BIẾN ĐỔI TƯƠNG TỰ SỐ

3.1.GIỚI THIỆU CHUNG

Về bản chất, đa số các nguồn tin là các nguồn tương tự, bao gồm tiếng nói, hình ảnh.... Trong chương này, ta xét một số kỹ thuật và phương pháp để biến đổi bản tin từ nguồn tương tự thành chuỗi số một cách hữu hiệu. Sự cần thiết phải biến đổi tương tự thành số bởi lẽ thông tin dưới dạng số thì: ***(i)*** dễ dàng xử lý hơn; ***(ii)*** dễ dàng truyền thông hơn; ***(iii)*** dễ dàng lưu trữ hơn. Chủ đề tổng quát về nén dữ liệu, trong đó biến đổi tương tự số ADC (*Analog-to-Digital Conversion*) là một trường hợp cụ thể, có thể được phân thành hai loại chính:

1. **Lượng tử hoá** (hay **nén dữ liệu có chất thoát**), trong đó nguồn tương tự được lượng tử hoá thành một số hữu hạn các mức. Trong quá trình này, chắc chắn sẽ xảy ra một số méo, dẫn đến sẽ làm thất thoát thông tin. Lượng tin thất thoát này không thể khôi phục lại được. Các kỹ thuật ADC tổng quát như: điều chế xung mã (PCM: *Pulse-Code Modulation*), điều chế xung mã vi sai (DPCM: *Differential Pulse-Code Modulation*), điều chế delta (DM: *Delta Modulation*), lượng tử hoá đều, lượng tử hoá không đều, và lượng tử hoá vec-tơ, đều thuộc loại này. Giới hạn căn bản về hiệu năng của lớp các sơ đồ nén dữ liệu này được cho bởi giới hạn *tỷ số méo* (*rate-distortion bound*).
2. **Mã hoá phi tạp âm** (hay **nén dữ liệu không thất thoát**), trong đó dữ liệu số (thường là kết quả của lượng tử hoá) được nén với mục đích biểu diễn chúng ít bit nhất có thể nhưng vẫn khôi phục hoàn toàn chuỗi dữ liệu gốc từ chuỗi đã được nén (*thể hiện tính hiệu quả của mã*). Các kỹ thuật mã hoá nguồn tin như: mã hoá Huffman, mã Lempel-Ziv, và mã số học, đều thuộc loại nén dữ liệu này. Trong lớp các sơ đồ mã hoá này, không xảy ra thất thoát thông tin. Đạt được giới hạn căn bản về việc nén dữ liệu được cho bởi ***entropy*** của nguồn. Lưu ý rằng, ***entropy*** *là phép đo đánh giá lượng tin có trong nguồn tin.*

Nhận xét: cả hai loại này đều thuộc loại mã hóa nguồn tin, trong đó ở loại 1 được thực hiện để ADC, sau khi đã có chuỗi số liệu từ ADC sử dụng loại 2 để thực hiện hiệu quả của mã hóa nguồn tin, hiệu quả của mã hóa nguồn tin có thể được hiểu là, cùng một lượng tin của nguồn tin nếu sau khi mã hóa nhận được ít bit nhất. Muốn vậy cần phải có phép đo đánh giá lượng tin của nguồn tin.

3.2. ĐO LƯỢNG TIN

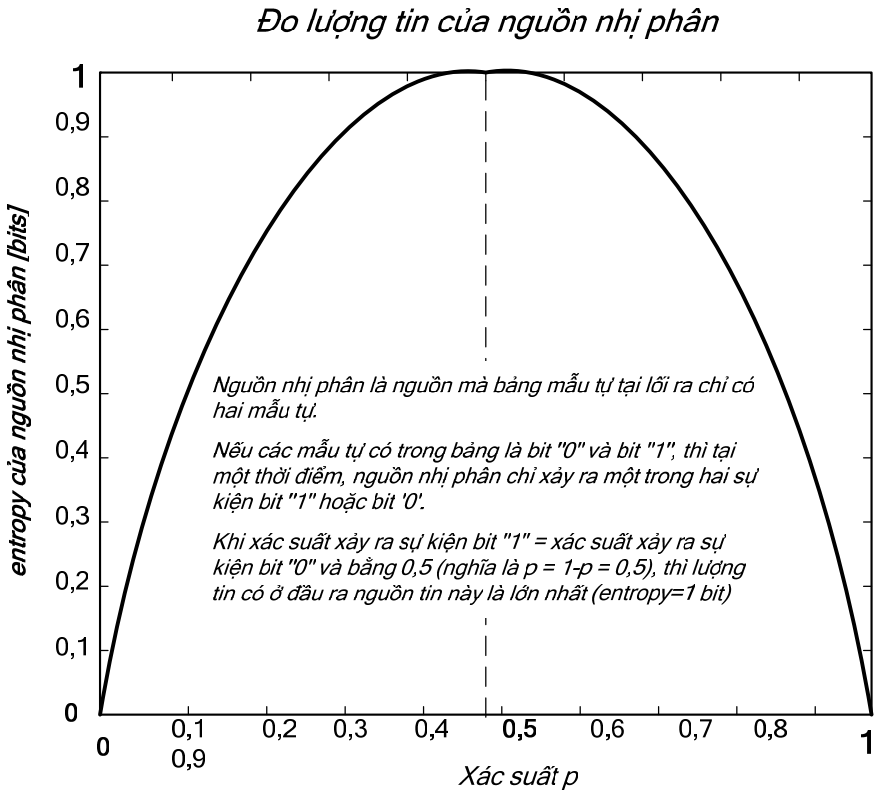
Tín hiệu lối ra của nguồn tin (dữ liệu, tiếng nói, hình ảnh...) có thể được mô hình hoá như một quá trình ngẫu nhiên. Đối với một quá trình ngẫu nhiên rời rạc không nhớ và *dừng*, có thể được xem như việc rút ra độc lập một biến ngẫu nhiên *X*, thì nội dung thông tin, hay *entropy* được định nghĩa là

$$H(X) = \sum_{x \in \chi} p(x).\log\big(p(x)\big) \qquad (3.2.1)$$

trong đó χ ký hiệu cho bảng mẫu tự (chữ cái) của nguồn tin, *p(x)* là xác xuất của mẫu tự (chữ cái) *x*. Cơ số của *logarit* thường được chọn là 2, dẫn đến *entropy* được tính theo đơn vị *bit*. Đối với bảng mẫu tự (chữ cái) nhị phân có các xác suất *p* và *p-1*, thì *entropy* $H_b(p)$ được cho bởi

$$H_b(p) = -p.\log p - (1-p).\log(1-p) \qquad (3.2.2)$$

Đồ thị hàm *entropy* nhị phân được cho ở hình 3.1.



Hình 3.1. Đồ thị hàm *entropy* nhị phân

Entropy của một nguồn tin cho ta một *giới hạn* ***chặn*** (*bound*) cơ bản về số bít cần thiết để biểu diễn một nguồn để khôi phục được đầy đủ. Nói cách khác, có thể thực hiện mã hóa nguồn tin với số bit trung bình cần thiết để khôi phục không lỗi gần bằng *H(X)* như ta muốn nhưng không thể nhỏ hơn *H(X)*.

(nghĩa là, để khôi phục tin không lỗi, thì số bít trung bình ở đầu ra bộ mã hóa nguồn tin phải không được nhỏ hơn *H(X)*, thì có thể mã hóa đầy đủ được lượng tin có trong ngồn tin *H(X)*)

3.2.1. Mã hoá phi tạp âm

Mã hoá không tạp là thuật ngữ chung cho mọi sơ đồ để làm giảm số bít cần thiết cho việc biểu diễn đầu ra nguồn tin nhằm khôi phục hoàn hảo. Định lý mã hoá không tạp, do Shannon phát biểu rằng, có thể sử dụng một mã có ***tốc độ*** gần với *entropy* của nguồn tin nhưng không được nhỏ hơn *entropy* của nguồn để khôi phục hoàn hảo nguồn tin. Nó một cách khác, với bất kỳ $\varepsilon > 0$ nào, ta có thể có một mã có tốc độ nhỏ hơn $H(X) + \varepsilon$, nhưng ta không thể có được một mã có tốc độ nhỏ hơn *H(X)*, bất luận tính phức tạp của bộ mã hoá và bộ giải mã. Tồn tại một nhiều giải thuật để mã hoá nguồn tin phi tạp âm: ví dụ như mã hoá Huffman, mã hoá Lempel-Ziv. Ta thảo luận thuật toán Huffman.

Mã hoá Huffman

Trong mã hoá Huffman, ta gán các từ mã dài hơn cho các lối ra của nguồn tin có khả năng xuất hiện ít hơn và các từ mã ngắn hơn cho các lối ra của nguồn có khả năng xuất hiện nhiều hơn (tăng hiệu năng của mã hóa nguồn tin). Muốn vậy,

Trước hết là tạo cây mã, ta bắt đầu bằng việc hợp nhất hai lối ra của nguồn có xác suất *(khả năng)* nhỏ nhất để tạo một lối hợp nhất mới có xác suất là tổng của các xác suất thành phần. Quá trình này được lặp lại cho tới khi chỉ còn lại một lối ra hợp nhất. Theo cách này, ta tạo ra một *cây mã*.

Sau đó là tạo mã từ cây mã, bằng cách bắt đầu từ gốc của cây và gán các *digit* 0 và 1 cho hai nhánh bất kỳ hợp nhất từ cùng một nút, ta tạo ra được mã.

Bằng cách này cho thấy, ta tạo ra một mã có độ dài trung bình cựu tiểu trong số *các mã tự không tiền tố (prefix-free codes, các mã không tiền tố là các mã trong đó không có từ mã nào là tiền tố "tiếp đầu"của từ mã khác*). Thí dụ dưới đây sẽ cho thấy làm thế nào để thiết kế một mã Huffman.

Bài tập 3.1
[Mã hoá Huffman]

Hãy thiết kế một mã Huffman cho một nguồn tin có bảng mẫu tự (chữ cái) và véctơ xác suất tương ứng là

$$\underbrace{\chi = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 \end{bmatrix}}_{\Downarrow}$$

$$p = \underbrace{\begin{bmatrix} 0,2 & 0,15 & 0,13 & 0,12 & 0,1 & 0,09 & 0,08 & 0,07 & 0,06 \end{bmatrix}}_{\text{Tổng các xác suất này bằng 1 thể hiện cho nguồn tin đã được mã hóa (sự kiện nguồn tin được mã hóa là chắc chắn)}}$$

Hãy tìm độ dài từ mã trung bình của mã được tạo ra, và so sánh nó với entropy của nguồn.

Lời giải

Độ dài từ mã trung bình của mã: Từ thuật toán ở trên ta được cây mã được cho ở 3.2. Độ dài từ mã trung bình cho mã này là

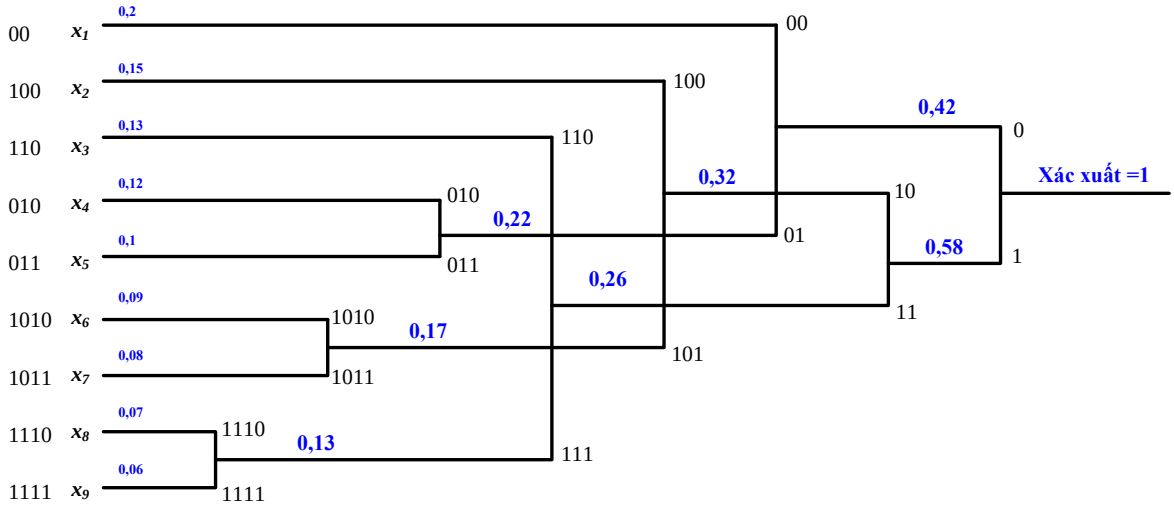
$$\bar{L} = \underbrace{\underbrace{2}_{2\text{ bit}} \times \underbrace{0,2}_{\text{Xác suất xuất hiện đầu ra 2 bit}}}_{\text{Lấy tổng trọng số}} + \underbrace{3 \times \underbrace{(0,15+0,13+0,12+0,1)}_{\text{Xác suất xuất hiện đầu ra 3 bit}}}_{\text{Lấy tổng trọng số}} + \underbrace{4 \times \underbrace{(0,09+0,08+0,07+0,06)}_{\text{Xác suất xuất hiện đầu ra 4 bit}}}_{\text{Lấy tổng trọng số}} = 3,1 \text{ bit}$$

Trên một lối ra của nguồn.

Entropy của nguồn: Entropy của nguồn được cho theo

$$H(X) = -\sum_{i=1}^{9} p_i \log p_i = 3,0371 \qquad \text{bits/đầu ra của nguồn}$$

Thấy rõ, $\bar{L} > H(X)$ như được mong đợi.



Hình 3.2 Cây mã Huffman

Chương trình Matlab **NVD3_entropy.m** dưới đây thực hiện tính toán entropy của một véctơ xác suất *p*.

```
function h=NVD3_entropy(p)
% N1
% p = [0.2 0.15 0.13 0.12 0.1 0.09 0.08 0.07 0.6];
%       H=NVD3_ENTROPY(P) tính entropy theo phương trình 4.2.1
%       vector xác suất đầu vào p.

% Kiểm tra điều kiện 1: tránh xác suất âm
if length(find(p<0))~=0,
  error('Không phải là vector xác suất, vì tồn tại phần tử âm')
end
% Kiểm tra điều kiện 2: Tổng các phần tử của vector xác suất p phải bằng 1
if abs(sum(p)-1)>10e-10,
  error(' Không phải là vector xác suất, vì cộng các thành phần khác 1')
end

h=sum(-p.*log2(p)); % Tính entropy theo phương trình 4.2.1
```

Đại lượng

$$\eta = \frac{H(X)}{\overline{L}} \qquad (3.2.3)$$

được gọi là hiệu quả của mã Huffman. Hiển nhiên, ta luôn luôn có $\eta \le 1$. Ở dạng tổng quát, có thể chỉ rằng, độ dài từ mã trung bình đối với một mã Huffman bất kỳ đều thoả mãn

$$H(X) \le \overline{L} < H(X)+1 \qquad (3.2.4)$$

Nếu ta thiết kế một mã Huffman cho các khối có độ dài K chứ không phải cho từng chữ cái đơn, thì ta sẽ có

$$H(X) \le \overline{L} < H(X)+\frac{1}{K} \qquad (4.2.5)$$

$\Rightarrow$ bằng cách tăng K, ta có thể tiến gần đến *H(X)* như mong muốn. Tất nhiên, việc tăng K sẽ làm tăng mức độ phức tạp. Cần lưu ý rằng, thuật toán mã hóa Huffman không dẫn đến một mã duy nhất do cách gắn tuỳ ý các bít 0 và 1 cho các nhánh khác nhau của cây. Đó là tại sao mà ta nói đến *một*

(a-mạo từ không xác định) mã Huffman chứ không nói mã (*the- mạo từ xác định*) Huffman. Nghĩa là, với cùng một nguồn tin, mỗi lần thực hiện mã hóa ta có thể nhận được kết quả khác.

Chương trình Matlab để thiết kế mã Huffman cho nguồn tin rời rạc không nhớ có véc-tơ xác suất *p* được cho ở file **NVD3_huffman.m** dưới đây. Kết quả chạy chương trình Matlab, ta được cả các từ mã lẫn độ dài trung bình của từ mã.

```
function [h,l]=NVD3_huffman(p);
% N2
% p = [0.2 0.15 0.13 0.12 0.1 0.09 0.08 0.07 0.06]
% NVD3_HUFFMAN   Bộ tạo mã Huffman
% [h,l]=NVD3_huffman(p), Bộ tạo mã Huffman
% 2 kết quả: h là ma trận mã Huffman
%            l là độ dài từ mã trung bình đối với nguồn tin có vector xác suất p.

% Kiểm tra điều kiện 1: tránh xác suất âm
if length(find(p<0))~=0,
  error('Không phải là vector xác suất, vì tồn tại phần tử âm')
end
% Kiểm tra điều kiện 2: Tổng các phần tử của vector xác suất p phải bằng 1
if abs(sum(p)-1)>10e-10,
  error(' Không phải là vector xác suất, vì cộng các thành phần khác 1')
end
%================================
n       = length(p);
q       = p;
m       = zeros(n-1,n);
for i=1:n-1
  [q,l]     = sort(q);
  m(i,:)    = [l(1:n-i+1),zeros(1,i-1)];
  q         = [q(1)+q(2),q(3:n),1];
end
for i=1:n-1
  c(i,:)    = blanks(n*n); % blanks là hàm của Matlab
end
c(n-1,n)    ='0';
c(n-1,2*n)  ='1';
for i=2:n-1
    c(n-i,1:n-1)      = c(n-i+1,n*(find(m(n-i+1,:)==1))...
        -(n-2):n*(find(m(n-i+1,:)==1)));
    c(n-i,n)      ='0';
    c(n-i,n+1:2*n-1)  = c(n-i,1:n-1);
    c(n-i,2*n)    ='1';
    for j=1:i-1
        c(n-i,(j+1)*n+1:(j+2)*n)    = c(n-i+1,...
            n*(find(m(n-i+1,:)==j+1)-1)+1:n*find(m(n-i+1,:)==j+1));
    end
end
%=======================================
for i=1:n
    h(i,1:n)  = c(1,n*(find(m(1,:)==i)-1)+1:find(m(1,:)==i)*n);
    l1(i)     = length(find(abs(h(i,:))~=32));
end
l           = sum(p.*l1); % Tính độ dài trung bình của từ mã
```

Bài tập 3.2
[Mã hóa Huffman]

Một nguồn tin rời rạc không nhớ có bảng chữ cái và các xác suất tương ứng như sau

$$\underbrace{\chi = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \end{bmatrix}}_{\Downarrow}$$

$$p = \begin{bmatrix} 0,1 & 0,3 & 0,05 & 0,09 & 0,21 & 0,25 \end{bmatrix}$$

sẽ được mã hóa bằng mã hóa Huffman. Hãy

1. Xác định *entropy* của nguồn này.
2. Tìm mã Huffman cho nguồn tin và xác định hiệu quả của mã Huffman này
3. Thiết kế mã Huffman cho các chuỗi nguồn có độ dài bằng 2, và so sánh hiệu quả của mã này với hiệu quả của mã đã tìm được trong phần 2.

Lời giải

1. Ta dùng chương trình Matlab **NVD3_entropy.m** để tính *entropy*, bằng cách nhập vectơ xác suất p = [0.1 0.3 0.05 0.09 0.21 0.25], gọi hàm **NVD3_entropy(p),** $\Rightarrow$ kết quả là 2,3549 bit trên ký hiệu nguồn tin.
2. Ta sử dụng chương trình Matlab **NVD3_huffman.m** để thiết kế mã Huffman cho nguồn này. Bằng cách, nhập vectỏ xác suất p = [0.1 0.3 0.05 0.09 0.21 0.25], gọi hàm **NVD3_huffman(p)** $\Rightarrow$Kết quả tìm được: ***(i)*** các từ mã là 010, 11, 0110, 00, và 10; ***(ii)*** độ dài từ mã trung bình đối với mã này là 2,38 ký hiệu nhị phân/đầu ra của nguồn. Vì vậy, hiệu quả của mã này là

$$\eta_1 = \frac{2,3549}{2,38} = 0,9895$$

3. Một nguồn mới mà đầu ra của nó là các cặp chữ cái của nguồn ban đầu có 36 chữ cái lối ra ở dạng $\left\{\left(x_i, x_j\right)\right\}_{i,j=1}^{6}$. Vì nguồn này là không nhớ, nên xác suất của mỗi cặp là tích của các xác suất chữ cái riêng lẻ. Vì vậy, để nhận được véctơ xác suất đối với nguồn mở rộng này, ta phải tạo ra một véc tơ có 36 phần tử, mỗi thành phần sẽ là tích của hai xác suất trong véctơ xác suất gốc *p*. Điều này được thực hiện bằng cách dùng hàm Matlab **kron.m** ở dạng **kron (p,p)**. Sau đó dùng hàm **NVD3_huffman(p)** để tạo mã**, cụ thể là**

Từ cửa sổ lệnh Matlab:

>> p = [0.1 0.3 0.05 0.09 0.21 0.25]

>>p2= kron(p,p)

$\Rightarrow$Kết quả là, tạo ra véctơ xác suất p2 có 36 phần tử là: (lưu ý dấu chấm "**.**" và dấu phẩy "**,**" trong của Matlab)

0.0100 0.0300 0.0050 0.0090 0.0210 0.0250 0.0300 0.0900 0.0150 0.0270
0.0630 0.0750 0.0050 0.0150 0.0025 0.0045 0.0105 0.0125 0.0090 0.0270
0.0045 0.0081 0.0189 0.0225 0.0210 0.0630 0.0105 0.0189 0.0441 0.0525
0.0250 0.0750 0.0125 0.0225 0.0525 0.0625

Từ vectơ xác suất p2, truyền vào hàm **[kq1,kq2] = NVD3_huffman(p2)**

$\Rightarrow$Kết quả là,

$\Rightarrow$Tạo ra các từ mã Huffman được tạo ra **kq1** là:

1110000; 01110; 10110111; 1011001; 111001; 00101; 01111; 000; 011010; 00111;
1001; 1100; 11101110; 011011; 111011110; 111011111; 1110001; 001000; 1011010;

01100; 10110110; 1011000; 101110; 111110; 111010; 1010; 110110; 101111; 11110;

0100; 00110; 1101; 001001; 111111; 0101; 1000

$\Rightarrow$ Độ dài từ mã trung bình đối với nguồn mở rộng này là: **kq2=** 4,7420.

$\Rightarrow$ Entropy của nguồn mở rộng tìm được bằng cách dùng hàm **kq3=NVD3_entropy(p2)**

$\Rightarrow$ **kq3 =** 4,7097

Vì vậy, hiệu suất của mã Huffman này là

$$\eta_2 = \frac{4,7097}{4,7420} = 0,9932$$

cho thấy có một sự cải thiện so với hiệu quả của mã Huffman đã được thiết kế trong phần 2.

Bài tập 3.3

[Một mã Huffman có hiệu quả cực đại]

Hãy thiết kế một mã Huffman cho một nguồn có véctơ xác suất

$$p = \left\{\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}, \frac{1}{64}, \frac{1}{128}, \frac{1}{256}, \frac{1}{256}\right\}$$

Lời giải

Ta sử dụng hàm **NVD3_huffuman.m** để xác định mã Huffman và độ dài từ mã trung bình tương ứng. Kết quả là, các từ mã 1,01, 001, 0001, 00001, 000001, 0000001, 00000000, và 00000001. Độ dài từ mã trung bình là 1,9922 ký hiệu nhị phân/ lối ra của nguồn. Nếu ta tìm *entropy* của nguồn này bằng hàm **NVD3_entropy.m**, thì ta sẽ thấy rằng entropy của nguồn này cũng là 1,9922bít/lối ra của nguồn; do đó hiệu quả của mã này bằng 1. Lưu ý rằng cách gọi hàm tương tự như bài tập 3.2

CÂU HỎI: Trong các điều kiện nào, thì hiệu quả của một mã Huffman bằng 1 ?

3.3. LƯỢNG TỬ HOÁ

Ta đã nghiên cứu hai phương pháp để mã hoá nguồn tin phi tạp âm đó là, việc nén chuỗi đầu ra của nguồn tin sao cho có thể khôi phục đầy đủ thông tin từ dữ liệu đã nén đó. Trong các phương thức này, dữ liệu đã nén là một hàm ***xác định*** của lối ra nguồn tin, và lối ra của nguồn tin cũng là một hàm ***xác định*** của dữ liệu đã nén. Sự tương ứng một–một này giữa dữ liệu đã nén và sản phẩm lối ra của nguồn tin nghĩa là các *entropy* của chúng là bằng nhau, và không bị thất thoát thông tin trong quá trình mã hoá-giải mã.

Trong nhiều ứng dụng, như trong xử lý số của các tín hiệu tương tự, ở đó bảng chữ cái của nguồn tin không phải là rời rạc, thì số các bít cần thiết để biểu diễn một sản phẩm lối ra của nguồn tin là không hữu hạn. Để xử lý lối ra của nguồn theo phương pháp số, thì nguồn phải được lượng tử hoá thành một số các mức hữu hạn. Quá trình này làm giảm số bit xuống thành một số hữu hạn nhưng đồng thời mang vào đó một lượng méo nhất định. Không bao giờ khôi phục được lượng tin bị thất thoát do quá trình lượng tử hóa gây ra.

Ở dạng tổng quát, các sơ đồ lượng tử hoá có thể được phân loại thành hai loại sau: ***(i)*** sơ đồ lượng tử hoá vô hướng, trong đó lại được phân thành lượng tử hóa đều và không đều; ***(ii)*** sơ đồ lượng tử hoá véctơ. Trong lượng tử hoá vô hướng, thì mỗi đầu ra của nguồn tin được lượng tử hoá riêng rẽ, trong khi đó lượng tử hoá véctơ, thì các khối đầu ra của nguồn tin được lượng tử hóa

Các bộ lượng tử hóa vô hướng lại được phân loại thành các bộ lượng tử hóa đều và các bộ lượng tử hóa không đều. Ở lượng tử hóa đều, thì các vùng được chọn có cùng chiều dài. Trong lượng tử hóa không đều, thì cho phép sử dụng các vùng có chiều dài có chiều dài khác nhau. Thấy rõ, ở dạng tổng quát, các bộ lượng tử hoá không đều tốt hơn các bộ lượng tử hoá đều.

3.3.1. Lượng tử hoá vô hướng

Trong quá trình lượng tử hoá vô hướng, dải giá trị của biến ngẫu nhiên X được chia thành N vùng $\Re_i$ với $1 \le i \le N$ không chồng chất nhau, được gọi là các khoảng lượng tử hoá, và trong mỗi vùng ta chọn một điểm được gọi là một mức lượng tử. Khi đó, mọi giá trị của biến ngẫu nhiên mà rơi vào vùng $\Re_i$ được lượng tử thành mức lượng tử thứ i, được ký hiệu là $\hat{x}_i$. Nghĩa là

$$x \in \Re_i \Leftrightarrow Q(x) = \hat{x}_i \tag{3.3.1}$$

trong đó

$$\hat{x}_i \in \Re_i \tag{3.3.2}$$

Hiển nhiên, lượng tử hoá kiểu này tạo ra một ***sai số bình phương trung bình*** **MSE** là $E\left[\left(x-\hat{x}_i\right)^2\right]$. Vì vậy, sai số lượng tử bình phương trung bình được cho bởi

$$D = \sum_{i=1}^{N} \underbrace{\underbrace{\int_{\Re_i} \underbrace{\left(x-\hat{x}_i\right)^2}_{\text{Lỗi}} f_X(x)dx}_{\text{Lấy bình phương}}}_{\substack{\text{Lấy trung bình của biến ngẫu nhiên } (x-\hat{x}_i) \\ \text{có pdf là } f_X(x) \text{ trên vùng } \Re_i}} \tag{3.3.3}$$

Lấy trung bình trên toàn bộ dải giá trị của BNN gồm N vùng ⇒ giả định phân bố lỗi trên tất cả các vùng trong toàn bộ dải giá trị của BNN là như nhau và độc lập thống kê *idd*

trong đó $f_X(x)$ ký hiệu cho hàm mật độ xác suất pdf của biến ngẫu nhiên của nguồn tin. Tỷ số tín hiệu trên tạp âm lượng tử SQNR (Signal-to-Quantization-Noise Ratio) được định nghĩa là

$$\text{SQNR}\Big|_{dB} = 10\log_{10}\frac{E\left[X^2\right]}{D}$$

❖ Lượng tử hoá đều

Trong quá trình lượng tử hoá đồng đều, tất cả các vùng lượng tử hóa ngoại trừ vùng đầu tiên và vùng cuối cùng, tức là vùng $\Re_1$ và $\Re_N$ đều có cùng độ dài được ký hiệu là Δ; vì vậy

$$\Re_1 = (-\infty, a]$$
$$\Re_2 = (a, a+\Delta]$$
$$\Re_3 = (a+\Delta, a+2\Delta]$$
$$\vdots$$
$$\Re_N = (+(N-2)\Delta, \infty]$$

Mức lượng tử hoá tối ưu trong mỗi khoảng lượng tử hoá có thể chứng minh được là trọng tâm của khoảng đó; nghĩa là

$$\hat{x}_i = E\left[X \middle| X \in \Re_i \right] = \frac{\int_{\Re_i} x.f_X(x)dx}{\int_{\Re_i} f_X(x)dx}, \quad 1 \le i \le N \tag{3.3.4}$$

Do đó, việc thiết kế bộ lượng tử hoá đều tương đương với việc xác định và Δ. Sau khi a và Δ đã được xác định, các giá trị của $\hat{x}_i$ và lượng méo dẫn xuất có thể xác định được dễ dàng bằng các biểu thức (3.3.3) và (3.3.4). Trong một số trường hợp, sẽ là thuận tiện hơn cả nếu chọn một cách đơn giản các điểm giữa của các miền lượng tử hoá làm các mức lượng tử hoá, nghĩa là tại khoảng cách $\Delta/2$ tính từ các biên của các miền lượng tử.

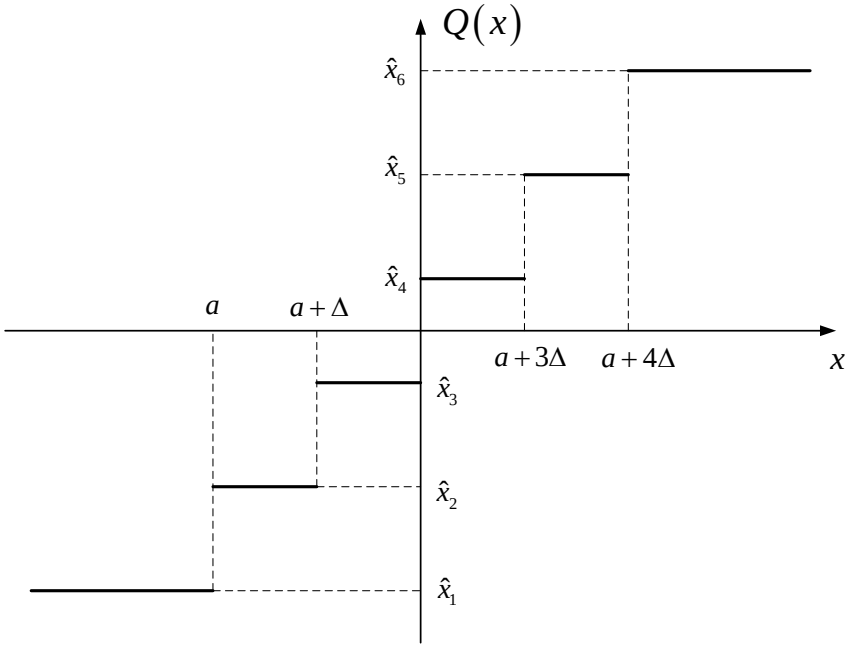
Các đồ thị của hàm lượng tử hoá Q(x) đối với một hàm mật độ xác suất đối xứng của X và các giá trị chẵn và lẻ của N được thể hiện trên các hình vẽ 3.3 và 3.4 một cách tương ứng.

Đối với hàm mật độ xác suất đối xứng, bài toán thậm chí trở nên đơn giản hơn. Trong trường hợp như vậy.

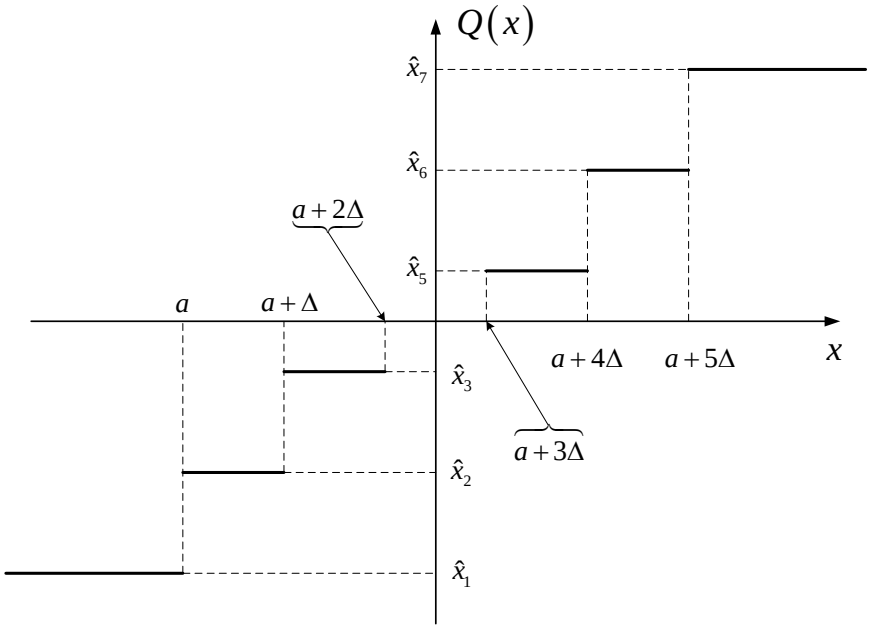
$$\Re_i = \begin{cases} (a_{i-1}, a_i], & 1 \le i \le N-1 \\ (a_{i-1}, a_N), & 1 = N \end{cases} \tag{4.3.5}$$

trong đó

$$\begin{cases} a_0 = -\infty \\ a_i = (i - N/2)\Delta \qquad 1 \le i \le N-1 \\ a_N = \infty \end{cases} \tag{4.3.6}$$



Hình 3.3 Bộ lượng tử hóa đồng đều với N=6 (lưu ý rằng ở đây $a + 2\Delta = 0$)



Hình 3.4 Hàm lượng tử hóa với N=7 (lưu ý rằng ở đây $\hat{x}_4 = 0$)

Cần lưu ý rằng, trong trường hợp này, ta chỉ có một tham số Δ phải được chọn để đạt được méo tối thiểu. Ba chương trình Matlab **NVD3_centroid.m**, **NVD3_mse_dist.m**, **NVD3_uq_dist.m** dưới đây thực hiện tìm trọng tâm của một vùng, sai số lượng tử bình phương trung bình (MSE) đối với một phân bố cho trước và các biên của vùng lượng tử hoá cho trước, và cuối cùng là lượng méo khi dùng bộ lượng tử đều để lượng tử hoá một nguồn cho trước. (Giả sử rằng, các mức lượng tử được đặt tại các trọng tâm của các vùng lượng tử hoá). Để sử dụng các hàm này, thì phân bố của nguồn, mà có thể phụ thuộc đến 3 tham số, phải được cho trong một hàm.

```
function y=NVD3_centroid(funfcn,a,b,tol,p1,p2,p3)
% N3
% CENTROID  tìm trọng tâm của hàm trên một vùng
%       Y = NVD3_CENTROID('F',A,B,TOL,P1,P2,P3) tìm trọng tâm của hàm F được định
%           nghĩa trong một file.m trên vùng [A,B] region. Hàm có thể chứa đếm 3
%           tham số,p 1, P2, P3.
%           tol=sai số tương đối (the relative error).

args   = [];
for n=1:nargin-4
  args =[args,',p',int2str(n)];
end
args   = [args,')'];
funfcn1 ='x_fnct';
y1      = eval(['quad(funfcn1,a,b,tol,[],funfcn',args]);
y2      = eval(['quad(funfcn,a,b,tol,[]',args]);

y       = y1/y2;
```

```
function [y,dist]=NVD3_mse_dist(funfcn,a,tol,p1,p2,p3)
% N4
%NVD3_MSE_DIST   trả lại sai số lượng tử trung bình bình phương.
%[Y,DIST]=NVD3_MSE_DIST(FUNFCN,A,TOL,P1,P2,P3)
% Trong đó:
% funfcn   = hàm phân bố được cho trong một file.m. Nó có thể phụ thuộc vào
             3 tham số,p1,p2,p3.
% a        = vector xác định các biên của các vùng lượng tử.
```

```
              (lưu ý rằng: [a(1),a(length(a))] là support of funfcn)
% p1,p2,p3 = các tha số của funfcn.
% tol      = sai số tương đối.

args  = [];
for n=1:nargin-3
  args =[args,',p',int2str(n)];
end
args   =[args,')'];
for i=1:length(a)-1
  y(i) = eval(['centroid(funfcn,a(i),a(i+1),tol',args]);
end
dist=0;
for i=1:length(a)-1
  newfun  = 'x_a2_fnct' ;
  dist    = dist+eval(['quad(newfun,a(i),a(i+1),tol,[],funfcn,', num2str(y(i)),
args]);
end
```

```
function [y,dist]=NVD3_uq_dist(funfcn,b,c,n,delta,s,tol,p1,p2,p3)

% N5
% NVD3_UQ_DIST  trả lại phân bố của bộ lượng tử hóa đều với các điểm lượng tử hóa
%               được thiết lập tại các trọng tâm

%[Y,DIST]=NVD3_UQ_DIST(FUNFCN,B,C,N,DELTA,S,TOL,P1,P2,P3)
% Trong đó:
% funfcn = hàm mật độ nguồn tin được cho ở một file.m với 3 tham số quan
%                trọng nhất p1,p2,p3.
% [b,c]  = sự hỗ trợ của hàm mật độ nguồn
% n      = số mức lượng tử
%  delta  = kích thước mức.
%  s      = biên của vùng lượng tử phía trái
%  p1,p2,p3 = các tham số của hàm đầu vào.
%  y      = các mức lượng tử.
%  dist   = méo.
%  tol    = sai số tương đối.

if (c-b<delta*(n-2))
  error('Quá nhiều mức đối với dải này.'); return
end
if (s<b)
  error('Biên phía trái ngoài cùng quá nhỏ.'); return
end
if (s+(n-2)*delta>c)
  error('Biên phía trái ngoài cùng quá lớn.'); return
end
args   = [];
for j=1:nargin-7
  args = [args,',p',int2str(j)];
end
args   = [args,')'];
a(1)   = b;
for i=2:n
  a(i) = s+(i-2)*delta;
end
a(n+1) = c;
[y,dist] = eval(['NVD3_mse_dist(funfcn,a,tol',args]);
```

Bài tập 3.4
[Xác định các trọng tâm]

Hãy xác định các trọng tâm của các vùng lượng tử hoá đối với phân bố Gausơ có kỳ vọng bằng 0 và phương sai bằng 1, trong đó các biên của các vùng lượng tử hoá được cho bởi (-5,-4,-2,0,1,3,5)

Lời giải

Phân bố Gausơ được cho trong hàm **NVD3_normal.m**. Phân bố này là một hàm của hai tham số, là trung bình và phương sai, được ký hiệu là m và s (hay σ). Miền xác định của phân bố Gausơ là $(-\infty,\infty)$, nhưng để áp dụng các thường trình tính toán số, thì chỉ cần sử dụng một dải rộng bằng nhiều lần độ lệch chuẩn của phân bố là đủ. Thí dụ có thể dùng dải $\left(m-10\sqrt{s}, m+10\sqrt{s}\right)$. chương trình Matlab **NVD3_sim34e.m** dưới đây sẽ xác định các trọng tâm (các mức lượng tử tối ưu)

```
% function y = NVD3_sim34e
% Chương trình liên kết cho bài tập 3.4.
a = [-10,-5,-4,-2,0,1,3,5,10];
for i=1:length(a)-1
  y(i) = NVD3_centroid('normal',a(i),a(i+1),0.001,0,1);
end
```

Kết quả là, các mức lượng tử hoá là: (-5,1865; -4,2168; -2,3706; -0,4599; 0,7228; 1,5101; 3,2827; 5,1865).

Bài tập 3.5
[Sai số trung bình bình phương MSE]

Trong bài tập minh họa 4.4, hãy xác định sai số trung bình bình phương MSE.

Lời giải

Cho a = (-10, -5, -4, -2, 0, 1, 3, 5, 10), và sử dụng hàm **NVD3_ mse_dist.m**, ta được sai số trung bình bình phương MSE là 0,177. Cụ thể, được cho ở file **NVD3_sim35e.m** dưới đây.

```
% function y = NVD3_sim35e
clc;
clear all;
close all;
a           =   [-10 -5 -4 -2 0 1 3 5 10];
[y,dist]    =   NVD3_mse_dist('normal',a,0.01,0,1);

display('==== XAC DINH SAI SO TRUNG BINH BINH PHUONG MSE - BAI TAP 4.5====');

display('---DU LIEU VAO: vector a ------ ');
disp(a);

display('==>KET QUA: Cac muc luong tu hoa y va MSE ');
display('   Cac muc luong tu hoa y la:');
disp(y)
display('   Sai so trung binh binh phong MSE la:');
disp(dist)
```

Kết quả chạy chương trình là:

```
==== XAC DINH SAI SO TRUNG BINH BINH PHUONG MSE - BAI TAP 3.5====
```

```
---DU LIEU VAO: vector a ------
  -10   -5   -4   -2    0    1    3    5   10
==>KET QUA: Cac muc luong tu hoa y va MSE
  Cac muc luong tu hoa y la:
  -5.1716   -4.2168   -2.3706   -0.7228    0.4599    1.5101    3.2826    5.1716
  Sai so trung binh binh phong MSE la:
   0.1770
```

Bài tập 3.6

[Méo của bộ lượng tử hoá đều]

Hãy xác định MSE đối với một bộ lượng tử hoá đều với 12 mức lượng tử, mỗi mức lượng tử có độ dài bằng 1, được thiết kế cho nguồn Gauso có trung bình 0 và phương sai bằng 4. Giả sử rằng, các vùng lượng tử hoá đối xứng nhau qua trung bình của phân bố.

Lời giải

Do giả thiết về tính đối xứng, nên các biên của các vùng lượng tử hoá là 0, ± 1, ± 2, ± 3, ± 4, và ± 5, và các vùng lượng tử hoá là $(-\infty,5]$, (-5, -4], (-4, -3], (-3, -2], (-2, -1], (-1, 0], (0, 1], (1,2], (2, 3], (3, 4], (4, 5], và $(5, +\infty]$. Điều này có nghĩa là trong hàm **NVD3_uq_dist.m**, ta đặt b = -20, c = 20, Δ = 1, n = 12, s = -5, tol = 0.001, $p_1 = 0$ và $p_2 = 2$. Thay các giá trị này vào hàm **NVD3_uq_dist.m**, ta nhận được méo sai số bình phương là 0,0851, và các giá trị lượng tử là $\pm 0{,}4897$; $\pm 1{,}4691$; $\pm 2{,}4487$; $\pm 3{,}4286$; $\pm 4{,}4089$; $\pm 5{,}6455$.

Hàm **NVD3_uq_mdqnt.m** dưới đây sẽ xác định méo sai số bình phương đối với hàm độ đối với xứng khi các mức lượng tử được chọn là điểm giữa của các khoảng lượng tử. Trong trường hợp này, các mức lượng tử tương ứng với các vùng lượng tử hoá đầu tiên và cuối cùng được chọn nằm cách hai bên lượng tử ngoài cùng một khoảng $\Delta/2$. Điều đó nghĩa là: ***(i)*** nếu số các mức lượng tử là chẵn, thì các biên lượng tử hoá là 0, $\pm\Delta$, $\pm 2\Delta$,...,$\pm(N/2\text{-}1)\Delta$ và các mức lượng tử là $\pm\Delta/2$, $\pm 3\Delta/2$,...,$\pm(N\text{-}1)\Delta/2$; ***(ii)*** Nếu số các mức lượng tử là lẻ, thì các biên lượng tử hoá là $\pm\Delta/2, \pm 3\Delta/2$,...,$\pm(N/2\text{-}1)\Delta$ và các mức lượng tử là $0, \pm\Delta, \pm 2\Delta$,....,$\pm(N\text{-}1)\Delta/2$.

```
function dist=NVD3_uq_mdpnt(funfcn,b,n,delta,tol,p1,p2,p3)
% N7
%UQ_MDPNT   trả lại phân bố của một bộ lượng tử hóa đều với các điểm lượng tử hóa
%           được đặt tại các điểm giữa

%DIST=UQ_MDPNT(FUNFCN,B,N,DELTA,TOL,P1,P2,P3).
% Trong đó:
%       funfcn = hàm mật độ nguồn tin được cho trong một file.m với 3 tham số
%                p1,p2,p3. Hàm mật độ được giả sử là hàm chẵn.
%       [-b,b] = sự hỗ trợ của hàm mật độ nguồn tin.
%       n      = số mức
%       delta  = kích cỡ (độ rộng) mức.
%       p1,p2,p3 = các tham số của hàm đầu vào.
%       dist     = lượng méo.
%       tol      = sai số tương đối.

if (2*b<delta*(n-1))
  error('Quá nhiều mức cho dải này.'); return
end
args   = [];
for j=1:nargin-5
  args = [args,',p',int2str(j)];
```

```
end
args  = [args,')'];
a(1)  = -b;
a(n+1)= b;
a(2)  = -(n/2-1)*delta;
y(1)  = a(2)-delta/2;
for i=3:n
  a(i)   = a(i-1)+delta;
  y(i-1) = a(i)-delta/2;
end
y(n) = a(n)+delta;
dist = 0;
for i=1:n
  newfun = 'x_a2_fnct' ;
  dist   = dist+eval(['quad(newfun,a(i),a(i+1),tol,[],funfcn,', num2str(y(i)),
args]);
end
```

Bài tập 3.7

[Bộ lượng tử hoá với các mức được đặt tại các điểm giữa]

Hãy tìm lượng méo khi sử dụng bộ lượng tử hoá đều để lượng tử hoá biến ngẫu nhiên Gausơ có trung bình 0 và phương sai bằng 1. Số các mức lượng tử hoá là 11, và độ dài của mỗi vùng lượng tử hoá là 1.

Lời giải

Trong hàm **NVD3_uq_mdpnt.m**, ta thay '***normal***' (lưu ý rằng, tên hàm cần phải được thay bằng '**normal**' với các dấu nháy) cho tên hàm mật độ, p_1=0 và p_2=1 đối với các tham số của hàm mật độ, n=11 cho số mức lượng tử, và Δ=1 cho độ dài của các mức lượng tử. Đối với tham số b, nó xác định dải tính của hàm mật độ, ta sử dụng giá trị b=10.p_2=10 và chọn độ phân giải ('tol' tolerance) là 0,001. Lượng méo tính được là 0,0833. Cụ thể, được thực thi bởi file **NVD3_sim37e.m** dưới đây

```
% function y = NVD3_sim37e
clc;
clear all;
close all;
%=======================
p1      = 0;
p2      = 1;
delta   = 1;
n       = 11;
tol     = 0.01;
b       = 10*p2;
data_input = [p1 p2 delta n b tol];
%=======================
result1 = NVD3_uq_mdpnt('normal',b,n,delta,tol,p1,p2);
%=======================
display('==== XAC DINH MEO BO LUONG TU VOI CAC MUC LUONG TU DAT O GIUA - BAI TAP
4.7 ====');
display('---DU LIEU VAO: [p1 p2  n   delta   b   tol] tuong ung nhu sau');
disp(data_input);
display('==>KET QUA: Luong meo');
disp(result1)
```

Kết quả chạy chương trình này là:

```
==== XAC DINH MEO BO LUONG TU VOI CAC MUC LUONG TU DAT O GIUA - BAI TAP 3.7 ====
```

```
---DU LIEU VAO: [p1 p2  n   delta   b   tol] tuong ung nhu sau
     0   1.0000   1.0000  11.0000   10.0000   0.0100
==>KET QUA: Luong meo
   0.0833
```

❖ Lượng tử hoá không đều

Trong lượng tử hoá không đều, yêu cầu về các vùng lượng tử hoá có độ dài như nhau, ngoại trừ các miền đầu và cuối, được xem nhẹ, và mỗi vùng lượng tử hóa có thể có một độ dài bất kỳ. Do trong trường hợp này việc tối ưu hoá được thực hiện với những điều kiện giảm nhẹ hơn, kết quả là có sự vượt trội rõ ràng hơn so với lượng tử hoá đều. Các điều kiện tối ưu đối khi này, được gọi là các điều kiện Lloyd-Max, được biểu diễn là

$$\begin{cases} \hat{x}_i = \dfrac{\int_{a_{i-1}}^{a_i} x f_X(x)dx}{\int_{a_{i-1}}^{a_i} f_X(x)dx} \\ a_i = \dfrac{\left(\hat{x}_{i-1} - \hat{x}_i\right)}{2} \end{cases} \qquad (3.3.7)$$

Từ những phương trình này, ta kết luận rằng các mức lượng tử hoá tối ưu là các trọng tâm của các vùng lượng tử hoá và các biên tối ưu giữa các vùng lượng tử hóa là các điểm giữa của các mức lượng tử hoá. Để có được lời giải đối với các phương trình Lloyd-Max, ta bắt đầu với tập các mức lượng tử hoá $\hat{x}_i$. Từ tập này, ta có thể tìm được một tập các biên vùng lượng tử hoá a_i. Từ tập các a_i này, ta có thể nhận được tập mới các mức lượng tử hóa. Quá trình này được tiếp tục cho đến khi việc cải thiện về méo từ bước lặp này sang bước lặp khác không thể nhận ra được. Thuật toán này bảo đảm hội tụ tới một trị tối thiểu cục bộ, nhưng ở dạng tổng quát thì không đảm bảo để đạt được giá trị tối thiểu toàn bộ.

Thủ tục thiết kế bộ lượng tử hoá tối ưu được trình bày trong file **NVD3_lloydmax.m** dưới đây.

```
function [a,y,dist]=NVD3_lloydmax(funfcn,b,n,tol,p1,p2,p3)
% N8
%LLOYDMAX  trả lại bộ lượng tử hóa Lloyd-Max và sai số lượng tử trung bình bình
%          phương đối với phân bố đối xứng

% [A,Y,DIST]=LLOYDMAX(FUNFCN,B,N,TOL,P1,P2,P3).
%    funfcn = hàm mật độ được cho ở một file.m. Nó có thết phụ thuộc tới 3 tham số
%             p1,p2,p3.
%    a      = vector tạo ra các biên của các vùng lượng tử
%    [-b,b]   lấy xấp xỉ sự hỗ trợ của hàm mật độ.
%    n      = số vùng lượng tử hóa
%    y      = các mức lượng tử hóa
%    p1,p2,p3 = Các tham số của funfcn.
%    tol      = Sai số tương đối

args  = [];
for j=1:nargin-4
  args =[args,',p',int2str(j)];
end
args   = [args,')'];
v      = eval(['variance(funfcn,-b,b,tol',args]);
a(1)   = -b;
d      = 2*b/n;
for i=2:n
  a(i) = a(i-1)+d;
```

```
end
a(n+1) = b;
dist   = v;

[y,newdist] = eval(['NVD3_mse_dist(funfcn,a,tol',args]);

while(newdist<0.99*dist),
  for i=2:n
    a(i) = (y(i-1)+y(i))/2;
  end
  dist   = newdist;
  [y,newdist] = eval(['NVD3_mse_dist(funfcn,a,tol',args]);
end
```

Bài tập 3.8.
[Thiết kế bộ lượng tử hoá Lloyd-Max]

Hãy thiết kế một bộ lượng tử hoá Lloyd-Max 10 mức cho nguồn Gausơ có trung bình không và phương sai bằng 1.

Lời giải

Sử dụng các giá trị b=10, n=10, tol=0.01, p_1=0, p_2=1 trong **NVD3_lloydmax.m**, ta nhận được các biên lượng tử hoá và véctơ mức lượng tử hoá a và y là

$a = \pm 10, \pm 2,6, \pm 1,51, \pm 0,98, \pm 0,48, 0$

$y = \pm 2,52, \pm 1,78, \pm 1,22, \pm 0,72, \pm 0,24$

và lượng méo tính được là 0,02. Các giá trị này là các giá trị xấp xỉ rất gần với các giá trị tối ưu đã được cho trong bảng [2]. Cụ thể, được thực thi bởi file **NVD3_sim38e.m** dưới đây

```
% function y = NVD3_sim38e
clc;
clear all;
close all;
%======================
n           = 10;
tol         = 0.01;
p1          = 0;
p2          = 1;
b           = 10*p2;
data_input  = [n tol p1 p2 b];
%======================
[a,y,dist]  = NVD3_lloydmax('normal',b,n,tol,p1,p2);
%======================
display('DU LIEU VAO la vector cac tham so [n tol p1 p2 b] tuong ung nhu sau:');
disp(data_input)
display('=>KET QUA:');
display('   Vector: a la');
disp(a)
display('   Vector: Y la');
disp(y)
display('   MEO LUONG TU: dist la');
disp(dist)
```

Kết quả chạy chương trình là:

```
DU LIEU VAO la vector cac tham so [n tol p1 p2 b] tuong ung nhu sau:
  10.0000    0.0100         0    1.0000   10.0000
=>KET QUA:
   Vector: a la
 -10.0000   -2.1569   -1.5075   -0.9753   -0.4813   -0.0000    0.4813    0.9753    1.5075    2.1569   10.0000
   Vector: Y la
   -2.5092   -1.7701   -1.2126   -0.7137   -0.2361    0.2361    0.7137    1.2126    1.7701    2.5092
   MEO LUONG TU: dist la
    0.0248
```