Sumário

Capítulo 1

Introdução

1.1 Introdução

Atualmente estamos vivenciando um grande desenvolvimento da telefonia celular digital e das aplicações de voz em computadores pessoais - freqüentemente no contexto de comunicações de multimídia. Mensagem de voz¸ teleconferência, internet, entre outras, são exemplos de aplicações que requerem o sinal de fala em formato digital, de modo que possa ser processado, armazenado e transmitido sob o controle de *software*. A codificação da fala tem a finalidade de obter representações digitais compactas e uma transmissão eficiente, ou seja, representar o sinal de voz com um número mínimo de bits, mantendo sua qualidade perceptiva. Uma das técnicas mais utilizadas para a codificação da fala é a predição linear com excitação por códigos de dicionários (*code excited linear prediction*¸CELP), pois proporciona boa qualidade a baixas taxas de bits.

Neste capítulo serão abordados conceitos e aspectos necessários para a análise desta técnica de codificação de voz. Primeiramente será realizada uma discussão do significado e dos aspectos relevantes para a codificação. Também serão abordadas características da fala e padrões de codificadores. Finalmente realizar-se-ão a descrição dos objetivos e a organização do projeto.

1.2 O que é Codificação

Nas últimas décadas testemunhamos o crescimento explosivo no campo da comunicação digital. Uma das tecnologias principais que permite este crescimento é a codificação da voz. Nesta seção será discutido o significado de codificar a voz¸ apontando as principais dificuldades e considerações envolvidas neste processo.

Desde a invenção do telefone em 1876, que as necessidades das comunicações à distância têm crescido. O desenvolvimento de tecnologias digitais nas últimas décadas criou novos tipos de serviços, que fazem transmissão de voz, imagens e dados. Podemos citar as redes telefônicas digitais, o telefone móvel digital e a videoconferência. Esta compreende a transmissão de áudio e vídeo em tempo real entre os vários participantes. Por todos estes motivos a pesquisa em processamento digital de voz assumiu um papel extremamente importante.

O uso da representação digital de sinais analógicos¸ como voz e vídeo, apresenta muitas vantagens. O sinal digital é menos suscetível a interferências, possibilta a regeneração eficiente do sinal codificado¸

aumenta a privacidade e segurança da comunicação. O processo de digitalização é essencial às redes de comunicações digitais, incluindo o sistema de telefone e é um pré-requisito para as mais novas técnicas de processamento de voz. A digitalização da voz, o processo de converter sinal analógico da voz numa série de valores digitais, é baseado em duas componentes técnicas: amostragem e quantização.

A amostragem é o processo de anotar o valor de um sinal em pontos de tempo regulares. Tecnicamente, e de acordo com o teorema de Nyquist, se for guardado o valor do sinal numa taxa pelo menos duas vezes mais rápida do que a maior frequência do sinal, então será possível utilizar esses valores para reproduzir o sinal exato, mesmo que se tenha ignorado por completo os valores do sinal entre os tempos de amostragem.

Quando se "guarda o nível do sinal", tem-se ainda uma quantidade analógica, que pode ter qualquer valor. Com a quantização, os valores são aproximados com o valor padrão mais próximo, onde o "padrão" depende das características do sistema digital. Por exemplo, um sistema que represente valores com oito bits permite 256 combinações de bit. Se o sinal da voz variar entre, por exemplo, +5V e -5V, então 256 níveis fornecem cerca de 0.039V por nível numa escala de 10V. Os valores da amostragem do sinal que ficam entre os valores padrão (quantizados) são arredondados ao valor mais próximo. Os sistemas de 16 bits permitem 65.536 valores; o mesmo sinal variando sobre uma escala de 10V pode ser aproximado de 0.000015V, ou com uma precisão até 256 vezes a mais do que com o sistema de 8 bits. Pesquisas mostram que a largura de banda para voz pode estender-se a 3,300Hz. O equipamento do telefone é projetado para passar freqüências até este limite, e filtra freqüências mais elevadas. Uma vez que os filtros requerem um intervalo de transição para que o filtro se torne eficaz, foi escolhida uma taxa de 8.000 amostras por segundo.

Mas o que seria codificação? Para entender o conceito de codificação¸ imaginemos o seguinte exemplo: ao lermos um texto qualquer, percebemos que o autor freqüentemente repete alguns fatos ou inclui algumas palavras desnecessárias para a compreensão do mesmo. Esta redundância é introduzida para facilitar o entendimento. Mas um leitor cuidadoso e experiente é capaz de compreender o texto, mesmo que este não tenha redundância. O mesmo ocorre com a codificação de voz. Estudando as propriedades da voz, compreendendo sua natureza de produção e de percepção, pode-se aplicar este conhecimento aos problemas de processar o discurso, utilizando um algoritmo capaz de tratar estas redundâncias. Este algoritmo de codificação atua na informação que o conversor A/D emite ao conversor de D/A. O resultado óbvio é a compressão do sinal digital.

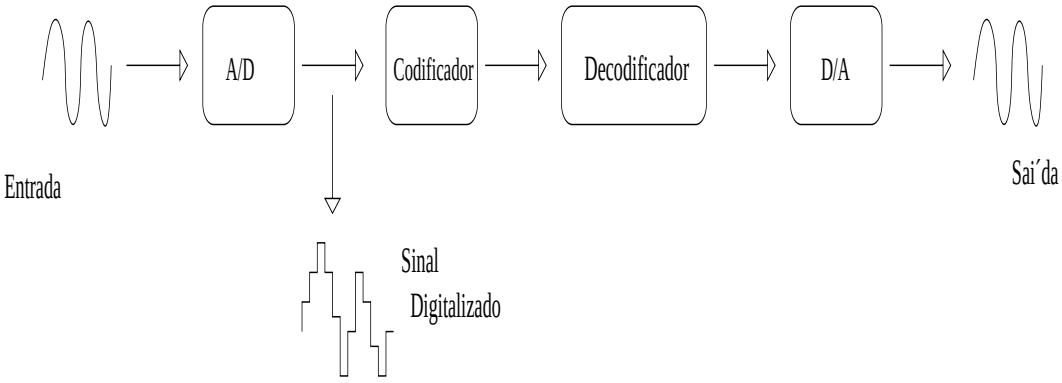


Figura 1.1: Esquema simplificado de codificação.

A performance de um codificador de voz é analisada pela sua taxa de bits (isto é, seu grau de compressão), a complexidade computacional (MIPs e memória), qualidade da voz e robustez. Portanto,

ao selecionar um codificador de voz é necessário considerar fatores específicos da aplicação. Como já dito, um dos fatores é a taxa de bits, que é determinada freqüentemente pela largura de faixa ou capacidade de armazenamento requerida. Outro aspecto é a complexidade que determina o MIPs (milhões de operações por segundo) e as exigências de memória de execução em tempo real. Na maioria das aplicações, abaixar a complexidade resulta em custos mais baixos de material e um consumo menor de potência. Um terceiro fator é a qualidade da voz. Esta é uma medida subjetiva, e que também pode ser examinada através da relação entre a potência do sinal e do ruído de quantificação (*signal to noise ratio*, SNR). A consideração destes fatores diferentes determina a robustez do codificador.

Em resumo, a codificação de voz e seus algoritmos de compressão têm a finalidade de obter representação digital compacta do sinal de áudio, visando sua transmissão e armazanamento. O objetivo principal da codificação é representar o sinal com um número mínimo de bits, gerando na saída um sinal o mais fiel possível do sinal original.

1.3 Características da Fala

O homem fala graças ao aparelho fonador¸ constituído pelos pulmões¸ traquéia e laringe. Através do controle dos centros nervosos, adaptamos estes orgãos, de modo que os sopros por ele emitidos são trabalhados pelos lábios, língua, dentes, palato e nariz.

Os sons ou fonemas da fala são classificados como sonoros, surdos e plosivos, cada um associado à uma determinada forma de excitação. A Figura 1.2 ilustra o processo da fala. O ar é empurrado do pulmão através do trato vocal. Para o som sonoro, as cordas vocais vibram . A taxa em que as cordas vocais vibram determina o *pitch* da voz. As mulheres e as crianças tendem a ter o *pitch* elevado (vibração rápida), enquanto os homens adultos tendem a ter o *pitch* baixo (vibração lenta). Para sons surdos, as cordas vocais não vibram, ficando abertas.

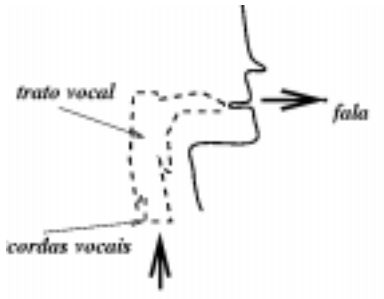


Figura 1.2: Origem da fala.

Os sons sonoros são quase periódicos no domínio do tempo, e o período relativo à sua freqüência fundamental é denominado *pitch*¸ sendo este parâmetro uma característica muito importante do sinal de fala.

A Tabela 1.1 ilustra os sons e suas respectivas excitações. Já nas Figuras 1.3 e 1.4 podemos notar uma representação de um sinal surdo e a peridiocidade de um sinal sonoro, respectivamente.

Tabela 1.1: Tipos de Excitações

Tipo de Excitação	Aproximação	Exemplo
Sonoro	Pulsos Periódicos	Vogal ´a´ de casa
Surdo	Ruído	Pronúncia ´ch´ em chapéu
Plosivo	Pulso Simples	Pronúncia ´b´ em bola

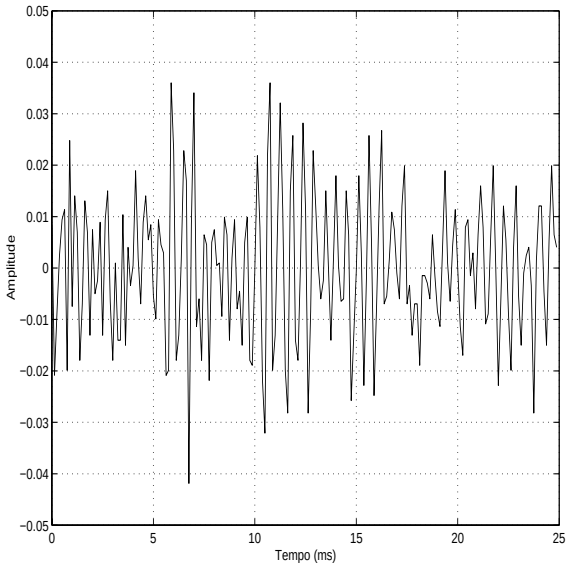


Figura 1.3: Trecho de som surdo.

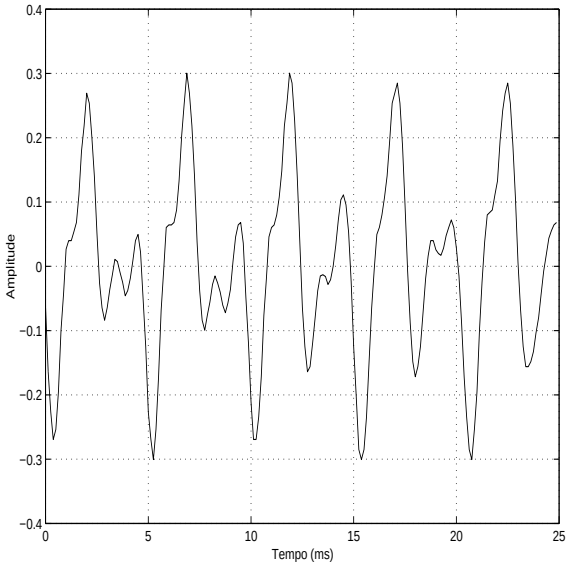


Figura 1.4: Trecho de som sonoro.

1.4 Codificadores

Nesta seção serão discutidas as principais técnicas de codificação. Os codificadores são divididos em três classes:

- *Codificadores de forma de onda;*

- *Codificadores fonte;*
- *Codificadores híbridos.*

Os codificadores de forma de onda são usados tipicamente em taxas de bit elevadas, sendo obtido um sinal com qualidade muito boa. Os codificadores fonte operam em taxas de bit muito baixas, mas não possuem boa qualidade e tendem a produzir um sinal sintético. Os codificadores híbridos procuram preencher a lacuna entre as duas primeiras classes apresentadas; obter boa qualidade à baixas taxas de bits.

O codificador estudado neste trabalho é um codificador híbrido.

1.4.1 Codificadores de Forma de Onda

Os codificadores de forma de onda tentam, sem usar nenhum conhecimento de como o sinal a ser codificado foi gerado, produzir um sinal reconstruído cuja forma de onda seja o mais próximo possível ao original. Geralmente são codificadores de baixa complexidade e produzem resultado de qualidade elevada em taxas acima 16 kbits/s. Quando a taxa de dados é menor que este valor, a qualidade do sinal obtido se degrada rapidamente.

Um exemplo muito conhecido de codificador de forma de onda é a modulação por códigos de pulso (*pulse code modulation*). O PCM para voz é um método de modulação em que o sinal é amostrado e digitalizado em patamares pré-definidos.

1.4.2 Codificadores Fonte

Os codificadores fonte operam usando um modelo de como a fonte foi gerada, na tentativa de extrair do sinal que está sendo codificado os parâmetros do modelo. Estes parâmetros do modelo são transmitidos ao decodificador. Os codificadores fonte para a fala são chamados *vocoders*. Nestes codificadores¸ o intervalo vocal é representado por um filtro e excitado por uma fonte de ruído branco para segmentos surdos, ou um trem de pulsos separados pelo período do *pitch* no caso de sons sonoros. Conseqüentemente, a informação que deve ser emitida ao decodificador é a especificação do filtro-modelo, um *flag* de surdo/sonoro, a energia do sinal da excitação e o período do *pitch*.

O sinal obtido embora inteligível não soa natural. Mesmo elevando-se a taxa de bit utilizada, o desempenho não apresenta melhora considerável¸ isto devido ao uso de um modelo simplificado da produção da fala.

A codificação por predição linear (*linear predictor coding*, LPC), desenvolvida por militares americanos, é um exemplo de codificador fonte[14]. Ela adapta o sinal de voz por um modelo analítico para a transmissão e depois decodifica para gerar uma voz sintética similar à original. O sinal final obtido possui qualidade robótica, o que torna seu uso somente aceitável para finalidades onde o principal objetivo é apenas a inteligibilidade da mensagem a ser transmitida.

1.4.3 Codificadores Híbridos

Os codificadores híbridos, como já dito, tentam preencher a lacuna entre os codifcadores de forma de onda e os fonte. Como descrito nas seções anteriores, os codificadores de forma de onda fornecem um sinal de boa qualidade em taxas de bit acima de 16 kbits/s. Já os codificadore fonte, operam em taxas menores¸ mas fornecem um sinal somente inteligível, de qualidade robótica. Portanto¸ a proposta dos codificadores híbridos é de se obter compressão do sinal sem perda da qualidade do mesmo. Procura integrar as vantagens do codificador fonte e do codificador de forma de onda, realizando extração de parâmetros como no primeiro e almejando o mesmo nível de qualidade obtido no segundo.

Para o caso de codificadores híbridos de fala, os mais bem sucedidos e usados são os baseados na análise por síntese. Tais codificadores usam o mesmo modelo linear do filtro de predição como encontrado no LPC. Porém, em vez de aplicar estados simples (surdo/sonoro) para a excitação, o sinal da excitação é escolhido tentando combinar a forma de onda reconstruída do discurso tão próxima quanto possível da original do discurso. Os codificadores baseados na técnica de análise por síntese foram introduzidos primeiramente em 1982 por Atal e por Remde[14].

A técnica de codificação analisada neste trabalho é a CELP (*code excited linear predictor*), que se encontra nesta classificação, sendo sua estrutura funcional baseada na análise por síntese. Durante esta análise¸ o sinal de voz é comparado com um modelo analítico do trato vocal. Este erro calculado passa por um filtro perceptual, cuja função é apontar o índice da excitação contida no dicionário que irá produzir um sinal de melhor qualidade. Este processo continua até que seja encontrada a excitação que fornecerá maior qualidade final. A Figura 1.5 ilustra o processo de análise por síntese.

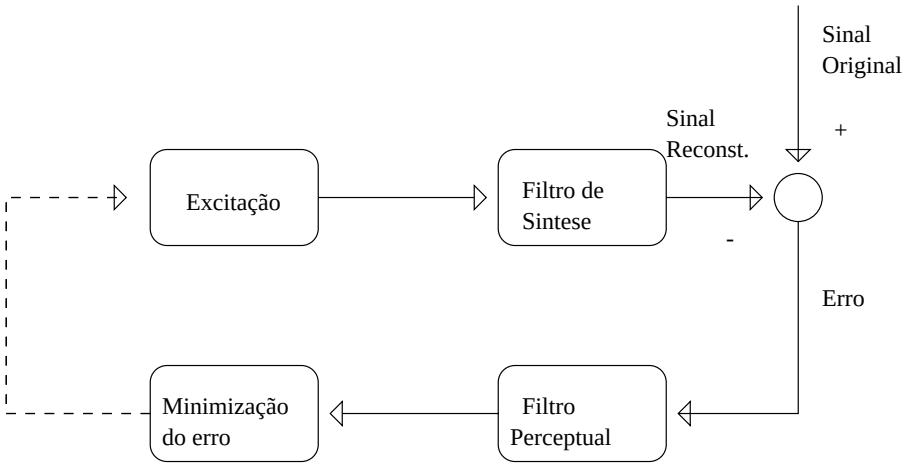


Figura 1.5: Técnica da estimativa de excitação usando o conceito de análise por síntese

1.4.4 Padrões Atuais

Em 24 de maio de 1844 Samuel Morse enviou sua primeira mensagem pública sobre uma linha de telégrafo entre Washington e Baltimore. Dez anos mais tarde a telegrafia se tornara disponível ao público geral. Neste período, entretanto, as linhas de telégrafo não cruzaram fronteiras nacionais porque cada país usou um sistema diferente, com um código próprio, visando proteger o conteúdo de suas mensagens, que tinham fins militares e políticos. Diante desta situação, os países decidiram criar acordos para que pudessem interconectar as suas redes nacionais.

Com a expansão de redes de telégrafo em vários países, servindo como uma ferramenta extraordinária de comunicação, 20 estados europeus decidiram encontrar-se para elaborar um acordo de estrutura. Foram estabelecidas regras comuns de padronização. Em 17 de maio de 1865 foi assinada a primeira convenção internacional do telégrafo. Isto marcou o nascimento do ITU (*International Telecommunication Union*). Desde esse tempo, as telecomunicações continuaram a se desenvolver e a história da ITU reflete inteiramente os avanços que foram feitos. Toda inovação no campo das telecomunicações é combinada pela ação específica da união, que visa integrar as descobertas e fornecer os recursos necessários para responder mais eficazmente às expectativas dos estados membros. Hoje, quase 130 anos mais tarde, as razões que conduziram ao estabelecimento da união e seus objetivos fundamentais são basicamente os mesmos.

A Figura 1.6 ilustra alguns algoritmos de padrões de codificação de fala. Nela podemos observar os comportamentos "taxa x qualidade" destes padrões. Dentre os padrões representados nesta figura podemos destacar os seguintes:

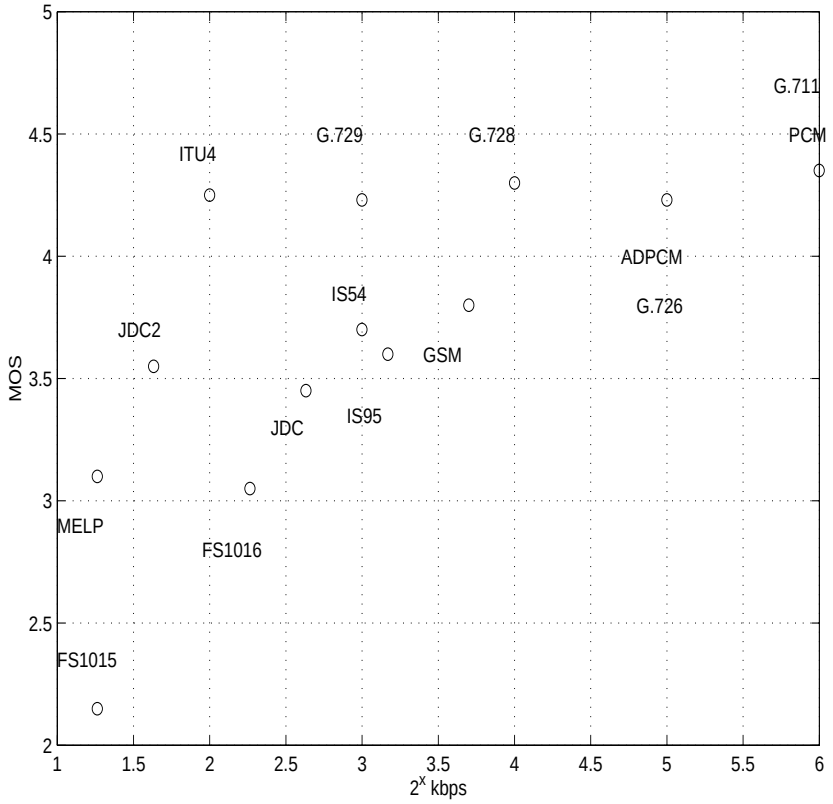


Figura 1.6: Comportamento taxa x qualidade de padrões atuais.

1.4.4.1 PCM (*pulse code modulation*)

O codificador de múltiplos pulsos (PCM) é um formato básico de codificação digital. Cada amostra é representada por uma palavra de código. Alguns formatos PCM como μ-law e A-law PCM promovem um certo grau de compressão, representando com 8 bits por amostra o que seria representado por 12 bits. A 8 kHz, 8 bits por amostra e 1 canal, as técnicas de compressão digital de áudio PCM μ-law e A-law

requerem uma taxa de codificação de 64kbps.

Como pode ser observado na Figura 1.7, as principais operações desta técnica de codificação são a amostragem, quantização e codificação. Na amostragem o sinal de entrada é amostrado como um trem de pulsos retangular, a uma taxa superior ao dobro da maior freqüência do sinal. Obtem-se assim um sinal cuja amplitude assume infinitos níveis de valores. Mas não é necessário transmitir a amplitude exata deste sinal, podendo o sinal ser quantizado, ou seja, as amplitudes do sinal são discretizadas. A existência de um número finito de níveis de amplitude é a condição básica do PCM. Por fim, cada nível de quantização é associado a uma palavra de código binário

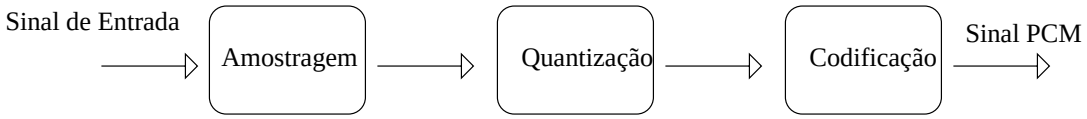


Figura 1.7: Elementos básicos de um sistema PCM

1.4.4.2 LPC (*linear predictive coding*)

A codificação por predição linear (LPC) é um método de compressão digital desenvolvido especificamente para voz. Ele adapta o sinal de voz por um modelo analítico para a transmissão e depois decodifica para gerar uma voz sintética similar à original. Nesta técnica, o sinal de fala é dividido em blocos. Cada bloco é classificado como sonoro ou surdo. Se o sinal for sonoro é calculado o período de pitch. Para ambos os casos determinam-se os parâmetros do filtro de síntese e um ganho. Sendo assim, as informações transmitidas ao decodificador são os parâmetros do filtro de síntese¸ o ganho¸ o *flag* sonoro ou surdo e o período de *pitch*¸ caso necessário.

A Figura1.8 ilustra o diagrama de blocos deste codificador.

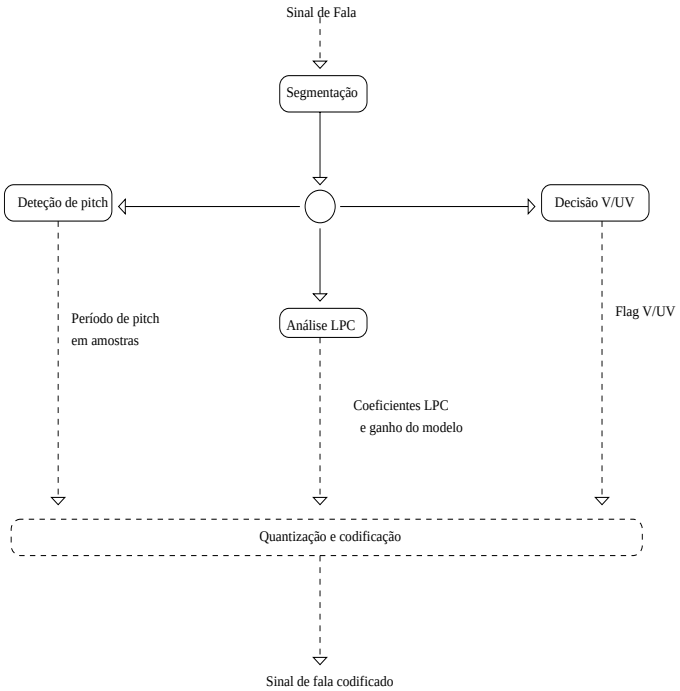


Figura 1.8: Diagrama de Blocos LPC

1.4.4.3 CELP (*code excited linear prediction*)

Esta é a técnica do codificador estudado neste trabalho. Faz o mesmo trato vocal que um codificador LPC. Em adição, calcula o erro para cada excitação, transmitindo os parâmetros que proporcionaram o menor erro. Portanto, produz uma qualidade maior do que a LPC. A recomendação ITU-T G.728 usa uma variação do CELP, LD-CELP, que requer uma banda passante de 16 kbps. O Capítulo 3 trata exclusivamente desta técnica de codificação, e por isto mesmo uma explicação mais detalhada desta família de codificadores é aqui omitida.

1.4.4.4 GSM (*group speciale mobile*)

Durante os anos 80, os sistemas de telefonia celular analógica tiveram um crescimento rápido na Europa. Cada país desenvolveu seu próprio sistema, que eram incompatíveis entre si, no que diz respeito a equipamento e a operação. Esta era uma situação indesejável, porque além do uso do equipamento móvel ser limitado aos limites nacionais, havia também um mercado muito restrito para cada tipo de equipamento. Em razão disto, a produção em escala não poderia ser realizada.

Em 1982, os europeus realizaram uma conferência e deram forma a um grupo de estudo chamado GSM (*group special mobile*) para estudar e desenvolver um sistema público de telefonia móvel. O sistema proposto obedecia a determinados critérios, como boa qualidade subjetiva do discurso, custo baixo do terminal e do serviço, entre outros.

Em 1989, a responsabilidade da GSM foi transferida ao Instituto Europeu dos Padrões da Telecomunicação (ETSI). O serviço comercial teve seu ínicio em meados de 1991, e em 1993 já abrangia 22 países. Embora padronizado na Europa, o GSM não é somente um padrão europeu [16].

1.5 Objetivos

Este trabalho tem a finalidade de descrever detalhadamente um sistema CELP de codificação de fala desenvolvido por Maia em [1]. São apresentadas as características deste codificador; modelagem LPC, dicionários utilizados, método de análise por síntese e quantização. Também é realizada a descrição completa do algoritmo do sistema implementado, com explicações de todas as funções e rotinas utilizadas¸ relacionando com seus respectivos aspectos teóricos.

Outra questão aqui abordada é a caracterização da influência dos dicionários no sinal final obtido. Para tanto¸ foram realizadas seqüências de testes, sendo os resultados obtidos analisados detalhadamente. Também são aplicadas técnicas de aceleração que visam obter a codificação em tempo real com perda mínima de qualidade do sinal. Esta aceleração foi importante para que se pudesse integrar o sistema de codificação ao programa desenvolvido em [15]. Com isto, se propõe a criar um arquivo em Linux capaz de gravar e de reproduzir arquivos de som com a extensão .wav em diversas freqüências. Além disso, ainda é possível se escolher o número de canais para a gravação, sendo 1 canal para mono e 2 canais para estéreo.

O código dos programas utilizados neste projeto são escritos em C para plataforma Unix, o que garante a portatibilidade do software.

1.6 Organização do Trabalho

O Capítulo 2 trata da descrição da estrutura funcional do codificador que utiliza a técnica CELP. Primeiramente é realizada uma abordagem histórica do desenvolvimento desta técnica, relatando as primeiras tentativas de implementação, as idéias iniciais e técnicas que a antecederam. Depois, são descritas as principais questões que envolvem seu projeto e implementação, relatando todos os componentes e técnicas envolvidas.

O Capítulo 3 traz a análise completa e detalhada do algoritmo e códigos utilizados na implementação do codificador. É mostrado o diagrama de blocos do sistema, e descrito, passo a passo, o código do programa. Todos os arquivos de leitura e escrita de dados, cabeçalhos e funções utilizadas são explicadas minuciosamente, sempre relacionando com a questão teórica estudada no Capítulo 2.

O Capítulo 4 é dedicado a testes que visam caracterizar os dicionários fixo e adptativo. É realizado um estudo da evolução qualitativa do codificador em questão, mostrando as características e influências destes dicionários utilizados. Também é feito o treinamento do dicionário fixo. Todos os procedimentos utilizados nos testes são descritos passo a passo, sendo realizadas análises comparativas entre as diferentes configurações utilizadas.

O Capítulo 5 mostra o processo de aceleração do sistema em questão. O objetivo deste capítulo foi o de implementar alterações na configuração inicial do codificador, visando um processamento em tempo real, priorizando também a conservação da qualidade do sinal obtido. São relatadas todas as etapas e técnicas envolvidas neste processo, sempre realizando uma análise do resultado obtido. Nele também é realizada a integração do codificador estudado ao programa de captura de voz pela placa de som. Neste capítulo são apresentadas as modificações realizadas em cada programa e seu resultado final.

O Capítulo 6 é o resumo do trabalho e sua conclusão final. Também são colocadas propostas para trabalhos futuros.

Capítulo 2

Codificador CELP

2.1 Introdução

CELP *(code excited linear predictor)* é uma poderosa técnica de análise da fala, cujo método de codificação propicia boa qualidade a uma pequena taxa de bits. Este capítulo tem como objetivo descrever as idéias básicas deste método de codificação, discutindo as principais questões que envolvem seu projeto e uso.

A Seção 2.2 apresenta a idéia básica deste codificador. Já nas seções seguintes são discutidos tópicos como janelamento e segmentação¸ filtro de síntese¸ interpolação¸ análise por síntese e dicionários fixo e adptativo.

2.2 Idéia Básica

A pesquisa de codificadores de voz começou na década de 40¸ com o trabalho pioneiro de Homer Dudley [12]. A motivação para a pesquisa era a necessidade de se desenvolver sistemas para a transmissão de voz em baixa largura de faixa. Dudley praticamente demonstrou a redundância no sinal de voz e propiciou o primeiro método de análise por síntese. A idéia básica do codificador de voz de Dudley era analisar o *pitch* e o espectro do sinal de voz, utilizando um banco de dez filtros analógicos passa-faixa (representando o trato vocal), sendo excitados por um sinal aleatório. Esta técnica de codificação recebeu atenção especial durante a Segunda Guerra Mundial¸ por possibilitar uma transmissão eficiente e segura.

Com a flexibilidade oferecida por computadores digitais, havia uma tendência natural em experimentar representações digitais mais sofisticadas do discurso. Uma técnica bastante interessante foi desenvolvida por Fant [13]. Seu modelo consistia num sistema que visava modelar o trato vocal e a glote, sendo excitado por um trem de pulsos periódicos, simulando o sinal sonoro, e uma excitação de freqüência e amplitude aleatórias, simulando o sinal surdo. A este sistema foi adicionado um filtro *all-pole*, cujos parâmetros eram obtidos pela análise de predição linear; um processo onde a amostra do sinal atual é predita pela combinação linear de amostras precedentes.

Os esforços da pesquisa nos anos 80 e 90 foram em desenvolver codificadores robustos, com baixa taxa de bits, e capazes de produzir um sinal de voz de alta qualidade para aplicações de comunicações. Estas pesquisas também foram fomentadas pela necessidade de transmissão segura em comunicações celulares e para fins militares. Atal e Schroeder propuseram um algoritmo linear de predição chamado predição

linear com excitação por códigos de dicionários (*Code Excited Linear Predictor*, CELP). Os codificadores CELP também são chamados de codificadores híbridos, pois combinam as características de codificadores fonte e de formas de onda. Na realidade, o padrão CELP compara a voz com um modelo analítico do trato vocal. Ele é derivado do LPC, o codificador de fala por predição linear. Este codificador também modela o processo de fala humana, mas o seu resultado é de baixa qualidade.

Os codificadores baseados na técnica CELP utilizam dicionários, que podem ser compreendidos como uma tabela onde são guardadas as possíveis excitações do sinal de fala. Em geral, são utilizados dois tipos de dicionários: o dicionário fixo, cujas amostras são seqüências gaussianas, visando simular os sons surdos; o outro dicionário é o adaptativo, cujo objetivo é de representar os sons sonoros, onde existem correlações de curto termo (devido, como para sons surdos, ao efeito do trato vocal), bem como correlações de longo termo (correlações da ordem de período de pitch).

O modelo de codificação CELP é baseado na técnica da análise por síntese. Nesta análise, cada seqüência candidata do dicionário passa pelo filtro H(z) e este resultado é comparado com o sinal original, obtendo-se o erro médio quadrático para tal seqüência. Seleciona-se então, a seqüência que proporciona menor erro (esta análise é realizada para os dois dicionários, sendo escolhida uma seqüência de cada um). Cada seqüência é multiplicada por seu respectivo ganho, formando o sinal de excitação x(n). A Figura 2.1 mostra o diagrama de blocos deste codificador.

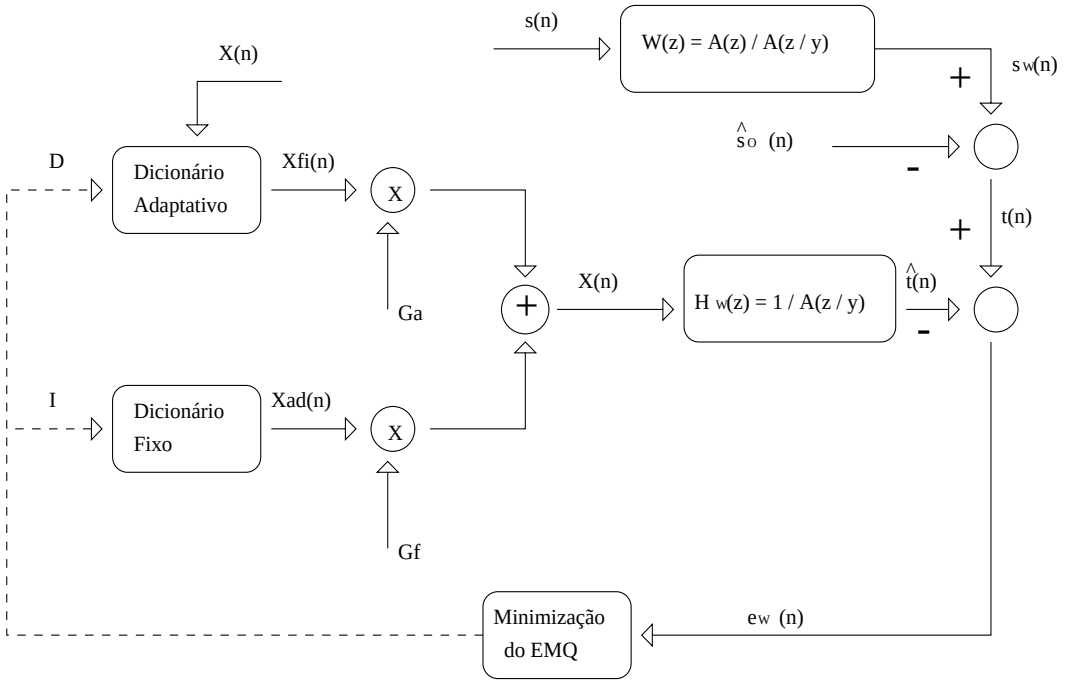


Figura 2.1 - Diagrama de blocos de um codificador CELP.

2.3 Janelamento e Segmentação

Na codificação paramétrica, o processamento do sinal de fala é realizado dividindo-se o sinal em segmentos de onde são extraídas as informações necessárias para sua codificação. Assume-se para isso a ergodicidade e estacionaridade do processo estocástico (sinal de fala) a ser analisado [4]. A primeira

consideração (ergodicidade) é facilmente compreendida, já que se estamos interessados em obter propriedades de uma certa amostra do sinal, faz-se necessário considerar que tais propriedades são comuns a outras amostras que contenham a mesma informação (vogal / a /, por exemplo). A segunda consideração (estacionaridade) tem sua validação para efeitos práticos, quando o segmento de voz é da ordem de 10 a 30 ms.

O sinal de fala é então segmentado para daí serem extraídas as informações. Esta segmentação é feita com uso de janelas, o que implica em distorções no domínio da freqüência e do tempo. Um exemplo são as janelas retangulares, cujas grandes descontinuidades de seus extremos provocam distorções no domínio da freqüência. Em razão disto, usam-se janelas que apresentam menores distorções no dois domínios. Este codificador utiliza uma janela de Hamming de 25ms (200 amostras para uma taxa de amostragem de 8 kHz).

Se uma janela cobrir muitos períodos de *pitch*, ocorrerá uma super resolução em freqüência, causando picos espectrais muito estreitos no espectro de curto termo de segmentos vozeados (produção de som não natural). Para evitar isto, utiliza-se o método de suavização espectral. Este método consiste em alterar a seqüência de autocorrelação do segmento de sinal de fala, fazendo com que o nível da amplitude do espectro de potência se eleve, alargando a banda dos formantes.

2.4 Filtro de Síntese H(z) e sua Quantização

O filtro de síntese é descrito por:

$$H(z) = \frac{1}{A(z)} = \frac{1}{1 - \sum a_i z^{-i}} \tag{2.1}$$

Sua função é modelar os efeitos do trato vocal humano. Os coeficientes do filtro inverso A(z) são obtidos atráves da análise LPC, e devem ser quantizados para serem enviados ao decodificador. A versão quantizada desses parâmetros tem que proporcionar um filtro estável¸ ou seja, os pólos de H(z) devem continuar dentro do círculo de raio unitário mesmo apoś a quantização dos coeficientes.

Os parâmetros utilizados para representar o filtro $H(z)$ foram as freqüências de espectro de linha (*line spectral frequencies*, LSF). Sua utilização proporciona menor sensibilidade a erros de quantização do que outras formas de representação, como os CR (coeficientes de reflexão) e LAR (logaritmo da razão das áreas)[7].

A partir do polinômio A(z) pode-se obter um polinômio simétrico P(z) e outro anti-simétrico Q(z), dados por:

$$P(z) = A(z) + z^{-p-1}A(z) \tag{2.2}$$

$$Q(z) = A(z) - z^{-p-1}A(z) \tag{2.3}$$

$$A(z) = \frac{P(z) + Q(z)}{2} \tag{2.4}$$

Sabendo que se A(z) possui todas as raízes dentro do círculo unitário (fase mínima), pode-se afirmar que todas as raízes de P(z) e Q(z) são distintas e alternam-se em cima da circunferência de raio unitário.

Soma-se a isso o fato de que sendo P(z) e Q(z) polinômios simétrico e anti-simétrico, possuem raízes em -1 e +1 respectivamente. Por serem redundantes, estas raízes podem ser removidas:

$$P_1(z) = \frac{P(z)}{(1+z^{-1})} \}se\ par \tag{2.5}$$

$$Q_1(z) = \frac{Q(z)}{(1-z^{-1})} \}se\ par \tag{2.6}$$

$$P_1(z) = P(z) \}se\ impar \tag{2.7}$$

$$Q_1(z) = \frac{Q(z)}{(1-z^{-2})} \}se\ impar \tag{2.8}$$

Os polinômios $P_1(z)$ e $Q_1(z)$ são simétricos de ordem par, sendo somente necessário determinar metade de suas raízes, pois estas aparecem em pares complexos conjugados. Isto quer dizer que se p é par, p/2 raízes de $P_1(z)$ juntamente com p/2 raízes de $Q_1(z)$ representam P(z) e Q(z), obtendo-se, assim, A(z). Neste trabalho, usa-se p=10. Como dito, as p raízes $\{jw_1\ ,\ ...\ ,\ jw_p\}$ estão sobre a cicunferência de raio unitário. Isto significa que com apenas os ângulos (ou freqüências) pode-se representar A(z). Estas freqüências, portanto, são as LSF.

Para o cálculo das LSF, primeiro obtêm-se os coeficientes de reflexão para cada bloco. Estes coeficientes são obtidos como subproduto do algoritmo de Levinson-Durbin[7] pelo método da autocorrelação, sendo posteriormente convertidos em LSF . Realiza-se, então, a quantização das diferenças entre as LSF consecutivas. Esta quantização é realizada através do treinamento da base de dados (60 segundos de fala, obtidos de 23 frases) pelo algoritmo de Lloyd I [7] para cada elemento a ser escalarmente quantizado. Ficou assim a distribuição de bits para esta quantização:

Tabela 2.1: Distribuição de bits para quantização das LSF.

Quantizador	W1	W2	W3	W4	W5	W6	W7	W8	W9	W10	TOTAL
QDLSF-32	4	4	3	3	3	3	3	3	3	3	32

2.5 Interpolação

Na técnica de codificação CELP, os parâmetros de H(z) são calculados a cada bloco, enquanto o sinal de excitação é determinado para cada sub-bloco pertencente ao bloco. Assim, os parâmetros de um dado bloco são iguais para todos os sub-blocos nele contido.

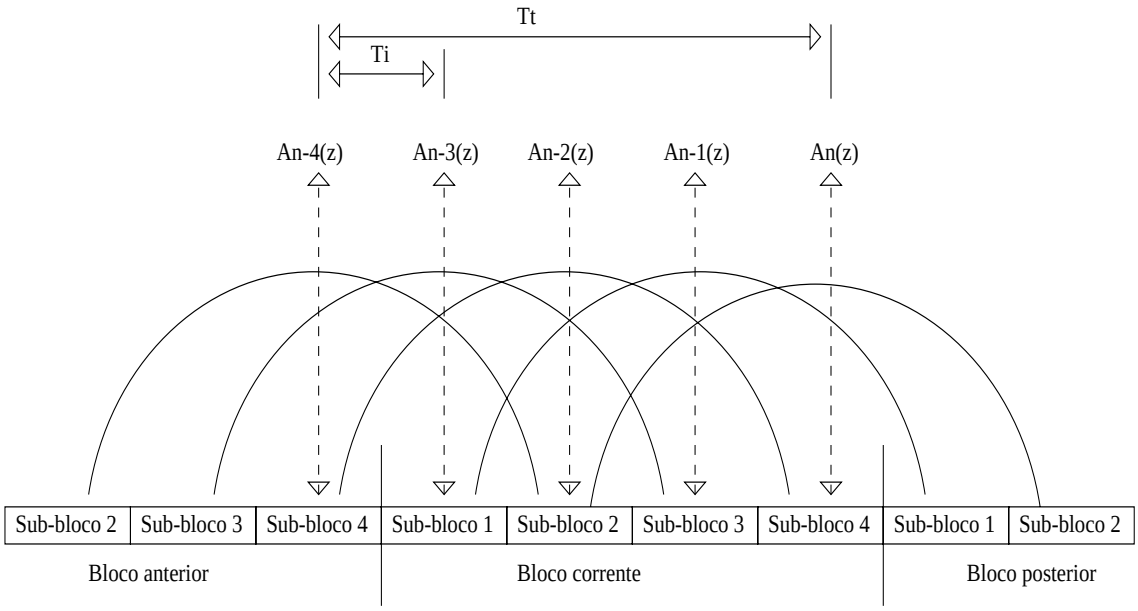


Figura 2.2 - Interpolação

A interpolação é utilizada para produzir uma transição suave entre os conjuntos adjacentes de CPL(coeficientes de predição linear). Como podemos observar na Figura 2.2, o bloco de T_t=20 ms é dividido em 4 sub-blocos com T_i=5 ms. Estes sub-blocos são numerados de 1 a 4, e o i-ésimo coeficiente do n-ésimo sub-bloco é dado por:

$$w_i^n = (1 - q_n)w_i^a + q_n * w_i^c \tag{2.9}$$

$$w_i^n \rightarrow coeficientes\, do\, enésimo\, bloco$$

$$w_i^a \rightarrow coeficientes\, do\, bloco\, anterior$$

$$w_i^c \rightarrow coeficientes\, do\, bloco\, corrente$$

$$q_n = \{0,25; 0,5; 0,75; 1\}$$

Os valores do vetor de peso q_n devem-se ao fato de que os CPL calculados para o bloco são determinados para o quarto subbloco, de acordo com o posicionamento da janela de Hamming.

2.6 Análise por Síntese

Nesta etapa é escolhida a melhor excitação do dicionário adaptativo, bem como a melhor seqüência do dicionário fixo. O sinal é reconstruído e comparado com o sinal original. Esta reconstrução é similar à realizada no decodificador, por isso pode-se afirmar que na estrutura do codificador encontramos o decodificador (Figura 2.3).

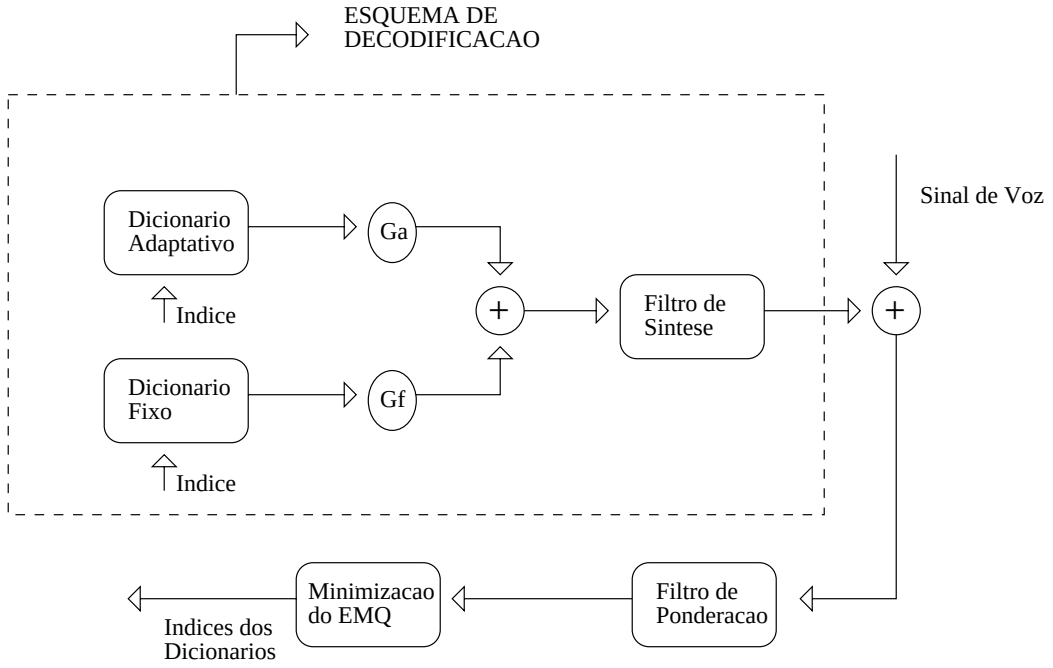


Figura 2.3 - Codificador CELP.

Cada seqüência candidata do dicionário passa pelo filtro H(z) e este resultado é comparado com o sinal original, obtendo-se o erro médio quadrático para tal seqüência. O erro é obtido passando o resultado da diferença entre estes sinais pelo filtro de ponderação W(z). Uma vez escolhida a melhor seqüência candidata, seu índice é guardado e seu ganho calculado. Ao final da análise, estes índices e ganhos referentes às excitações de ambos os dicionários são enviados ao decodificador. Este processo é realizado para cada sub-bloco. Como podemos observar na Figura 2.1, o filtro H(z) é transformado em $H_w(z)$, devido ao deslocamento do filtro perceptual W(z). A resposta do filtro $H_w(z)$ é composta pela resposta com estado inicial zero $\widehat{t}(n)$ e pela resposta à entrada zero $\widehat{s}_o(n)$.

Primeiramente é feita a busca da melhor excitação do dicionário adaptativo, considerando-o como único dicionário. Neste momento, o sinal alvo, ou seja, o sinal com que a excitação filtrada por $H_w(z)$ deve ser comparada, é dado por $s_w(n)$- $s_0(n)$, onde $s_w(n)$ é o sub-bloco de fala perceptual e $s_0(n)$ a resposta a entrada zero. Determina-se também o ganho G_a deste dicionário. Feito isto, determina-se a melhor excitação do dicionário fixo¸ considerando a excitação do dicionário adaptativo escolhida, bem como seu ganho.

Se usarmos notação vetorial, teremos um vetor t gerado pelos vetores x_{aD} e x_{fI} , e pelos ganhos G_a e G_f, dado por:

$$t = G_a H_w x_{aD} + G_f H_w x_{fi} \tag{2.10}$$

onde H_w é uma matriz Toeplitz triangular inferior cujos elementos são amostras da resposta ao impulso do filtro $H_w(z)$. Os ganhos dos dicionários adaptativo e fixo são calculados da seguinte maneira:

$$G_a = \frac{t^T Y_a}{Y_{aD}^T Y_{aD}} \tag{2.11}$$

$$G_f = \frac{t^T Y_{fi}}{Y_{fi}^T Y_{fi}} \tag{2.12}$$

onde y_{aD} e y_{fI} são, respectivamente, as respostas de $H_w(z)$ à entrada x_{aD} e x_{fI}

Foram realizadas quantizações não-uniformes para os ganhos dos dicionários, sendo estes quantizadores obtidos através de um treinamento pelo algoritmo de Lloyd[7] de uma determinada base de dados.

Tabela 2.2 :Alocação de bits.

Parâmetro	Faixa	No. de bits
Ganho do dicionário fixo (Gf)	-0.05 a 0.05	6
Índice do dicionário fixo (I)	0 a 511	9
Ganho do dicionário adaptativo (Ga)	0 a 2	6
Índice do dicionário adaptativo (L)	0 a 511	9
Total		30 bits

2.6.1 O Filtro W(z)

Este filtro tem a finalidade de modificar o espectro de freqüência do sinal de erro e(n) entre o sinal original e o reconstruído. Este sinal de erro possui componentes de freqüências uniformemente distribuídas por todo o espectro. Já o sinal de fala apresenta maior energia principalmente nas baixas freqüências. O objetivo funcional do filtro W(z) é de modificar o espectro de freqüências do sinal de erro de tal forma que ele acompanhe o formato do espectro de sinal de fala, minimizando o erro.

$$W(z) = \frac{A(z)}{A(z/\gamma)} \tag{2.13}$$

$$\gamma = 0.8 \tag{2.14}$$

2.7 Dicionário Fixo

O dicionário fixo é constituído por seqüências de um processo estocástico gaussiano com média zero. Ele pode ser representado por $C_f = \{\{X_{f0}(n)\}, \{X_{f1}(n)\}, ..., \{X_{fK_{f-1}}(n)\}\}$, armazenando K_f seqüências $X_{fI(n)}$, onde I indica o índice das seqüências. Tais seqüências são utilizadas para reconstruir várias versões do mesmo sub-bloco do sinal de fala, sendo escolhida a que proporcionar melhor resultado quantitativo.

São muitas as técnicas de projeto de dicionários fixos. O dicionário aqui utilizado foi gerado no Matlab com o comando *rand*, anulando-se todas as amostras que estivessem abaixo do limiar de 1,645. O resultado final foi um dicionário esparso onde 90% das amostras são nulas. Esta ceifagem proporciona qualidade e busca mais rápida, pois diminui a complexidade computacional. A Figura 2.4 ilustra a estrutura deste dicionário. São K seqüências $X_{fI(n)}$ armazendas.

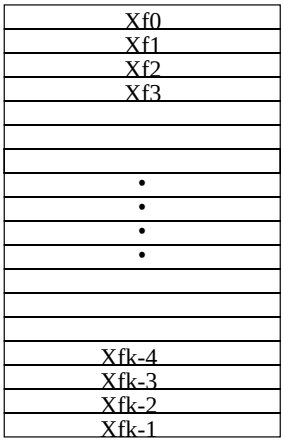


Figura 2.4: Estrutura do dicionário fixo.

No Capítulo 4 é vista uma outra técnica de construção de dicionário fixo: o dicionário treinado. Suas seqüências são o resultado de uma seleção qualitativa de certas bases de dados, dicionário(s) maior(es) de onde são extraídas as melhores seqüências. O intuito neste caso é diminuir a complexidade computacional e aumentar a qualidade do sinal.

2.8 Dicionário Adaptativo

O dicionário adaptativo, da mesma forma que o dicionário fixo, pode ser representado por $C_f = \{\{X_{f0}(n)\}, \{X_{f1}(n)\}, ..., \{X_{fK_{f-1}}(n)\}\}$, armazenando K_f seqüências $X_{fI(n)}$. Só que suas amostras são renovadas constantemente. Suas amostras são utilizadas para recontruir várias versões do mesmo sub-bloco do sinal de fala, sendo utilizada aquela que proporcionar melhor resultado qualitativo.

O dicionário adptativo utilizado neste trabalho contém atrasos fracionários, cobrindo a faixa de 20 a 147 amostras com diferentes resoluções. São utilizados um total de 511 atrasos fracionários, como ilustrado na Tabela 4.3.

Tabela 4.3: Características do Dicionário adaptativo

Faixa (atrasos)	Faixa(Hz)	Resolução(amostras)	No. atrasos
20 a 55	400 a 145	1/8	281
55 a 101	145 a 79	1/4	184
101 a 147	79 a 54	1	46
Total de atrasos	511		

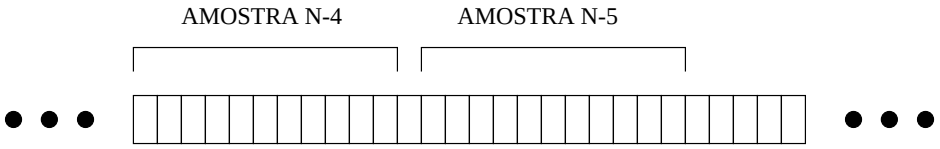


Figura 2.6: Estrutura do Dicionário Adaptativo

A resposta do filtro de síntese perceptual $H_w(z)$ em relação a seqüência candidata x_{aD} é correlacionada com o sinal alvo t(n). Tal correlação é dada por:

$$C = \sum_{n=0}^{N-1} Y_{aD}(n)t(n) \tag{2.15}$$

onde N é o número de amostras das seqüências y_{aD}(n) e t(n), e corresponde ao tamanho de cada subbloco em amostras. As seqüências candidatas de C>0 são levadas em conta. Caso C<0, para todas as seqüências candidatas de um determinado sub-bloco, o índice D é ajustado para 0 para indicar que naquele sub-bloco a excitação será apenas formada pelo dicionário fixo. Outra ponto a ser observado é que o ganho G_a assume somente valores positivos.

A obtenção de cada seqüência candidata x_{aD}(n) é feita filtrando o vetor de amostras das melhores excitações passadas, do qual é composto o dicionário adaptativo, pelas componentes polifásicas dos filtros interpoladores.

2.9 Dados Transmitidos

A taxa de bits do sistema aqui implementado tem seu valor aproximadamente igual a 8Kbps¸como pode ser analisado na tabela abaixo. Os dados referentes ao sinal de excitação transmitidos ao decodificador são descritos na tabela 2.4. Nesta tabela também temos a faixa de valores assumidos e o número de bits alocados para cada parâmetro. Outro fato importante a ser lembrado é que os dados transmitidos ao decodificador são guardados em arquivos binários, o que vai ser melhor explicado no Capítulo 4¸ que apresenta a descrição detalhada do algoritmo do sistema aqui estudado. Estes dados são transmitidos ao decodificador, que reconstrói o sinal de voz original.

Na tabela 2.5 temos a configuração do codificador CELP utilizado.

Tabela 2.4: Distribuição de bits.

Parâmetro	Faixa	No. de bits
Ganho do dicionário fixo	-0,05 a 0,05	6
Índice do dicionário fixo	0 a 511	9
Ganho do dicionário adaptativo	0 a 2	6
Índice do dicionário adaptativo	0 a 511	9
Total	30 bits	

Tabela 2.5: Configuração do Codificador

Tamanho de cada bloco	20 ms
Tamanho de cada sub-bloco	5 ms
Análise LPC:	Método de autocorrelação com janela de Hamming de 25ms (200 amostras), centrada no último sub-bloco de cada bloco.
Número CPL:	10 (p=10)
Quantização dos CPL:	Quantizador QDLSF-32
Interpolação dos CPL:	Linear, no domínio das LSF
Dicionário Adptativo:	Atrasos fracionários de 20 a 147 com diferentes resoluções em determinadas faixas
Dicionário Fixo:	512 seqüências geradas no Matlab
Quantização de G_a	Não uniforme de 0 a 2
Quantização de Gf	Não uniforme de -0,05 a 0,05
Parâmetro perceptual de W(z):	$\delta = 0,8$
Taxa de bits:	CPL: 40 bits/20ms =2 Kbps Excitação:30 bits/5 ms =6 Kbps Totalizando:8 Kbps

2.10 Conclusão

O sistema CELP aqui apresentado opera a uma taxa de 8 kbps. Suas principais características são a utilização de dois dicionários (fixo e adaptativo com atrasos fracionários), bem como a quantização dos parâmetros do filtro de síntese, LSF diferenciais e dos ganhos dos dicionários.

Capítulo 3

Descrição do Algoritmo do Celp

3.1 Introdução

O objetivo deste capítulo é descrever o algoritmo utilizado na implementação de um codificador CELP a uma taxa de 8 kbps. O sistema codifica o sinal de fala em blocos de 20 ms, que são posteriormente divididos em sub-blocos de 5 ms. A análise LPC é feita a cada 20 ms (para cada bloco), enquanto a determinação da melhor excitação do filtro de síntese é feita a cada 5 ms para cada sub-bloco.

O código do programa é escrito em C, o que dá um bom grau de portatibilidade do sistema. Outro aspecto importante é a sua velocidade de processamento¸ indispensável para a codificação de voz em tempo real.

Na Seção 3.2 é mostrada uma forma simplificada do algoritmo do programa. Já as Seções 3.3 e 3.4 tratam o mesmo algoritmo de forma mais detalhada¸ apresentando passo a passo o seu desenvolvimento.

3.2 Algoritmo do Codificador CELP

O algoritmo do codificador CELP pode ser entendido como sendo dividido em cinco etapas, descritas a seguir:

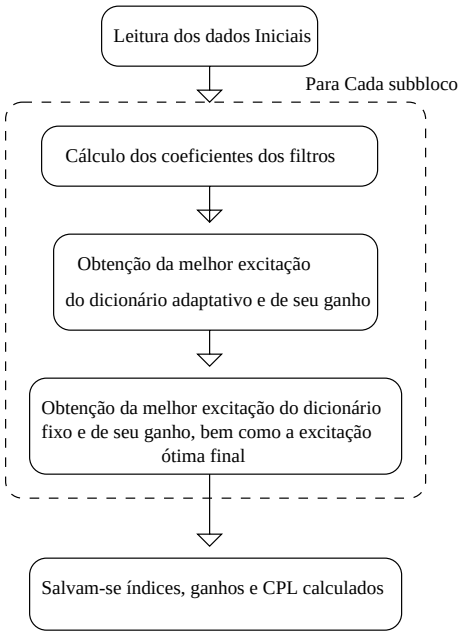


Figura 3.1: Algoritmo do decodificador.

3.2.1 Leitura dos Dados Iniciais

Nesta etapa são realizadas as leituras do sinal de voz a ser codificado, das seqüências do dicionário fixo e das componentes do filtro polifásico. Estes valores são guardados em vetores e lidos somente uma vez. Também é feita alocação de memória e inicializa-se (para zero) a resposta a entrada zero e os parâmetros do filtro e do dicionário adaptativo. A janela de Hamming também é calculada nesta etapa.

As etapas descritas nos subitens 3.2.2, 3.2.3 e 3.2.4 ilustram a análise por síntese. Estes procedimentos são realizados para cada sub-bloco.

3.2.2 Cálculo dos Coeficientes dos Filtros.

Primeiro armazenam-se os coeficientes LAR do segmento anterior. Depois, segmenta-se o bloco a ser analisado multiplicando-o pela janela de Hamming calculada anteriormente.

A determinação dos coeficientes CPL, a conversão dos coeficientes preditores em LSF e a quantização diferencial são realizadas pelas funções analpc, a2lsp e quant_lsf, respectivamente. As DLSF calculadas são guardadas em arquivos de dados, pois serão transmitidas para o decodificador.

Realiza-se, então, a interpolação pela função interpola. A transformação das LSF quantizadas em coeficientes preditores é feita através da função lsf2a.

Já para o cálculo dos coeficientes do filtro de ponderação W(z), utilizam-se as seguintes expressões:

$$num_{fpeso}[j] = -coef_a[j-1] \tag{3.1}$$

$$den_{fpeso}[j] = pow(gamma, j) * num_{fpeso}[j] \tag{3.2}$$

A equação (3.1) representa o cálculo do numerador do filtro W(z), onde $coef_a$ são as LSF quantizadas em coeficientes preditores. Já a equação (3.2) é o cálculo do denominador, que irá ser igual ao numerador multiplicado por um fator de ponderação (no programa em questão, pow (a,b) é uma função em C que retorna b^a).

3.2.3 Busca no Dicionário Adaptativo

Depois de serem calculados os coeficientes dos filtros, obtém-se o sub-bloco de fala original, que é filtrado através da função filt_pz. A resposta do filtro de síntese à entrada zero é calculada pela função filt_sp1, e obtém-se o sinal alvo, ou seja, o resultado da subtração entre o sub-bloco original filtrado e da resposta à entrada zero.

Em seguida, utiliza-se a função le_seq, que fornece a seqüência candidata com atrasos fracionários. Para os índices do dicionário adaptativo compreendidos entre 1 e 281 são usadas as componentes polifásicas e4; caso o índice seja maior que 281 são usadas as componentes e8. O índice também determina qual o atraso a ser utilizado.

Utilizando a seqüência candidata, obtém-se o sinal reconstruído através da função filt_sp2. Calcula-se em seguida o termo que minimiza o erro médio quadrático, guardando-se sempre o índice da seqüência responsável pelo menor erro quadrático. É importante lembrar que este procedimento é feito para todas as seqüências candidatas e para cada sub-bloco.

Escolhida a melhor seqüência do dicionário adaptativo e tendo calculado sua resposta, determina-se o seu ganho, através da seguinte equação:

$$G_f = \frac{prod_{int}(sinal_{alvo}, sinal_{reconstrudo}, tamanho_{sub-blco}}{prod_{int}(sinal_{reconstruido}, sinal_{reconstruido}, tamanho_{sub-bloco}} \tag{3.3}$$

A quantização do ganho do dicionários adaptativo é realizada pela função quant_esc. Finalmente obtém-se o sinal alvo parcial multiplicando-se o sinal reconstruído (resultado de filt_sp2) com o seu respectivo ganho.

3.2.4 Busca no Dicionário Fixo

Para cada seqüência candidata do dicionário fixo, calcula-se a resposta desta excitação utilizando-se a função filt_sp2 . Depois realizam-se os mesmos procedimentos de cálculo do menor erro quadrático, computação do índice da respectiva excitação (responsável pelo menor erro) e cálculo do ganho da resposta realizados no dicionário adaptativo. A quantização do ganho também é feita de forma semelhante, como indicado pela equação (3.3).

Em seguida, calcula-se a excitação ótima completa¸ que será a soma da seqüência do dicionário adaptativo com a excitação do dicionário fixo multiplicadas pelos seus respectivos ganhos.

A atualização do dicionário adaptativo com a excitação ótima completa é feita nesta etapa. Este procedimento é realizado para cada bloco, e no final da análise, são salvos os índices e ganhos dos dicionários, bem como os CPL, todos em arquivos binários.

3.3 Diagrama de Blocos Detalhado

A seguir tem-se o diagrama de blocos detalhado do sistema implementado.

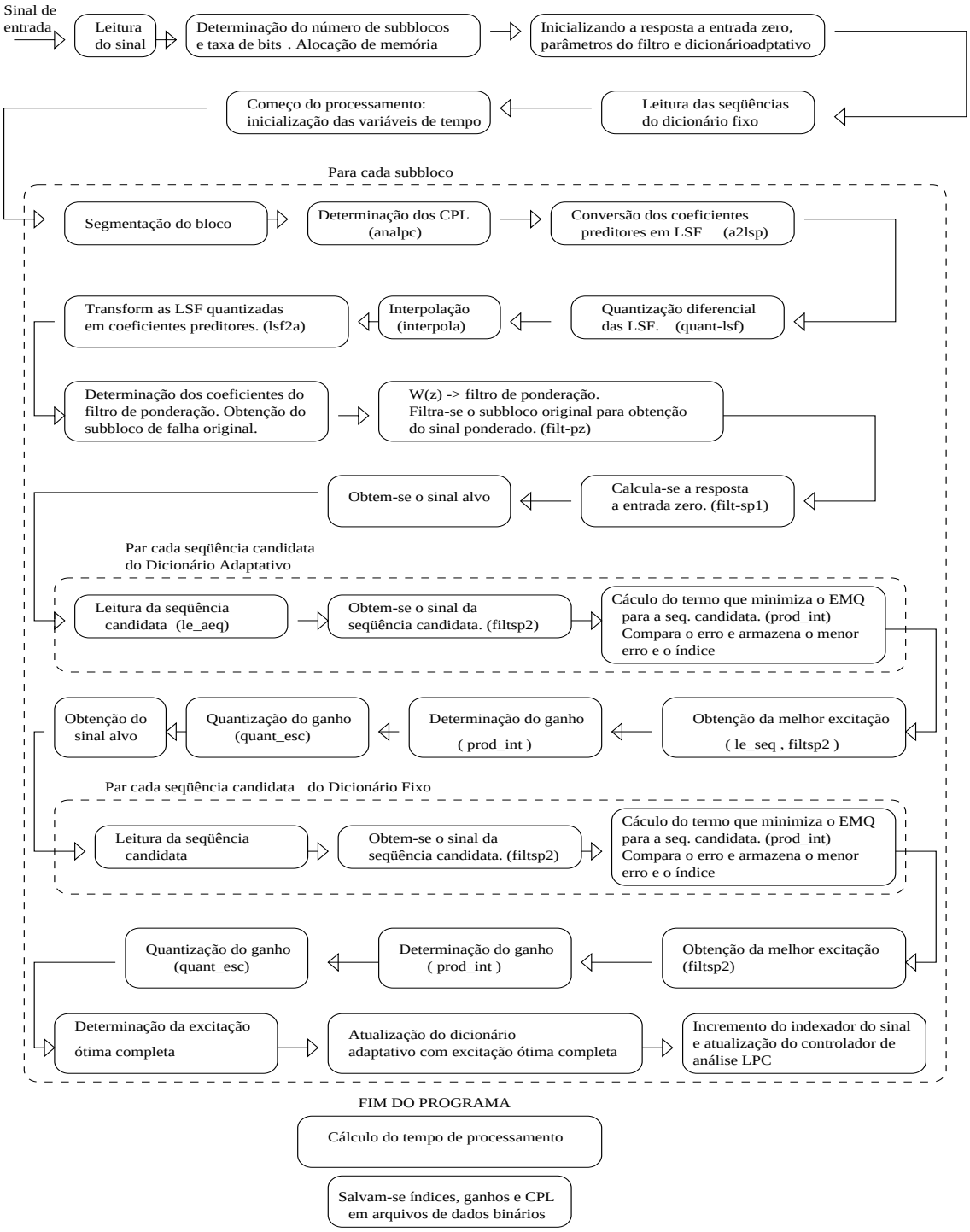


Figura 3.2: Diagrama de Blocos detalhado.

3.4 Documentação do Programa

A seguir tem-se a descrição detalhada de cabeçalhos e funções utilizadas no programa.

3.4.1 Cabeçalho

São criados dois arquivos de cabeçalho: quantiza.h e dados.h. Neles são definidas algumas macros utilizadas no programa.

- quantiza.h

Este arquivo de cabeçalho contém as definições das partições e dicionários para a quantização escalar diferencial das freqüências do espectro de linha e quantização escalar dos ganhos dos dicionários adaptativo e fixo. Esta base de dados foi obtida ao codificar 11 sinais de fala, frases foneticamente balanceadas de dois locutores do sexo masculino e dois do sexo feminino.

- dados.h

  Este arquivo de cabeçalho define valores de dados do programa. São definidos:

  - *tam_bloco* (duração de cada bloco de analise LPC).
  - *tam_sbl* (duração de cada bloco de excitação).
  - *tam_jan* (duração da janela para análise LPC).
  - *num_cpl* (número de coeficientes de predição linear).
  - *tam_df* (número de seqüências do dicionário fixo).
  - *tam_da* (número de seqüências geradas no dicionário adaptativo).
  - *nam_da* (número de amostras do dicionário adaptativo).
  - *gamma* (fator perceptual).

3.4.2 Funções

- auread

  Esta função tem como objetivo a leitura do sinal de voz no formato ".au" . Posui os seguuintes parâmetros :

  - *arquivo[ ]* (é o arquivo de extensão .au a ser lido).
  - *num_amostras* (número de amostras).
  - *swap* .

  Ela também utiliza outras duas funções auxiliares:

  - *troca_bytes_long* (sua finalidade é realizar a mudança de ordem dos bytes de uma variável do tipo "long").
  - *mu2lin* (sua finalidade é transformar o sinal quantizado com a lei u para um com a quantização linear).

  Primeiramente faz-se a leitura do cabeçalho do arquivo a ser lido utilizando-se o comando fread. A seguir, testa-se o formato lei-u e o flag *swap*. Após a leitura correta, utiliza-se a função mu2lin já mencionada e retorna-se o sinal lido (sinal_li) e o número de amostras (*num_amostras).

- blocos

  O objetivo desta função é determinar o número de sub-blocos do sinal de fala, bem como preencher com zeros o bloco restante. Esta função possui três parâmetros:

  - *sinal[ ]* (sinal de fala (resultado da leitura em auread)).
  - *num_am* (número de amostras).
  - *blo_sbl* (número de segmentos de excitação contidos em cada segmento).

  Primeiro determina-se o número de blocos e sub-blocos, para depois preencher com zeros as amostras que restam, totalizando outro bloco. Finalmente é retornado o número de sub-blocos.

- taxa_bits

  Esta função determina a taxa de bits do codificador. Seus parâmetros são:

  - *nbits_lsf* (número de bits para coeficientes lsf).
  - *nbits_alfa* (número de bits para alfa).
  - *nbits_beta* (número de bits para beta).

  O cálculo da taxa de bits é feito considerando-se o número de bits utilizados nos dicionários adaptativo e fixo, em seus respectivos ganhos e nos coeficientes de reflexão da análise LPC.

- entrada

  A função entrada tem a finalidade de ler um arquivo de dados e carregar a variável especificada. Tem como parâmetros:

  - *arquivo[ ]* (arquivo de dados a ser lido).
  - *variável* (variável que irá ser carregada).
  - *num_item* (número de itens a serem lidos).

- hamming

  A função hamming, como o nome sugere, gera uma janela de Hamming com o tamanho especificado em seus parâmetros, que são:

  - *janela[ ]* (janela que irá ser retornada).
  - *tamanho* (tamanho especificado da janela).

- analpc

  O objetivo desta função é determinar os CR e os ganhos do modelo LPC para cada bloco do sinal de fala usando o método recursivo de Levinson-Durbin. Seus parâmetros são:

  - *sinal [ ]* (sinal a ser analisado).
  - *a[ ]* (CRs obtidos (retorno da função).
  - *tamanho* (tamanho da janela de hamming, ou seja, duração da janela para análise LPC).
  - *num_cpl* (número de coeficientes de predição linear).

  Primeiro são feitas as alocações dinâmicas de memória. Depois determina-se a seqüência de autocorrelação para o bloco de voz, para enfim calcular os CR usando o algoritmo de Levinson-Durbin.

- a2lsp

  Esta função visa a conversão dos coeficientes do filtro em coeficientes LSP. Seus parâmetros são:

  - *coef_a* (coeficientes obtidos em ANALPC) .
  - *coef_lsf_c* (coeficientes LSF calculados que são retorno da função).
  - *num_cpl* (número de coeficientes de predição linear).

- quant_lsf

  O objetivo desta função é a realização da quantização diferencial escalar das LSF, de acordo com os vetores de partição e codebook para cada elemento. Utiliza a função quant_esc, cuja finalidade é quantizar um determinado escalar, possuindo os seguintes parâmetros:

  - *x* (variável de comparação).
  - *partição* (partição a ser usada).
  - *codebook [ ]* (codebook a ser usado).
  - *tam_codebook* (tamanho do codebook).

  Já a função quant_lsf possui os seguintes parâmetros:

  - *coef_lsf_c* (coeficientes obtidos em a2lsp).
  - *coef_lsf_cq* (coeficientes da quantização diferencial a serem obtidos).
  - *coef_dlsf_cq* (coeficientes da quantização diferencial a serem obtidos).

- interpola

  Como o próprio nome sugere, o objetivo desta função é realizar a interpolação das LSF. O vetor de interpolação é determinado de acordo com o tamanho do bloco, sendo aceitos 4 ou 5 subblocos. Os parâmetros desta função são:

  - *coef_lsf_a* (coeficientes anteriores).
  - *coef_lsf_cq* (coeficientes correntes).
  - *coef_lsf* (coeficientes interpolados que são retorno da função).
  - *cont_lpc* (bloco a ser analisado).
  - *blo_sbl* (número de segmentos de excitação contidos em cada segmento).
  - *num_cpl* (número de coeficientes de predição linear).

- lsf2a

  A finalidade desta função é realizar a conversão das freqüências do espectro de linha para coeficientes preditores. Seus parâmetros são:

  - *coef_lsf* (coeficientes interpolados¸ que são resultado da função interpola).
  - *coef_a* (coeficientes preditores obtidos).
  - *num_cpl* (número de coeficientes de predição linear).

  Esta função utiliza a função polimulti, cuja finalidade é a multiplicação de dois polinômios, sendo seus parâmetros:

  - *poli1[ ]* (polinômio 1).
  - *poli2[ ]* (polinômio 2).
  - *polir[ ]* (resultado da operação efetuada).

- *ordem 1* (ordem do polinômio 1).
- *ordem 2* (ordem do polinômio 2).

- filt_pz

  Esta função tem como objetivo passar um sinal por um filtro digital com pólos e zeros, considerando os estados iniciais. Com ela, filtra-se o sub-bloco original para ser obtido o sub-bloco pesado. Seus parâmetros são:

  - *subloco* (subbloco a ser filtrado).
  - *subloco_p* (subbloco filtrado).
  - *num_fpeso* (numerador do filtro de ponderação).
  - *den_fpeso* (denominador do filtro de ponderação).
  - *est_nfpeso* (estado inicial numerador).
  - *est_dfpeso* (estado inicial denominador).
  - *tam_sbl* (tamanho do subbloco).
  - *num_cpl* (número de coeficientes preditores).

- filt_sp1

  A função filt_sp1 visa determinar a resposta de um sinal que passa por um filtro digital com somente pólos, considerando seu estado inicial. Seus parâmetros são:

  - *ent_zero* (resposta a entrada zero).
  - *s0_p* (estado inicial).
  - *den_fpeso* (denominador do filtro de peso).
  - *est_fsint* (1).
  - *tam_sbl* (tamanho do subbloco).
  - *num_cpl* (número de coeficientes preditores).

- le_seq

  Esta função realiza a leitura de uma seqüência candidata no dicionário adaptativo com atrasos fracionários. Seus parâmetros são:

  - *dicio_adp* (dicionário adaptativo).
  - *x_a* (seqüência lida, que é seu resultado).
  - *j* (índice do dicionário adaptativo).
  - *nam_da* (número de amostras do dicionário adaptativo).
  - *tam_sbl* (tamanho do subbloco).
  - *e4 ou e8* (componentes polifásicas do filtro interpolador do dicionário adaptativo fracionário).

- filt_sp2

  A função filt_sp2 visa determinar a resposta de um sinal que passa por um filtro digital que possui somente pólos, mas não sendo considerado seu estado inicial. Este sinal é a seqüência candidata lida em le_seq. Seus parâmetros são:

  - *x_a* (seqüência (sinal) a ser filtrado).
  - *sr_p* (sinal filtrado obtido que é seu resultado).
  - *den_fpeso* (denominador do filtro de peso).

- *tam_sbl* (tamanho do subbloco).
- *num_cpl* (número de coeficientes preditores).

- prod_int

  Esta função tem o objetivo de realizar o produto interno entre dois vetores, retornando este valor. É utilizada na determinação dos ganhos dos dicionários e no cálculo do erro médio quadrático do sinal obtido. Seus parâmetros são :

  - *vetor 1* .
  - *vetor 2* .
  - *tamanho* (tamanho dos vetores).

- saida1 e saida2

  O objetivo destas duas funções é o mesmo: salvar dados em arquivo binário. A diferença é que saida1 trata de dados do tipo "unsigned short", e a outra função "float". Seus parâmetros são:

  - *arquivo* (arquivo onde serão gravados os dados).
  - *variável* (variável a ser gravada).
  - *num_itens* (número de itens a serem gravados).

3.5 Conclusão

Este capítulo apresentou a descrição detalhada do algoritmo utilizado na implementação de um codificador CELP a uma taxa de 8 kbps. Foram explicadas minuciosamente cada etapa e função do programa, no intuito de elucidar a funcionalidade do mesmo. Foi incluído um fluxograma detalhado do funcionamento do sistema.

Capítulo 4

Testes de Performance

4.1 Introdução

Como vimos, é crescente e contínua a necessidade de se obter codificadores que proporcionem boa relação entre qualidade e taxa de bits. Isto porque aplicações como teleconferência, telefonia móvel e internet exigem cada vez mais qualidade do sinal, com menor consumo de energia e tempo.

Este capítulo tem a finalidade de proporcionar um estudo da evolução qualitativa do codificador em questão, procurando mostrar as características e influências práticas de cada elemento que o compõe. São realizadas simulações para várias configurações do codificador, observando-se suas performances, principalmente no que diz respeito a relação entre qualidade e taxa de bits.

Os primeiros testes são em relação aos dicionários fixos e adaptativos. São simulações onde são realizadas diferentes configurações dos dicionários: apenas um dicionário fixo, dois dicionários fixos, um dicionário fixo e um dicionário adpatativo não fracionário e finalmente um dicionário fixo com um dicionário adptativo fracionário. Em seguida, temos na Seção 4.3, testes referentes a suavização espectral, analisando-se os espectros de frequências e a qualidade subjetiva dos sinais obtidos. Por último, na Seção 4.4, é mostrado um estudo de treinamento do dicionário fixo, visando obter melhor aproveitamento das amostras e maior qualidade dos sinais reconstruídos.

4.2 Configurações e Objetivos

Foram realizados quatro diferentes testes para observar o comportamento do codificador CELP em relação a sua evolução qualitativa, no que diz respeito a qualidade do sinal obtido, taxa de bits e tempo de processamento. O intuito das primeiras simulações é estudar a influência direta dos dicionários fixo e adptativo, observando-se os seguintes parâmetros de medidas:

- Tamanho de dicionário

Na técnica de codificação CELP utilizamos dois diferentes tipos de dicionários; o adaptativo e o fixo. Como já explicado no Capítulo 2, os dicionários fixos são compostos por um conjunto de seqüências de ruído branco gaussiano de média zero e variância unitária. Essas seqüências passaram por uma ceifagem, no intuito de aumentar a quantidade de elementos nulos, o que proporciona menor complexidade computacional para o codificador, além de aumentar qualitativamente o sinal obtido. As seqüências utilizadas nestas simulações foram geradas no Matlab, através do programa descrito na Tabela 4.1.

Tabela 4.1: Geração do dicionário fixo.

```
N=40
M=512
DF=zeros(1,M*N)
for i = 1:M
DFR((40*(i-1) + 1) = randn(1,N);
for j=1:N
if abs(DFR(40*(i-1)+j))|1.645
DFR(40*(i-1)+j)=0;
end;
end;
end;
```

Para os testes referentes aos dicionários, foram utilizados dicionários fixos de tamanhos variados com 64, 128, 256, 512, 1024 amostras. Já o dicionário adaptativo é formado pelas excitações escolhidas durante o processo de análise por síntese. Primeiro é utilizado um dicionário adaptativo não-fracionário, de tamanho igual a 512. Depois é simulado um codificador com o diocionário fracionário de tamanho 512, e compara-se o resultado obtido com as configurações anteriores. As Tabelas 4.2 e 4.3 mostram as configurações de dicionários utilizadas nos testes deste capítulo.

Tabela 4.2: Testes com os dicionários.

	Primeiro teste	**Segundo teste**	**Terceiro teste**	**Quarto teste**
Dicionário Fixo	Apenas um	Dois	Um de 512	Um de 512
Dicionário Adaptativo	Não utilizou	Não utilizou	Um com atrasos não-frac.	Um com atrasos frac.

Tabela 4.3: Outros Testes.

	Suavização Espectral	**Treinamento do DF**
Dicionário Fixo	Um de 512	Um de 512
Dicionário Adaptativo	Um com atrasos frac.	Um com atrasos frac.

- RSRSP (dB): a razão sinal-ruído segmentada perceptual (RSRSP) é calculada dividindo-se o sinal em M segmentos iguais de comprimento N, calculando assim a RSRSP para cada um deles. O valor final será a média aritmética de todas as medidas realizadas, isto é

$$RSR_{SP} = \frac{1}{M} \sum 10log \left\{ \frac{\sum_{n=0}^{n-1} v^2(n+iN)}{\sum_{n=0}^{n-1} e_w^2(n+iN)} (dB) \right. \tag{4.1}$$

Nas simulações aqui realizadas foi utilizado N =80, e a análise LPC foi feita pelo método da auto-correlação de ordem 10 com uma janela de Hamming de 25 ms centrada em cada segmento de 10 ms correspondente. Esta medida é usada para compararmos quantitativamente os resultados obtidos.

- Tempo de execução (s): o tempo de execução do programa é calculado como visto no Capítulo 3. Este parâmetro é muito importante, pois com ele pode-se comparar a variação da complexidade computacional em relação às várias configurações de dicionáios utilizadas nas simulações.
- Taxa de bits (kbps): a taxa de bits é outro fator muito importante para nível de comparação de performance. O seu cálculo é feito considerando-se o número de bits utilizados nos dicionários adaptativo e fixo, em seus respectivos ganhos e nos coeficientes de reflexão da análise LPC.

- Sinais de fala obtidos: para validar as experiências, não foram quantizados os ganhos dos dicionários. O sinal utilizado no teste foi uma frase foneticamente balanceada, pronunciada por um locutor do sexo feminino. Este sinal possui frequência de 8 kHz com 16 bits por amostra. Os sinais decodificados obtidos nas simulações são utilizados para análise subjetiva de qualidade.

- Configuração da Máquina: os testes foram realizados em uma máquina com 128 MB de memória, processador K6 II 450 e placa de som Sound Blaster. Para aumentar a validação das medições, só o processo do codificador era rodado na máquina.

4.2.1 Dicionário Fixo de Vários Tamanhos

Neste teste realizaram-se simulações com apenas um dicionário fixo de tamanhos diversos. A configuração do codificador pode ser observada na figura 4.1.

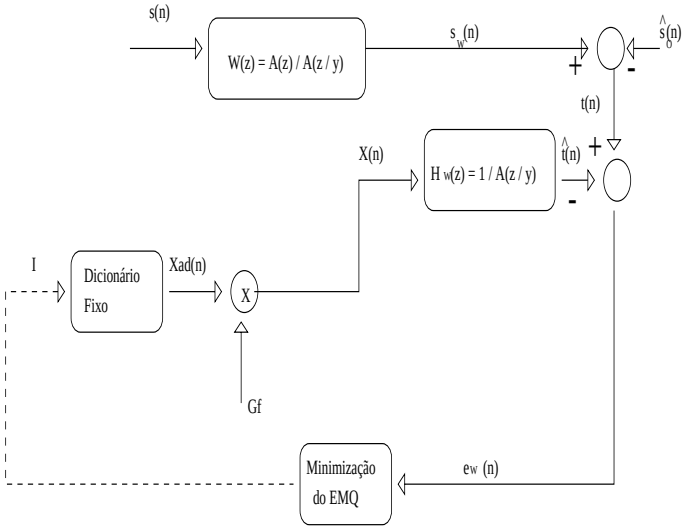


Figura 4.1: Diagrama de Blocos 1 DF

Os resultados obtidos para esta configuração podem ser vistos na Tabela 4.4. À medida em que se aumentou o número de amostras, tivemos um aumento da complexidade computacional. Em relação a qualidade do sinal, houve uma certa melhora, que ainda está longe dos patamares almejados.

Tabela 4.4: Resultados para 1 DF.

Tam. do D.F.	RSRSP (dB)	MSE (dB)	t (s)	Taxa (kpbs)	Arquivo de som
64	13.2600	19.6571	1	4.2	teste-1_64-f1.au
128	13.8462	19.1771	3	4.4	teste-1_128-f1.au
256	14.1177	18.7632	6	4.6	teste-1_256-f1.au
512	14.5838	18.3621	11	4.8	teste-1_512-f1.au
1024	14.8047	17.9785	22	5.0	teste-1_1024_-f1.au

Conclui-se que com o aumento do tamanho do dicionário, a qualidade do sinal reconstruído também aumenta consideravelmente, ou seja, a qualidade do sistema nesta configuração está muito relacionada ao tamanho do dicionário usado. Em contrapartida, o tempo de execução e a taxa de bits também têm seus valores acrescidos. Aqui constata-se que para obter melhor qualidade nesta configuração, é necessário utilizar dicionários grandes, elevando-se o tempo de procesamento. O fato observado acima será o principal foco da análise. O objetivo é achar uma situação ótima; com boa qualidade em um tempo

razoável a uma taxa de bits menor possível.

A Figura 4.2 ilustra a análise dos parâmetros medidos com o tamanho do dicionário fixo.

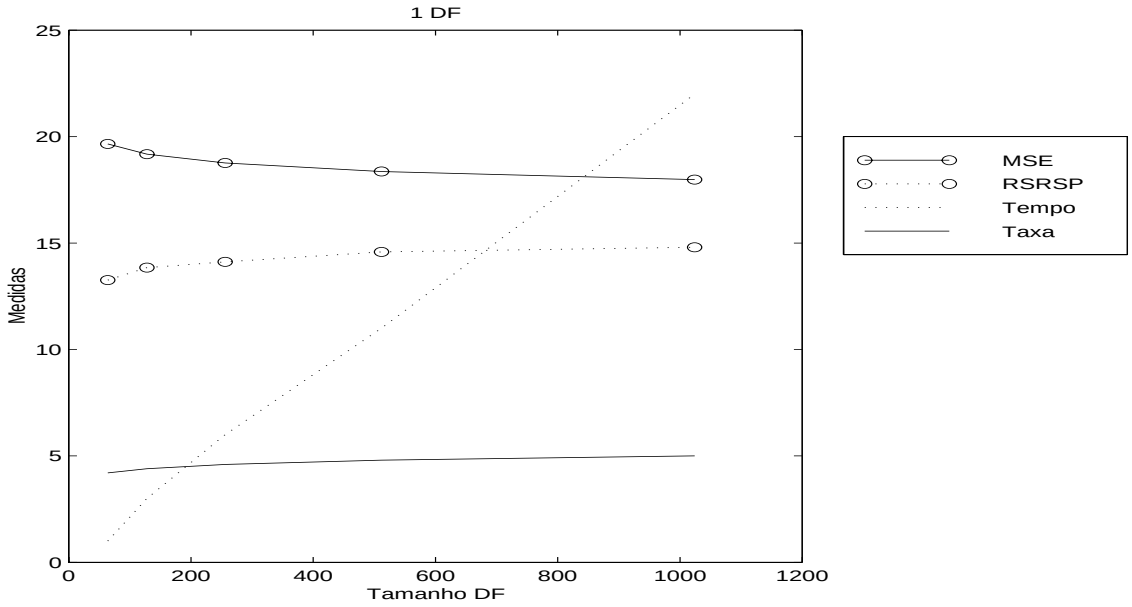


Figura 4.2: Medidas para 1 DF.

O próximo teste a ser realizado é a utilização de dois dicionários fixos de mesmo tamanho. Com mais seqüências de excitação¸ espera-se melhorar a qualidade do sinal obtido.

4.2.2 Dois Dicionários Fixos de Mesmo Tamanho

Este teste trata-se da simulação do codificador sendo composto por dois dicionários fixos de mesmo tamanho, ao invés de um diocionário. A figura 4.3 mostra o diagrama de blocos desta configuração. Novamente¸ foram utilizados dicionários de vários tamanhos com 64, 128, 256, 512 e 1024 amostras.

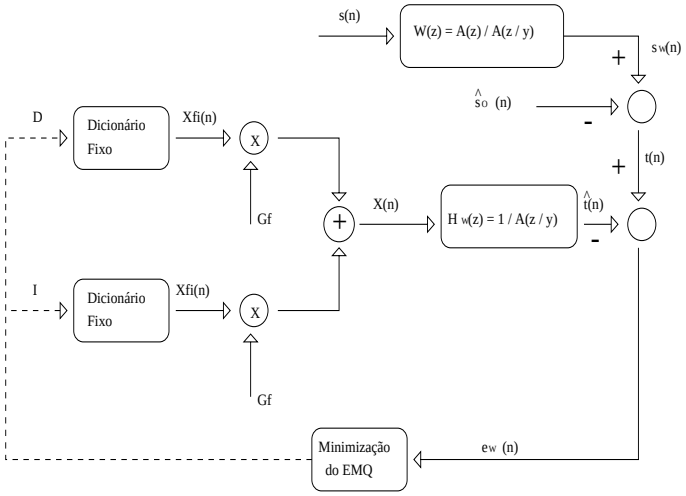


Figura 4.3: Diagrama de Blocos 2DF.

Os resultados obtidos para esta configuração podem ser vistos na Tabela 4.5, onde temos a comparação final dos resultados obtidos para o codificadores que utilizam dois DF, com tamanhos variados. Nota-se que o aumento do número de amostras acarreta o aumento da taxa de bits e da complexidade computacional do sistema, bem como a melhoria na qualidade do sinal obtido.

Tabela 4.5 : Resultados para 2 DF.

Tam. do D.F.	RSRSP (dB)	MSE (dB)	T (s)	Taxa (kbps)	Arquivo de som
64	14.9741	17.7398	3	6.8	teste-2_64-f1.au
128	15.7105	16.8159	5	7.2	teste-2_128-f1.au
256	16.1101	16.2986	11	7.6	teste-2_256-f1.au
512	16.0607	15.7226	22	8.0	teste-2_512-f1.au
1024	16.9748	15.3181	55	8.4	teste-2_1024-f1.au

Como esperado, os resultados indicam melhora qualitativa em relação ao uso de somente um dicionário fixo, isto porque temos X^2 seqüências (ao invés de X seq.). Mas houve um acréscimo no tempo de execução e principalmente na taxa de bits, fator que torna a sua implementação desaconselhável. Ou seja, mais uma vez a questão da qualidade estar relacionada ao tamanho do dicionário utilizado foi constatada.

Em termos gerais, o resultado qualitativo melhorou em relação à primeira configuração testada. E isso não só em números medidos; a medida subjetiva (os arquivos de som gravados) também comprovam esta melhora. Em contrapartida, essa melhora foi acompanhada por um aumento excessivo da complexidade computacional, o que elevou em muito o tempo de execução do programa. Além disso, por serem usados dois dicionários fixos, acarretou também um aumento na taxa de bits.

Temos na Figura 4.4 os resultados obtidos na simulação.

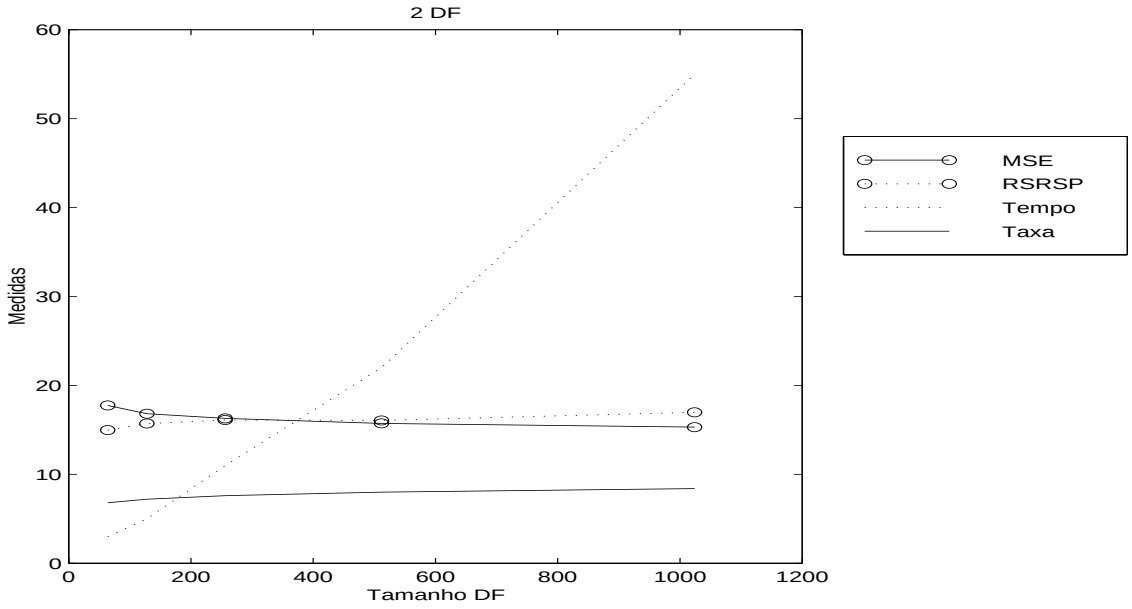


Figura 4.4: Medidas par 2 DF

O próximo teste a ser realizado e a utilização de um dicionário adpatativo com atrasos não-fracionários de tamanho definido igual a 512, combinado com um dicionário fixo com tamanho variável.

4.2.3 Um Dicionário Adaptativo com Atrasos não Fracionários (512 amostras) com Dicionários Fixos de Vários Tamanhos

Neste experimento foram utilizados dicionários fixos de tamanhos variados e um dicionário adptativo sem atrasos fracionários, com faixa de 20 a 146. A introdução deste dicionário implicará numa extraordinária melhora da qualidade do sinal codificado, bem como no aumento da complexidade computacional do sistema, o que acarretará maiores tempos de execução.

Os resultados podem ser vistos na Tabela 4.6. Aqui foram utilizados um dicionário fixo juntamente com um dicionário adaptativo não fracionário. Conceitualmente, o dicionário adptativo representa uma melhor forma de visualizar o filtro de *pitch*. A vantagem de se utilizar um dicionário adptativo é a determinação dos parâmetros de maneira a minimizar o erro perceptual e a possibilidade de otimização dos ganhos dos dois dicionários simultaneamente. O diagrama de blocos desta estrutura é mostrado na figura 4.5.

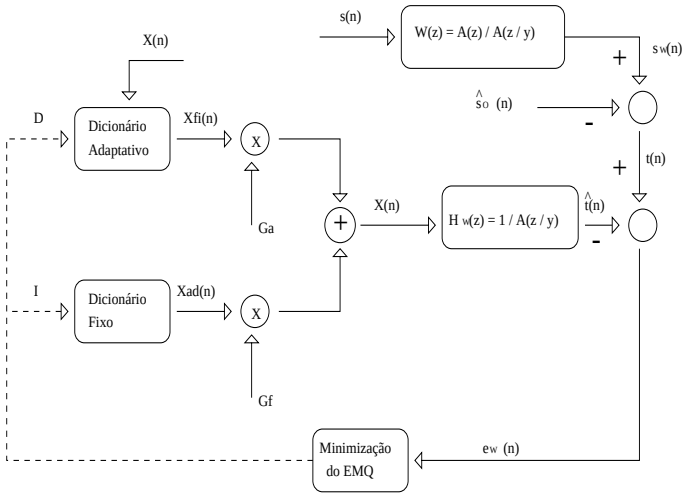


Figura 4.5: Diagrama de Blocos 1DF e 1DA

De acordo com a tabela 4.6¸ nota-se um pulo qualitativo no sinal reconstruído. Em compensação, o dicionário adaptativo proporcionou ao sistema uma elevação de sua complexidade computacional, o que pode ser observado no tempo de execução medido. Como esperado¸ não houve aumento considerável da taxa de bits em relação à configuração anterior (dois dicionários fixos).

Tabela 4.6: Resultados para 1DF e 1 DAnF.

Tam. do D.F.	RSRSP (dB)	MSE (dB)	T (s)	Taxa (kbps)	Arquivo de som
64	17.0224	13.2331	13	7.4	teste-3_64-f1.au
128	17.4213	13.1233	15	7.6	teste-3_128-f1.au
256	17.6021	12.7824	18	7.8	teste-3_256-f1.au
512	17.7603	12.3266	29	8.0	teste-3_512-f1.au
1024	18.0121	11.8733	56	8.2	teste-3_1024-f1.au

O aumento na qualidade do sinal era previsto¸ já que a inclusão do dicionário adpatativo proporciona uma melhor representação do período de pitch. Os arquivos de som gravados também comprovam esta melhora qualitativa obtida.

Um fato importante observado é que com esta configuração¸ pode-se diminuir o tamanho do dicionáio fixo, ou seja, a qualidade do sinal não teve variação muito alta em relação ao número de seqüências deste

dicionário.

A Figura 4.6 ilustra os valores medidos.

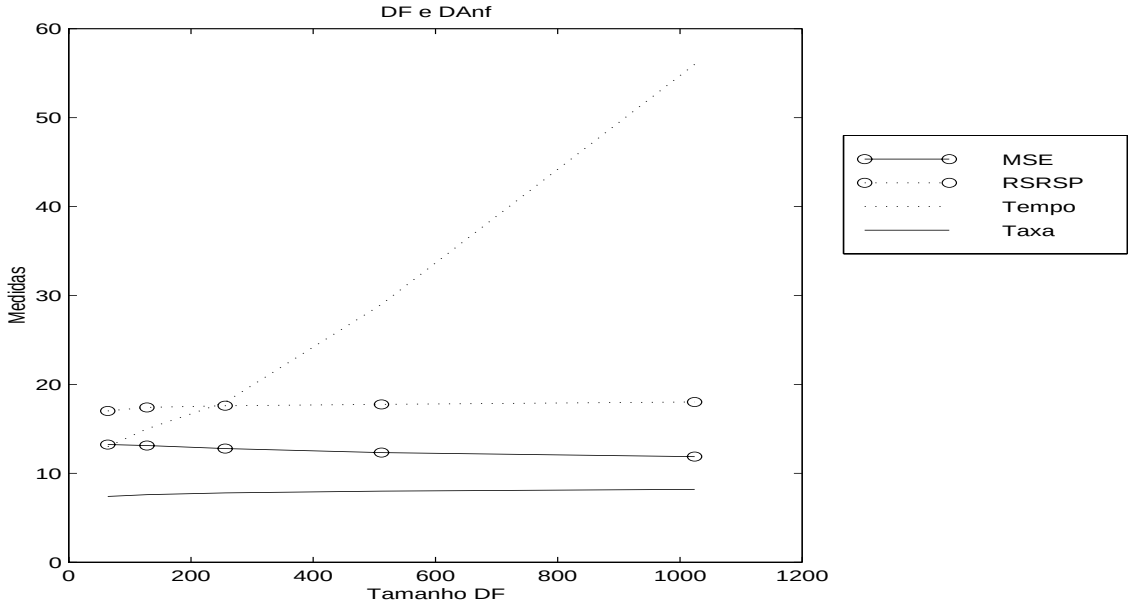


Figura 4.6: Medidas para 1DF e 1 DAnF.

O próximo teste a ser realizado e a utilização de um dicionário adpatativo com atrasos fracionários de tamanho definido igual a 512, combinado com um dicionário fixo com tamanho variável.

4.2.4 Um Dicionário Adaptativo com Atrasos Fracionários (512 amostras) com Dicionários Fixos de Vários Tamanhos

O quarto teste trata-se da implementação de atrasos fracionários ao dicionário adaptativo usado no item anterior, visando melhorar ainda mais a qualidade do sinal final. Isto porque o atraso fracionário permite uma representação mais precisa das amostras passadas. Este aumento qualitativo, no entanto, é acompanhado por um aumento de complexidade computacional, o que representa aumento no tempo de execução do programa. O diagrama de blocos desta configuração é o mesmo do item anterior.

O dicionário adaptativo aqui usado possui atrasos fracionários cujas resoluções foram distribuídas da seguinte forma: oitavas de 20 a 55, quartas de 55 a 101 e unitárias de 101 a 146. A frase utilizada foi a mesma utilizada nos testes anteriores.

Tabela 4.7: Resultados para 1DF e 1DAF.

Tam. do D.F.	RSRSP (dB)	MSE (dB)	T (s)	Taxa (kbps)	Arquivo de som
64	17.5674	12.6687	58	7.4	teste-4_64-f1.au
128	17.7591	12.6845	59	7.6	teste-4_128-f1.au
256	17.9142	12.1919	62	7.8	teste-4_256-f1.au
512	18.1622	11.6409	68	8.0	teste-4_512-f1.au
1024	18.3241	11.2113	83	8.2	teste-4_1024-f1.au

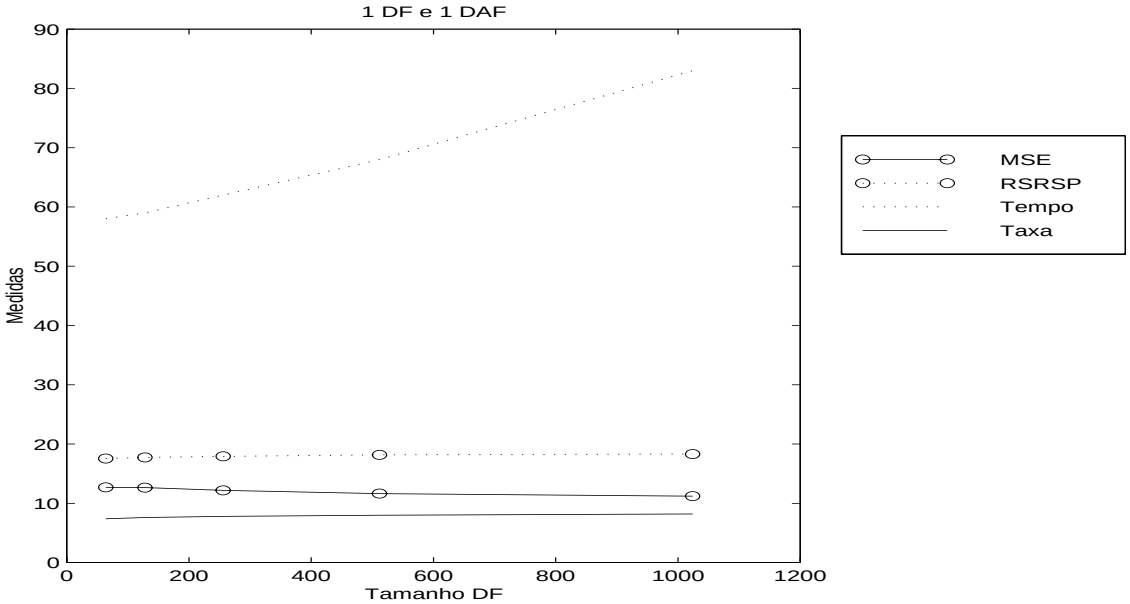


Figura 4.7: Medidas para 1DF e 1DAF.

Partindo para a análise subjetiva da qualidade do sinal obtido, não foram constatadas grandes diferenças, ou seja, para efeito de sensibilidade do ouvido humano, o resultado aqui obtido é comparável qualitativamente a simulação com o dicionário adptativo não-fracionário. Este fato é importante para o próximo passo do projeto, que visa a aceleração do codificador, procurando manter um nível de qualidade. Outro fato também observado (como na configuração anterior), é a possibilidade de se reduzir o número de amostras do dicionário fixo. A figura 4.7 ilustra os resultados aqui obtidos.

4.2.5 Gráficos Comparativos

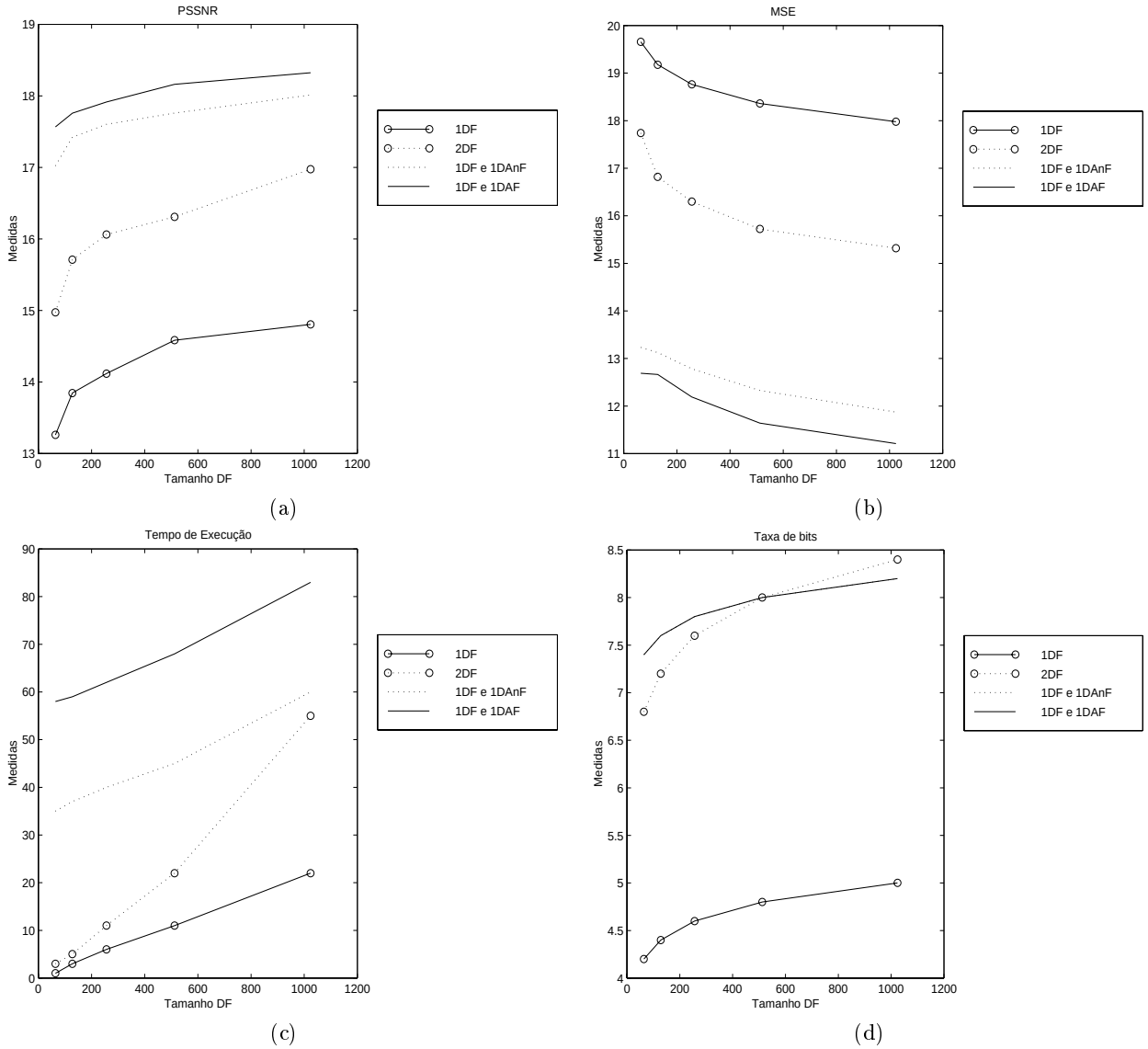


Figura 4.8: a) Comparação de RSR-SP; b) Comparação de MSE;
c) Comparação de Tempo de Execução; d)Comparação de Taxa de Bits.

Na figura 4.8 são mostradas comparações entre os resultados obtidos nas simulações com os dicionários fixo e adaptativo. Pode-se observar a relação entre os fatores *qualidade*¸ *tempo de execução* e *taxa de bits.*

Observando as figuras 4.8a) e 4.8b), que mostram, repectivamente, as medidas de RSRSP e MSE, conclui-se que a diferença qualitativa entre as configurações que utilizam o dicionário adaptativo para as que não utilizam é muito grande. Já a figura 4.8c) mostra que esse aumento de qualidade está relacionado à um aumento de complexidade computacional, o que ocasiona aumento no tempo de execução.

4.3 Suavização Espectral

O objetivo da suavização espectral é evitar que ocorra super resolução em freqüência quando a janela de Hamming cobre muitos períodos de pitch. Para isto este metódo tenta alargar a banda dos formantes.

Para verificar isto foram realizadas simulações com e sem a suavização espectral, numa configuração utilizando dicionário fixo de 256 amostras, e dicionário adptativo fracionário de tamanho 512, cujas resoluções foram assim distribuídas: oitavas de 20 a 55, quartas de 55 a 101 e unitárias de 101 a 146. Foram utilizados blocos de 20 ms com sub-blocos de 5 ms.

A Figura 4.9 representa o resultado quando aplicado neste método. Observa-se um pequeno aumento da banda no sinal onde é aplicada a suavização espectral. Em termos de medida qualitativa, não foi observada diferença entre as duas simulações (com e sem a suavização espectral).

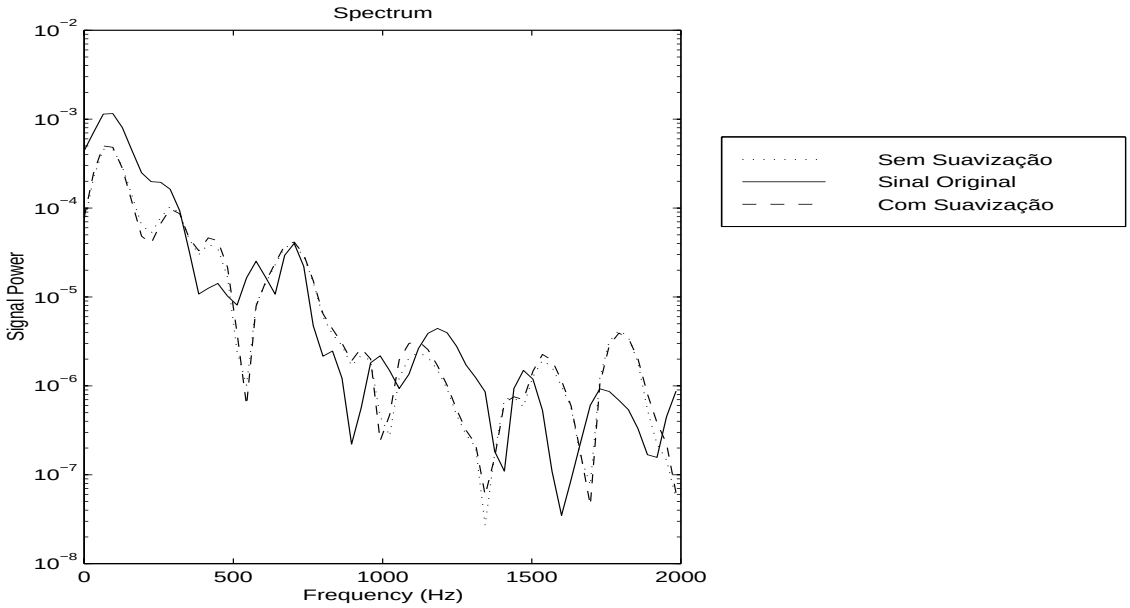


Figura 4.9: Testes de Suavização Espectral.

4.4 Treinamento do Dicionário Fixo

O objetivo deste treinamento é fazer uma seleção qualitativa das amostras de um dicionário fixo. As amostras ou excitações selecionadas irão compor um outro dicionário, dito treinado.

Durante o processo de análise por síntese, todas as amostras do dicionário fixo são testadas. Algumas destas amostras são pouco utilizadas, o que implica num disperdício de tempo ou de qualidade do sinal obtido, já que tal amostra poderia ser substituída por outra mais eficiente. Portanto, a meta do treinamento de dicionário fixo é a sua otimização, ou seja, obter um dicionário que proporcione melhor qualidade no sinal e menor complexidade computacional.

Neste procedimento foi utilizado um dicionário fixo de tamanho 1024, gerado no Matlab. O dicionario fixo resultante é composto das 512 amostras mais utilizadas no dicionário de origem, isto para a codificação de um sinal X. Para testar a eficácia do procedimento, um outro sinal Y foi codificado utilizando um dicionário não-treinado e depois o treinado.

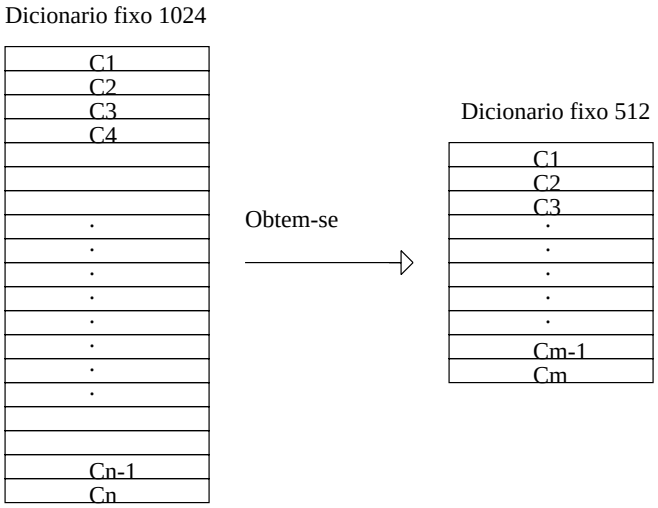


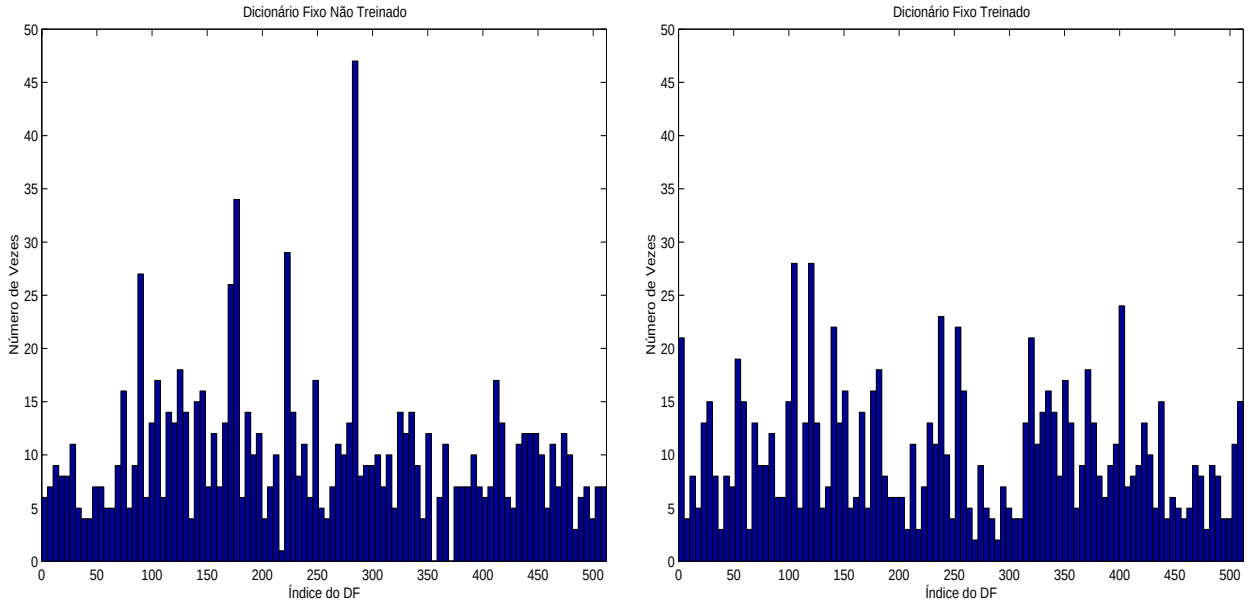
Figura 4.10:Obtenção do DF treinado.

A Tabela 4.8 e os gráficos mostram que foi obtido um ganho no MSE de 0,24 dB com o dicionário treinado, validando este procedimento de melhoria da qualidade do sinal final.

Tabela 4.8: Comparação entre DF.

	Dicionário Treinado	**Dicionário Nao Treinado**
MSE	11.4058	11.6409
RSRPS	18.2120	18.1622

Nota-se também uma distribuição um pouco mais homogênea das freqüências em que as excitações candidatas são escolhidas (figura 4.11).



(a) (b)

Figura 4.11: Histogramas a)DF não treinado ; b)DF treinado.

4.5 Conclusão

Neste capítulo foi realizado um estudo da influência dos dicionários fixo e adptativo no sinal obtido. Para tanto, foram feitas medidas visando caracterizar a contribuição de cada tipo de dicionário, chegando-se a conclusão de que o uso do dicionário adaptativo proporciona grande melhora na qualidade do sinal obtido, podendo-se até diminuir o número de amostras do dicionário fixo.

Outro ponto importante deste capítulo foi a análise da suavização espectral. Notou-se um pequeno alargamento da banda, não proporcionando, entretanto, aumento considerável da qualidade do sinal final. Também foram realizados testes para treinar o dicionário fixo, onde conclui-se possível obter melhoras qualitativas a partir deste artifício.

Capítulo 5

Aceleração do Codificador CELP

5.1 Introdução

No intuito de melhorar a performance do codificador descrito neste projeto, principalmente em relação à complexidade computacional envolvida no processo de codificação, foram aplicadas técnicas para diminuir o seu tempo de processamento. Neste capítulo serão mostradas e explicadas passo a passo todas as modificações realizadas e suas implicações. O objetivo foi de alcançar um tempo de processamento igual a 1/6 do tempo de duração do sinal codificado. Isto para poder acoplar o codificador ao programa que captura vetores de som.

O sistema implementado utiliza blocos de 20 ms e sub-blocos de 5 ms, e foi simulado em uma máquina K6II-450 com 128 MB de memória. O primeiro passo a ser dado foi a retirada dos atrasos fracionários do dicionário adptativo, seguido da redução do tamanho do dicionário fixo. Estas alterações foram realizadas baseadas no resultados obtidos no Capítulo 4. Depois implementou-se uma nova técnica de procura da melhor excitação no dicionário adaptaivo, o que diminuiu muito o tempo de execução. Finalmente fez uma modificação na estrutura do diocionário fixo, chegando-se ao resultado almejado. As seções a seguir apresentam os detalhes destas alterações.

5.2 Retirada dos Atrasos Fracionários

A configuração inicial testada foi a seguinte: dicionário fixo com 512 amostras, dicionário adpatativo com 512 amostras e atrasos fracionários com resoluções distribuídas da seguinte maneira: oitavas de 20 a 55, quartas de 55 a 101 e unitárias de 101 a 146. Foram considerados blocos com duração de 20 ms e sub-blocos com duração de 5 ms. A frase utilizada para o teste foi a de um locutor feminio.

Os resultados obtidos não foram satisfatórios, já que apesar da boa qualidade do sinal, o tempo de processamento foi muito elevado. Os resultados obtidos podem ser vistos na Tabela 5.1. Já a Figura 5.1 ilustra a estrutura funcional desta configuração.

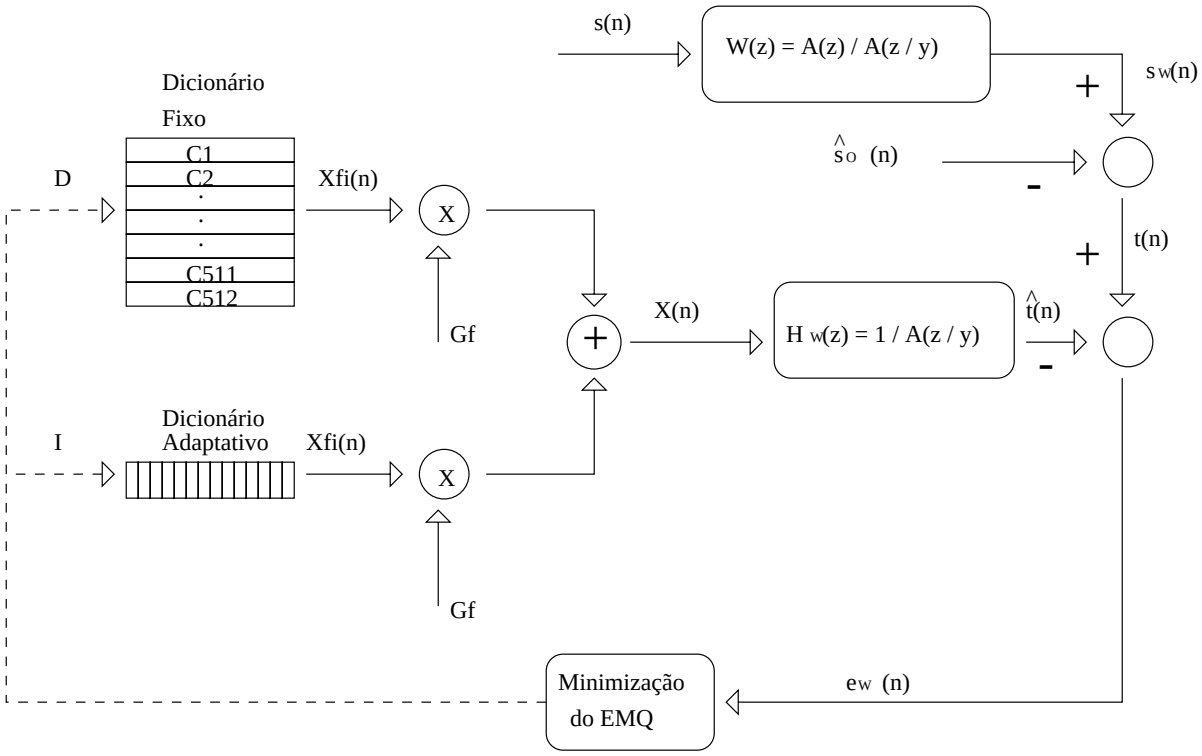


Figura 5.1: Estrutura do Codificador Celp.

Em vista dos resultados obtidos, a primeira ação na tentativa de acelerar o processo de codificação foi a retirada dos atrasos fracionários do dicionário adaptativo. O cálculo destes atrasos fracionários implica em uma grande complexidade computacional, resultando em um tempo de processamento muito alto. Tendo em vista os resultados apresentados no Capítulo 4, onde observou-se que os sinais obtidos de codificadores que utilizavam DA não fracionários possuíam qualidade similar aos do que usavam DA fracionários, foi feita a retirada destes atrasos. Esta mudança acarretou uma considerável redução do tempo de execução. Mesmo assim ficamos longe do ideal, que é a solução em tempo real. A Tabela 5.1 mostra os resultados obtidos.

Tabela 5.1: Resultado da retirada do atraso fracionário.

	RSRSP (dB)	MSE (dB)	t (s)	Taxa (kbps)	Arquivo de som
1DF de 512 e 1DAnF de 512	17.7603	12.3266	29	8.0	teste-3_512-f1.au
1DF de 512 e 1DAF de 512	18.1622	11.6409	68	8.0	teste-4_512-f1.au

5.3 Redução do Número de Amostras dos Dicionários

Depois de ter retirado os atrasos fracionários do dicionário adaptativo, a próxima modificação a ser realizada foi a diminuição do número de amostras do dicionário fixo. Vale lembrar que a configuração atual utiliza um dicionário fixo com 512 amostras e um dicionário adaptativo não fracionário também de tamanho 512 com faixa de 20 a 146.

Baseado nos resultados apresentados na tabela , foi escolhido um DF com 128 amostras para substituir o atual, que é de 512. Esta mudança¸ segundo o experimento realizado no capítulo 4, diminui o tempo de execução e pouco influencia na qualidade final do sinal. Reduziu-se o número de amostras do dicionário

adaptativo para 128.

Os resultados destas alterações podem ser vistos na tabela abaixo. Nota-se que houve uma significativa redução da complexidade computacional, acompanhada de uma pequena perda na qualidade do sinal.

Tabela 5.2: Resultado da redução do número de amostras do DF e do DA.

	RSRSP (dB)	MSE (dB)	t (s)	Taxa (kbps)	Arquivo de som
1DF de 512 e 1DAnF de 512	17.7603	12.3266	25	8.0	teste-3_512-f1.au
1DF de 512 e 1DAF de 512	18.1622	11.6409	68	8.0	teste-4_512-f1.au
1DF de 128 e 1DAF de 128	**17.4497**	**12.9387**	**15**	**8.0**	**teste3.1_128-f1.au**

5.4 Alteração na Estrutura do Codificador

Depois de realizadas alterações no número de amostras dos dicionários e na utilização de dicionário adaptativo não fracionário, realizou-se uma mudança na estrutura do codificador.

A filtragem de cada seqüência candidata do dicionário adaptativo por $H_w(z)$ é uma operação de elevada complexidade computacional, acarretando grande tempo de execução. Uma alternativa para evitar isto é deslocar o filtro $H_w(z)$ como mostrado na figura 5.2. Este deslocamento faz com que a seqüência do dicionário adaptativo seja inteiramente filtrada por $H_w(z)$, sendo gerada uma outra seqüência que formará o dicionário adaptativo filtrado. A busca da melhor excitação neste dicionário passa a ser realizada sem que seja necessário filtrar cada seqüência individualmente, diminuindo o número de operações a serem realziadas¸ e consequentemente¸ o tempo de processamento.

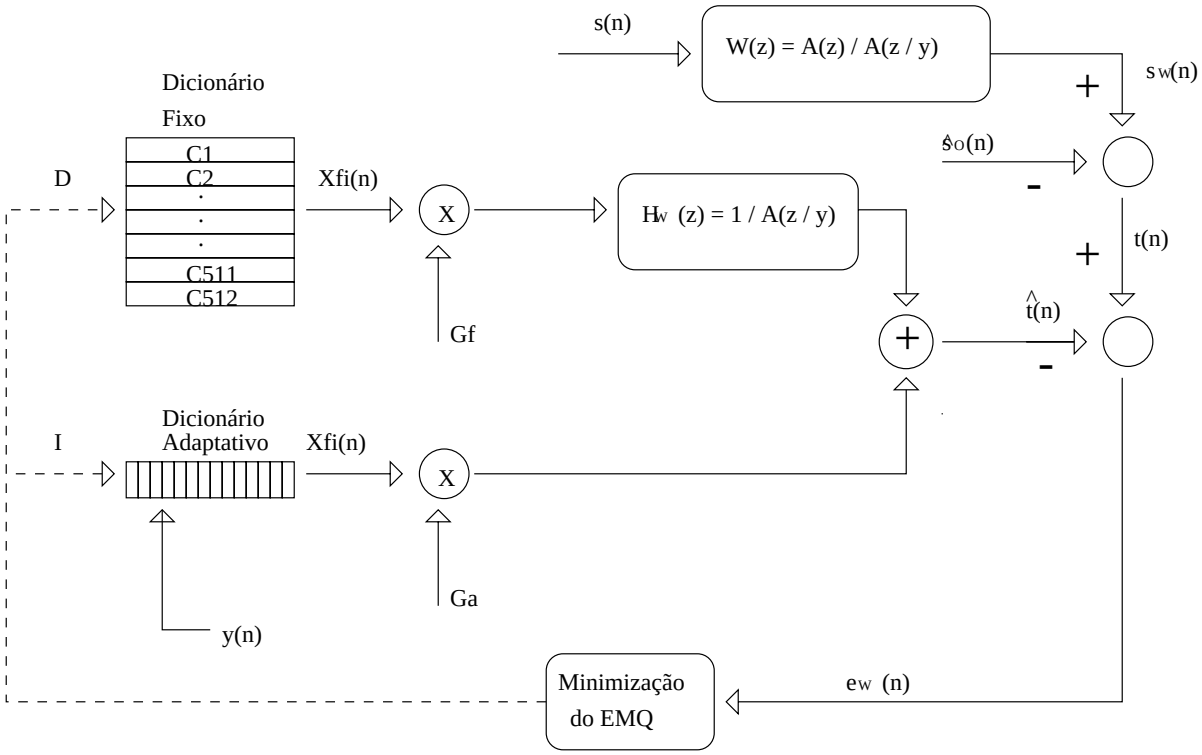


Figura 5.2: Deslocamento do DA.

Assim que é determinado o melhor atraso no dicionário adaptativo filtrado, a melhor excitação can-

didata é lida a partir do dicionário adaptativo não filtrado, que é atualizado com a seqüência escolhida. Temos então duas filtragens do dicionário adaptativo para cada sub-bloco: uma no início do processo de busca e a outra após ser determinado o melhor atraso.

Outra mudança aqui realizada se deu na estrutura do dicionário fixo. Até aqui, usamos um dicionário fixo com 128 amostras, sendo sua estrutura ilustrada na Figura 5.3.

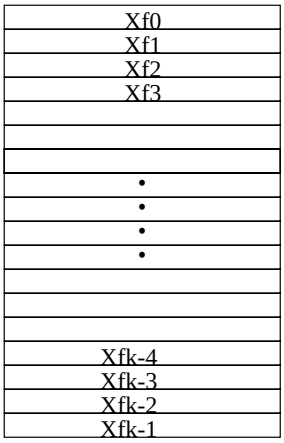


Figura 5.3: Estrutura do DF.

Este dicionário é constituído de seqüências de um processo estocástico gaussiano de média zero, realizado no Matlab. A busca neste dicionário requer um certo esforço computacional. Para substituí-lo, optou-se pela técnica de superposição de vetores-código. A figura 5.4 ilustra a estrutura deste novo dicionário fixo. Ele tem suas seqüências definidas por:

$$c_{2I}(n) = c(n + mI) \tag{5.1}$$

$$I = 1, ..., K_2$$

$$n = 0, ..., N_s - 1$$

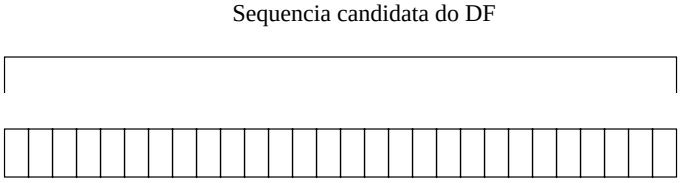


Figura 5.4: Nova estrutura do DF.

C(n) é uma seqüência de amostras aleatórias de comprimento ($mK_2 + N_s$) e m é uma constante inteira maior do que um. Sua estrutura é similar à do dicionário adaptativo não fracionário.

Com esta alteração, a memória requerida para o armazenamento do dicionário fixo foi reduzida drasticamente. Foi utilizado um dicionário com m=2, n=128 e $K_2 = 40$.

A última alteração realizada foi a filtragem das amostras do dicionário fixo antes de se iniciar a análise por síntese. Assim¸ ao invés de filtrarmos as seqüências do dicionário fixo para cada sub-bloco do sinal, faz-se a filtragem antes e utiliza-se este resultado diretamente para a procura da melhor excitação. Esta mudança pode ser visualizada na figura 5.5.

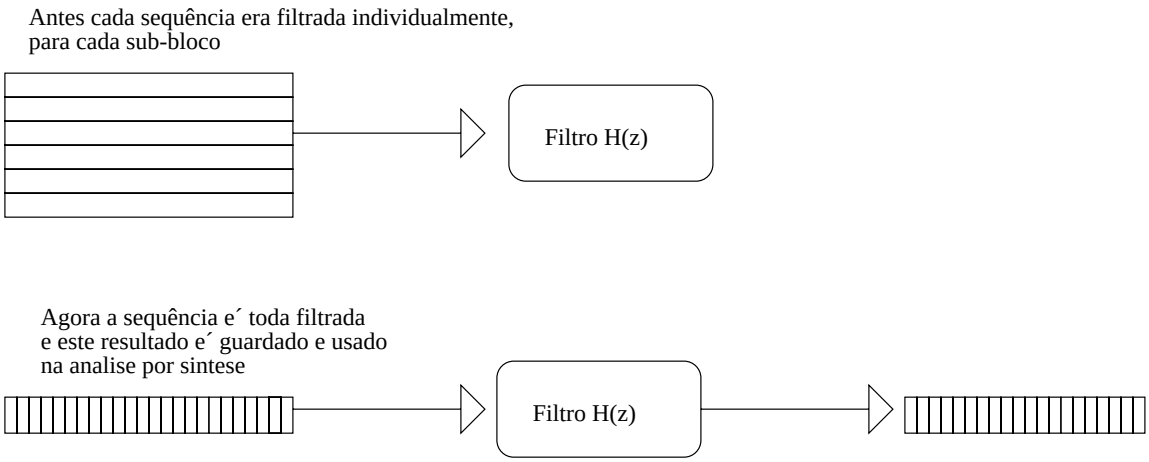


Figura 5.5: Deslocamento do DF.

A tabela 5.3 mostra os resultados encontrados. Notamos que o tempo de processamento caiu drasticamente, acompanhado pela queda da qualidade do sinal obtido. Esta perda de qualidade, no entanto, não comprometeu a performance do codificador.

Tabela 5.3: Resultado do deslocamento do DAnF

	RSRSP (dB)	**MSE (dB)**	**t (s)**	**Taxa (kpbs)**	**Arquivo de som**
1DF de 512 e 1DAF de 512	18.1622	11.6409	68	8.0	teste-4_512-f1.au
1DF de 512 e 1DAnF de 512	17.7603	12.3266	45	8.0	teste-3_512-f1.au
Mudanças Estruturais	**17.4761**	14.1836	1	8.0	teste3.1-128-f1.au

Conclui-se que o objetivo foi alcançado, já que obteve-se um sinal de boa qualidade, a uma baixa taxa de bits e com tempo de processamento aproximadamente igual a 1/6 da duração do sinal a ser codificado.

5.5 Implementação em Tempo Real

Com as técnicas de aceleração ao codificador CELP, o resultado obtido foi excelente, já que as mudanças realizadas no programa proporcionaram um processamento do sinal em tempo real, com uma boa qualidade e a uma baixa taxa de bits.

O objetivo de fazer com que seja possível codificar o sinal de fala de um microfone e reproduzi-lo em tempo real foi cumprido. Para a codificação do sinal usaremos o sistema final obtido. Para a captura dos vetores de som (oriundos do microfone) e da execução de sua versão codificada, utilizou-se o programa [15]. Este programa será detalhado em seguida, bem como as modificações realizadas.

Resumindo, a última etapa deste projeto foi fazer um sistema capaz de capturar, codificar, decodificar e executar um sinal de áudio oriundo de um microfone, sendo todo o processamento realizado em tempo real. A figura 5.6 ilustra o diagrama de blocos do sistema resultante.

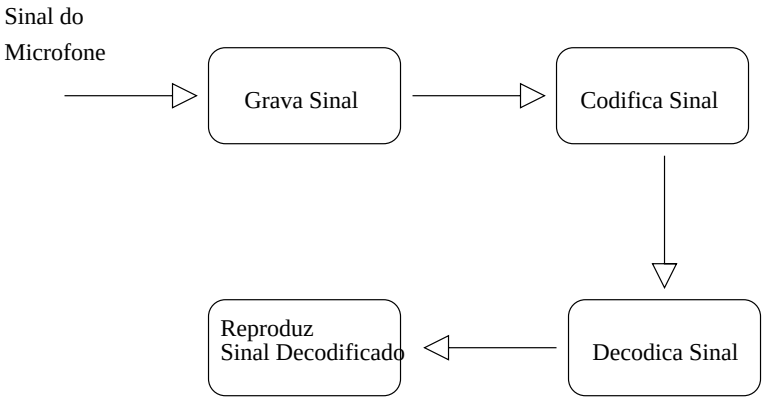


Figura 5.6: Sistema resultante.

Para implementar o sistema mostrado na figura 5.6, utilizou-se cinco módulos diferentes. Um responsável por gravar o sinal oriundo do microfone, outro para fazer a codificação do sinal, um terceiro para decodificar, o quarto para reproduzi-lo, e o último módulo contendo esta seqüência de chamadas. Nas próximas seções teremos um estudo detalhado de cada um deles.

5.5.1 Obtenção do Vetor de Som

O trabalho descrito em [15] propõe-se a criar um arquivo em Linux capaz de gravar e de reproduzir arquivos de som em diversas freqüências e com a possibilidade de gravar os mesmos em 8 ou 16 bits. Além disso, existe a possibilidade de se escolher o número de canais para a gravação, mono ou stereo.

Na tabela 5.4 temos as características gerais da classe wave¸ que é a principal classe deste programa. Foi preciso estudar sua funcionalidade para realizar as modificações necessárias.

Para a implementação do primeiro bloco do sistema mostrado na figura 5.6, usou-se a classe *rec.* Esta classe recebe como parâmetro o tempo de duração do sinal a ser gravado, setado neste caso para 5 s. Outros parâmetros por esta classe também recebidos são o canal e o número de bits, setados em mono e 16 respectivamente.

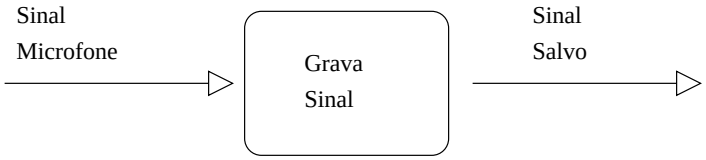


Figura 5.7: Bloco grava sinal.

5.5.2 Codificação do Sinal

Nesta etapa é realizada a codificação do sinal gravado, sendo usada a versão acelerada do codificador CELP. Para relembrarmos, este codificador possui a seguinte configuração:

- Dicionário adaptativo não fracionário com 128 amostras, com uma faixa de resolução de 20 a 146.
- Dicionário fixo com superposição dos vetores-código (128 amostras).
- Bloco de 20 ms e sub-blocos com 5 ms.
- Taxa de bits total de 8 kbps.
- Tempo de execução aproximadamente $\frac{1}{6}$do tempo de duração do sinal.

- Filtragem das seqüências do DF e do DAnF por inteiro.

Codifica-se então o sinal salvo em 5.5.1, sendo os ganhos e os índices dos dicionários, bem como os coeficientes das LSF quantizadas gravadas em arquivos binários. Estes arquivos serão utilizados pelo decodificador, que é o próximo bloco do sistema. O decodificador irá reconstruir o sinal.

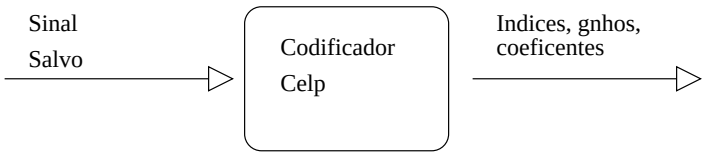


Figura 5.8: Codifica sinal.

5.5.3 Decodificação do Sinal

Logo após a codificação do sinal, realiza-se sua decodificação. Este processo utiliza os dados de saída do codificador, que são: índices e ganhos do dicionário fixo e adaptativo, e coeficientes das LSF.

O tempo requerido para decodificação é mínimo, e o sinal reconstruído é salvo em um arquivo para posteriormente ser reproduzido.

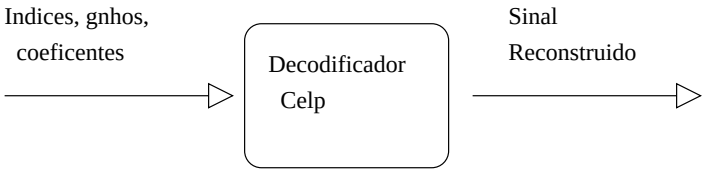


Figura 5.9: Decodifica Sinal.

5.5.4 Reprodução do Sinal

A última etapa é a reprodução do sinal reconstruído. Para esta reprodução, utiliza-se o comando *play* próprio do Linux.

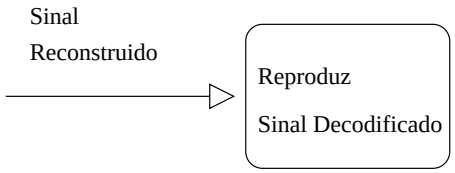


Figura 5.9: Reproduz Sinal.

5.6 Conclusão

Este capítulo mostrou modificações na configuração e estrutura do codificador CELP no intuito de se obter uma boa qualidade em tempo real. Primeiramente, tendo como base os testes de performance realizados no Capítulo 4, retirou-se os atrasos fracionários do dicionário adaptativo. Esta alteração diminuiu o tempo de processamento para cerca de 37% do valor inicial, e a qualidade em 0.4 dB.

A segunda alteração realizada, também baseada nos testes do Capítulo 4, foi a redução do número de amostras tanto no dicionário fixo quanto no dicionário adaptativo, o que proporcionou um tempo quatro

vezes menor e uma queda na qualidade do sinal de 0.3 dB. Até esta etapa, as alterações realizadas foram em relação a configuração do codificador.

As alterações posteriores foram de ordem estrutural. Estando com um tempo aproximado de 15 s, o que representava 3 vezes a duração do sinal codificado, partiu-se para um novo método de busca no dicionário adaptativo. Esta alteração consistiu em filtrar toda a seqüência do dicionário adaptativo antes de ser realizada a análise por síntese para os sub-blocos em questão. Outra alteração foi em relação a estrutura do dicionário fixo, realizando-se a superposição dos vetores-código. Também as amostras do dicionário fixo foram filtradas antes da análise por síntese, o que acarretou, como no caso do dicionário adaptativo, redução do número de filtragens a serem realizadas. Com estas mudanças, a complexidade computacional do sistema foi reduzida drasticamente, sendo o tempo de processamento obtido cerca de $\frac{1}{6}$ da duração do sinal a ser codificado, para uma redução na qualidade de 0.9 dB.

Depois de realizada com sucesso à aceleração do codificador, integrou-se a este sistema um programa que capturava vetores de som e reproduzia o sinal. Feitas as modificações necessárias, foi possível falar ao microfone, capturar este sinal de entrada, codificá-lo, decodificá-lo e reproduzi-lo em tempo real, sendo assim alcançados os objetivos do trabalho.

Tabela 5.4: Classe Wave
class wave {
wave_head header;
// Estrutura que define o cabeçalho padrãode uma wave.
public:
char *filename;
// Objeto que define o arquivo em que a wave atual esta gravada.
U4 samples;
// Objeto que define a quantidade de amostras em cada canal da wave atual.
S2 *data[MAXCHANNELS];
// Matriz de som onde ficam armazenadas as amostras da wave. Cada linha
// corresponde a um canal. O padrão S2 utilizado em toda a classe é do tipo
// short (16 bits)
wave (U2 channels=1, U4 rate=11025);
// Construtor da classe. Nele podem ser setados a quantidade
//de canais e a freqüência de amostragem do som.
~wave();
// Destrutor.
void set (U2 channels=1, U4 rate=11025);
// Objeto utilizado para se alterar os valores padrão setados pelo construtor.
bool save (char *my_filename="sample.wav");
// Objeto utilizado para salvar os dados de som num arquivo de formato *.wav.
bool open (char *my_filename="sample.wav");
// Objeto utilizado para abrir arquivos wave e deixar as amostraz disponives
// numa matriz de som.
bool play();
// Objeto que reproduz as amostras da matriz de som em tempo real. É
// necessário uma placa de som para que tal aconteça.
bool rec(float time=1.0);
// Objeto que grava som em tempo real. O tempo de gravação, em segundos,
// é passado como paramêtro. Também aqui é necessário uma placa de som.
bool convert(char *filesource="temp.wav", char *filetarget="sample.wav", U4 bits=16);
// Objeto utilizado para converter arquivos gravados no padrão do programa para outros
// padrões, dentre eles quantidade de bits de cada amostra (8 ou 16) e outros formatos
// de arquivos (*.raw, *.au, etc...)

Capítulo 6

Conclusão

6.1 Resumo do Projeto

Este trabalho descreveu detalhadamente o sistema CELP de codificação de fala desenvolvido por Ranniery Maia, em [1]. Foram apresentadas as características deste codificador, sendo abordados tópicos como modelagem LPC, dicionários utilizados, método de análise por síntese e quantização. Também foi visto a descrição completa do algoritmo do sistema implementado, com explicações de todas as funções e rotinas utilizadas¸ relacionando com seus respectivos aspectos teóricos.

Outra ponto aqui abordado foi a caracterização da influência dicionários no sinal final obtido, sendo realizados testes cujos resultados foram analisados detalhadamente. Também foram aplicadas técnicas de aceleração, obtendo-se a codificação em tempo real com perda mínima de qualidade do sinal. Esta aceleração foi importante para que se pudesse integrar o codificador a um sistema onde foi possível falar ao microfone, codificar o sinal, decodificá-lo e reproduzi-lo com ótima qualidade e em tempo real. A seguir são resumidas as contribuições de cada capítulo.

No Capítulo 1 vimos:

- importância da compressão digital da voz, que permite um maior armazanamento da mesma;
- abordagem teórica do codificador CELP, sendo descritas características da fala, padrões de codificadores, objetivos e organização do projeto.

O Capítulo 2 apresentou o sistema Celp estudado. Foram estudadas suas principais características:

- utilização de dois dicionários; fixo e adaptativo com atrasos fracionários;
- interpolação;
- janelamento;
- análise por síntese;
- filtro de ponderação;
- filtro de síntese;
- quantização dos parâmetros do filtro de síntese e dos ganhos dos dicionários.

O Capítulo 3 apresentou a descrição detalhada do algoritmo utilizado na implementação do codificador CELP. Foram explicadas minuciosamente cada etapa e função do programa¸ no intuito de elucidar a funcionalidade do mesmo.

No Capítulo 4 realizou-se os seguintes testes:

- estudo da influência dos dicionários fixo e adptativo no sinal obtido. Para tanto¸ foram feitas medidas visando caracterizar quantitativamente a contribuição de cada tipo de dicionário;
- suavização espectral;
- treinamento do dicionário fixo.

No Capítulo 5 realizou-se modificações na configuração e estrutura do codificador Celp no intuito de se obter uma boa qualidade em tempo real :

- foram retirados os atrasos fracionários do dicionário adaptativo;
- reduziu-se o número de amostras tanto no dicionário fixo quanto no dicionário adaptativo;
- alterações de ordem estrutural, sendo aplicado um novo método de busca no dicionário adaptativo;
- alterações na estrutura do dicionário fixo, realizando-se a superposição dos vetores-código;
- integração deste sistema ao um programa [15].

6.2 Propostas para Trabalhos Futuros

As propostas para trabalhos futuros são relacionadas a seguir. O objetivo é a continuidade deste projeto final.

- Implementação da versão acelerada do sistema em DSP;
- Estudo mais detalhado de técnicas de projeto de dicionário fixo, com o objetivo de se reduzir o espaço de armazenamento e melhorar a qualidade do sinal reconstruído;
- Aplicação de técnicas para diminuir o caráter ruidoso do codificador, que geralmente imprimem alguma aspereza (rouquidão) ao sinal reproduzido. Um exemplo é o uso de pós-filtragens adaptativas.

Tabela 6.1: Configuração final do Codificador Celp

Tamanho de cada bloco	20 ms
Tamanho de cada sub-bloco	5 ms
Análise LPC:	Método de autocorrelação com janela de Hamming de 25ms (200 amostras), centrada no último sub-bloco de cada bloco.
Número CPL:	10 (p=10)
Quantização dos CPL:	Quantizador QDLSF-32
Interpolação dos CPL:	Linear, no domínio das LSF
Dicionário Adptativo:	Atrasos não fracionários de 20 a 147 com 128 amostras
Dicionário Fixo:	128 amostras, usando a técnica de superposição dos vetores código
Quantização de G_a	Não uniforme de 0 a 2
Quantização de Gf	Não uniforme de -0,05 a 0,05
Parâmetro perceptual de W(z):	$\delta = 0,8$
Taxa de bits:	CPL: 40 bits/20ms =2 Kbps Excitação:30 bits/5 ms =6 Kbps Totalizando:8 Kbps

Referências Bibliográficas

[1] MAIA, R. da S. ¸"Codificação CELP: análise e novas técnicas'' M. Sc. dissertation¸ COPPE/UFRJ¸Rio de Janeiro¸ Brazil¸ 2000.

[2] MAIA¸ R. da S., NETTO, S. L., RESENDE, F. G. V.¸Jr.¸"A Fast Search Method for CELP Speech Coding", *First IEEE South-American Workshop on Circuits and Systems*, Rio de Janeiro/Bahía Blanca, Brazil/Argentina, pp. 10-12, November, 2000.

[3] KROON¸Peter, SWAMINATHAN, Kumar, "A High-Quality Muiltirate Real-Time CELP Coder" ,*IEE Journal on Selected Areas in Communication*¸ v. 10, n. 10, June 1992

[4] RABINER¸L. R., SCHAFER, R. W., Digital Processing of Speech Signals. Englewwod Cliffs, N. J., Prentice-Hall, 1978

[5] SCHROEDER, M. R., "Vocoders: analysis and synthesis of speech", Proceedings of the IEEE, v. 54, n. 5, May 1996.

[6] da SILVA, L. M., Contribuições para a melhoria da codificação CELP a baixas taxas de bits. Tese de D.Sc., PUC-RJ, Rio de Janeiro, RJ, Brasil, Fev. 1996.

[7] SPANIAS, A. S., " Speech Coding: a tutorial review", Proceedings of the IEEE, v. 82, pp. 1541-1582, Oct. 1994.

[8] GERSHO, A.¸ "Advanceds in speech and audio compression", Proceedings of the IEEE, v. 82, n. 6, pp. 900-918, Jun. 1994.

[9] SCHROEDER, M. R., ATAL, B. S., "Code-excited linear prediction (CELP): high-quality speech at very low bit rates". In: Proc. of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, pp. 937-940, 1985.

[10] da SILVA, L. M. ALCAIM, A.¸" A modified CELP model with computacionally efficient adaptative codebook search", IEEE Signal Processing Letters, v.2, n. 3, pp. 44-45, Mar. 1995.

[11] H. Dudley, "Remaking Speech," J. Acoust. Soc. Am., Vol. 11, p. 169, 1939.

[12] H. Dudley, "The Vocoder," Bell Labs. Record., 17, p. 122, 1939.

[13] G. Fant, "Acoustic Theory of Speech Production," Mounton and Co., Gravenhage, The Netherlands, 1960.

[14] B. Atal and J. Remde, "A new model for LPC excitation for producing natural sounding speech at low bit rates," Proc. ICASSP-82, pp. 614-617, April 1982.

[15] TERUNSKIN,R., "Controle de um robô por comandos de voz"¸ Projeto Final em andamento, Escola de Engenharia, Departamento de Eletrônica, UFRJ, Rio de Janeiro, Brazil, 2001.

[16] CAMPOS, M. L. R. de , "CDMA: Mobile Communications Systems", Notas de aula, Escola de Engenharia, Departamento de Eletrônica, UFRJ, Rio de Janeiro, Brazil, 2000.