Escola Politécnica da USP
Departamento de Engenharia de Sistemas Eletrônicos

Codificação de sinais de voz por análise preditiva

Vítor H. Nascimento Magno T. M. Silva

Setembro de 2018

Nesta experiência será vista uma técnica para codificação de sinais de voz conhecida como *codificação linear preditiva* (LPC, *linear predictive coding*). Com essa abordagem um sinal de voz é representado com um pequeno número de parâmetros (em torno de 12), que podem ser codificados com um número pequeno de bits, levando a uma taxa de até 2400 bits por segundo, para um sinal de voz amostrado a 8 kHz. Note que isso representa uma taxa bem baixa, de apenas 0,3 bit/amostra, mas com perda considerável da qualidade do sinal. Codificadores usados em telefonia celular hoje em dia trabalham com uma taxa um pouco maior (entre 4,75 e 11,2 kbps), com uma qualidade consideravelmente maior.

Na segunda parte da experiência, veremos como a qualidade da reconstituição do sinal pode ser melhorada usando-se uma técnica conhecida por análise-por-síntese (*analysis-by-synthesis*, A-b-S). O texto a seguir é baseado em [2, 1].

Sumário

1 Introdução **2**

2 Estimação de *pitch* **3**

3 Estimação do ganho e de $H(z)$ **4**
3.1 O método da autocorrelação 7
3.2 O método da covariância 9
3.3 Cálculo do ganho do modelo 12

4 Parte experimental **12**

5 Codificação com realimentação (análise-por-síntese) **15**
5.1 Otimização dos parâmetros da excitação 17

6 Parte experimental **18**

1 Introdução

Um sinal de voz é produzido pelo ar passando pela laringe e excitando (ou não) as cordas vocais. O sinal acústico resultante é modificado pela passagem pelo trato vocal (faringe, boca, e, dependendo do caso, pela cavidade nasal). O espectro do sinal resultante é modificado dependendo da posição da mandíbula, língua, lábios, boca e palato (este último controla a passagem do ar pela cavidade nasal). O codificador que será estudado nesta experiência é baseado em um modelo idealizado e simplificado para a geração do sinal de voz, que separa a produção do sinal em quatro partes (veja a figura 1):

- Um gerador de sinal de excitação, que fornece dois tipos de sinal:
  1. Ruído branco Gaussiano (representa a excitação para o caso das cordas vocais não serem excitadas, gerando os assim chamados sons "surdos" ("não sonoros") — consoantes como /s/, /f/),
  2. Trem de pulsos com uma certa frequência fundamental F_0, chamada *pitch* (representa a excitação para o caso das cordas vocais serem excitadas, gerando os sons ditos "sonoros", como vogais),
- Uma chave que escolhe entre os dois tipos de excitação,
- Um ganho G que permite ajustar a intensidade do sinal sendo emitido,
- Um filtro linear variante no tempo que aproxima o espectro do sinal resultante (modelando o efeito combinado do espectro do sinal realmente gerado — que não é exatamente nem ruído branco nem um trem de pulsos — mais o efeito da passagem pelo trato vocal). A resposta do filtro $H(z)$ é suposta constante pelo período em que cada fonema estiver sendo emitido.

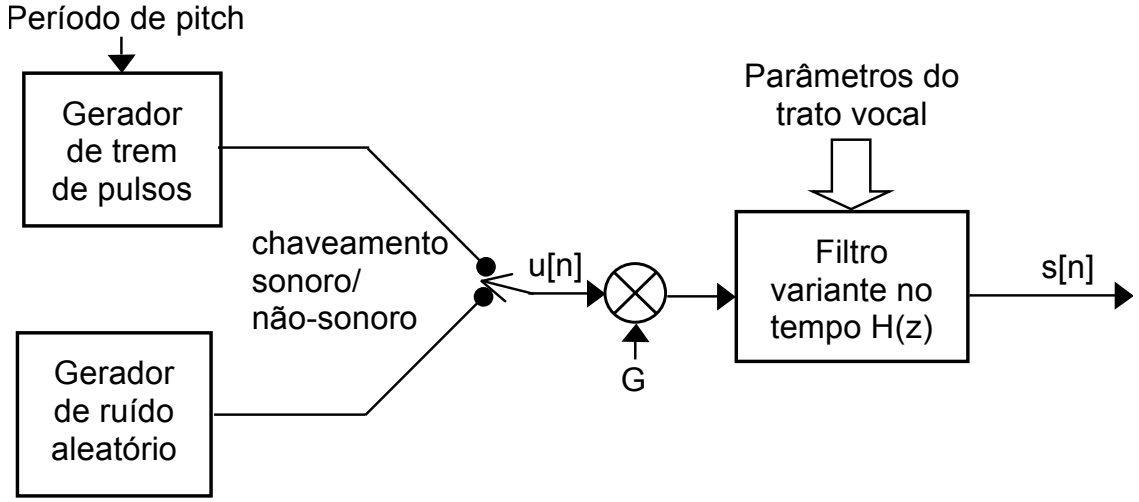

Figura 1: Modelo linear de produção de voz.

Este modelo é razoável, mas como veremos a seguir, não é perfeito. O filtro $H(z)$ é modelado como um filtro IIR com apenas pólos (um modelo conhecido como *autoregressivo*

— AR), ou seja,

$$H(z) = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2} - \cdots - a_p z^{-p}}. \quad (1)$$

Para sinais de voz o valor de p é usualmente limitado a 10.

O codificador que descreveremos a seguir corta o sinal de voz em pequenos trechos (de em torno de 30ms cada), e para cada trecho procura estimar os parâmetros do modelo da figura 1: se o sinal é sonoro ou não, e caso for, o período de *pitch*, o ganho G, e os parâmetros $a_1, \ldots, a_p$ de $H(z)$. Os parâmetros são então quantizados (na verdade, é usada uma estrutura em treliça para o filtro $H(z)$, para reduzir problemas com quantização dos coeficientes) e transmitidos. O decodificador então gera um sinal de excitação (um trem de pulsos com período igual ao período de *pitch* estimado, ou uma sequência pseudo-aleatória), multiplica-o por G, e aplica o filtro $H(z)$, gerando um trecho de sinal de voz.

As janelas utilizadas para estimação do sinal de voz na verdade são escolhidas com sobreposição entre si (no exemplo que faremos a seguir, as janelas têm duração de 30ms e sobreposição de 20ms), e para a reconstrução são utilizadas apenas as primeiras amostras (no caso desta experiência 10ms = 30ms - 20ms).

Como mencionado anteriormente, o codificador descrito acima é apenas um exemplo bem simplificado. Codificadores modernos incorporam diversas melhorias para aumentar a fidelidade do sinal reconstruído mantendo uma baixa taxa de bits.

Vamos discutir rapidamente como estimar os parâmetros do sinal. A determinação do período de *pitch* e da decisão se um trecho de voz é sonoro ou não pode ser feita simultaneamente, e não é um problema simples, devido ao fato do sinal de voz nunca ser exatamente periódico, e do período de *pitch* variar lentamente mesmo para um mesmo trecho de sinal. Alguns algoritmos importantes para estimação de *pitch* são baseados na função de correlação cruzada normalizada (NCCF), como é o caso do algoritmo que será usado nesta experiência, YAAPT [3]. Esta função é usada porque a autocorrelação de uma função periódica é também periódica, com os picos realçados.

Para estimar os parâmetros do filtro $H(z)$ é usada uma técnica chamada análise preditiva linear, que dá o nome ao codificador que estamos descrevendo. Devido ao sucesso do codificador linear preditivo, técnicas baseadas em análise linear preditiva são frequentemente conhecidas por "LPC", mesmo quando não aplicadas em codificação. Análise LPC é usada também em reconhecimento automático de voz e locutor.

2 Estimação de *pitch*

Para a estimação de *pitch* será usada a função `yaapt.m`, escrita pelos autores de [3] e disponível em `http://ws2.binghamton.edu/zahorian/yaapt.htm`. O algoritmo aplica diversas técnicas para calcular valores possíveis para o *pitch* de um trecho curto de voz (por default são usados trechos de 35 ms), e usa uma heurística para escolher o valor mais provável. Uma das estimativas é baseada na função de correlação cruzada normalizada (NCCF), definida

para um sinal $s[n]$ $(0 \leq n \leq N-1)$ como

$$\text{NCCF}[k] = \frac{1}{\sqrt{e_0 e_k}} \sum_{n=0}^{N-K_{\max}} s[n]s[n+k], \tag{2}$$

em que $K_{\min} \leq k \leq K_{\max}$,

$$e_0 = \sum_{n=0}^{N-K_{\max}} s^2[n], \qquad e_k = \sum_{n=k}^{k+N-K_{\max}} s^2[n].$$

Esta função realça periodicidades de sinais (a função de autocorrelação mesmo sem normalização tem máximos repetidos a cada período de um sinal periódico). Veja por exemplo o trecho de 30 ms de sinal de voz (a vogal "a") na figura 2, e a sua NCCF na figura 3. Repare que o máximo ocorre para $k = 70$, enquanto que a taxa de repetição aparente medida na figura 2 é 69. Note também como há um outro pico quase da mesma amplitude. A determinação robusta do *pitch* requer muitos cuidados, que não serão tratados aqui — para mais detalhes, consulte [3, 2].

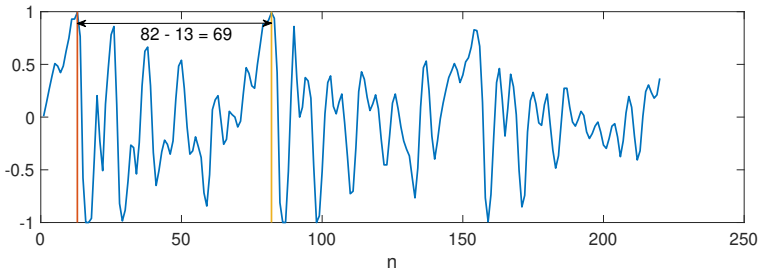

Figura 2: Trecho de sinal de voz (vogal "a").

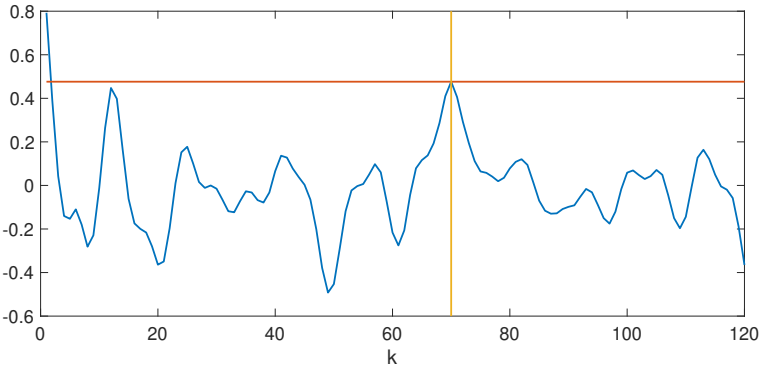

Figura 3: NCCF do trecho de sinal de voz da figura 3.

3 Estimação do ganho e de $H(z)$

Como mencionado acima, usa-se a estrutura de um filtro IIR com apenas pólos (modelo AR) da equação (1) para $H(z)$. Modelos deste tipo, com um valor de p grande o suficiente

podem aproximar bem diversos tipos de sinais — em particular, por experiência notou-se que $p = 10$ é um bom compromisso para sinais de voz. Além disso, esse modelo presta-se particularmente bem à aplicação de codificação de voz, pela facilidade com que os parâmetros a_k podem ser estimados, e também pelas propriedades do erro definido a seguir. Note então que, como indicado pela figura 1, a relação entre as transformadas de $s[n]$ e $Gu[n]$ para um trecho curto de sinal em que $H(z)$ permanece constante é

$$H(z) = \frac{S(z)}{GU(z)} = \frac{1}{1 - \sum_{k=1}^{p} a_k z^{-k}}. \tag{3}$$

Considerando o modelo de produção de voz da figura 1, $s[n]$ está relacionado com $u[n]$ pela seguinte equação de diferenças

$$s[n] = \sum_{k=1}^{p} a_k s[n-k] + Gu[n]. \tag{4}$$

Para entender o princípio de funcionamento do algoritmo que vamos descrever para determinar os parâmetros a_k e G, imagine que estejamos estudando um trecho de $s[n]$ correspondente a um sinal surdo (isto é, em que $u[n]$ é ruído branco). Se os parâmetros de $H(z)$ forem conhecidos perfeitamente, a diferença

$$s[n] - \sum_{k=1}^{p} a_k s[n-k] = Gu[n]$$

é portanto um ruído branco: $u[n]$ é não correlacionado com $u[n-k]$ para qualquer k. Como pela recursão (4) $s[n-k]$ depende linearmente apenas de $u[n-k], u[n-k-1], u[n-k-2], \ldots$, isso significa que $s[n-k]$ é não-correlacionado com $u[n]$, para qualquer $k > 0$. Podemos dizer então que $\sum_{k=1}^{p} a_k s[n-k]$ é a parte de $s[n]$ que depende de valores passados $s[n-k]$ (ou seja, a parte de $s[n]$ que pode ser *predita* a partir dos valores passados), e $Gu[n]$ é a novidade (a *inovação*). Ou seja, $\sum_{k=1}^{p} a_k s[n-k]$ contém toda a informação sobre $s[n]$ disponível em $s[n-k]$, $k \geq 1$.

Daqui segue a estrutura do estimador dos parâmetros a_k: vamos construir um *preditor linear*, que procure aproximar $s[n]$ a partir de $s[n-k]$ para $k = 1, \cdots, p$, minimizando o erro resultante. Suponha que tenhamos escolhido coeficientes de predição α_k, $k = 1, 2, \cdots, p$ para o nosso preditor, e vamos chamar sua saída de $\hat{s}[n]$:

$$\hat{s}[n] = \sum_{k=1}^{p} \alpha_k s[n-k]. \tag{5}$$

A função de transferência desse preditor linear de ordem p é o polinômio

$$P(z) = \sum_{k=1}^{p} \alpha_k z^{-k} = \frac{\hat{S}(z)}{S(z)}. \tag{6}$$

O erro de predição $e[n]$ é dado por

$$e[n] = s[n] - \hat{s}[n] = s[n] - \sum_{k=1}^{p} \alpha_k s[n-k]. \tag{7}$$

Para calcular os valores dos coeficientes α_k, vamos minimizar a energia total do erro em um intervalo de tempo, ou seja, vamos minimizar

$$\xi_{\hat{n}} = \sum_m e_{\hat{n}}^2[m] = \sum_m [s_{\hat{n}}[m] - \hat{s}_{\hat{n}}[m]]^2 = \sum_m \left[s_{\hat{n}}[m] - \sum_{k=1}^{p} \alpha_k s_{\hat{n}}[m-k] \right]^2, \tag{8}$$

em que $s_{\hat{n}}[m]$ é a amostra de um trecho do sinal de voz selecionado na vizinhança de $\hat{n}$, ou seja, $s_{\hat{n}}[m] = s[\hat{n}+m]w[n]$ para $w[m]$ uma janela definida em um intervalo finito em torno de $\hat{n}$. Veremos mais adiante como escolher a janela e os intervalos em que as somatórias devem ser realizadas.

Como se considera que $e[n]$ é a excitação do trato vocal, ele deve ser um trem de pulsos quase-periódico para voz sonora ou um ruído aleatório para voz não sonora. Dessa forma, o sinal $s[n]$ é obtido, filtrando o sinal de erro pelo filtro só-polo

$$H(z) = \frac{1}{A(z)} = \frac{1}{1 - P(z)} = \frac{1}{1 - \sum_{k=1}^{p} \alpha_k z^{-k}}. \tag{9}$$

O sistema cuja função de transferência é $H(z)$ é chamado de sistema de modelo do trato vocal ou sistema LPC.

Nosso objetivo é determinar o conjunto de coeficientes de predição $\{\alpha_k,\ k = 1,2,\cdots,p\}$ a partir do sinal de voz para obter uma boa estimativa das propriedades espectrais variantes no tempo do sinal de voz. Devido às propriedades serem variantes no tempo, os coeficientes α_k devem ser estimados através de análises de curto prazo a fim de encontrar o conjunto dos coeficientes que minimizam o erro de predição ao quadrado em um trecho curto do sinal. Os parâmetros resultantes são assumidos como os da função de transferência do trato vocal $H(z)$ no modelo da Figura 1.

Igualando a zero a derivada de $\xi_{\hat{n}}$ em relação a α_i para $i = 1,2,\cdots,p$, obtemos

$$\sum_m s_{\hat{n}}[m-i]s_{\hat{n}}[m] = \sum_{k=1}^{p} \alpha_k \sum_m s_{\hat{n}}[m-i]s_{\hat{n}}[m-k], \quad 1 \le i \le p. \tag{10}$$

Definindo

$$\varphi_{\hat{n}}(i,k) \triangleq \sum_m s_{\hat{n}}[m-i]s_{\hat{n}}[m-k], \tag{11}$$

(10) pode ser reescrita como

$$\sum_{k=1}^{p} \alpha_k \varphi_{\hat{n}}(i,k) = \varphi(i,0), \quad i = 1,2,\cdots,p. \tag{12}$$

Esse conjunto de p equações e p incógnitas pode ser resolvido de forma eficiente para encontrar os coeficientes $\{\alpha_k\}$ que minimizam $\xi_{\hat{n}}$.

O valor mínimo de $\xi_{\hat{n}}$ é dado por

$$\xi_{\hat{n},\,\min} = \sum_m s_{\hat{n}}^2[m] - \sum_{k=1}^{p} \alpha_k \sum_m s_{\hat{n}}[m]s_{\hat{n}}[m-k], \tag{13}$$

que pode ser reescrito usando (11) como

$$\xi_{\hat{n},\,\min} = \varphi_{\hat{n}}(0,0) - \sum_{k=1}^{p} \alpha_k \varphi_{\hat{n}}(0,k). \tag{14}$$

O erro mínimo $\xi_{\hat{n},\,\min}$ é composto de um componente fixo $\varphi_{\hat{n}}(0,0)$ que corresponde à energia de $s_{\hat{n}}[m]$ e um componente que depende dos coeficientes de predição.

Para conseguir obter os coeficientes de predição ótimos, precisamos primeiramente calcular as quantidades $\varphi(i,k)$ para $1 \leq i \leq p$ e $0 \leq k \leq p$. Assim que fizermos isso, temos somente que resolver (12) para obter os coeficientes α_k. Assim, em princípio, a análise preditiva linear é bem direta.

Até agora, não especificamos os limites dos somatórios em (8) e em (10), que devem estar em um intervalo finito em torno de $\hat{n}$. Há duas abordagens básicas para esse problema e veremos que dois métodos de predição surgem de uma consideração dos limites dos somatórios e da definição da janela $w[m]$.

3.1 O método da autocorrelação

Uma abordagem para determinar os limites dos somatórios em (8) e em (10) é assumir que o segmento $s_{\hat{n}}[m]$ é identicamente nulo fora do intervalo $0 \leq m \leq L-1$, o que pode ser expresso escolhendo uma janela $w[m]$ de comprimento finito (por exemplo, retangular ou Hamming) que é nula fora de um intervalo $0 \leq m \leq L-1$, ou seja,

$$s_{\hat{n}}[m] = s[\hat{n}+m]w[m], \quad 0 \leq m \leq L-1. \tag{15}$$

Na Figura 4, é mostrado um sinal $s[n]$ em função do índice de tempo n em conjunto com uma janela de Hamming $w[n-\hat{n}]$ posicionada para começar em $n = \hat{n}$. Redefinindo a origem do segmento para começar em $\hat{n}$ pela substituição de $m = n - \hat{n}$, obtemos o segmento janelado $s_{\hat{n}}[m] = s(m+\hat{n})w[m]$ em função de m. Ainda na Figura 4, é mostrada a saída de um preditor linear ótimo com ordem p. Podemos observar dessa figura que $s_{\hat{n}}[m]$ é não nulo só para $0 \leq m \leq L-1$, mas $e_{\hat{n}}[m]$ é não nulo só para $0 \leq m \leq L-1+p$. Dessa forma, $\xi_{\hat{n}}$ pode ser expresso como

$$\xi_{\hat{n}} = \sum_{m=0}^{L-1+p} e_{\hat{n}}^2[m] = \sum_{m=\infty}^{\infty} e_{\hat{n}}^2[m], \tag{16}$$

devido às definições de $s_{\hat{n}}[m]$ e $e_{\hat{n}}[m]$.

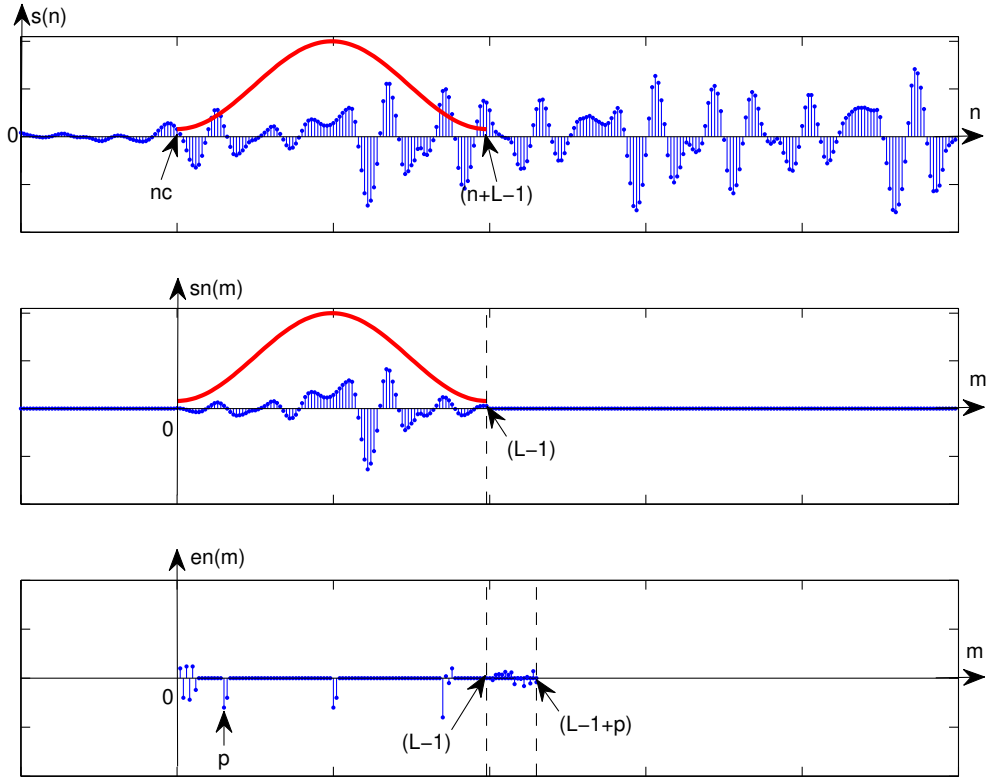

Figura 4: Ilustração da análise de curto prazo para o método da autocorrelação.

Da Equação (7) e da Figura 4 é possível verificar que o erro de predição

$$e_{\hat{n}}[m] = s_{\hat{n}}[m] - \sum_{k=1}^{p} \alpha_k s[m-k]$$

é relativamente grande no início do intervalo $0 \leq m \leq p-1$ porque o preditor deve predizer as amostras não nulas a partir de amostras anteriores nulas devido ao janelamento por uma janela de duração finita. Algo semelhante acontece no final do intervalo ($L \leq m \leq L-1+p$) porque o preditor deve predizer p amostras nulas fora da janela a partir de amostras não nulas dentro da janela. Por essa razão, uma janela como a Hamming que diminui o valor da amostras de $s_{\hat{n}}[m]$ no início e no final do segmento é preferida para mitigar esses efeitos. Repare que o preditor ótimo construído desta maneira nunca irá zerar completamente o valor da energia total do erro $\xi_{\hat{n}}$, já que a primeira e a última amostra do erro serão sempre necessariamente não nulas se $s_{\hat{n}}[0] \neq 0$ e $s_{\hat{n}}(L-1) \neq 0$.

Os limites para $\varphi_{\hat{n}}(i,k)$ em (11) são idênticos aos da Equação (16), assim

$$\varphi_{\hat{n}}(i,k) = \sum_{m=0}^{L-1+p} s_{\hat{n}}[m-i]s_{\hat{n}}[m-k] \tag{17}$$

para $1 \leq i \leq p$ e $0 \leq k \leq p$. Lembrando que $s_{\hat{n}}[m-i] = 0$ para $m-i < 0$ ou $m-i > L-1$, e similarmente $s_{\hat{n}}[m-k] = 0$ para $m-k < 0$ ou $m-k > L-1$, e considerando os limites para i e k, concluímos que $s_{\hat{n}}[m-i]s_{\hat{n}}[m-k] \neq 0$ apenas para $m \geq 0$, $m \leq L+p-1$. Fazendo a

troca de variáveis $\ell = m - i$, vem $m = \ell + i$, e levando em conta os intervalos para os quais $s_{\hat{n}}[m]$ é não nula, conclui-se que

$$\varphi_{\hat{n}}(i,k) = \sum_{\ell=0}^{L-1-(i-k)} s_{\hat{n}}[\ell] s_{\hat{n}}[\ell + i - k]. \tag{18}$$

para $1 \leq i \leq p$ e $0 \leq k \leq p$. A função $\varphi_{\hat{n}}(i,k)$ tem a forma de uma função de autocorrelação de curto prazo calculada em $i - k$, e portanto herda todas as propriedades de funções de autocorrelação. Em particular, $\varphi_{\hat{n}}(i,k) = \varphi_{\hat{n}}(k,i)$, $\varphi_{\hat{n}}(i,k) = \varphi_{\hat{n}}(i-k,0)$, e $|\varphi_{\hat{n}}(i,k)| \leq |\varphi_{\hat{n}}(i,i)|$.

Escrevendo, para simplificar a notação,

$$\varphi_{\hat{n}}(i,k) = R_{\hat{n}}[i-k], \quad \text{para } 1 \leq i \leq p \quad \text{e} \quad 0 \leq k \leq p. \tag{19}$$

Como $R_{\hat{n}}$ é uma função par, então

$$\varphi(i,k) = R_{\hat{n}}(|i-k|) \quad \text{para } 1 \leq i \leq p \quad \text{e} \quad 0 \leq k \leq p,$$

e (12) e (14) podem ser expressas respectivamente como

$$\sum_{k=1}^{p} \alpha_k R_{\hat{n}}(|i-k|) = R_{\hat{n}}(i), \quad 1 \leq i \leq p \tag{20}$$

e

$$\xi_{\hat{n},\,\min} = R_{\hat{n}}[0] - \sum_{k=1}^{p} \alpha_k R_{\hat{n}}[k]. \tag{21}$$

A Equação (20) ainda pode ser escrita de forma matricial na forma conhecida como *equações de Yule-Walker*, como

$$\begin{bmatrix} R_{\hat{n}}[0] & R_{\hat{n}}[1] & \cdots & R_{\hat{n}}[p-1] \\ R_{\hat{n}}[1] & R_{\hat{n}}[0] & \cdots & R_{\hat{n}}[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ R_{\hat{n}}[p-1] & R_{\hat{n}}[p-2] & \cdots & R_{\hat{n}}[0] \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_p \end{bmatrix} = \begin{bmatrix} R_{\hat{n}}[1] \\ R_{\hat{n}}[2] \\ \vdots \\ R_{\hat{n}}[p] \end{bmatrix} \tag{22}$$

Essa matriz de autocorrelação $p \times p$ é uma matriz simétrica e de Toeplitz (com os elementos constantes ao longo das diagonais). Pode-se mostrar que a matriz também é positiva-definida (e portanto não singular) para sinais de voz. Equações de Yule-Walker podem ser resolvidas de maneira muito eficiente (aproveitando a estrutura das equações) usando o algoritmo de Levinson-Durbin [2].

3.2 O método da covariância

A segunda abordagem para definir o segmento de voz $s_{\hat{n}}[m]$ e os limites dos somatórios é fixar o intervalo sobre o qual o erro quadrático médio é calculado e então considerar o efeito no cálculo de $\varphi_{\hat{n}}(i,k)$. Neste caso, define-se

$$\xi_{\hat{n}} = \sum_{m=0}^{L-1} e_{\hat{n}}^2[m] = \sum_{m=0}^{L-1} \left(s_{\hat{n}}[m] - \sum_{k=1}^{p} \alpha_k s_{\hat{n}}[m-k] \right)^2, \tag{23}$$

mas agora para evitar efeitos de borda no cálculo de $e_{\hat{n}}[m]$, vamos estender a janela para a esquerda, admitindo valores de $s_{\hat{n}}[m]$ para $-p \leq m \leq L-1$. Por esse motivo, $\varphi_{\hat{n}}(i,k)$ se torna

$$\varphi_{\hat{n}}(i,k) = \phi(i,k) = \sum_{m=0}^{L-1} s_{\hat{n}}[m-i]s_{\hat{n}}[m-k], \quad \text{para } 1 \leq i \leq p \quad \text{e} \quad 0 \leq k \leq p, \tag{24}$$

com $s_{\hat{n}}[m] = s[\hat{n}+m]$ para $-p \leq m \leq L-1$. Neste caso, se alterarmos o índice do somatório, podemos expressar $\phi(i,k)$ como

$$\phi(i,k) = \sum_{m=-i}^{L-i-1} s_{\hat{n}}[m]s_{\hat{n}}(m+i-k) \quad \text{para } 1 \leq i \leq p \quad \text{e} \quad 0 \leq k \leq p, \tag{25}$$

ou

$$\phi(i,k) = \sum_{m=-k}^{L-k-1} s_{\hat{n}}[m]s_{\hat{n}}(m+k-i) \quad \text{para } 1 \leq i \leq p \quad \text{e} \quad 0 \leq k \leq p, \tag{26}$$

Embora essas equações sejam similares à equação (18), os limites dos somatórios não são os mesmos, já que as equações (25) e (26) usam $s_{\hat{n}}[m]$ fora do intervalo $0 \leq m \leq L-1$.

Na Figura 5, é mostrada a forma de onda de um sinal de voz $s[n]$ e uma janela retangular de comprimento $p+L$ posicionada para começar em $\hat{n}-p$ na escala de tempo n. Também é mostrado o segmento do sinal de voz selecionado pela janela depois da redefinição da origem do segmento, usando a substituição $m = n - \hat{n}$. Note que a janela se estende de $m = -p$ a $m = L-1$. As p amostras no intervalo $-p \leq m \leq -1$ são requeridas para usar apropriadamente o filtro de erro de predição de ordem p, i.e,

$$e_{\hat{n}}[m] = s_{\hat{n}}[m] - \sum_{k=1}^{p} \alpha_k s[m-k]$$

para calcular $e_{\hat{n}}[m]$ para $0 \leq m \leq L-1$. O erro de predição também é mostrado nessa figura. É possível observar que ele não apresenta mais o comportamento transitório no início e no fim do intervalo como acontece com o método da autocorrelação. Neste caso, se o sinal de voz satisfizer exatamente o modelo, a energia mínima será nula, diferentemente do método da correlação.

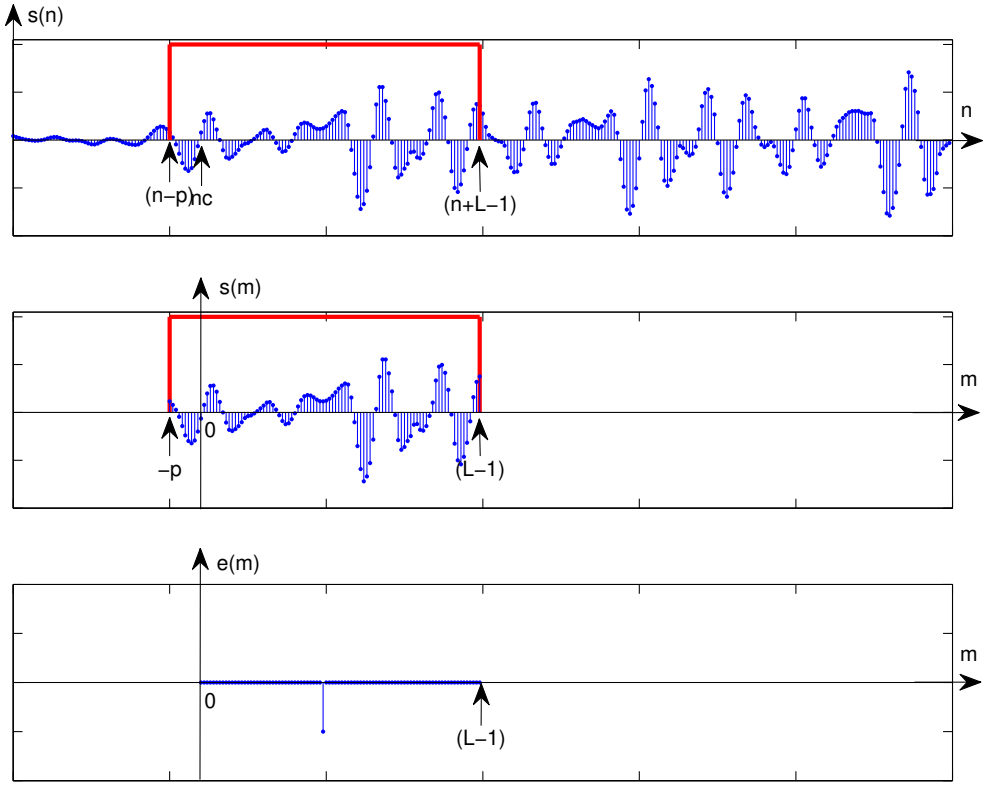

Figura 5: Ilustração da análise de curto prazo para o método da covariância.

Embora as diferenças entre os métodos (25), (26) e (16) pareçam estar em detalhes computacionais, o conjunto de equações

$$\sum_{k=1}^{p} \alpha_k \phi_{\hat{n}}(i,k) = \phi_{\hat{n}}(i,0), \quad i = 1,2,\cdots,p \tag{27}$$

tem propriedades significativamente diferentes que afetam a solução e as propriedades do preditor ótimo resultante. Na forma matricial, essas equações ficam

$$\begin{bmatrix} \phi_{\hat{n}}(1,1) & \phi_{\hat{n}}(1,2) & \cdots & \phi_{\hat{n}}(1,p) \\ \phi_{\hat{n}}(2,1) & \phi_{\hat{n}}(2,2) & \cdots & \phi_{\hat{n}}(2,p) \\ \cdots & \cdots & \cdots & \cdots \\ \phi_{\hat{n}}(p,1) & \phi_{\hat{n}}(p,2) & \cdots & \phi_{\hat{n}}(p,p) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \cdots \\ \alpha_p \end{bmatrix} = \begin{bmatrix} \phi_{\hat{n}}(1,0) \\ \phi_{\hat{n}}(2,0) \\ \cdots \\ \phi_{\hat{n}}(p,0) \end{bmatrix}. \tag{28}$$

Como $\phi_{\hat{n}}(i,k) = \phi_{\hat{n}}(k,i)$ a matriz do tipo correlação é simétrica e positiva definida, mas não é Toeplitz. Os elementos da diagonal estão relacionados pela equação

$$\phi_{\hat{n}}(i+1,k+1) = \phi_{\hat{n}}(i,k) + s_{\hat{n}}(-i-1)s_{\hat{n}}(-k-1) - s_{\hat{n}}(L-1-i)s_{\hat{n}}(L-1-k) \tag{29}$$

e o erro quadrático mínimo satisfaz

$$\xi_{\hat{n},\,\min} = \phi_{\hat{n}}(0,0) - \sum_{k=1}^{p} \alpha_k \phi_{\hat{n}}(0,k). \tag{30}$$

Esse método é conhecido como método da covariância pois a matriz tem propriedades de uma matriz de covariância, mesmo não sendo uma matriz de covariância de variáveis aleatórias.

3.3 Cálculo do ganho do modelo

Como vimos, baseando-se na hipótese de que os coeficientes do preditor calculados a partir do sinal de voz são idênticos aos coeficientes do modelo, i.e, $\alpha_k = a_k$ e que a saída do modelo é precisamente $s[n]$, obtemos

$$e[n] = Gu[n]. \tag{31}$$

Isso é, o sinal de entrada do modelo é proporcional ao sinal de erro com uma constante de proporcionalidade G.

Para calcular G de maneira mais precisa, em vez de tentar resolver (31) para um valor qualquer de n, é melhor considerar o intervalo todo de cálculo, e definir G como a raiz quadrada da energia média do sinal de erro sobre a potência média da excitação. Por exemplo, considerando o método da autocorrelação, obtemos

$$G^2 \frac{1}{L-1+p} \sum_{m=0}^{L-1+p} u^2[m] = \frac{1}{L-1+p} \sum_{m=0}^{L-1+p} e^2[m] = \frac{1}{L+p-1} \xi_{\hat{n}}. \tag{32}$$

Considerando que o período de pitch seja L_p, para sinais sonoros temos a excitação $u[n] = \delta[n]$ repetida a cada L_p amostras. Para sinais não-sonoros, assume-se que $u[n]$ é um ruído branco de média zero e variância unitária.

Neste caso, a potência média do sinal $u[n]$ para o caso sonoro é $1/L_\mathrm{p}$, enquanto que no caso surdo, a potência média será 1 (igual à variância do ruído). Portanto, o valor de G deve ser

$$G = \sqrt{L_\mathrm{p} \xi_{\hat{n}}}, \qquad \text{se o sinal for sonoro,} \tag{33}$$

$$G = \sqrt{\xi_{\hat{n}}}, \qquad \text{se o sinal for surdo.} \tag{34}$$

Os valores de $\xi_{\hat{n}}$ podem ser calculados por (21) para o método da correlação, ou por (30) para o método da covariância.

4 Parte experimental

Vamos testar o que foi visto aqui em um exemplo. O arquivo `Antarctica.wav` contém um trecho de sinal de voz amostrado a $fs = 8$kHz. Ele pode ser lido no Matlab com o comando

```
[sinal, fs]=audioread('Antarctica.wav');
```

e em Julia com o comando

```
using WAV # Basta dar este comando uma vez no início do trabalho
sinal, fs = wavread("Antarctica.wav")
```

Faça as seguintes tarefas:

1. Corte um trecho do sinal de voz entre $n = 200$ e $n = 439$ (correspondente a uma janela de $L = 240$ amostras, ou 30ms), e calcule os parâmetros de um modelo LPC com 10 coeficientes usando o comando (em Matlab)

```
[ak, sig2]=lpc(trecho.*hamming(240), 10);
```

ou em Julia

```
using DSP
ak, sig2 = lpc(trecho.*hamming(240), 10; padmode = :longest)
```

Repare que o comando `lpc` usa o método da correlação, portanto é conveniente usar uma janela de Hamming para os cálculos. No caso de usar Julia, note que o vetor `ak` tem elementos $a_1, a_2, \ldots, a_p$, enquanto que em Matlab, `ak` tem elementos $1, a_1, a_2, \ldots, a_p$.

2. Calcule a resposta em frequência do filtro $G.H(z)$ correspondente, e desenhe um gráfico da resposta com escala em decibéis. Use 512 pontos para a resposta em frequência.

3. Calcule o periodograma (uma estimativa da transformada) do trecho de sinal de voz com o comando (no Matlab):

   ```
   hold on
   periodogram(trecho,[],512);
   ```

   Em Julia a função `periodogram` normaliza a potência da saída, então é necessário aplicar uma correção:

   ```
   per = periodogram(trecho; fs = fs, nfft = 512)
   plot(per.freq, pow2db.(per.power * fs / pi)
   ```

   Repare que a resposta em frequência do modelo LPC acompanha a envoltória da transformada do sinal. Os picos observados na envoltória são denominados *formantes*, e são importantes para o reconhecimento do fonema.

4. Experimente aumentar a ordem do modelo LPC para 20, 40 ou 80, e veja que o modelo começa a procurar reproduzir também os detalhes do periodograma. No caso de codificação LPC, isso não é desejado, e limita-se p a um valor pequeno (no nosso caso, 10).

5. O arquivo `yaapt.zip` tem código do programa `yaapt.m`, um estimador de *pitch* desenvolvido pelos autores de [3]. Veja o resultado do estimador rodando a função

   ```
   pitch=yaapt(sinal,fs,1,[],1,1);
   ```

   Detalhes sobre o funcionamento do programa podem ser encontrados na referência acima (que está também no arquivo `yaapt.zip`), mas para esta experiência basta que você veja o gráfico da figura 4 gerada pelo Matlab, que contém o espectrograma do sinal de voz (a STFT) desenhada num gráfico sobreposto aos valores estimados de *pitch*, e ao detector de sinal sonoro/surdo.

   Os valores T_p de período *pitch* são estimados por default para janelas de 35ms, a cada 10ms. Para simplificar a experiência, vamos usar esses valores default. Os valores são deslocados, então use

```
pitch=[0 pitch];
```

para que o primeiro valor de *pitch* corresponda ao primeiro trecho de 240 amostras do sinal.

Se você estiver usando Julia, a função `pitch` do arquivo `pitch.jl` calcula uma estimativa do *pitch* para um trecho de voz. Ao contrário do programa `yaapt`, a função `pitch` recebe como entrada o trecho do sinal de voz para o qual se quer estimar o *pitch*, então você precisa fornecer um trecho de cada vez. O resultado fica melhor se você usar trechos de 30ms para as estimativas, com sobreposição (saltando 10ms entre cada estimativa). A saída da função `pitch` é zero se o trecho for surdo.

6. Finalmente, para testar o codificador, escreva um programa que:

   (a) Pegue um trecho de sinal de comprimento de 30ms (240 amostras). Para cada trecho, calcule os parâmetros do modelo LPC como acima.

   (b) Use a sua estimativa de *pitch* para decidir se o trecho é sonoro ou não (é sonoro se $T_p[i] > 0$).

   (c) Vamos gerar trechos de sinal de 80 amostras cada um para o sinal reconstruído. Caso o sinal seja sonoro, gere uma sequência de pulsos distanciados do período de *pitch*. Escolha o sinal de excitação com 80 amostras. Caso o trecho não seja sonoro, gere um trecho de ruído de 80 amostras com o comando `randn`.

   (d) O ganho é calculado usando a segunda saída da função `lpc` (no exemplo acima, a variável `sig2`) e as expressões (33) ou (34), conforme o caso.

   (e) Filtre o sinal de excitação por um filtro $H(z)$ com numerador igual a G e coeficientes do denominador dados pelos α_k calculados pela função `lpc`.

   No Matlab, o filtro é obtido usando a função `filter`. Se `aq` for o vetor de coeficientes quantizados, e `exc` for o sinal de excitação, então

   ```
   trechosint = filter(G, aq, exc)
   ```

   Em Julia, o filtro usa a função `filt` (lembre que a saída do comando `lpc` em Julia não tem o 1 inicial, é necessário acrescentar esse termo no vetor `aq`):

   ```
   trechosint = filt(G, aq, exc)
   ```

   (f) O sinal de saída pode ser construído acrescentando-se as primeiras 80 amostras do sinal filtrado no item anterior à saída calculada até então.

   (g) Modifique o programa anterior para quantizar os coeficientes. No Matlab, use a função `quantize3`, e quantize os coeficientes LPC com $B_a = 7$ bits cada um, e o ganho e o período de *pitch* com $B_g = 5$ bits cada um. Calcule a taxa de bits por segundo conseguida com o seu codificador nessas condições.

   Em Julia, você pode quantizar uma variável `G` ou um vetor `a` usando o pacote `FixedPointNumbers`:

```
using FixedPointNumbers
Gq = Fixed{Int16, Bg - 1}(G)
aq = Fixed{Int16, Ba - 1}.(a)
```

É possível que alguns valores de Gq acabem ficando iguais a zero devido à quantização. Neste caso, você pode ter um erro na função `filt` nos casos em que $Gq = 0$. Se isto ocorrer, antes de aplicar o filtro confira se o ganho quantizado é nulo, e se for, defina a saída do decodificador daquele trecho como igual a zero.

O codificador obtido como descrito acima é bastante imperfeito. Diversos cuidados podem ser tomados para melhorar o seu desempenho, como, dentre muitos outros,

- Cuidar melhor da transição entre um trecho de sinal reconstruído e outro,
- Implementar os filtros de uma maneira mais eficiente, e não quantizar diretamente os coeficientes do filtro de predição, mas uma transformação (que pode ser não-linear) dos coeficientes.

5 Codificação com realimentação (análise-por-síntese)

A ideia básica dos métodos de codificação por análise-por-síntese é procurar usar um sinal de excitação mais adequado para reconstruir o sinal de voz. De fato, se fosse usado exatamente o erro obtido do filtro de predição (equação (7)), o sinal reconstituído seria igual ao sinal original. Nos codificadores por análise-por-síntese, escolhe-se previamente uma família de funções-base $\mathcal{F} = \{f_1[n], f_2[n], \ldots, f_Q[n]\}$, e para cada trecho de sinal de voz, são calculados K coeficientes β_q tais que o sinal de excitação (doravante chamado $d[n]$, veja a figura 6) seja

$$d[n] = \sum_{q=1}^{K} \beta_k f_k[n]. \tag{35}$$

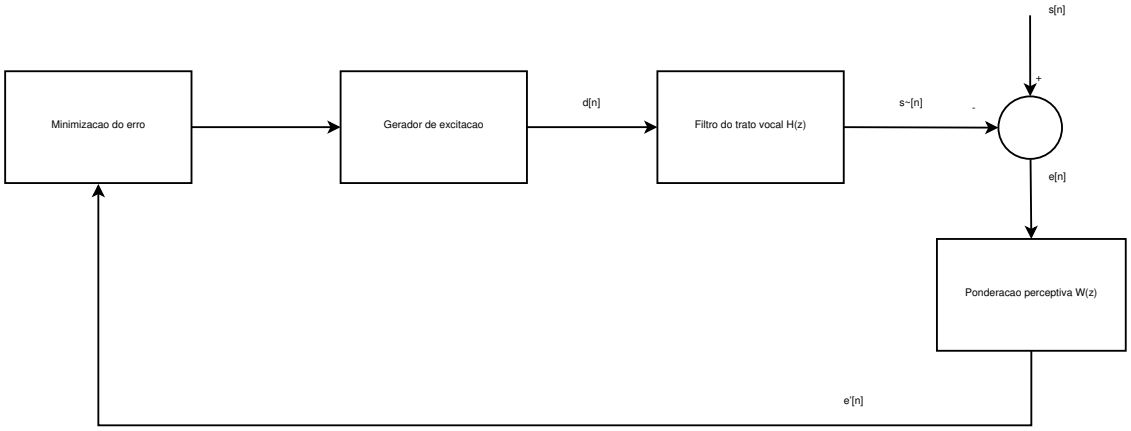

Figura 6: Codificação linear preditiva usando análise-por-síntese (codificador)

O procedimento para a codificação é o seguinte, de acordo com [2]:

1. Cortar um trecho $s[n]$ de sinal de voz, e estimar seus parâmetros LPC α_k.

2. Gerar uma estimativa inicial para o sinal de excitação $d[n]$.

3. Repetir, até que seja atingido um critério de parada (número de iterações ou valor do erro de reconstrução):

   (a) A partir da estimativa atual do sinal de excitação $d[n]$, e dos parâmetros LPC α_k, calcular o sinal de voz reconstruído $\hat{s}[n]$ e o erro $e[n] = s[n] - \hat{s}[n]$.

   (b) Aplicar ao erro um filtro de ponderação perceptiva $W(z)$, obtendo o erro ponderado $e'[n]$. O objetivo do filtro de ponderação é dar ênfase também às faixas do espectro de menor potência, para evitar que o critério de qualidade seja baseado somente nas faixas de frequência com maior energia. O filtro de ponderação é normalmente escolhido como

   $$W(z) = \frac{1 - P(z)}{1 - P(\gamma^{-1} z)} = \frac{1 - \sum_{k=1}^{p} \alpha_k z^{-k}}{1 - \sum_{k=1}^{p} \alpha_k \gamma^k z^{-k}}, \tag{36}$$

   em que $P(z) = \sum_{k=1}^{p} \alpha_k z^{-k}$ é o preditor LPC (veja a equação (9)). O parâmetro $0 \leq \gamma \leq 1$ controla a forma do filtro. Para $\gamma = 0$, $W(z)$ é igual à inversa de $H(z)$, enquanto que para $\gamma = 1$, $W(z) \equiv 1$ (ver figura 7).

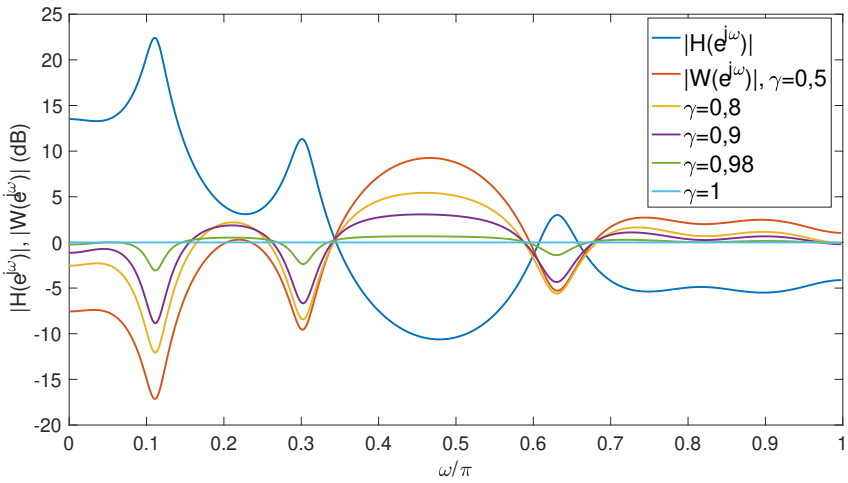

Figura 7: Filtro de ponderação perceptiva $W(z)$ para diversos valores de γ.

   (c) Escolher os parâmetros para o gerador de excitação (no caso do modelo da equação (35), os valores dos β_q) de modo a reduzir o erro ponderado $e'[n]$ por algum critério.

4. Os parâmetros LPC e os parâmetros do sinal de excitação são então quantizados e armazenados para representar uma parte do trecho de sinal de voz sendo tratado.

O decodificador usa os parâmetros LPC e os parâmetros do sinal de excitação para construir uma aproximação do sinal de voz, como indicado na figura 8.

Vamos descrever a seguir uma maneira de calcular os parâmetros de excitação, e uma opção para a família de funções-base, que dá origem ao codificador MPLPC (*multipulse LPC*).

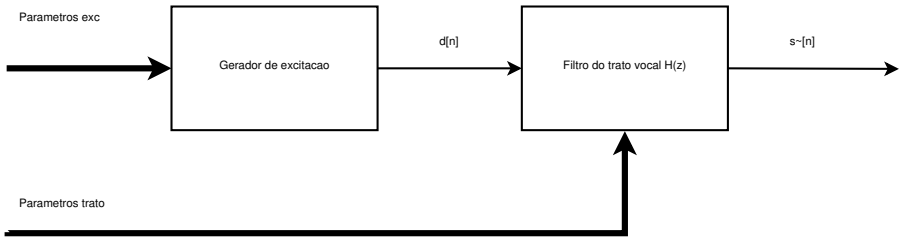

Figura 8: Decodificador para o codificador A-b-S.

5.1 Otimização dos parâmetros da excitação

Suponha que o sinal de excitação seja conhecido até o final do último quadro do sinal de voz, e vamos supor por simplicidade que o quadro atual começa em $n = 0$. Ou seja, a seguir estamos supondo que $d[n]$ é conhecido para $n < 0$. Nesse caso, para $n \geq 0$ precisamos levar em conta a resposta do filtro do trato vocal para o sinal de excitação calculado anteriormente, ou seja, definindo

$$d_0[n] = \begin{cases} d[n], & n < 0, \\ 0, & n \geq 0, \end{cases}$$

temos o sinal residual (o transitório final) do quadro anterior igual a $y_0[n] = \hat{s}[n] = d_0[n] * h[n]$. Levando este transitório em consideração, o sinal de erro ponderado inicial é

$$e'_0[n] = (s[n] - d_0[n] * h[n]) * w[n] = s[n] * w[n] - d_0[n] * (h[n] * w[n]) \stackrel{\Delta}{=} s'[n] - d_0[n] * h'[n], \quad (37)$$

em que $*$ representa convolução, $h[n]$ e $w[n]$ são as respostas ao impulso das funções $H(z)$ e $W(z)$, respectivamente, e definimos o sinal de voz ponderado $s'[n] = s[n] * w[n]$, e o filtro equivalente total $h'[n] = h[n] * w[n]$.

Queremos agora calcular o sinal de excitação $d[n]$ para o quadro atual. Podemos definir como função-custo para minimização para o primeiro passo do processo de análise-por-síntese o erro ponderado médio quadrático

$$\varepsilon_1 = \sum_{n=0}^{L-1} \left(e_0[n] - \beta_q f_q[n] * h'[n]\right)^2, \quad (38)$$

que devemos minimizar:

$$(q_1, \beta_{q_1}^{\text{o}}) = \arg\min_{q,\beta_q} \sum_{n=0}^{L-1} \left(e_0[n] - \beta_q f_q[n] * h'[n]\right)^2. \quad (39)$$

Para um determinado valor de q, é fácil obter o valor de β_q correspondente, diferenciando (38), resultando

$$\beta_q^{\text{o}} = \frac{\sum_{n=0}^{L-1} e'_0[n] \cdot (f_q[n] * h'[n])}{\sum_{n=0}^{L-1} (f_q[n] * h'[n])^2}. \quad (40)$$

O erro resultante é

$$\varepsilon^{(q)\text{o}} = \sum_{n=0}^{L-1} \left(e'_0[n]\right)^2 - \left(\beta_q^{\text{o}}\right)^2 \sum_{n=0}^{L-1} (f_q[n] * h'[n])^2. \quad (41)$$

A seguir é feita uma busca entre os valores de q para encontrar aquele que minimiza $\varepsilon^{(q)o}$.

O procedimento acima é repetido, mantendo-se os termos já encontrados e acrescentando-se a cada iteração um termo $\beta_q f_q[n]$ a mais à aproximação para $e'_0[n]$, até o limite pré-especificado de Q termos. Uma vez encontrados todos os termos $q_1, q_2, \ldots, q_Q$, é feita uma re-estimação dos parâmetros β_{q_i}, já que em geral os sinais da base não são necessariamente ortogonais entre si. Como, dadas as funções-base escolhidas, o problema é quadrático, a solução se reduz a resolver um sistema de equações lineares $\boldsymbol{A}\boldsymbol{\beta} = \boldsymbol{b}$, em que $\boldsymbol{\beta} = \begin{bmatrix}\beta_{q_1} & \beta_{q_2} & \ldots & \beta_{q_Q}\end{bmatrix}^T$, e, definindo $y'_q[n] = f_q[n] * h'[n]$, os elementos de $\boldsymbol{A}$ e de $\boldsymbol{b}$ são

$$[\boldsymbol{A}]_{ij} = \sum_{n=0}^{L-1} y'_{q_i}[n] y'_{q_j}[n], \qquad [\boldsymbol{b}]_i = \sum_{n=0}^{L-1} e'_0[n] y'_{q_i}[n]. \tag{42}$$

Uma opção para as funções-base é um conjunto de pulsos com atrasos arbitrários, o que dá origem ao codificador MPLPC . No codificador MPLPC, as funções-base são simplesmente

$$f_q[n] = \delta[n - q],$$

$0 \leq q \leq L - 1$.

Outra opção é usar diversas realizações de processos Gaussianos e brancos, o que leva ao codificador CELP (*code excited linear prediction*), como será feito na parte experimental. Note que com este tipo de codificador, não é necessário estimar-se o período de *pitch*, nem o valor de G.

Mais detalhes e diversas modificações sobre os algoritmos básicos que levam a melhor desempenho podem ser consultados em [2, 1].

6 Parte experimental

Vamos implementar um codificador CELP simples seguindo os passos a seguir. Para simplificar, vamos usar $\gamma = 1$ e desconsiderar o filtro de ponderação. Considere novamente o mesmo sinal de antes, amostrado a 8 kHz.

1. Defina quadros de $L = 240$ amostras (equivalente a 30 ms). Vamos gerar aproximações para trechos de $N = 80$ amostras (10 ms) de cada vez.

2. Defina uma base com $Q = 512$ funções aleatórias, cada uma de comprimento N, usando o comando `randn`. Vamos usar combinações de $K = 2$ funções-base para as aproximações.

3. Defina uma variável `zs` para armazenar a condição inicial do filtro de trato vocal do decodificador, inicializada com zeros.

4. Apesar de calcularmos os parâmetros LPC para quadros de comprimento L, vamos calcular parâmetros de excitação apenas para trechos de comprimento N. Repita para quadros de comprimento L, deslocados de N amostras cada um:

(a) Determine os coeficientes LPC do quadro atual, usando uma janela de Hamming e $p = 10$.

(b) Defina um sub-quadro de comprimento N usando as amostras centrais do quadro atual.

(c) Filtre todas as $Q = 512$ sequências da base pelo filtro $H(z) = 1/(1 - P(z))$ definido pelos parâmetros LPC.

(d) Para melhorar a transição entre quadros, calcule o transitório final da excitação anterior (isto é, $d_0[n] * h[n]$), passando um sinal com N zeros pelo filtro $H(z)$. Se o vetor `aq` for a saída quantirada da função `lpc`, em Matlab, use `[y0, zs] = filter(1, aq, zeros(N,1), zs);`. A condição inicial deve ser inicializada com p zeros no início.

No caso de Julia, use a função `fxfilt`, disponível no arquivo `fxfilt.jl`:

```
y0 = fxfilt(1, [1;aq], zeros(N), zs)
```

(e) Calcule o sinal $e_0[n]$ como a diferença entre $s[n]$ e a saída do item anterior, isto é, `e0 = subquadro - y0`.

(f) Use a função `find_Nbest.m` (escrita pelos autores de [1]) ou a versão portada para Julia `find_Nbest.jl` para calcular a melhor combinação de funções da base:

```
[ganhos, indices] = find_Nbest_components(e0, fnc_base_filt, K);
```

ou em Julia,

```
ganhos, indices = find_Nbest_components(e0, fnc_base_filt, K);
```

(g) Defina o sinal de excitação usando as funções-base e os ganhos calculados no item anterior.

(h) Neste ponto, o codificador calculou todos os seus parâmetros. As variáveis que seriam armazenadas seriam os valores dos coeficientes α_k, e os índices q_i e ganhos β_{q_i} calculados, quantizados com o menor número de bits possível. Para reconstruir o sinal, calcule um trecho de N amostras do sinal reconstruído usando[1]

```
[sinal_saida, zs] = filter(1, ai, d, zs);
```

em Julia

```
y = fxfilt(1, ai, d, zs)
zs = y[end-p+1:end]
```

em que a variável `zs` armazena as condições iniciais do filtro a partir do estado final do quadro anterior, e deve ser inicializada com p zeros antes do primeiro quadro (veja que `zs` é usada para calcular o sinal `y0`).

[1] O codificador contém um decodificador também.

(i) Experimente reconstruir o sinal agora quantizando os coeficientes do preditor com 7 bits, os β_{q_i} com 5 bits. Estime a taxa de bits/s necessária para este codificador[2]. Você pode conseguir uma precisão melhor se verificar qual é a faixa dinâmica das variáveis para aproveitar melhor o número de bits disponível. No momento de escolher os ganhos do livro-código, é melhor usar os coeficientes do filtro quantizados ou não quantizados?

Referências

[1] T. Dutoit e F. Marqués. *Applied Signal Processing*. Springer, 2009.

[2] L. R. Rabiner e R. W. Schafer. *Theory and applications of digital speech processing*. Pearson, Upper Saddle River, 1st ed Edição, 2011. OCLC: ocn476834107.

[3] S. A. Zahorian e H. Hu. A spectral/temporal method for robust fundamental frequency tracking. *The Journal of the Acoustical Society of America*, 123(6):4559–4571, jun. 2008.

[2]Lembre que, na prática, é aplicada uma transformação não-linear aos coeficientes α_k para permitir que sejam quantizados com um número menor de bits. Também são usados diversos outros truques (como quantização vetorial) para reduzir a taxa de transmissão, mantendo a qualidade do sinal — desta maneira consegue-se uma taxa de transmissão menor, podendo-se chegar a 4800 bps.