NELSON CAMPANER

EXCITAÇÃO MULTI-TAXA USANDO QUANTIZAÇÃO VETORIAL ESTRUTURADA EM ÁRVORE PARA O CODIFICADOR CS-ACELP COM APLICAÇÃO EM VOIP

FLORIANÓPOLIS
2006

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

EXCITAÇÃO MULTI-TAXA USANDO QUANTIZAÇÃO VETORIAL ESTRUTURADA EM ÁRVORE PARA O CODIFICADOR CS-ACELP COM APLICAÇÃO EM VOIP

Dissertação submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a
obtenção de grau de Mestre em Engenharia Elétrica.

NELSON CAMPANER

Florianópolis, Setembro de 2006.

EXCITAÇÃO MULTI-TAXA USANDO QUANTIZAÇÃO VETORIAL ESTRUTURADA EM ÁRVORE PARA O CODIFICADOR CS-ACELP COM APLICAÇÃO EM VOIP

Nelson Campaner

'Esta Dissertação foi julgada adequada para obtenção do Título de Mestre em Engenharia Elétrica, Área de Concentração em *Comunicações e Processamento de Sinais*, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina.'

Prof. JOCELI MAYER, Ph. D.
Orientador

Prof. NELSON SADOWSKI, Dr.
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca examinadora:

Prof. JOCELI MAYER, Ph. D.
Presidente

Prof. BARTOLOMEU FERREIRA UCHÔA FILHO, Ph.D.

Prof. JOSÉ CARLOS MOREIRA BERMUDEZ, Ph.D.

À minha esposa Andreia,
pela compreensão infinita
e por me incentivar nos
momentos difíceis.

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para a obtenção do grau de Mestre em Engenharia Elétrica.

EXCITAÇÃO MULTI-TAXA USANDO QUANTIZAÇÃO VETORIAL ESTRUTURADA EM ÁRVORE PARA O CODIFICADOR CS-ACELP COM APLICAÇÃO EM VOIP

Nelson Campaner
Setembro/2006

Orientador: Prof. JOCELI MAYER, Ph. D.
Área de Concentração: Comunicações e Processamento de Sinais.
Palavras-chave: codificação de voz, taxa variável, VoIP, árvore, ACELP.
Número de Páginas: 116.

Este trabalho apresenta um estudo sobre codificação multi-taxa estruturada sobre o algoritmo CS-ACELP (*Conjugate-Structure Algebraic-Code-Excited Linear-Prediction*) e a especificação G.729, cujo objetivo é propor um codificador com taxa variável, através da busca da melhor excitação fixa usando *codebook* estruturado em árvore, para aplicações VoIP (*Voice-over-IP*). A mudança progressiva do transporte de voz das redes de circuito para as redes IP (*Internet Protocol*), apesar dos diversos aspectos positivos, tem exposto algumas deficiências intrínsecas destas, mais apropriadas ao tráfego de "melhor esforço" do que ao tráfego com requisitos de tempo. Esta proposta está inserida no conjunto das iniciativas, no âmbito do transmissor, que procuram minimizar os efeitos danosos da rede sobre a qualidade da voz reconstruída. O *codebook* proposto tem estrutura em árvore binária, concebida a partir de uma heurística onde os vetores CS-ACELP são ordenados por valor de forma decrescente. Uma estratégia particular de armazenamento dos nós, envolvendo simplificação nos centróides, codificação diferencial e geração automática dos dois últimos níveis da árvore, permite reduzir o espaço de armazenamento de 640 para apenas 7 kwords. Através deste modelo chega-se a 13 taxas de codificação, de 5,6 a 8,0 kbit/s, com passo de 0,2 kbit/s. A relação sinal ruído fica em 1,5 dB abaixo da mesma medida na especificação G.729 para a taxa de 5,6 kbit/s, e apenas 0,6 dB abaixo quando na taxa 8,0 kbit/s. Testes subjetivos mostraram uma qualidade bastante aceitável para a taxa mínima e praticamente indistinguível do *codec* original na taxa máxima. Além disso, a busca da melhor excitação é 2,4 vezes mais rápida em comparação ao *codec* G.729 e pode ser totalmente compatível com este se a taxa for fixa em 8,0 kbit/s.

Abstract of Dissertation presented to UFSC as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

MULTI-RATE CS-ACELP EXCITATION USING TREE STRUCTURED VECTOR QUANTIZATION FOR VOIP APLICATIONS

Nelson Campaner
September/2006

Advisor: Prof. JOCELI MAYER, Ph. D.
Area of Concentration: Comunicações e Processamento de Sinais.
Keywords: voice coding, variable rate, VoIP, tree, ACELP.
Number of Pages: 116.

This work presents a study about multi-rate coding structured over CS-ACELP (*Conjugate-Structure Algebraic-Code-Excited Linear-Prediction*) algorithm and G.729 standard, whose purpose is to come up with a variable rate codec by means of best fixed excitation search using a tree structured codebook, for VoIP (*Voice-over-IP*) applications. The progressive change of voice transmission from circuit switched to IP (*Internet Protocol*) networks, besides its many positive aspects, has exposed some natural deficiencies of the latter, better suited to best effort traffics than traffics with time requirements. This proposition can be inserted in the bunch of efforts, related to the sender, that seek to reduce the network impairments over the quality of reconstructed voice. The suggested codebook has a binary tree structure heuristically conceived where algebraic CS-ACELP vectors are disposed by value in a decreasing order. Additionally, a particular approach to store the tree nodes are considered, which involves centroid simplification, differential coding and automatic generation of the last two layers of the tree, squeezing the storing space from 640 down to 7 kwords. Through this model we reach 13 coding rates, ranging from 5.6 to 8.0 kbit/s, with 0.2 kbit/s step. The signal-to-noise ratio is 1.5 dB below the same measure for G.729 standard at the rate 5.6 kbit/s, and just 0.6 dB lower at 8.0 kbit/s. Subjective tests pointed to an acceptable quality at minimum rate and virtually indistinguishable quality from the original codec at the maximum one. Also, searching for the best fixed excitation is 2.4 times faster than G.729 and can be truly compatible with it if the rate is fixed in 8 kbit/s.

SUMÁRIO

LISTA DA FIGURAS

LISTA DE TABELAS

C A P Í T U L O 1

INTRODUÇÃO

1.1 Introdução

Embora a transmissão de voz sobre redes de dados venha sendo estudada desde a década de 70 [1, 2], quando a agência DARPA (*Defense Advanced Research Projects Agency*) do governo dos Estados Unidos era responsável pela condução das pesquisas, somente na década de 90, com a explosão da *internet*, é que o assunto ganhou profusão e hoje definitivamente ameaça a hegemonia da secular rede de telefonia tradicional. Da mesma forma que a necessidade de reduzir custo de serviços, a melhoria da qualidade e o desejo de adicionar novas funcionalidades ao sistema impulsionaram a introdução da tecnologia digital na rede de telefonia tradicional [3], conhecida como rede de circuitos, estes mesmos fatores (exceto talvez a melhoria da qualidade) estão motivando a migração dos serviços de telefonia da rede de circuitos para a rede de dados não orientada a conexão, especificamente a rede IP (*Internet Protocol*).

Um dos fatores que estimulam essa migração é a facilidade de convergência entre voz e dados na camada de aplicação [4]. A integração do serviço de telefonia com o serviço de dados (este disponível na rede IP) abre espaço para uma série de novas aplicações [5] que, a princípio, não seriam facilmente exeqüíveis sobre a rede de circuitos, por exemplo, devido à necessidade de reprogramação das centrais de comutação [6]. Alguns exemplos destas novas aplicações são: o tele-atendimento (também conhecido como *call centers*) integrado com a *web*; trabalho a distância (conhecido como *teleworking*); vídeo telefonia; tarifação em tempo real; teleconferência usando quadro eletrônico (ou *whiteboards*); acesso a FAX, correio eletrônico (*email*) e correio de voz a partir de um ponto na rede IP.

Além da simplicidade para a criação de novos serviços, a redução de custo é outro importante aspecto relacionado à introdução da telefonia na rede IP. A rede de circuitos é pouco eficiente no transporte da fala pois permanece inteiramente dedicada aos locutores durante uma interação, não permitindo que seus recursos sejam compartilhados durante os períodos de inatividade, como ocorre na pausa entre palavras e principalmente quando um locutor está em silêncio enquanto o outro fala. A rede IP, por outro lado, além de suportar

na mesma estrutura os diversos tipos de tráfego, incluindo voz, vídeo e dados, com conseqüente redução de custo [7], ainda permite que a natureza aleatória da fala seja melhor explorada, por exemplo, transmitindo quadros (datagramas IP) apenas durante os períodos de atividade do locutor. A flexibilidade da rede ainda proporciona um ambiente onde técnicas de transmissão multi-taxa podem ser empregadas, tanto para reduzir a taxa média de bits em uma ligação ponto a ponto quanto para adaptar a taxa às condições da rede. Outros fatores, talvez não tão relevantes quanto os mencionados mas que também contribuem para a utilização do transporte de voz sobre a rede IP são [2, 8, 9]: utilização de técnicas de segurança quando necessário, como encriptação e autenticação; possibilidade de interconexão de redes heterogêneas; facilidade de ampliação dos sistemas; desregulamentação; interfaces de usuários mais amigáveis, pois a "inteligência" não está na rede (como ocorre na telefonia tradicional) e sim nos dispositivos terminais.

A transmissão de voz sobre a rede IP, também conhecida como VoIP (*Voice-over-IP*), embora seja uma alternativa atraente sob vários aspectos, encontra resistência na própria concepção da rede, mais voltada para o tráfego de "melhor esforço" [1] do que para o tráfego com requisitos de tempo. Muitos esforços têm sido direcionados para esta questão na tentativa de solucionar os desafios técnicos que contrapõem e oferecer um serviço de transmissão de voz com qualidade adequada [10, 5, 8]. Dentre estes desafios estão: o problema da perda de quadros, o atraso e a variação do atraso (conhecida como *jitter*). A tecnologia VoIP possibilita a transmissão de sinais de voz sobre a rede IP na forma de quadros contendo amostras digitalizadas e processadas. Na rede, cada quadro pode ser roteado de forma independente e assim percorrer caminhos diferentes, chegando no destino em tempos distintos, talvez fora de seqüência ou até mesmo não chegando. No destino estes quadros são agrupados novamente, reordenados e executados, para que o recebedor possa ouvir a voz reconstruída. Por ser uma rede não orientada a conexão, não se pode conhecer *a priori* a distribuição do atraso e perda de quadro, dado que dependem do comportamento de outros acessos ao longo da rede. Desta forma, não havendo mecanismos de controle na própria rede, é praticamente impossível prover garantia de qualidade [11].

A perda de quadro na rede é um fenômeno comum e afeta severamente a qualidade da voz reconstruída pois cada quadro carrega aproximadamente a informação de um fonema. Apesar de o cérebro conseguir "reconstruir" alguns fonemas perdidos, o excesso de perda pode tornar a voz incompreensível [5]. Vários fatores contribuem para esse problema, embora o mais significativo seja o congestionamento na rede, ou seja, os

quadros não conseguem espaço nas filas dos roteadores e são eliminados. Além disso, os quadros podem ser corrompidos durante sua travessia; podem não chegar ao destino devido a um mau funcionamento da rede ou podem chegar no destino tarde demais, não havendo outra opção senão descartá-los. Existem diversas técnicas para abordar essa questão [12-15, 5, 16, 17] e foge do escopo deste trabalho discorrer sobre elas embora seja interessante mencionar que, de modo geral, podem ser classificadas em dois grandes grupos: no primeiro grupo estão as técnicas baseadas no emissor, como a adaptação da taxa de transmissão às condições da rede; o envio de informação redundante, para que em caso de perda de quadro parte da informação original possa ser recuperada a partir da redundância (técnica também conhecida como FEC – *Forward Error Correction*); a técnica do entrelaçamento, em que pequenos segmentos de quadros sucessivos são agrupados para formar o quadro a ser transmitido. No segundo grupo estão as técnicas aplicadas ao receptor, que buscam minimizar os danos causados pela perda de quadro. Citam-se: os esquemas de inserção; as técnicas de interpolação e regeneração. Essas técnicas são possíveis porque os sinais de fala exibem forte similaridade entre segmentos curtos sucessivos e funcionam para taxas de perda relativamente baixas (abaixo de 15%) e quadros pequenos (4 – 40 ms) [12].

O atraso e sua variação são, possivelmente, os problemas mais críticos da telefonia IP. Atrasos de até 150 ms são considerados normais para uma interação telefônica [5]. Entre 150 e 400 ms ainda são toleráveis, embora a conversação fique comprometida pois quem ouve precisa aguardar alguns instantes antes de começar a falar, para que haja a certeza de que o outro locutor concluiu sua fala. Acima de 400 ms a comunicação se torna impraticável. O atraso tem uma componente fixa, que está presente mesmo quando não existe tráfego, e uma componente variável, que é de difícil predição e depende da carga na rede [5, 18]. A parte fixa é composta, basicamente, pelo tempo de codificação, inserção do quadro na linha, processamento nos nós da rede (escolha da rota; alterações no preâmbulo do quadro etc), propagação e descodificação. A parte variável é provocada, principalmente, pelos tempos de fila nos roteadores, pelos caminhos diferentes que os quadros percorrem durante sua travessia na rede e pelo tempo de retenção no destino para absorção do *jitter*. A redução da componente fixa do atraso depende de uma série de fatores, que nem sempre estão sob controle do usuário, como por exemplo a escolha do *codec* (de preferência com alta qualidade, baixo atraso de processamento e alta taxa de compressão), a utilização de

linhas de alta capacidade (banda larga), a priorização do tráfego de voz nos roteadores e a escolha de caminhos mais curtos.

A variação do atraso, ou *jitter*, é um problema que aparece sempre que o canal introduz atrasos aleatórios independentes, fazendo com que os quadros cheguem ao destino em intervalos de tempo irregulares [19, 20]. O método comum para resolver esse problema é criar no destino um espaço em memória (*buffer*) para retenção dos quadros de modo que, no momento apropriado, haja um "estoque" de quadros suficiente para a execução de forma regular. Mas existe um compromisso entre o atraso causado pela retenção e a perda de quadro. Se a previsão da execução for retardada (tempo de retenção maior) aumenta a possibilidade de que quadros com grande atraso possam ser executados em tempo, diminuindo assim a perda de quadros, mas sob pena de maior atraso, que por sua vez afeta a interatividade da comunicação. Por outro lado, se a previsão da execução for acelerada (tempo de retenção menor) o atraso na comunicação diminui, mas com conseqüente incremento na perda de quadro, que prejudica a qualidade percebida. Encontrar o ponto ótimo para estes dois aspectos conflitantes, minimizando o tempo de retenção e a perda de quadro, tem sido objeto de estudo de muitos pesquisadores [21-27]. A linha de atuação mais recente nesse sentido procura desenvolver algoritmos adaptáveis que buscam maximizar a qualidade da voz percebida, baseando a otimização em métricas objetivas correlacionadas com o grau de satisfação do usuário [28-30].

A maneira mais efetiva de resolver os problemas relativos à qualidade do áudio na telefonia IP, causados pela própria rede ou pelo impacto do tráfego sobre ela, seria a introdução de mecanismos de controle na própria rede, identificando o tráfego de voz para que a ele possa ser dado tratamento preferencial [18]. Essa estratégia requer o emprego de controle de admissão de chamadas, reserva de recursos ao longo do caminho de transmissão e algoritmos sofisticados de organização de fila nos nós intermediários da rede. Alguns destes mecanismos estão presentes nas soluções *Integrated Services*, *Differentiated Services* e *Multi-Protocol Label Switching* do IETF (*Internet Engineering Task Force*) [31].

Apesar de eficaz, essa estratégia pressupõe uma tarefa complexa e difícil, pois exige que os mecanismos sejam implementados por todo o caminho de transmissão, de forma que a utilização em larga escala não parece uma possibilidade imediata. A outra opção consiste em adaptar as aplicações ao serviço oferecido pela rede, criando mecanismos, tanto no emissor quanto no receptor, que procuram minimizar os impactos

causados pela perda de quadro, pelo atraso e pelo *jitter*. A motivação para este trabalho partiu, em primeiro lugar, da grande exposição da tecnologia VoIP nos últimos anos, que além de despertar o interesse pelo assunto ainda nos impeliu ao seu encontro, justamente por estarmos inseridos em uma empresa cuja tecnologia embarcada em seus produtos repousa na tradicional comutação por circuitos. O segundo fator determinante foi justamente a observação de que é interessante e recomendável a inserção de mecanismos nos extremos da rede, que possam contribuir para a melhoria da qualidade da transmissão de voz.

Este trabalho está inserido no âmbito das iniciativas aplicadas ao transmissor, especificamente naquela relacionada à variação da taxa de transmissão conforme as condições da rede, que sabidamente não garante decréscimo na perda de quadro ou no atraso geral, mas colabora para este resultado e enquadra as aplicações que a utilizam nas chamadas "*boas cidadãs da rede*" [32]. A proposta não é criar um novo esquema para controle de taxa, mas apresentar uma abordagem para codificação multi-taxa (que seja atraente para os mecanismos de controle) baseada no codificador ACELP (*Algebraic-Code-Excited Linear-Prediction*), através da busca da melhor seqüência de excitação em um *codebook* fixo estruturado em árvore.

1.2 Organização do trabalho

O Capítulo 2 traz uma breve discussão sobre codificação e quantização vetorial. Em seguida, faz uma revisão bibliográfica sobre estratégias relacionadas à codificação multi-taxa, tanto aquelas dirigidas pela fonte quanto as orientadas pelas condições de carga no canal.

Como a grande maioria dos *codecs* padronizados tem como base o processo de análise-síntese na estrutura de codificação, o Capítulo 3 primeiramente aborda os princípios da análise-pela-síntese e o desenvolvimento matemático das partes integrantes. Em seguida, trata especificamente do algoritmo CS-ACELP (*Conjugate-Structure Algebraic-Code-Excited Linear-Prediction*), usado na especificação do *codec* G.729, sobre o qual este trabalho se desenvolve.

O Capítulo 4 apresenta a abordagem multi-taxa aplicada à codificação da excitação fixa do algoritmo CS-ACELP, através da busca da melhor seqüência em *codebook* estruturado em árvore. No início são apresentadas as justificativas para a abordagem

TSVQ (*Tree-Structured Vector Quantization*) aplicada ao algoritmo CS-ACELP. Depois são abordados o processo de construção do *codebook* fixo estruturado em árvore e a estratégia de busca no codificador, resultando nas diversas taxas do modelo. Em seguida são apresentados alguns aspectos relativos à transmissão dos parâmetros e finalmente é descrito o processo no descodificador.

No Capítulo 5 são apresentados os resultados práticos obtidos a partir da abordagem multi-taxa usando árvore. Os resultados são analisados em termos de medidas objetivas de distorção. São avaliados também os dados referentes à complexidade computacional, tempo de codificação e variabilidade da taxa. O início do capítulo ainda inclui uma descrição do ambiente de teste.

Finalmente, no Capítulo 6, são apresentadas as conclusões finais e sugestões para futuros trabalhos.

C A P Í T U L O 2

CODIFICAÇÃO DE SINAIS DE VOZ COM TAXA VARIÁVEL

2.1 Introdução

Apesar de o comportamento “altruísta” do codificador - quando reduz sua taxa média de bit ou adapta-se às condições do canal - não redundar necessariamente em garantia de qualidade (possível apenas em redes coordenadas, mas não na internet [33]), tal conduta representa uma opção para que o codificador se adapte, pelo menos em princípio, à natureza aleatória do canal [34].

O desenvolvimento de técnicas para codificação da fala com taxa variável tem sido objeto de grande interesse na última década, devido principalmente à necessidade de maximizar a capacidade de utilização das redes celulares de telefonia móvel e mais recentemente motivado pelo desafio de melhorar a qualidade do serviço de voz sobre as redes de dados.

Este capítulo faz uma revisão bibliográfica, mostrando as estratégias adotadas em alguns trabalhos que propõem abordagens multi-taxa. Embora o tema discorra sobre assuntos exaustivamente discutidos na literatura que trata sobre processamento de sinais de voz, é conveniente uma rápida passagem por alguns pontos relevantes em codificação e quantização vetorial, antes de tratar propriamente das propostas multi-taxa.

2.2 Breve discussão sobre codificação

O objetivo da codificação da fala é representá-la com o menor número de bits possível, mantendo sua qualidade subjetiva [3]. De acordo com a teoria da informação, o melhor desempenho que um esquema de compressão sem perdas pode alcançar é codificar a saída de uma fonte com um número médio de bits igual a sua entropia [35]. Neste caso, o codificador ótimo gera uma seqüência de códigos perfeitamente não correlacionados, em que toda a redundância da fonte é removida. Qualquer compressão acima disso estará associada à perda de informação ou distorção. Em termos práticos, a taxa de bit correspondente à entropia é conseguida apenas assintoticamente, com o atraso ou o espaço de memória requeridos para a codificação tendendo ao infinito (devido à necessidade de se

tomar blocos de dados para análise cada vez maiores) [36]. No entanto, é possível reduzir o valor estimado da entropia quando se tem algum conhecimento sobre a estrutura dos dados da fonte, criando-se modelos para esta estrutura e codificando com base nesses modelos, que podem ser estáticos ou adaptáveis [35].

A análise da forma de onda do sinal da fala, Fig. 2.1 por exemplo, indica que existem níveis de redundância consideráveis entre amostras, possibilitando ganhos em termos de redução de banda de transmissão caso se utilize métodos de codificação que explorem estas características de maneira eficiente. Basicamente pode-se observar as seguintes formas de redundância no sinal de fala [3]: distribuição não uniforme da amplitude; correlação entre amostras adjacentes; correlação entre ciclos adjacentes; correlação entre intervalos de *pitch*; períodos de inatividade.

Distribuição não uniforme da amplitude: amostras com pequenos valores de amplitude são mais freqüentes do que amostras com grandes valores, portanto pode-se melhorar a qualidade da codificação diminuindo os intervalos de quantização para valores pequenos e aumentando para valores grandes.

Correlação entre amostras adjacentes: a maneira mais comum de explorar esta redundância é codificar a diferença entre amostras sucessivas.

Correlação entre ciclos adjacentes: apesar de o sinal da fala ocupar toda a banda disponível do canal telefônico, em determinados momentos da fala vozeada o som compõe-se de algumas poucas freqüências e neste caso apresenta forte correlação entre conjuntos sucessivos de amostras, correspondendo a ciclos de oscilação. Codificadores que exploram a redundância entre ciclos sucessivos são notoriamente mais complexos do que aqueles que removem apenas a redundância entre amostras adjacentes e representam uma transição entre os codificadores de forma de onda de alta taxa de bit, e os codificadores paramétricos de baixa taxa.

Correlação entre intervalos de *pitch*: os sons vozeados resultam da passagem de ar forçado pelas cordas vocais tencionadas e ajustadas de modo a produzir pulsos de ar quase periódicos que excitam o trato vocal. O intervalo entre estes pulsos é chamado de *pitch*. Portanto, os sons vozeados além de exibirem redundância entre ciclos sucessivos ainda apresentam repetição de padrões de longa duração correspondentes ao *pitch*. Tipicamente o intervalo de *pitch* varia de 5 a 20 ms para homens e 2,5 a 10 ms para mulheres. Uma das maneiras mais eficientes de codificar porções vozeadas da fala é codificar um intervalo de *pitch* e depois usá-lo como padrão de repetição para cada intervalo subseqüente do mesmo

som. Embora a codificação do *pitch* provoque uma redução significativa na taxa de bit, sua detecção não é uma tarefa simples e pode resultar em sons notoriamente sintetizados se feita de forma inadequada.

Períodos de inatividade: em uma interação telefônica existe atividade da fala (para cada locutor) em aproximadamente 40% do tempo de duração da ligação. Estes intervalos de inatividade podem ser detectados e explorados adequadamente para redução da taxa de bit.

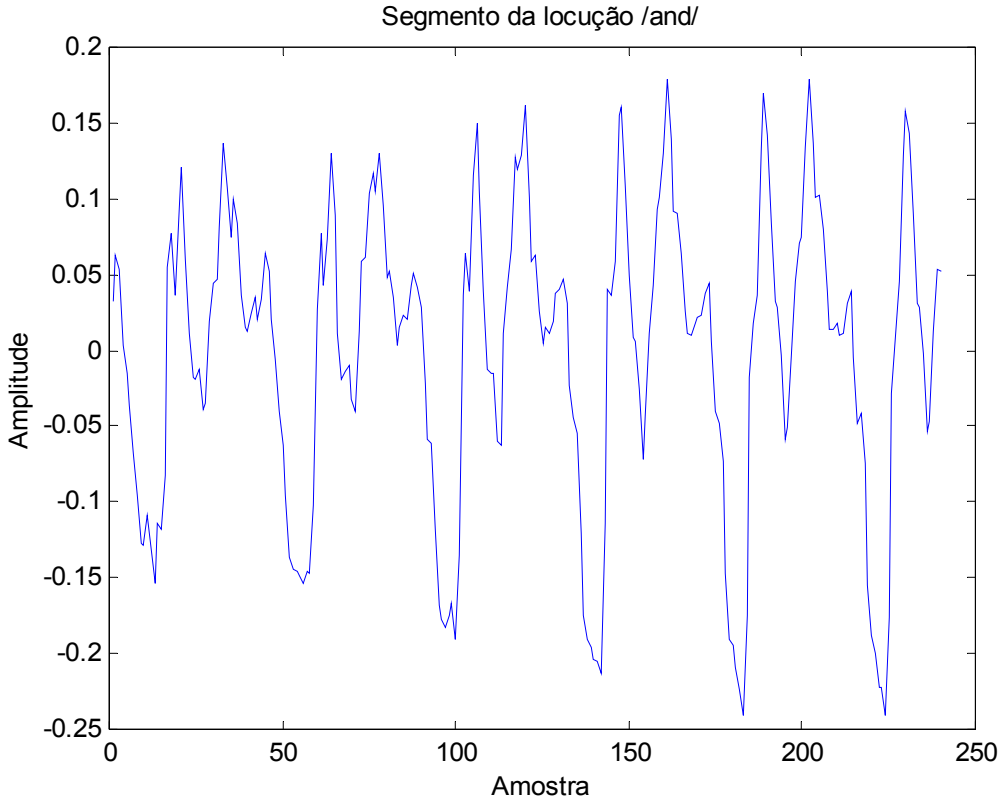


Fig. 2.1 Trecho da locução /*and*/.

Codificadores que exploram estas características, operando em taxas médias ou baixas, geralmente empregam o processo de análise-síntese e quantização vetorial. Na etapa de análise, a fala é representada por um conjunto de parâmetros adequadamente codificados. Na síntese, estes parâmetros são descodificados para reconstruir a fala. O processo de análise pode ser em laço aberto ou fechado. No processo em laço fechado os parâmetros são extraídos e codificados minimizando explicitamente uma medida da diferença entre a fala original e a reconstruída, portanto incorporando a síntese como parte da análise, por isso chamada de “análise-pela-síntese”.

De modo geral os codificadores podem ser classificados em três grupos: codificadores de forma de onda, *vocoders*, e codificadores híbridos [37]. Os codificadores de forma de onda são projetados para serem independentes de sinal. Seu objetivo é mapear a forma de onda da entrada do codificador, de modo a produzir uma réplica na saída do descodificador. Portanto, pouco ou nenhum conhecimento da natureza do sinal é necessário, e esses codificadores praticamente se aplicam a qualquer forma de onda presente no canal de voz, como tons de sinalização, dados e até música. Devido a essa transparência, a eficiência na codificação geralmente é modesta, mas pode ser melhorada para certos tipos mais freqüentes de sinal quando exploradas algumas características estatísticas. Na classe de codificadores de forma de onda também estão os codificadores diferenciais, projetados especificamente para explorar a redundância entre amostras sucessivas do sinal da fala. O valor presente da amostra do sinal é estimado através da combinação linear de valores passados, e o resíduo de predição, que é a diferença entre a amostra presente e a estimada, é calculado e codificado com um número reduzido de bits pois sua variância é menor do que a variância do sinal de entrada. Devido à natureza não estacionária do sinal da fala, o preditor deve ser atualizado periodicamente, para acompanhar as mudanças na envoltória espectral do sinal. Seguindo essa linha de atuação, dois preditores variantes no tempo são utilizados para descrever melhor o sinal da fala: o preditor de curto atraso (PCA), usado para modelar a envoltória espectral, e o preditor de longo atraso (PLA), usado para modelar os detalhes do conteúdo espectral, representando a quase periodicidade para os sons vozeados [36].

A filosofia dos *vocoders* (*voice coders*), por outro lado, está baseada no conhecimento a priori da maneira como o sinal da fala é gerado pelo locutor (modelo) e são mais focados na inteligibilidade da geração, sem que haja necessariamente a replicação da entrada. Os *vocoders* conseguem baixas taxas de bit mas normalmente tendem a produzir um sinal de fala sintetizado. O requisito fundamental para se manter uma boa qualidade da fala consiste em preservar o espectro de potência do sinal, uma vez que a relação de fase entre as componentes de freqüência tem menor importância em termos da percepção. Isto se deve ao fato de que o ouvido é mais sensível à quantidade de energia presente em cada componente de freqüência da fala do que na relação de fase entre as componentes. Por esta razão a forma de onda temporal produzida por um *vocoder* não apresenta, necessariamente, semelhança em relação à forma de onda original, pois a ênfase é tentar reproduzir o espectro de potência do sinal original [3]. Portanto, a qualidade da

fala associada a esse tipo de sistema é determinada mais pela adequação do modelo adotado para a fonte do sinal do que pela precisão na quantização dos parâmetros. Isto significa que a qualidade da fala nos *codecs* de fonte não pode ser melhorada, por exemplo, pelo simples incremento do número de bits de quantização. O fator limitante para a qualidade da fala está essencialmente ligado à fidelidade do modelo usado [36].

Existem vários tipos de *vocoder* (*Vocoder* de Canal, Formante, Homomórfico), mas o que obteve mais êxito foi o *Vocoder* LPC (*Linear Predictive Coding*), assim chamado porque utiliza a codificação linear preditiva. Nesse tipo de *vocoder* o mecanismo de produção da fala é modelado por um sistema linear e lentamente variante no tempo, excitado por um trem de impulsos ou por um sinal aleatório. Esse tipo de sistema fonte tornou-se associado aos métodos de séries temporais auto-regressivas [37] nas quais o trato vocal é representado por um filtro auto-regressivo cujos parâmetros são obtidos pela análise linear preditiva, um processo em que a amostra presente do sinal de fala é predita pela combinação linear de amostras passadas. Nessa técnica, as características significativas da fala são extraídas diretamente da forma de onda temporal e não do espectro de freqüência como fazem os *vocoders* de canal e formante. A excitação ideal para a síntese LPC é o resíduo de predição, uma vez que o resíduo carrega toda a informação não capturada pela análise. Entretanto, na implementação clássica, a excitação é modelada por uma seqüência de impulsos "*pitch*-periódica" para sons vozeados e por uma seqüência aleatória (ruído) para sons não vozeados. Como os parâmetros da excitação e do modelo do trato vocal são atualizados periodicamente, o sintetizador acompanha a característica não estacionária do sinal de fala.

Os codificadores híbridos combinam as características de eficiência de codificação dos *vocoders* com o potencial de qualidade dos codificadores de forma de onda. De modo geral utilizam predição adaptável para descrever o comportamento espectral do trato vocal, juntamente com um modelo adequado para a descrição da excitação [36]. Os codificadores de "análise-pela-síntese", que incorporam a síntese como parte fundamental da análise do sinal, compreendem a família mais importante dentro da categoria de codificadores híbridos [38].

2.3 Breve discussão sobre quantização vetorial

Quantização vetorial é um processo em que o dado a ser quantizado é “quebrado” em blocos, ou vetores, e codificados seqüencialmente, bloco por bloco. O fator decisivo para o método é identificar um conjunto de possíveis vetores (*codebook*) que sejam representativos da informação que se deseja quantizar [39]. Assim, a tarefa do codificador consiste em encontrar o vetor do *codebook* que esteja mais próximo do vetor a ser quantizado. A esse vetor encontrado está associada uma palavra binária, que é usada pelo descodificador para obtê-lo novamente a partir de um *codebook* idêntico, através de simples pesquisa em memória de dados.

De modo mais formal, um quantizador vetorial Q de tamanho N e dimensão k é um processo de mapeamento de um vetor de dimensão k do espaço euclidiano $\boldsymbol{R}^k$ em um conjunto finito C contendo N vetores de saída, conhecidos como *code vectors* ou *codewords*. Então, $Q : \boldsymbol{R}^k \rightarrow C$, em que $C = \{\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_N\}$ e $\boldsymbol{y}_i \in \boldsymbol{R}^k$, para cada $i \in J \equiv \{1, 2, ..., N\}$. O conjunto C é conhecido como *codebook* e tem tamanho N, significando que existem N elementos distintos, cada qual um vetor em $\boldsymbol{R}^k$ [40]. A resolução (ou taxa) dada por $r = (\log_2 N) / k$ mede o número de bits por componente do vetor e fornece uma indicação da precisão conseguida com o quantizador. Cada um dos N vetores está associado a uma partição de $\boldsymbol{R}^k$, ou célula R_i, $i \in J$. Esta célula é definida como:

$$R_i = \{\boldsymbol{x} \in \boldsymbol{R}^k : \mathrm{Q}(\boldsymbol{x}) = \boldsymbol{y}_i\} \tag{2.1}$$

Da definição da célula tem-se que:

$$\bigcup_i R_i = R^k \quad \text{e} \quad R_i \cap R_j = \varnothing \quad \text{para} \quad i \neq j \tag{2.2}$$

Um quantizador vetorial pode ser decomposto em dois componentes básicos, o codificador e o descodificador. O codificador E faz o mapeamento do espaço $\boldsymbol{R}^k$ em um índice do conjunto J e o descodificador D mapeia este índice em um vetor do conjunto de reprodução C. Ou seja, $E : \boldsymbol{R}^k \rightarrow J$ e $D : J \rightarrow \boldsymbol{R}^k$. A partição do espaço em células determina completamente a forma como o codificador associa um dado vetor de entrada a

um índice e não é necessário que o codificador conheça o *codebook* para desempenhar sua função. Por outro lado, o *codebook* define perfeitamente o modo como o descodificador gera o vetor de saída a partir do índice.

Uma classe importante de quantizadores conhecidos como Voronoi, ou *Nearest Neighbor*, tem a característica de que a partição é completamente definida pelo *codebook* e pela medida de distorção, sendo portanto desnecessário o conhecimento da sua descrição geométrica. Assim, um quantizador Voronoi é dado por:

$$R_i = \{\boldsymbol{x} : \mathrm{d}(\boldsymbol{x}, \boldsymbol{y_i}) \leq \mathrm{d}(\boldsymbol{x}, \boldsymbol{y_j}) \text{ para todo } j \in J\} \qquad (2.3)$$

em que d(***x***, ***y***) é uma medida de distorção entre o vetor de entrada ***x*** e o vetor de saída **y**. Especificamente, se a medida de distorção é o erro médio quadrático (d(***x***, ***y***) = || ***x*** - ***y*** ||^2), as partições do quantizador Voronoi são *polytopes*[1] e ficam explicitamente determinadas pelos vetores do *codebook*.

Algoritmos de codificação em quantizador Voronoi sem restrição são tidos como algoritmos de *busca exaustiva*, pois a medida de distorção é calculada para cada vetor do *codebook*, de forma a encontrar aquele que resulta no seu menor valor. Este processo exige grande demanda computacional e a complexidade é função exponencial da resolução *r* e dimensão *k*, pois $N = 2^{rk}$. Essa carga computacional motivou inúmeros estudos, dando origem a diversas técnicas que impõem certas restrições na criação do *codebook*, e assim possibilitam o desenvolvimento de algoritmos eficientes, que encontram o vetor do *codebook* mais próximo sem que seja necessária a busca exaustiva. Estes métodos normalmente comprometem o desempenho em relação à quantização sem restrição, mas geralmente alcançam um compromisso favorável entre complexidade computacional e eficiência [40].

A quantização vetorial estruturada em árvore faz parte deste conjunto de iniciativas e é uma das técnicas mais efetivas no sentido da redução de complexidade. Nela o processo de busca é realizado em estágios, eliminando um subconjunto considerável de vetores

[1] Um quantizador é dito *polytopal* se cada célula da partição de R^k for um *polytope* convexo, ou seja, as faces (de dimensão *k*-1) das células consistem de segmentos de hiperplanos. Especificamente, dado um subespaço $Hv = \left\{x \in R^k : u_v \cdot x + \beta_v \geq 0\right\}$, caracterizado pelo vetor $\boldsymbol{u_v}$ e escalar $\boldsymbol{\beta_v}$, então, qualquer região *polytope* pode ser representada pela interseção finita de subespaços, escrita como: $R_i = \bigcap_{v=1}^{L_i} H_v$, onde L_i corresponde ao número de faces da célula R_i.

candidatos em cada etapa. Em uma árvore *m*-ária, em cada nó, o vetor a ser quantizado é comparado de forma exaustiva com um subconjunto de *m* vetores teste, para determinar aquele que resulta na menor distorção e assim definir o caminho para o próximo estágio. Portanto, para um codebook de tamanho $N = m^d$, o número de operações em cada estágio é proporcional a *m* e então, para uma árvore de altura *d*, a complexidade de busca é proporcional a *md*, e não m^d. A árvore considerada neste trabalho é binária ($m = 2$) e balanceada, ou seja, cada nó pai gera dois nós descendentes, e cada caminho, desde o nó inicial (raiz) até o nó terminal (folha), apresenta o mesmo número de nós.

A estrutura em árvore, por outro lado, acresce os requisitos com armazenamento se comparado ao método de quantização sem restrição. Além de armazenar os m^d vetores do *codebook* (folhas), a estrutura em árvore ainda precisa armazenar os vetores de teste, ou nós intermediários, que são *m* no primeiro estágio, m^2 no segundo e assim sucessivamente. Logo é preciso armazenar $m + m^2 + ... + m^d$, ou

$$\sum_{l=1}^{d} m^l = \frac{m\left(m^d - 1\right)}{m-1} \tag{2.4}$$

A questão de como se compara a quantização TSVQ frente à quantização sem restrição (busca exaustiva) é uma tarefa difícil, e normalmente a saída para esse problema é obter a resposta por meio de simulação (tipicamente a degradação usando TSVQ é modesta, mas não desprezível) [40].

A busca em árvore geralmente não encontra o vetor mais próximo (ou de menor distorção), conforme mostra o exemplo da Fig. 2.2, que é um quantizador de 2 bits em duas dimensões. A linha ***A*** é um hiperplano associado ao nó raiz e as linhas ***B*** e ***C*** são hiperplanos associados aos nós do segundo estágio. Os vetores de teste rotulados com 0 e 1 determinam em que lado de ***A*** o vetor ***X*** se encontra. Fica claro que a escolha do vetor teste 0 no primeiro estágio define, erroneamente, o vetor 0,1 como vetor de quantização de ***X***, embora o vetor 1,0 esteja mais próximo. A Fig. 2.3 mostra a árvore correspondente para este quantizador.

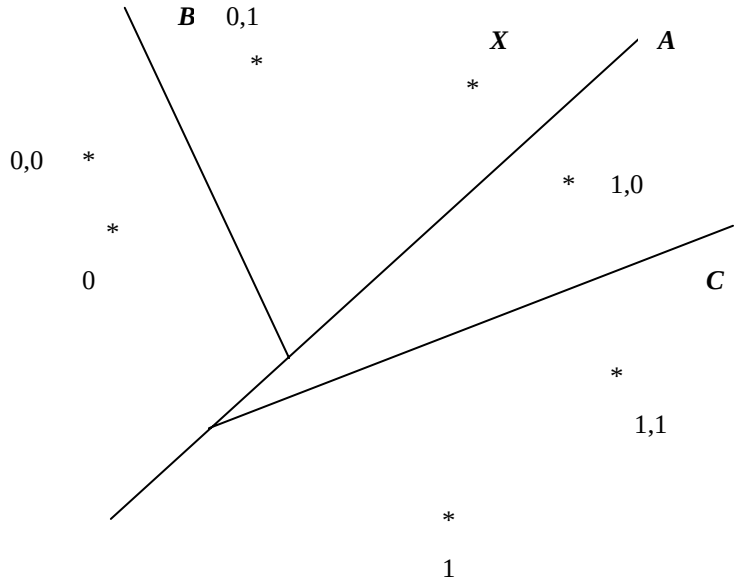


Fig. 2.2 Quantizador usando estrutura em árvore de 2 estágios. O vetor de entrada ***X*** é quantizado com a seqüência 0,1 embora o vetor mais próximo tenha a seqüência 1,0.

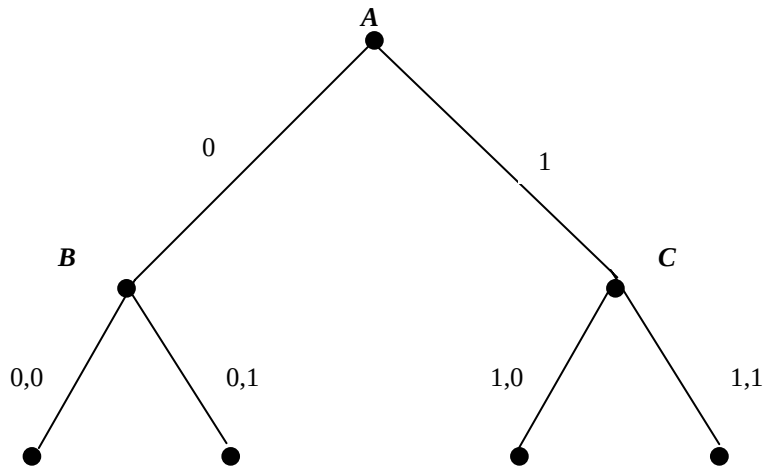


Fig. 2.3 Árvore correspondente.

2.4 Estratégias de codificação com taxa variável

Os codificadores com taxa variável podem ser classificados em duas categorias gerais [41]: os codificadores controlados pela fonte, em que os algoritmos de codificação respondem às características locais do sinal de fala gerando taxas diferenciadas; e os codificadores controlados pela rede, nos quais a taxa é determinada por fatores externos,

tipicamente associados às condições da rede. Esses últimos ainda podem ser subdivididos em multi-modo e *embedded*. Nos codificadores multi-modo, diferentes esquemas de codificação, ou até famílias de codificadores, são empregados para produzir a taxa desejada. Nesses sistemas, mudanças freqüentes de taxa podem exigir a preservação de contexto (e talvez até de algoritmo) entre um esquema de codificação e outro, para que a transição seja suave. O codificador *embedded* opera em uma taxa fixa a partir da qual taxas reduzidas são obtidas pelo simples descarte de bit.

2.4.1 Arquiteturas multi-taxa com base na fonte

Os métodos de codificação controlados pela fonte distribuem os bits dinamicamente, seguindo uma determinada estratégia, de acordo com as características locais da fala, mantendo um nível desejável de qualidade com a menor quantidade de bits possível. Embora a partição da fala em períodos de atividade e não atividade seja um componente essencial dos codificadores de taxa variável, mesmo durante os segmentos de atividade pode-se observar uma variação da entropia de curta duração devido à mudança nas características fonéticas. Assim, o número de bits necessário para representar os segmentos ativos, mantendo uma certa qualidade subjetiva, apresenta grande variação. Portanto, ao invés de transmitirem constantemente em uma taxa fixa máxima (necessária para o pior caso), os codificadores podem atribuir a cada segmento apenas a quantidade de bits necessária, reduzindo a taxa média. Existem várias estratégias de codificação para obter taxa variável, sendo grande parte delas baseada na estrutura CELP (*Code-Excited Linear Prediction*).

Um grande número de abordagens multi-taxa com base na fonte utiliza a classificação fonética como ponto de partida [42-45]. A justificativa para a classificação parte da percepção de que um esquema rígido de codificação, em que o sinal de voz é dividido em quadros de tamanho fixo e submetido a uma estrutura fixa de representação e alocação de bit, apresenta certa deficiência pois dispensa o mesmo tratamento para segmentos da fala que apresentam conteúdos fonéticos distintos [42]. Um exemplo dessa constatação pode ser observado na codificação de um segmento não vozeado da fala. Nesse caso, alocar bits valiosos para modelar a periodicidade de longo atraso é nitidamente um gasto desnecessário. Da mesma forma, a característica não estacionária dos segmentos não vozeados sugere que uma estimativa mais freqüente da envoltória espectral seja

relevante para preservar a integridade fonética daqueles sons. Por outro lado, a resolução espectral (ordem do filtro PCA) para esses segmentos é menos crítica (em termos de percepção fonética) do que para segmentos vozeados. Assim, a classificação fonética procura adaptar o sistema de codificação a cada tipo de segmento da fala de forma que, preservando as características das sucessivas unidades de fonema, o processo de reconstrução resulte em ganhos qualitativos. A estratégia da classificação fonética é viabilizada através da variação de alguns parâmetros, como o tamanho dos quadros, a ordem do filtro de síntese, a presença ou não do preditor de longo atraso, a alocação de bit etc. Algumas estratégias encontradas nestas propostas:

- Utilização de detector de atividade de voz (VAD – *Voice Activity Detection*);
- Para os segmentos vozeados: *codebook* para a excitação; parâmetros do filtro de síntese codificados com um número maior de bits; filtro de síntese de maior ordem; preditor de longo atraso;
- Para segmentos não vozeados: *codebook* específico para a representação espectral enfatizando a região de altas freqüências; filtro de síntese de baixa ordem devido à pouca correlação entre amostras; sem preditor de longo atraso;
- Para segmentos de transição: é um tipo de segmento de baixa predição e muito importante para a distinção das consoantes. Para este tipo de segmento o ouvido é pouco sensível a erros de espectro e assim a ênfase recai sobre a codificação da excitação, através da redução da dimensão do seu vetor. Os parâmetros espectrais são tratados da mesma forma que nos segmentos não vozeados e, da mesma forma, o preditor de longo atraso não é empregado.

Seguindo a mesma filosofia apresentada anteriormente [42], porém com várias modificações com o intuito de melhorar a qualidade, reduzir a complexidade, atraso e taxa de bit, Wang e Gersho [43] apresentam um codificador com taxa média de 3,0 kbit/s e pico de 3.4 kbit/s. Dentre as várias alterações, introduzem duas excitações para codificar as transições fonéticas, cuja codificação é uma das maiores limitações do CELP/VXC (*Vector Excitation Coders*) convencional.

A proposta de Di Francesco et al. [46] para obtenção de taxa variável explora a segmentação da fala com base nas suas características estatísticas e codificação usando a estrutura CELP. Antes da classificação dos segmentos ativos da fala, um algoritmo detecta os períodos de atividade e define se um segmento é silêncio ou não. Posteriormente, cada uma das três classes: vozeada, não vozeada passa-baixa e não vozeada passa-alta, é

codificada por um algoritmo específico usando *codebooks* algébricos na excitação. Em linhas gerais, a diferença de tratamento dos segmentos vozeados para os não vozeados é que no caso destes últimos não se calcula os parâmetros do preditor de longo atraso e codifica-se o preditor de curto atraso com ordem reduzida.

Das e Gersho [47] aplicam a estratégia da classificação fonética em *vocoder* de excitação multi-banda (MBE – *Multiband Excitation*) na proposta de um *codec* de taxa variável operando de 0,15 a 2,6 kbit/s, com média de 1,4 kbit/s. Cada segmento é classificado como silêncio, ruído, não vozeado (U), vozeado misto (MV) ou totalmente vozeado (FV). As classes com atividade de fala (U, MV, FV) têm características espectrais distintas e, portanto, a elas são aplicados diferentes modelos paramétricos. A classe vozeada mista é a mais complexa em termos de representação por apresentar um espectro misto, com picos em intervalos irregulares, que lembra um pouco a característica dos formantes, e partes planas, semelhante ao espectro de um ruído. Nessa classe são alocados bits em todos os parâmetros (*pitch*, *voicing vector*, *spectral gain*, *spectral shape*). Na classe totalmente vozeada, apenas os parâmetros correspondentes ao *pitch* e ao conteúdo espectral (*spectral gain*, *spectral shape*) são transmitidos. Nas classes não vozeada e ruído são estimados apenas os parâmetros do conteúdo espectral. Seguindo essa linha, a técnica é aperfeiçoada com a inclusão da quantização vetorial com dimensão variável (VDVQ - *Variable Dimension Vector Quantization*) no codificador MBE, resultando no esquema de codificação chamado EMBE (*Enhanced Multiband Exitation*) [48]. O esquema VDVQ permite a quantização dos parâmetros espectrais com um número reduzido de bits, mas preservando as características subjetivas mais importantes.

A estratégia de McClellan e Gibson [49] para codificação multi-taxa, usando a arquitetura CELP, consiste em determinar a banda através de medida da monotonicidade espectral em laço aberto. A taxa de processamento dos segmentos é constante, ou seja, o comprimento do segmento não varia conforme sua classificação. Assim, a variação da taxa ocorre pela alocação de bit diferenciada para os parâmetros espectrais e vetores de excitação, conforme as medidas de monotonicidade. O codificador divide os segmentos em quatro modos: silêncio, transição (silêncio para vozeado), banda cheia e meia banda. O tamanho do *codebook* da excitação fixa varia conforme o modo, uma vez que segmentos classificados como silêncio ou meia banda requerem menos bits do que os de transição ou banda cheia. Além disso, a proposta não aloca bits para o preditor de longo atraso (*pitch*)

nos modos silêncio e transição e limita a quantização vetorial dos parâmetros do filtro de síntese para os segmentos classificados como meia banda e silêncio.

Vaseghi [50] apresenta um sistema de codificação multi-taxa baseado em um arranjo de *codecs* CELP em paralelo com diferentes padrões de quantização dos parâmetros LPC e tamanho do *codebook*. A proposta explora basicamente a codificação variável da excitação (que para os sistemas CELP representa cerca de 70% dos bits). Argumenta que sinais de voz altamente correlacionados podem ser reproduzidos com uma excitação relativamente pobre. Por outro lado, mais bits são necessários para representar adequadamente a excitação durante os segmentos de voz com característica não vozeada ou nas transições. O sistema, chamado *Finite State – CELP* (baseado no conceito de FSVQ - *Finite State Vector Quantization*), seleciona o estado de codificação conforme as características do sinal de entrada, a taxa de bit para a relação sinal-ruído desejada (SNR – *Signal-to-Noise Ratio*) e o estado corrente. Estados de baixa taxa de bit, principalmente usados para codificar segmentos fortemente correlacionados, possuem *codebooks* de excitação pequenos. Por outro lado, estados de alta taxa são usados para codificar transientes da fala e segmentos com componentes de alta freqüência. Paksoy e Gersho [51] apesar de considerarem a proposta interessante apontam certa ineficiência neste codificador, especificamente relativa à codificação dos segmentos não vozeados, que no FS-CELP consomem grande parte dos bits, o que contraria os resultados experimentais comumente conhecidos em termos da percepção.

Um codificador com taxas de 8, 4 ou 1 kbit/s baseado na especificação G.729 [52] é a proposta de Chung et al. [53]. Um esquema de classificação fonética determina se o segmento de fala é vozeado, não vozeado (que também inclui segmentos de transição) ou silêncio, para os quais são designadas as taxas de 8, 4 e 1 kbit/s respectivamente. A taxa máxima utiliza a própria especificação G.729 para tamanho de segmento e subquadro, esquema de quantização e alocação de bit. Para a taxa de 4 kbit/s é usado apenas um subquadro (expandido para 10 ms) com a metade dos bits alocados para cada parâmetro da excitação fixa e adaptável, em relação à especificação original. Os parâmetros do filtro de síntese também têm redução de bit. Na taxa mínima apenas os parâmetros do filtro de síntese, ganho de excitação e uma semente para um gerador de número aleatório são codificados, com poucos bits. As excitações fixa e adaptável são substituídas por um vetor pseudo-aleatório cujo gerador é o mesmo para o codificador e descodificador.

Beritelli [54] propõe uma versão modificada do algoritmo CS-ACELP num codificador com 8 taxas, desde 0 até 8 kbit/s. A proposta é bastante focada na questão do ruído de fundo e para tal inclui um classificador fonético robusto, com 4 níveis de classificação e 8 classes fonéticas. Os segmentos mistos e vozeados (classes 7 e 8) são codificados basicamente seguindo a especificação G.729. As seis classes restantes são dedicadas à codificação dos períodos de inatividade da fala (silêncio e ruído de fundo) e distribuem os bits entre alguns parâmetros: filtro de síntese (para caracterização do ruído acústico), ganho de excitação e índice para o *codebook* de excitação. Esses parâmetros nem sempre estão presentes em cada uma das seis classes.

Um esquema de codificação que também apresenta taxa escalonável aparece na proposta de Dong e Gibson [55], que empregam o conceito de refinamento sucessivo para codificadores baseados no modelo CELP. Trata-se de uma estrutura em duas camadas. Na primeira, um codificador CELP é usado de forma transparente (na proposta testou 3 codificadores: um codificador CELP de 3,65 kbps, o G.723.1 e o G.729), ao mesmo tempo em que sua saída descodificada é subtraída do sinal original para gerar a entrada da camada seguinte. Assim, o sinal de erro de reconstrução da primeira etapa é a excitação da segunda etapa, que utiliza um código em árvore, adaptável, e erro médio quadrático. A camada de codificação do erro pode ser ativada apenas nos segmentos vozeados ou indiscriminadamente. A variabilidade da taxa aparece em função do tipo de codificador usado na primeira camada, do grau de refinamento da segunda camada e do tipo de segmento usado no refinamento (se vozeado ou todos).

O codificador QCELP [56], baseado no modelo CELP, seleciona uma de suas quatro taxas (8, 4, 2 e 1 kbps) a cada 20 ms de acordo com a atividade do sinal da fala. A atividade da fala é codificada à taxa de 8 kbps enquanto o silêncio e o ruído de fundo são codificados com as demais taxas. A mudança de taxa ocorre através da medida do ruído de fundo comparada a três valores de referência. Se o nível ficar acima dos três valores a codificação de taxa máxima é aplicada. Se ficar abaixo de todos os níveis é usada a codificação de taxa mínima e se ficar entre os extremos, são usadas as codificações de taxas intermediárias.

Outra estratégia que resulta em variação de taxa é a alocação dinâmica de bits. Jayant e Chen [57] mostram os benefícios de distribuir dinamicamente os bits entre os parâmetros da excitação e os coeficientes LPC, para uma mesma taxa, sem modificar o algoritmo de codificação. O estudo foi realizado sobre a estrutura de codificação CELP

com três taxas: 4.8, 6.4, e 8.0 kbit/s. A escolha da melhor distribuição ocorre pelo teste exaustivo das possibilidades, minimizando duas medidas de distorção. Os resultados mostram que apenas a alocação de bit não aparenta ser a forma eficiente de conseguir boa qualidade em baixas taxas, mas reforça a necessidade de se ter flexibilidade de distribuição para atender melhor às características não estacionárias de produção da fala.

2.4.2 Arquiteturas multi-taxa com base na rede

Um codificador multi-modo com taxas de 4,7 e 6,5 kbit/s, com controle baseado nas condições de canal, é apresentado por Hanzo e Woodard [58]. O codificador usa a estrutura ACELP, obtendo variação de taxa por meio da partição diferenciada dos subquadros das excitações fixa e adaptável. O segmento para a determinação dos parâmetros do filtro de síntese tem duração de 30 ms, e é subdividido em 4 subquadros de 7,5 ms para a taxa de 4,7 kbit/s e em 6 subquadros de 5 ms para a taxa de 6,5 kbit/s.

Erdmann, Bauer e Vary [59] trazem o conceito da codificação piramidal de imagem para o campo da fala. Aplicada a imagem, a codificação piramidal consiste em representá-la em múltipla resolução, em que cada nível corresponde a uma camada da pirâmide. A maior resolução encontra-se no topo, enquanto a menor fica na base da pirâmide. Na proposta de Erdmann et al., um algoritmo de codificação gera um fluxo de dados hierarquicamente estruturado a uma taxa fixa a partir da qual taxas reduzidas podem ser extraídas. A escalabilidade de taxa surge da aplicação da codificação em pirâmide (Laplace) sobre o sinal residual da codificação CELP. Ao invés de codificar o resíduo seguindo o modelo CELP, o sinal de erro é decomposto em quatro camadas, a camada base e mais três, de resolução progressivamente crescente. O resultado da codificação sobre segmentos de 20 ms de fala é distribuído em três quadros UDP (*User Datagram Protocol*), cada qual contendo as informações básicas (parâmetros do filtro de síntese, *pitch* e ganho, e camada base da pirâmide) e uma das camadas de refinamento.

A especificação AMR (*Adaptive Multi-Rate*) surge com o objetivo de atender a um conjunto de requisitos de qualidade, atraso e complexidade para a codificação da fala em situações adversas (ruído de fundo, degradação de canal etc). O AMR é um *codec* multi-taxa, baseado no modelo ACELP, em que a variabilidade da taxa resulta basicamente da forma como os diferentes parâmetros são quantizados [60]. Os segmentos têm comprimento de 20 ms, divididos em 4 subquadros de 5 ms, com 5 ms de *lookahead*. A

Tab. 2.1 mostra a distribuição de bits entre os diversos parâmetros do codificador. Observa-se que a mudança significativa na taxa está relacionada com a alocação de bits para codificar os parâmetros do *codebook* fixo. Essa variação é possível devido à mudança no número de pulsos não-zero de cada *codeword* (de 10 a 2 pulsos). Apesar de a especificação abranger um conjunto de oito taxas, o AMR por si só não é um *codec* multi-taxa completo [33] na medida em que não seleciona sua taxa de operação e também não inclui um algoritmo de seleção. Nesse sentido, várias propostas têm sido apresentadas [61-65], tanto para aplicações em que a variação da taxa é função da rede quanto para aquelas em que a variação da taxa é função da fonte.

Modo	**Parâmetro**	**Subquadro**				**Total**
		1	**2**	**3**	**4**	
12.2 kbit/s	LSF					38
	Codebook Adaptável	9	6	9	6	30
	Ganho do *Codebook* Adaptável	4	4	4	4	16
	Codebook Fixo	35	35	35	35	140
	Ganho do *Codebook* Fixo	5	5	5	5	20
10.2 kbit/s	LSF					26
	Codebook Adaptável	8	5	8	5	26
	Codebook Fixo	31	31	31	31	124
	Ganhos	7	7	7	7	28
7.95 kbit/s	LSF					27
	Codebook Adaptável	8	6	8	6	28
	Ganho do *Codebook* Adaptável	4	4	4	4	16
	Codebook Fixo	17	17	17	17	68
	Ganho do *Codebook* Fixo	5	5	5	5	20
7.40 kbit/s	LSF					26
	Codebook Adaptável	8	5	8	5	26
	Codebook Fixo	17	17	17	17	68
	Ganhos	7	7	7	7	28
6.70 kbit/s	LSF					26
	Codebook Adaptável	8	4	8	4	24
	Codebook Fixo	14	14	14	14	56
	Ganhos	7	7	7	7	28
5.90 kbit/s	LSF					26
	Codebook Adaptável	8	4	8	4	24
	Codebook Fixo	11	11	11	11	44
	Ganhos	6	6	6	6	24
5.15 kbit/s	LSF					23
	Codebook Adaptável	8	4	4	4	20
	Codebook Fixo	9	9	9	9	36
	Ganhos	6	6	6	6	24
4.75 kbit/s	LSF					23
	Codebook Adaptável	8	4	4	4	20
	Codebook Fixo	9	9	9	9	36
	Ganhos	8		8		16

Tab. 2.1 – Distribuição de bits no *codec* AMR

2.4.3 Arquiteturas multi-taxa com base na fonte ou na rede

Um codificador baseado na estrutura CELP com variação de taxa controlada pela fonte ou pela rede é apresentado por Lupini et al. [66]. A proposta explora a grande porcentagem de silêncio em uma conversação e a variação da entropia dos segmentos de fala. Todas as configurações são baseadas na taxa mais alta, obtendo-se as taxas menores pela variação do comprimento dos segmentos e subquadros, usando *codebooks* menores para quantização e até mesmo não ativando componentes do *codec*. Os vetores de excitação, esparsos e ternários, se sobrepõem para a redução de complexidade, e são escolhidos usando-se análise-pela-síntese em *codebook* de múltiplo estágio.

A abordagem de Cellario e Sereno [67] para a codificação multi-taxa também apresenta seleção dupla, pela fonte ou pela rede. A variação da taxa é conseguida pela classificação da fala e pela utilização de estruturas de codificação específicas para cada classe. O codificador possui sete taxas de operação, de 400 a 16 kbit/s. O esquema de codificação é baseado no algoritmo CELP, com duplo *codebook* fixo. A classificação final dos segmentos de fala é concluída em duas etapas. Na primeira, em laço aberto, o segmento é classificado como ruído de conforto, vozeado ou não vozeado. Na segunda etapa, o algoritmo de classificação cria duas subclasses para o ruído de conforto, duas para os segmentos vozeados e três para os não vozeados. De modo geral, nos segmentos classificados como ruído de conforto apenas o nível do ruído e o conteúdo espectral são estimados, dependendo da subclasse. Nos segmentos não vozeados, apenas o conteúdo espectral e as excitações fixas são avaliados, também segundo a subclasse. Finalmente, nos segmentos vozeados as subclasses criam arranjos entre o conteúdo espectral, o preditor de longo atraso e duas excitações (*codebook* fixo). A estrutura determinística da excitação fixa dispensa armazenamento e torna eficiente o processo de busca.

A proposta de codificação híbrida multi-modo / multi-taxa de Ruggeri et al. [68] está fundamentada no trabalho anterior de Beritelli [54] e nas duas extensões do *codec* G.729, os anexos D (6,4 kbit/s) e E (11,8 kbit/s) [69, 70]. A maior mudança dos anexos D e E, em termos de alocação de bit, em relação ao padrão, reside no *codebook* fixo. O anexo D é basicamente o G.729 com *codebook* reduzido e com quantização menos elaborada dos ganhos e do atraso de *pitch*. O anexo E é o padrão com maior alocação de bits para o *codebook*. O novo *codec* também apresenta quatro níveis de classificação, porém com nove classes fonéticas.

Nas classes vozeadas e mistas são empregados os codificadores padrão G.729 em 6,4, 8 e 11,8 kbit/s. A classe não vozeada é modelada por um filtro LPC excitado por um ruído gaussiano. Duas classes são reservadas para modelar o ruído de fundo não estacionário, que recebe tratamento semelhante aos segmentos não vozeados, com filtro LPC e *codebook* para o ruído acústico. Quatro classes modelam o ruído de fundo estacionário, que distribuem os bits entre os parâmetros do *codebook* e o ganho. A abordagem também permite a variação de taxa segundo as condições da rede. Nessa situação, o codificador combina os modos de codificação em quatro categorias usadas conforme as condições de carga na rede.

Uma abordagem diferente para obter variação de taxa, baseada na transformada discreta do cosseno (DCT - *Discrete-Cosine Transform*) e *run-length coding*, é apresentada por Kwong et al. [71].

2.5 Conclusão

Este capítulo mostrou diversas estratégias relacionadas à codificação multi-taxa, tanto aquelas dirigidas pela fonte quanto as orientadas pelas condições de carga no canal. Apesar da diversidade de propostas, é possível identificar certas estratégias comuns, bastante aceitas e experimentadas. Nessas estratégias, estão incluídas por exemplo: a detecção da atividade da fala, mais ou menos robusta quanto ao ruído de fundo; a classificação fonética como ponto de partida para a codificação sob medida de cada segmento de voz; a segmentação variável, para conseguir melhor foco naquelas porções da fala que exibem riqueza de eventos, como as transições da fala não vozeada para a vozeada; a utilização de *codebook* específico, filtro de síntese de baixa ordem e omissão do preditor de longo atraso para os segmentos não vozeados; a codificação adequada do ruído de fundo.

Dentre as diversas estratégias de atuação sobre o problema, a que fica mais evidente é a quase unanimidade na adoção da abordagem CELP (e suas variações) para a codificação. A justificativa resulta do grande êxito dessa proposta, haja vista a abundância de padrões estruturados sobre ela: TIA VSELP, TIA QCELP, G.728, G.729, G.723.1 [55]; MPEG-4, US 1016, GSM, IS-54, IS-136 [71], AMR. A escolha da especificação G.729 (CS-ACELP) como base para a proposta desse trabalho partiu dessa constatação (as especificações mais recentes: G.729, G.723.1 a 5.3 kbit/s, GSM Full-Rate, IS-136 e AMR

empregam a busca ACELP) e da larga utilização desse padrão nas comunicações VoIP. O próximo capítulo, por conseqüência, trata especificamente da codificação CS-ACELP do padrão G.729.

C A P Í T U L O 3

CODIFICAÇÃO ACELP

3.1 Introdução

A proposição ACELP, assim como a grande maioria dos *codecs* padronizados, tem como base o processo de análise-síntese na estrutura de codificação. Essa abordagem, por sua vez, está vinculada à disponibilidade de modelos para a representação da fala, conhecidos desde o final da década de 50 com o trabalho de Fant [72].

Na etapa da análise, a fala é examinada e dela é extraído um conjunto de parâmetros representativos do modelo adotado. Esses parâmetros são então quantizados e transmitidos ao descodificador que, juntamente com o modelo em questão, reconstrói uma versão aproximada da fala original, encerrando a etapa da síntese. Quando o mecanismo de extração e quantização ocorre em laço fechado, minimizando explicitamente uma medida de distorção (usualmente o erro médio quadrático) entre a fala original e a reconstruída, o processo passa a ser conhecido como análise-pela-síntese, pois incorpora a síntese como parte da análise (o modelo ACELP adota esta estratégia).

Para compreender melhor o esquema de codificação ACELP e situar a proposta deste trabalho é importante rever os princípios da análise-pela-síntese.

3.2 Princípios da análise-pela-síntese

Na forma clássica de síntese a fala é modelada como a saída de um sistema linear, lentamente variante no tempo, excitado por um trem de impulsos quase-periódicos ou por um ruído. Neste sistema, mostrado no diagrama de blocos da Fig. 3.1, o mecanismo de geração do som (excitação) está linearmente separado do mecanismo de conformação, desempenhado pelo filtro de síntese que representa o trato vocal. Neste modelo a forma da excitação depende do tipo de som, que se divide em fala vozeada e não vozeada. Para os sons classificados como fala vozeada (como no /*a*/ de *a*ve) a excitação é representada por uma seqüência de impulsos quase-periódica, enquanto para os sons classificados como fala não vozeada (como no /*f*/ de *f*aca) a excitação consiste simplesmente de uma seqüência de

amostras aleatórias, do tipo ruído branco (a distribuição de probabilidade das amostras do ruído parece não ser crítica [73]).

Desta forma, o espectro de curta duração do sinal da fala sintetizado resulta da multiplicação do espectro da excitação pela resposta em freqüência do filtro de síntese. Independente do tipo de excitação, seja ela um trem de impulsos ou ruído branco, sua envoltória espectral é plana, mostrando que o conteúdo espectral do sinal de curta duração da fala é determinado basicamente pela resposta em freqüência do filtro de síntese.

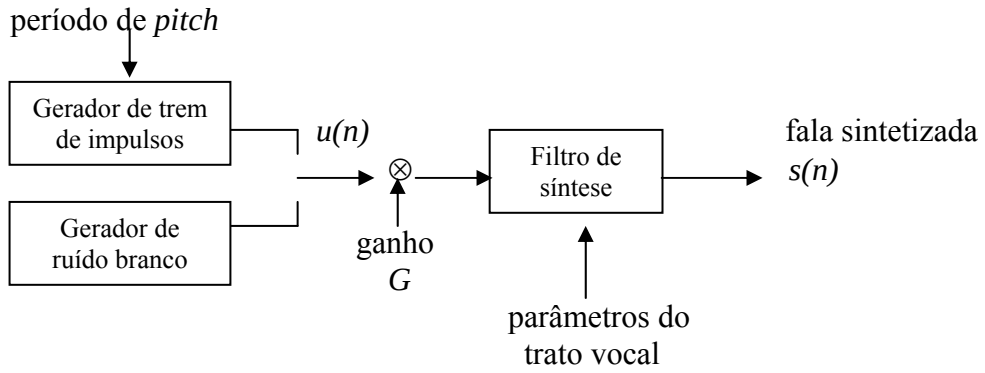


Fig. 3.1 Diagrama de blocos simplificado do modelo de produção da fala

Para este modelo, as amostras da fala sintetizada *s(n)* estão relacionadas com a excitação *u(n)* através da equação de diferenças

$$s(n) = \sum_{k=1}^{p} a_k s(n-k) + Gu(n) \tag{3.1}$$

em que a_k representa os coeficientes do filtro de síntese, p a ordem e G o ganho aplicado à excitação. Portanto, o filtro de síntese, que representa o trato vocal, tem a seguinte função de transferência:

$$H(z) = \frac{S(z)}{U(z)} = \frac{G}{1 - \sum_{k=1}^{p} a_k z^{-k}} \tag{3.2}$$

Esse modelo contendo apenas pólos é a representação natural para os sons vozeados. Porém, para os sons nasalados e fricativos a representação adequada deveria incluir zeros na função de transferência. Entretanto, se a ordem *p* for suficientemente grande, o modelo contendo apenas pólos fornece uma boa representação para a quase totalidade dos sons da fala [74].

A representação da fonte nesta forma está intimamente relacionada à modelagem auto-regressiva (AR) de processos estocásticos estacionários (compare as equações 3.1 e 3.3), que satisfaz a equação de diferenças:

$$u(n) = \sum_{k=1}^{M} w_k^* u(n-k) + v(n) \tag{3.3}$$

em que a série temporal *u(n)*, *u(n-1)*, ..., *u(n-M)* representa a realização de um processo auto-regressivo de ordem *M*, w^*_k são os parâmetros de regressão (* denota a operação complexo conjugado) e *v(n)* é um ruído branco de média zero e variância constante.

A grande importância da representação de fonte mostrada na Fig. 3.1, posta sua relação com a modelagem AR, é que os coeficientes de regressão de um processo AR (e por analogia os coeficientes do filtro de síntese do modelo da Fig. 3.1) podem ser estimados usando a análise linear preditiva, cuja idéia básica é a de que o valor futuro de um processo estocástico estacionário discreto pode ser estimado a partir da observação de um conjunto de amostras passadas deste mesmo processo. Ou seja, é possível estimar os parâmetros do modelo de produção da fala diretamente de suas amostras.

Essa possibilidade surge, por sua vez, devido à equivalência matemática entre a modelagem auto-regressiva de processos estocásticos estacionários e a operação do filtro de erro de predição *forward* (as equações de Wiener-Hopf, da predição linear, têm a mesma forma matemática das equações de Yule-Walker, da modelagem AR). Esta equivalência decorre da complementaridade das operações [73], como ilustrado a seguir:

- A operação do filtro de erro de predição aplicada a um processo estocástico estacionário *u(n)* pode ser vista como um processo de **análise**. Nessa operação, mostrada na Fig. 3.2, se a ordem *M* do preditor for suficientemente grande o processo erro de predição *f(n)*, na saída do filtro, será composto por amostras não

correlacionadas (operação de "branqueamento" do processo *u(n)*). Quando isso ocorre, o processo original *u(n)* fica representado por um conjunto de coeficientes $\{w^*_{fk}$, para $k = 1, 2, ..., M\}$ e pela potência do erro $\{P\}$.

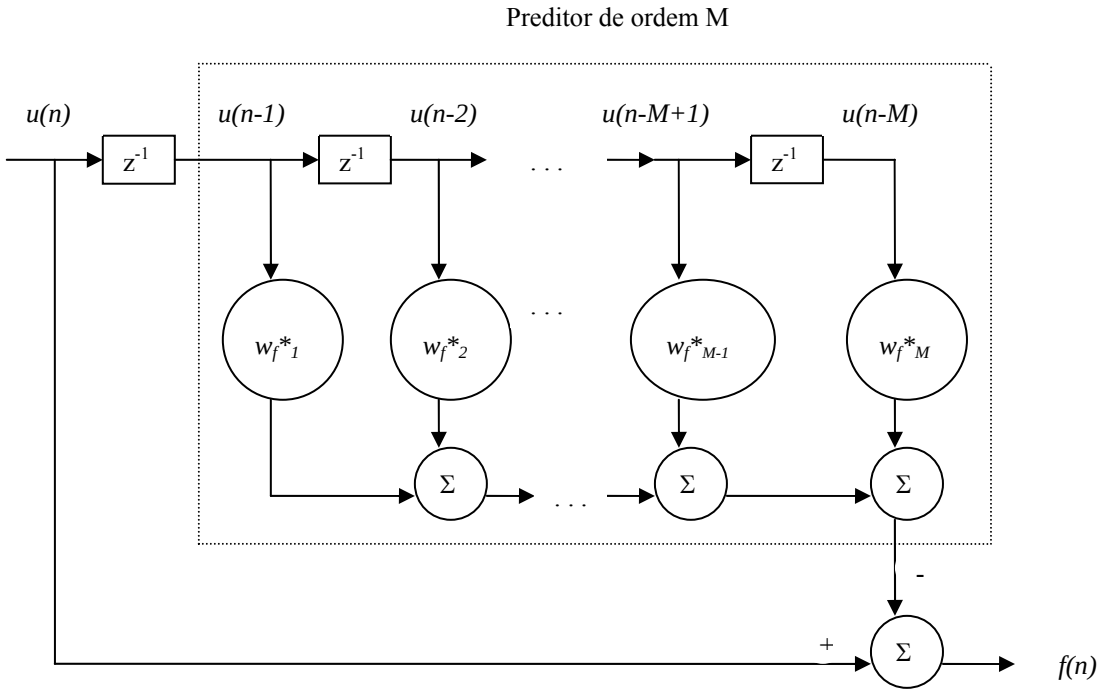


Fig. 3.2 Filtro de erro de predição (apenas zeros na função de transferência)

O erro de predição é dado por,

$$f(n) = u(n) - \sum_{k=1}^{M} w^*_{fk} u(n-k) \tag{3.4}$$

Tomando $a_k = -w_{fk}$, para $k = 1, 2, ..., M$ e $a_0 = 1$, como sendo os coeficientes do filtro de erro de predição, a equação (3.4) pode ser escrita como soma única

$$f(n) = \sum_{k=0}^{M} a^*_k u(n-k) \tag{3.5}$$

- A modelagem auto-regressiva de um processo estacionário *u(n)* pode ser vista como um processo de **síntese**. Nessa operação, mostrada na Fig. 3.3, pode-se gerar um processo AR *u(n)* aplicando um processo ruído branco *v(n)*, de média zero e variância σ^2_v, na entrada de um filtro inverso cujos coeficientes são iguais aos parâmetros w^*_k, $k = 1, 2, ..., M$ do modelo AR.

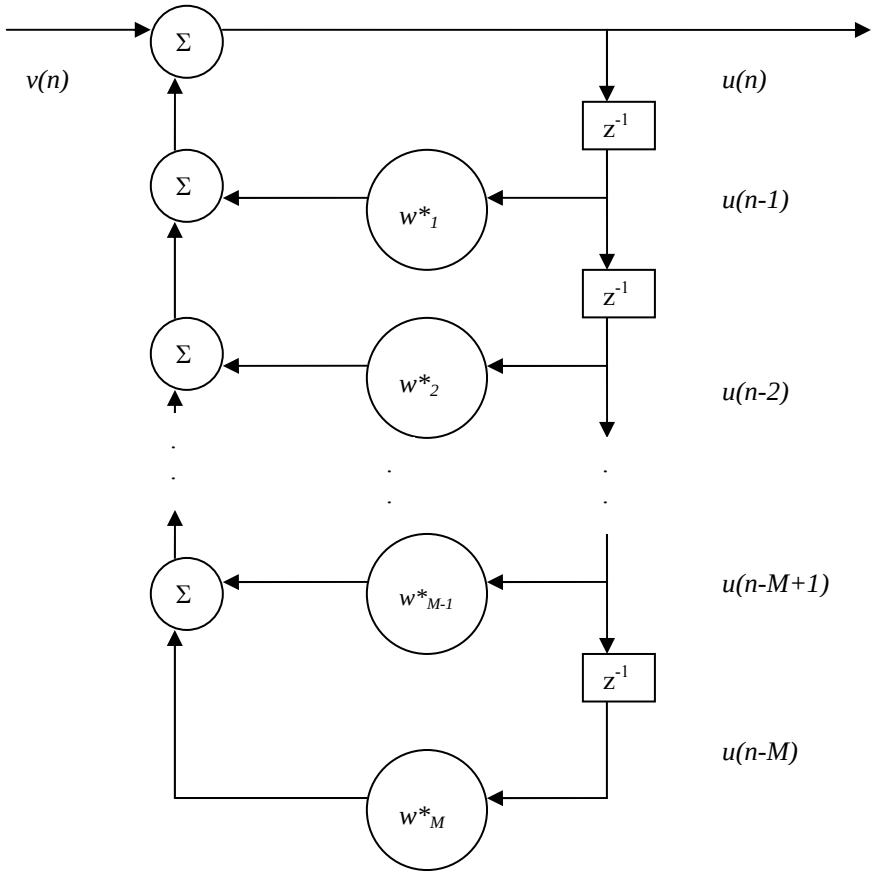


Fig. 3.3 Modelo auto-regressivo (apenas pólos na função de transferência)

O sinal *u(n)* é descrito pela equação (3.3). As duas estruturas constituem um par "casado", com seus parâmetros obedecendo à relação: $w_{fk} = w_k$ para $k = 1, 2, ..., M$ e $P = \sigma^2_v$.

Portanto, dado um processo auto-regressivo de ordem *M* pode-se afirmar que, quando o preditor *forward* estiver otimizado em termos do erro médio quadrático, seus coeficientes ponderadores assumem os mesmos valores dos parâmetros correspondentes do processo AR. Uma característica importante decorrente desse caminho para o cálculo dos coeficientes de regressão (filtro de síntese) é que o filtro de erro de predição é de fase

mínima, ou seja, os zeros da função de transferência estão localizados no interior do círculo de raio unitário do plano *z*, da mesma forma que os pólos da função de transferência do filtro inverso, assegurando a estabilidade deste.

A formulação de Wiener-Hopf aplicada ao problema da predição linear *forward* leva a um conjunto de equações descritas na forma compacta

$$\mathrm{R}\mathrm{w}_f = \mathrm{r} \tag{3.6}$$

em que, $\mathbf{R} = E[\mathbf{u}(n-1)\mathbf{u}^H(n-1)]$ é à matriz de autocorrelação do vetor de entrada de amostras do sinal da fala $\mathbf{u}(n-1)$ ($E[.]$ denota o valor esperado), definido como:

$$\mathbf{u}(n-1) = \left[u(n-1), u(n-2), ..., u(n-M)\right]^T \tag{3.7}$$

portanto,

$$\mathbf{R} = \begin{bmatrix} r(0) & r(1) & \dots & r(M-1) \\ r^*(1) & r(0) & \dots & r(M-2) \\ \vdots & \vdots & \ddots & \vdots \\ r^*(M-1) & r^*(M-2) & \dots & r(0) \end{bmatrix} \tag{3.8}$$

e $r(k)$ é a autocorrelação do processo de entrada $u(n)$ no ponto k.

O vetor $\mathbf{r} = E[\mathbf{u}(n-1)\mathrm{u}^*(n)]$ é o vetor de correlação cruzada entre o vetor de entrada $\mathbf{u}(n-1)$ e a resposta desejada $u(n)$, assim

$$\mathbf{r} = \begin{bmatrix} r(-1) \\ r(-2) \\ \vdots \\ r(-M) \end{bmatrix} \tag{3.9}$$

Finalmente $\mathbf{w}_f$ corresponde ao vetor ótimo dos coeficientes do preditor *forward* de ordem M, que por conseqüência determina os coeficientes do processo AR e assim do próprio filtro de síntese. Portanto,

$$\mathbf{w}_f = \left[w_{f1}, w_{f2}, ..., w_{fM} \right]^T \tag{3.10}$$

A abordagem geralmente usada para resolver o conjunto de equações de Wiener-Hopf (3.6), inclusive na especificação G.729, emprega o algoritmo de Levinson-Durbin [73, 74]. Esse método recursivo explora a natureza *Toeplitz* da matriz de correlação para o cálculo dos coeficientes preditores (também gera os coeficientes de reflexão, assim chamados devido à analogia com a teoria das linhas de transmissão). É importante notar que a estabilidade teórica do filtro de síntese *H(z)*, garantida pelo método, pode não ser confirmada na prática caso a estimativa da função autocorrelação[2], *r(k)*, do processo de entrada não tenha exatidão suficiente [74]. Além do método da autocorrelação, existem outras formas de resolver o conjunto de equações lineares, como por exemplo a decomposição de Cholesky do *método da covariância* [74]. Existe ainda a formulação *em treliça*, que combina as etapas do cálculo da matriz de correlação e a resolução das equações em um único algoritmo recursivo que determina os coeficientes preditores [73, 74].

Conforme exposto, o filtro de erro de predição atua como um descorrelacionador do processo de entrada e, portanto, seu resíduo é a excitação perfeita para o filtro de síntese, pois carrega toda a informação não capturada pela análise linear preditiva [37]. Entretanto, embora a codificação do resíduo com alta qualidade seja um critério suficiente para que se tenha qualidade elevada na fala reconstruída, esse critério não se constitui uma condição necessária [36]. A codificação através da **análise-pela-síntese** procura reduzir a taxa de bits sem a codificação direta do resíduo e ainda com boa qualidade da fala reconstruída. Nessa estratégia de codificação, os parâmetros do sistema são determinados através da análise linear preditiva e a seqüência de excitação através de um processo de otimização em laço fechado, dando origem ao nome. O processo de otimização determina a seqüência de excitação que minimiza uma medida de diferença ponderada entre a fala original, de entrada, e a fala sintetizada. O "ponderador" na verdade é um filtro, escolhido de forma que o codificador seja otimizado para o processo de audição humano. O diagrama de blocos da Fig. 3.4 mostra a estrutura típica de um codificador/descodificador que emprega

---

[2] Na prática a estimativa polarizada é preferível sobre a não polarizada porque resulta em menor variância dos valores estimados *r'(k)* para valores de *k* próximos do comprimento *N* do segmento do sinal de fala [73]. A estimativa polarizada é dada por: $r'(k) = \frac{1}{N} \sum_{n=1+k}^{N} s(n)s^*(n-k) \quad k = 0, 1, ..., M$

a análise-pela-síntese. O sistema de codificação é composto por um filtro preditor de curto atraso (filtro de síntese PCA), cuja função é representar a estrutura dos formantes da fala (determina a envoltória espectral); um preditor de longo atraso (filtro de síntese PLA), que determina a estrutura de *pitch* (detalhes do espectro); o filtro ponderador do erro, que funciona como um "conformador", de forma que o ruído de quantização fique mascarado pela grande energia dos formantes e permita um maior erro de reconstrução nessa região (o resultado é que a relação sinal-ruído é aproximadamente constante sobre o espectro de freqüência da fala sintetizada ao invés da constância na densidade espectral de potência do ruído de quantização [36]); e finalmente o gerador de excitação, que seleciona a melhor seqüência através da minimização do erro médio quadrático ponderado. O preditor de curto atraso geralmente é atualizado entre 30 e 100 vezes por segundo, enquanto no de longo atraso a atualização é mais freqüente, entre 60 e 200 vezes por segundo [37]. Além disso, técnicas de quantização vetorial são empregadas, tanto na codificação dos parâmetros dos filtros de síntese quanto na codificação da excitação.

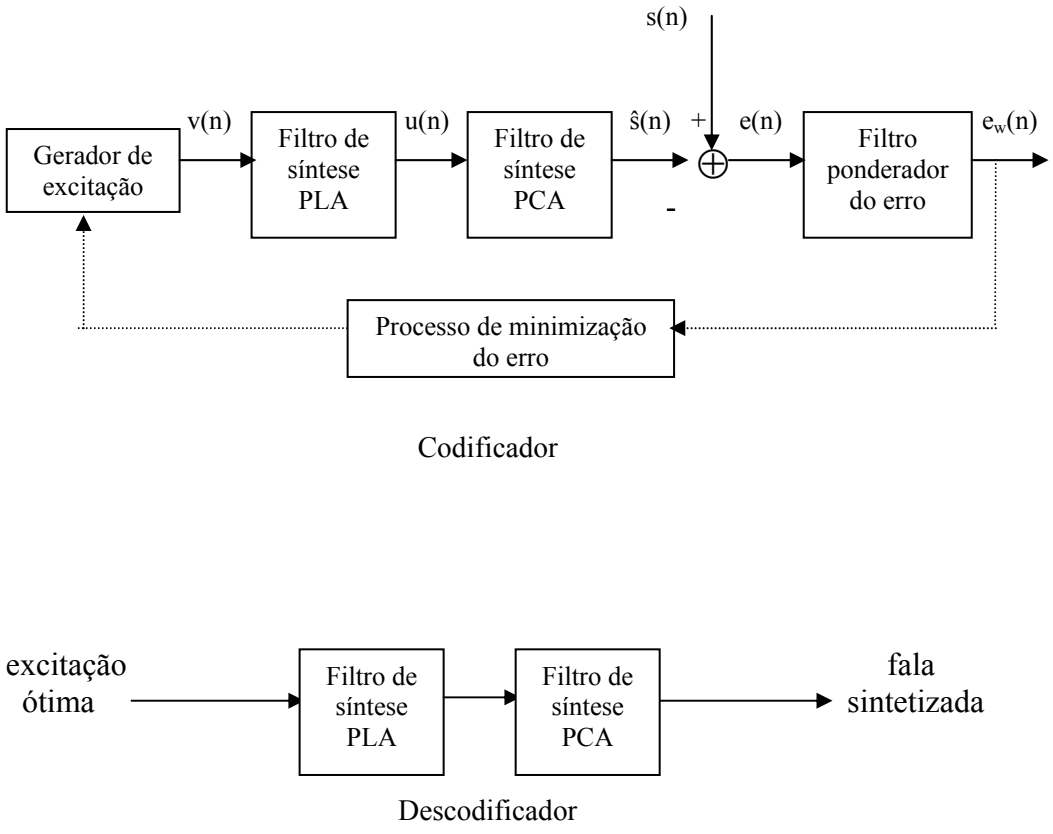


Fig. 3.4 Estrutura típica de um codificador/descodificador baseado na análise-pela-síntese

O descodificador usa uma estrutura idêntica à estrutura dos filtros de síntese do codificador, só que o processo é consideravelmente mais simples pois é excitado apenas com o vetor ótimo recebido do codificador e não passa, portanto, por todo o processo de busca do melhor vetor.

Antes de abordar a estrutura ACELP é interessante considerar alguns aspectos importantes de cada bloco que compõe a estrutura de análise-pela-síntese bem como sua descrição matemática.

3.2.1 Filtro Preditor de Curto Atraso (PCA)

O objetivo do filtro preditor de curto atraso é modelar a envoltória espectral do sinal da fala, e portanto determinar a estrutura dos formantes. O preditor atua reconstruindo a correlação entre amostras adjacentes presentes no sinal original. Embora seja capaz de reconstruir a redundância entre amostras, o filtro PCA falha na predição adequada da quase periodicidade dos segmentos vozeados, não conseguindo modelar os picos de longo atraso presentes no resíduo.

Rabiner e Schafer [74] mostram que a escolha da ordem do preditor de curto atraso depende primeiramente da taxa de amostragem, e efetivamente independe do método de predição utilizado. Uma vez que o conteúdo espectral da fala geralmente pode ser representado com uma densidade média de 2 pólos (um pólo complexo-conjugado) por kHz devido à banda passante do trato vocal, então para uma taxa de amostragem igual a F_s kHz (F_s é o dobro da banda passante) seriam necessários F_s pólos para representar o trato vocal. Além disso, para representar com qualidade os sons da fala é preciso uma taxa de amostragem maior ou igual a 20 kHz (10 formantes). Por outro lado, a resposta em freqüência de uma linha de transmissão telefônica tipicamente atenua o sinal na ordem de 20 dB para a freqüência de 3.5 kHz, o que leva à taxa de amostragem de 8 kHz (4 formantes) e portanto 8 pólos. Ainda demonstram que, para uma dada freqüência de amostragem, o erro de predição decresce com o aumento da ordem, mas tende a se estabilizar depois de um determinado ponto. Por exemplo, para uma taxa de amostragem de 10 kHz, o erro se estabiliza com a ordem entre 13 e 14. Hanzo, Somerville e Woodard [36] simulam o efeito da variação da ordem no ganho de predição[3] e mostram que o ganho

[3] Definido como a energia das amostras do segmento original da fala dividida pela energia das amostras do erro de predição [36].

aumenta à medida que a ordem do preditor aumenta. Entretanto, como no modelo *forward* cada coeficiente do filtro de síntese precisa ser enviado ao descodificador, a escolha da ordem igual a 10 representa um compromisso razoável entre ter um ganho de predição elevado e ter uma baixa taxa de bit.

Os coeficientes do filtro PCA são calculados de forma a minimizar o erro de predição dentro de um determinado intervalo de amostras de tamanho N. Segundo Rabiner e Schafer [74], como a carga computacional para os métodos de análise linear preditiva é proporcional ao comprimento do segmento, é interessante manter N tão pequeno quanto possível. A mudança das propriedades estatísticas dos sinais da fala também contribui para que o comprimento se mantenha o menor possível. Por outro lado, como no método da autocorrelação o sinal é "janelado", o comprimento deve ser grande o suficiente para que o efeito do achatamento nos extremos dos segmentos de voz não comprometa os resultados. Além disso, para se ter uma indicação de periodicidade na função autocorrelação o segmento deve ter comprimento suficiente para representar no mínimo dois períodos da forma de onda (a medida em que k aumenta, a autocorrelação $r_n(k)$ tem menos dados envolvidos no processamento). Tendo em vista esses compromissos antagônicos, implementações de análise LPC usando o método da autocorrelação utilizam segmentos que variam entre 100 e 400 amostras (para amostragem de 10 kHz). Hanzo, Somerville e Woodard [36] apresentam o resultado de simulações para analisar o efeito do comprimento do segmento sobre o ganho de predição. Usando um preditor com ordem fixa igual a 10, mostram que o melhor valor para o ganho de predição ocorre para o segmento de 160 amostras (20 ms amostrando em 8 kHz). O desenvolvimento matemático para determinar os coeficientes do filtro PCA obedece às formulações apresentadas na seção 3.2.

3.2.2 Filtro Preditor de Longo Atraso (PLA)

O objetivo do filtro preditor de longo atraso é modelar os detalhes do conteúdo espectral do sinal da fala, melhorando a qualidade subjetiva do sinal sintetizado. O preditor consegue estabelecer uma periodicidade de longo atraso na excitação do filtro PCA, relacionada com o *pitch* dos segmentos vozeados, ou seja, consegue modelar a redundância de longo atraso e portanto remover aqueles picos presentes no resíduo quando apenas o filtro PCA está em ação. A redundância de longo atraso no sinal sintetizado reduz, conseqüentemente, a variância do resíduo e proporciona maior economia na taxa de bit.

Para os segmentos não vozeados o filtro PLA simplesmente determina a excitação do filtro PCA como combinação de excitações passadas altamente correlacionadas com a excitação desejada [75].

3.2.2.1 Determinação dos parâmetros do filtro PLA em laço aberto

Basicamente, se a periodicidade do *pitch* é quase estacionária então uma forma de reduzir a redundância de longo atraso do resíduo PCA é subtrair do segmento do resíduo atual o segmento anterior adequadamente posicionado. A variância do resíduo PLA pode ser reduzida ainda mais se antes de subtrair o resíduo PCA passado for aplicado um ganho G ao segmento, que pode ser otimizado de forma a reduzir a energia do resíduo PLA. O resíduo PLA de um estágio, $e_L(n)$, pode ser expresso como:

$$e_L(n) = r(n) - G_1 r(n-\alpha) \tag{3.11}$$

em que, $r(n)$ corresponde ao resíduo PCA, G_1 é o ganho e α o atraso.

A periodicidade do *pitch* é fortemente atrelada ao locutor (valores típicos estão entre 100 e 300 Hz, ou entre 3 e 10 ms). Além disso, o resíduo PLA é altamente não predizível e normalmente pode ser modelado a partir de um conjunto (*book*) de seqüências aleatórias gaussianas de média zero e variância unitária, resultando em uma forma eficiente de quantização vetorial. Este conceito deu origem à abordagem CELP, importante membro da família de codificadores de análise-pela-síntese. No descodificador, a seqüência escolhida passa pelo filtro de síntese para reconstruir o sinal da fala. A melhor seqüência não necessariamente se assemelha ao resíduo PLA, tampouco garante que as formas de onda da fala original e sintetizada terão a melhor semelhança. Na verdade, a seqüência escolhida objetiva sintetizar a fala com a melhor qualidade subjetiva.

O erro quadrático médio total E_L do resíduo PLA sobre um segmento de N amostras pode ser formulado como [36]:

$$E_L = \frac{1}{N}\sum_{n=0}^{N-1} e_L^2(n) \tag{3.12}$$

$$= \frac{1}{N}\sum_{n=0}^{N-1} [r(n) - G_1 r(n-\alpha)]^2 \tag{3.13}$$

O ganho G_1 é determinado fazendo $\partial E_L / \partial G_1 = 0$, o que resulta em:

$$G_1 = \frac{\sum_{n=0}^{N-1} r(n)r(n-\alpha)}{\sum_{n=0}^{N-1}[r(n-\alpha)]^2} \tag{3.14}$$

O ganho pode ser interpretado como a correlação cruzada normalizada de *r(n)*, em que o denominador representa a energia do segmento do resíduo PCA. Da equação (3.14), se o segmento atual e o anterior forem perfeitamente correlacionados então $G_1 = 1$. Por outro lado, se praticamente não existir correlação entre os segmentos, como é o caso para a fala não vozeada, $G_1 \approx 0$. Uma vez conhecido o ganho em laço aberto pode-se calcular a energia residual mínima,

$$E_{L\min} = \sum_{n=0}^{N-1} r^2(n) - \frac{\left[\sum_{n=0}^{N-1} r(n)r(n-\alpha)\right]^2}{\sum_{n=0}^{N-1}[r(n-\alpha)]^2} \tag{3.15}$$

Assim, minimizar o erro E_L equivale a maximizar o segundo termo da equação (3.15). Portanto, o atraso ótimo pode ser determinado calculando o termo acima para todos os valores de α dentro do intervalo típico de 20 a 147 amostras (amostragem em 8 kHz).

3.2.2.2 Determinação dos parâmetros do filtro PLA em laço fechado

A qualidade da fala sintetizada pode ser melhorada significativamente se os parâmetros do filtro PLA forem calculados dentro do laço de análise-síntese (ao custo de uma maior complexidade) [76]. É dessa estratégia que resulta o termo *codebook* adaptável, conforme mostrado na Fig. 3.5 [36].

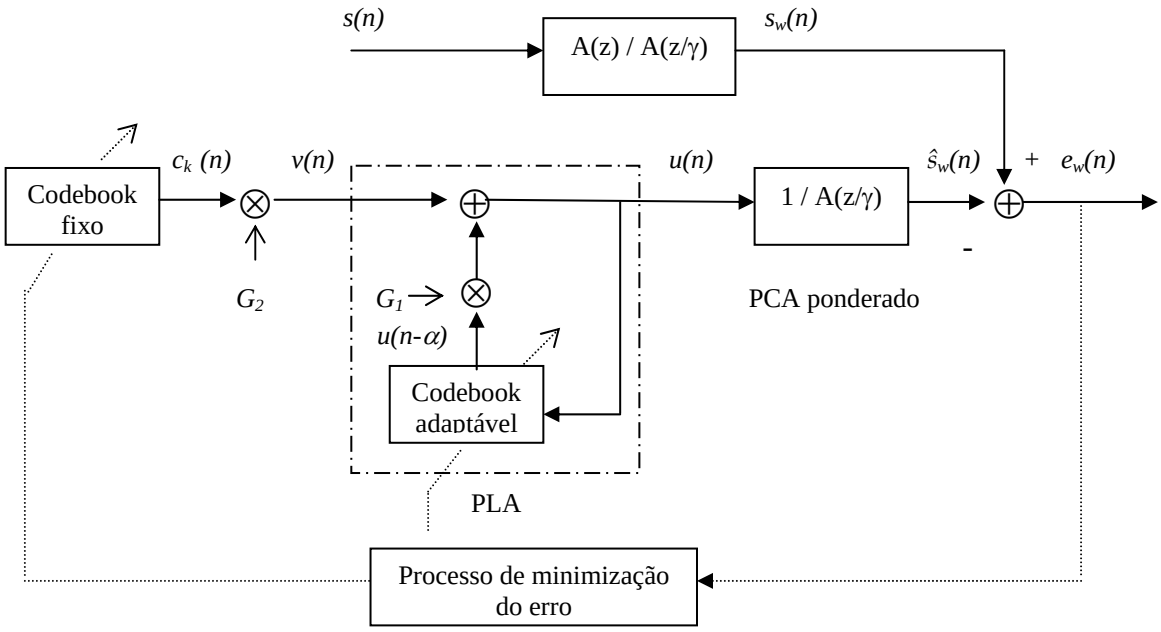


Fig. 3.5 Codificador de análise-pela-síntese com a estratégia de *codebook* adaptável

A excitação composta do filtro PCA é dada por: $u(n) = v(n) + G_1u(n-\alpha)$, que é a superposição da excitação do filtro PLA com a excitação do próprio filtro PCA ponderado, atrasada e escalonada. O filtro PCA ponderado ($1/A(z/\gamma)$) é a junção do filtro PCA ($1/A(z)$) com o filtro ponderador do erro ($A(z)/A(z/\gamma)$), detalhado na seção 3.2.4, e $s_w(n)$ é a fala original depois de passar pelo filtro ponderador. O deslocamento do filtro ponderador do erro é justificável porque torna mais simples a determinação dos parâmetros do *codebook* e é possível porque se trata de um filtro linear [36]. Em *codecs* de análise-pela-síntese convencionais a excitação fixa $v(n)$ é recalculada para cada subquadro, tipicamente de 5 a 7,5 ms. Portanto este procedimento se repete de 2 a 6 vezes para cada quadro PCA (de 10 a 30 ms) e proporciona uma flexibilidade extra para o *codec* acompanhar a natureza não estacionária do sinal da fala.

Idealmente, a excitação composta ótima $u(n)$, que resultaria na fala sintetizada de melhor qualidade subjetiva, deveria ser encontrada conjuntamente, testando todas as combinações possíveis de suas componentes. Infelizmente, tal procedimento implicaria em uma carga computacional excessiva. Então, normalmente se utiliza uma estratégia alternativa sub-ótima, segundo a qual primeiramente se calcula os parâmetros do *codebook* adaptável assumindo que no início não há excitação fixa no filtro de síntese, ou seja, $v(n) \approx 0$. Assim, no estágio inicial a excitação $u(n)$ torna-se $\approx G_1u(n-\alpha)$ [36].

Durante o processo de escolha da melhor excitação *u(n)*, que resulta no segmento de fala sintetizada de melhor qualidade subjetiva, cada seqüência candidata é submetida ao filtro de síntese ponderado, que é um filtro cuja resposta ao impulso tem duração infinita (IIR – *Infinite Impulse Response*). Desta forma, a fala sintetizada ponderada ($\hat{s}_w(n)$) no intervalo de otimização corrente pode ser descrita como a superposição da resposta do filtro à seqüência de excitação corrente com a resposta do filtro a todas as seqüências de excitação ótimas passadas. Essa contribuição da memória não é influenciada pela seqüência de excitação corrente e normalmente é conhecida como a resposta do filtro IIR à entrada zero. Já que essa memória não é influenciada pela excitação corrente, a forma eficiente de se conduzir a otimização é subtraí-la da fala original ponderada antes da comparação das seqüências candidatas, já que essa operação ocorre uma única vez [77]. Após a subtração chega-se a um vetor alvo, que deve ser comparado com todas as seqüências candidatas filtradas. Entretanto, nesta estratégia, a contribuição da memória do filtro IIR deve ser iniciada com zero antes de cada excitação candidata ser filtrada, já que o efeito dela foi considerado pela modificação do sinal da fala original ponderado. Assim, a fala sintetizada é dada por:

$$\hat{s}_w(n) = \sum_{i=0}^{n} u(i) h_w(n-i) + \hat{s}_0(n) \tag{3.16}$$

em que $h_w(n)$ corresponde à resposta IIR do filtro de síntese ponderado, $\hat{s}_0(n)$ é a resposta do filtro de síntese ponderado à entrada zero (memória do filtro às excitações ótimas passadas). Assim, o erro ponderado fica definido como:

$$e_w(n) = x'(n) - (\hat{s}_w(n) - \hat{s}_0(n)) \tag{3.17}$$

em que $x'(n) = s_w(n) - \hat{s}_0(n)$ representa o sinal da fala original ponderado, depois da subtração da memória do filtro de síntese resultante das excitações ótimas passadas. Então,

$$e_w(n) = x'(n) - \sum_{i=0}^{n} u(i) h_w(n-i) \tag{3.18}$$

Como na abordagem sub-ótima inicialmente $v(n)$ é zero,

$$= x'(n) - G_1 \sum_{i=0}^{n} u(i-\alpha) h_w(n-i)$$

$$= x'(n) - G_1 y_\alpha(n) \tag{3.19}$$

em que,

$$y_\alpha(n) = \sum_{i=0}^{n} u(i-\alpha) h_w(n-i) \tag{3.20}$$

O erro quadrático médio ponderado para o segmento de *N* amostras é dado por:

$$E_w = \frac{1}{N} \sum_{n=0}^{N-1} [x'(n) - G_1 y_\alpha(n)]^2 \tag{3.21}$$

Expandindo E_w e fazendo $\delta E_w / \delta G_1 = 0$ fica,

$$G_1 = \frac{\sum_{n=0}^{N-1} x'(n) y_\alpha(n)}{\sum_{n=0}^{N-1} [y_\alpha(n)]^2} \tag{3.22}$$

Uma vez conhecido o ganho em laço fechado pode-se calcular o erro ponderado mínimo,

$$E_w = \sum_{n=0}^{N-1} [x'(n)]^2 - \frac{\left[\sum_{n=0}^{N-1} x'(n) y_\alpha(n)\right]^2}{\sum_{n=0}^{N-1} [y_\alpha(n)]^2} \tag{3.23}$$

Aqui vale o procedimento análogo ao do laço aberto, ou seja, o atraso α é encontrado maximizando o segundo termo da equação (3.23) para todos os valores de α no intervalo desejado.

3.2.3 Modelos de excitação

Os modelos clássicos de excitação para os *codecs* que utilizam a análise-pela-síntese são: MPE *(Multi-Pulse Excited)*, RPE *(Regular Pulse Excited)* e CELP *(Code-Excited Linear Prediction)* [37].

No modelo MPE, proposto por Atal e Remde [78], a excitação consiste em múltiplos pulsos espaçados de maneira não uniforme. Na etapa de análise, a posição e a amplitude desses pulsos dentro do segmento são encontradas seqüencialmente, de forma sub-ótima, um pulso por vez, minimizando o erro médio quadrático ponderado. A qualidade da fala reconstruída nos *codecs* que utilizam a excitação MPE depende muito da quantidade de pulsos, que por outro lado é restringida pela taxa de bits necessária para transmitir seus parâmetros (geralmente são usados entre 4 e 6 pulsos para cada 5 ms) [37].

O RPE é similar ao MPE, mas os pulsos ocorrem a intervalos regulares. Portanto, basta o codificador determinar a posição do primeiro pulso, o espaço entre eles e todas as amplitudes. Assim, para uma dada taxa, os codecs RPE podem usar mais pulsos do que os codecs MPE. Por exemplo, para a taxa de 10 kbps os codecs RPE usam 10 pulsos, enquanto os codecs MPE usam apenas 4. Isto faz com que os codecs RPE tenham uma qualidade ligeiramente melhor do que os codecs MPE, porém com complexidade ligeiramente maior [36].

Embora os codecs MPE e RPE possam proporcionar alta qualidade a taxas de 10 kbps, não são apropriados quando se deseja taxas significativamente menores, devido à grande quantidade de informação sobre a posição e a amplitude dos pulsos [36]. O modelo CELP foi a proposta de Schroeder e Atal [79] para representar a excitação de forma eficiente em *codecs* de baixa taxa e alta qualidade, e difere das outras na medida em que a excitação é quantizada vetorialmente. A melhor excitação é representada por um índice que determina a sua posição em um dicionário (*codebook*) de seqüências aleatórias gaussianas. A desvantagem dessa proposta reside no alto esforço computacional para a busca da melhor excitação.

O algoritmo CELP se tornou uma referência a partir da qual sucessivos avanços [80-87], no sentido de resolver o problema da complexidade de busca da melhor excitação, resultaram na maior parte dos padrões de codificação atuais e, por essa razão, é importante rever o desenvolvimento matemático da determinação dos parâmetros da excitação.

3.2.3.1 Determinação dos parâmetros da excitação do modelo CELP

Nessa etapa, a excitação do filtro de síntese assume sua forma completa, ou seja, $u(n) = G_2 c_k(n) + G_1 u(n-\alpha)$. Seguindo o diagrama da Fig. 3.5, e considerando a forma completa da excitação, a equação (3.16) fica,

$$\hat{s}_w(n) = \sum_{i=0}^{n} G_2 c_k(i) h_w(n-i) + \sum_{i=0}^{n} G_1 u(i-\alpha) h_w(n-i) + \hat{s}_0(n) \tag{3.24}$$

Assim, o erro ponderado pode ser escrito como:

$$e_w(n) = S_w(n) - (\hat{s}_0(n) + G_1 Y_\alpha(n)) - G_2 c_k(n) * h_w(n)$$

$$= \tilde{x}(n) - G_2 c_k(n) * h_w(n) \tag{3.25}$$

em que $\tilde{x}(n) = s_w(n) - \hat{s}_o(n) - G_1 y_\alpha(n)$ é o sinal alvo para a busca no *codebook* fixo; $c_k(n)$ é a seqüência do *codebook* fixo; G_2 é o ganho do *codebook* fixo; $h_w(n)$ é a resposta do filtro PCA ponderado; $\hat{s}_0(n)$ é a resposta do filtro PCA ponderado à entrada zero; $s_w(n) - \hat{s}_0(n)$ representa o sinal da fala de entrada ponderado depois da subtração da memória do filtro PCA ponderado devido a todas as excitações ótimas passadas; G_1 é o ganho do *codebook* adaptável; $Y_\alpha(n)$ é a saída filtrada do *codebook* adaptável. O erro quadrático médio fica:

$$E_w = \frac{1}{N} \sum_{n=0}^{N-1} \left(\tilde{x}(n) - G_2 \left[c_k(n) * h_w(n) \right] \right)^2 \tag{3.26}$$

Expandindo E_w , e fazendo $\delta E_w / \delta G_2 = 0$, resulta no ganho ótimo para uma dada *codeword* c_k,

$$G_2 = \frac{\sum_{n=0}^{N-1} \tilde{x}(n) \left[c_k(n) * h_w(n) \right]}{\sum_{n=0}^{N-1} \left[c_k(n) * h_w(n) \right]^2} = \frac{\tilde{C}_k}{\xi_k} \tag{3.27}$$

em que $\tilde{C}_k$ representa a correlação entre o sinal alvo e a seqüência de excitação filtrada e ξ_k a energia da seqüência de excitação filtrada.

Durante o processo de otimização, $\tilde{C}_k$ e ξ_k são calculados para cada seqüência c_k. Substituindo o ganho quantizado $\hat{G}_2$ de volta na equação (3.26) fica,

$$E_w = \frac{1}{N}\sum_{n=0}^{N-1}\left(\tilde{x}(n) - \hat{G}_2\left[c_k(n) * h_w(n)\right]\right)^2$$

$$= \frac{1}{N}\sum_{n=0}^{N-1}\left(\tilde{x}^2(n) - 2\hat{G}_2\tilde{C}_k + \tilde{G}_2^2\xi_k\right)$$

$$= \frac{1}{N}\sum_{n=0}^{N-1}\left(\tilde{x}^2(n) - \hat{G}_2\left(2\tilde{C}_k - \hat{G}_2\xi_k\right)\right) \qquad (3.28)$$

O termo $\hat{G}_2\left(2\tilde{C}_k - \hat{G}_2\xi_k\right)$ é calculado para cada seqüência candidata e o índice *k* que maximiza seu valor é o escolhido. Este índice juntamente com o ganho quantizado são os parâmetros do *codebook* fixo enviados ao descodificador. A maior parte da complexidade na codificação CELP vem do cálculo de $\tilde{C}_k$ e ξ_k para cada entrada do *codebook*. A forma de cálculo destes parâmetros pode ser modificada sob a forma [36]:

$$\tilde{C}_k = \sum_{n=0}^{N-1}\psi(n)c_k(n) \qquad (3.29)$$

e

$$\xi_k = \sum_{i=0}^{N-1}c_k^2(i)\phi(i,i) + 2\sum_{i=0}^{N-2}\sum_{j=i+1}^{N-1}c_k(i)c_k(j)\phi(i,j) \qquad (3.30)$$

em que

$$\psi(i) = \sum_{n=i}^{N-1}\tilde{x}(n)h_w(n-i) \qquad \text{para i = 0, ..., N-1}$$

e

$$\phi(i,j) = \sum_{n=\max(i,j)}^{N-1}h_w(n-i)h_w(n-j) \qquad \text{para i, j = 0, ..., N-1}$$

$\psi(i)$ e $\phi(i, j)$ são calculados uma vez a cada subquadro. O erro quadrático médio agora pode ser expresso como:

$$E_{\min} = \sum_{n=0}^{N-1} x^2(n) - \frac{\left[\sum_{i=0}^{N-1} \psi(i) c_k(i)\right]^2}{\sum_{i=0}^{N-1} c_k^2(i)\phi(i,i) + 2\sum_{i=0}^{N-2}\sum_{j=i+1}^{N-1} c_k(i)c_k(j)\phi(i,j)} = \sum_{n=0}^{N-1} x^2(n) - \frac{\left(\tilde{C}_k\right)^2}{\xi_k} \qquad (3.31)$$

Novamente, a melhor seqüência de excitação é aquela que maximiza o segundo termo da equação (3.31).

3.2.4 Filtro ponderador do erro

O filtro ponderador do erro concentra ruído de quantização na região dos formantes, aproveitando a característica do ouvido humano, que tem a capacidade de mascarar um sinal de espectro espalhado (como o ruído de quantização) quando está presente na mesma região espectral um sinal de grande energia e espectro concentrado (como os formantes) [88]. A escolha razoável para a função de transferência do filtro, considerando nossa característica auditiva, é fazê-la depender da característica espectral do sinal da fala, que é representada pelos coeficientes do filtro de síntese a_i. A forma geral da função de transferência do filtro ponderador é:

$$W(z) = \frac{A(z)}{A(z/\gamma)} = \frac{1 - \sum_{k=1}^{p} a_k z^{-k}}{1 - \sum_{k=1}^{p} a_k \gamma^k z^{-k}} \qquad (3.32)$$

em que γ determina o grau de ponderação do filtro (valores típicos estão entre 0,6 e 0,85) e *A(z)* é a função de transferência do filtro de síntese. Para γ = 1 o filtro ponderador não tem qualquer influência, enquanto para γ = 0 o filtro reduz-se ao inverso do filtro de síntese, conformando o erro de forma recíproca ao envelope espectral do sinal de voz [75].

3.2.5 Quantização dos coeficientes do filtro PCA

O objetivo da quantização dos coeficientes do filtro PCA é representá-los com a menor quantidade de bits possível, sem que ocorra degradação sensível na qualidade do sinal sintetizado. A quantização dos coeficientes pode ser realizada isoladamente

(quantização escalar) ou em conjunto (quantização vetorial), neste caso reduzindo significativamente a taxa de bit ao custo de aumento na complexidade computacional.

É do conhecimento prático, também, a necessidade de se ter alta resolução no quantizador, para que a estabilidade do filtro de síntese continue garantida. Tal requisito pode ser alcançado através da quantização dos coeficientes PARCOR (k_i) ou dos coeficientes de reflexão, obtidos como subproduto no método de Levinson-Durbin [73, 74]. A estabilidade de *H(z)* é assegurada se $| k_i | < 1$. Entretanto, para valores $| k_i | \approx 1$ é necessário um quantizador bastante preciso pois *H(z)* torna-se muito sensível aos erros de quantização. Uma forma de tornar os coeficientes menos sensíveis à quantização é usar a transformação do seno inverso ou o logaritmo da razão de área [37] ($LAR = \log (1- k_i) / (1 + k_i)$). Outra opção, derivada dos coeficientes de reflexão e do filtro *A(z),* é a quantização dos parâmetros LSFs (*Line Spectral Frequencies*), também conhecidos na forma alternativa como LSPs (*Line Spectrum Pairs*) [36].

3.3 Algoritmo CS-ACELP

A descrição do algoritmo a seguir está baseada essencialmente na especificação G.729 [52] e na referência [89]. Como a proposta deste trabalho envolve principalmente a excitação fixa do codificador, a ela será dedicada mais atenção, cabendo uma explanação breve dos demais componentes.

O algoritmo CS-ACELP é o resultado da junção de dois grupos de pesquisa, France Telecom CNET (França) / *University of Sherbrook* (Canadá) e NTT (Japão), para atender aos requisitos do grupo 15 (SG15) do ITU-T envolvido na padronização de um algoritmo de codificação da fala à taxa de 8 kb/s. Esse trabalho resultou no padrão G.729, aprovado sob a resolução No. 1 do WTSC (*World Telecommunication Standardization Conference)* em 19 de março de 1996.

O algoritmo CS-ACELP está baseado no modelo CELP e opera com segmentos de fala de 10 ms (80 amostras, na taxa de amostragem de 8000 amostras/s). A cada segmento o sinal de fala é analisado e dele são extraídos os parâmetros do modelo CELP (coeficientes do filtro de síntese de curto atraso, índices das excitações fixa e adaptável e ganhos), que são codificados e transmitidos, conforme a distribuição de bits da Tab. 3.1.

No destino, esses parâmetros são decodificados para recuperar os coeficientes dos blocos de excitação e filtro de síntese. A fala é reconstruída fazendo a excitação recomposta passar pelo filtro de síntese de curto atraso e depois por um pós-filtro, para melhorar a qualidade subjetiva.

Parâmetro	Palavra de código	Subquadro 1	Subquadro 2	Total
LSP (*Line Spectrum Pairs*)	L0, L1, L2, L3			18
Atraso do *codebook* adaptável	P1, P2	8	5	13
Paridade do atraso de *pitch*	P0	1		1
Índice do *codebook* fixo	C1, C2	13	13	26
Sinal do *codebook* fixo	S1, S2	4	4	8
Ganhos do *codebook* (estágio 1)	GA1, GA2	3	3	6
Ganhos do *codebook* (estágio 2)	GB1, GB2	4	4	8
Total				80

Tab. 3.1 Distribuição de bit do algoritmo CS-ACELP (segmento de 10 ms)

3.3.1 O codificador

O processo de codificação está representado na Fig. 3.6. O sinal de entrada passa inicialmente por uma etapa de pré-processamento, composta de um filtro passa-alta e um ajuste de ganho. A partir deste sinal pré-processado é feita a análise linear preditiva (LP), que determina os coeficientes do filtro de síntese de curto atraso. Estes coeficientes são convertidos em *Line Spectrum Pairs* e quantizados, usando quantização vetorial preditiva de dois estágios com 18 bits. A excitação é escolhida através de um processo de busca usando análise-pela-síntese, onde o erro entre a fala original e a sintetizada é minimizado segundo uma medida de distorção ponderada, que incorpora as características da audição humana através de um filtro ponderador cujos parâmetros derivam dos coeficientes não quantizados do filtro de síntese de curto atraso.

Os parâmetros da excitação (*codebook* fixo e adaptável) são determinados para cada subquadro de 5 ms (40 amostras). Neste procedimento, os coeficientes da análise LP, quantizados e não quantizados, são usados para o segundo subquadro, enquanto para o primeiro são usados apenas os coeficientes LP interpolados. O sinal alvo para a busca no *codebook* adaptável é calculado subtraindo a resposta do filtro de síntese ponderado à entrada zero (*zero-input response*) do sinal de fala ponderado. Os parâmetros do *codebook* adaptável são encontrados em duas etapas: na primeira, o atraso de *pitch* em laço aberto é

estimado usando o segmento de entrada de 10 ms após a ponderação subjetiva. Na segunda etapa, o índice do *codebook* adaptável (atraso) e o ganho são encontrados usando a estratégia de laço fechado no entorno do atraso determinado em laço aberto. O índice do *codebook* adaptável é codificado com 8 bits para o primeiro subquadro e 5 bits diferencial para o segundo. O sinal alvo para a busca no *codebook* fixo é atualizado subtraindo a contribuição devido ao *codebook* adaptável. O *codebook* fixo é algébrico e utiliza 17 bits para representar a excitação. Finalmente, os ganhos dos *codebooks* fixo e adaptável são quantizados vetorialmente com 7 bits.

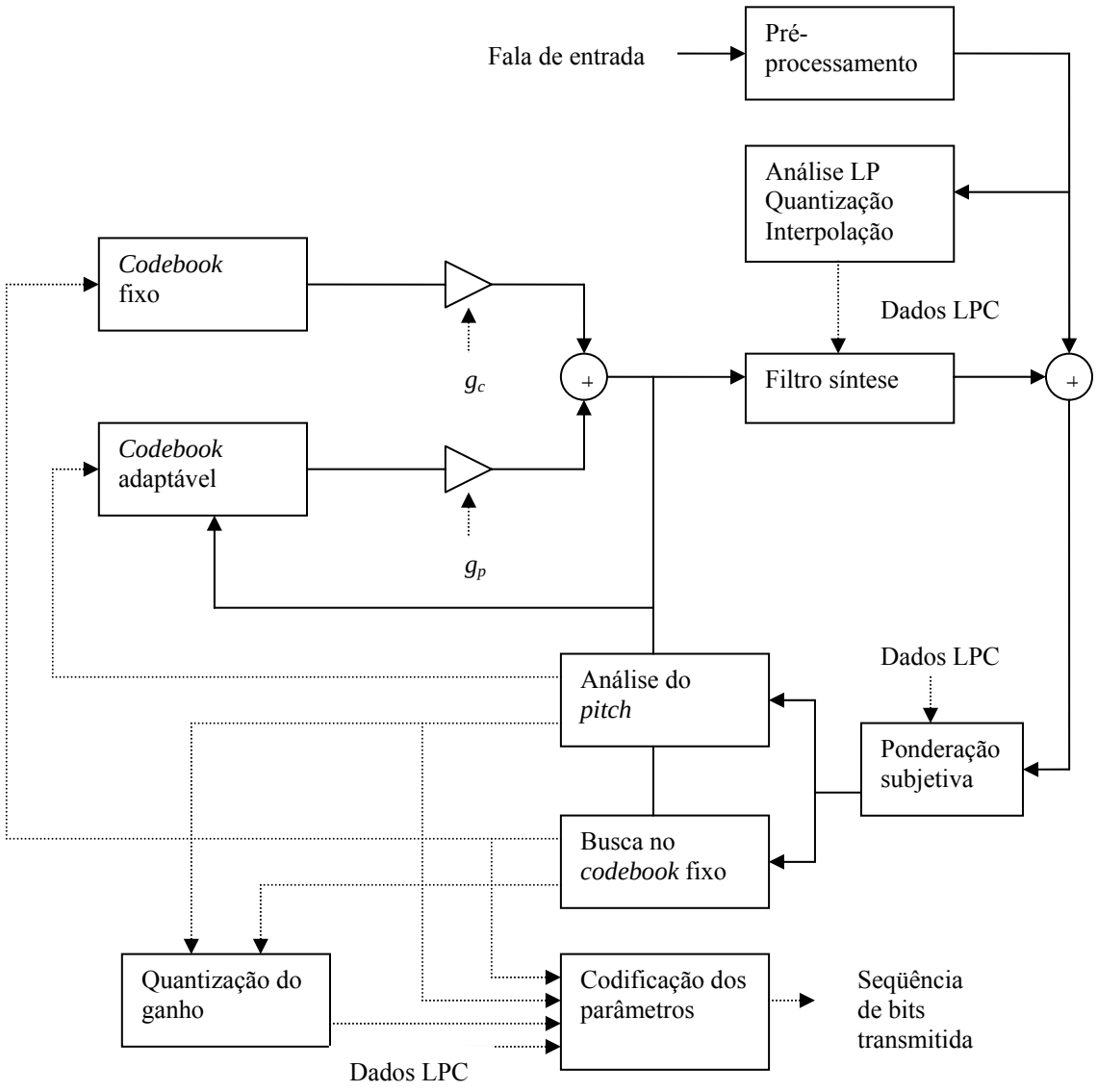


Fig. 3.6 Estrutura do codificador CS-ACELP (especificação G.729)

3.3.1.1 Pré-processamento

O pré-processamento é feito por um filtro passa-alta com freqüência de corte em 140 Hz e sua função é evitar a presença de componentes DC e de baixa freqüência no segmento de entrada, cujas amostras devem estar codificadas em 16 bits PCM linear. Além disso a entrada é dividida por 2 para reduzir a possibilidade de *overflow* nas operações em ponto fixo.

3.3.1.2 Análise LP, quantização e interpolação

A análise LP é realizada uma vez a cada segmento de 10 ms, usando o método da autocorrelação. Antes da análise, o sinal passa por uma janela assimétrica de 30 ms, composta de duas partes: metade de uma Hamming nos 25 ms iniciais e quarto de co-seno nos 5 ms finais.

A janela abrange 15 ms (120 amostras) da fala passada, 10 ms (80 amostras) da fala presente e 5 ms (40 amostras) da fala futura (5 ms de *look-ahead*, o que confere ao algoritmo um atraso intrínseco de 15 ms). A janela de 30 ms garante uma evolução suave do filtro LP, melhorando assim a qualidade da fala reconstruída. Depois do "janelamento" são calculados os coeficientes de autocorrelação, que serão usados no algoritmo de Levinson-Durbin para determinar os coeficientes LP. Para evitar o mau condicionamento do algoritmo de Levinson-Durbin os coeficientes de autocorrelação sofrem expansão de banda e o valor *r(0)* é multiplicado por um fator conhecido como "fator de correção tipo ruído branco".

Os coeficientes do filtro de síntese de ordem 10, obtidos da análise LP, são convertidos em coeficientes LSP usando polinômios Chebyshev e em seguida em coeficientes LSF (por ser mais conveniente à quantização dos coeficientes representados na freqüência normalizada em radianos [0, A]). A análise LP a cada 10 ms pode produzir tanto coeficientes com forte correlação entre segmentos (durante segmentos de fala vozeada) quanto coeficientes com correlação branda (durante segmentos não vozeados e de transição). Para acomodar os dois aspectos o codificador utiliza quantização preditiva, com seleção entre dois modos (índice L0). Basicamente, um preditor de quarta ordem baseado em filtro MA (*Moving-Average*) é usado para a predição dos coeficientes LSF do segmento corrente. A diferença entre o valor calculado e o predito é quantizada usando um

quantizador de dois estágios. O primeiro estágio é um quantizador vetorial de dimensão 10, usando um *codebook* (índice L1) de 128 entradas (7 bits). O segundo estágio é um quantizador implementado em *split* VQ, usando dois *codebooks* (índices L2 e L3) de 32 entradas (5 bits) cada. O melhor preditor é aquele que minimiza uma medida de distorção (erro médio quadrático ponderado) entre o coeficiente LSF calculado e o predito.

Os coeficientes, quantizados e não quantizados, obtidos da predição linear são usados diretamente apenas no segundo subquadro. No primeiro subquadro os coeficientes, quantizados e não quantizados, são obtidos através da interpolação linear dos respectivos parâmetros em segmentos de 10 ms adjacentes (interpolação entre o segmento de 10 ms atual e o anterior). A interpolação ocorre sobre os parâmetros LSP (e não LSF) apenas por motivos de implementação. Depois de quantizados e interpolados os coeficientes LSP são convertidos de volta em coeficientes LP.

3.3.1.3 Ponderação subjetiva

A ponderação do sinal de entrada é necessária para que se possa encontrar o atraso de *pitch* e acontece em cada subquadro, passando o segmento pelo filtro ponderador

$$W(z) = \frac{A(z/\gamma_1)}{A(z/\gamma_2)} = \frac{1+\sum_{i=1}^{10}\gamma_1^i a_i z^{-i}}{1+\sum_{i=1}^{10}\gamma_2^i a_i z^{-i}} \tag{3.33}$$

cujos parâmetros a_i vêm dos coeficientes não quantizados da análise LP (maior proximidade ao espectro original). Além disso outros dois parâmetros, η_1 e η_2, são usados para controlar a resposta em freqüência do filtro e por conseqüência a ponderação do ruído. Estes parâmetros de ajuste dependem da característica do sinal de entrada, devido à dificuldade de se obter valores fixos que produzam bons resultados. A característica do sinal de entrada, no caso a forma do espectro, é extraída por um filtro preditor linear de segunda ordem obtido como subproduto da recursão de Levinson-Durbin. Os coeficientes de reflexão convertidos para LAR (*Log Area Ratio*), correspondentes ao segmento de 10 ms corrente, são usados no segundo subquadro. Para o primeiro, os coeficientes LAR são obtidos da interpolação entre os coeficientes LAR do segmento corrente e anterior.

3.3.1.4 Busca no *codebook* adaptável

O *codebook* adaptável é responsável pela construção da componente periódica de longo atraso da excitação. A complexidade de busca no *codebook* adaptável em laço fechado é reduzida restringindo-a em torno do atraso obtido da análise de *pitch* em laço aberto. A estimativa em laço aberto é feita para cada segmento de entrada ponderado (a ponderação do segmento de entrada de 10 ms é feita em cada subquadro, conforme item 3.3.1.3) em duas etapas: primeiro obtendo o valor máximo da correlação do sinal de entrada ponderado

$$R(k) = \sum_{n=0}^{79} s_w(n) s_w(n-k) \qquad (3.34)$$

em cada uma das três faixas: k = {80...143, 40...79, 20...39} (quando $n - k < 0$ é usado o valor do segmento anterior). Depois o valor máximo de cada faixa é normalizado e ponderado, vencendo o maior dos três. A estratégia de divisão em três faixas e a ponderação, de forma a favorecer os valores das faixas menores, reduz a escolha de *pitch* múltiplos.

Antes de encontrar o atraso em laço fechado são calculados a resposta ao impulso do filtro de síntese ponderado e o sinal alvo. A resposta ao impulso $h_w(n)$ é calculada para cada subquadro, filtrando o sinal composto dos coeficientes do filtro $A(z/\gamma_1)$, estendido por zeros, através dos filtros $1/\hat{A}(z)$ e $1/A(z/\gamma_2)$. O sinal alvo $x(n)$ geralmente[4] é calculado subtraindo a resposta à entrada zero do filtro de síntese ponderado $W(z)/\hat{A}(z)$ do sinal de entrada ponderado $s_w(n)$.

Os parâmetros do *codebook* adaptável são o atraso e o ganho. O atraso em laço fechado é calculado para cada subquadro (5 ms) minimizando o erro médio quadrático ponderado entre os sinais da fala original e reconstruída, o que equivale a maximizar o termo:

$$R(k) = \frac{\sum_{n=0}^{39} x(n) y_k(n)}{\sqrt{\sum_{n=0}^{39} y_k(n) y_k(n)}} \qquad (3.35)$$

[4] A especificação CS-ACELP utiliza um procedimento equivalente, onde o sinal alvo é obtido filtrando o sinal residual $r(n)$ através da combinação do filtro síntese $1/\hat{A}(z)$ e filtro ponderador $A(z/\gamma_1)/A(z/\gamma_2)$. Depois de encontrada a excitação do subquadro, os estados iniciais destes filtros são atualizados filtrando a diferença entre o sinal residual e a excitação. O sinal residual é dado por: $r(n) = s(n) + \sum_{i=1}^{10} \hat{a}_i s(n-i) \quad n = 0, ..., 39$

em que *x(n)* é o sinal alvo, $y_k(n)$ é a excitação passada filtrada para o atraso *k* (excitação passada filtrada pela resposta ao impulso do filtro de síntese ponderado, $h_w(n)$). A busca no primeiro subquadro é limitada na vizinhança (3 amostras anteriores e 3 posteriores) do atraso obtido em laço aberto e, no segundo, é limitada na vizinhança do atraso obtido no primeiro subquadro. O atraso do primeiro subquadro é codificado com 8 bits (P1) e o do segundo, diferencialmente com 5 bits (P2). Depois de encontrado o atraso, o ganho é calculado como:

$$g_p = \frac{\sum_{n=0}^{39} x(n)y(n)}{\sum_{n=0}^{39} y(n)y(n)} \tag{3.36}$$

em que *x(n)* é o sinal alvo e *y(n)* é o vetor do *codebook* adaptável (gerado a partir do atraso calculado) convolvido com $h_w(n)$.

3.3.1.5 Busca no *codebook* fixo

O *codebook* fixo apresenta estrutura algébrica, ou seja, os vetores são obtidos a partir do índice transmitido usando álgebra simples ao invés de pesquisa em tabela de dados. Neste *codebook*, especificamente, cada vetor é composto por quatro pulsos não nulos distribuídos conforme mostra a Tab. 3.2. Essa estrutura tem algumas vantagens [90]: não requer armazenamento, uma vez que a excitação é gerada em tempo real; é robusta quanto a erro de canal, na medida em que um erro simples corrompe apenas a posição de um pulso no vetor de excitação; a complexidade de busca é reduzida substancialmente devido ao cálculo eficiente de $\widetilde{C}_k$ e ξ_k da equação (3.31).

Pulso	Amplitude	Posição	Bits
i_0	s_0: C1	m_0: 0, 5, 10, 15, 20, 25, 30, 35	1+3
i_1	s_1: C1	m_1: 1, 6, 11, 16, 21, 26, 31, 36	1+3
i_2	s_2: C1	m_2: 2, 7, 12, 17, 22, 27, 32, 37	1+3
i_3	s_3: C1	m_3: 3, 8, 13, 18, 23, 28, 33, 38 4, 9, 14, 19, 24, 29, 34, 39	1+4

Tab. 3.2 Estrutura do *codebook* fixo do algoritmo CS-ACELP

O vetor de excitação *c(n)* é gerado tomando-se um vetor zero de dimensão 40 e em seguida acrescentando os quatro pulsos não nulos nas posições calculadas, multiplicados pela amplitude correspondente,

$$c(n) = s_0 V(n-m_0) + s_1 V(n-m_1) + s_2 V(n-m_2) + s_3 V(n-m_3) \qquad n = 0, ..., 39 \qquad (3.37)$$

em que *V(n)* corresponde ao pulso unitário. O vetor ótimo $c_k(n)$ é encontrado minimizando o erro quadrático médio ponderado entre a fala original e a fala sintetizada, o que equivale a maximizar o termo

$$\tau_k = \frac{C_k^2}{E_k} = \frac{(d^t c_k)^2}{c_k^t \Phi c_k} \qquad (3.38)$$

em que $d = \mathbf{H}^t\mathbf{x}$ é o vetor de correlação entre o sinal alvo *x'(n)* e a resposta ao impulso $h_w(n)$ do filtro de síntese ponderado, c_k é o vetor de excitação procurado e $\Phi = \mathbf{H}^t\mathbf{H}$ é a matriz das correlações de $h_w(n)$ (**H** é uma matriz triangular inferior contendo a resposta ao impulso do filtro de síntese ponderado). O sinal alvo é atualizado subtraindo a contribuição devido ao *codebook* adaptável, ou seja,

$$x'(n) = x(n) - g_p\, y(n) \qquad n = 0, ..., 39 \qquad (3.39)$$

lembrando que *y(n)* é o vetor do *codebook* adaptável filtrado e g_p é o ganho dado pela equação (3.36). O vetor correlação *d* é obtido da expressão:

$$d(n) = \sum_{i=n}^{39} x'(i) h_w(i-n) \qquad n = 0, ..., 39 \qquad (3.40)$$

e a matriz simétrica contendo as correlações de $h_w(n)$ é dada por:

$$\phi(i,j) = \sum_{n=j}^{39} h_w(n-i) h_w(n-j) \qquad \text{i} = 0, ..., 39 \qquad \text{j} = \text{i}, ..., 39 \qquad (3.41)$$

O sinal *d(n)* e a matriz Φ são determinados antes de iniciar o processo de busca no *codebook* e apenas os elementos realmente necessários são utilizados no cálculo. Além

disso, os dados obtidos da determinação de Φ são armazenados de forma estruturada, para acelerar as operações subseqüentes. A estrutura algébrica do *codebook* possibilita a redução da complexidade de busca uma vez que o vetor $c_k(n)$ contém apenas quatro pulsos não nulos com amplitude C 1. Assim, a correlação no numerador da equação (3.38) para um dado vetor c_k fica:

$$C = \sum_{i=0}^{3} s_i d(m_i) \tag{3.42}$$

em que m_i corresponde à posição do i-ésimo pulso e s_i a sua amplitude. A energia no denominador da equação (3.38) torna-se:

$$E = \sum_{i=0}^{3} \phi(m_i, m_i) + 2\sum_{i=0}^{2} \sum_{j=i+1}^{3} s_i s_j \phi(m_i, m_j) \tag{3.43}$$

Para simplificar ainda mais o processo de busca, a amplitude do pulso em uma determinada posição é estimada usando o sinal *d(n)*, mais precisamente, fazendo a amplitude do pulso igual ao sinal de *d(n)* naquela posição. Essa escolha de sinal, para uma dada combinação de pulsos, maximiza a correlação *C* da equação (3.38). Assim, antes da busca no *codebook* o sinal *d(n)* é decomposto em duas partes: seu valor absoluto, |*d(n)*|, e sinal, *sign[d(n)]*. Também, a matriz Φ é modificada para incluir essa informação de sinal, ou seja,

$$\phi'(i,j) = sign[d(i)]\ sign[d(j)]\ \phi(i,j) \qquad i = 0, ..., 39 \qquad j = i+1, ..., 39$$

Além disso, a diagonal principal de Φ é ajustada para remover o fator 2 da equação (3.43),

$$\phi'(i,i) = 0.5\ \phi(i,i) \qquad i = 0, ..., 39$$

Agora a correlação dada por (3.42) fica,

$$C = |d(m_0)| + |d(m_1)| + |d(m_2)| + |d(m_3)| \tag{3.44}$$

e a energia dada por (3.43),

$$E/2 = \phi'(m_0, m_0) + \phi'(m_1, m_1) + \phi'(m_0, m_1) + \phi'(m_2, m_2) + \phi'(m_0, m_2) + \phi'(m_1, m_2) + \phi'(m_3, m_3) + \phi'(m_0, m_3) + \phi'(m_1, m_3) + \phi'(m_2, m_3) \quad (3.45)$$

O algoritmo faz a busca através de quatro laços aninhados, onde em cada laço é adicionada a contribuição de um pulso. O teste exaustivo de todas as combinações requereria a análise de $2^3 2^3 2^3 2^4 = 8192$ casos, porém, para reduzir a complexidade, a busca é conduzida apenas sobre as combinações com boas possibilidades de sucesso. A estratégia adotada para isso foi limitar o número de vezes em que o quarto laço é solicitado (na busca exaustiva seria $2^9 = 512$ vezes), usando uma técnica denominada "busca focada", na qual um valor de referência, baseado na correlação C, é pré-calculado e usado para determinar a entrada (se a referência for ultrapassada) ou não no quarto laço. O valor de referência depende da correlação máxima C_{3max} e média C_{3m} devido à contribuição dos 3 primeiros pulsos:

$$V_r = C_{3m} + K_3 (C_{3max} - C_{3m}) \quad (3.46)$$

Em que $0 \leq K_3 < 1$. O fator K_3 controla a porcentagem do *codebook* pesquisada e, utilizando $K_3 = 0.4$, descobriu-se produzir desempenho próximo do obtido pela exaustão. Com esse valor o número de vezes em que o algoritmo entra no quarto laço baixa de 512 para 60, em média, e em apenas 5% das vezes excede 90. Portanto, no pior caso o número de combinações testadas chega a 90 x 16 = 1440, ao invés dos 8192 casos da busca exaustiva.

As posições dos pulsos i_0, i_1 e i_2 são codificadas com 3 bits cada, enquanto a do pulso i_3 com 4 bits (palavras de código C1 e C2, para os subquadros 1 e 2, respectivamente). A amplitude de cada pulso é codificada com 1 bit (palavras de código S1 e S2, para os subquadros 1 e 2, respectivamente).

3.3.1.6 Quantização dos ganhos

O ganho dos *codebooks* adaptável e fixo são quantizados vetorialmente usando 7 bits. Essa quantização conjunta resulta na economia de 2 bits em relação à quantização escalar e não provoca qualquer degradação notável na qualidade da fala, se comparada à utilização de ganhos não quantizados. Os ganhos ótimos são calculados minimizando o erro médio quadrático ponderado entre a fala original e a reconstruída usando:

$$E_w = \sum_{n=0}^{39} (x(n) - g_p y(n) - g_c z(n))^2 \tag{3.47}$$

em que *x(n)* é o vetor alvo (seção 3.3.1.4), *y(n)* é o vetor do *codebook* adaptável filtrado e *z(n)* é o vetor do *codebook* fixo filtrado (os ganhos são encontrados fazendo $\partial E_w/\partial g_p = 0$ e $\partial E_w/\partial g_c = 0$).

Os ganhos do *codebook* fixo em segmentos adjacentes são correlacionados e por esta razão não são quantizados diretamente e sim através de um fator de correção γ entre o ganho ótimo, g_c, e o ganho predito, g'_c ($g_c = \gamma\, g'_c$). O preditor é baseado no logaritmo da energia do vetor do *codebook* fixo e além de ajudar a reduzir a faixa dinâmica do ganho ainda a torna menos dependente da variação do nível do sinal de entrada.

O ganho do *codebook* adaptável, g_p, e o fator γ do *codebook* fixo são quantizados em dois estágios usando um *codebook* de estrutura conjugada. O termo "conjugado" refere-se ao fato de que o vetor de entrada (g_p, γ) é quantizado como uma combinação linear de dois *codebooks*. O primeiro estágio consiste em um *codebook GA* de dimensão 2 e 3 bits, e o segundo um *codebook GB* de dimensão 2 e 4 bits. Dados os índices i_{GA} e i_{GB}, relativos a *GA* e *GB*, respectivamente, o ganho quantizado do *codebook* adaptável é dado por:

$$\hat{g}_p = GA_1(i_{GA}) + GB_1(i_{GB}) \tag{3.48}$$

e o ganho quantizado do *codebook* fixo por:

$$\hat{g}_c = g'_c\,(GA_2(i_{GA}) + GB_2(i_{GB})) \tag{3.49}$$

em que GA_1 e GA_2 são porções do mesmo *codebook GA* e serve para mostrar a existência de um processo de pré-seleção onde, durante a busca, apenas parte do *codebook GA* é

testada (o mesmo processo se aplica a *GB*). As palavras de código GA e GB (Tab. 3.1) são geradas a partir dos índices i_{GA} e i_{GB} que determinam a melhor escolha.

3.3.2 O descodificador

No descodificador, representado na Fig. 3.7, primeiramente os parâmetros são extraídos e decodificados a partir dos dados recebidos. Estes parâmetros representam um segmento de fala de 10 ms e compreendem: os coeficientes LSP, os dois atrasos de *pitch*, os dois vetores do *codebook* fixo e os ganhos dos dois *codebooks*. Os coeficientes LSP são interpolados e convertidos nos coeficientes do filtro de síntese de curto atraso para os dois subquadros. Então, para cada subquadro de 5 ms, a excitação é construída adicionando-se os vetores referentes à excitação fixa e adaptável, com seus respectivos ganhos, e filtrada através do filtro de síntese de curto atraso para reconstruir a fala. Finalmente, a fala reconstruída passa pelo processo de pós-filtragem, que inclui o pós-filtro, baseado nos filtros de síntese de longo e curto atrasos, um filtro passa-alta e uma operação para ajuste de escala.

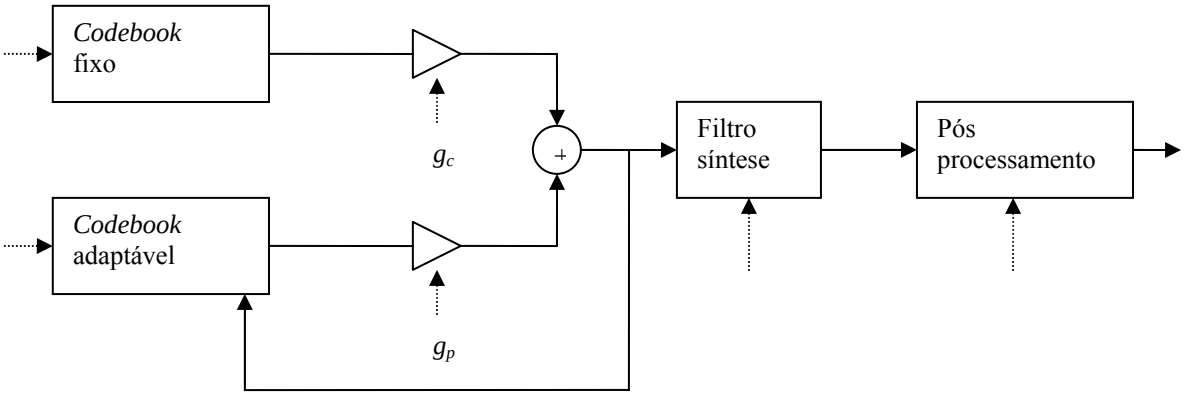


Fig. 3.7 Estrutura do descodificador CS-ACELP (especificação G.729)

3.3.2.1 Reconstrução da fala

O processo de reconstrução da fala começa com a decodificação dos parâmetros L0, L1, L2 e L3 para a geração dos coeficientes do filtro de síntese de cada subquadro. Em seguida os parâmetros P1 e P2 são decodificados para gerar os vetores do *codebook* adaptável, *v(n)*, de cada subquadro. Os vetores do *codebook* fixo, *c(n)*, de cada subquadro são gerados a partir dos índices C, que determinam a posição dos pulsos, e dos sinais S que

determinam as respectivas amplitudes. Finalmente os índices dos *codebooks* de ganho, GA e GB, são decodificados para gerar os ganhos dos *codebooks* adaptável e fixo de cada subquadro. A excitação *u(n)* dada por:

$$u(n) = \hat{g}_p\, v(n) + \hat{g}_c\, c(n) \qquad n = 0, ..., 39 \qquad (3.50)$$

é a entrada para o filtro de síntese, que reconstrói a fala em cada subquadro como:

$$\hat{s}(n) = u(n) - \sum_{i=1}^{10} \hat{a}_i\, \hat{s}(n-i) \qquad n = 0, ..., 39 \qquad (3.51)$$

em que $\hat{a}_i$ são os coeficientes do filtro de síntese de cada subquadro.

3.3.2.2 Pós-filtragem

Depois de reconstruída, a fala passa por um processo de pós-filtragem cujo efeito é enfatizar os picos espectrais, ao mesmo tempo reduzindo sua largura de banda e atenuando os vales entre estes picos. Essa conformação espectral inevitavelmente altera a forma de onda do sinal, mas é justificável frente ao ganho de qualidade percebido.

Este pós-processamento consiste em uma pós-filtragem adaptável e em uma filtragem passa-alta. O pós-filtro adaptável é uma cascata de três filtros: um filtro de longo atraso $H_p(z)$, um filtro de curto atraso $H_f(z)$ e um filtro compensador de inclinação $H_t(z)$ seguido de um ajuste de ganho (os coeficientes destes pós-filtro são atualizados a cada subquadro). Depois de passar pelo filtro passa-alta o sinal é multiplicado por 2 antes da saída do descodificador.

3.4 Conclusão

Este capítulo discorreu sobre os principais tópicos relacionados ao método de codificação de análise-pela-síntese e depois especificamente sobre o algoritmo de codificação CS-ACELP, sempre enfatizando os mecanismos envolvendo a excitação fixa, já que é nesse contexto que se desenvolve este trabalho.

O próximo capítulo trata particularmente da estratégia multi-taxa adotada para a geração da excitação fixa no algoritmo CS-ACELP, usando *codebook* estruturado em

árvore, e ainda do aspecto positivo relacionado à redução da complexidade de busca e compatibilidade com a especificação G.729. A perda de qualidade intrínseca relacionada à busca em árvore também é considerada.

C A P Í T U L O 4

EXCITAÇÃO MULTI-TAXA USANDO ESTRUTURA EM ÁRVORE

4.1 Introdução

Este capítulo apresenta a abordagem multi-taxa aplicada à codificação da excitação fixa do algoritmo CS-ACELP, através da busca da melhor seqüência em *codebook* estruturado em árvore. Apesar de a quantização vetorial estruturada em árvore, aplicada à codificação da fala, ser conhecida desde o início dos anos 80 [91-93], seu emprego em codificadores padrão parece não ter ocorrido de forma notória, provavelmente devido ao alto custo com armazenamento e acréscimo na distorção. Este capítulo mostra que, sob certos aspectos particulares de construção do *codebook*, é possível alcançar os benefícios da busca em árvore - a aproximação sucessiva (que resulta também na variabilidade da taxa) e a redução da complexidade computacional - com custo moderado de armazenamento e distorção relativamente branda. Além disso, pela proposta ainda é possível manter a compatibilidade com o algoritmo original, caso a codificação da excitação ocorra sempre na taxa máxima.

O capítulo primeiramente discorre sobre as justificativas para a abordagem TSVQ aplicada ao algoritmo CS-ACELP e em seguida recorda a técnica conhecida como *Pairwise Nearest Neighbor* (PNN), a partir da qual uma modificação permitiu que o *codebook* fosse estruturado. Na seqüência são conhecidos os detalhes da construção do *codebook* e da estratégia adotada para a redução do seu espaço de armazenamento. Por fim são expostos os aspectos relevantes sobre o procedimento de busca e sobre a transmissão multi-taxa, juntamente com as ações necessárias durante o processo de descodificação.

4.2 Justificativas para a abordagem TSVQ aplicada ao algoritmo CS-ACELP

A grande motivação para experimentar a técnica TSVQ na codificação da fala, e assim atender ao objetivo primeiro da proposição multi-taxa para aplicações VoIP, é que nela a variabilidade da taxa é uma conseqüência, restando por investigar o benefício da redução de complexidade e os danos relativos ao armazenamento e distorção. O fato de não existirem trabalhos empregando essa abordagem em codificadores padrão, de certa

forma desencorajava, mas por outro lado, a perspectiva de que era possível minimizar o problema do armazenamento do *codebook*, estimulava.

A redução da complexidade é uma vantagem importante da busca em árvore binária, pois o número de operações cresce apenas linearmente com o número de bits e não exponencialmente, como ocorre na busca completa. Por exemplo, para um codebook de β bits o número de seqüências a serem avaliadas se restringe a 2β e não 2^{β}. O problema é que o método requer o dobro de espaço de memória para armazenar o *codebook* e leva a certo aumento na distorção [93].

A escolha do algoritmo CS-ACELP como base para o desenvolvimento do trabalho tem explicação em sua ampla aceitação, comprovada pelos diversos padrões de codificação mencionados no Capítulo 2. Especificamente, a escolha do padrão G.729 decorreu de sua utilização nas comunicações VoIP e também devido ao fato de que o algoritmo CS-ACELP é o esquema de codificação que apresenta o melhor compromisso entre taxa de bit e "granularidade", referente ao atraso de empacotamento[5] [31].

Outro aspecto importante da busca em árvore é que não existe troca de contexto ao mudar a taxa, ou seja, o codificador não muda sua estratégia de codificação, diferente do que ocorre nos codificadores multi-modo onde por vezes, para que haja transições suaves, é preciso preservar o estado corrente ao iniciar um novo esquema de codificação [41].

A escolha do bloco de codificação dedicado à excitação fixa, como foco do trabalho, decorreu de alguns motivos: primeiro porque um codificador envolve muitos assuntos e não havia a pretensão de avançar em múltiplas direções. Segundo porque a excitação fixa é o bloco de codificação com maior peso na alocação de bit e onde freqüentemente as propostas multi-taxa se concentram, caso do *codec* AMR [60] e das extensões D e E do G.729 [69, 70], por exemplo. Terceiro porque o processo de busca da melhor excitação é a parte do *codec* de maior carga de processamento, portanto interessante para se experimentar uma pesquisa em árvore. E por último, é a etapa de codificação em que aparentemente se tem maior latitude, ou seja, onde existe certa tolerância relativa ao afastamento da excitação ótima.

[5] À medida que o atraso de empacotamento aumenta o mesmo acontece com a banda aparente (banda reservada na rede, para uma chamada telefônica, de modo que os requisitos de QoS permaneçam garantidos), devido à dificuldade de manter os compromissos de atraso durante o tratamento de fila nos roteadores. Por outro lado, a mesma ocorrência se verifica à medida que o atraso se torna muito pequeno, mas agora por conseqüência do peso relativo do preâmbulo.

4.3 Busca da melhor excitação em *codebook* fixo estruturado em árvore

O problema da complexidade de busca associada ao algoritmo CELP foi motivo de intenso trabalho após sua concepção, e basicamente gerou duas linhas de pesquisa [85]: a que propunha soluções derivadas de *codebooks* estocásticos e a que partia para abordagens algébricas, derivadas de códigos corretores de erro e estruturas *em treliça*. O êxito desta última (comprovado pelas especificações mais recentes: G.729, G.723.1, GSM Full-Rate, IS-136, AMR) está associado às propriedades geométricas destes *codebooks* estruturados, que proporcionam forte redução na carga computacional e espaço de armazenamento (estruturas algébricas foram originalmente propostas por Adoul [82]).

Em [83], por exemplo, Adoul e Lamblin mostram como os códigos corretores de erro podem ser usados para gerar *codebooks* que aceleram o processo de busca na estrutura CELP. Na proposição inicial, de Atal e Schroeder, a seqüência de excitação é composta de M amostras (M = 40) gaussianas *iid* (*independent, identically distributed*), todas normalizadas para que tenham energia unitária. Geometricamente, as seqüências podem ser vistas como vetores no espaço R^M distribuídos sobre a superfície de uma esfera M dimensional. Quantizar o resíduo da predição de curto e longo atraso (normalizado), cuja distribuição é uniforme sobre a esfera M dimensional, passa pela escolha de um *codebook* de tamanho N que cubra adequadamente e esteja ajustado a essa distribuição. Segundo a pesquisa, uma alternativa seria povoar o *codebook* com amostras gaussianas *iid*, mas os autores sugerem que outra estratégia seria usar vetores binários $(+1, -1)^M$, e que portanto os códigos binários, da teoria dos códigos corretores de erro, constituiriam um caminho sensato para se conseguir um conjunto de pontos uniformemente distribuídos sobre a hiperesfera.

Em [página 151, 36] Hanzo faz referência a uma interpretação esclarecedora sobre modelos de excitação CELP onde, a partir de uma representação geométrica para diversos *codebooks* defende que, independente de suas distribuições estatísticas, eles proporcionariam resultados subjetivos similares pois representariam um quantizador vetorial onde suas entradas estariam sobre a superfície de uma hiperesfera de raio unitário. Então, se esta superfície for suficientemente povoada os resultados serão de certa forma similares, mas possivelmente com diferenças enormes em termos de complexidade de codificação.

Na proposta CS-ACELP, o *codebook* algébrico é composto por vetores (Tab.3.2) distribuídos sobre a superfície de um hipercubo em $\boldsymbol{R}^k$ (k = 40). Como no processo de otimização (busca do melhor vetor de excitação) a medida de distorção empregada é o erro médio quadrático, então cada vetor do *codebook* está associado a uma partição do espaço $\boldsymbol{R}^k$ delimitada por segmentos de hiperplanos (em $\boldsymbol{R}^{k-1}$), formando uma região convexa *polytope*. Esse tipo de quantizador vetorial é um caso especial dos quantizadores Voronoi, ou *Nearest Neighbor Quantizers* (NN), onde as partições são *polytopes*, explicitamente determinadas a partir dos vetores do *codebook* [40].

Para manter a compatibilidade com o padrão G.729, a estrutura em árvore deve ser montada de forma que os nós terminais sejam os próprios vetores do *codebook* CS-ACELP, ou seja, a árvore deve ser gerada a partir dos centróides terminais e não de uma seqüência de treinamento, como acontece no caso geral. Uma saída para esse problema seria usar o método de busca BHT (*Binary Hyperplane Testing*) [94], técnica cuja proposta é encontrar uma estrutura em árvore eficiente a partir de qualquer *codebook* com qualquer dimensão de vetor. Neste método a árvore é construída a partir das partições Voronoi do *codebook* dado, encontrando o hiperplano associado a cada nó interno que melhor contribui para o resultado da busca. Durante a busca, o vetor a ser quantizado é testado em cada nó, para determinar em qual dos dois lados do hiperplano ele se encontra, reduzindo substancialmente o número de vetores candidatos na etapa subseqüente.

O problema dessa abordagem é que a árvore binária gerada não é balanceada e se aplica apenas aos casos em que $N >> 2^k$, em que N é o tamanho do *codebook* e k a dimensão de seus vetores (no CS-ACELP $N = 2^{17}$ e k = 40). Apesar de o método não ser aplicável à proposta em questão, dele pode-se valer de uma importante observação: a escolha do hiperplano associado a cada nó não terminal, de modo que sua contribuição para a busca resulte na menor complexidade média possível, é um problema de difícil solução, qualificado como *NP-Complete* [95], ou seja, existem boas evidências de que seja intratável.

A opção que se vislumbrava então era experimentar a geração da árvore binária, a partir do *codebook* CS-ACELP, usando o algoritmo PNN ou alguma estratégia semelhante.

O *codebook* estruturado em árvore geralmente é determinado empregando uma variação do método *splitting* proposto por Linde *et al*. [96] para a busca exaustiva. Neste trabalho, porém, a estratégia consiste em usar algo próximo ao algoritmo PNN para essa tarefa.

O algoritmo PNN [39] foi originalmente apresentado como uma alternativa ao algoritmo de Linde-Buzo-Gray (ou LBG) [96] para o projeto de *codebook*, entretanto geralmente é apontado [40, 35] como uma boa opção para a geração de *codebook* inicial. O algoritmo inicia assumindo que cada vetor de uma seqüência de treinamento de tamanho *L* é um *cluster* de um único vetor. O objetivo é ir agrupando os vetores da seqüência em *clusters*, cada qual representado por um único vetor centróide, até que o *codebook* alcance o tamanho desejado. Na etapa inicial, os dois vetores mais próximos são combinados em um único *cluster*, gerando um codebook com *L-1 clusters*, um deles com dois vetores e os demais com um vetor cada. Assim, em cada etapa dois *clusters* são fundidos, e seus respectivos centróides substituídos por um centróide único. A fusão deve ocorrer para a dupla de *clusters* que provoque o menor acréscimo na distorção média. Este acréscimo, ao fundir os *clusters* C_i e C_j, foi determinado por Equitz [39] como sendo:

$$\frac{n_i n_j}{n_i + n_j} \left\| Y_i - Y_j \right\|^2 \tag{4.1}$$

em que n_i é o número de vetores de treinamento no *cluster* C_i e Y_i o centróide correspondente.

4.3.1 Construção do *codebook* fixo estruturado em árvore

Um dos objetivos deste trabalho, com a proposição multi-taxa usando *codebook* fixo estruturado em árvore, é manter a compatibilidade com a especificação G.729, quando operando na taxa máxima. Para isso, o processo de quantização proposto deve levar a um vetor do *codebook* fixo da especificação G.729, ou seja, os nós terminais da árvore devem ser os próprios vetores do *codebook* CS-ACELP.

Portanto, gerar uma árvore binária de altura *d*, a partir dos vetores do *codebook* CS-ACELP, significaria empregar o algoritmo PNN *d* vezes, desde o nível dos nós terminais até o primeiro nível, no topo da árvore. Em cada rodada, que corresponde a um nível, o algoritmo deveria parar quando o número de *clusters* caísse pela metade, ou seja, quando o número de vetores centróide fosse igual ao número de vetores do nível imediatamente acima.

Infelizmente a aplicação direta do algoritmo PNN sobre os vetores do *codebook* CS-ACELP resulta em árvore desbalanceada. A solução encontrada foi dirigir a construção da árvore, admitindo distorção no processo de fusão dos *clusters*. O resultado é a árvore desejada: no último nível cada *cluster* tem apenas 1 vetor da seqüência de treinamento, representada pelos 2^d vetores do *codebook*. No nível *d*-1, cada *cluster* (dentre os possíveis 2^{d-1}), representado por um vetor centróide (nó intermediário), tem obrigatoriamente 2 vetores da seqüência inicial. No nível *d*-2, cada *cluster* tem 4 vetores, e assim sucessivamente até chegar no *cluster* raiz, contendo os 2^d vetores da seqüência inicial.

Outro ponto relevante, antes de detalhar a construção orientada, diz respeito ao *codebook*. O número de vetores do *codebook* CS-ACELP é $2^3 2^3 2^3 2^4 2^4 = 2^{17} = 131072$. Mas, conforme visto na seção 3.3.1.5, existe uma simplificação no procedimento de busca, fazendo a amplitude do pulso igual ao sinal de *d(n)* naquela posição. Ou seja, a busca na verdade ocorre em um subconjunto do *codebook* CS-ACELP, aqui denominado *P*, formado por vetores cujas componentes têm amplitude 0 ou 1. Essa simplificação reduz o universo dos vetores pesquisados, de 131072 para apenas 8192 (2^{13}), e também foi adotada nesta proposta. Assim, a árvore fica reduzida à altura $d = 13$, construída a partir deste subconjunto *P*.

A heurística adotada para a construção da árvore, admitindo distorção no agrupamento dos *clusters*, foi a seguinte:

- Os vetores do nível 13, que são os nós terminais da árvore e correspondem aos vetores do subconjunto *P*, são ordenados por valor de forma decrescente, conforme mostra a Tab. 4.1. Os vetores foram numerados desde $\boldsymbol{v}_{8192}$ a $\boldsymbol{v}_{16383}$, para coincidir com a numeração dos nós da árvore resultante;

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	
1	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8192}$
1	**1**	**1**	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8193}$
1	**1**	**1**	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8194}$
1	**1**	**1**	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8195}$
1	**1**	**1**	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8196}$
1	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8197}$
1	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8198}$
1	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8199}$
1	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8200}$
1	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8201}$
1	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8202}$
1	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8203}$
1	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8204}$
1	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8205}$
1	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	**1**	0	0	0	0	0	0	0	$\boldsymbol{v}_{8206}$
1	**1**	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	**1**	0	0	0	0	0	0	$\boldsymbol{v}_{8207}$
1	**1**	0	**1**	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8208}$
1	**1**	0	**1**	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8209}$
1	**1**	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8210}$
1	**1**	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8211}$
1	**1**	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8212}$
1	**1**	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8213}$
1	**1**	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8214}$
1	**1**	0	0	**1**	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8215}$
1	**1**	0	0	**1**	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\boldsymbol{v}_{8216}$
																			]																					]
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	**1**	0	0	**1**	**1**	0	0		$\boldsymbol{v}_{16364}$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	**1**	0	**1**	**1**	0	0		$\boldsymbol{v}_{16365}$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	**1**	**1**	**1**	0		$\boldsymbol{v}_{16366}$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	**1**	**1**	0	**1**		$\boldsymbol{v}_{16367}$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	**1**	0	**1**	0	0	0	0	0		$\boldsymbol{v}_{16368}$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	**1**	**1**	0	0	0	0	0		$\boldsymbol{v}_{16369}$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	**1**	0	0	**1**	0			$\boldsymbol{v}_{16370}$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	0	**1**	0	0	0	**1**			$\boldsymbol{v}_{16371}$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	**1**	0	**1**	0	**1**	0	0			$\boldsymbol{v}_{16372}$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	**1**	**1**	0	**1**	0	0			$\boldsymbol{v}_{16373}$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	**1**	0	**1**	**1**	0			$\boldsymbol{v}_{16374}$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0	**1**	0	**1**	0	**1**			$\boldsymbol{v}_{16375}$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	0	**1**	**1**	0	0	0				$\boldsymbol{v}_{16376}$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	**1**	**1**	**1**	0	0	0				$\boldsymbol{v}_{16377}$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	**1**	**1**	0	**1**	0				$\boldsymbol{v}_{16378}$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	**1**	**1**	0	0	**1**				$\boldsymbol{v}_{16379}$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0	**1**	**1**	**1**	0	0					$\boldsymbol{v}_{16380}$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	**1**	**1**	0	0						$\boldsymbol{v}_{16381}$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	**1**	**1**	0							$\boldsymbol{v}_{16382}$
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**	**1**	**1**	0	**1**							$\boldsymbol{v}_{16383}$

Tab. 4.1 Vetores do subconjunto *P* do *codebook* CS-ACELP, cujas componentes valem 0 ou 1, ordenados por valor decrescente. A numeração de 0 a 39 corresponde à posição das componentes.

- O nível 12 da árvore é gerado agrupando aos pares os *clusters* do nível 13, conforme mostra a Fig. 4.1. Especificamente, os 4096 (2^{12}) vetores centróide do nível 12 (nós intermediários) são determinados pela média aritmética dos vetores centróide filhos, segundo a equação (4.2), que determina todos os nós intermediários da árvore:

$$v_{FI-k} = cent(R_k) = \frac{1}{2} \sum_{i=2(FI-k)}^{2(FI-k)+1} v_i \qquad \text{para } k = 1 \ldots FI - 1 \qquad (4.2)$$

em que $FI = 8192$ é a folha início; $cent(R_k)$ corresponde ao centróide do novo *cluster* R_k e $\boldsymbol{v}_i$ são os vetores geradores do centróide. A equação (4.2) leva em conta que cada vetor do *codebook* reduzido tem probabilidade 1/8192, e que foi usado o erro médio quadrático;

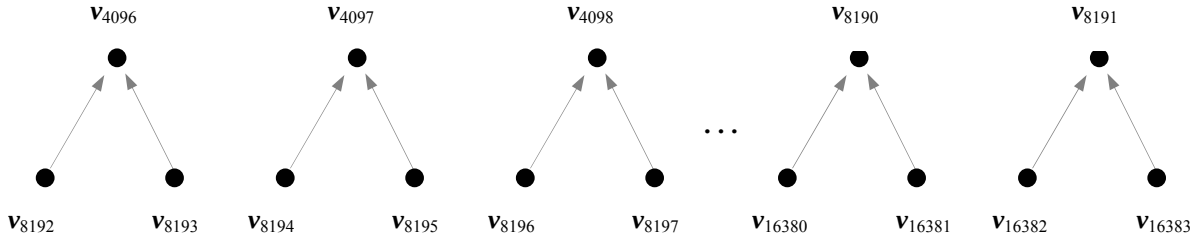


Fig. 4.1 Geração do nível 12 da árvore a partir dos vetores do nível 13.

- O procedimento para a geração dos demais níveis da árvore é análogo ao do nível 12. Por exemplo, o vetor centróide $\boldsymbol{v}_{2048}$ é a média aritmética dos vetores $\boldsymbol{v}_{4096}$ e $\boldsymbol{v}_{4097}$.

A distorção incidente ao usar essa estratégia fica evidente quando se observa o início da construção do nível 12. A geração dos vetores centróide $\boldsymbol{v}_{4096}$ ao $\boldsymbol{v}_{4106}$ está correta se considerada a aplicação direta do algoritmo PNN. Isto porque em todos os casos o acréscimo na distorção (equação (4.1)) é igual a 1, que é o menor acréscimo de distorção possível ao agrupar dois vetores do subconjunto *P* em questão. Porém, do ponto de vista do algoritmo PNN, esse procedimento não se manteria para o centróide $\boldsymbol{v}_{4107}$, ou seja, o *cluster* do centróide $\boldsymbol{v}_{8214}$ (de um vetor apenas) seria agrupado ao *cluster* do centróide $\boldsymbol{v}_{4106}$ (de 2

vetores), por exemplo, e não ao cluster do centróide $\boldsymbol{v}_{8215}$ (de um vetor apenas), pois neste caso o acréscimo na distorção é igual a 2, enquanto naquele é igual a 1.

Aceitar esse tipo de distorção na construção da árvore agrava o problema evidenciado na Fig. 2.2, mas por outro lado, essa estratégia além de levar a uma árvore balanceada ainda possibilita a criação de algumas regras que tornam desnecessário o armazenamento dos níveis 12 e 13 da árvore.

Uma das deficiências da abordagem em árvore está relacionada ao espaço dedicado ao armazenamento dos centróides. Neste aspecto, a estratégia adotada procura minimizar esse custo através de algumas ações, mas novamente incorrendo em algum acréscimo na distorção:

- Se cada centróide da árvore de 13 níveis fosse armazenado por completo, com cada componente ocupando 16 bits (1 word), então seria preciso um espaço igual a 16383x40x1 = 655320 words, ou aproximadamente 640 kwords. Para reduzir este espaço e acelerar o processo de busca, cada centróide retém apenas a posição (0 a 39) das 4 componentes de maior energia. Durante a busca, a amplitude destas componentes não nulas é considerada igual a +1. Além disso, caso haja mais de uma componente com o mesmo valor de energia, retem-se aquela de menor posição. Essa simplificação no centróide afeta o desempenho do codificador de algum modo, mas é o custo pago pela redução do espaço de memória. Fazendo um cálculo aproximado, se cada posição consumir 6 bits para seu armazenamento, então a simplificação reduz o espaço para 16383x4x6 = 393192 bits, ou aproximadamente 24 kwords;

- Conforme sugerido, a árvore proposta na verdade requer armazenamento até o nível 11. Mesmo assim, um esquema de diferenças poupa parte dos 24 bits que seriam necessários para guardar as 4 componentes de cada vetor centróide modificado. Dado um centróide genérico de 4 componentes $\{comp_0, comp_1, comp_2, comp_3\}$, com cada componente assumindo um valor de 0 a 39, então o conteúdo efetivamente armazenado, d_k, obedece a seguinte equação de diferenças:

$$d_k = comp_k - comp_{k-1} \qquad k = 0...3 \qquad \text{e} \qquad comp_{-1} = 0 \tag{4.3}$$

A partir dos vetores de diferenças $\{d_0, d_1, d_2, d_3\}$ pôde-se constatar que muitas seqüências $\{d_1, d_2, d_3\}$ se repetiam ao longo dos vetores. Esse fato possibilitou a

geração de um arquivo intermediário contendo o conjunto das seqüências $\{d_1, d_2, d_3\}$ suficientes para atender todas as ocorrências. Então, ao invés de guardar todos os vetores $\{d_0, d_1, d_2, d_3\}$, guarda-se apenas d_0 e um índice que dá acesso à seqüência $\{d_1, d_2, d_3\}$ desejada. Esse conjunto também seguiu um ordenamento no arquivo, necessário apenas para planejar a forma de armazenamento da componente d_0: primeiramente aparecem no arquivo intermediário as seqüências $\{d_1, d_2, d_3\}$ para as quais a componente d_0 exige 6 bits (valores de 32 a 39) para a codificação (3 seqüências). Depois as seqüências para as quais a componente d_0 exige 5 bits (valores de 16 a 31) para a codificação (197 seqüências). Depois as seqüências que exigem 4 bits (valores de 8 a 15) (259 seqüências) e por último aquelas que exigem 3 bits (valores de 0 a 7) para a codificação (558 seqüências para este caso e 1017 no total geral). A grande maioria das seqüências $\{d_1, d_2, d_3\}$ usou a distribuição de bits mostrada na Fig. 4.2, em que as componentes d_1, d_2 e d_3 usam 5 bits cada:

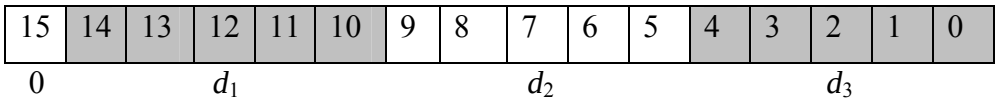


Fig. 4.2 Distribuição de bits (5,5,5) usual para as componentes d_1, d_2 e d_3.

Para os casos onde essa distribuição não comportava, então uma das possibilidades mostradas na Fig. 4.3 atendia:

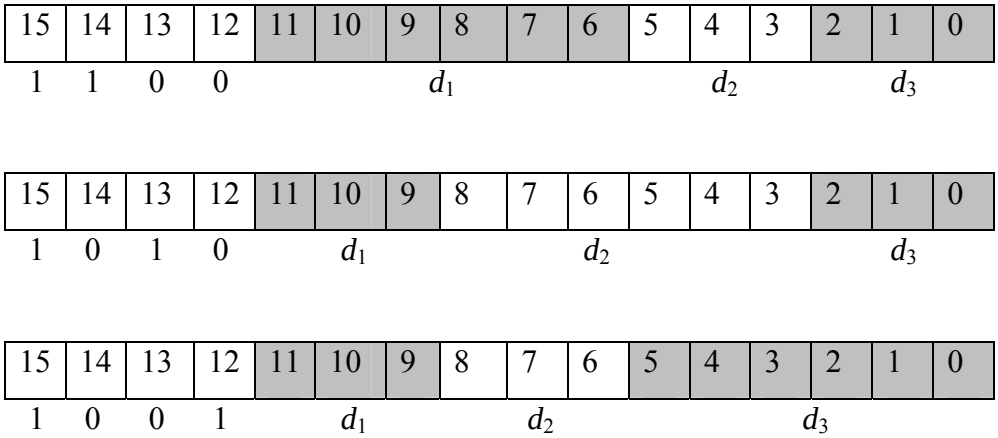


Fig. 4.3 Possíveis distribuições de bits para as componentes d_1, d_2 e d_3.

Finalmente, os 4095 centróides correspondentes aos 11 níveis da árvore são armazenados obedecendo a uma das distribuições de bits mostradas na Fig. 4.4:

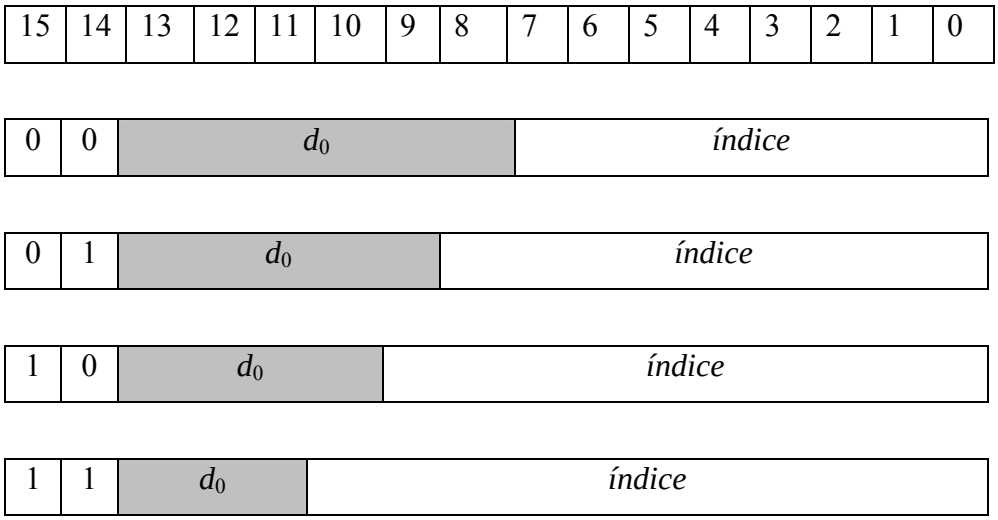


Fig. 4.4 Possíveis distribuições de bits para a componente d_0 e índice de acesso ao conjunto das seqüências $\{d_1, d_2, d_3\}$.

Em que *índice* corresponde à posição, no conjunto das seqüências $\{d_1, d_2, d_3\}$, onde se encontra a seqüência desejada. Portanto o espaço requerido para armazenar os 4095 centróides se resume a: 4095 words, que guardam a componente d_0 e o *índice*, mais 1017 words para o conjunto das seqüências $\{d_1, d_2, d_3\}$, ou seja, 5112 words (aproximadamente 5 kwords), 1 kwords a menos do que seria necessário caso se optasse por guardar as 4 posições usando 6 bits por posição;

- O nível 12 da árvore é parcialmente armazenado. Um conjunto de regras de formação, junto com alguns bits de controle para cada nó pai no nível 11, garante a geração dos nós filho no nível 12 (Anexo 1). Para chegar ao conjunto das regras foi preciso antes gerar o nível 12, e analisar o comportamento de seus vetores, usando uma estratégia ligeiramente diferente daquela adotada para os demais níveis. Assim, depois de determinados os centróides no nível 12, eles também foram modificados para reter apenas a posição das 4 componentes de maior energia, mas com uma ligeira diferença. O primeiro centróide, $\boldsymbol{v}_{4096}$, seguiu o procedimento já descrito. Para os demais, o procedimento também foi o mesmo, exceto para a última das 4 componentes de maior

energia. Para ela, caso seu valor surgisse também em outras posições, convencionou-se pela escolha daquela tal que o valor resultante do vetor binário fosse menor ou igual ao valor do vetor do nó anterior, de modo a manter o ordenamento decrescente do nível, tal como no ordenamento do nível 13. A partir daí foi possível chegar a algumas regras bastante específicas e dependentes de alguns bits de controle para cada um dos 2048 nós pai no nível 11. Mesmo assim ainda foram necessárias 1665 words para os casos particulares, onde as regras não funcionam (o que significa ter de armazenar os dois nós filhos). Cada nó pai no nível 11 precisa de 4 bits de controle, conforme mostra o exemplo da Fig. 4.5 para o vetor pai $\boldsymbol{v}_{2049}$:

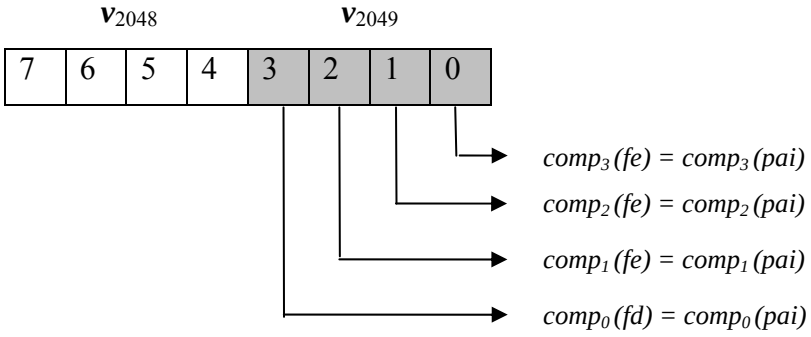


Fig. 4.5 Bits de controle para geração dos nós filho no nível 12 a partir do nó pai $\boldsymbol{v}_{2049}$.

Em que *fe* significa *filho da esquerda* e *fd*, *filho da direita*. As regras dependem de quais bits de controle estão marcados;

- O nível 13 não exigiu armazenamento. Para esse nível, basta o conhecimento do nó pai e algumas regras de formação (Anexo 2);

- Os 11 primeiros níveis da árvore não puderam seguir estratégia de geração semelhante devido ao excesso de casos particulares.

O espaço total de memória exigido para a estratégia proposta é de aproximadamente 7 kwords, conforme mostra o resumo na Tab. 4.2. É um espaço bastante razoável e suportável para os atuais processadores digitais de sinais.

Parte da árvore	words
Primeiros 11 níveis (controle, d_0 e índice)	4095
Codebook das seqüências $\{d_1, d_2, d_3\}$	1017
Bits de controle para a geração do nível 12	512
Casos particulares para a geração do nível 12	1665
Total	**7289**

Tab. 4.2 Espaço total de memória exigido para o *codebook* fixo estruturado em árvore.

4.4 Procedimento de busca no *codebook* fixo

O vetor ótimo $c_k(n)$ é encontrado minimizando o erro médio quadrático ponderado entre a fala original e a reconstruída, o que equivale a maximizar o termo τ_k dado pela equação (3.38), aqui reescrita para facilitar a explanação:

$$\tau_k = \frac{C_k^2}{E_k} = \frac{(d^t c_k)^2}{c_k^t \Phi c_k}$$

Da mesma forma que no algoritmo CS-ACELP, antes de iniciar a busca do vetor ótimo $c_k(n)$ é preciso encontrar o vetor correlação $d(n)$, dado pela equação (3.40), e a matriz das correlações de $h_w(n)$, Φ, determinada pela equação (3.41). O procedimento para encontrar $d(n)$ manteve-se inalterado em relação à especificação G.729. Já no caso da matriz Φ, diferente da especificação, todos os elementos são necessários (a matriz Φ_{40x40} é simétrica, exigindo o cálculo de 820 correlações).

Uma vez determinadas as entradas $d(n)$ e Φ, a busca inicia. Ao invés de efetuá-la usando posicionamento de pulsos através de laços aninhados, a estratégia proposta encontra o melhor vetor usando pesquisa em árvore binária. A quantidade de nós percorridos a partir do nó raiz depende da taxa desejada, que por sua vez está vinculada a mecanismos de controle baseados nas condições da rede (o controle da taxa não foi objeto de estudo deste trabalho). Definida a taxa de operação, o codificador poderia transmití-la ao descodificador como informação de controle. A Tab. 4.3 mostra a distribuição de bit e a taxa final do codificador, para cada nível da árvore. É possível ainda manter a

compatibilidade com a especificação G.729. Nesse caso, percorrem-se sempre os 13 níveis da árvore, chegando em um vetor do *codebook* CS-ACELP; a estrutura dos bits enviada ao descodificador segue a especificação G.729 e não existe possibilidade de variação da taxa.

Nível na árvore	**Bits do codebook fixo, subquadro 1**	**Bits do codebook fixo, subquadro 2**	**Bits para os demais parâmetros do codec**	**Total de bits (em 10 ms)**	**Taxa (kbit/s)**
1	4+1	4+1	46	56	5,6
2	4+2	4+2	46	58	5,8
3	4+3	4+3	46	60	6,0
4	4+4	4+4	46	62	6,2
5	4+5	4+5	46	64	6,4
6	4+6	4+6	46	66	6,6
7	4+7	4+7	46	68	6,8
8	4+8	4+8	46	70	7,0
9	4+9	4+9	46	72	7,2
10	4+10	4+10	46	74	7,4
11	4+11	4+11	46	76	7,6
12	4+12	4+12	46	78	7,8
13	4+13	4+13	46	80	8,0

Tab. 4.3 Distribuição de bit no algoritmo CS-ACELP usando *codebook* fixo estruturado em árvore. Na coluna do *codebook* fixo 4+*k* significa: 4 bits para codificar o sinal e *k* bits para descrever o caminho percorrido na árvore.

A pesquisa na árvore é bastante simples, sendo desempenhada por uma função C ANSI (como todo o *codec*) cujos parâmetros de entrada são: o vetor *d(n)*; os elementos da matriz Φ; a resposta ao impulso do filtro de síntese ponderado $h_w(n)$ e um indicador de subquadro. Os parâmetros de saída são: o vetor excitação escolhido; o vetor de excitação filtrado por $h_w(n)$ e o caminho percorrido na árvore. A função retorna o sinal dos 4 pulsos não nulos.

A simplificação nos centróides, mantendo apenas a posição das 4 componentes não nulas, possibilita a redução da complexidade de busca e também permite calcular o termo I_k usando as equações (3.44) e (3.45). A diferença é que aqui as posições $m_0...m_3$ podem ser quaisquer, dentro do intervalo [0...39]. O diagrama da Fig. 4.6 mostra o procedimento desempenhado pela função C para a busca do melhor vetor de excitação usando a árvore binária.

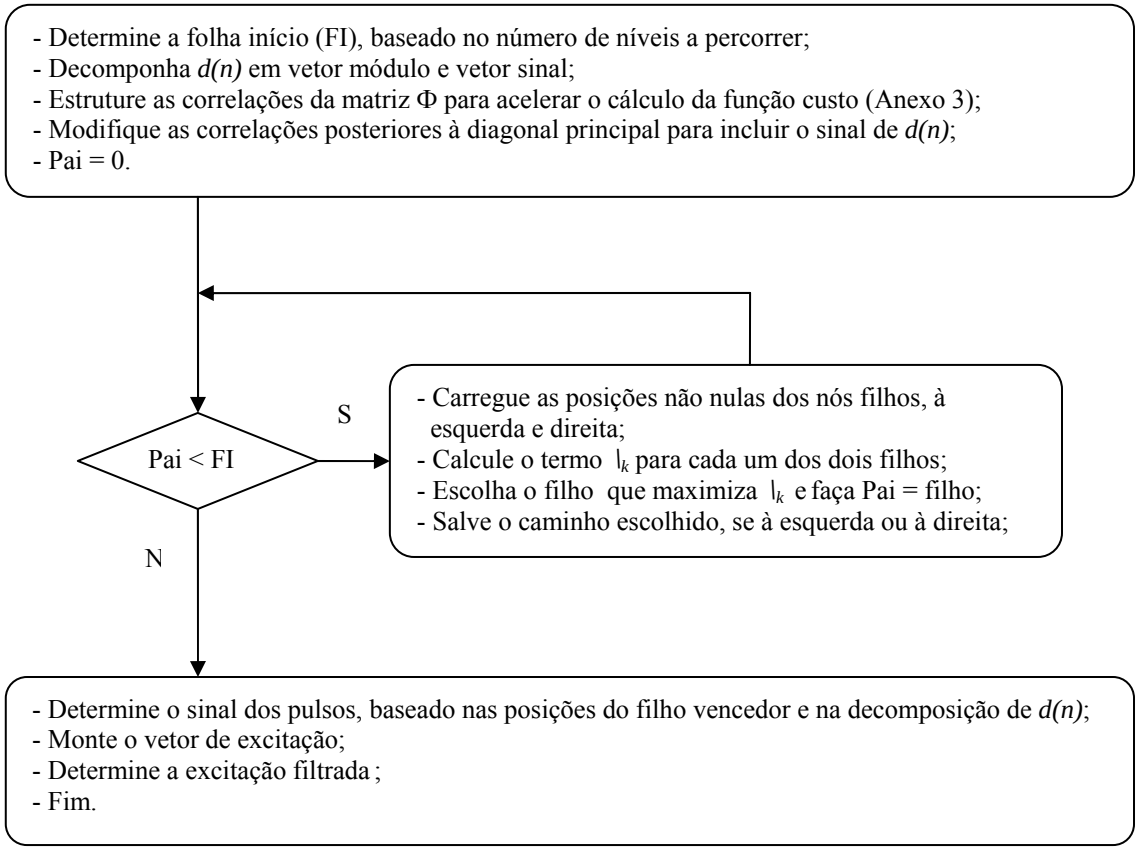


Fig. 4.6 Procedimento de busca do melhor vetor de excitação usando árvore binária.

Na especificação G.729 os parâmetros determinados na codificação, Tab. 3.1, são convertidos em uma estrutura de bits serial, conforme mostra a Tab. 4.4, que em seguida é armazenada em um arquivo de saída (entrada para o descodificador). Neste esquema de conversão cada bit do parâmetro é convertido em uma palavra de 16 bits. Assim, se o bit do parâmetro vale 0 ele é convertido para 0x007F, e se vale 1, ele é convertido em 0x0081. Além disso, duas palavras de controle iniciam a estrutura: SYNC_WORD = 0x6B21 e SIZE_WORD = 80. Esta estrutura é armazenada em arquivo de saída a cada 10 ms.

	word
1	SYNC_WORD
2	SIZE_WORD
3	L0
4	L1
]	]
10	L1
11	L2
]	]
15	L2
16	L3
]	]
20	L3
21	P1
]	]
28	P1
29	P0
30	C1
]	]
42	C1
43	S1
]	]
46	S1
47	GA1, GB1
]	]
53	GA1, GB1
54	P2
]	]
58	P2
59	C2
]	]
71	C2
72	S2
]	]
75	S2
76	GA2, GB2
]	]
82	GA2, GB2

Tab. 4.4 Estrutura dos parâmetros convertidos no algoritmo CS-ACELP (10 ms)

Nessa proposta, os parâmetros determinados na codificação, Tab. 4.5, permanecem praticamente os mesmos, apenas substituindo o índice do *codebook* pelo caminho percorrido na árvore.

Parâmetro	Palavra de código	Subquadro 1	Subquadro 2	Total
LSP (*Line Spectrum Pairs*)	L0, L1, L2, L3			18
Atraso do *codebook* adaptável	P1, P2	8	5	13
Paridade do atraso de *pitch*	P0	1		1
Caminho percorrido na árvore	Cm1, Cm2	1 a 13	1 a 13	2 a 26
Sinal do *codebook* fixo	S1, S2	4	4	8
Ganhos dos *codebook* (estágio 1)	GA1, GA2	3	3	6
Ganhos dos *codebook* (estágio 2)	GB1, GB2	4	4	8
Total				56 a 80

Tab. 4.5 Distribuição de bit no algoritmo CS-ACELP usando busca em árvore (10 ms)

Nesta proposta a estrutura serial é modificada, conforme mostra a Tab. 4.6, para a inclusão do caminho percorrido (Cm) e número de níveis (NN) da árvore. A modificação é a seguinte: no lugar do índice do *codebook* vão o sinal e o número de níveis, e no lugar do sinal vai o caminho percorrido na árvore. Aqui SYNC_WORD continua igual, mas SIZE_WORD = 94. O parâmetro NN é incluído em cada estrutura apenas por comodidade, mas poderia ser transmitido fora da estrutura, como mensagem de controle. O parâmetro Cm ocupa sempre 16 words, embora dependa do número de níveis, que varia de 1 a 13.

	word
1	SYNC_WORD
2	SIZE_WORD
3	L0
4	L1
]	]
10	L1
11	L2
]	]
15	L2
16	L3
]	]
20	L3
21	P1
]	]
28	P1
29	P0
30	S1, NN
]	]
37	S1, NN
38	Cm1
]	]
53	Cm1
54	GA1, GB1
]	]
60	GA1, GB1
61	P2
]	]
65	P2
66	S2, NN
]	]
73	S2, NN
74	Cm2
]	]
89	Cm2
90	GA2, GB2
]	]
96	GA2, GB2

Tab. 4.6 Estrutura dos parâmetros convertidos na presente proposta (10 ms)

4.5 Procedimento no descodificador

No descodificador, o procedimento permanece o mesmo do padrão G.729, exceto pela função que faz a descodificação do *codebook* algébrico. Os parâmetros de entrada dessa função são: o caminho percorrido na árvore (que usa a mesma variável de entrada, no algoritmo original, destinada ao sinal); o sinal dos pulsos não nulos (que usa a variável de entrada, no algoritmo original, destinada à posição dos pulsos). O parâmetro de saída é o vetor de excitação. O diagrama da Fig. 4.7 mostra o procedimento desempenhado pela função C que determina o vetor de excitação. Vale enfatizar ainda que o descodificador também precisa armazenar os dados da árvore.

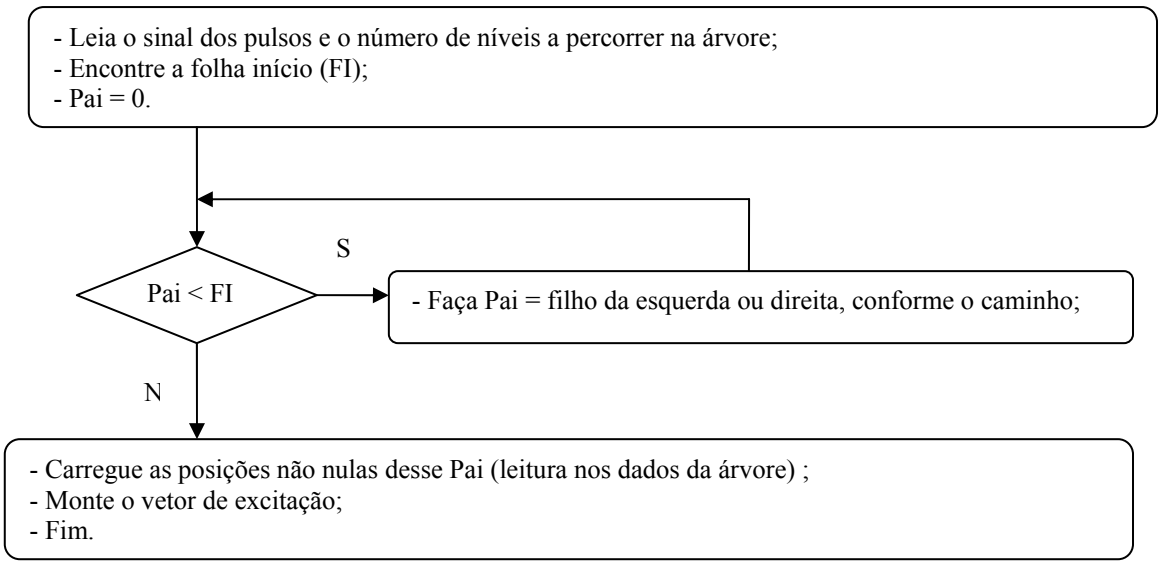


Fig. 4.7 Procedimento no descodificador para gerar o vetor de excitação usando árvore binária.

4.6 Conclusão

Este capítulo apresentou efetivamente a proposta de excitação multi-taxa usando estrutura em árvore. Primeiramente foram expostas as justificativas para a abordagem em árvore para o *codebook* fixo, seguidas por uma breve discussão sobre o algoritmo PNN. Por último foram apresentadas as estratégias usadas na construção do *codebook*, o procedimento de busca no codificador e a geração da excitação no descodificador.

No próximo capítulo são apresentados os resultados da proposta, o que inclui as medidas de distorção, complexidade e tempo de busca. Além disso os resultados mostram uma comparação com as medidas obtidas a partir da especificação G.729.

C A P Í T U L O 5

RESULTADOS

5.1 Introdução

Este capítulo apresenta os resultados práticos obtidos a partir da abordagem multi-taxa usando *codebook* fixo estruturado em árvore. Antes disso, é detalhado o ambiente sobre o qual ocorreram os testes e como foi configurado para chegar nos resultados. Em seguida, são analisados os resultados das medidas objetivas, em termos da distorção da fala codificada. Depois são avaliados os dados referentes à complexidade computacional e ao tempo de codificação. Finalmente são discutidos alguns aspectos relativos à variabilidade da taxa.

5.2 Ambiente de teste

Todos os testes foram realizados em um computador AMD Atlhon 1.60 GHz, com 512 MB de RAM e sistema operacional Microsoft Windows XP 2002. Os arquivos fontes, C ANSI, para simulação do algoritmo são parte integrante da especificação G.729 e foram obtidos diretamente do *site* do ITU (*www.itu.int*). O pacote de software (distribuição versão 3.3 usando aritmética de 16 bits em ponto fixo) inclui o arquivo READ.ME, que traz orientações acerca da compilação, utilização e verificação da correta execução do algoritmo.

Os arquivos fontes do projeto (*CELP_G729.dev*) foram alterados usando a ferramenta IDE (*Integrated Development Environment*) Dev-C++ versão 4.9.8.0. Para a compilação e ligação (*link*) dos arquivos do projeto foi usado o compilador GCC, presente no ambiente de desenvolvimento não proprietário DJGPP. O pacote de software do ITU inclui um *makefile* para o codificador (*CODER.MAK*) e outro para o descodificador (*DECODER.MAK*). No Anexo 4 estão os procedimentos para a configuração do ambiente de teste e execução do algoritmo.

O codificador está preparado para operar com um sinal digital que tenha sido amostrado em 8 kHz e cujas amostras estejam no formato PCM 16 bits linear. Ou seja, o

arquivo de entrada para o codificador deve ter essa formatação e o arquivo de saída é gerado pelo descodificador conforme essa especificação.

As medidas de distorção e tempo de processamento foram efetuadas usando o arquivo de teste *speech.in* (obtido do ITU juntamente com o pacote de software) de aproximadamente 37s de locução, cujas amostras são do tipo PCM 16 bits linear com taxa de amostragem de 8 kHz.

5.3 Medidas de distorção

A análise objetiva foi realizada em termos da relação sinal-ruído de quantização (*SNR*) e relação sinal-ruído de quantização segmentada (*SEGSNR – Segmental Signal-to-Noise Ratio*). A Fig. 5.1 mostra a evolução da relação SNR, para o codificador G.729 usando árvore, em função da taxa, utilizados durante a busca no *codebook* fixo. A figura mostra uma evolução da relação sinal-ruído bastante suave, desde a taxa mínima até a máxima, de aproximadamente 1 dB. Esse resultado se ajusta à análise subjetiva (não criteriosa e com poucos indivíduos), bastante satisfatória já na taxa mínima (5,6 kbit/s). Na codificação com 13 níveis, portanto na mesma taxa do G.729 (8 kbit/s), a relação sinal-ruído ficou aproximadamente 0,6 dB abaixo da obtida com o *codec* G.729. É difícil perceber a diferença entre segmentos de fala onde a distorção relativa seja da ordem de 0,5 dB [91] e foi exatamente o que se observou no teste subjetivo.

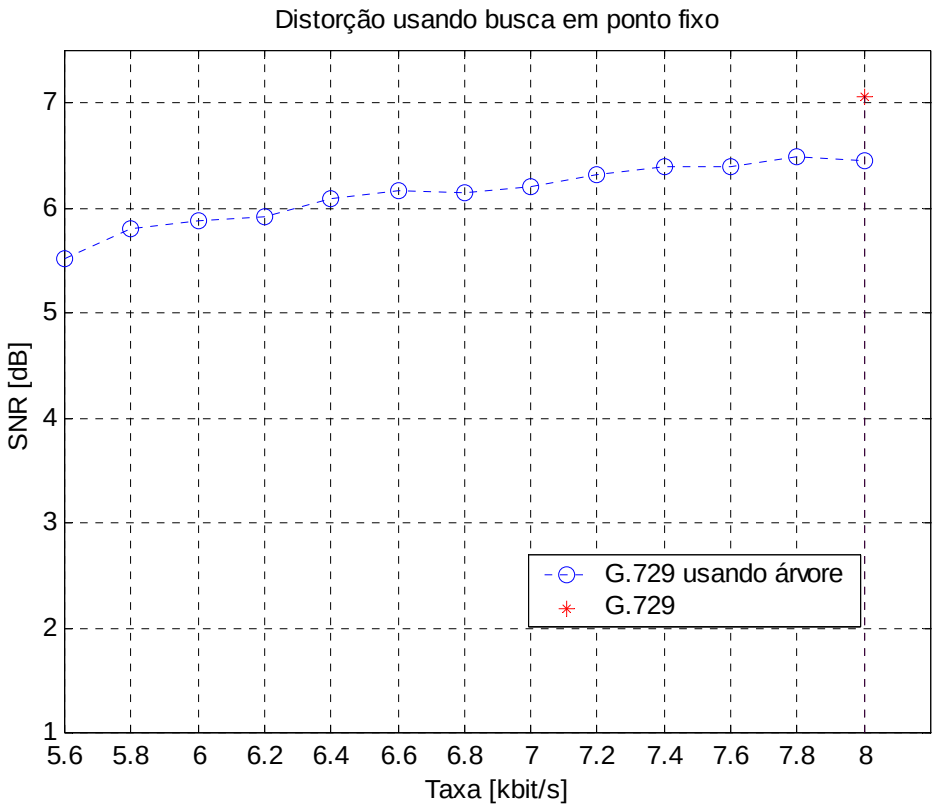


Fig. 5.1 Relação sinal-ruído para o codificador G.729 e G.729 usando *codebook* estruturado em árvore.

A Fig. 5.2 mostra resultados semelhantes, mas agora usando a relação sinal-ruído segmentada. Nela também foi incluída a medida para o G.729 ótimo, que é o G.729 modificado para que a busca no *codebook* fixo analise todos os vetores ACELP em cada subquadro. Aqui fica evidente a eficiência da busca focada, usada na especificação ACELP.

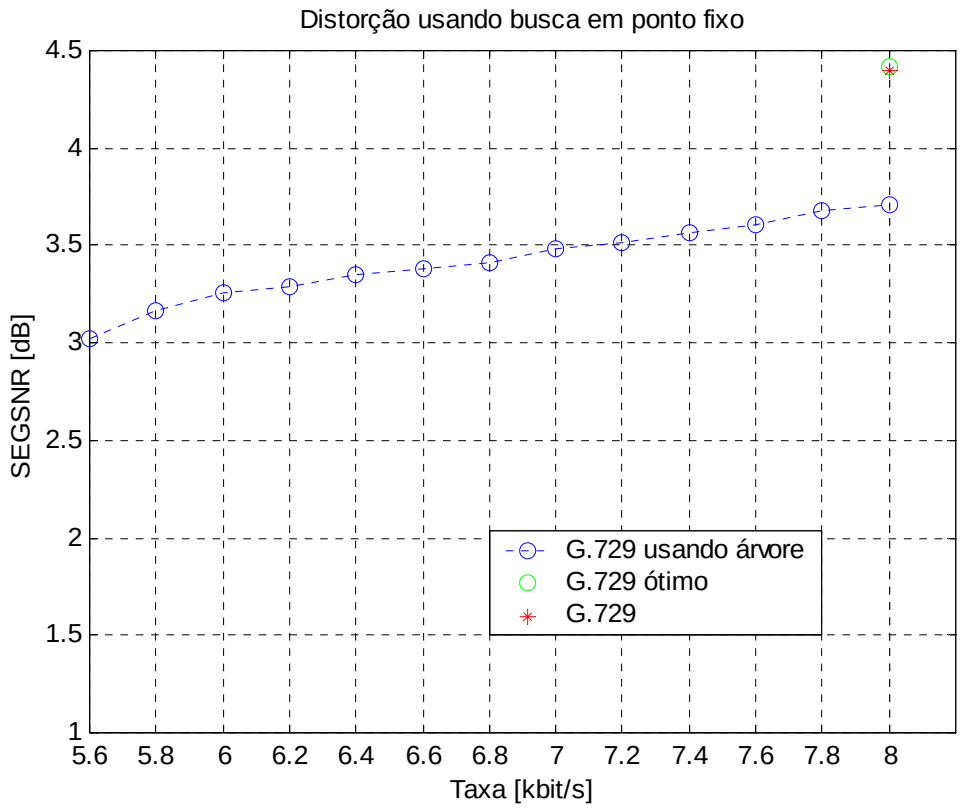


Fig. 5.2 Relação sinal-ruído segmentada para o codificador G.729, G.729 ótimo e G.729 usando *codebook* estruturado em árvore.

Apesar de o procedimento de construção da árvore incorrer em distorção intrínseca, e apesar de essa distorção aumentar ao se truncar os centróides, os resultados mostram que os danos causados por estas iniciativas não destruíram a característica da aproximação sucessiva quando se caminha na árvore em direção às folhas. Para reforçar essa afirmação, a Fig. 5.3 mostra o mesmo resultado da Fig. 5.2, mas acrescenta a distorção ao se caminhar na árvore escolhendo sempre o centróide oposto àquele que maximiza l_k . Ou seja, a figura mostra também a curva do filho errado (aquele que não maximiza l_k, da equação (3.38)) e por ela fica claro que, mesmo com todas as simplificações, a estrutura do *codebook* proposta ainda carrega as características de uma estrutura em árvore construída pelos métodos clássicos.

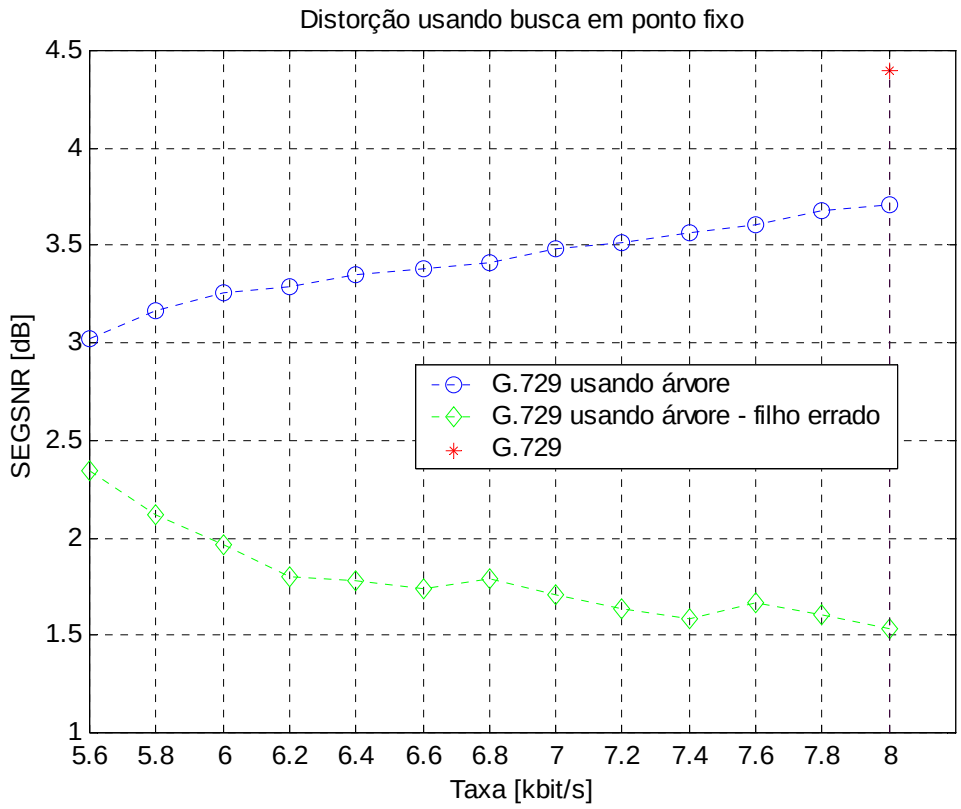


Fig. 5.3 Relação sinal-ruído segmentada para o codificador G.729, G.729 usando *codebook* estruturado em árvore e G.729 usando árvore, porém escolhendo sempre o filho errado.

A Fig. 5.4 inclui a distorção ao se caminhar na árvore por meio de busca em ponto flutuante com centróide completo, ou seja, sem a simplificação na qual se mantém apenas as quatro componentes de maior energia. A intenção foi medir de que modo o procedimento de truncar os centróides alterava a distorção. A questão é que, depois do cálculo do vetor algébrico escolhido filtrado (em ponto flutuante), algumas componentes ultrapassavam os valores limite (+0,999755859375 ou -1,0), provocando distorção acima do esperado (lembrando que neste caso apenas a busca foi calculada em ponto flutuante, ou seja, era preciso converter novamente o vetor filtrado para ponto fixo). O problema foi amenizado saturando as componentes fora da faixa, mas ainda assim houve uma degradação crescente com o número de níveis, pois o número médio de componentes fora da faixa também cresceu com essa variável, conforme mostra a Fig. 5.5 (a opção de normalizar as componentes também foi testada, mas os resultados não foram tão bons quanto usando a saturação). Subjetivamente é difícil perceber alguma diferença em relação ao *codec* G.729 original.

O resultado mostrado na Fig. 5.4 poderia, em um primeiro momento, nos estimular a efetuar a busca em ponto flutuante com centróide não truncado. Assim, poderiam ser armazenados apenas os dois primeiros centróides, exigindo somente 1 bit para descrever o caminho. O problema é que usar o centróide sem truncar implica em usar, no processo de busca, o sinal de *d(n)* das 40 componentes, penalizando assim a vantagem vislumbrada.

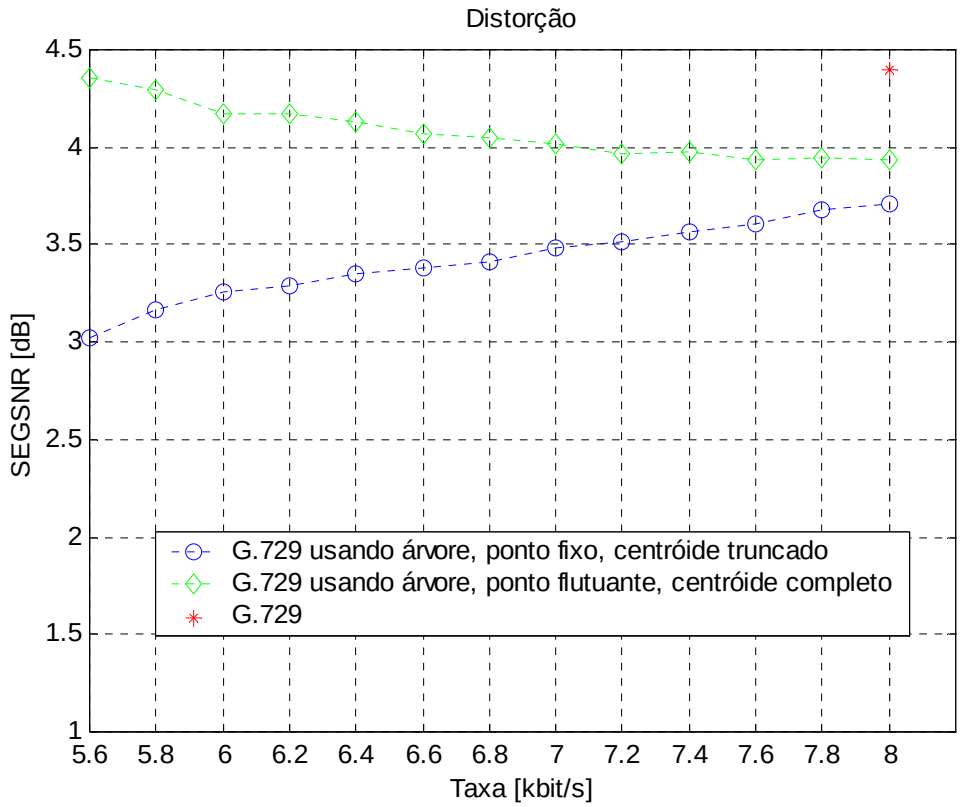


Fig. 5.4 Relação sinal-ruído segmentada para o codificador G.729, G.729 usando árvore com busca em ponto fixo e centróide truncado, e G.729 usando árvore com busca em ponto flutuante e centróide completo.

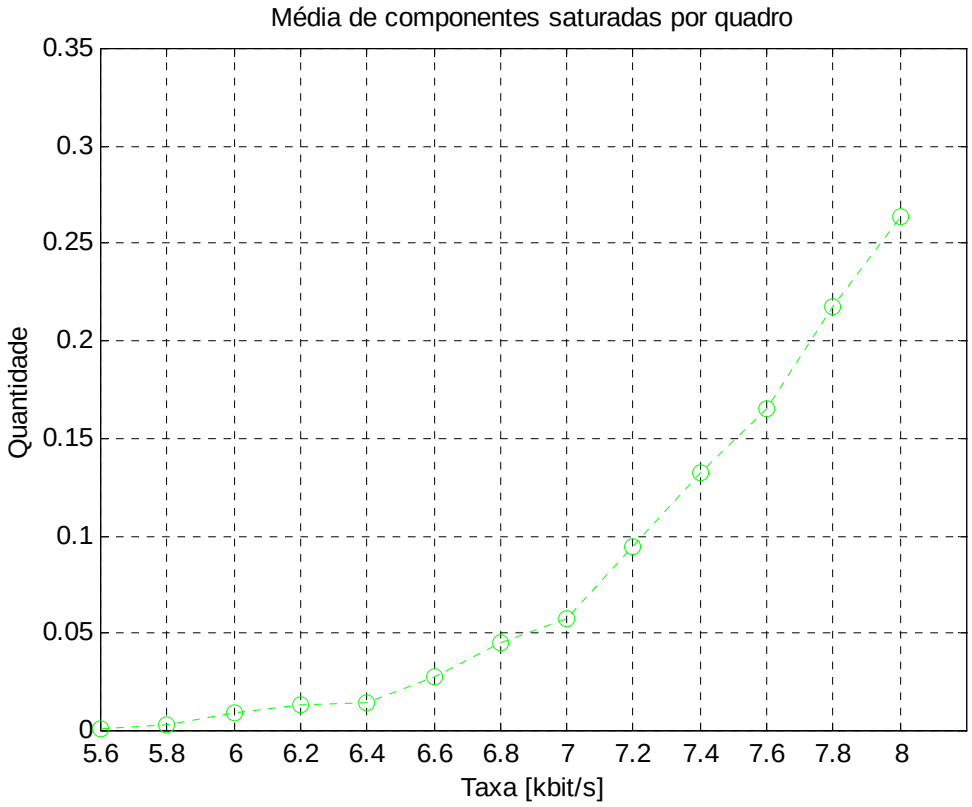


Fig. 5.5 Número médio de componentes saturadas por quadro para a codificação G.729 usando árvore com busca em ponto flutuante e centróide completo.

5.4 Complexidade computacional

A complexidade foi avaliada em termos do número de operações (e tipos) para cada subquadro durante o processo de busca da melhor excitação fixa. No caso do *codec* G.729 padrão, os valores resultam da média calculada sobre 906 subquadros, do arquivo de locução *speech.in*. Para o G.729 com busca em árvore, o número de operações é fixo para os 11 níveis iniciais, mas sofre variação nos dois últimos em função do procedimento para a geração dos mesmos. A opção adotada neste caso foi mostrar o número de operações no pior caso. A Fig. 5.6 mostra as operações empregadas na busca da excitação fixa e as respectivas quantidades, para o G.729 padrão e G.729 usando árvore. Aqui fica evidente o aspecto favorável da redução de complexidade para abordagens usando árvore.

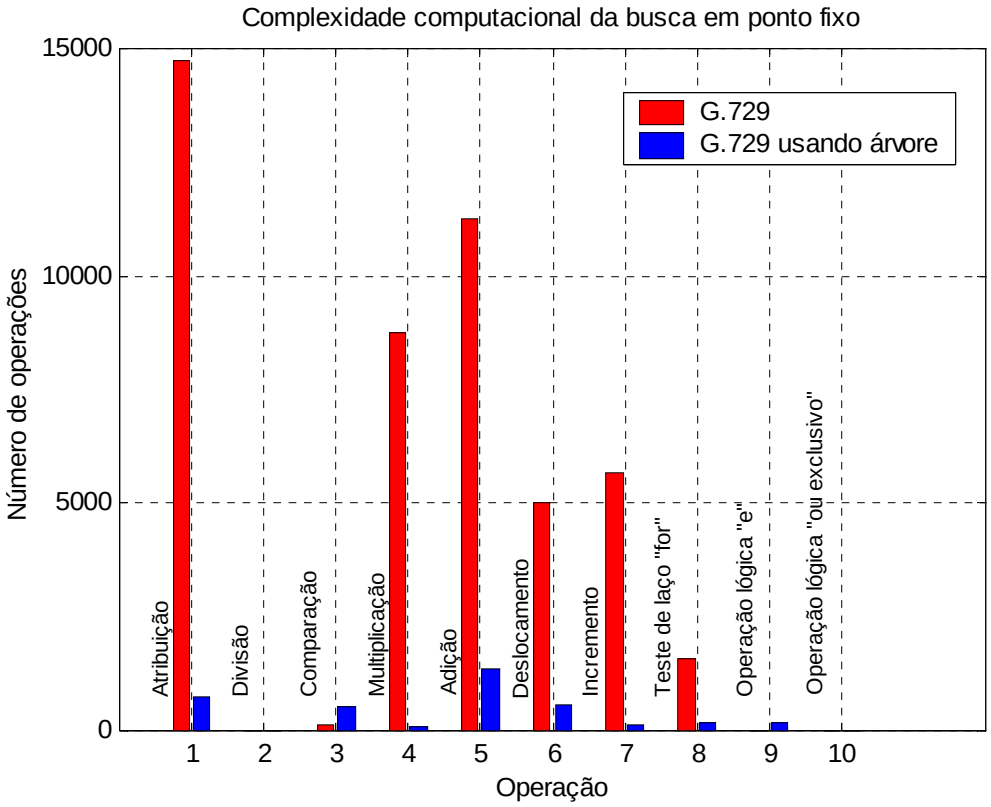


Fig. 5.6 Número de operações em função do tipo, medidas sobre um subquadro, para a busca da melhor excitação fixa nos codificadores G.729 padrão e G.729 usando árvore. Para o G.729 padrão os valores resultam da média sobre 906 subquadros.

5.5 Tempo de codificação

A medida do tempo de execução permite uma idéia mais clara da redução de complexidade obtida com a abordagem em árvore. As medidas de tempo foram tomadas usando um arquivo contendo uma locução de 6 minutos e 15 segundos, na verdade uma concatenação de arquivos *speech.in*. Além disso, para cada tipo de medida foi tomada a média sobre 4 realizações. Isso porque houve uma pequena variação para o mesmo tipo de medida. A Fig. 5.7 mostra o tempo de codificação para o *codec* G.729 padrão, para o G.729 usando apenas o primeiro nível da árvore e G.729 usando os 13 níveis. Além disso o gráfico em barras mostra o tempo de codificação quando a excitação fixa está fora do esquema de codificação.

A porção do *codec* G.729 padrão encarregada da excitação fixa consome aproximadamente 34,23% do tempo total de codificação. A busca usando árvore, por outro lado, consome apenas 41,90% (considerando os 13 níveis) do valor consumido pelo *codec*

padrão. Ou seja, a busca no *codebook* fixo usando árvore é aproximadamente 2,4 vezes mais rápida do que a busca análoga no *codec* padrão. Pelo resultado apresentado na Fig. 5.6, essa rapidez deveria acontecer com uma proporção maior, entretanto essa aparente predominância fica um pouco comprometida em virtude dos preparativos para a busca, que inclui por exemplo o cálculo de todos os elementos da matriz de correlações. Na verdade, como pode ser observado, esse é o maior custo da busca, pois praticamente não existe diferença de tempo ao se usar 1 ou mais níveis. Mesmo assim, o resultado é substantivo, principalmente quando se pensa em aplicações envolvendo um grande número de canais.

Apesar de não ter sido alvo de teste, é justo mencionar o desempenho do *codec* G.729A [97], uma simplificação do G.729. A especificação do ITU nasceu a partir da necessidade de multiplexar voz e dados, simultaneamente, para atender determinadas aplicações, onde um dos requisitos era a baixa complexidade (havia uma referência máxima de 10 MIPS – *Million Instructions Per Second*). O G.729A foi a proposta vencedora ao conjunto de requisitos e um dos fatores preponderantes foi sua compatibilidade com o G.729. Para atender a especificação do ITU, o G.729 passou por algumas simplificações no seu algoritmo, principalmente: no filtro ponderador do erro, na análise de *pitch* em laço aberto, no cálculo da resposta ao impulso do filtro de síntese ponderado, no procedimento de busca no *codebook* adaptável e busca no *codebook* fixo. De todas as modificações, a da busca no *codebook* algébrico foi a que produziu maior efeito. O anexo A reduz a complexidade computacional do G.729 (estimada em 20 MIPS aproximadamente) em 50%, e desse total, 50% (ou 5 MIPS) são atribuídos à alteração no processo de busca da excitação fixa, tudo isso ao custo de 0,2 dB de perda na relação sinal-ruído, SNR. Enquanto no G.729 o número de combinações de posição testadas chega a 1440 (no pior caso), no G.729A esse número cai para 320 (22%). É uma redução bastante significativa para uma perda na relação sinal-ruído tão pequena. Vale salientar apenas que o G.729A é o resultado de um trabalho focado na redução da complexidade, enquanto que o objetivo dessa proposta vai na direção de se testar alternativas multi-taxa.

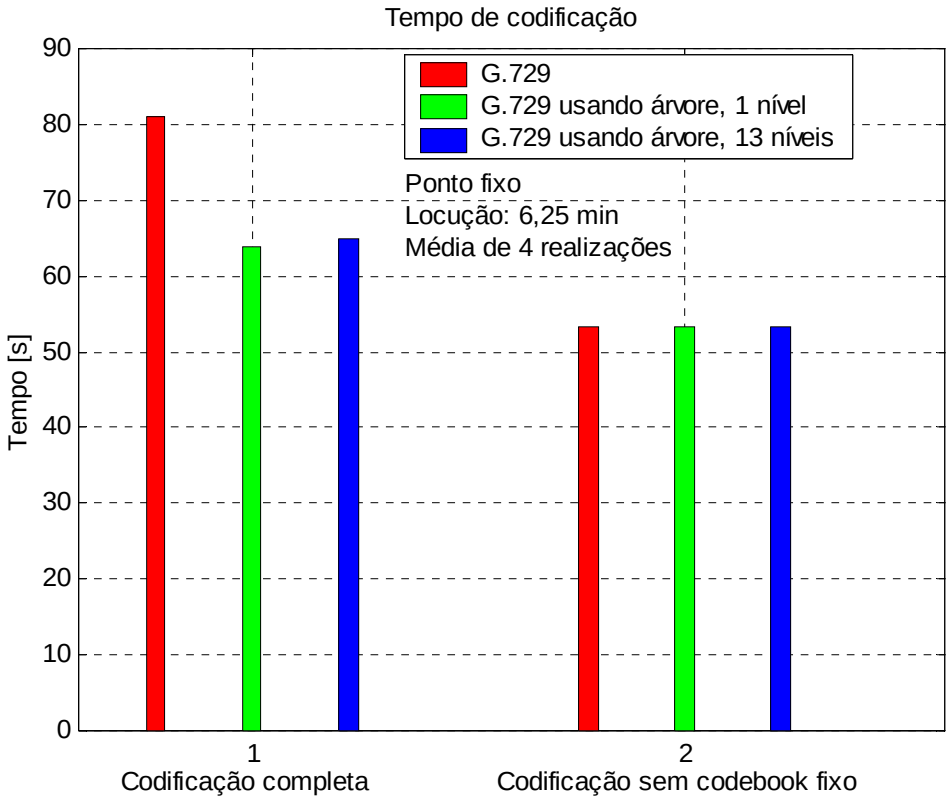


Fig. 5.7 Tempo de execução para os codificadores G.729 e G.729 usando árvore. Os valores obtidos resultam da média sobre 4 realizações. A locução usada tem duração de 6 minutos e 15 segundos, resultante da concatenação de vários arquivos *speech.in*.

5.6 Variabilidade da taxa

A estrutura proposta oferece 13 possibilidades de taxa, separadas por um passo fixo de 0,2 kbit/s (Tab. 4.3), com diferença na relação sinal-ruído entre a taxa mínima (5,6 kbit/s) e a máxima (8,0 kbit/s) de aproximadamente 1 dB (Fig. 5.1). A variação quase linear da relação sinal-ruído com a taxa, a pequena taxa de variação (0,07 dB/passo), observados na Fig. 5.1, e a ausência de troca de contexto de codificação permite ao codificador adequar-se às condições da rede sem que haja um forte impacto perceptível na qualidade da voz reconstruída. A Fig. 5.8 é apenas ilustrativa e mostra o conjunto de taxas, para esta proposta e para o *codec* AMR, que é um dos poucos com possibilidade de variação relativamente ampla.

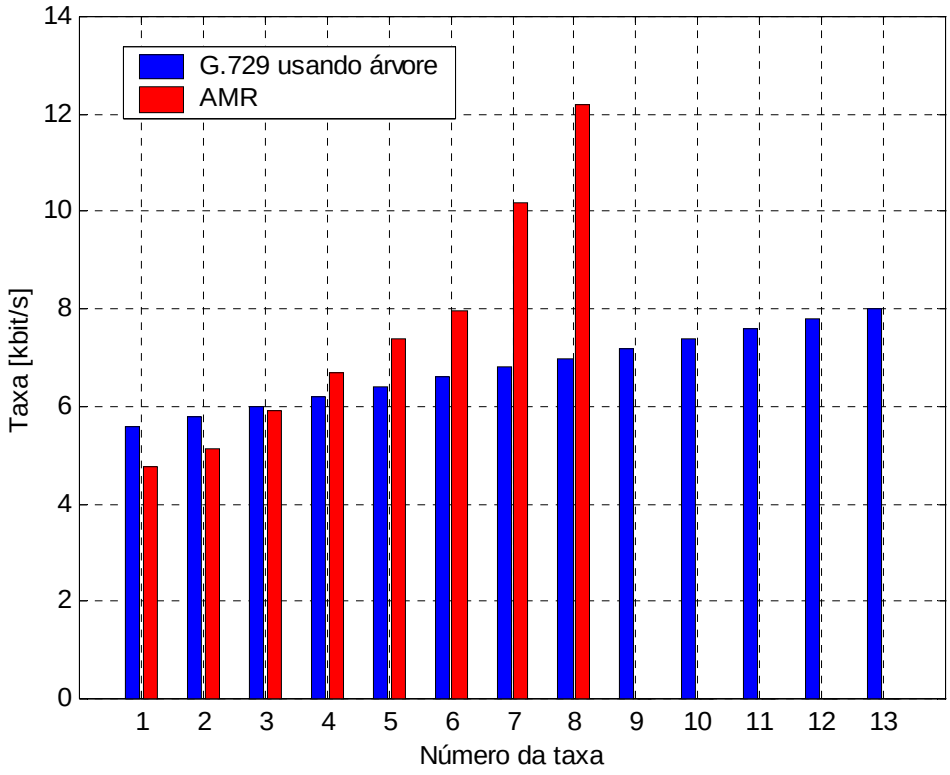


Fig. 5.8 Taxas disponíveis nos codificadores G.729 usando árvore e AMR.

A partir de um codificador de taxa variável é possível a concepção de mecanismos de controle de taxa com base nas condições da rede ou qualidade subjetiva da voz reconstruída [98]. A Fig. 5.9 mostra um esquema básico de como seria um sistema de transmissão VoIP com adaptação da taxa de envio guiada pelas condições da rede ou qualidade da voz reconstruída.

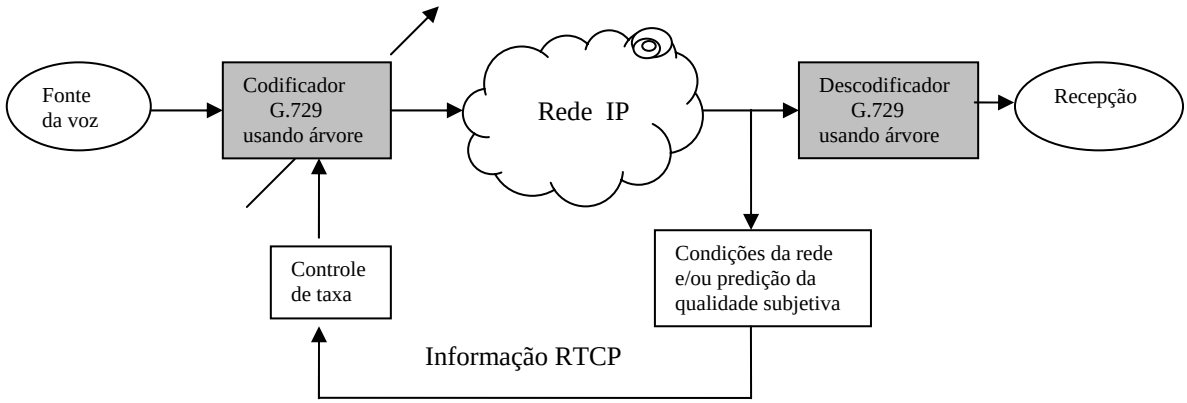


Fig. 5.9 Esquema de controle de taxa para o codificador G.729 usando árvore com base nas condições da rede e qualidade subjetiva da voz reconstruída.

Outro aspecto a ser considerado é a importância de se ter várias taxas com passo pequeno entre elas. Isso permite um aproveitamento melhor da banda disponível na rede, pois permite que o transmissor se ajuste de forma mais precisa à sua variação.

5.7 Conclusão

Os resultados apresentados neste capítulo mostram que a busca em *codebook* fixo, na estrutura CS-ACELP, usando árvore, é uma estratégia possível e viável. Apesar da ausência de testes subjetivos com critério, o relato de duas pessoas foi bastante positivo, o que concorda com as medidas de distorção. Apesar da especificação G.729 contar com uma variante de baixa complexidade (G.729A), a estrutura em árvore apresentada mostra resultados bastante satisfatórios em termos de complexidade computacional e tempo de execução. Embora a busca seja extremamente rápida, é preciso ainda algum trabalho no sentido da redução do tempo de preparação, especificamente no cálculo dos elementos da matriz de correlação.

O aspecto pouco favorável da abordagem fica por conta das 7 kwords necessárias para armazenar os dados da árvore binária, embora este seja um custo aceitável para os atuais processadores digitais de sinais e fica diluído em aplicações com múltiplos canais.

Finalmente, os resultados a respeito das medidas de distorção e complexidade sustentam o que foi o objetivo maior deste estudo: chegar em um codificador multi-taxa usando uma abordagem em árvore para a excitação fixa no algoritmo CS-ACELP. Este codificador G.729 modificado disponibiliza um conjunto de 13 taxas, de 5,6 a 8,0 kbit/s com passo fixo de 0,2 kbit/s, com qualidade bastante aceitável.

C A P Í T U L O 6

CONCLUSÕES

6.1 Conclusões

Este trabalho apresentou um estudo sobre codificação multi-taxa, a partir do algoritmo CS-ACELP e da especificação G.729, cujo objetivo era propor um codificador com taxa variável, através da busca em *codebook* fixo estruturado em árvore, para aplicações VoIP. A variabilidade da taxa nos *codecs* é uma característica que tem despertado forte apelo, devido ao movimento de migração da transmissão de voz das redes de circuito para as redes de dados, muito apropriadas para o tráfego de "melhor esforço" mas ainda pouco adequadas para tráfegos com requisitos de tempo. Essa característica oferece flexibilidade para que o sistema se adapte às condições da rede e permite atender os compromissos de qualidade e capacidade. Com este escopo iniciamos o trabalho, abordando os principais problemas relacionados à transmissão de voz sobre as redes IP, passando por uma revisão sobre diversas estratégias relacionadas à codificação multi-taxa, sejam elas dirigidas pela fonte, pela rede ou por ambas. Em seguida discorremos sobre os princípios da análise-pela-síntese, base para a grande maioria dos *codecs* disponíveis, onde se inclui a especificação G.729, assunto tratado na seqüência.

O emprego de árvore na abordagem foi motivado pela relação direta com a variabilidade da taxa. Já a intenção de restringir o estudo ao bloco do codificador que determina a excitação fixa, partiu da observação de que tal etapa concentra a maior parte dos bits alocados na codificação; é onde está a maior carga computacional, e principalmente, porque existe certa tolerância de desvio da excitação fixa em relação ao seu valor ótimo. Além de limitar o trabalho à busca da melhor excitação fixa, optamos por estruturar o *codebook* fixo em árvore binária de forma que os nós terminais coincidissem com os vetores do *codebook* algébrico do modelo CS-ACELP, para que uma das possibilidades fosse justamente a compatibilidade com o padrão. Então mostramos, por outro trabalho, que a construção de uma árvore a partir dos nós terminais é uma tarefa difícil, o que nos levou a experimentar uma heurística onde os vetores CS-ACELP (apenas os não negativos) foram ordenados por valor de forma decrescente e a partir daí cada nó pai foi determinado pela média aritmética dos nós filhos.

Apesar de essa estratégia de construção apresentar problemas intrínsecos, dado que a geração do nó pai não segue o critério do menor impacto na distorção, os testes objetivos e subjetivos usando essa árvore, com centróides completos, foram bastante satisfatórios, praticamente indistinguíveis quando comparados com os resultados subjetivos obtidos usando a codificação padrão. Entretanto, dois fatores proibiram a utilização dessa construção: o grande espaço em memória para armazenar os nós dessa árvore de altura 13 (~ 640 kwords) e a grande quantidade de informação necessária para transmitir os sinais dos pulsos. Então propusemos uma simplificação nos centróides, retendo apenas as 4 componentes de maior energia, e também um procedimento para sua geração, nos níveis 12 e 13 da árvore. Armazenando apenas a posição das componentes de maior energia, através de um esquema que incluiu codificação diferencial, baixamos o espaço de armazenamento para 7 kwords aproximadamente.

Apesar de a simplificação ter sido significativa, o impacto na qualidade da voz reconstruída foi bastante moderado para os primeiros níveis da árvore, e praticamente imperceptível para os últimos. Por outro lado, essa distorção foi observável nas medidas de relação sinal-ruído e relação sinal-ruído segmentada. Mesmo assim, os resultados mostraram uma melhoria contínua nestas medidas conforme se caminhava para os nós finais, terminando com uma diferença de 0,60 e 0,68 dB nas relações SNR e SEGSNR, respectivamente, comparadas com as medidas da especificação padrão.

O trabalho ainda mostrou que o número de operações efetivamente consumidas no processo de busca da melhor excitação apresenta uma redução drástica, comparado ao padrão, usando a alternativa em árvore, mas contudo não se refletindo por completo no tempo de codificação devido aos cálculos preliminares, especificamente devido à determinação de todos os elementos da matriz de correlação. Ainda assim a busca em árvore é aproximadamente 2,4 vezes mais rápida do que a busca análoga no *codec* padrão.

Finalmente mostramos um esboço de como seria um sistema de codificação de taxa variável usando a estrutura proposta.

6.2 Algumas possibilidades de estudo

Alguns pontos poderiam ser investigados:

- Avaliar os resultados subjetivos por meio de critérios estabelecidos. Verificar os resultados na presença de ruído de fundo e em operações tipo *tandem*. Fazer uma comparação com codificadores multi-taxa;
- Implementar o algoritmo para funcionar em ambiente embarcado, usando DSP, e avaliar seu comportamento;
- Propor uma alternativa de construção da árvore que cause menos distorção; e reduzir o número de operações no cálculo da matriz de correlação.

A N E X O 1

Regras para a geração do nível 12

O conjunto de regras a seguir permite gerar os nós filhos no nível 12 a partir do nó pai no nível 11 e seus bits de controle definidos na Fig. 4.5. A função que gera os nós filhos recebe como parâmetros de entrada: o número do nó pai no nível 11 e um ponteiro para as 4 componentes do nó pai. Como parâmetros de saída: um ponteiro para as 4 componentes do filho da esquerda e um ponteiro para as 4 componentes do filho da direita.

Caso 1: pai e filho da esquerda são iguais (1311 dos 2048 nós do nível 11 caem neste caso).

- Primeiramente são pesquisados os casos particulares. Se o nó pai pertencer a um conjunto de 102 nós exceção, então os filhos da esquerda e direita são lidos diretamente da memória;
- Se não é exceção faz:

fe(0...3) = pai(0...3)
fd(0...2) = fe(0...2)
r = resto (fe(3) / 5)
se *fe(3) < 30* ou *fe(3) = 33* então,
se *r = 3* faz *fd(3) = fe(3) + 5*
senão faz *fd(3) = fe(3) + 10*
senão, se *fe(3) = 30* faz *fd(2) = 30* e *fd(3) = 33*
senão, se *fe(3) = 31* faz *fd(2) = 31* e *fd(3) = 35*
senão, se *fe(3) = 32* faz *fd(2) = 32* e *fd(3) = 35*
senão, se *fe(3) = 35* faz *fd(2) = 35* e *fd(3) = 38*
senão, se *fe(3) = 36* faz *fd(2) = 36* e *fd(3) = 38*
senão, se *fe(3) = 37* faz *fd(2) = 37* e *fd(3) = 38*

Caso 2: pai e filho da esquerda são iguais apenas nas três primeiras componentes (562 dos 2048 nós do nível 11 caem neste caso).

- Faz-se a pesquisa nos casos particulares. Se o nó pai pertencer a um conjunto de 56 nós exceção, então os filhos da esquerda e direita são lidos diretamente da memória;
- Se não é exceção faz:

MaxRaia(0...4) = {35,36,37,38,39}
faz *fe(0) = pai(0)* e *MaxRaia(resto (pai(0) / 5)) = 0*
faz *fe(1) = pai(1)* e *MaxRaia(resto (pai(1) / 5)) = 0*
faz *fe(2) = pai(2)* e *MaxRaia(resto (pai(2) / 5)) = 0*
se *fd(0) ! = pai(0)* então,
    faz *i* variar de *0* até *4*
        se *MaxRaia(i) ! = 0* e *MaxRaia(i) ! = 38* e *MaxRaia(i) ! = 39* então,
            fe(3) = MaxRaia(i) - 5 e marca que *Achou*
senão,
    faz *i* variar de *0* até *4*
        se *MaxRaia(i) ! = 0* e *MaxRaia(i) ! = 38* e *MaxRaia(i) ! = 39* então
            fe(3) = MaxRaia(i)
fd(0) = fe(0)
fd(1) = fe(1)
fd(2) = pai(3)
MaxRaia(0...4) = {35,36,37,38,39}
faz *MaxRaia(resto (fd(0) / 5)) = 0*
faz *MaxRaia(resto (fd(1) / 5)) = 0*
faz *MaxRaia(resto (fd(2) / 5)) = 0*
faz *i* variar de *0* até *4*
    se *MaxRaia(i) ! = 0* então faz *aux = MaxRaia(i)*
fd(3) = aux
enquanto *fd(3) > fd(2)* faz *fd(3) = fd(3) - 5*
fd(3) = fd(3) + 5
se não *Achou* faz *fd(3) = fd(3) + 5*

Caso 3: pai e filho da esquerda são iguais nas duas primeiras componentes (136 dos 2048 nós do nível 11 caem neste caso).

- Nesta situação não foi possível formular regras. Nos 136 casos os filhos da esquerda e direita são lidos diretamente da memória.

Caso 4: pai e filho da esquerda são iguais apenas na primeira componente (9 dos 2048 nós do nível 11 caem neste caso).

- Novamente não foi possível formular regras. Nos 9 casos os filhos da esquerda e direita são lidos diretamente da memória.

Caso 5: pai e filho da esquerda são iguais na primeira e última componentes (17 dos 2048 nós do nível 11 caem neste caso).

- Nos 17 casos os filhos da esquerda e direita são lidos diretamente da memória.

Caso 6: pai e filho da esquerda diferem apenas na terceira componente (13 dos 2048 nós do nível 11 caem neste caso).

- Nos 13 casos os filhos da esquerda e direita são lidos diretamente da memória.

A N E X O 2

Regras para a geração do nível 13

O conjunto de regras a seguir permite gerar os nós filhos no nível 13 a partir do nó pai no nível 12. A função geradora recebe como parâmetros de entrada: o número do nó pai no nível 12 e um ponteiro para as 4 componentes do nó pai. Como saída: um ponteiro para as 4 componentes do filho da esquerda e outro para as 4 componentes do filho da direita.

MaxRaia(0...4) = {35,36,37,38,39}
fe(0...3) = pai(0...3) \\ pai e filhos da esquerda são iguais
se *fe(3) ! = MaxRaia(r = resto (fe(3) / 5))* ou se *fe(3) = = 38*
fd(0) = fe(0)
fd(1) = fe(1)
fd(2) = fe(2)
se *r == 3* então *fd(3) = fe(3) + 1*
senão, se *r == 4* então *fd(3) = fe(3) + 4*
senão, *fd(3) = fe(3) + 5*
senão, se *fe(2) ! = MaxRaia(r = resto (fe(2) / 5))* ou se *fe(2) = = 38*
fd(0) = fe(0)
fd(1) = fe(1)
se *r == 3* então *fd(2) = fe(2) + 1*
senão, se *r == 4* então *fd(2) = fe(2) + 4*
senão, *fd(2) = fe(2) + 5*
aux = MaxRaia(fe(3) / 5)
se *aux = 39* então faz *DEC = 01* e *VALXOR = 05*
senão faz *DEC = 05* e *VALXOR = 00*
enquanto *(aux – DEC) > fe(2)* faz
aux = aux – DEC
DEC = DEC ^= VALXOR // faz um “ou exclusivo”
fd(3) = aux
senão, se *fe(1) ! = MaxRaia(r = resto (fe(1) / 5))* ou se *fe(1) = = 38*

fd(0) = fe(0)

se *r == 3* então *fd(1) = fe(1) + 1*

senão, se *r == 4* então *fd(1) = fe(1) + 4*

senão, *fd(1) = fe(1) + 5*

aux = MaxRaia(fe(2) / 5)

se *aux = 39* então faz *DEC = 01* e *VALXOR = 05*

senão faz *DEC = 05* e *VALXOR = 00*

enquanto *(aux – DEC) > fe(1)* faz

aux = aux – DEC

DEC = DEC ^= VALXOR

fd(2) = aux

aux = MaxRaia(fe(3) / 5)

se *aux = 39* então faz *DEC = 01* e *VALXOR = 05*

senão faz *DEC = 05* e *VALXOR = 00*

enquanto *(aux – DEC) > fe(1)* faz

aux = aux – DEC

DEC = DEC ^= VALXOR

fd(3) = aux

senão, se *fe(0) ! = MaxRaia(r = resto (fe(0) / 5))* ou se *fe(0) = = 38*

se *r == 3* então *fd(0) = fe(0) + 1*

senão, se *r == 4* então *fd(0) = fe(0) + 4*

senão, *fd(0) = fe(0) + 5*

aux = MaxRaia(fe(1) / 5)

se *aux = 39* então faz *DEC = 01* e *VALXOR = 05*

senão faz *DEC = 05* e *VALXOR = 00*

enquanto *(aux – DEC) > fe(0)* faz

aux = aux – DEC

DEC = DEC ^= VALXOR

fd(1) = aux

aux = MaxRaia(fe(2) / 5)

se *aux = 39* então faz *DEC = 01* e *VALXOR = 05*

senão faz *DEC = 05* e *VALXOR = 00*

enquanto *(aux – DEC) > fe(0)* faz

aux = aux – DEC

DEC = DEC ^= VALXOR

fd(2) = aux

aux = MaxRaia(fe(3) / 5)

se *aux = 39* então faz *DEC = 01* e *VALXOR = 05*

senão faz *DEC = 05* e *VALXOR = 00*

enquanto *(aux – DEC) > fe(0)* faz

aux = aux – DEC

DEC = DEC ^= VALXOR

fd(3) = aux

dispor *fd(0...3)* em ordem crescente

A N E X O 3

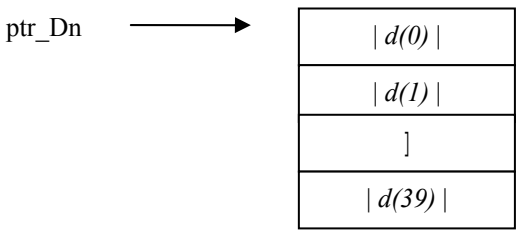
Estrutura para as correlações da matriz Φ

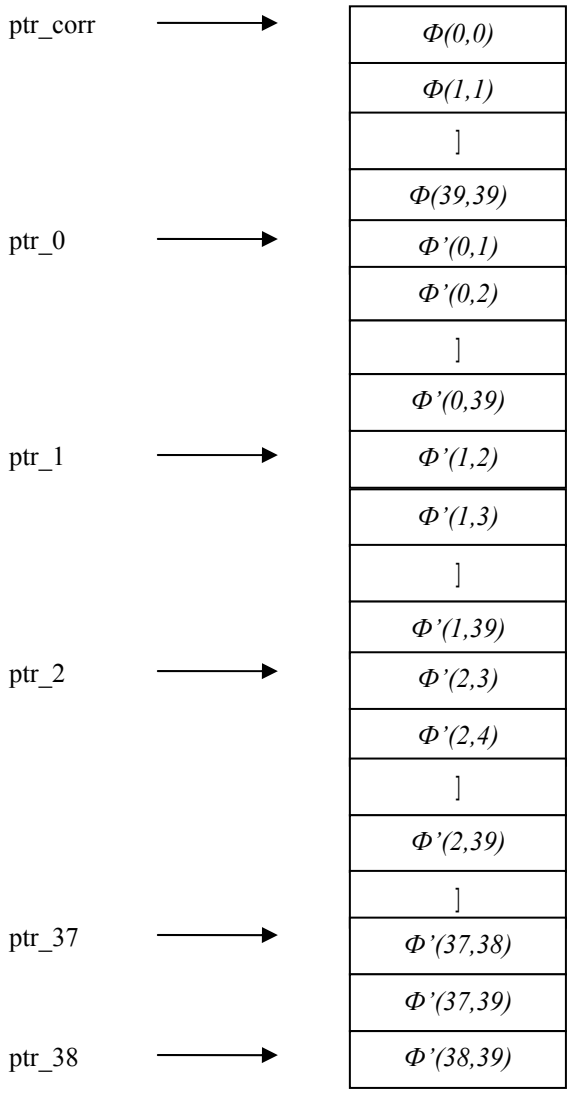
A melhor excitação é aquela que maximiza o termo $\backslash_k$ dado pela equação 3.38, mas como na estrutura proposta existem apenas 4 componentes não nulas para cada vetor de excitação candidato, torna-se possível determinar $\backslash_k$ usando as equações 3.44 e 3.45, aqui reescritas por conveniência.

$$C = |d(m_0)| + |d(m_1)| + |d(m_2)| + |d(m_3)| \tag{3.44}$$

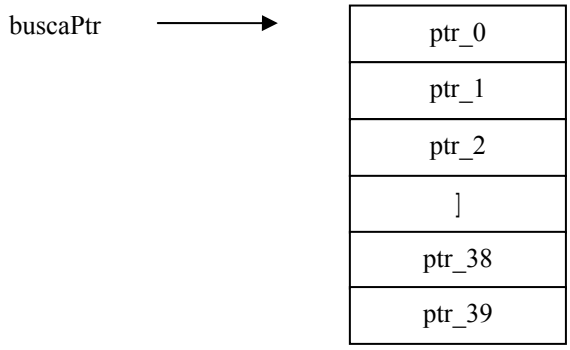
$$\begin{aligned} E/2 = {} & \phi'(m_0, m_0) + \\ & \phi'(m_1, m_1) + \phi'(m_0, m_1) + \\ & \phi'(m_2, m_2) + \phi'(m_0, m_2) + \phi'(m_1, m_2) \\ & \phi'(m_3, m_3) + \phi'(m_0, m_3) + \phi'(m_1, m_3) + \phi'(m_2, m_3) \end{aligned} \tag{3.45}$$

Para acelerar o cálculo das equações 3.44 e 3.45 o módulo de *d(n)* e as correlações modificadas da matriz Φ são dispostos segundo a estrutura:





O esquema ainda envolve a estrutura auxiliar:



O trecho de código C abaixo mostra como é simples o cálculo de l_k para os filhos da esquerda e direita. O denominador E/2 da equação 3.45 é dividido em duas partes: Et1_, para as correlações da diagonal da matriz Φ e Et2_, para as correlações fora da diagonal.

```
// carrega as posições das componentes não nulas, para os filhos a esquerda
// (FE) e direita (FD)

FEval_0 = *ptrFE;
FEval_1 = *(ptrFE+1);
FEval_2 = *(ptrFE+2);
FEval_3 = *(ptrFE+3);
FDval_0 = *ptrFD;
FDval_1 = *(ptrFD+1);
FDval_2 = *(ptrFD+2);
FDval_3 = *(ptrFD+3);

// carrega os ponteiros para as correlações fora da diagonal principal, para
// os dois filhos

ptrfe_a  =  *(buscaPtr + FEval_0);
ptrfe_b  =  *(buscaPtr + FEval_1);
ptrfe_c  =  *(buscaPtr + FEval_2);
ptrfd_a  =  *(buscaPtr + FDval_0);
ptrfd_b  =  *(buscaPtr + FDval_1);
ptrfd_c  =  *(buscaPtr + FDval_2);

// calculo do numerador e primeiro termo do denominador, para os dois filhos

C_fe   = add(*(ptr_Dn+FEval_0),
         add(*(ptr_Dn+FEval_1),
         add(*(ptr_Dn+FEval_2),*(ptr_Dn+FEval_3))));
C_fd   = add(*(ptr_Dn+FDval_0),
         add(*(ptr_Dn+FDval_1),
         add(*(ptr_Dn+FDval_2),*(ptr_Dn+FDval_3))));
Et1_fe = L_add(*(ptr_corr+FEval_0),
         L_add(*(ptr_corr+FEval_1),
         L_add(*(ptr_corr+FEval_2),*(ptr_corr+FEval_3))));
Et1_fd = L_add(*(ptr_corr+FDval_0),
         L_add(*(ptr_corr+FDval_1),
         L_add(*(ptr_corr+FDval_2),*(ptr_corr+FDval_3))));

// calculo do segundo termo do denominador, para os dois filhos
```

```
Et2_fe = L_add(*(ptrfe_a+FEval_1-FEval_0-1),
         L_add(*(ptrfe_a+FEval_2-FEval_0-1),
         L_add(*(ptrfe_a+FEval_3-FEval_0-1),
         L_add(*(ptrfe_b+FEval_2-FEval_1-1),
         L_add(*(ptrfe_b+FEval_3-FEval_1-1),
               *(ptrfe_c+FEval_3-FEval_2-1))))));
Et2_fd = L_add(*(ptrfd_a+FDval_1-FDval_0-1),
         L_add(*(ptrfd_a+FDval_2-FDval_0-1),
         L_add(*(ptrfd_a+FDval_3-FDval_0-1),
         L_add(*(ptrfd_b+FDval_2-FDval_1-1),
         L_add(*(ptrfd_b+FDval_3-FDval_1-1),
               *(ptrfd_c+FDval_3-FDval_2-1))))));
```

A N E X O 4

Configuração do ambiente de teste e procedimento para execução do algoritmo

Procedimento para a instalação dos aplicativos

1. Instalar o ambiente DJGPP no diretório C, pasta C:\DJGPP (tem de ser o C);

2. No Windows XP ou 2000:
   - Clique com o botão da direita do mouse em “Meu Computador”;
   - Selecione “Propriedades”;
   - Selecione “Avançado”;
   - Selecione “Variáveis de ambiente”;
   - Em “variáveis de usuário” entre com “Nova” variável = DJGPP, valor = C:\DJGPP\DJGPP.ENV;
   - Ainda em “variáveis de usuário” entre com “Nova” variável = Path, valor = C:\DJGPP\BIN;
   - Em “variáveis do sistema” entre com “Nova” variável = DJGPP, valor = C:\DJGPP\DJGPP.ENV;

3. Copiar o projeto e fontes do G.729 na pasta C:\C++\CELP;

4. Instalar o Dev-C++ na pasta C:\Dev-Cpp;

5. Instalar um programa para processamento de áudio (exemplo: SoundForge 4.0) no diretório C.

Procedimento para a codificação e descodificação

1. Definir o tipo de busca e número de níveis da árvore:
   - Abrir o arquivo Coder.c e procurar a atribuição da variável N_C. Faça N_C = 9 (para taxa variável; 4 pulsos positivos por vetor; níveis 12 e 13 não armazenados; ponto fixo). Abrir o arquivo DECODER.c e fazer o mesmo;

- Abrir o arquivo ld8k.h e procurar a definição: #define NIVEL 13. Esse valor pode variar de 1 a 13, de acordo com o número de níveis desejado para a árvore;
- No mesmo arquivo ld8k.h definir SYNC (para que não haja atraso do arquivo de saída em relação ao de entrada), TAXAVAR e ARVORE_4PP_STORED.

2. Compilar o coder e o decoder:
   - C:\C++\CELP\make –f coder.mak
   - C:\C++\CELP\make –f decoder.mak

3. Executar o coder com o arquivo desejado, ex:
   - C:\C++\CELP\coder speech.in speechTeste.bit

4. Executar o decoder:
   - C:\C++\CELP\decoder speechTeste.bit speechTeste.pst

5. Para executar o coder e decoder G.729 padrão:
   - Abrir o arquivo Coder.c e procurar a atribuição da variável N_C. Faça N_C = 0;
   - No arquivo ld8k.h definir SYNC e não definir TAXAVAR.

Procedimento para o teste do áudio

1. Abrir o SoundeForge;
2. Abrir o arquivo de entrada speech.in e o arquivo gerado speechTeste.pst, usando o tipo: “Raw File (*.raw;*.*)”;
3. Na abertura dos arquivos de áudio os seguintes parâmetros devem ser definidos:
   - Sample rate: 8.000;
   - Sample type: 16-bit PCM;
   - Format: signed;
   - Channels: mono;
   - Byte order: Little endian (Intel);
   - Header...: 0;
   - Trailer:...: 0.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] COMER, D. E. *Internetworking with TCP/IP Principles, Protocols, and Architectures*. 4 ed. Prentice Hall, 2000.

[2] WEINSTEIN, C. J.; FORGIE, J.W. Experience with Speech Communication in Packet Networks. *IEEE Journal on Selected Areas in Communications*, v. SAC-1, n. 6, p.1022-1028, Dec. 1983.

[3] BELLAMY, J. C. *Digital Telephony*, 3 ed. John Wiley & Sons, Inc., 2000.

[4] TANENBAUM, A. S. *Redes de* Computadores. 3 ed. Editora Campus, 1997.

[5] HASSAN, M.; NAYANDORO, A.; ATIQUZZAMAN, M. Internet Telephony: Services, Technical Challenges, and Products. *IEEE Communications Magazine*, p. 96-103, Apr. 2000.

[6] CHERRY, S. Seven Myths About Voice over IP. *IEEE Spectrum*, p. 52-57, Mar. 2005.

[7] GITMAN, I.; FRANK, H. Economic Analysis of Integrated Voice and Data Networks: A Case Study. *Proceedings of IEEE*, v. 66, n. 11, p. 1549-1570, Nov. 1978.

[8] CHONG, H. M.; MATTHEWS, H. S. Comparative Analysis of Traditional Telephone and Voice-over-Internet Protocol (VoIP) Systems. In: INTERNATIONAL SYMPOSIUM ON ELECTRONICS AND THE ENVIRONMENT. *Conference Record*. May 2004, p. 106-111.

[9] SCHULZRINNE, H.; ROSENBERG, J. Internet Telephony: Architecture and Protocols an IETF Perspective. *Computer Networks*, v. 31, n. 3, Feb. 1999.

[10] MARKOPOULOU, A. P.; TOBAGI, F. A.; KARAM, M. J. Assessment of VoIP Quality over Internet Backbones. *IEEE INFOCOM'02*. 2002, v.1, p. 150-159.

[11] BOLOT, J.-C.; VEGA-GARCÍA, A. Control Mechanisms for Packet Audio in the Internet. *IEEE INFOCOM'96*. Nov.1996, v. 1, p. 232-239.

[12] PERKINS, C.; HODSON, O.; HARDMAN, V. A Survey of Packet Loss Recovery Techniques for Streaming Audio. *IEEE Network*, v. 12, n. 5, p. 40-48, Sep.-Oct. 1998.

[13] BOLOT, J.-C.; FOSSE-PARISIS, S.; TOWSLEY, D. Adaptive FEC-Based Error Control for Internet Telephony. *IEEE INFOCOM'99*. Mar. 1999, v. 3, p. 1453-1460.

[14] WANG, J.; GIBSON, J. D. Parameter Interpolation to Enhance the Frame Erasure Robustness of CELP Coders in Packet Networks. *IEEE ICASSP'01*. Salt Lake City, UT, Jun. 2001, p.745-748.

[15] WANG, J.; GIBSON, J. D. Erasure-Robust Speech Coding and Concealment in VoIP Systems with Frame Packetization. In: SYMPOSIUM ON BROADBAND COMMUNICATIONS FOR THE INTERNET. *Era Symposium digest*. Richardson, TX, Sep. 2001, p.23-27.

[16] AZAMI, S. B. Z.; YONGAÇOGLU, A.; OROZCO-BARBOSA, L. et al. Evaluating the Effects of Buffer Management on Voice Transmission over Packet Switching Networks. In: ITERNATIONAL CONFERENCE ON COMMUNICATIONS. Helsinki, 2001, p. 738-742.

[17] SERIZAWA, M.; ITO, H. A Packet Loss Recovery Method Using Packet Arrived Behind the Playout Time for CELP Decoding. *IEEE ICASSP'02*. 2002, v.1, p. 69-172.

[18] KOS, A.; KLEPEC, B.; TOMAZIC, S. Techniques for Performance Improvement of VoIP Applications. *IEEE MELECON'02*. Cairo, May 2002, p. 250-254.

[19] MONTGOMERY, W. A. Techniques for Packet Voice Synchronization. *IEEE Journal on Selected Areas in Communications*, v. SAC-1, n. 6, p. 1022-1028, Dec. 1983.

[20] ALVAREZ-CUEVAS, F.; BERTRAN, M.; OLLER, F. et al. Voice Synchronization in Packet Switching Networks. *IEEE Network*, v. 7, p. 20-25, Sep. 1993.

[21] RAMJEE, R.; KUROSE, J.; TOWSLEY, D. et al. Adaptive Playout Mechanisms for Packetized Audio Applications in Wide-Area Networks. *IEEE INFOCOM'04*. 1994, p. 680-688.

[22] MOON, S. B.; KUROSE, J.; TOWSLEY, D. Packet Audio Playout Delay Adjustment: Performance Bounds and Algorithms. *Multimedia Systems*, v. 5, n. 1, p. 17-28, 1998.

[23] MATIC, V.; BEZANT, A.; KOS, M. Predictive Playout Delay Adaptation for Voice over Internet. *IEEE MELECON'00*. 2000, v. 1, p. 348-352.

[24] LIANG, Y. J.; FARBER, N.; GIROD, B. Adaptive Playout Scheduling Using Time-Scale Modification in Packet Voice Communications. *IEEE ICASSP'01*. Salt Lake City, UT, 2001, p. 1445-1448.

[25] AGNIHOTRI, S.; ARAVINDHAN, K.; JAMADAGNI, H. S. et al. A New Technique for Improving Quality of Speech in Voice over IP Using Time-Scale Modification. *IEEE ICASSP'02*. Orlando, FL, 2002, v. 2, p. 2085-2088.

[26] HATA, H. Playout Buffering Algorithm Using of Randomwalk in VoIP. In: INTERNATIONAL SYMPOSIUM ON COMMUNICATIONS AND INFORMATION TECHNOLOGIES. Sapporo, 2004, p. 457-460.

[27] NARBUTT, M.; MURPHY, L. A New VoIP Adaptive Playout Algorithm. *IEE Telecommunications Quality of Services*, p. 99-103, Mar. 2004.

[28] SUN, L.; IFEACHOR, E. New Models for Perceived Voice Quality Prediction and their Applications in Playout Buffer Optimization for VoIP Networks. In: INTERNATIONAL CONFERENCE ON COMMUNICATIONS. 2004, p. 1478-1483.

[29] TSENG, K.-K.; LAI, Y.-C.; LIN, Y.-D. Perceptual Codec and Interaction Aware Playout Algorithms and Quality Measurement for VoIP Systems. *IEEE Transactions on Consumer Electronics*, V. 50, N. 1, p. 297-305, Feb. 2004.

[30] NARBUTT, M.; KELLY, A.; MURPHY, L. et al. Adaptive VoIP Playout Scheduling: Assessing User Satisfaction. *IEEE Internet Computing*, v. 9, p. 28-34, Jul.-Aug. 2005.

[31] BALDI, M.; RISSO, F. Efficiency of Packet Voice with Deterministic Delay. *IEEE Communications Magazine*, p. 170-177, May 2000.

[32] REJAIE, R.; HANDLEY, M.; ESTRIN, D. RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet. *IEEE INFOCOM'99*. Mar. 1999, p. 1337-1345.

[33] HOMAYOUNFAR, K. Rate Adaptive Speech Coding for Universal Multimedia Access. *IEEE Signal Processing Magazine*, p. 30-39, Mar. 2003.

[34] BARBERIS, A.; CASETTI, C.; DE MARTIN, J. C. et al. A Simulation Study of Adaptive Voice Communications on IP Networks. *Computer Communications*, p. 757-767, 2001.

[35] Khalid Sayood, *Introduction to Data Compression*, 2 ed. Academic Press, 2000.

[36] HANZO, L.; SOMERVILLE, F. C. A.; WOODARD, J. P. *Voice Compression and Communications – Principles and Applications for Fixed and Wireless Channels*. Wiley-Interscience, 2001.

[37] SPANIAS, A. S. Speech Coding: A Tutorial Reviews. *Proceedings of the IEEE,* vol. 82, n. 10, pp. 1541–1582, Oct. 1994.

[38] WICHMAN, S. A Comparison of Speech Coding Algorithms ADPCM *vs* CELP. *Department of Electrical Engineering – The University of Texas at Dallas,* pp. 1-10, Dec. 1999.

[39] EQUITZ, W. H. A New Vector Quantization Clustering Algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing.* v. 37, n. 10, p. 1568-1575, Oct. 1989.

[40] GERSHO, A.; GRAY, R. M. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.

[41] GERSHO, A.; PAKSOY, E. An Overview of Variable Rate Speech Coding for Cellular Networks. In: INTERNATIONAL CONFERENCE SELECTED TOPICS IN WIRELESS COMMUNICATIONS. *Proceedings*. 1992, p.172-175.

[42] WANG, S.; GERSHO, A. Phonetically-Based Vector Excitation Coding of Speech at 3.6 kbps. *IEEE ICASSP'89*. Glasgow, UK, May 1989, v. 1, p.49-52.

[43] WANG, S.; GERSHO, A. Improved Phonetically-Segmented Vector Excitation Coding at 3.4 kb/s. *IEE*E ICASSP'92. San Francisco, CA, Mar. 1992, v. 1, p.349-352.

[44] PAKSOY, E.; SRINIVASAN, K.; GERSHO, A. Variable Rate Speech Coding with Phonetic Segmentation. *IEEE ICASSP'93*. Minneapolis, MN, Apr. 1993, v. 2, p.155-158.

[45] PAKSOY, E.; GERSHO, A. A Variable Rate Speech Coding Algorithm for Cellular Networks. *IEEE Workshop Speech Coding Telecommunications*. Ste. Adele, P.Q., 1993, p.109-110.

[46] DI FRANCESCO, R.; LAMBLIN, C.; LEGUYADER, A. et al. Variable Rate Speech Coding with Online Segmentation and Fast Algebraic Codes. *IEEE ICASSP'90*. Albuquerque, NM, Apr.1990, v. 1, p.233-236.

[47] DAS, A.; GERSHO, A. A Variable-Rate Natural-Quality Parametric Speech Coder. *IEEE SUPERCOMM/ICC'94*. May 1994, v. 1, p.216-220.

[48] DAS, A.; GERSHO, A. Variable Dimension Spectral Coding of Speech at 2400 bps and Below with Phonetic Classification. *IEEE ICASSP'95*. Detroit, MI, May 1995, v. 1, p.492-495.

[49] MCCLELLAN, S. A.; GIBSON, J. D. Variable-Rate CELP Based on Subband Flatness. *IEEE IEEE Transactions on Speech and Audio Processing*, v. 5, n. 2, p.120-130, 1997.

[50] VASEGHI, S. V. Finite State CELP for Variable Rate Speech Coding. *IEEE ICASSP'90*. Albuquerque, NM, Apr. 1990, v. 1, p.37-40.

[51] PAKSOY, E.; GERSHO, A. A Variable Rate Speech Coding for Multiple Access Wireless Networks. *IEEE MELECON'94*. Antalya, Turkey, Apr. 1994, vol. 1, p.47-50.

[52] ITU-T RECOMMENDATION G.729. Coding of Speech at 8 kb/s Using Conjugate-Structure Algebraic Code-Excited Linear-Prediction (CS-ACELP). Mar. 1996.

[53] CHUNG, W.-S.; KANG, S.-W.; SUNG, H.-S. et al. Design of a Variable Rate Algorithm for the 8 kb/s CS-ACELP Coder. In: VEHICULAR TECHNOLOGY CONFERENCE. Ottawa, Ont., May 1998, v. 3, p. 2378-2382.

[54] BERITELLI, F. A Modified CS-ACELP Algorithm for Variable-Rate Speech Coding Robust in Noisy Environments. *IEEE Signal Processing Letters*, v. 6, n. 2, p. 31-34, Feb. 1999.

[55] DONG, H.; GIBSON, J. D. Universal Successive Refinement of CELP Speech Coders. *IEEE ICASSP'01*. Salt Lake City, UT, 2001, p. 713-716.

[56] DeJaco, A.; GARDNER, W.; JACOBS, P. et al. QCELP: The North American CDMA Digital Cellular Variable Rate Speech Coding Standard. In: IEEE WORKSHOP ON SPEECH CODING FOR TELECOMMUNICATIONS. 1993, p. 5-6.

[57] JAYANT, N. S.; CHEN, J.-H. Speech Coding with Time-Varying Bit Allocations to Excitation and LPC Parameters. *IEEE ICASSP'89*. Glasgow, UK, May 1989, v. 1, p.65-68.

[58] HANZO, L.; WOODARD, J. P.; An Intelligent Multimode Voice Communications System for Indoor Communications. *IEEE Transactions on Vehicular Technology*, v. 44, n. 4, p. 735-748, Nov. 1995.

[59] ERDMANN, C.; BAUER, D.; VARY, P. Pyramid CELP: Embedded Speech Coding for Packet Communications. *IEEE ICASSP'02*. 2002, v. 1, p. I.181-I.184.

[60] EKUDDEN, E.; HAGEN, R.; JOHANSSON, I. et al. The Adaptive Multi-Rate Speech Coder". In: IEEE WORKSHOP ON SPEECH CODING. Porvoo, Finland, Jun. 1999, p.117-119.

[61] ERDMANN, C.; VARY, P.; FISCHER, K. et al. An Adaptive Multi Rate Wideband Speech Codec with Adaptive Gain Re-Quantization. In: IEEE WORKSHOP ON SPEECH CODING. Delavan, Wisconsin, Sep. 2000, p. 17-20.

[62] SEO, J. W.; WOO, S. J.; BAE, K. S. A Study on the Application of an AMR Speech Codec to VoIP. *IEEE ICASSP'01*. 2001, v. 3, p. 1373-1376.

[63] MAKINEN, J.; OJALA, P.; VAINIO J. The Effect of Source Based Rate Adaptation Extension in AMR-WB Speech Codec. In: IEEE WORKSHOP ON SPEECH CODING. Tsukuba, Ibaraki, Japan, Oct. 2002, p. 153-155.

[64] JOHANSSON, I.; FRANKKILA, T.; SYNNERGREN, P. Bandwidth Efficient AMR Operation for VoIP. In: IEEE WORKSHOP ON SPEECH CODING. Oct. 2002, p. 150-152.

[65] TAMMI, M.; JELÍNEK, M.; RUOPPILA, V. T. Signal Modification Method for Variable Bit Rate Wide-band Speech Coding. *IEEE Transactions on Speech and Audio Processing*, v. 13, n. 5, Sep. 2005.

[66] LUPINI, P.; COX, N. B.; CUPERMAN, V. A Multi-Mode Variable Rate CELP Coder Based on Frame Classification. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS. Geneva, Switzerland, May 1993, v. 1, p.406-409.

[67] CELLARIO, L.; SERENO, D. Variable Rate Speech Coding for UMTS. In: IEEE WORKSHOP ON SPEECH CODING FOR TELECOMMUNICATIONS. 1993, p.1-2.

[68] RUGGERI, G.; BERITELLI, F.; CASALE, S. Hybrid Multi-Mode/Multi-Rate CS-ACELP Speech Coding for Adaptive Voice Over IP. *IEEE ICASSP'01*. Salt Lake City, UT, May 2001, v. 2, p.733-736.

[69] ITU-T RECOMMENDATION Annex D. Annex D to Recommendation G.729: 6.4 kbit/s CS-ACELP Speech Coding. Tech. Rep., ITU-T, Sep. 1998.

[70] ITU-T RECOMMENDATION Annex E. Annex E to Recommendation G.729: 11.8 kbit/s CS-ACELP Speech Coding. Tech. Rep., ITU-T, Sep. 1998.

[71] KWONG, C. F.; PANG, W. M.; WU, H. C. et al. Simple DCT-Based Speech Coder for Internet Applications. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS. Vancouver, BC, Jun. 1999, v. 1, p.344-348.

[72] FANT, G. *Acoustic Theory of Speech Production*. Mounton and Co., 1960.

[73] HAYKIN, S. *Adaptive Filter Theory*. 4. ed. Prentice-Hall, Upper Saddle River, N. J., 2002.

[74] RABINER, L. R.; SCHAFER, R. W. *Digital Processing of Speech Signals*. Prentice-Hall, Upper Saddle River, N. J.,1978.

[75] FERNANDES, D. *Codificadores CELP: Implementação, Análise e Propostas para Redução de Complexidade e Taxa de Transmissão*. Florianópolis, 1992. Dissertação (Mestrado em Engenharia Elétrica) – Centro Tecnológico, Universidade Federal de Santa Catarina.

[76] SINGHAL, S.; ATAL, B. S. Improving Performance of Multi-Pulse LPC Coders at Low Bit Rates. *IEEE ICASSP'04*. Mar. 1984, v. 1, p. 1.3.1-1.3.4.

[77] ATAL, B. S. High-Quality Speech at Low Bit Rates: Multi-Pulse and Stochastically Excited Linear Predictive Coders. *IEEE ICASSP'86*. Tokyo, 1986, p. 1681-1684.

[78] ATAL, B. S.; REMDE, J. R. A New Model of LPC Excitation for Producing Natural-Sounding Speech at Low Bit Rates. *IEEE ICASSP'82*. 1892, p. 614-617.

[79] SCHROEDER, R.; ATAL, B. S. Code-Excited Linear Prediction (CELP): High-Quality Speech at Very Low Bit Rates. *IEEE ICASSP'85*. Mar. 1985, v. 1, p. 937-940.

[80] TRANCOSO, I. M.; ATAL, B. S. Efficient Procedures for Finding the Optimum Innovation in Stochastic Coders. *IEEE ICASSP'86*. Tokyo, 1986, p. 44.5.1-44.5.4.

[81] DAVIDSON, G.; GERSHO, A. Complexity Reduction Methods for Vector Excitation Coding. *IEEE ICASSP'86*. Tokyo, 1986, p. 56.8.1-56.8.4.

[82] ADOUL, J.-P.; MABILLEAU, P.; DELPRAT M. et al. Fast CELP Coding Based on Algebraic Codes. *IEEE ICASSP'87*. 1987, p. 1957-1960.

[83] ADOUL, J.-P; LAMBLIN, C. A Comparison of Some Algebraic Structures for CELP Coding of Speech. *IEEE ICASSP'87*. Apr. 1987, v. 12, p. 1953-1956.

[84] IRETON, M. A.; XYDEAS, C.S. On Improving Vector Excitation Coders Through the Use of Spherical Lattice Codebooks (SLC'S). *IEEE ICASSP'89*. Glasgow, UK, May 1989, v. 1, p. 57-60.

[85] LAMBLIN, C.; ADOUL, J. P.; MASSALOUX, D. et al. Fast Coding Based on the Barnes-Wall Lattice in 16 Dimensions. *IEEE ICASSP'89*. Glasgow, UK, May 1989, v. 1, p. 61-64.

[86] LAFLAMME, C.; ADOUL, J.-P.; SU, H. Y. et al. On Reducing Computational Complexity of Codebook Search in CELP Coder Through the Use of Algebraic Codes. *IEEE ICASSP'90*. Albuquerque, NM, Apr. 1990, v. 1, p. 177-180.

[87] KLEIJN, W. B.; KRASINSKI, D. J.; KETCHUM, R. H. Fast Methods for the CELP Speech Coding Algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. 38, n. 8, p. 1330-1342, Aug. 1990.

[88] ATAL, B. S. Predictive Coding of Speech at Low Bit Rates. *IEEE Transactions on Communications*, v. COM-30, n. 4, p. 600-614, Apr. 1982.

[89] SALAMI, R.; LAFLAMME, C.; ADOUL, J.-P. et al. Design and Description of CS-ACELP: A Toll Quality 8 kb/s Speech Coder. *IEEE Transactions on Speech and Audio Processing*, v. 6, n. 2, p. 116-130, Mar. 1998.

[90] LAFLAMME, C.; ADOUL, J. P.; SALAMI, R. et al. 16 kbps Wideband Speech Coding Technique Based on Algebraic CELP. *IEEE ICASSP'91*. Toronto, Ont., Apr. 1991, v. 1, p.13-16.

[91] BUZO, A.; GRAY JR., A. H.; GRAY, R. M. et al. Speech Coding Based Upon Vector Quantization. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, v. ASSP-28, n. 5, p. 562-574, Oct. 1980.

[92] GRAY, R. M.; LINDE, Y. Vector Quantizers and Predictive Quantizers for Gauss-Markov Sources. *IEEE Transactions on Communications*, v. COM-30, N. 2, Feb. 1982.

[93] GRAY, R. M.; ABUT, H. Full Search and Tree Search Vector Quantization of Speech Waveforms. *IEEE ICASSP'82*. 1982, p. 593-596.

[94] CHENG, D.-Y.; GERSHO, A. A Fast Codebook Search Algorithm for Nearest-Neighbor Pattern Matching. *IEEE ICASSP'86*. Tokyo, Apr. 1986, v. 1, p. 265-268.

[95] CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L *et al*. *Introduction to Algorithms*. 2. ed. The MIT Press, 2001.

[96] LINDE, Y.; BUZO, A.; GRAY, R. M. An Algorithm for Vector Quantizer Design. *IEEE Transactions on Communications*, v. COM-28, n. 1, p. 84-95, Jan. 1980.

[97] SALAMI, R.; LAFLAMME, C.; BESSETTE, B. et al. ITU-T G.729 Annex A: Reduced Complexity 8 kb/s CS-ACELP Codec for Digital Simultaneous Voice and Data. *IEEE Communications Magazine*, p. 56-63, Sep. 1997.

[98] QIAO, Z.; SUN, L.; HEILEMANN, N. et al. A new method for VoIP Quality of Service control use combined adaptive sender rate and priority marking. *IEEE Communications Society*, p. 1473-1477, 2004.