quantization

August 21, 2019

1 Quantização Escalar

Quando representamos um sinal contínuo através de amostras, formando um sinal discreto, se a taxa de amostragem uniforme for superior à taxa de Nyquist, não haverá perda de informação, e assim será possível reconstituir o sinal contínuo a partir de suas amostras. O processo de amostragem não resulta em perda em fidelidade ao sinal original, desde que seja respeitada a taxa de Nyquist. A quantização gera uma representação em termos de amplitudes discretas, resultando em uma perda irreparável de fidedignidade. Combinando amostragem e quantização geramos uma sequência discreta de amostras com amplitudes discretas, às quais associamos palavras binárias distintas para cada uma das possíveis amplitudes discretas na saída do quantizador, produzindo assim uma representação digital de um sinal.

Quantização escalar é um mapeamento Q de valores reais x de uma variável aleatória contínua X nos valores $y = Q(x)$, mais próximos de x (em termos de uma determinada medida de distorção), de um conjunto discreto e finito $Y = y_1, y_2, \ldots, y_M$. Os valores y_i , $i = 1, 2, \ldots, M$, são chamados níveis de saída, ou valores de representação, ou ainda valores de aproximação. Y é chamado de *codebook* ou conjunto de aproximação.

O quantizador escalar é determinado pelo conjunto de limiares $\mathcal{T} = \{t_i\}$, $i = 0, 1, \ldots, M$ e pelo conjunto de pontos de representação $\mathcal{Y} = \{y_i\}$, $i = 1, \ldots, M$. Os limiares dividem exaustivamente o domínio R em subintervalos (ou células, regiões de representação) $\Delta_i = (t_{i-1}, t_i]$ disjuntas, ou seja, $\Delta_i \cap \Delta_j = \emptyset$. Diz-se que a divisão é exaustiva pois $\bigcup_{i=1}^{M} \Delta_i = R$. Esta divisão é tal que existe apenas um y_i associado a cada intervalo Δ_i, ou seja, $y_i = Q(x)$ se e somente se $x \in \Delta_i$.

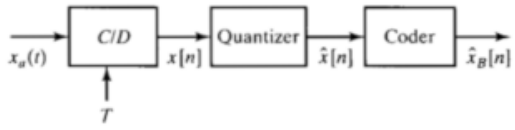
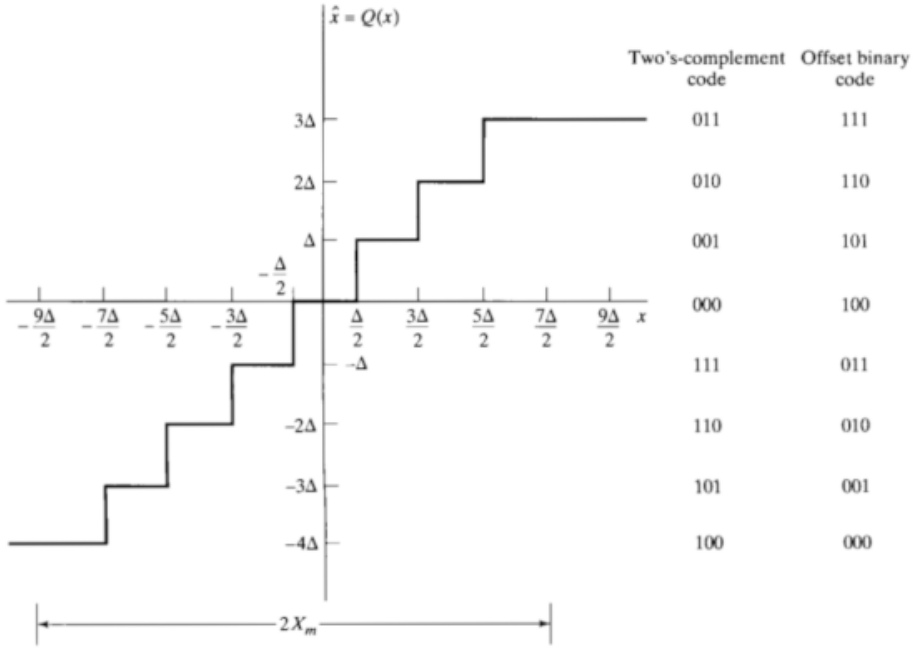


Figure 4.47 Conceptual representation of the system in Figure 4.45.

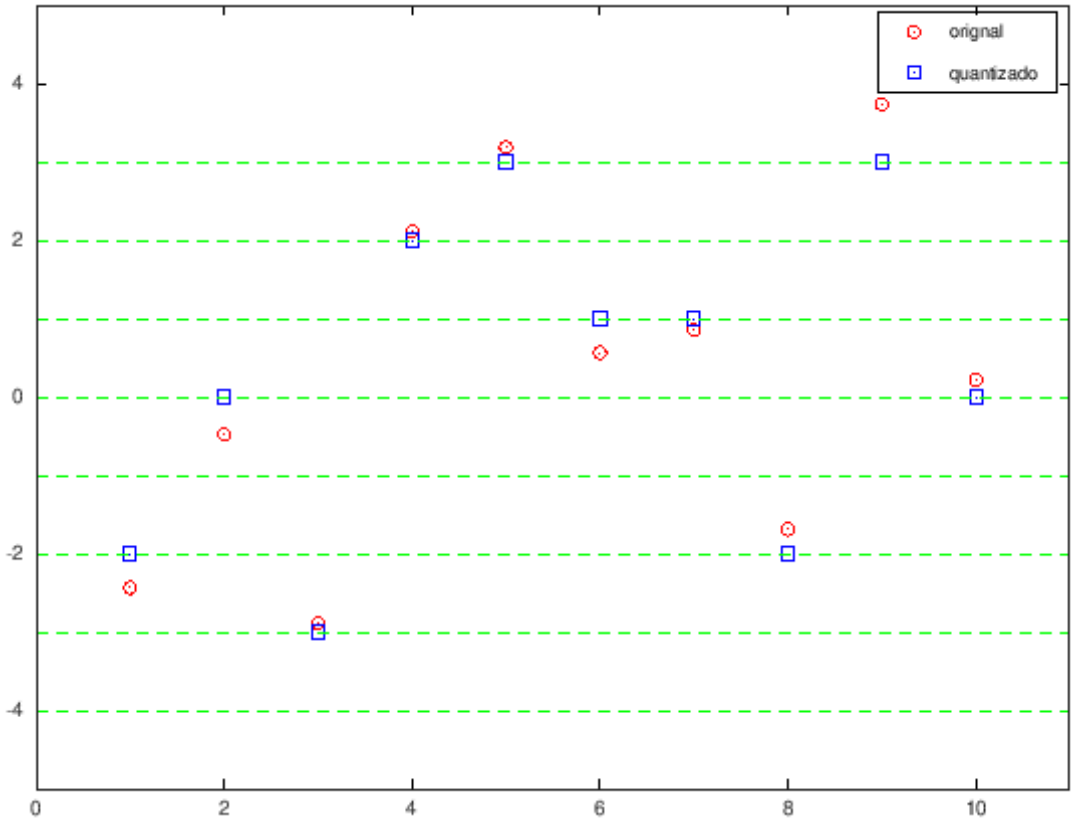


(Alan V. Oppenheim e Ronald W. Schafer, Digital Signal Processing)

```
In [2]: deta = 1;
        N = 10; % numero de amostras
        y = [-4, -3, -2, -1, 0, 1, 2, 3]; % codebook (pontos de representacao)
        t = [-3.5, -2.5, -1.5, -0.5, 0.5, 1.5, 2.5]; % partition (limiares)
        Xm = 4; % 2*Xm extensao do sinal
        x = 2*Xm*rand(1,N)-Xm; % sinal aleatorio
        xq = []; idx = [];
        for xx=x,
          i = find(t >= xx, 1);
          if isempty(i), i=length(y); endif
          idx = [idx, i];
          xq = [xq, y(i)];
        endfor
        [x;xq;idx]

ans =

 Columns 1 through 8:

  -2.43484  -0.47520  -2.88671   2.11114   3.18706   0.56415   0.85590  -1.68972
  -2.00000   0.00000  -3.00000   2.00000   3.00000   1.00000   1.00000  -2.00000
   3.00000   5.00000   2.00000   7.00000   8.00000   6.00000   6.00000   3.00000
```

```
 Columns 9 and 10:

   3.73441   0.21679
   3.00000   0.00000
   8.00000   5.00000
```

```
In [3]: figure; plot(x,'ro',xq,'bs');
        legend('orignal','quantizado');
        axis([0 N+1 -Xm-1 Xm+1]);
        for yy = y,
          line ([0 N+1], [yy yy], "linestyle", "--", "color", "g");
        endfor
```

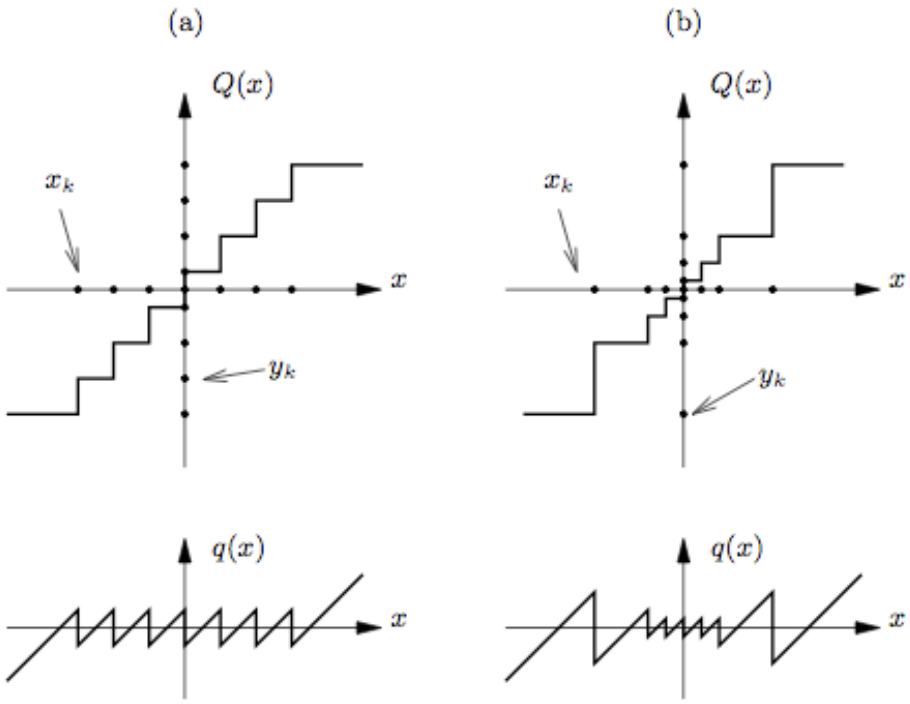


1.1 Ruído de Quantização

É inerente ao processo de quantização a introdução de um erro, chamado *erro de quantização* ou *ruído de quantização*.

O erro de quantização (ou distorção) esperado é dado por

$$D(Q) = E\left\{d(x, Q(x))\right\}, \tag{1}$$



ruído de quantização

onde $d(x, Q(x))$ é uma medida de distorção entre x e $Q(x)$, dada por $d(\cdot)$.

A taxa de quantização é o número de bits R que é utilizado na representação de um valor x. Ela é dada em bits por amostra.

Para um quantizador com taxa fixa temos $R = \log_2 M$ bits por amostra.

```
In [4]: M = length(y);
        R = log2(M);
        printf('O quantizador possui uma taxa de %.1f bits por amostra.\n',R);
        function d = distorcao(x, xq, dist)
           d = sum(dist(x,xq))/length(x);
        endfunction
        mse = @(x,y) (x-y).^2;
        d = distorcao(x,xq,mse);
        printf('A distorcao media foi de %.3f.\n',d);
```

```
O quantizador possui uma taxa de 3.0 bits por amostra.
A distorcao media foi de 0.137.
```

1.2 Função para quantização no GNU Octave

O pacote *communications* possui algumas uma função para fazer a quantização de um sinal: *quantiz*.

```
In [1]: pkg load communications
        help quantiz
```

```
'quantiz' is a function from the file /usr/share/octave/packages/communications-1.2.1/quantiz.m

 -- Function File: QIDX = quantiz (X, TABLE)
 -- Function File: [QIDX, Q] = quantiz (X, TABLE, CODES)
 -- Function File: [ QIDX, Q, D] = quantiz (...)

     Quantization of an arbitrary signal relative to a partitioning.

     'qidx = quantiz (x, table)'
          Determine position of x in strictly monotonic table.  The
          first interval, using index 0, corresponds to x <= table(1).
          Subsequent intervals are table(i-1) < x <= table(i).

     '[qidx, q] = quantiz (x, table, codes)'
          Associate each interval of the table with a code.  Use
          codes(1) for x <= table(1) and codes(n+1) for table(n) < x <=
          table(n+1).

     '[qidx, q, d] = quantiz (...)'
          Compute distortion as mean squared distance of x from the
          corresponding quantization values.

Additional help for built-in functions and operators is
available in the online version of the manual.  Use the command
'doc <topic>' to search the manual index.

Help and information about Octave is also available on the WWW
at https://www.octave.org and via the help@octave.org
mailing list.
```

O código anterior pode ser substituído por esta função, conforme o exemplo abaixo. Note ainda que a função pode ser chamada sem passar o *codebook* como parâmetro. Se o resultado desejado for apenas os índices, o *codebook* não é necessário. Caso o resultado desejado contemple também os valores quantizados, o *codebook* deverá ser passado como parâmetro.

```
In [7]: [idx2, xq2, d2] = quantiz (x, t, y);
        disp(d);
        disp(sum(xq - xq2))

 0.13684
0
```

1.3 Entropia na saída do quantizador

As probabilidades dos níveis de representação de um quantizador podem ser determinadas, conhecendo-se a pdf do sinal.

Seja $f(x)$ a pdf (função densidade de probabilidade) de X. Podemos calcular a probabilidade do i-ésimo nível de reprodução (a probabilidade de $x \in \Delta_i$) como

$$P(y_i) = \int_{t_i-1}^{t_i} f(x)\mathrm{d}x. \tag{2}$$

A entropia da saída do quantizador é igual a

$$H(Y) = -\sum_{i=1}^{M} P(y_i)\log_2 P(y_i). \tag{3}$$

Um código de comprimento variável poder ser utilizado para representar a saída do quantizador (exemplo: código de Shannon, código Huffman, ou codificação aritmética).

1.4 Quantização escalar uniforme

A quantização escalar é uniforme quando os limiares estão igualmente espaçados, e desta forma, as células possuem o mesmo tamanho (exceto as extremas, primeira e última), ou seja, $|\Delta_i| = \delta$, e o ponto de representação localiza-se no ponto médio da célula,

$$y_i = \frac{t_{i-1} + t_i}{2} = t_{i-1} + \frac{\delta}{2}, \quad i = 1, 2, \ldots, M. \tag{4}$$

1.4.1 Quantizador escalar uniforme para valores não negativos

Dada a entrada x, a célula associada a x é determinada por

$$i = \lfloor x/\delta \rfloor, \tag{5}$$

onde δ é a largura de cada célula e $\lfloor \cdot \rfloor$ representa a função piso, operação de arredondamento para o maior inteiro menor ou igual ao argumento.

O valor de aproximação para a entrada x é dado por

$$y = Q(x) = \delta \left\lfloor \frac{x}{\delta} \right\rfloor \tag{6}$$

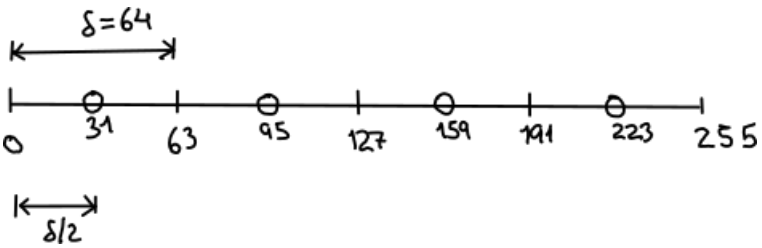
isto é, a i-ésima célula é determinada por $\Delta_i = (i\delta, (i+1)\delta]$ e $y_i = i\delta + \delta/2$.

1.4.2 Exemplo: quantização de uma imagem

Vejamos o exemplo da quantização de uma imagem em tons de cinza.

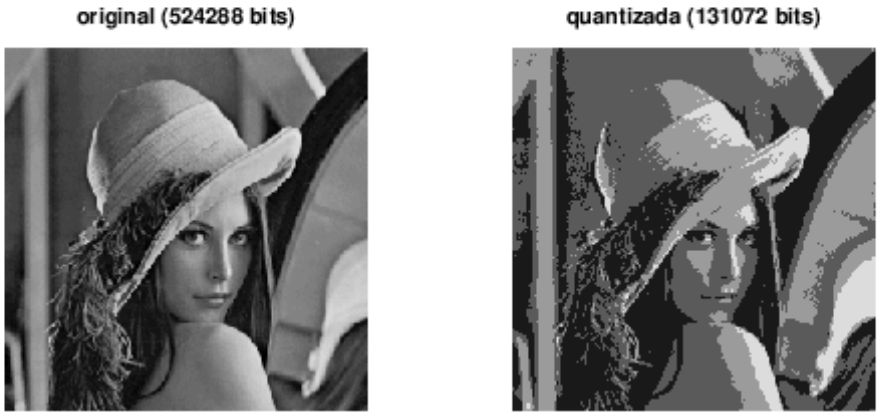
As valores dos pixels da imagem original em tons de cinza, codificada com 8 bits/amostra, estão no invervalor [0,255], ou seja, o valor dos pixels assume qualquer valor inteiro em [0,255]. No exemplo abaixo iremos quantizar esta imagem utilizando 2 bits/amostra. Para tanto, iremos utilizar o quantizador uniforme apresentado abaixo. A seguir segue o exemplo em Octave.

```
In [9]: X = imread('imgs/lena256bw.pnm');
        nbits = 2;
        delta = 256/2^nbits; % valor dos pixels no intervalo [0, 255]
        codebook = [delta/2-1:delta:255-delta/2];
        partition = [delta-1:delta:255-delta];
```



quantizador

```
x = double(X(:));
idx = floor(x/delta);
% de forma equivalente, podemos utilizar a funcao quantiz
% [idx, xq, d] = quantiz(x,partition,codebook);
xq = codebook(idx+1)';
Xq = reshape(xq,size(X));
sizeX = prod(size(X))*8; % numero de bits utilizados para armazenar a imagem original
sizeXq = prod(size(Xq))*nbits; %     "      imagem quantizada
tit1 = sprintf('original (%d bits)',sizeX);
tit2 = sprintf('quantizada (%d bits)',sizeXq);
figure; subplot(1,2,1); imshow(X); title(tit1); subplot(1,2,2); imshow(uint8(Xq)); titl
```



```
In [10]: d = distorcao(x,xq,mse);
         printf('A distorcao media foi de %.3f.\n',d);

         % comparando com o resultado da funcao quantiz
         [idx, xq, d] = quantiz(x,partition,codebook);
         d

A distorcao media foi de 345.015.
d =  345.01
```

1.5 Distorção no quantizador escalar uniforme

Se $f(x)$ é conhecida, então podemos calcular a distorção esperada do quantizador

$$D(Q) = \int_{-\infty}^{\infty} f(x)d(x, Q(x))dx = \sum_i \int_{t_{i-1}}^{t_i} f(x)d(x, y_i)\mathrm{d}x. \tag{7}$$

Se o erro de distorção é medido pelo erro quadrático, então $D(Q)$ fornecerá o erro quadrático médio (MSE, *Mean Squared Error*):

$$D(Q) = \sum_i \int_{t_{i-1}}^{t_i} f(x)(x - y_i)^2\mathrm{d}x. \tag{8}$$

1.6 Quantizador não uniforme

Suponha que $X \sim f$, onde f é uma distribuição não-uniforme.

Por exemplo:

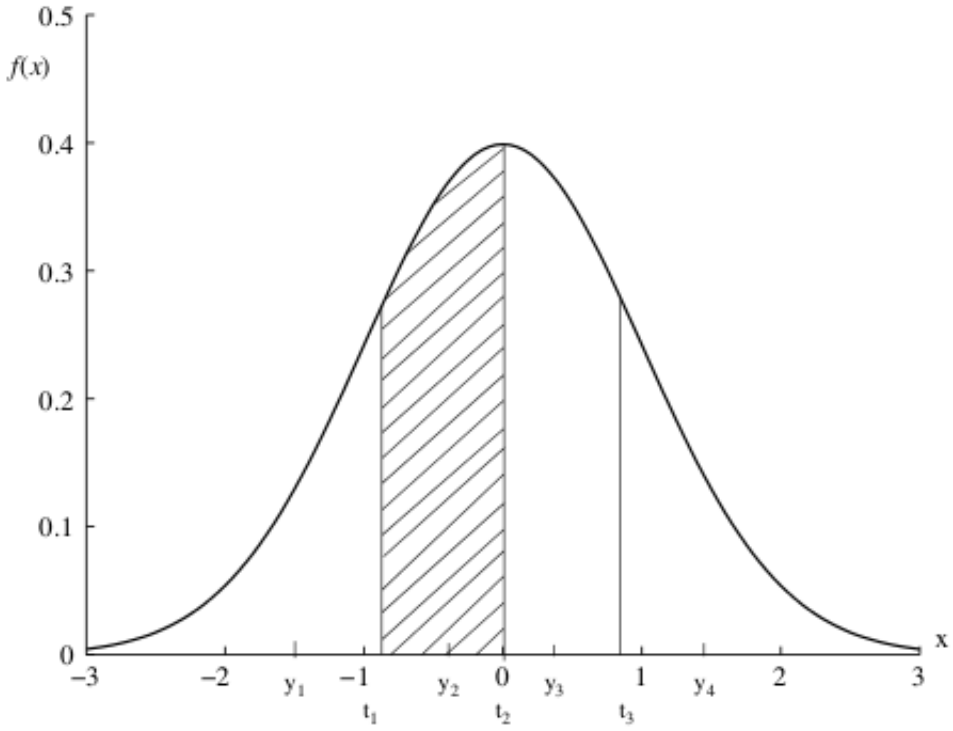
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-(x-\mu)^2/(2\sigma^2)\right) \tag{9}$$

Note que neste caso a quantização uniforme não será aquela que fornecerá o menor valor esperado da distorção. O quantizador com menor distorção será tal que cada intervalo Δ_i terá a mesma massa de probabilidade e o ponto de representação de cada intervalo será dado pelo valor esperado no intervalo.

De forma geral, não conhecemos a distribuição subjacente do sinal que queremos quantizar. Iremos então ver a abordagem proposta por Lloyds para buscar o quantizador ótimo.

1.7 Método de Lloyd

Lloyd propôs um método iterativo, chamado de 'método I', para o projeto de um quantizador escalar (Lloyd, 1982). O algoritmo para o caso de dados com uma distribuição de probabilidade desconhecida é dado a seguir: 1. Seja M o número de pontos de representação e $\epsilon \geq 0$, um limiar de distorção. Considere um conjunto de representação inicial $\mathcal{C}^{(0)} = \{\hat{x}_1^{(0)}, \ldots, \hat{x}_M^{(0)}\}$, um conjunto de treinamento $\mathcal{T} = \{x_1, \ldots, x_N\}$ e $m = 0$, o número inicial de iterações. 2. Dado $\mathcal{C}^{(m)}$ encontre a partição mínima $P(\mathcal{C}^{(m)}) = \{S_i^{(m)} : i = 1, \ldots, M\}$ das sequências de treinamento: $x_j \in S_i^{(m)}$ se



distribuição gaussiana

$d(x_j, \hat{x}_i^{(m)}) \leq d(x_j, \hat{x}_l^{(m)})$, para todo $l \neq i$. Calcule, então, a distorção média definida por:

$$D^{(m)} = D[(\mathcal{C}^{(m)}, P(\mathcal{C}^{(m)}))] \tag{10}$$

$$= \frac{1}{N} \sum_{j=1}^{N} \min_{\hat{x} \in \mathcal{C}^{(m)}} d(x_j, \hat{x}). \tag{11}$$

3. Se $(D^{(m-1)} - D^{(m)})/D^{(m)} \leq \epsilon$, interrompe a iteração e considera-se $\mathcal{C}^{(m)}$ como o alfabeto final de representação (*codebook*); caso contrário, continue. 4. Encontre o alfabeto ótimo de representação para $P(\mathcal{C}^{(m)})$, $\hat{x}(P(\mathcal{C}^{(m)})) = \{\hat{x}(S_i) : i = 1, \ldots, M\}$ para $P(\mathcal{C}^{(m)})$, onde

$$\hat{x}(S_i) = \frac{1}{||S_i||} \sum_{x \in S_i} x. \tag{12}$$

5. Faça $\mathcal{C}^{(m+1)} = \hat{x}(P(\mathcal{C}^{(m)}))$, incremente m e volte ao passo 2.

O algoritmo descrito acima utiliza um determinado $\mathcal{C}^{(0)}$ como *codebook* inicial para o processo de otimização. Existem diversas maneiras de escolher um alfabeto inicial, a mais simples delas consiste em escolher pontos de representação espaçados e coincidentes com os dados de treinamento, ou seja, os dados utilizados para se estabelecer o *codebook*.

O problema das k médias é um problema NP-difícil, desta forma, qualquer algoritmo utilizável levará a uma solução ótima local para o problema. Métodos iterativos, como o de Lloyds, garantem melhorias sucessivas na solução a partir de um ponto inicial. Como a função de minimização, em geral, não é convexa, cada mínimo local terá uma região de atração para si. Os parâmetros de inicialização podem levar a diferentes soluções, algumas melhores que as demais.

1.7.1 Lloyds no GNU Octave

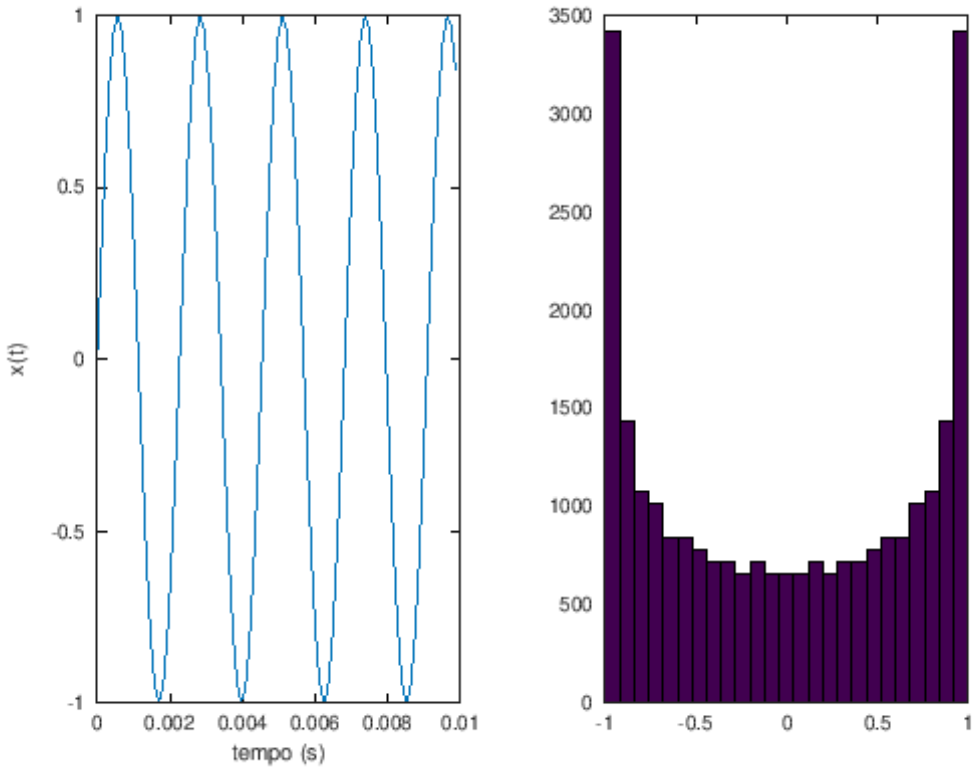
No pacote *communications* do GNU Octave há a função *lloyds* que implementa o método de Lloyds. Basta fornecer o sinal e um *codebook* inicial (ou o tamanho do *codebook*, neste caso o método assumirá como *codebook* inicial o *codebook* uniforme). A saída do algoritmo será o quantizador (limiares e pontos de representação) que minimiza a distorção para o sinal dado.

```
In [ ]: help lloyds
```

```
'lloyds' is a function from the file /usr/share/octave/packages/communications-1.2.1/lloyds.m

 -- Function File: [TABLE, CODES] = lloyds (SIG, INIT_CODES)
 -- Function File: [TABLE, CODES] = lloyds (SIG, LEN)
 -- Function File: [TABLE, CODES] = lloyds (SIG, ..., TOL)
 -- Function File: [TABLE, CODES] = lloyds (SIG, ..., TOL, TYPE)
 -- Function File: [TABLE, CODES, DIST] = lloyds (...)
 -- Function File: [TABLE, CODES, DIST, RELDIST] = lloyds (...)

     Optimize the quantization table and codes to reduce distortion.
     This is based on the article by Lloyd

     S. Lloyd _Least squared quantization in PCM_, IEEE Trans Inform
     Theory, Mar 1982, no 2, p129-137

     which describes an iterative technique to reduce the quantization
     error by making the intervals of the table such that each interval
     has the same area under the PDF of the training signal SIG.  The
     initial codes to try can either be given in the vector INIT_CODES
     or as scalar LEN.  In the case of a scalar the initial codes will
     be an equi-spaced vector of length LEN between the minimum and
     maximum value of the training signal.

     The stopping criteria of the iterative algorithm is given by

          abs(DIST(n) - DIST(n-1)) < max(TOL, abs(EPS*max(SIG))

     By default TOL is 1.e-7.  The final input argument determines how
     the updated table is created.  By default the centroid of the
     values of the training signal that fall within the interval
     described by CODES are used to update TABLE.  If TYPE is any other
     string than "centroid", this behavior is overridden and TABLE is
     updated as follows.

          TABLE = (CODE(2:length(CODE)) + CODE(1:length(CODE-1))) / 2

     The optimized values are returned as TABLE and CODE.  In addition
     the distortion of the optimized codes representing the training
     signal is returned as DIST.  The relative distortion in the final
     iteration is also returned as RELDIST.
```

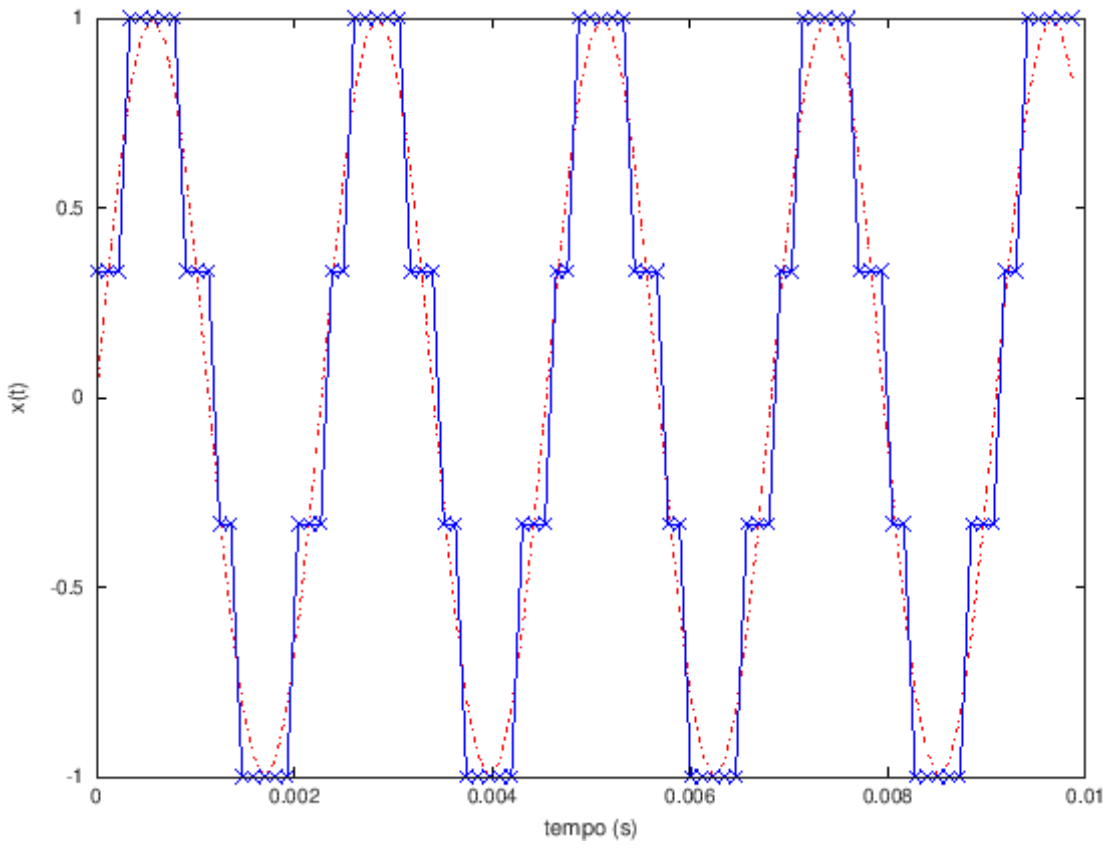
```
     See also: quantiz.

Additional help for built-in functions and operators is
available in the online version of the manual.  Use the command
'doc <topic>' to search the manual index.

Help and information about Octave is also available on the WWW
at https://www.octave.org and via the help@octave.org
mailing list.
```

```
In [11]: fs = 8820; % Hz
         f0 = 440; % Hz  (A4)
         t = [0:1/fs:3]'; % vetor tempo (s)
         x = sin(2*pi*f0*t);
         W = 10E-3; % janela de 10 ms
         WL = floor(W*fs); % # de amostras
         figure; subplot(1,2,1);
         plot(t(1:WL),x(1:WL)); ylabel('x(t)'); xlabel('tempo (s)');
         subplot(1,2,2); hist(x,25);
         % player = audioplayer(x, fs, 12);
         % play(player);
         sound(x,fs);
```



In [12]:
```
nbits = 2;
M = 2^nbits;
xmax = 1; xmin = -1;
u_codebook = [0:M-1]' / (M - 1) * (xmax - xmin) + xmin;
u_partition = (u_codebook(1:M-1) + u_codebook(2:M))/2;
[idx, xq, d] = quantiz(x, u_partition, u_codebook);
figure; plot(t(1:WL),x(1:WL),'r-.',t(1:WL),xq(1:WL),'b-x'); ylabel('x(t)'); xlabel('t
% player = audioplayer(xq, fs, 12);
% play(player);
sound(xq,fs);
printf('distorcao = %.3f\n',d);
```

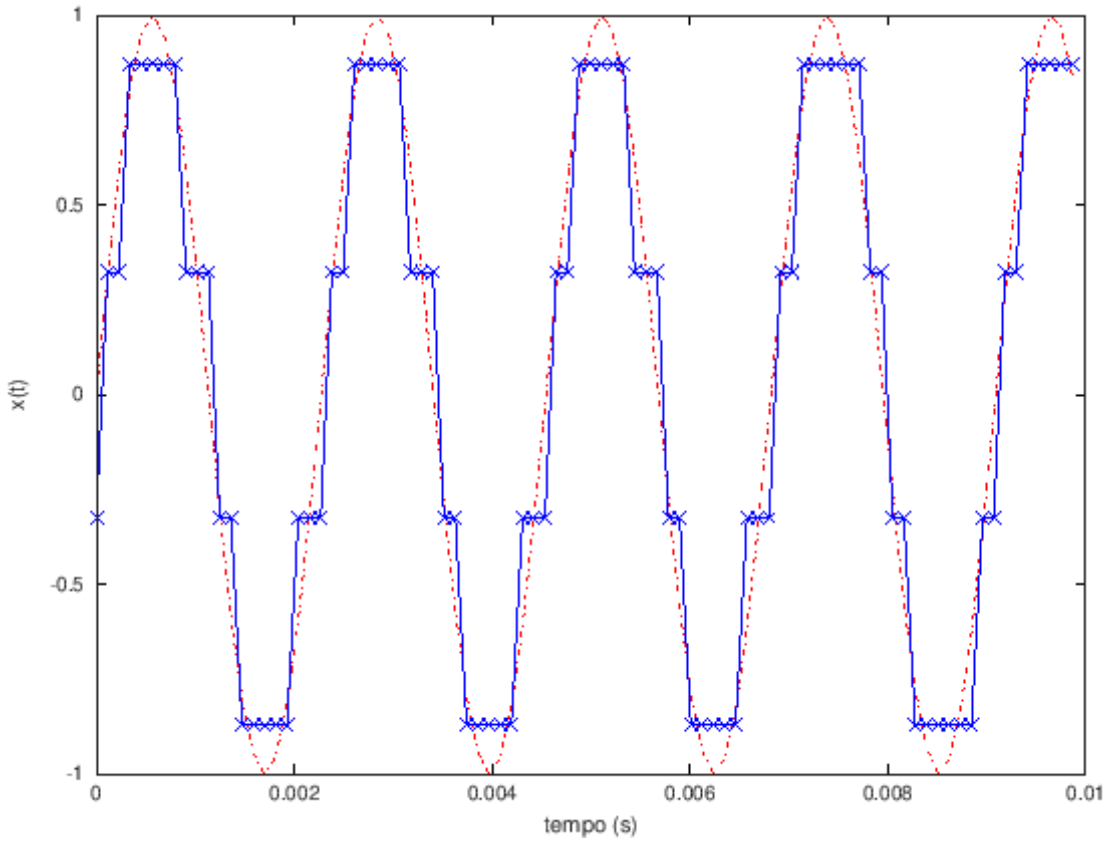
```
distorcao = 0.030
```



In [13]:
```
[l_partition, l_codebook] = lloyds (x, M);
[idx, xq, d] = quantiz(x, l_partition, l_codebook);
figure; plot(t(1:WL),x(1:WL),'r-.',t(1:WL),xq(1:WL),'b-x'); ylabel('x(t)'); xlabel('t
```

```
% player = audioplayer(xq, fs, 12);
% play(player);
sound(xq,fs);
printf('distorcao = %.3f\n',d);
```

```
distorcao = 0.021
```



1.8 Performance do Quantizador

Seja $p(x)$ a pdf do sinal de entrada x, então o erro médio quadrático (MSE) devido à quantização será dado por

$$\sigma_q^2 = \sum_{k=1}^{M} \int_{t_{k-1}}^{t_k} (x - y_k)^2 p(x) \mathrm{d}x. \qquad \text{(eq-1)}$$

Se M for grande e a pdf $p(x)$ for suave, poderemos aproximar $p(x)$ no intervalo $(t_{k-1}, t_k]$ como

$$p(x) \approx p\left(\frac{t_{k-1} + t_k}{2}\right), \quad t_{k-1} < x \leq t_k, \qquad \text{(eq-2)}$$

e assim, a equação **??** poderá ser reescrita como

$$\sigma_q^2 = \sum_{k=1}^{M} p\left(\frac{t_{k-1}+t_k}{2}\right) \int_{t_{k-1}}^{t_k} (x-y_k)^2 \mathrm{d}x. \tag{eq-3}$$

Mostra-se que

$$\int_{t_{k-1}}^{t_k} (x-y_k)^2 \mathrm{d}x = \Delta_k \left[\left(y_k - \frac{t_{k-1}+t_k}{2} \right)^2 + \frac{\Delta_k^2}{12} \right], \tag{eq-4}$$

onde $\Delta_k = t_k - t_{k-1}$ é o tamanho do passo do quantizador.

Para minimizar o MSE devemos escolher $y_k = (t_{k-1}+t_k)/2$, de forma que o primeiro termo em **??** se anule. Ou seja, devemos escolher os pontos de representação como o ponto médio dos limiares dos intervalos (obs.: isto ocorre devido à aproximação feita para p suave e M grande; no caso geral, deveremos ter os pontos de representação no valor esperado de cada intervalo).

Vamos definir p_k como a probabilidade de x pertencer ao intervalo $(t_{k-1}, t_k]$. Usando a aproximação feita anteriormente, teremos

$$p_k = \Pr(t_{k-1} < x \leq t_k) \approx p\left(\frac{t_{k-1}+t_k}{2}\right) \Delta_k, \tag{eq-5}$$

e assim podemos reescrever a equação **??** como

$$\sigma_q^2 = \frac{1}{12} \sum_{k=1}^{M} p_k \Delta_k^2. \tag{13}$$

1.8.1 Performance do Quantizador Uniforme

Para o quantizador uniforme o passo é constante ($\Delta_k = \Delta$ para todo k). Teremos assim

$$\sigma_q^2 = \frac{\Delta^2}{12} \underbrace{\sum_{k=1}^{M} p_k}_{=1} = \frac{\Delta^2}{12}. \tag{eq-6}$$

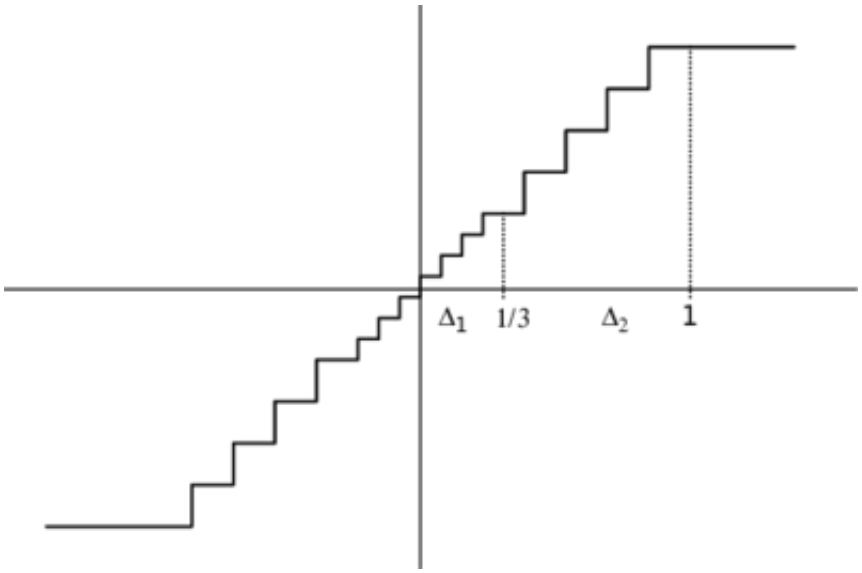
Note que a potência do ruído de quantização é independente da distribuição do sinal.

A performance do quantizador será expressa pela relação sinal-ruído de quantização (SQNR),

$$\mathrm{SQNR} = 10 \log \left(\frac{\sigma_x^2}{\sigma_q^2} \right) = 10 \log \left(\frac{12\sigma_x^2}{\Delta^2} \right) \mathrm{dB}. \tag{eq-7}$$

Sinal Senoide Vamos supor que o sinal de entrada seja da forma $A \sin \omega t$ e um quantizador uniforme com n bits ($2^n = M$). Podemos escolher Δ para que não ocorra saturação. Faremos então $\Delta = A/2^{n-1}$. A potência do sinal senoidal é $\sigma_x^2 = A^2/2$. Usando agora a Equação **??**, teremos

$$\mathrm{SQNR\ (senoide)} = 6n + 1.76\mathrm{dB}. \tag{eq-8}$$



Exemplo de um quantizador não uniforme com 4 bits.

Sinal Gaussiano Iremos supor agora um sinal de entrada com distribuição gaussiana: $p(x) = 1/\sqrt{2\pi\sigma}e^{-(x^2/2\sigma^2)}$. Para que a distorção por saturação seja desprezível, iremos fazer $2^{n-1}\Delta = 4\sigma$, ou seja, teremos $\Delta = \sigma/2^{n-3}$. A potência média quadrática do sinal de entrada é $\sigma_x^2 = \sigma^2$. Usando agora a Equação **??**, teremos

$$\text{SQNR (gauss)} = 6n - 7.3\text{dB}. \qquad \text{(eq-9)}$$

```
In [19]: function S=snr(x,y), S=10*log10(sum(x.^2)/sum(y.^2)); endfunction
         sqnr = snr(x,xq-x);
         printf('A SQNR obtiada foi de %.2f dB\n',sqnr);
         printf('A SQNR dada pela fórmula para sinal senoidal é de %.2f dB\n', 6*nbits+1.76);
```

```
A SQNR obtiada foi de 13.67 dB
A SQNR dada pela fórmula para sinal senoidal é de 13.76 dB
```

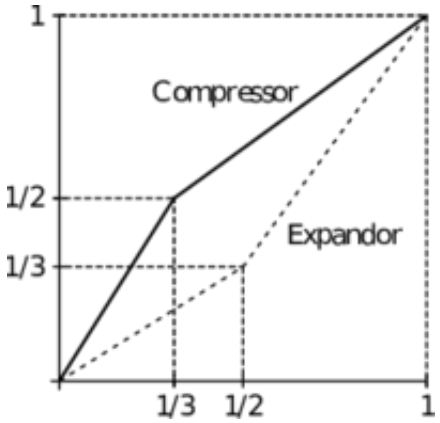
1.8.2 Quantizador Não-Uniforme

Os sinais de fala, por exemplo, estão geralmente concentrados em torno da origem. Desta forma, seria interessante propor um quantizador em que os passos de quantização fossem menores na região de menor amplitude do sinal e maiores na região de maior amplitude. Isto levaria a uma redução do ruído de quantização total.

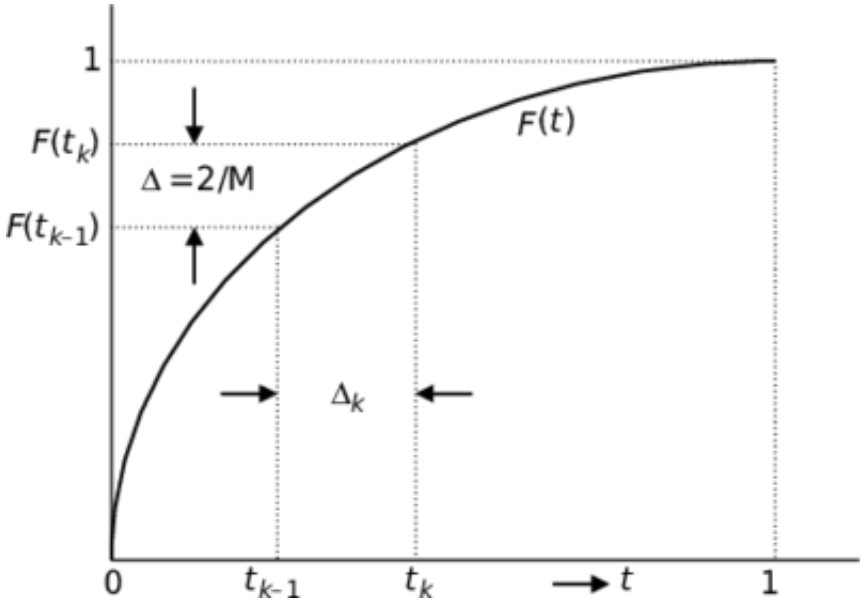
1.8.3 Compressor e Expansor

Podemos utilizar compressor e expansor para implementar um quantizador não uniforme.

- compressor: é feito para amplificar os sinais de baixa amplitude, às custas de atenuar os sinais de alta amplitude;
- expansor: faz o inverso.



Compressor e Expansor



Exemplo de compressor.

O erro médio quadrático devido à quantização é dado pela Equação **??**, repetida a seguir,

$$\sigma_q^2 = \frac{1}{12}\sum_{k=1}^{M} p_k \Delta_k^2, \qquad \text{(eq-10)}$$

onde $p_k = \Pr(t_{k-1} < x \leq t_k)$ e $\Delta_k = (t_k - t_{k-1})$.

Suponha que o compressor apresentado na Figura abaixo seja utilizado.

Os limiares t_{k-1} e t_k, correspondentes a um codificador não uniforme, são mapeados através da função compressora $F(\cdot)$ nos limiares $F(t_{k-1})$ e $F(t_k)$, uniformemente espaçados. Supondo o sinal no intervalo $[-1, +1]$, teremos $\Delta = 2/M$, o passo do codificador uniforme. Conhecendo Δ e a derivada (inclinação) de $F(t)$ no intervalo $[t_{k-1}, t_k]$, podemos determinar Δ_k,

$$\Delta_k = \frac{\Delta}{F'(t_k^*)} = \frac{2}{MF'(t_k^*)}, \quad t_{k-1} < t_k^* < t_k. \qquad \text{(eq-11)}$$

Substituindo Δ_k na Equação **??**, teremos

$$\sigma_q^2 = \frac{1}{3}\sum_{k=1}^{M} \frac{p_k}{M^2(F'(t_k^*))^2}, \quad t_{k-1} < t_k^* < t_k. \tag{eq-12}$$

Se o número de níveis M for grande, o somatório em **??** poderá ser aproximado por uma integral:

$$\sigma_q^2 = \frac{1}{3M^2}\int_{-1}^{+1} \frac{p(x)}{(F'(x))^2}\mathrm{d}x = \frac{2}{3M^2}\int_{0}^{+1} \frac{p(x)}{(F'(x))^2}\mathrm{d}x, \tag{eq-13}$$

onde utilizamos a simplificação em que a $p(x)$ é simétrico par.

Para um sinal com excursão entre $-X_m$ e $+X_m$, teremos

$$\sigma_q^2 = \frac{2X_m^2}{3M^2}\int_{0}^{X_m} \frac{p(x)}{(F'(x))^2}\mathrm{d}x. \tag{eq-14}$$

1.8.4 Compressão Logarítmica

Em um sistema de telecomunicações, desejamos uma SNR constante, independente da distribuição do sinal de entrada. Desejamos então encontrar o compressor F que alcança este objetivo.

$$\mathrm{SNR} = \frac{\sigma_x^2}{\sigma_q^2} = \frac{2\int_0^1 x^2 p(x)\mathrm{d}x}{\frac{2}{3M^2}\int_0^1 \frac{p(x)}{(F'(x))^2}\mathrm{d}x}. \tag{eq-15}$$

A expressão em **??** pode ser feita constante escolhendo

$$F'(x) = \frac{k^{-1}}{x}, \tag{eq-16}$$

com parâmetro k a ser especificado.

A curva de compressão $F(x)$ é obtida realizando-se a integração e escolhendo a constante de integração para que a condição de contorno $F(1) = 1$ seja satisfeita, obtendo assim

$$F(x) = 1 + k^{-1}\ln x. \tag{eq-17}$$

Para este caso a SNR obtida será

$$\mathrm{SNR} = \frac{3M^2}{k^2}. \tag{eq-18}$$

Para sinal com extensão de $-X_m$ a X_m, teremos

$$F(x) = X_m + k^{-1}\ln\left(\frac{x}{X_m}\right). \tag{eq-19}$$

Através da Equação **??** e da escolha feita em **??**, podemos verificar que o tamanho do passo de quantização é proporcional à amplitude do sinal, quando utilizamos a compressão logarítmica dada por $F(x) = 1 + k^{-1}\ln x$.

$$\Delta_k = \frac{2}{MF'(t_k^*)} = \frac{2}{Mk^{-1}}t_k^*. \tag{eq-19}$$

Para valores próximo da origem, esta proporcionalidade não poderá ser mantida, pois $\ln x$ diverge quando $x \to 0$.

Uma lei de compressão prática não pode ter tal descontinuidade, devendo também especificar a compressão dada a sinais de baixa amplitude.