplicação que deseje receber dados por UDP precisa especificar uma porta para a qual esses dados devem ser enviados. A porta é um número inteiro, de dois bytes, podendo variar entre 1 e 65535. Várias aplicações em um mesmo computador podem receber dados por UDP, desde que "escutem" em portas distintas.

Pode-se imaginar a figura de um "gerenciador" de protocolos UDP, responsável por ler a porta de destino de todos os pacotes recebidos e direcioná-los para a aplicação que estiver ouvindo naquela porta.

O cabeçalho de um pacote UDP possui a forma mostrada na figura 3.2. É importante lembrar que esse cabeçalho será inserido no início do campo de dados do pacote IP 3.1.

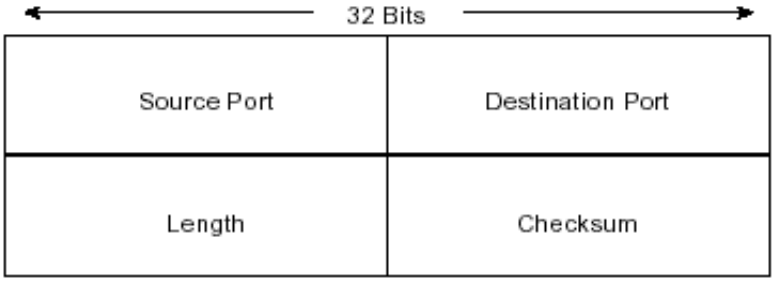

Figura 3.2: Formato de um pacote UDP, conforme [16].

Além das portas de origem e de destino, o cabeçalho do pacote UDP possui outra ferramenta importante, o *checksum*. Os dois bytes de *checksum* são preenchidos com o resultado de uma função aplicada aos dados transportados, de forma que o receptor possa aplicar a mesma função ao receber a mensagem, e confirmar a integridade dos dados transmitidos. Caso algum bit tenha sido corrompido o resultado da função aplicada pelo receptor diferirá do *checksum*, o que apontará a corrupção, e fará com que o pacote seja descartado.

Embora apresente essas duas melhorias, o protocolo UDP ainda apresenta problemas. Assim como o pacote IP, ele é orientado a transação (ou a pacote), e não a conexão. Também como o IP, ele não possui nenhum tipo de confirmação de recebimento: o transmissor não é toma conhecimento sobre o não recebimento de um pacote previamente enviado. Para resolver esses e outros problemas, um protocolo ainda mais complexo, o TCP, foi desenvolvido.

3.2.2 TCP

No protocolo TCP, o formato dos pacotes é definido pela RFC 793 [17]. Tal qual os pacotes UDP, os pacotes TCP também devem ser encapsulados dentro de um pacote IP. Entretanto, o protocolo TCP apresenta uma diferença significativa em relação ao UDP: ele é orientado a conexão.

Ser orientado a conexão significa que os pacotes não são interpretados isoladamente. Antes de ser possível realizar troca de dados entre dois computadores via protocolo TCP, um procedimento conhecido como *Three-way handshake* é realizado. Esse procedimento sincroniza os dois lados, permitindo que a ordem dos pacotes seja mantida, ainda que eles possam chegar ao receptor em ordem diferente da que foram enviados.

Além disso, o protocolo TCP determina o envio de um pequeno pacote de confirmação (*acknowledge*) para cada pacote de dados recebido, e também o envio de um pacote de *keep-alive* periódico, para que um lado da comunicação informe ao outro que ele ainda está presente.

Ao final da comunicação, um pacote de finalização é enviado, e confirmado pelo outro lado, o que fecha o canal.

O modelo de operação do TCP é fundamentalmente diferente do observado no UDP. O TCP é orientado a conexão, e o canal criado pelo protocolo TCP é bi-direcional. O formato do cabeçalho TCP, com as adições necessárias para implementar essas novas funcionalidades, pode ser visto na figura 3.3.

Assim como visto no protocolo UDP, os cabeçalhos TCP também são adicionados no início do campo de dados dos pacotes IP.

Além de possuir campos para *checksum* e portas, como visto no protocolo UDP, os pacotes TCP também possuem os campos *Sequence number* e *Acknowledgement* utilizados para sequenciamento e confirmação de recebimento.

3.2.3 Comparação entre os Protocolos da Camada de Transporte

A transmissão de dados pela Internet pode ser representada por um modelo de várias camadas, onde a passagem de uma camada para outra é feita pelo (des-)encapsulamento.

O modelo mais comum é o OSI [18], de sete camadas. Aplicando esse modelo à transmissão de dados pela Internet, o protocolo IP é responsável pela terceita camada,

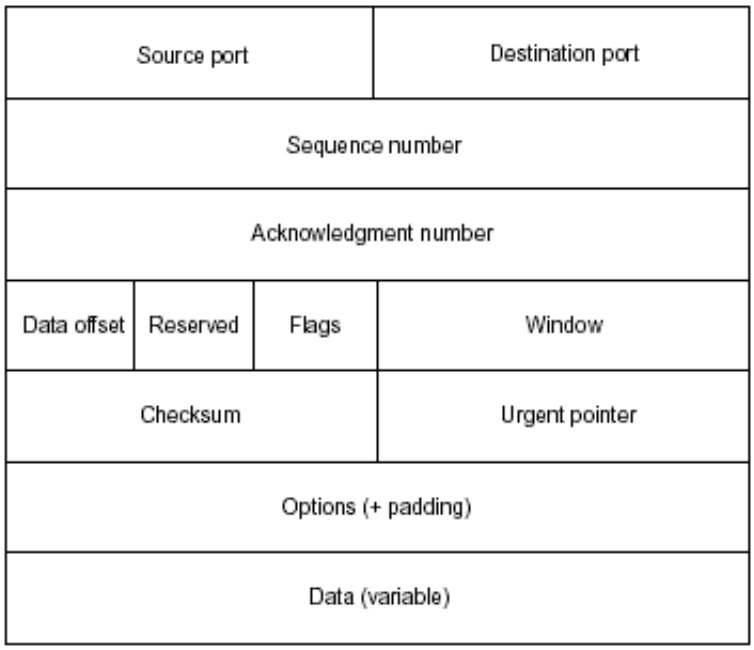

Figura 3.3: Formato de um pacote TCP, conforme [17].

enquanto os protocolos TCP e UDP formam a quarta camada.

Figura 3.4: As sete camadas do modelo OSI.

Como visto, as aplicações não utilizam o protocolo IP diretamente. Quando precisam transmitir dados elas utilizam os protocolos da camada de transporte, o TCP ou o UDP. Ao desenhar uma aplicação para a Internet é importante definir que protocolos serão utilizados.

Embora o protocolo TCP tenha sido apresentado como uma evolução do UDP, por resolver alguns dos problemas apresentados no último, ele não é necessariamente a melhor opção para todos os casos. O protocolo TCP apresenta melhorias óbvias, como a bi-direcionalidade da comunicação, e, principalmente, a confirmação de recebimento.

Essas melhorias possuem um custo. A maior complexidade do protocolo TCP requer

um cabeçalho maior. Conforme mostrado nas figuras 3.2 e 3.3, o cabeçalho de um pacote TCP possui 24 bytes, enquanto os cabeçalhos de um pacote IP possui apenas 8 bytes. Esses cabeçalhos são transmitidos em cada pacote IP que possui outros 24 bytes.

Como a banda para transmissão é limitada, é fácil perceber que os pacotes TCP são menos eficientes que os pacotes UDP. Ao transmitir dados via TCP gasta-se banda com o volume extra de cabeçalho adicionado. Além disso, para cada pacote TCP de dados transmitido, um pacote de confirmação é gerado, o que compromete ainda mais o desempenho.

Não existe uma regra geral para decidir que tipo de protocolo usar em cada situação. Em geral, aplicações onde a confiabilidade é fundamental, como a transferência de arquivos, utilizam TCP, enquanto aplicações onde o desempenho é mais importante, como jogos *on-line* ou transmissão de voz, utilizam UDP.

Para esse projeto optou-se por utilizar uma estratégia mista, com canais UDP e TCP. Essa estratégia será detalhada posteriormente.

3.3 Resultados da Codificação CELP

Como visto no capítulo anterior, cada janela de 160 amostras de voz gera os seguintes dados:

- dez coeficientes LSF
- quatro índices do dicionário adaptativo
- quatro ganhos do dicionário adaptativo
- quatro índices do dicionário fixo
- quatro ganhos do dicionário fixo

Os índices dos dicionários são valores inteiros, indicando uma excitação dentre as candidatas existentes em cada dicionário. O número de bits necessários para cada índice depende diretamente do tamanho do dicionário (medido em número de excitações, e não em amostras).

Um dicionário com, por exemplo, 1024 excitações pode gerar índices entre 0 e 1023, sendo necessários dez bits para armazenar esse valor.

Os coeficientes LSF e os ganhos são números com ponto flutuante, e transmiti-los diretamente, utilizando os padrões ISO, seria desnecessariamente caro. Dessa forma, eles precisam ser quantizados antes da transmissão, como visto no Capítulo 2. Para isso, gera-se uma tabela de quantização para cada um desses valores, e transmite-se um índice dessa tabela de quantização, ao invés do valor propriamente dito. O número de bits para esses índices é determinado pelo tamanho das tabelas de quantização.

O tamanho das tabelas de quantização e dos dicionários determinam, portanto, o tamanho do bloco de dados a serem transmitidos para cada janela de voz processada.

3.3.1 Tamanho dos Dicionários

A codificação CELP depende da existência dos dicionários de excitação para que o receptor possa saber o que deve ser utilizado como entrada do filtro digital que irá gerar o sinal de voz reconstruído. Pode-se deduzir que quanto maiores os dicionários mais fiel ao sinal original será o reconstruído, pois mais opções estarão a disposição do transmissor ao realizar o processo de análise por síntese.

Entretanto, quanto maior forem os dicionários, mais bits serão necessários para representar o índice selecionado, aumentando, assim, o volume de dados a ser transmitido. É fundamental, portanto, definir dicionários que apresentem um compromisso razoável entre o volume de dados gerado e a fidelidade da reconstrução.

Para o sistema descrito neste projeto, depois de diversos testes, optou-se por utilizar os tamanhos apresentados na tabela 3.1.

Tabela 3.1: Tamanhos dos dicionários.

Dicionário	Número de Excitações	Tamanho do índice (bits)
Adaptativo	1024	10
Fixo	256	8

3.3.2 Quantização

Assim como no tamanho dos dicionários, também é necessário utilizar tamanho de tabelas de quantização que apresentem um compromisso razoável entre a fidelidade da reprodução e o volume de tráfego gerado pelos índices das tabelas de quantização.

O número de bits utilizados na quantização dos coeficientes LSF já foi apresentado no Capítulo 2, na figura 2.1.

Para os ganhos dos dicionários, optou-se por utilizar seis bits para cada um. Dessa forma, a tabela de quantização dos ganhos foi construída com 64 possíveis valores. Uma tabela foi construída para os ganhos do dicionário adaptativo e outra para os ganhos do dicionário fixo.

3.4 Formato do *Bitstream*

A estrutura final do *bitstream* utilizado no projeto pode ser vista na tabela 3.4.

Tabela 3.2: Formato final do *bistream*.

	Coeficientes LSF	Índices no Dic. Adaptativo	Índices no Dic. Fixo	Ganhos no Dic. Adaptativo	Ganhos no Dic. Fixo
Número de Bits por Valor:	4 ou 3	10	8	6	6
Quantidade de Valores:	10	4	4	4	4
Número Total de Bits:	32	40	32	24	24

Os valores estão representados no formato em que se encontram dispostos no *bistream* final. É importante lembrar que tanto para os coeficientes LSF quanto para os ganhos dos dicionários o que é transmitido é o índice na tabela de quantização em questão, e não o valor propriamente dito.

Como pode ser visto, para cada janela de dados são gerados 152 bytes.

3.5 Estratégia Adotada

No início do capítulo os dois principais protocolos da camada de transporte da Internet, TCP e UDP, foram apresentados. Mostrou-se que o protocolo TCP é mais adequado para situações onde a confiabilidade da transmissão é mais importante que a necessidade de diminuir o volume de dados trafegado. Também mostrou-se que o protocolo UDP deve ser utilizado para troca de dados nas quais busca-se diminuir ao máximo o volume de dados transmitidos, ainda que isso limite a confiabilidade.

Para o sistema de voz sobre IP apresentado nesse projeto, optou-se pela estratégia de utilizar três canais de comunicação:

- um canal TCP, bi-direcional, para troca de informações de controle.
- dois canais UDP, unidirecionais, para enviar o resultado da codificação de um lado para o outro.

Na atual implementação, o canal TCP é utilizado apenas no ínico da comunicação, para iniciar o processo de transmissão. Posteriormente, a conexão TCP permanece aberta, embora não sejam enviados dados por ela. Todo o tráfego de voz é realizado pelos canais UDP.

Para diminuir o *overhead* gerado pelos cabeçalhos dos protocolos envolvidos, optou-se por enviar dados relativos a cinco janelas em cada pacote UDP. Dessa forma, diminui-se o tamanho relativo utilizado para transmiti-los.

A comunicação ocorre da seguinte forma:

Um dos computadores - o cliente - inicia a conexão, conectando-se na porta TCP 20041 do outro - o servidor. Os dois computadores trocam informação de login e senha, e criam duas *threads*. As *threads* são semelhantes a programas, uma sendo executada de forma independente da outra.

Uma das *threads* é a de recebimento. Essa *thread*, executada nos dois computadores, fica, permanentemente, ouvindo na porta UDP 20042. Ela recebe pacotes UDP, e os divide nas cinco janelas de dados. As janelas são sequencialmente decodificadas, gerando o sinal de voz reconstruído.

A outra *thread* é a de envio, que também é executada permanentemente nos dois computadores. Essa *thread* captura continuamente as informações recebidas pelo microfone do computador onde ela está sendo executada, e codifica as informações, gerando um conjunto de dados. Após cinco janelas ela agrupa todo o bitstream gerado e o envia para a porta UDP 20042 do outro computador.

As duas *threads* são executas em *loop*, simultaneamente, nos dois computadores.

3.6 Conexão com Rede Telefônica

Além de permitir a comunicação entre dois computadores ligados à Internet, o sistema também foi adaptado para fazer ligações entre um computador ligado à Internet e um telefone ligado à rede telefônica convencional, evitando as tarifas inter-urbanas e internacionais.

O modelo adotado é semelhante ao utilizado, posteriormente, por serviços como o *SkypeOut* [19].

Para realizar a ligação, um servidor localizado próximo ao destinatário faz a passagem dos dados da Internet para a linha telefônica e a Internet, e vice-versa, como mostrado na figura 3.5.

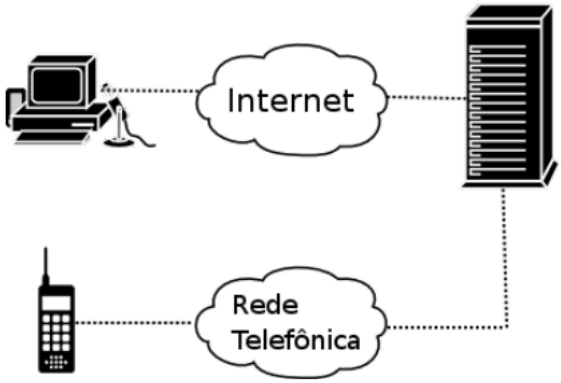

Figura 3.5: Arquitetura da comunicação entre internet e linha telefônica.

A comunicação entre o cliente e servidor é realizada da mesma forma descrita anteriormente, com a utilização de um canal TCP de controle e dois canais UDP de dados. As mesmas duas *threads* apresentadas anteriormente são executadas em paralelo no servidor. Todos os dados recebidos pela *thread* de recebimento são enviados para a linha telefônica. A *thread* de recebimento continua a ler dados do microfone, que, neste caso, está conectado à linha telefônica. Os dados lidos por essa *thread* são enviados para a Internet.

A comunicação entre o servidor e a linha telefônica foi viabilizada pelo circuito mostrado na figura 3.6, apresentado em [20]. A arquitetura completa do sistema, evidenciando as *threads* utilizadas e a interface PC/Telefone pode ser vista na figura 3.7.

O uso do circuito apresentado gerou um problema de eco. Ele captura todos os dados que ele mesmo envia para a linha telefônica, e os interpreta como dados de entrada. Esse eco é codificado e enviado novamente para a Internet, fazendo com que o emissor da mensagem (do lado do computador) ouça o que ele mesmo fala.

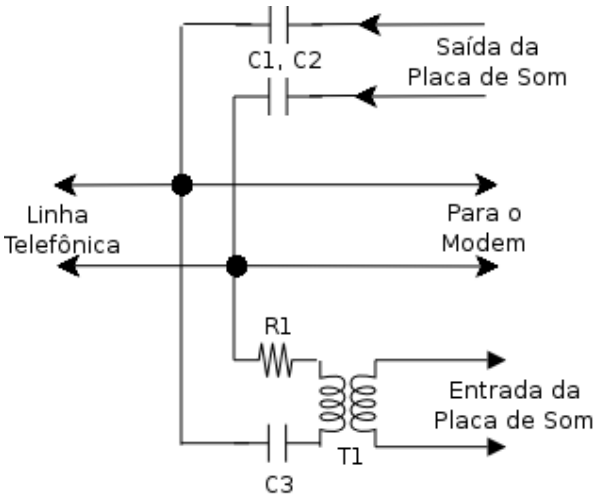

Figura 3.6: Circuito para conexão entre computador e linha telefônica, onde $C1 = C2 = 1\mu F$, $C3 = 0.1\mu F$, $R1 = 470k\Omega$, $T1 = 600\Omega : 600\Omega$.

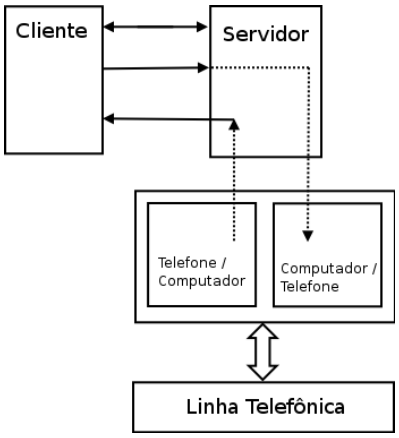

Figura 3.7: Arquitetura completa do sistema.

O problema do eco poderia ter sido resolvido com um circuito integrado específico para ligação PC/Telefone, ou então pela utilização de um Modem específico para essa função. Entretanto, como o objetivo do projeto era acadêmico, optou-se por utilizar algoritmos de cancelamento de eco baseados em filtragem adaptativa. Esses algoritmos serão discutidos no próximo capítulo.

3.7 Conclusão

Nesse capítulo os principais protocolos de troca de dados pela Internet foram apresentados, bem como suas vantagens e desvantagens. Os dados gerados pela codificação CELP foram retomados, e definiu-se a forma como deveriam ser transmitidos pela Internet.

Também foi apresentada uma forma alternativa de utilização do sistema, como ponte

entre um computador ligado à Internet e um telefone ligado à rede telefônica convencional. Como explicado, para implementar essa alternativa foi preciso aplicar algoritmos adaptativos de cancelamento de eco, que serão apresentados no próximo capítulo.

Capítulo 4

Filtros Adaptativos

4.1 Introdução

Como visto no capítulo anterior, é necessário utilizar alguma ferramenta para cancelar o eco gerado na transmissão de voz entre computador e telefone. Nesse capítulo, a ferramenta escolhida, a filtragem adaptativa, será apresentada.

Na seção 4.2 o problema do eco será descrito de forma genérica. Na seção 4.3, o conceito de filtragem adaptativa será apresentado. As suas principais aplicações serão apresentadas na seção 4.4. A aplicação de filtros adaptativos específicos para o cancelamento de eco será discutida na seção 4.5. Algumas medidas dos filtros adaptativos serão discutidas na seção 4.6, e os modelos de filtros adaptativos utilizados no projeto serão descritos na seção 4.7. O sistema completo de VoIP, com a adição do filtro adaptativo, será apresentada na seção 4.8

4.2 O Problema do Eco

No capítulo anterior, o problema de eco existente no sistema de voz sobre IP, causado pelo circuito de interface entre PC e linha telefônica, foi brevemente apresentado. O eco é gerado pelo ruído de interface, que captura os sinais da linha telefônica, inclusive os sinais por ela enviados, e os retorna para a placa de som do computador.

O sinal enviado para a Internet era, portanto, contaminado pelo sinal que havia sido recebido anteriormente, conforme mostra a figura 4.1.

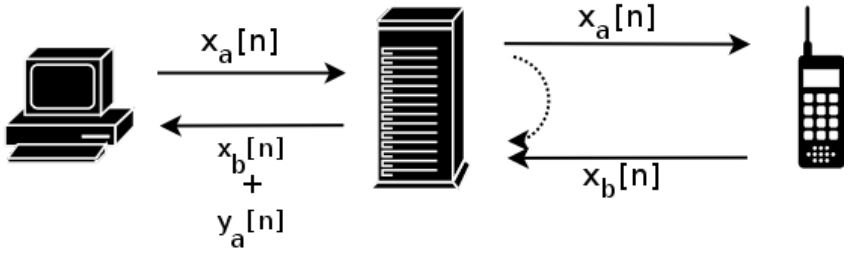

Figura 4.1: Contaminação do sinal telefônico pelo eco.

Na figura 4.1 o sinal $x_a[n]$ represente o sinal que o servidor recebe, vindo da Internet, enquanto $x_b[n]$ é o sinal enviado do telefone para o servidor. Entretanto, o sinal capturado da linha telefônica, pelo circuito, não é composto apenas por $x_b[n]$, mas também por informações referentes ao sinal enviado para a linha telefônica, representados na figura pelo sinal $y_a[n]$. O sinal que será codificado e enviado para a Internet conterá, portanto, o sinal $y_a[n]$, o eco.

Pode-se entender a formação desse sinal $y_a[n]$ como o resultado da passagem do sinal $x_a[n]$ por um filtro digital $H(z)$, conforme mostrado na figura 4.2.

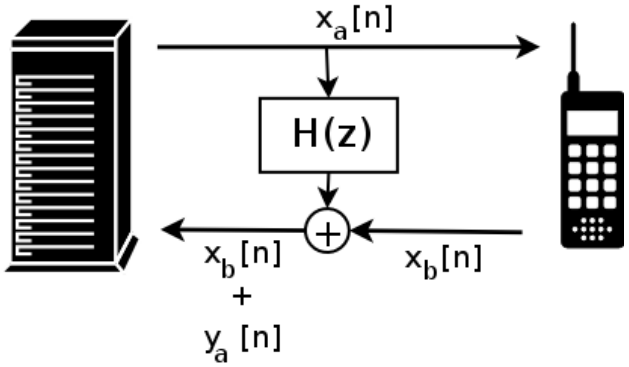

Figura 4.2: Formação do sinal de eco.

O filtro $H(z)$ representa as transformações sofridas pelo sinal $x_a[n]$ ao ser enviado para o circuito, transmitido para a linha telefônica e ao ser novamente capturado pelo circuito.

O mesmo problema de eco pode ser observado em sistemas de viva-voz, onde o som recebido é reproduzido em um alto-falante, enquanto um microfone capta o que está sendo falado. O sinal captado pelo microfone conterá informações referentes ao som reproduzido pelo alto-falante. Nesse caso, o filtro $H(z)$ seria referente a transformação sofrida pelo sinal desde a sua reprodução pelo alto-falante até a sua captação pelo microfone.

4.2.1 Remoção do Eco

Uma forma de retirar o ruído - $y_a[n]$ - do sinal capturado é saber com antecedência o filtro $H(z)$. Dessa forma, seria possível determinar, a partir de $x_a[n]$, qual seria, com exatidão, a componente do ruído gerado pelo eco. Subtraindo-se esse valor do sinal capturado tem-se o sinal $x_b[n]$ sem ruído.

Contudo, determinar o filtro $H(z)$ não é uma tarefa simples. No caso do eco gerado pelo circuito é necessário determinar um modelo para a linha telefônica, que seria diferente para cada linha telefônica onde o circuito fosse testado. Para o eco de um sistema viva-voz, o valor de $H(z)$ também poderia variar cada vez que a posição do microfone fosse alterada.

Para viabilizar a retirada do eco do sinal capturado é necessário utilizar uma ferramenta que consiga, em tempo real, gerar um modelo do sistema causador do eco. Esse modelo também precisa ser atualizado caso as características do gerador de eco sejam alteradas durante a operação.

Filtros adaptativos podem ser utilizados para gerar esse modelo, necessitando de pouco conhecimento prévio sobre como o eco será gerado.

4.3 Filtragem Adaptativa

Os filtros adaptativos podem ser definidos como um filtro que se projeta de forma automática através de um algoritmo, tornando possível operar em um ambiente onde o conhecimento das características dos sinais a processar não é completo [22].

O modo básico de funcionamento de um filtro adaptativo é apresentado na figura 4.3. Um sinal digital, $x[n]$, possivelmente contaminado por ruído, $r[n]$, passa por um filtro linear digital, formado por coeficientes $w[n]$, gerando a resposta $y[n]$. O sinal $d[n]$ representa o sinal de referência, ou seja, o valor esperado para a saída do filtro, caso não houvesse ruído. A comparação entre $y[n]$ e $d[n]$ gera o sinal de erro, $e[n] = d[n] - y[n]$.

O algoritmo recursivo do filtro adaptativo utiliza esse valor de erro, $e[n]$, para ajustar os coeficientes do filtro, $w[n]$, de forma a minimizar o erro, de acordo com algum critério estatístico.

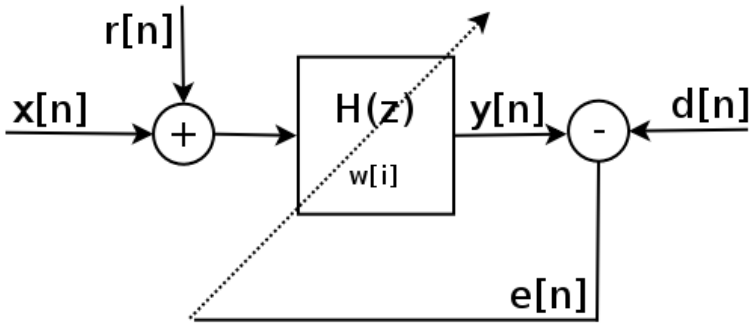

Figura 4.3: Funcionamento básico de um filtro adaptativo.

4.3.1 O Filtro de Wiener

Uma solução para o determinar os coeficientes $w[i]$ que minimizam o erro do sistema mostrado na figura 4.3 é o Filtro de Wiener. Nessa solução parte-se da premissa que o ruído $r[n]$ é branco e não correlacionado com o sinal de entrada, $x[n]$, e de que as informações estatísticas sobre o sinal de entrada sejam conhecidas *a priori*.

Considerando-se o filtro de N-ésima ordem, seus coeficientes podem ser representados por W:

$$W = [w_0 \quad w_1 \quad \dots \quad w_{N-1}]^t \tag{4.1}$$

As N últimas amostras da entrada, no instante n, podem ser representadas por X_n:

$$X_n = [x[n] \quad x[n-1] \quad \dots \quad x[n+1-N]]^t \tag{4.2}$$

A saída do filtro, em um instante n, pode ser determinada pelo produto interno entre os vetores W e X_n:

$$y[n] = X_n^t W = \sum_{i=0}^{N-1} w_i x[n-i] \tag{4.3}$$

Seja R a matriz de auto-correlação do vetor de entrada, $x[n]$:

$$R = E\left[X_n X_n^t\right] \tag{4.4}$$

e P o vetor de correlação cruzada entre o sinal observado, $x[n]$, e a saída do filtro, $y[n]$:

$$P = E\left[d[n]x[n]\right]^t \tag{4.5}$$

O sinal de erro pode ser representado por:

$$e[n] = d[n] - y[n] = d[n] - X_n^t W \tag{4.6}$$

Definindo o erro quadrático médio como critério de minimização, podemos extrair seu valor esperado, conforme [12]:

$$E[e^2[n]] = E[d^2[n]] + W^t R W - 2P^t W \tag{4.7}$$

O valor do erro médio quadrático pode ser plotado como uma hiper-parabolóide, de N dimensões, e possui um único mínimo, obtido quando:

$$\nabla = \frac{\partial E[e^2[n]}{\partial W} = 2RW - 2P = 0 \tag{4.8}$$

A solução para o filtro de Wiener é, portanto, o vetor de coeficientes W^* que minimiza o gradiente da função erro, definida na equação (4.7). Esse vetor ótimo pode ser calculado, a partir da equação (4.8), por:

$$W^* = R^{-1} P \tag{4.9}$$

Com esse conjunto de coeficientes, o filtro de Wiener apresentará resultado ótimo. Pela equação (4.9) pode-se perceber que o vetor W^* só poderá ser calculado caso a matriz de auto-correlação do sinal de entrada seja conhecida. Caso ela não o seja, ou caso o sinal não seja estacionário, a solução exata de Wiener não será possível, e deverá ser aproximada.

Posteriormente serão discutidos algoritmos que substituem o filtro de Wiener, convergindo para a mesma solução ótima de forma iterativa, e para os quais não é necessário conhecer as informações estatísticas dos sinais.

4.4 Aplicações da Filtragem Adaptativa

Para apresentar as formas de utilização de filtros adaptativos, partiremos da premissa de que existe um algoritmo de atualização dos coeficientes do filtro que os leva para o valor ótimo do filtro de Wiener. Os algoritmos para esse fim serão apresentados na seção 4.7.

Nessa seção serão apresentadas quatro arquiteturas em que os filtros adaptativos podem ser utilizados.

4.4.1 Identificação de Sistema

O objetivo dessa arquitetura é utilizar um filtro adaptativo para obter um modelo que reproduza o comportamento de um sistema desconhecido. A arquitetura utilizada pode ser vista na figura 4.4.

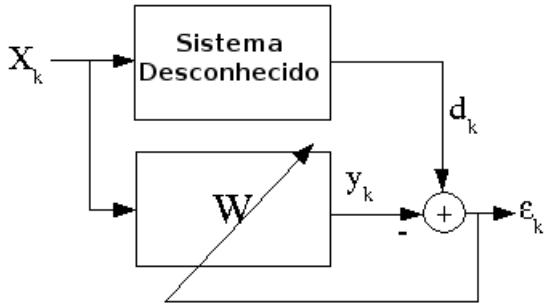

Figura 4.4: Filtro adaptativo para identificação de sistema.

O sinal de entrada é filtrado, ao mesmo tempo, pelo filtro adaptativo e pelo sistema real. A saída do sistema real é utilizada como sinal de referência, e comparada com a saída do filtro adaptativo para determinar o sinal de erro. O algoritmo de adaptação utilizará essa informação para ajustar, progressivamente, o vetor de coeficientes W, de forma a minimizar o valor do erro.

Após o filtro convergir, os coeficientes do filtro adaptativo serão semelhantes aos do sistema desconhecido. É necessário, para isso, que a ordem do filtro adaptativo seja igual ou maior que a do sistema real.

Caso o sistema varie com o tempo, o algoritmo de adaptação, que funcionará continuamente, garantirá que os coeficientes do filtro adaptativo seguirão aqueles do sistema real.

4.4.2 Predição de Sinal

Nessa arquitetura, exibida na figura 4.5, utiliza-se um filtro adaptativo para obter um preditor de um sinal.

Nessa arquitetura, o sinal $x[n]$ sofre um atraso, e as amostras atrasadas passam pelo filtro adaptativo. O resultado da filtragem é comparada com o valor atual do sinal, e o

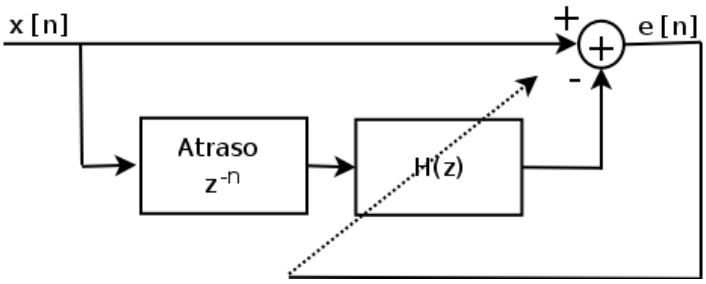

Figura 4.5: Filtro adaptativo para predição de sinal.

algoritmo de adaptação corrige os valores dos coeficientes do filtro, de forma a minimizar o erro.

Quando o filtro tiver chegado aos coeficientes ótimos, ele será capaz de receber um determinado número de amostras de um sinal e predizer quais serão as próximas.

4.4.3 Equalização de Canal

Os filtros adaptativos também podem ser utilizados para equalizar um canal. Nessa arquitetura, mostrada na figura 4.6, busca-se obter o modelo inverso do canal, de forma a garantir que a saída do filtro adaptativo seja igual aos dados originais, cancelando, assim, a degradação causada pelo canal.

Na prática, o filtro adaptativo busca obter o modelo inverso do canal, de forma que um compense o efeito do outro, fazendo com que o conjunto canal/filtro comporte-se apenas como um atrasador ideal.

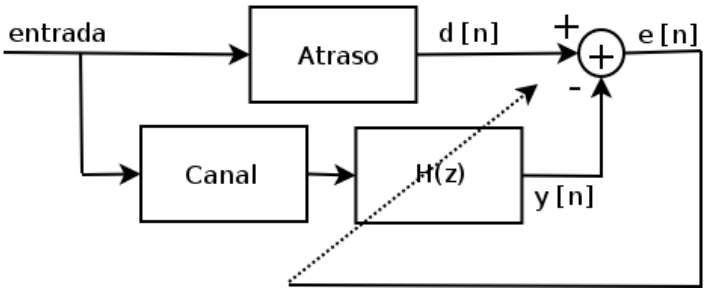

Figura 4.6: Filtro adaptativo para equalização de canal.

O filtro compara os valores originais do sinal $x[n]$, atrasados, com o resultado da filtragem do sinal proveniente do canal. Quando o erro for minimizado, a saída do filtro será igual à entrada do canal, cancelando, assim, os efeitos do canal.

4.4.4 Cancelamento de Interferência

Nessa arquitetura, exibida na figura 4.7, utiliza-se um filtro adaptativo para retirar a interferência de um sinal.

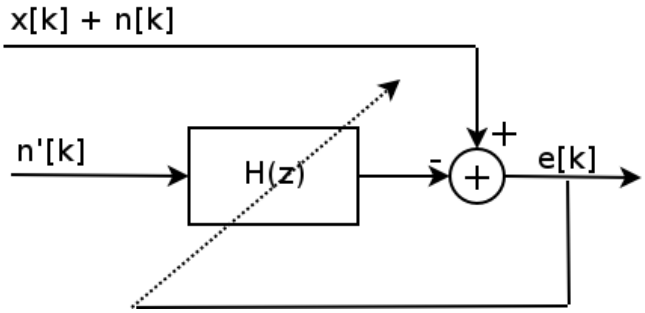

Figura 4.7: Filtro adaptativo para cancelamento de interferência.

Na figura, o sinal contaminado é representado por $x[k] + n[k]$, onde $x[k]$ é o sinal desejado, e $n[k]$ é o ruído, ou a interferência. O sinal utilizado como referência no processo de adaptação é $n'[k]$, que é um sinal correlacionado com o ruído, $n[k]$. Para o correto funcionamento do filtro é fundamental que o sinal $x[k]$ não apresente correlação com o ruído.

Após a convergência, o filtro adaptativo conseguirá transformar o sinal $n'[k]$ em $n[k]$. Nesse momento, o resultado da comparação será exatamente o sinal original. Portanto, o sinal livre da contaminação poderá ser obtido a partir do sinal do erro, $e[k]$.

O filtro adaptativo não conseguirá transformar $n'[k]$ em $x[k] + n[k]$, o que zeraria o erro, pois o sinal original não apresente correlação com o ruído.

4.5 Filtragem Adaptativa para Cancelamento de Eco

A figura 4.8 mostra o surgimento do eco na comunicação entre duas partes, A e B.

O sinal enviado por A, x_a, chega perfeitamente ao B. Entretanto o sinal enviado por B, x_b, é corrompido por x'_a, o eco de x_a. Dessa forma, o sinal efetivamente recebido por A é composto pelo sinal enviado por B somado com um outro sinal, derivado do que o próprio A enviou anteriormente. Os coeficientes do filtro gerador do eco, $H(z)$, responsável por transformar x_a em x'_a, são dependentes da forma como o eco é gerado, e não são conhecidos *a priori*.

Para resolver o problema, utiliza-se um segundo filtro, $H'(z)$, cujos coeficientes são

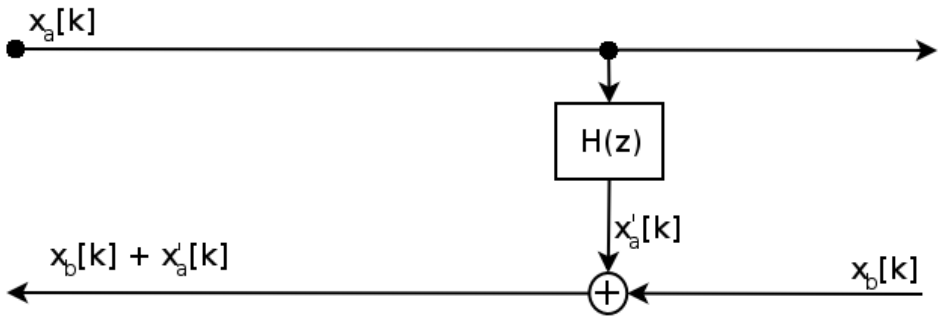

Figura 4.8: Surgimento do eco.

atualizados em tempo real, por algum algoritmo de adaptação, conforme mostrado da figura 4.9.

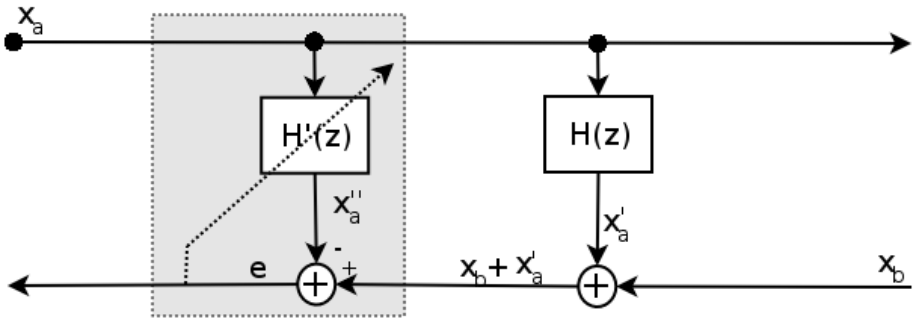

Figura 4.9: Utilização de filtragem adaptativa para cancelar o eco.

Como pode ser observado, a arquitetura de filtragem adaptativa utilizada para o cancelamento de eco é a mesma vista para o cancelamento de interferências, já que o eco, x'_a, nada mais é do que uma interferência.

No item 4.4.4 vimos que essa aplicação de filtragem adaptativa requer a utilização, como entrada do filtro, de um sinal correlacionado com a interferência. Para o cancelamento de eco, o sinal utilizado é o próprio sinal original enviado por A, x_a.

O sinal x_a é filtrado por $H'(z)$, e o resultado, x''_a é comparado com o sinal corrompido, $x_b + x'_a$, e o erro é utilizado para ajustar os coeficientes de $H'(z)$.

O algoritmo de adaptação deverá ajustar os coeficientes de $H'(z)$ de forma a minimizar o erro. Idealmente, o erro não chegará a zero, já que uma parte do sinal corrompido, o x_b de $x_b+x'_a$, não apresenta qualquer correlação com o sinal original, x_a. Após a convergência, o máximo que o filtro conseguirá fazer será gerar, a partir de x_a, uma cópia da parte do sinal corrompido que é relacionado com esse sinal original. A partir desse momento, a saída do filtro adaptativo, x''_a, será idêntica ao componente de eco do sinal corrompido,

x'_a. O sinal de erro será, então:

$$e = (x_b + x'_a) - (x''_a) = (x_b + x'_a) - (x'_a) = x_b \tag{4.10}$$

Ou seja, após a convergência do filtro adaptativo, o sinal de erro desse filtro será igual ao sinal original enviado por B, já que os coeficientes terão se adaptado ao filtro gerador de eco.

4.6 Medidas

Ao projetar um filtro adaptativo, algumas considerações de ordem prática devem ser observadas. Pode-se citar:

- **Tempo de Convergência**

  Define quantas iterações são necessárias para que os coeficientes do filtro cheguem a um valor em qual o erro fica abaixo de um valor pré-determinado.

- **Erro até a Convergência;**

  Descreve a forma de onda do erro medido desde o início da adaptação até o momento da convergência.

- **Erro após a Convergência;**

  Diz respeito a parte do sinal que o filtro adaptativo não consegue mapear. Mesmo após a convergência o sinal gerado pelo filtro ainda pode ser diferente do sinal de referência. Esse erro pode ser gerado por diversos fatores, como, por exemplo, o fato de parte do sinal de referência não apresentar correlação com o sinal de entrada.

- **Complexidade Computacional;**

  É o número de operações básicas, como adições e multiplicações, que devem ser realizadas em cada iteração do filtro. Para dispositivos móveis, como um telefone celular, é necessário reduzir ao máximo o número de operações, de forma a economizar energia e se adequar as limitações do *hardware* disponível.

4.7 Modelos de Filtro Adaptativo

Agora os principais modelos de filtros adaptativos serão apresentados.

4.7.1 Método do Gradiente

O método de busca por gradiente é um algoritmo recursivo, onde valores iniciais são atribuídos aos coeficientes do filtro adaptativo. Esses coeficientes são atualizados em cada iteração, de forma a atingir, após a convergência, a solução do filtro de Wiener.

A busca por gradiente é um método de otimização, onde se procura minimizar o valor de uma função. No caso da filtragem adaptativa, a função a ser minimizada é relacionada ao erro do filtro.

Para utilizar a busca por gradiente, é necessário conhecer, previamente, o gradiente da função que buscamos minimizar. Em cada iteração, o valor dos coeficientes do filtro adaptativo é alterado de forma a se deslocar no sentido contrário ao do gradiente, e por valor proporcional ao módulo do gradiente.

Para ilustrar o algoritmo, pode-se pensar em um filtro com dois coeficientes, onde o erro do filtro adaptativo (o valor previsto menos o valor real - $d - y$) é descrito pela superfície mostrada na figura 4.10.

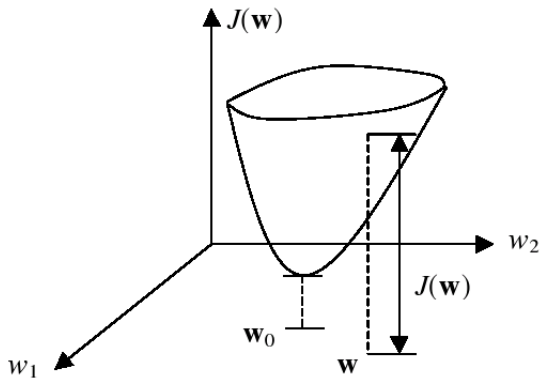

Figura 4.10: Erro de predição de um filtro adaptativo com dois coeficientes.

Na figura, os eixos w_1 e w_2 correspondem aos valores que os coeficientes podem assumir. A função $J(w)$ é o valor médio quadrático do erro associado a um par de coeficientes:

$$J(w) = E|e_{w_1,w_2}(n)|^2 \tag{4.11}$$

O ponto w_0 corresponde ao par ótimo de coeficientes para o filtro adaptativo, e é a mesma solução obtida no filtro de Wiener. A mesma função erro pode ser representada em um gráfico bi-dimensional, como visto na figura 4.11.

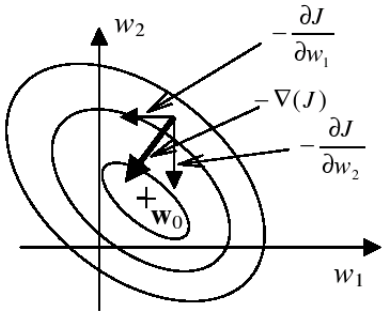

Figura 4.11: Erro de predição de um filtro adaptivo com dois coeficientes, em duas dimensões.

No gráfico em duas dimensões, cada elipse representa o conjunto de pares de coeficientes que possuem um mesmo valor para o erro médio quadrático. Para cada par (w_1, w_2), o gradiente $\nabla(J)$ da função erro é um vetor que aponta "para fora" da elipse. O vetor $-\nabla(J)$ aponta, portanto, para o interior da elipse, embora não necessariamente para o ponto w_0.

Em cada iteração, o vetor $-\nabla(J)$ é decomposto nos seus dois componentes, representados por $-\frac{\partial J}{\partial w_1}$ e $-\frac{\partial J}{\partial w_2}$. Esses componentes são adicionados ao coeficiente correspondente, gerando o conjunto de coeficientes que será utilizado na próxima iteração. Ou seja:

$$w_k(n+1) = w_k(n) - \mu \frac{\partial J}{\partial w_k} \tag{4.12}$$

Ou seja, o valor do k-ésimo coeficiente na instante $(n+1)$ é obtido subtraindo-se do valor do mesmo coeficiente no instante n a derivada parcial da função erro em relação àquele coeficiente multiplicado por alguma constante μ.

Como os coeficientes serão atualizados por um vetor que aponta para dentro da elipse, esses coeficientes atingirão w_0, após um determinado número de iterações, conforme visto na figura 4.12.

O valor μ, chamado de taxa ou coeficiente de adaptação, define a velocidade com que o ponto (w_1, w_2) se aproxima de w_0. Quanto maior for μ, maior será a velocidade. Entretanto, valores muito grandes poderão fazer com que $\mu\nabla(J)$ seja grande o suficiente

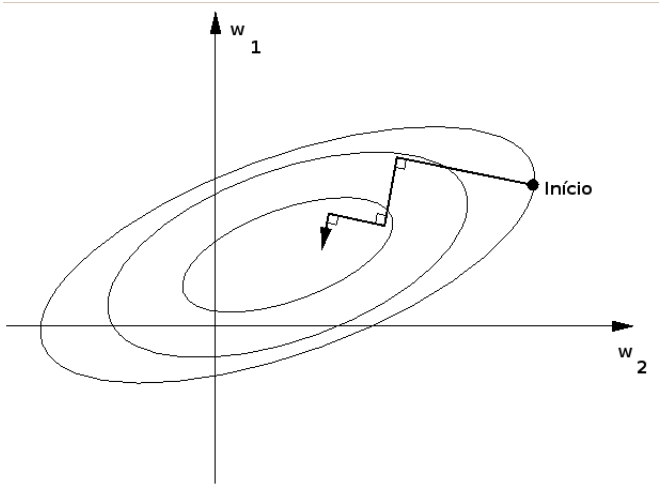

Figura 4.12: Alteração nos coeficientes, durante o processo de convergência.

para levar o ponto (w_1, w_2) para uma região ainda mais distante de w_0 do que o observado antes da atualização, causando oscilação instável. Para que isso não aconteça, como pode ser visto em [12], é necessário que:

$$0 < \mu < \frac{2}{y_{max}} \tag{4.13}$$

onde y_{max} é o maior autovalor da matriz de auto-correlação do sinal de entrada.

4.7.2 LMS

O algoritmo de adaptação por gradiente é simples e eficiente, mas depende do conhecimento prévio do gradiente da função de erro, o que, em geral, não está disponível.

O algoritmo conhecido como LMS (*Least Mean Squares*) utiliza uma aproximação para a função gradiente, da forma [12]:

$$\hat{\nabla} J = -2x(n)e(n) \tag{4.14}$$

A adaptação, realizada em cada iteração, pode ser representada por:

$$w_k(n+1) = w_k(n) + 2\mu e(n)x_k(n) \tag{4.15}$$

O algoritmo LMS pode ser resumido por:

1. Inicializar os coeficientes do filtro adaptativo com um valores aleatórios, ou zero;

2. Calcular o erro de uma iteração: $e(n) = d(n) - y(n) = d(n) - \mathbf{w}^t(n)x(n)$;

3. Atualizar os coeficientes: $\mathbf{w}(n+1) = \mathbf{w}(n) + 2\mu e(n)\mathbf{x}(n)$;

4. Voltar para 2.

O algoritmo LMS pode ser considerado da família dos algoritmos adaptativos de busca por gradiente, pois ele trabalha com uma aproximação do que foi visto no item 4.7.1.

4.7.3 NLMS

O NLMS, ou *Normalized LMS*, escala a constante μ utilizada no LMS de forma a adequá-la à energia do sinal de entrada.

No NLMS, o valor de μ efetivamente utilizado é:

$$\mu_{NLMS} = \frac{\alpha}{\mathbf{x}_k^t \mathbf{x}_k + \gamma} \tag{4.16}$$

onde $0 < \alpha < 2$ e γ é uma constante positiva pequena, para evitar uma divisão por zero.

O algoritmo de adaptação NLMS em geral apresenta tempo de convergência menor que o LMS, por ser mais fácil se escolher um valor adequado de μ, e por isso é constantemente utilizado na prática.

Para o NLMS, a atualização do vetor de coeficientes é da forma:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \frac{\alpha}{\mathbf{x}_k^t \mathbf{x}_k + \gamma} e_k \mathbf{x}_k \tag{4.17}$$

Em geral, usa-se $\alpha = 0,5$, que foi a opção utilizada neste trabalho.

4.7.4 SDLMS

O SDLMS, ou *Sign-data LMS*, é uma adaptação do algoritmo LMS para ambientes onde a complexidade computacional seja determinante. Ao invés de utilizar os valores de entrada para calcular a aproximação do gradiente, este algoritmo utiliza apenas o sinal do valor de entrada.

O processo de atualização é feito conforme mostrado na equação 4.18.

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mu e_k sgn\left[\mathbf{x}_k\right] \tag{4.18}$$

onde,

$$sgn\left[\mathbf{x}_k\right] = \begin{cases} 1, & \mathbf{x}_k > 0 \\ 0, & \mathbf{x}_k = 0 \\ -1, & \mathbf{x}_k < 0 \end{cases} \tag{4.19}$$

4.7.5 SELMS

O SELMS, ou *Sign-error LMS*, também é uma adaptação do algoritmo LMS para ambientes onde a complexidade computacional seja determinante. Ele é semelhante ao SDLMS. Entretanto, no SELMS, o valor substituído pelo seu sinal é o erro.

O processo de atualização é feito conforme mostrado na equação 4.20.

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mu sgn\left[e_k\right]\mathbf{x}_k \tag{4.20}$$

onde,

$$sgn\left[\mathbf{x}_k\right] = \begin{cases} 1, & \mathbf{x}_k > 0 \\ 0, & \mathbf{x}_k = 0 \\ -1, & \mathbf{x}_k < 0 \end{cases} \tag{4.21}$$

4.7.6 SSLMS

O SSLMS, ou *Sign-sign LMS*, é a terceira adaptação do algoritmo LMS para ambientes onde a complexidade computacional seja determinante. Nesse algoritmo, tanto o valor do erro quanto os valores do sinal são substituídos pelo seus sinais. Essa substituição reduz significativamente a complexidade computacional, embora tenha impacto negativo em outros indicadores, como tempo de convergência.

O processo de atualização é feito conforme mostrado na equação 4.22.

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mu sgn\left[e_k\right] sgn\left[\mathbf{x}_k\right] \tag{4.22}$$

onde,

$$sgn\left[\mathbf{x}_k\right] = \begin{cases} 1, & \mathbf{x}_k > 0 \\ 0, & \mathbf{x}_k = 0 \\ -1, & \mathbf{x}_k < 0 \end{cases} \tag{4.23}$$

O algoritmo SSLMS é parte do padrão internacional CCITT para a telefonia com ADPCM de 32 kb/s.

4.7.7 BNDR

O BNDR LMS (*Binormalized Data-Reusing LMS*), apresentado em [24], é outro algoritmo adaptativo baseado no LMS. Mostrou-se que ele converge mais rapidamente do que outros algoritmos da família LMS quando o sinal de entrada possui alta correlação entre suas amostras.

Esse algoritmo utiliza uma constante, ϵ, com baixo valor. Em cada iteração, dois vetores são montados, utilizando os valores anteriores de entrada:

$$\mathbf{x}_1 = \mathbf{x}(k); \mathbf{x}_2 = \mathbf{x}(k-1)$$

Os valores de referência da iteração atual e da anterior são utilizados:

$$d_1 = d(k); d_2 = d(k-1)$$

Os seguintes produtos escalares de vetores são calculados:

$$a = \mathbf{x}_1^t \mathbf{x}_2$$

$$b = \mathbf{x}_1^t \mathbf{x}_1$$

$$c = \mathbf{x}_2^t \mathbf{x}_2$$

$$d = \mathbf{x}_1^t \mathbf{w}(k)$$

$$e = \mathbf{x}_2^t \mathbf{w}(k)$$

Calcula-se:

$$den = bc - a^2 + \epsilon$$

$$A = (d_1 c + ea - dc - d_2 a)/den$$

$$B = (d_2 b + da - eb - d_1 a)/den$$

Finalmente, o vetor de coeficientes é atualizado:

$$\mathbf{w}(k+1) = \mathbf{w}(k) + A\mathbf{x}_1 + B\mathbf{x}_2$$

4.7.8 RLS

Os algoritmos de adaptação estudados até esse ponto são baseados no método de busca por gradiente. O algoritmo RLS (*Recursive Least Squares*, [12]), é baseado no Filtro de Kalman. Em geral, esse filtro possui tempos de convergência sensivelmente menores do que os filtros LMS, embora a sua complexidade computacional seja superior. Para aplicações onde a complexidade computacional não seja um fator limitante, os filtros do tipo RLS são recomendados.

A idéia por trás do algoritmo RLS é minimizar a soma ponderada da função de erro. Para isso, um coeficiente de esquecimento, λ, é definido. Esse coeficiente determina a medida com que a importância dos valores antigos de erro diminui.

O algoritmo RLS é inicializado com um vetor de coeficientes, todos com valor zero, e com uma matriz quadrada $\mathbf{P}$, de ordem $N + 1$, inicializada de acordo com uma constante δ:

$$\mathbf{P}_0 = \delta^{-1}\mathbf{I}_{N+1}$$

onde $\mathbf{I}_{N+1}$ é a matriz identidade de ordem $N + 1$.

Em cada iteração, o vetor com os N últimos valores de entrada é atualizado:

$$\mathbf{x}_k = \begin{bmatrix} x(k) \\ x(k-1) \\ \vdots \\ x(k-N+1) \end{bmatrix}$$

E o valor do erro é calculado:

$$e_k = d_k - \mathbf{w}_k^t \mathbf{x}_k$$

O vetor auxiliar $\mathbf{g}_k$ é definido:

$$\mathbf{g}_k = \mathbf{P}_{k-1}\mathbf{x}_k^* \left\{\lambda + \mathbf{x}_k^t \mathbf{P}_{k-1}\mathbf{x}_k^*\right\}^{-1}$$

A matrix $\mathbf{P}$ é atualizada:

$$\mathbf{P}_k = \lambda^{-1}\mathbf{P}_{k-1} - \mathbf{g}_k \mathbf{x}_k^t \lambda^{-1}\mathbf{P}_{k-1}$$

Finalmente, o vetor dos coeficientes é atualizado:

$$\mathbf{w}_k = \mathbf{w}_{k-1} + e_k \mathbf{g}_k$$

O cálculo do vetor $\mathbf{g}$ envolve a operações com matrizes, o que aumenta significativamente a complexidade computacional desse algoritmo.

4.8 Adição de Filtros Adaptativos no Sistema de VoIP

Após adicionar um filtro adaptativo para cancelamento de eco ao sistema de voz sobre IP apresentado, o desenho completo da solução fica como mostrado na figura 4.13.

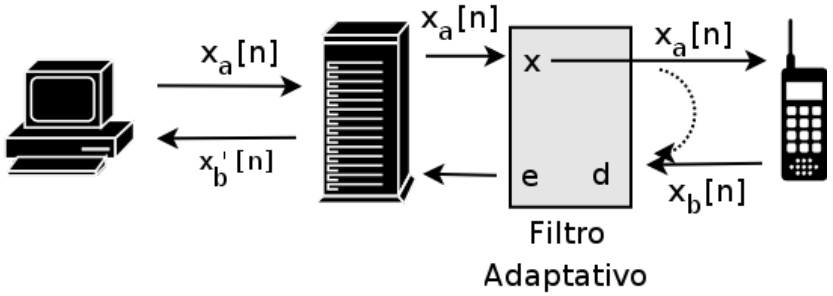

Figura 4.13: Adição de um filtro para cancelamento de eco ao sistema VoIP.

O sinal de voz recebido da Internet (x_a), que eventualmente gerará o eco, é utilizado como entrada para o filtro adaptativo. O sinal recebido da linha telefônica, que contém o sinal desejado (x_b) e o eco de x_a, é utilizado como sinal de referência.

O filtro adaptativo tentará transformar o sinal de entrada, x, no sinal de referência, d. Entretanto, como o sinal de entrada não possui correlação com o sinal x_b, o máximo que o filtro adaptativo conseguirá será transformar o sinal de entrada no seu eco, com o qual ele possui correlação. Nesse caso, após a convergência, o sinal de erro do filtro adaptativo será o sinal desejado, x_b, e será enviado para a Internet.

4.9 Conclusão

Nesse capítulo os principais conceitos teóricos sobre os filtros adaptativos foram apresentados, e mostrou-se como eles podem ser utilizados no cancelamento de eco.

Alguns modelos de filtros, das famílias LMS e RLS, foram apresentados, e seus algoritmos de adaptação foram definidos.

No próximo capítulo, esses modelos de filtros serão utilizados para cancelar o eco em sistemas VoIP, e seus resultados serão analisados.

Capítulo 5

Filtros Adaptativos - Resultados

5.1 Introdução

Neste capítulo, os modelos de filtros adaptativos apresentados no capítulo anterior serão testados, e seus resultados serão comparados.

A primeira parte dos testes aplicará os algoritmos a sinais contaminados com eco gerado artificialmente, na seção 5.2. O eco será gerado, na sub-seção 5.2.1, a partir de um sinal aproximadamente estacionário. Esse sinal será utilizado em dois testes. No primeiro, a ordem dos filtros será variada, e o efeito dessa variação será apresentado. No segundo, a ordem dos filtros será fixa, mas os coeficientes de adaptação serão variados. Serão considerados os seguintes algoritmos adaptativos: LMS, NLMS, SDLMS, SELMS, SSLMS, BNDR e RLS.

A segunda parte dos testes, descrita na seção 5.3 utiliza um ruído real, do tipo acústico. Um dos algoritmos de filtragem adaptativa será aplicado a um sinal contaminado por um eco real. Devido às características do eco real, será necessário cancelar o atraso existente entre o ruído original e o seu eco, o que será discutido na sub-seção 5.3.1.

5.2 Remoção de Eco Artificial

Inicialmente, os filtros adaptativos foram utilizados para cancelar um tipo de eco gerado artificialmente, utilizado para contaminar manualmente um sinal de voz conhecido, contido no arquivo `sinal.wav`.

Um segundo sinal, do arquivo `ruido.wav`, foi utilizado como valor original do ruído. As amostras desse sinal passaram por um filtro FIR de 10^a ordem, para gerar o eco:

```
ruido = wavread ('ruido.wav');
sinal = wavread ('sinal.wav');

ruido = ruido / max (ruido);
sinal = sinal / max (sinal);

h = [7 -9 3 -1 5 -2 8 3 1 -6]';
h = h / sum (h .\^ 2);

cont = filter (h, 1, ruido);
cont = cont / max (cont);

ruido = ruido (1:length (sinal));
cont = cont (1:length (sinal));
cont = cont + sinal;
cont = cont / max (cont);

wavwrite (ruido, 8000, 16, 'ruido.wav');
wavwrite (cont, 8000, 16, 'cont.wav');
```

O ruído original passa pelo filtro, com coeficientes definidos, arbitrariamente, como `[7 -9 3 -1 5 -2 8 3 1 -6]`. O sinal contaminado, armazenado no vetor `cont`, é formado adicionando-se o sinal original ao resultado da filtragem do ruído.

Os dados utilizados pelo filtro adaptativo são, portanto, os contidos nos arquivos `ruido.wav` e `cont.wav`. Na nomenclatura utilizada no capítulo anterior, o ruído será x e o sinal contaminado será d. O erro do filtro adaptativo deverá ser semelhante ao sinal original, armazenado em `sinal.wav`.

O filtro adaptativo é construído sem utilizar os valores dos coeficientes do filtro gerador do eco, pois esse conhecimento não está disponível nas aplicações que interessam ao projeto.

Os tipos de filtro adaptativos utilizados para esses testes foram: LMS, NLMS, SELMS,

SDLMS, SSLMS, BNDR e RLS.

Nesses testes, dois tipos de ruído foram utilizados. Inicialmente, utilizou-se um ruído de características estatísticas estacionárias: o som gerado por um secador de cabelo. Posteriormente, utilizou-se um ruído não estacionário: uma gravação de sons de trânsito.

5.2.1 Ruído Estacionário

Na primeira parte do teste utilizou-se um ruído estacionário, exibido na figura 5.1, obtido a partir de um secador de cabelos. Isso possibilitou a análise e a comparação dos resultados obtidos com os diversos algoritmos. Em especial, buscou-se comparar o efeito da variação da ordem e do coeficiente de adaptação sobre o resultado obtido.

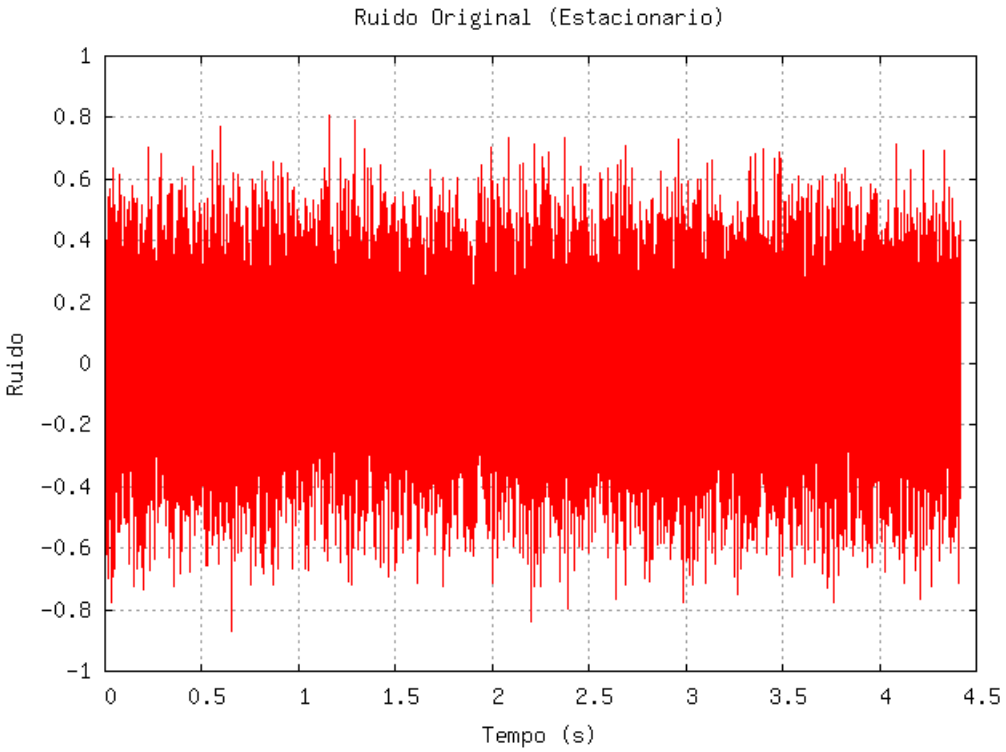

Figura 5.1: Ruído estacionário.

Nas duas análises, o resultado gerado pelo filtro foi dividido em duas partes distintas: até a convergência e após a convergência. Embora o tempo de convergência entre os diferentes algoritmos (e diferentes coeficientes de adaptação) variasse, utilizou-se um tempo único de convergência para todos os casos. Escolheu-se um tempo de convergência suficiente para que a maioria dois filtros utilizados houvesse convergido. Para os sinais utilizados, o tempo de convergência foi de aproximadamente $0,5$ segundo (4000 amostras).

Variando-se a ordem dos filtros (para todos os algoritmos) e o coeficiente de adaptação (para os algoritmo NLMS e RLS), calculou-se o valor médio do módulo do erro, obtido subtraindo-se o sinal original do sinal gerado pelo filtro.

Variação da Ordem do Filtro

Em geral, ao construir um filtro adaptativo, é necessário efetuar uma estimativa da ordem do filtro original, a planta. Essa estimativa é crucial para o correto funcionamento do filtro. Caso a ordem do filtro seja subestimada, o algoritmo não conseguirá levar os coeficientes para o ponto ótimo, pois sequer existirão coeficientes suficientes para simular o funcionamento da planta.

O impacto causado no resultado por uma superestimação da ordem é menor. Mesmo com a ordem superestimada é possível atingir o ponto ótimo de funcionamento, desde que os coeficientes de maior ordem sejam zero (anulando, assim, o efeito por eles produzidos). Entretanto, coeficientes desnecessários aumentam, significativamente, a complexidade computacional do algoritmo de adaptação do filtro. Além disso, coeficientes extras podem causar problemas numéricos e aumentar o desajuste do filtro.

Como o eco a ser cancelado nesse item foi gerado de forma artificial, a ordem do filtro necessária para cancelá-lo é conhecida (10). O filtro utilizado para gerar o eco possuía ordem 10 e, portanto, essa deve ser a ordem ótima para o filtro adaptativo. Para observar os efeitos da sub e da superestimação da ordem do filtro, quatro testes foram realizados. No primeiro, utilizou-se a ordem ideal do filtro, 10. No segundo, a ordem foi superestimada para 20. No terceiro, a ordem foi subestimada, para 5. No quarto, utilizou-se ordem 100.

A figura 5.2 mostra o valor médio do erro absoluto obtido, para todos os 7 filtros, com as 4 ordens, após o tempo determinado como convergência. Os resultados obtidos estão resumidos na tabela 5.1.

Tabela 5.1: Erro absoluto médio para diferentes ordens.

	Erro médio Após a Convergência			
Ordem	**5**	**10**	**20**	**100**
LMS	0,08253	0,01494	0,01567	0,027914
NLMS	0,08276	0,01520	0,01507	0,017739
SDLMS	0,08258	0,01657	0,01592	0,022138
SELMS	0,08278	0,01913	0,01970	0,030609
SSLMS	0,11867	0,12306	0,12583	0,128382
BNDR	0,08202	0,01585	0,01484	0,016655
RLS	0,08353	0,01095	0,01103	0,010585

Pode-se observar, nos gráficos e na tabela, que a variação obtida no erro absoluto

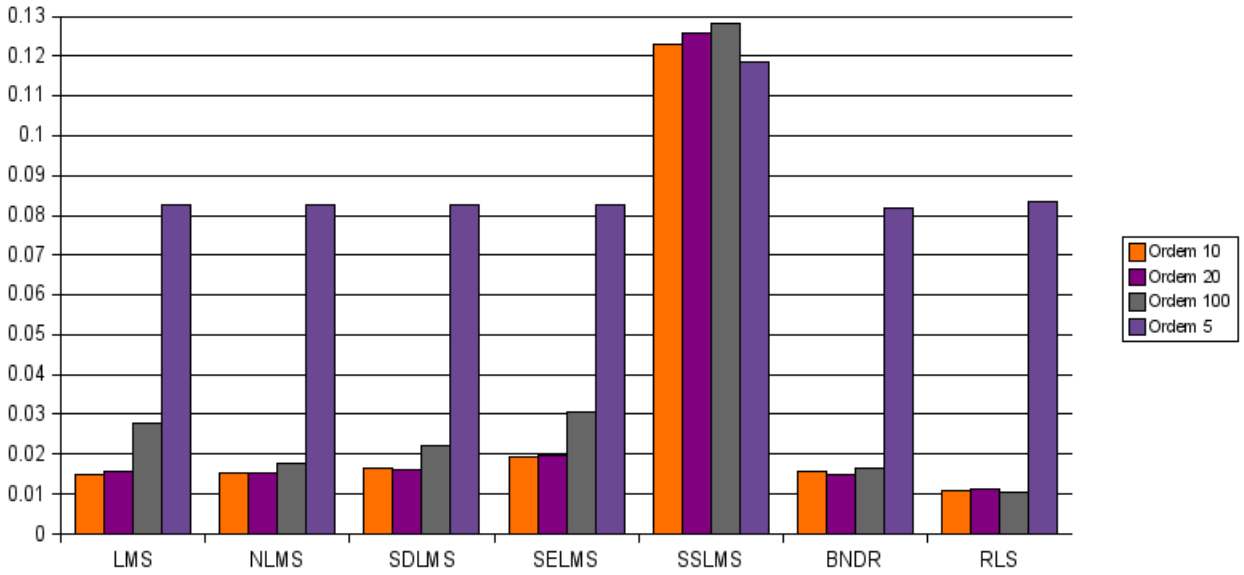

Figura 5.2: Erro absoluto médio após a convergência.

médio obtido quando a ordem foi superestimada para 20 é desprezível. O valor do erro, nesse caso, ficou sempre próximo ao do obtido com a ordem ótima, 10. Os testes dos algoritmos LMS e SELMS mostraram um grande aumento no erro quando a ordem foi superestimada para 100. Esses exemplos mostram que a superestimação da ordem, além de aumentar a complexidade computacional, aumenta o erro da filtragem e o desajuste, além de gerar potenciais problemas numéricos.

O efeito causado pela subestimação é visível, em todos os algoritmos. Em todos os testes realizados com filtro de ordem 5 o resultado foi significativamente inferior ao obtido com ordem ideal ou superestimada (exceto no caso do SSLMS, onde nenhuma das tentativas gerou um resultado aceitável).

Além dos testes objetivos, testes subjetivos informais também foram realizados. Os resultados desses testes comprovaram o que foi medido: o algoritmo SSLMS gerou um resultado incompreensível, enquanto que os outros algoritmos da família LMS mostraram resultados semelhantes, e o RLS mostrou um resultado sensivelmente melhor.

Além disso, o teste informal também confirmou o efeito da variação da ordem sobre o resultado. Os sinais gerados com filtro de ordem 5 mostraram-se de qualidade muito inferior aos outros, de ordem 10, 20 e 100, enquanto esses mostraram resultados semelhantes (exceto para o LMS e SELMS com ordem 100).

Devido aos resultados obtidos, decidiu-se continuar a investigar o funcionamento de dois filtros, um da família LMS e o RLS. O escolhido da família LMS foi o NLMS, pelo

bom resultado apresentado e pelo fato de conseguir lidar melhor com variações na energia do sinal de entrada. Para os próximos testes, a ordem utilizada será 20. Essa escolha justifica-se pois, em situações reais, não será possível determinar com exatidão a ordem da planta, fazendo-se necessário utilizar uma ordem ligeiramente superior.

Conforme será visto no final do capítulo, sinais de eco reais apresentarão um significativo atraso em relação ao sinal que os gerou. Isso poderia ser interpretado como um filtro de ordem grande, onde a grande maioria dos coeficientes é zero, existindo apenas para gerar o atraso, sendo necessário, portanto, projetar um filtro adaptativo de ordem igualmente alta. Essa abordagem não é indicada, pois esses coeficientes, ainda que sejam nulos, aumentarão a complexidade computacional do algoritmo e o erro de desajuste. A melhor abordagem para os casos onde há presença de atraso é adicionar um atrasador artificial fixo na entrada do filtro, de forma que o eco fique sincronizado com o sinal que o gerou. A inserção desse atrasador será detalhada na sub-seção 5.3.1.

Variação do Coeficiente de Adaptação do Filtro

Tendo explorado o resultado da variação da ordem do filtro adaptativo, buscou-se descobrir os efeitos da variação do coeficiente de adaptação sobre o funcionamento do filtro.

Conforme explicado no Capítulo 4, o coeficiente de adaptação, representado, em geral, por μ, determina a velocidade com que o algoritmo irá movimentar o vetor de coeficientes do filtro, de forma a buscar os coeficientes reais.

Tal qual a ordem do filtro, não se deve utilizar valores extremos para o coeficiente de adaptação. Valores muito baixos farão com que o filtro seja "lento", atrasando a convergência. Em casos onde a planta varie isso é especialmente preocupante, pois, nesse caso, os coeficientes do filtro deveriam seguir permanentemente os da planta, o que será impactado pela lentidão do filtro. Por outro lado, caso o coeficiente de adaptação seja muito alto, as variações nos coeficientes do filtro serão muito bruscas, podendo, inclusive, gerar filtros adaptativos instáveis.

Para obter uma medida básica do impacto do coeficiente de adaptação, o sinal contaminado usado no item anterior foi utilizado como entrada do filtro adaptativo. Para validar os resultados obtidos anteriormente, dois filtros foram utilizados: NLMS e RLS, ambos com ordem 20.

O NLMS foi escolhido como o representante da família LMS, por ter apresentado bons resultados, tanto nos testes objetivos quanto nos subjetivos, e por apresentar a vantagem de não ser tão sensível a variações no sinal de entrada. O RLS foi escolhido como o representante de sua própria família.

Para o NLMS, os valores de μ foram variados de 0 até 1, com intervalos de 0.01. A mesma medida do item anterior foi utilizada. O resultado obtido pode ser visto na figura 5.3. Os mesmos valores podem ser vistos na tabela 5.2.

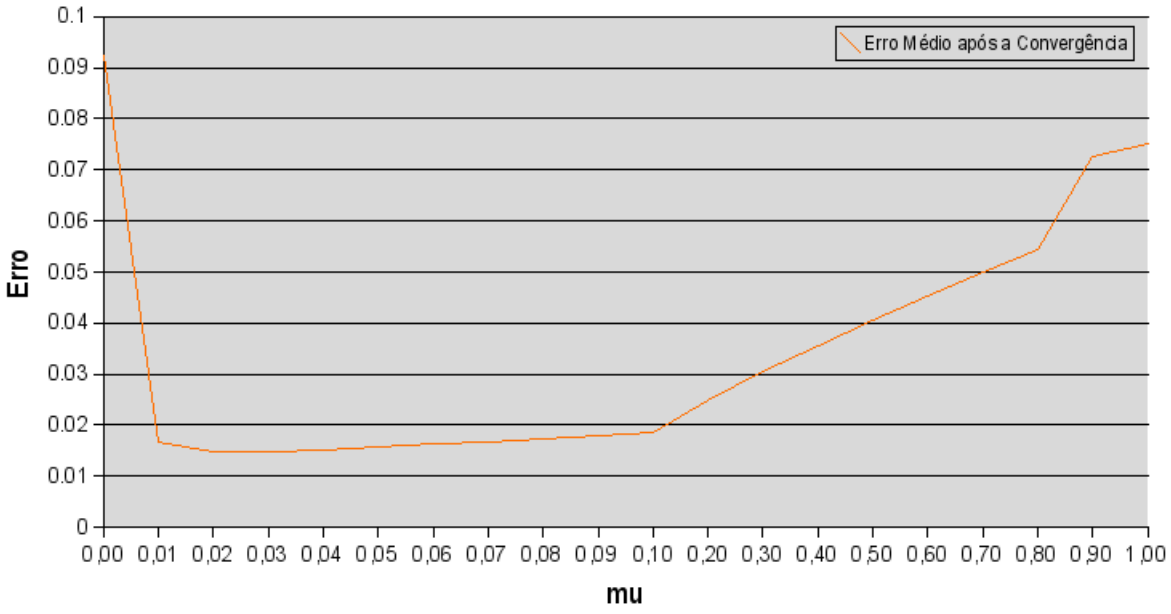

Figura 5.3: Efeito da variação do coeficiente de adaptação.

Tabela 5.2: Erro absoluto médio para diferentes valores de μ.

mu	Erro Médio após a Convergência
0,00	0.09241
0,01	0.01679
0,02	0.01495
0,03	0.01485
0,04	0.01518
0,05	0.01567
0,06	0.01623
0,07	0.01683
0,08	0.01744
0,09	0.01805
0,10	0.01867
0,20	0.02472
0,30	0.03038
0,40	0.03563
0,50	0.04060
0,60	0.04536
0,70	0.04993
0,80	0.05431
0,90	0.07272
1,00	0.07503

Pode-se ver claramente a degradação causada por valores extremos - altos ou baixos - do coeficiente de adaptação.

Para o RLS, os valores de λ foram variados de 0.90 até 1.00, com intervalos de 0.01. A mesma medida do item anterior foi utilizada. O resultado obtido pode ser visto na figura 5.4. Os mesmos valores podem ser vistos na tabela 5.3.

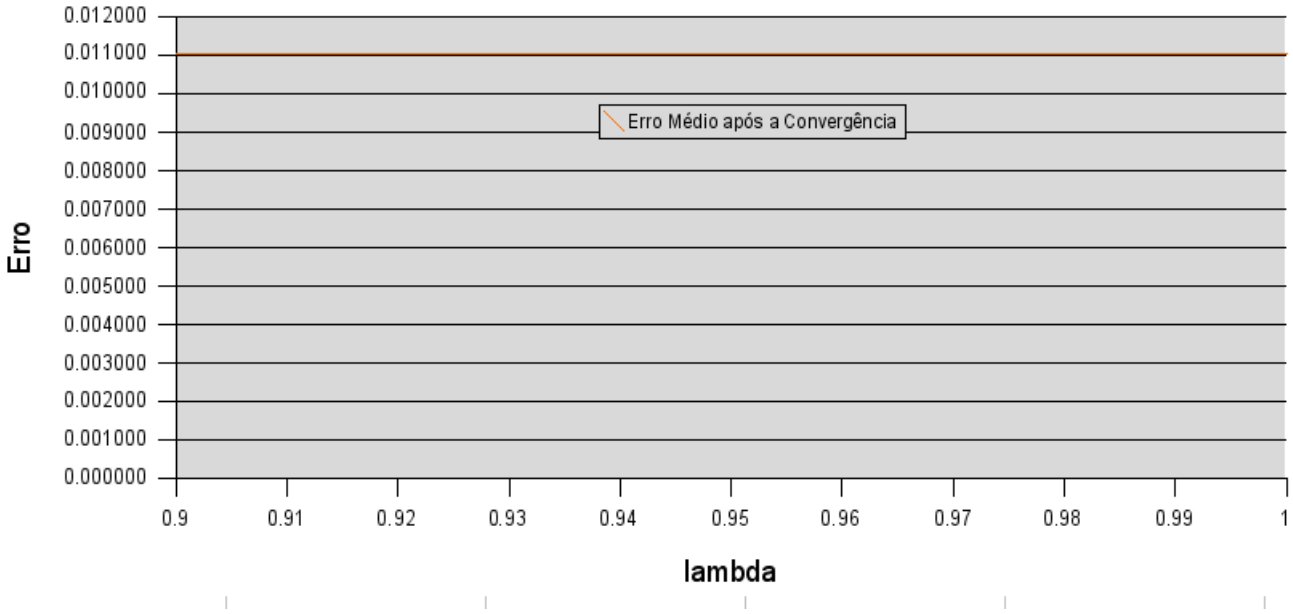

Figura 5.4: Efeito da variação do coeficiente de adaptação.

Tabela 5.3: Erro absoluto médio para diferentes valores de λ.

lambda	Erro Médio após a Convergência
0.90	0.011041
0.91	0.011040
0.92	0.011038
0.93	0.011036
0.94	0.011035
0.95	0.011033
0.96	0.011032
0.97	0.011031
0.98	0.011029
0.99	0.011028
1.00	0.011028

No caso do RLS, não foi possível perceber sensibilidade em relação à variação do coeficiente de adaptação, λ, sobre o resultado final.

Mais uma vez, os testes objetivos foram validados por testes subjetivos, que confirmaram a informação obtida.

Embora esse resultado, assim como o obtido variando-se a ordem do filtro, sejam válidos apenas para os algoritmos em questão, aplicados sobre o sinal em questão, eles demonstram claramente a importância das estimativas da ordem do filtro adaptativo e do seu coeficiente de adaptação. Foi visto, também, que uma pequena superestimação da ordem não causa grandes impactos sobre o resultado. Como, em situações reais, não

é possível determinar com exatidão a ordem da planta, o ideal é fazer uma estimativa e utilizar um valor ligeiramente superior.

5.2.2 Ruído não-Estacionário

Tendo utilizado-se o eco artificial aplicado ao ruído estacionário para mensurar o resultado obtido variando-se os parâmetros do filtro, tornou-se necessário estudar o funcionamento do filtro quando utilizado com ruídos não estacionários.

Para isso, manteve-se a geração artificial do eco, baseada na mesma planta utilizada no item anterior. O sinal utilizado para gerar o eco, entretanto, foi alterado. Utilizou-se o sinal de buzinas de carros, de características não-estacionárias, exibido na figura 5.5

Esse sinal de ruído foi filtrado pelo mesmo gerado de eco do item 5.2, e utilizado para contaminar um sinal de voz conhecido previamente. Posteriormente, o ruído original e o sinal contaminado foram utilizados como entrada para os 2 tipos de filtro que vêem sendo testados: NLMS e RLS. Baseado nos itens anteriores, escolheu-se utilizar filtros de ordem 20. O coeficiente de adaptação foi escolhido como o melhor valor encontrado para cada um dos filtros, no exemplo anterior.

Os resultados podem ser vistos nas figuras 5.6 e 5.7.

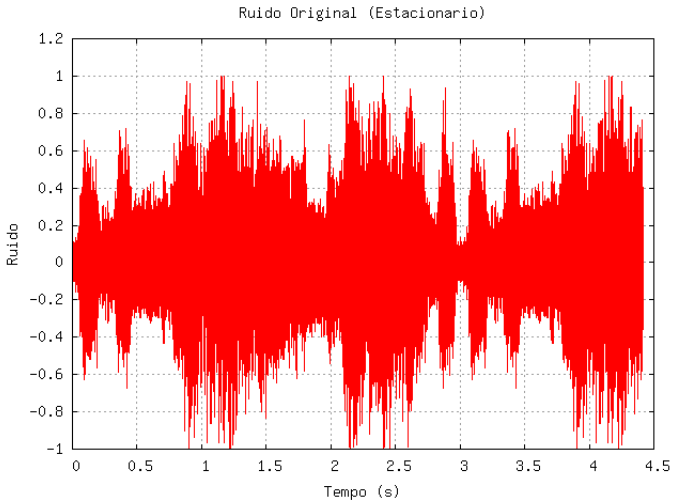

Figura 5.5: Ruído não-estacionário.

Pode-se observar que o filtro adaptativo chega perto do vetor ótimo de coeficientes, e que o erro fica insignificante após poucas amostras. Entretanto, quando as características do sinal variam, o erro volta a aumentar, e o filtro leva mais algum tempo para convergir

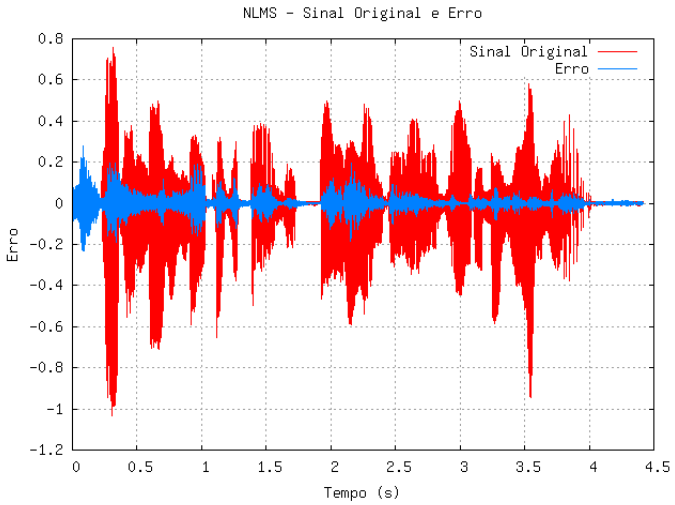

Figura 5.6: NLMS - Comparação entre os dois sinais.

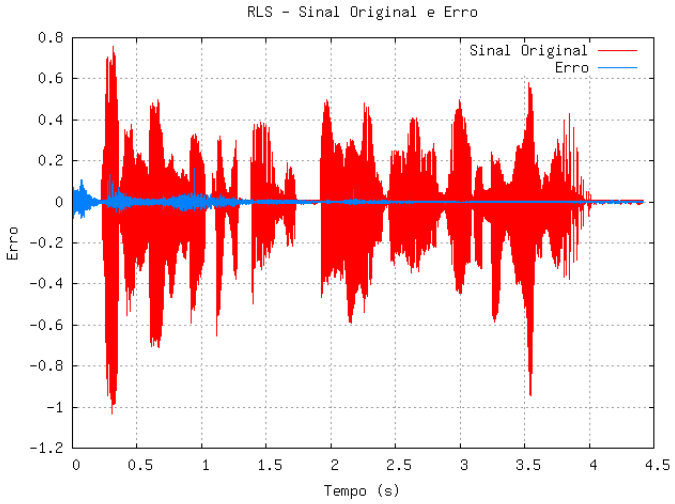

Figura 5.7: RLS - Comparação entre os dois sinais.

e eliminar o erro novamente. Esse efeito é menos perceptível no filtro RLS, onde mesmo as bruscas mudanças no sinal de entrada não conseguem produzir valores altos de erro.

Os sinais gerados foram, mais uma vez, submetidos a testes informais, onde observou-se, principalmente no caso do NLMS, que durante poucos momentos o eco ficava audível, e depois era novamente eliminado. Esse comportamento se repetia algumas vezes, confirmando o resultado obtido nos testes objetivos.

5.3 Remoção de Eco Real

Após testar os algoritmos de adaptação com o eco artificial, dois deles (NLMS e RLS) foram utilizados para cancelar o eco acústico real.

Para gerar o sinal, uma música foi reproduzida em um alto-falante, junto ao qual foi posicionado um microfone. Enquanto a música era reproduzida, uma frase foi lida próxima ao microfone. Durante a gravação, o microfone capturou o sinal de voz e um sinal produzido a partir do sinal da música que estava sendo reproduzida.

O eco gerado é semelhante ao discutido no Capítulo 4, conforme pode ser visto na figura 5.8.

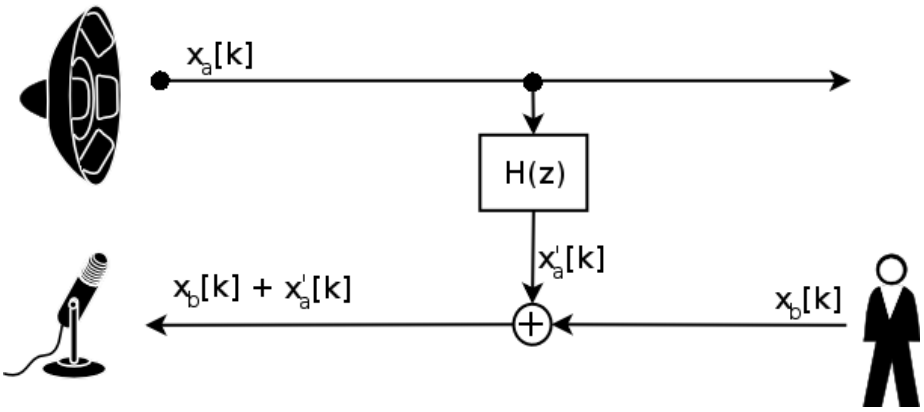

Figura 5.8: Geração de eco real.

Após compensar o atraso da planta - o que será discutido no item 5.3.1 - os sinais foram aplicados aos dois filtros. Em ambos os casos, o resultado obtido foi inferior ao obtido com o eco artificial. Entretanto, boa parte do eco foi retirada do sinal, e a mensagem transmitida no sinal original ficou perfeitamente compreensível. Devido ao cancelamento não ser perfeito, parte do sinal de eco podia ser percebida ao fundo do sinal filtrado.

Nesse teste, o sinal original não está disponível, então nenhum tipo de comparação objetiva pode ser realizada.

5.3.1 Remoção do Atraso

Ao contrário da eco gerado artificialmente, o eco gerado em uma situação real apresenta um grande atraso em comparação à ordem do filtro equivalente. No caso do eco gerado para o trabalho, o atraso entre o sinal da música original e a primeira amostra por ele contaminada foi próximo a 0,5 segundos. Como os sinais são reproduzidos/capturados à uma taxa de amostragem de 8 kHz, esse atraso equivale a aproximadamente 4000 amostras.

O atraso pode ser "incluído" na planta, transformando-a em uma planta de ordem elevada, onde a maior parte dos coeficientes é nula, existindo apenas para simular o

atraso do sinal. Entretanto, não é interessante projetar um filtro adaptativo de ordem alta, pois esses coeficientes responsáveis pelo atraso irão aumentar significativamente a complexidade computacional do algoritmo, as dificuldades numéricas e o desajuste.

A melhor forma de eliminar o atraso é adicionar um atrasador artificial a uma das entradas do filtro adaptativo, de forma que o ruído original e o seu eco fiquem sincronizados. Com a adição do atrasador, pode-se simular uma planta de ordem alta, através do conjunto atrasador-filtro, sem que o processo de adaptação seja realizado sobre um grande número de coeficientes. A forma de inserir esse atrasador pode ser vista na figura 5.9.

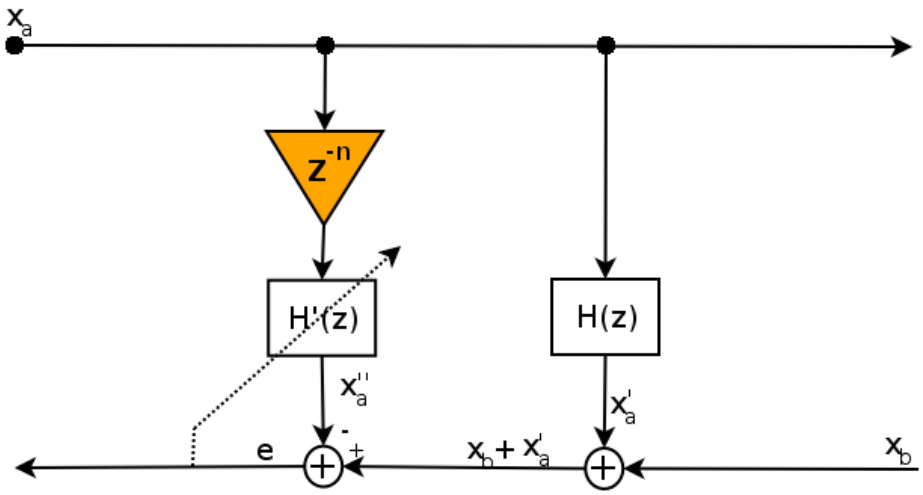

Figura 5.9: Inserção de um atrasador.

Para o correto funcionamento do atrasador artificial, o atraso a ser utilizado precisa ser bem definido, de forma a ser suficientemente próximo do atraso real. Para o teste, o atraso exato foi calculado *off-line*. O valor de atraso obtido foi próximo a 4000 amostras, ou quase $0,5$ segundo.

5.4 Adição do Cancelador de Eco ao sistema VoIP

Após realizar os testes descritos neste capítulo, escolheu-se utilizar o filtro adaptativo NLMS, de ordem 20, com $\mu = 0.05$, para cancelar o eco no sistema VoIP, devido aos bons resultados obtidos nos testes objetivos e subjetivos, e à sua baixa complexidade computacional. Para simplificar o funcionamento, o cancelamento de eco foi adicionado apenas a um dos lados da comunicação, o servidor.

Os testes iniciais do sistema com cancelamento de eco não mostraram resultados satisfatórios. O filtro adaptativo não foi capaz de cancelar, ou mesmo amenizar, o eco.

Para investigar o problema, o servidor, onde o algoritmo de cancelamento de eco foi incluído, foi modificado de forma a armazenar em disco os sinais que ele envia para a caixa de som e os sinais que ele recebe do microfone. Os sinais enviados para a caixa de som foram armazenados no arquivo `adapt_x.wav`, e os recebidos pelo microfone foram armazenados no arquivo `adapt_d.wav`[1].

Após preparar o servidor para armazenar em arquivo os sinais recebidos e enviados, 20 testes foram realizados. Através de análise *off-line*, percebemos que o atraso entre o ruído e o eco não era constante, variando em torno de 726 amostras, com um desvio padrão de 44 amostras[2].

O cancelamento *off-line* do eco foi aplicado aos sinais gerados nesse teste, utilizando o atraso adequado para cada caso, e o resultado foi satisfatório. Testes informais mostraram que grande parte do sinal de eco foi eliminada, e o sinal original ficou perfeitamente compreensível.

Para tentar observar os efeitos do cancelamento de eco no sistema VoIP, um segundo teste foi realizado. Para esse novo teste, utilizou-se um atraso fixo de 480 amostras, o equivalente a três janelas do codificador CELP, e aumentou-se a ordem do filtro para 100. Com o aumento da ordem do filtro, buscou-se ganhar um pouco de flexibilidade na escolha do atraso, já que os coeficientes de ordem mais alta podem simular um atraso adicional.

Nesse segundo teste, foi possível perceber uma diminuição no ruído recebido pelo cliente, embora ele não tenha sido completamente eliminado. Caso o atraso fosse estimado com maior precisão, a ordem do filtro utilizada poderia ter sido menor, o que contribuiria para um melhor funcionamento do cancelamento de eco. Ainda assim, o teste demonstrou a viabilidade de utilizar-se algoritmos adaptativos para o cancelamento de eco *on-line* no sistema VoIP.

Idealmente, o sistema VoIP deveria estimar, em tempo real, o atraso entre o sinal original e o eco, e passar essa informação para o algoritmo adaptativo, de forma que ele

[1] Os arquivos efetivamente gerados pelo codificador não estão no formato WAV, e devem ser convertidos utilizando a ferramenta SOX.

[2] O tamanho do atraso nesse teste é menor dos que os observados anteriormente, pois o programa VoIP acessa os dispositivos multimídia de forma diferente. Ao invés de utilizar o acesso direto, pelo arquivo /dev/dsp, ele utiliza a biblioteca ALSA, que apresenta um desempenho superior. Originalmente, o programa utilizava o acesso direto, e mostrava um atraso muito superior ao observado com a biblioteca ALSA. Maiores informações sobre a biblioteca ALSA podem ser encontradas em [26].

pudesse apresentar um resultado mais satisfatório.

5.5 Conclusão

Nesse capítulo, os filtros adaptativos apresentados no capítulo 4 foram comparados entre si. Além disso, diversos testes foram realizados para entender o impacto da variação dos parâmetros de projeto sobre o resultado da filtragem.

Os dois parâmetros discutidos foram a ordem do filtro adaptativo e o coeficiente de adaptação. Mostrou-se que os dois precisam ser cuidadosamente escolhidos, já que um valor extremo de qualquer um deles compromete o correto funcionamento do filtro.

Os testes realizados após a adição do cancelamento de eco ao sistema VoIP não se mostraram satisfatórios, devido à dificuldade de estimar o valor do atraso entre o ruído original e o seu eco.

Capítulo 6

Conclusão

6.1 Resumo do Trabalho

Nesse trabalho um codificador de voz CELP foi adaptado para viabilizar a transmissão de voz pela Internet.

O codificador, originalmente desenvolvido por alunos dos Profs. Sérgio Lima Netto e Fernando Gil Vianna Resende Júnior, do Departamento de Engenharia Eletrônica e de Computação (DEL/UFRJ) e do Laboratório de Processamento de Sinais (LPS/UFRJ), sofreu algumas modificações, de forma a adaptá-lo as características da Internet. Dentre as modificações realizadas pode-se citar a adição da quantização dos ganhos e coeficientes LSF e a utilização da biblioteca ALSA para aquisição e reprodução de áudio.

O funcionamento do codificador de voz CELP foi detalhado no capítulo 2. Nesse capítulo, foi apresentado um estudo sobre o funcionamento de codificadores de voz em geral, e, especificamente, sobre o codificador CELP. O algoritmo por ele utilizado foi dividido nos seus principais sub-blocos, e cada um deles foi explicado.

Toda a infra-estrutura necessária para a transmissão de dados pela Internet foi desenvolvida e detalhada no capítulo 3. Nesse capítulo foram apresentadas as principais características dos protocolos básicos da Internet (IP, TCP e UDP), e como as suas particularidades influenciaram as decisões sobre a arquitetura do programa VoIP.

Os capítulos 4 e 5 foram dedicados ao estudo e teste dos filtros adaptativos. A teoria da filtragem adaptativa foi apresentada, junto com as suas quatro principais utilizações. Maiores detalhes foram apresentados sobre utilização de interesse, o cancelamento de

eco. Sete tipo de filtros adaptativos, das famílias LMS e RLS, foram apresentados, e utilizados em diversos testes, com diferentes tipos de sinais. Os resultados dos testes foram comparados, e levaram à escolha de um tipo de filtro mais indicado para o sistema VoIP. Os resultados da adição do cancelamento de eco ao sistema foram apresentados, e o possível motivo para o seu não-funcionamento foi identificado.

6.2 Trabalhos Futuros

A implementação do sistema ainda não está completa. Alguns pontos ainda podem ser trabalhados:

6.2.1 Adição do Cancelador de Eco ao Sistema VoIP

A adição do filtro para cancelamento de eco ao sistema VoIP ainda não foi realizada de forma satisfatória. Para finalizar a adição ainda é necessário resolver alguns problemas, como o fato de o atraso entre o sinal original e o eco não ser conhecido (e ser, possivelmente, variável), e de não ter sido desenvolvida uma forma eficiente de estimar os coeficientes do filtro.

6.2.2 Adaptar o Sistema para Redes com Perdas

A Internet é uma rede onde pacotes de dados muitas vezes são perdidos. A perda de pacotes causa uma grande degradação no funcionamento do codificador CELP, já que um pacote perdido é suficiente para de-sincrozinar os dicionários adaptativos das duas partes envolvidas, o que causará um erro crescente na decodificação, até que ela fique ininteligível.

Uma possível solução para esse problema é adicionar uma rotina de sincronização dos dicionários adaptativos, de forma a garantir que os efeitos de um pacote perdido sejam rapidamente eliminados

6.2.3 Detecção de Silêncio

Em uma conversa normal, é esperado que apenas uma das partes esteja falando em um determinado momento. O sistema VoIP pode ser modificado, de forma a tirar proveito

desse fato.

A modificação deveria detectar o silêncio de uma das partes, e enviar, ao invés de um pacote completo de dados, um pacote simplificado, contendo apenas a informação de que foi detectado silêncio. A outra parte deveria receber essa informação e reproduzir um leve ruído de fundo, para simular o silêncio.

A detecção de silêncio oferece duas vantagens: a redução da quantidade de dados transferidos, já que os "pacotes de silêncio" seriam menores do que os pacotes normais, e a possibilidade de efetuar a sincronia dos dicionários adaptativos do codificador CELP, o que deixaria o sistema mais adaptado ao ambiente da Internet.

6.2.4 Conclusão

O trabalho desenvolvido ao longo desse projeto final abordou diferentes tópicos, como codificação de voz, filtragem adaptativa e arquitura TCP/IP.

O sistema desenvolvido ainda está aquém dos sistemas comerciais, por ser apenas uma prova de conceito da transmissão de voz sobre IP. Alguns problemas críticos do sistema atual foram apresentados, de forma a orientar trabalhos futuros.

Referências Bibliográficas

[1] Maia, Ranniery da Silva, *Codificação CELP e Análise Espectral de Voz*, Rio de Janeiro, 2000.

[2] Moura, Natasha da Rocha, *Análise Preliminar Da Robustez Do Codificador De Voz CELP*, Rio de Janeiro, 2005.

[3] Diniz, Filipe Castello da Costa Beltrão, *Implementação De Um Codificador De Voz CELP Em Tempo Real*, Rio de Janeiro, 2003.

[4] Diniz, Paulo Sergio Ramirez., da Silva, Eduardo Antônio Barros., Netto, Sergio Lima, *Processamento Digital de Sinais*, 2005, Bookman.

[5] FLAC, *FLAC - Free Lossless Audio Codec*, http://flac.sourceforge.net/, Agosto/2006.

[6] Grupo VOIP NCE/UFRJ, *VOIP - Laboratorio de Voz sobre IP*, http://www.voip.nce.ufrj.br/, Agosto/2006.

[7] Fraunhofer-Gesellschaft, *Fraunhofer IIS - Audio & Multimedia Realtime Systems - MP3: MPEG Audio Layer-3*, http://www.iis.fraunhofer.de/amm/techinf/layer3/, Agosto/2006.

[8] Nelson, Mark, *The Data Compression Book*, 1995, M & T Books.

[9] Press, W. H., Teukolsky, S. A., Vetterling, W. T., Flannery, B. P., *Numerical Recipes in C*, 1992, Cambridge University Press.

[10] Haykin, Simon, *Communication Systems*, 2001, John Wiley & Sons.

[11] Stevens, W.R., *TCP/IP Illustrated, Volume 1 - The Protocols*, 1994, Addison-Wesley Pub Co.

[12] Widrow, B., *Adaptive Signal Processing*, 1985, Prentice Hall.

[13] Treichler, J. R., *Theory and Design of Adaptive Filters*, 1987, John Wiley & Sons, Inc.

[14] Internet Engineering Task Force, http://www.ietf.org/overview.html, Maio/2006.

[15] Internet Engineering Task Force, *Internet Protocol*, http://www.ietf.org/rfc/rfc0791.txt.

[16] Internet Engineering Task Force, *User Datagram Protocol*, http://www.ietf.org/rfc/rfc0768.txt.

[17] Internet Engineering Task Force, *Transmission Control Protocol*, http://www.ietf.org/rfc/rfc0793.txt.

[18] Zimmermann, Hubert, *OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection*, IEEE Transactions on Communications, vol. 28, no. 4, Abril/1980, pp. 425 - 432.

[19] Skype Limited, *About SkypeOut*, http://www.skype.com/products/skypeout/, Maio/2006.

[20] Brasil, E.F., Lamas, R. de M. L. S., Maia, H. C., Moura, N. da R., *μfone: Conexão entre Internet e Telefone*, 2004.

[21] Haykin, Simon, *Adaptive Filter Theory*, 3rd Edition, Prentice Hall.

[22] Marques, P. A. C., *Introdução à Filtragem Adaptativa*, Instituto de Engenharia de Lisboa.

[23] Connexions, *Adaptive Filters*, http://cnx.org/content/col10280/latest/, Maio/2006.

[24] Apolinário Jr., J. A., Campos, M. L. R. de, Diniz, P. S. R., *The Binormalized Data-Reusing LMS Algorithm*, Revista da Sociedade Brasileira de Telecomunicações, Rio de Janeiro, RJ, Brasil., v. 13, n. 1, p. 49-55, 1998.

[25] Trench, W. F., *An Algorithm for the Inversion of Finite Toeplitz Matrices*, J. SIAM, 1964, vol. 12, pp. 515-522.

[26] *Advanced Linux Sound Architecture*, http://www.alsa-project.org/, Jul/2006.

Apêndice A

Descrição dos Programas Gerados

Esse apêndice lista os programas desenvolvidos, e descreve a forma correta de utilizá-los.

A.1 Codificação CELP Offline

Esse programa realiza a codificação e decodificação CELP, offline. Ele pode receber como entrada arquivos WAV, ou arquivos codificados, e pode gerar os mesmos tipos de arquivo de saída.

Para realizar a codificação um arquivo WAV, deve-se utilizar a seguinte linha de comando:

```
$ ./celp -c -w -i in.wav -o out.celp
```

Esse comando irá ler o arquivo `in.wav`, e escrever o arquivo `out.celp`.

Para decodificar um arquivo codificado, o comando a ser utilizado é:

```
$ ./celp -d -w -i out.celp -o out.wav
```

Esse comando irá ler o arquivo `out.celp`, e escreverá o arquivo `out.wav`.

Alternativamente, o programa pode ler um arquivo wav, codificá-lo, decodificá-lo, e salvar o arquivo WAV resultante, sem a necessidade de salvar o arquivo intermediário. O comando para essa operação é:

```
$ ./celp -e -w -i in.wav -o out.wav
```

Esse comando irá ler o arquivo `in.wav` e escreverá o arquivo `out.wav`.

O programa está no diretório `celp`.

A.2 Transmissão de Voz pela Internet

A transmissão de voz por IP é feita por dois programas: `ufone_server` e `ufone_client`.

Para a correta operação, deve-se inicializar, em primeiro lugar, o `ufone_server`:

```
$ ./ufone_server
```

Posteriormente, deve-se inicializar o `ufone_client`:

```
$ ./ufone_client
```

Ao ser inicializado, o `ufone_client` irá se conectar ao IP definido pela constante `REMOTE_HOST`. Para alterar esse valor é necessário alterar o código-fonte e re-compilar o programa.

Ao final da operação, ambos os programas exibirão a quantidade de bytes transferida.

A.3 Cancelamento de Eco Offline

O cancelamento de eco offline é feito através do programa `adapt`. Ao ser executado, ele lê dois arquivos: `data/cont.dat` (o sinal contaminado) e `data/ruido.dat` (o ruído original). É importante que o diretório `data` exista, e que ele contenha esses dois arquivos.

O programa deve ser executado da seguinte forma:

```
$ ./adapt
```

Após ler os dois arquivos de entrada, ele aplicará cada um dos sete tipos de filtros adaptativos disponíveis aos sinais, e salvará o resultado no diretório `results`, que também deve existir. Os arquivos terão o nome no formato `filtro.dat`, onde `filtro` será o nome de um dos filtros (lms, nlms, sdlms, selms, sslms, bndr ou rls).

Os scripts `geradat.m` e `gerawav.m` podem ser utilizados para converter arquivos `dat` em `wav` e vice-versa.

A.4 Gerador de Eco Artificial

A geração do eco artificial foi feita utilizando-se o script `contamina.m`. Ele lê os arquivos `ruido.wav` (o ruído original) e `sinal.wav` (o sinal não-contaminado), aplica o filtro ao ruído e salva o sinal contaminado no arquivo `cont.wav`.

A.5 Gerador de Eco Real

O programa `echo` foi escrito para facilitar a gravação do eco real. Ele opera lendo um arquivo `wav` do computador e reproduzindo-o ao mesmo tempo que captura o que estiver sendo falado ao microfone. Dessa forma, é possível saber qual amostra do ruído original estava sendo reproduzida quando uma determinada amostra do sinal contaminado foi capturada.

Os arquivos utilizados pelo programa são: `data/orig_noise.wav`, dê onde é lido o ruído original; `data/syncd_noise.wav`, onde é escrito uma versão do ruído sincronizada com o sinal contaminado; `data/cont.wav`, onde o sinal contaminado é armazenado.