Chương 7
TẠO VÀ XỬ LÝ TÍN HIỆU NGẪU NHIÊN

7.1. Mở đầu

Ta đã xét các tín hiệu tất định trong mô phỏng. Trong tất cả các hệ thống truyền thông thực tế, các ảnh hưởng ngẫu nhiên như tạp âm kênh, nhiễu, pha đinh, đều làm suy thoái tín hiệu mang tin. Để mô phỏng chính xác những hệ thống này ở mức dạng sóng cần phải mô hình hóa chính xác các ảnh hưởng ngẫu nhiên. Vì vậy, cần có các thuật toán để tạo các ảnh hưởng ngẫu nhiên này. Khối cơ bản nền tảng là bộ tạo số ngẫu nhiên. Tồn tại nhiều tài liệu được viết cho các bộ tạo số ngẫu nhiên, chương này chỉ tập trung vào việc sử dụng các bộ tạo số ngẫu nhiên trong mô phỏng. Vì vậy, ta sẽ chỉ tập trung nghiên cứu việc tạo các phiên bản mẫu của các dạng sóng ngẫu nhiên (tín hiệu, nhiễu, tạp âm,...) để sử dụng trong các chương trình mô phỏng. Trong môi trường mô phỏng, tất cả các quá trình ngẫu nhiên đều được biểu diễn bằng các chuỗi của các biến ngẫu nhiên. Tạo và kiểm tra các chuỗi ngẫu nhiên này là chủ đề của chương này. Nhiều ngôn ngữ lập trình hữu hiệu để triển khai các chương trình mô phỏng (như Matlab) có sẵn các bộ tạo số ngẫu nhiên trong thư viện. Hiểu những khái niệm về tạo số ngẫu nhiên, giúp hiểu toàn bộ chương trình mô phỏng.

Nói một cách chính xác, các bộ tạo số ngẫu nhiên không thể tạo ra các số hoàn toàn ngẫu nhiên nhưng nó tạo ra các chuỗi số *xuất hiện ngẫu nhiên* trong khoảng thời gian quan sát (mô phỏng), sao cho chúng có thể được dùng để *xấp xỉ* hàm mẫu của một quá trình ngẫu nhiên trong chương trình mô phỏng cụ thể. Do "*xuất hiện ngẫu nhiên*" nghĩa là các chuỗi số được tạo ra trong khoảng thời gian mô phỏng sẽ có các đặc tính cần thiết để mô hình hóa quá trình ngẫu nhiên ở mức độ chính xác cần thiết cho một ứng dụng cụ thể. Những chuỗi số như vậy được coi là các chuỗi số *giả ngẫu nhiên*, mặc dù chúng là tất định nhưng chúng xuất hiện ngẫu nhiên khi được dùng trong ứng dụng cụ thể. Yêu cầu về độ chính xác phụ thuộc vào ứng dụng đó. Ví dụ, nếu ta phải tạo dạng sóng để biểu diễn tạp âm đầu vào PLL, thì yêu cầu độ chính xác để mô hình hóa dạng sóng tạp âm khi SNR đầu vào 50 dB là cao hơn trường hợp 8 dB. Yêu cầu độ chính xác để mô hình hóa thành phần tạp âm trong hệ thống truyền tham số cho trường hợp xác suất lỗi bit 10^{-7} cao hơn so với trường hợp xác suất lỗi bit 10^{-3}.

Trước tiên ta xét việc tạo các hàm mẫu của một quá trình ngẫu nhiên. Ngiên cứu khái niệm dừng trong môi trường mô phỏng. Sau đó, ta xét vắn tắt các mô hình mô phỏng cho các bộ điều chế số. Vì vậy, chương 7 sẽ tập trung chủ yếu vào các vấn đề sau đây:

- Tạo số ngẫu nhiên *không* tương quan phân bố đều trong khoảng (0,1).

- Ánh xạ số ngẫu nhiên *không* tương quan và phân bố đều thành số ngẫu nhiên không tương quan và có hàm mật độ xác suất pdf tuỳ ý (mong muốn).

- Tạo số ngẫu nhiên *không* tương quan và có pdf phân bố Gausơ.

- Tạo số ngẫu nhiên *tương quan* và có pdf phân bố Gausơ.

- Tạo số ngẫu nhiên *tương quan* và có pdf tuỳ ý (mong muốn).

- Sau đó xét vắt tắt việc tạo chuỗi giả ngẫu nhiên PN và một ứng dụng.

Lưu ý rằng: Do tính ngẫu nhiên của tín hiệu và hệ thống truyền thông, nên để triển khai mô phỏng chính xác nhất có thể và hiệu quả cần phải hiểu sâu rộng về các khái niệm như: quá trình ngẫu nhiên, biến ngẫu nhiên, hàm mẫu quá trình ngẫu nhiên, con số thực,... cũng như các tham số đặc trưng... Tồn tại rất nhiều tài liệu cho chủ đề này, để tiện bạn đọc cũng có thể tham khảo Phụ lục **7B1 và 7B2** nếu cần, ở đó trình bày súc tích, cô đọng một cách căn bản về các biến ngẫu nhiên, quá trình ngẫu nhiên, phân loại cũng như nhiều ứng dụng minh họa. Theo quan điểm của tác giả nên đọc phần phụ lục này đặc biệt đối với sinh viên.

7.2. Quá trình dừng và Ergodic

Các quan hệ giữa quá trình ngẫu nhiên, biến ngẫu nhiên, hàm mẫu và số thực (phức) được cho ở hình 7.1. *Ta nên tham khảo Phụ lục 7B1 và 7B2*.

Nếu cho Q là không gian mẫu và gán cho mỗi kết cục $\xi = \xi_i \in Q$ của một thí nghiệm ngẫu nhiên một hàm thời gian $x(t,\xi_i)$ theo một quy tắc, thì với từng $\xi_i \in Q$, hàm $x(t,\xi_i)$ ký hiệu cho ánh xạ *IR* đến *IR* (hay *C*) theo:

$$x(.,\xi_i) : IR \to IR (hay C), \quad t \mapsto x(t,\xi_i)$$

Các hàm $x(t,\xi_i)$ phụ thuộc vào thời gian được gọi là các thực hiện hay các hàm mẫu. Một quá trình ngẫu nhiên $x(t,\xi)$ là một họ (hay một tập hợp các hàm mẫu $x(t,\xi_i)$, nghĩa là $x(t,\xi) = \{x(t,\xi_i) | \xi_i \in Q\} = \{x(t,\xi_1), x(t,\xi_2), ...\}$. Mặt khác, tại một thời điểm $t = t_0 \in IR$, quá trình ngẫu nhiên $x(t_0,\xi)$ chỉ phụ thuộc vào kết cục ξ, là một biến ngẫu nhiên. Vì thế với $t_0 \in IR$, $x(t_0,\xi)$ ký hiệu cho ánh xạ từ Q vào *IR* (hay *C*) theo quy tắc:

$$x(t_0,.) : Q \to IR (hay C), \quad s \mapsto x(t_0,\xi)$$

Hàm mật độ xác suất của biến ngẫu nhiên $x(t_0,\xi)$ được xác định bởi sự xuất hiện của các kết cục. Vì quá trình ngẫu nhiên là ***một hàm hai biến:*** $t \in IR$ và $\xi \in Q$, nên ký hiệu chính xác là $x(t,\xi)$. Tuy nhiên trong thực tế để đơn giản về ký hiệu thường viết $x(t)$.

Vì vậy, có thể kết luận rằng một quá trình ngẫu nhiên được thể hiện như sau:

1. Nếu t là một biến và ξ là một biến ngẫu nhiên, thì $x(t)$ thể hiện một họ hay một tập hợp các hàm mẫu $x(t,\xi)$.

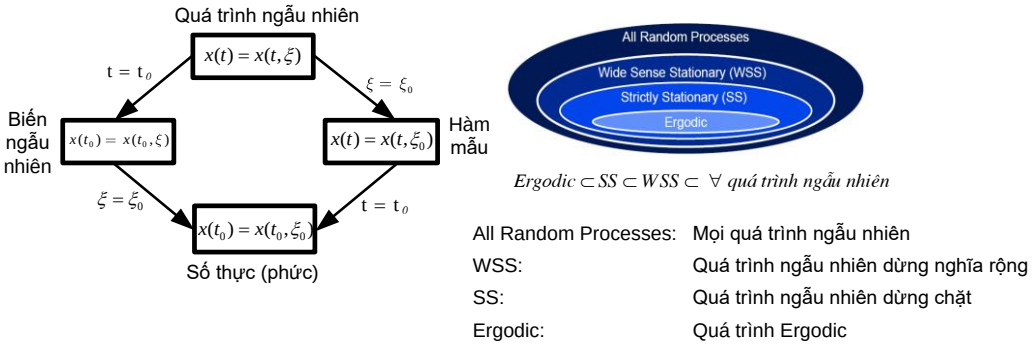
2. Nếu t là một biến và $\xi = \xi_0$ là hằng số, thì $x(t) = x(t,\xi_0)$ là một thực hiện hay một hàm mẫu của quá trình ngẫu nhiên.

3. Nếu $t = t_0$ là một hằng số và ξ là một biến ngẫu nhiên thì $x(t_0)$ cũng là một biến ngẫu nhiên.

4. Nếu cả $t = t_0$ và $\xi = \xi_0$ đều hằng số, thì $x(t_0)$ là một số thực (hoặc phức).

Quan hệ giữa các quá trình ngẫu nhiên, các biến ngẫu nhiên, các hàm mẫu và các số thực (phức) được cho ở hình 7.1.

Khi mô phỏng hệ thống truyền thông, các hàm mẫu được tạo ra để biểu diễn tín hiệu, tạp âm, nhiễu thường được coi là ergodic. Cần có giả định này vì ta xử lý các mẫu dạng sóng đi qua hệ thống một cách liên tiếp và, tại mỗi điểm trong hệ thống có một dạng sóng (hàm mẫu). Ta giả định rằng dạng sóng được xử lý bằng mô phỏng là bộ phận điển hình của toàn bộ được định nghĩa bởi mô hình thống kê cơ bản. Các đại lượng thống kê khác nhau như: các moment, SNR, BER sẽ được tính toán như là các đại lượng trung bình thời gian. Khi so sánh các kết quả mô phỏng với các kết quả lý thuyết tương ứng, thường giả định các trung bình thời gian được tính bởi mô phỏng là tương đương với các trung bình toàn bộ. Kết quả là, có một giả định rằng các quá trình ngẫu nhiên cơ bản là các quá trình ***ergodic***.



Hình 7.1: Quan hệ giữa các quá trình ngẫu nhiên, các biến ngẫu nhiên, các hàm mẫu và các số thực (phức)

Các quá trình ergodic luôn luôn là các quá trình dừng. Vì vậy, các hàm mẫu được tạo ra trong mô phỏng luôn luôn được coi là các thành phần của một quá trình ngẫu nhiên dừng. Từ lý thuyết quá trình ngẫu nhiên cơ bản định nghĩa về vấn đề dừng: nghĩa là tất cả các đại lượng thống kê đều độc lập (không phụ thuộc) vào gốc thời gian. Để được rõ, ta xét một số ví dụ đơn giản sau:

Ví dụ 7.1.1: *Trường hợp thứ nhất*: Giả sử các hàm mẫu của một quá trình ngẫu nhiên được định nghĩa là:

$$x(t,\xi_i) = A\cos(2\pi ft + \phi_i) \tag{7.1}$$

Trong đó ξ_i là một kết cục trong không gian mẫu của thí nghiệm ngẫu nhiên cơ bản, và mỗi kết cục ξ_i được sắp xếp thành một pha ϕ_i. Ta cũng coi thí nghiệm ngẫu nhiên cơ bản bao gồm việc lấy ra một con số từ một bộ tạo số ngẫu nhiên phân bố đều. Kết quả của việc rút ra này là kết cục $\xi_i = u_i$, trong đó u_i được phân bố đều trong khoảng (0,1). Sau đó u_i được ánh xạ

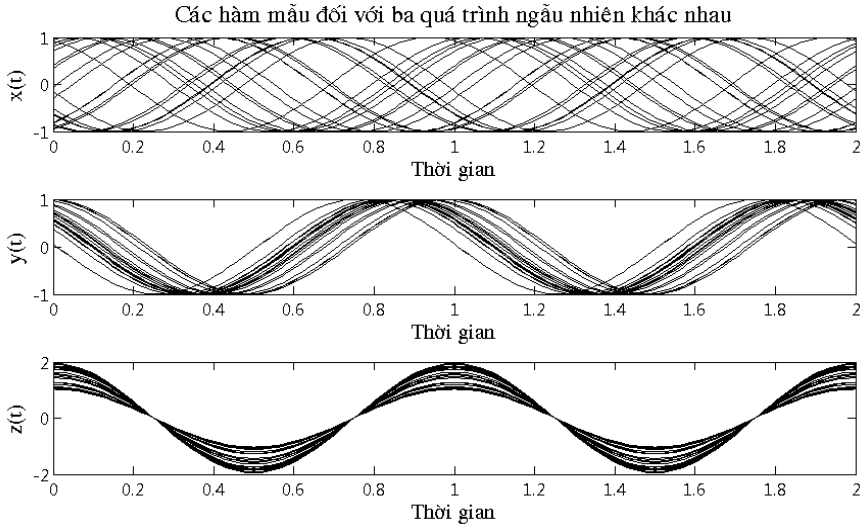
lên pha $\phi_i = ku_i$. Khi A và f không đổi, thì giá trị của ϕ_i xác định dạng sóng. Trong ví dụ này cần lưu ý đến 2 giá trị của k là $k = 2\pi$ và $k = \frac{\pi}{2}$, nếu $k = 2\pi$ thì pha được phân bố đều trong khoảng $(0,2\pi)$, nếu $k = \frac{\pi}{2}$ thì pha được phân bố đều trong khoảng $(0,\pi/2)$.

Trường hợp thứ hai: Giả sử rằng quá trình ngẫu nhiên được mô tả bằng biểu thức:

$$x(t,\xi_i) = A(1+u_i).cos(2\pi ft) \tag{7.2}$$

Trong trường hợp này biên độ được phân bố đều trong khoảng *(A, 2A)*.

Chương trình Matlab **NVD7_ sinewave.m** dưới đây (có ở Phụ lục 7A) tạo ra 3 tập hàm mẫu của một quá trình ngẫu nhiên. Tập dạng sóng đầu tiên được ký hiệu là *x(t)* tương ứng với (7.1) khi $k = 2\pi$. Tập dạng sóng thứ 2 được ký hiệu là *y(t)* tương ứng với (7.1) khi $k = \pi/2$. Tập dạng sóng thứ 3 được ký hiệu là *z(t)* được định nghĩa bởi (7.2). Tất cả đều có $A = 1$ và $f = 1$. Hai giây (thời gian mô phỏng) dữ liệu và 20 hàm mẫu được tạo ra cho mỗi mô phỏng. Kết quả chạy chương trình được minh họa ở hình 7.2(a).



Hình 7.2(a): Các hàm mẫu cho ba quá trình ngẫu nhiên khác nhau

```
% File: NVD7_sinewave.m
f = 1; % Tần số của hình sin
fs = 100; % Tần số lấy mẫu
t = (0:200)/fs; % vectơ thời gian
for i = 1:20
 x(:,i) = cos(2*pi*f*t+rand(1)*2*pi)';
 y(:,i) = cos(2*pi*f*t+rand(1)*pi/2)';
 z(:,i) = (1+rand(1))*cos(2*pi*f*t)';
end
```

```
subplot(3,1,1); plot(t,x,'k'); ylabel('x(t)');
subplot(3,1,2); plot(t,y,'k'); ylabel('y(t)');
subplot(3,1,3); plot(t,z,'k'); ylabel('z(t)');
```

Các trung bình thời gian của tất cả các hàm mẫu *x(t), y(t), và z(t)* đều bằng 0. Ta dễ dàng kiểm chứng bằng cách lấy trung bình toàn bộ *x(t)* xấp xỉ 0 khi được tính tại một số lượng lớn các điểm $t_i,\ 0 \le t_i \le 2$. *Các trung bình thời gian sẽ hội tụ về 0 khi số các hàm mẫu tiến đến* ∞. Tuy nhiên, đối với *y(t)* thì trung bình toàn bộ xấp xỉ 1 khi *t* lân cận 0,375; 1,875, xấp xỉ -1 khi *t* lân cận 0,375; 1,375 và xấp xỉ 0 khi *t* lân cận 0,125; 0,625; 1,125; 1,625. Đây là một ví dụ về quá trình dừng *tuần hoàn (cyclostationary process)*, trong đó các moment là tuần hoàn (*periodic*). Để được rõ hơn, bạn đọc nên tham khảo phần phụ lục 7B2, ở đó trình bày vắt tắt, súc tích, và đặc biệt trực quan hóa bằng nhiều hình vẽ.

Các hàm mẫu *z(t)* cũng là các hàm mẫu từ quá trình dừng tuần hoàn. Lưu ý rằng, việc lấy mẫu quá trình đó tại *t = 0,5k* tạo ra một biến ngẫu nhiên có trung bình xấp xỉ +1,5 khi *k* chẵn, và xấp xỉ -1,5 khi *k* lẻ.

Trong ví dụ 7.1 ta đã sử dụng bộ tạo số ngẫu nhiên đều **rand** có sẵn trong thư viện Matlab. Trong ví dụ 7.2 ta minh hoạ việc sử dụng bộ tạo số ngẫu nhiên để mô hình hóa bộ điều chế số. Trong phần sau sẽ khảo sát chi tiết các thuật toán thực hiện các bộ tạo số đều.

Ví dụ 7.2: Ta thường cần có các mô hình của các bộ điều chế số. Khối cơ bản cho các bộ điều chế này là hàm **NVD7_random_binary.m**, nó tạo dạng sóng nhị phân có các giá trị +1 và -1. Số các bit được tạo ra và số các mẫu trên bit là các đối số của hàm này. Để dùng hàm này trong các chương trình mô phỏng, phải nhập giá trị cho hai tham số là: số bit (*nbit*) và số mẫu trên bit (*nsamples*) theo dòng lệnh sau:

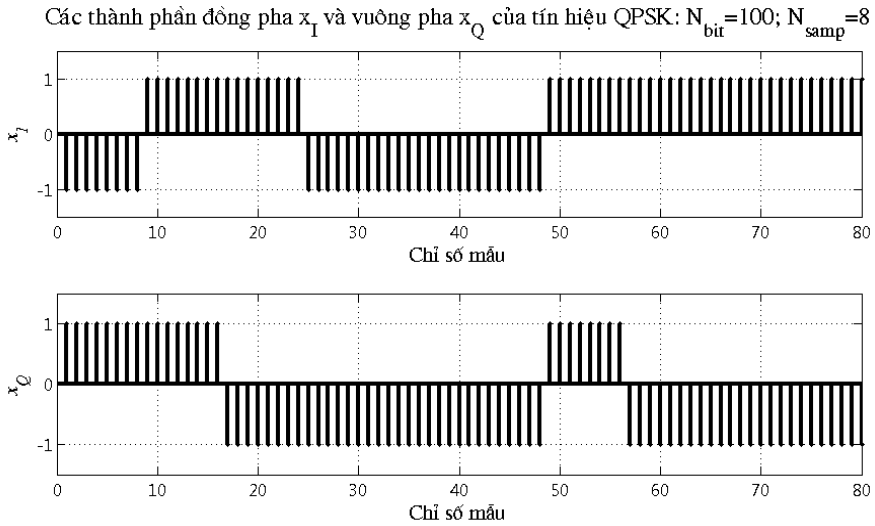
$$\left[\underbrace{x,\ bits}_{\text{kết quả mô phỏng}}\right] = \underbrace{NVD7\_random\_binary}_{\text{Tên hàm cũng là tên file}}\left(\underbrace{nbit,\ nsamples}_{\text{Các tham số đầu vào (hay đối số) của hàm}}\right)$$

```
function [x, bits] = NVD7_random_binary(nbits,nsamples)
% Hàm này tạo dạng sóng nhị phân ngẫu nhiên dài nbits được lấy mẫu
  tại tốc độ % nsamples/bit.
x = zeros(1,nbits*nsamples);
bits = round(rand(1,nbits));
for m = 1:nbits
    for n = 1:nsamples
        index = (m-1)*nsamples + n;
        x(1,index) = (-1)^bits(m);
 end
end
```

Hàm **NVD7_random_binary.m** được dùng để mô phỏng một số bộ điều chế số. Ví dụ dùng câu lệnh sau để mô phỏng bộ điều chế QPSK.

$$x = NVD7\_random\_binary(nbit, nsamples) + i * NVD7\_random\_binary(nbit, nsamples);$$

Chương trình Matlab **NVD7_ sim2.m** tạo tín hiệu QPSK cho 10 bit với tần số lấy mẫu là 8 mẫu trên bit. Chạy chương trình tạo ra tín hiệu QPSK có các thành phần đồng pha và vuông pha được minh hoạ trong hình 7.2 (b). Lưu ý rằng, ở đây xét tín QPSK băng tần cơ sở.



Hình 7.2(b): Các thành phần đồng pha và vuông pha của tín hiệu QPSK

```
% File: NVD7_sim2
nbits = 10;
nsamples = 8;
x  =  NVD7_random_binary(nbits,nsamples)+i*NVD7_random_binary(nbits,
     nsamples);
xd = real(x);
xq = imag(x);
% ==================
% Hiển thị kết quả
% ================
subplot(2,1,1)
stem(xd,'.');
grid;
axis([0 80 -1.5 1.5]);
xlabel('Chỉ số mẫu','fontname','.vntime','fontsize',12);
ylabel('X_I','fontname','.vntime','fontsize',16)
subplot(2,1,2)
stem(xq,'.');
grid;
axis([0 80 -1.5 1.5]);
```

```
xlabel('Chỉ số mẫu','fontname','.vntime','fontsize',12);
ylabel('X_Q','fontname','.vntime','fontsize',16);
```

7.3. Bộ tạo số ngẫu nhiên phân bố đều

Biến ngẫu nhiên có hàm mật độ xác suất phân bố đều dễ dàng chuyển thành biến ngẫu nhiên có pdf mong muốn khác (không phải là phân bố đều). Vì vậy, bước đầu tiên trong việc tạo biến ngẫu nhiên có pdf cụ thể là tạo biến ngẫu nhiên phân bố đều trong khoảng (0,1) bằng cách: trước hết tạo ra một chuỗi số (nguyên) giữa 0 và M và sau đó chia mỗi phần tử của chuỗi cho M. Kỹ thuật thông dụng nhất để thực hiện các bộ tạo số ngẫu nhiên được biết là đồng dư tuyến tính.

7.3.1. Đồng dư tuyến tính

Bộ tạo đồng dư tuyến tính (LCG) được định nghĩa bằng phép toán:

$$x_{i+1} = [ax_i + c] mod(m) \tag{7.3}$$

Trong đó *a* được coi là số nhân, *c* được coi là số gia, và tham số *m* được gọi là modulus. Tất nhiên, đây là thuật toán tuần tự tất định trong đó các giá trị kế tiếp của *x* được tạo ra lần lượt. Giá trị khởi đầu của *x* ký hiệu là x_0 được gọi là số gốc của bộ tạo số. Giả thiết rằng x_0, *a*, *c*, *m* là các số nguyên, thì tất cả các số được tạo ra bởi LCG sẽ là những số nguyên. Tính chất mong muốn tại đầu ra bộ tạo số là nó có chu kỳ dài sao cho tạo được nhiều số nguyên nhất trong chuỗi đầu ra trước khi chuỗi được lặp lại. Với giá trị của *m* cho trước khi chu kỳ được cực đại hóa ta nói rằng bộ tạo số là *tuần hoàn đầy đủ (full period)*. Ngoài ra, khi ứng dụng vào một chương trình mô phỏng cụ thể đặt các yêu cầu khác lên LCG. Ví dụ ta thường cần có các mẫu x_i và x_{i+1} không tương quan nhau. Hơn nữa, tùy vào ứng dụng cụ thể mà đầu ra LCG phải qua các phép thử thống kê khác. LCG có thể có nhiều dạng khác nhau. Trong phần này, chỉ xét những thuật toán phổ biến nhất.

Kỹ thuật A: Thuật toán đồng dư hỗn hợp

Thuật toán đồng dư tổng quát nhất là thuật toán đồng dư "***hỗn hợp***" trong đó $c \neq 0$. Sở dĩ gọi là hỗn hợp vì cả phép cộng và phép nhân đều được dùng để tính x_{i+1}. Thuật toán tuyến tính hỗn hợp có dạng được cho bởi (7.3):

$$x_{i+1} = [ax_i + c] mod(m) \tag{7.4}$$

Khi $c \neq 0$ thì bộ tạo số có chu kỳ lớn nhất là *m*. Đạt được chu kỳ này nếu và chỉ nếu:

- Gia số *c* là nguyên tố tương đối đối với *m*. Nói cách khác *c* và *m* không có các thừa số nguyên tố chung.
- *a-1* là bội số của *p*, trong đó *p* thể hiện các thừa số nguyên tố của mô đun *m*.
- *a-1* là bội số của 4 nếu *m* là bội số của *m*.

Ví dụ 7.3: Thiết kế một bộ tạo số đồng dư hỗn hợp có chu kỳ *m* = 5000. Vì:

$$5000 = \left(2^3\right)\left(5^4\right) \tag{7.5}$$

Ta có thể đảm bảo rằng m và c là nguyên tố tương đối bằng cách đặt c bằng tích của các số nguyên tố chứ không phải 2 và 5. Điều này thoả mãn tính chất thứ nhất. Một trong nhiều khả năng có thể được là đặt:

$$c = \left(3^2\right)\left(7^2\right) = 1323 \tag{7.6}$$

Bây giờ phải chọn giá trị của a. Tính chất thứ 2 được thoả mãn bằng cách đặt:

$$a - 1 = k_1 p_1 \tag{7.7}$$

$$a - 1 = k_2 p_2 \tag{7.8}$$

Trong đó $p_1 = 2$ và $p_2 = 5$ (các thừa số của m) và k_1 & k_2 là các số nguyên tuỳ ý.

Vì 4 là một thừa số của m = 5000 nên ta làm thoả mãn chấm thứ 3 bằng cách đặt:

$$a - 1 = 4k_3 \tag{7.9}$$

Trong đó k_3 là số nguyên tuỳ ý. Một sự lựa chọn rõ ràng cho a là đặt:

$$a - 1 = 4.k.p_1.p_2 \tag{7.10}$$

Hoặc:

$$a - 1 = 2.4.5.k = 40k \tag{7.11}$$

Trong đó k là một số nguyên. Với k = 6 thì a = 241. Vì vậy:

$$x_{i+1} = \left[241.x_i + 1323\right] mod\left(5000\right) \tag{7.12}$$

Là một bộ tạo số chu kỳ đầy đủ. Lưu ý rằng, còn có nhiều sự lựa chọn khác cho các tham số để tạo ra bộ tạo số chu kỳ đầy với m = 5000.

Ví dụ 7.4: Ví dụ này, ta chỉ ra rằng, LCG được thiết kế trong ví dụ 7.3 thực ra có chu kỳ m = 5000. Chương trình Matlab **NVD7_LCGperiod.m** dưới đây (có ở Phụ lục 7A), khi nhập số gốc và chương trình chạy cho đến khi số gốc lại xuất hiện. Nếu n số nguyên được tạo ra và $n>m$ mà không xuất hiện lại số gốc đó thì coi rằng bộ tạo số bị rơi vào một vòng trong đó một chuỗi ngắn được tạo ra một cách lặp.

Chạy chương trình Matlab, chương trình yêu cầu nhập các tham số sau:

```
>> NVD7_LCGperiod.m
Nhập a:          241
Nhập c:          1323
Nhập m:          5000
Nhập số gốc:     1
Kết quả chu kỳ là:  5000.
```

Ta thấy rằng chu kỳ thực sự là 5000 như được mong đợi.

Kỹ thuật B: Thuật toán nhân với mô đun nguyên tố

Bộ tạo số nhân được định nghĩa là:

$$x_{i+1} = [ax_i] mod(m) \tag{7.13}$$

Là thuật toán hỗn hợp khi $c = 0$. Lưu ý rằng, x_i không thể bằng không khi $c = 0$. Vì vậy, *chu kỳ đầy là m-1 chứ không phải là m* như trường hợp trước. Thuật toán nhân tạo ra chu kỳ đầy nếu:

- *m* là nguyên tố (thường yêu cầu *m* lớn)
- *a* là phần tử *nguyên thuỷ* mod(*m*)

Biết rằng, số nguyên tố là số chỉ chia hết cho 1 hoặc chính nó. Cần làm sáng tỏ tính chất thứ 2. Ta hiểu *a* là phần tử nguyên thuỷ *mod(m)* nếu $a^i - 1$ là bộ số của *m* với *i = m-1*, nhưng không có giá trị *i* nhỏ hơn. Nói cách khác *a* là phần tử nguyên thuỷ *mod(m)* nếu:

$$\frac{a^{m-1} - 1}{m} = k \tag{7.14}$$

Và:

$$\frac{a^i - 1}{m} \neq k, \quad i = 1, 2, 3..., m-2 \tag{7.15}$$

với số nguyên *k* tuỳ ý.

Kỹ thuật C: Thuật toán nhân với mô đun phi nguyên tố

Quan trọng nhất là trường hợp mođun *m* không phải là số nguyên tố mà *m* bằng luỹ thừa của 2. Nói cách khác:

$$x_{i+1} = [ax_i] mod(2^n), \text{ với n là số nguyên} \tag{7.16}$$

Trường hợp được định nghĩa bởi (7.16), thì chu kỳ lớn nhất là $2^n / 4 = 2^{n-2}$. Đạt được chu kỳ này nếu:

- Số nhân *a* là 3 hoặc 5 mod(8)
- Số gốc x_0 là lẻ.

Vì tích của 2 số lẻ là một số lẻ nên tất cả các giá trị được tạo ra bởi (7.16) là lẻ nếu x_0 là lẻ. Vì vậy, không có các giá trị chẵn của x_i được tạo ra, nó làm giảm chu kỳ đi hai lần. Các số nguyên lẻ được tạo ra bởi (7.16) được phân thành 2 tập, chỉ một tập được tạo ra từ số gốc cho trước. Điều này làm giảm chu kỳ bởi một hệ số 2 khác. Tập các số nguyên lẻ thực ra được tạo ra phụ thuộc vào việc chọn số gốc.

Ưu điểm sử dụng $m = 2^k$ là dùng tràn số nguyên để thực hiện phép toán *mod(m)*, giảm thời gian tính toán. Thực ra mong muốn giảm thời gian tính toán song chương trình không có khả năng chuyển tải dễ dàng.

7.3.2. Kiểm tra bộ tạo số ngẫu nhiên

Phần trước cho ta các công cụ tạo các số giả ngẫu nhiên phân bố đều trong khoảng (0,1). Tại đây ta mới xét chu kỳ của chuỗi số được tạo ra bởi LCG. Ta vừa muốn có chu kỳ dài vừa muốn có các thuộc tính đáp ứng cho một ứng dụng cụ thể. Ít nhất cần có chuỗi số tương quan *delta* (*trắng*). Các yêu cầu khác có thể cần đến cho ứng dụng đó.

Tồn tại nhiều thuật toán để kiểm tra tính ngẫu nhiên của chuỗi. Thuật toán thông dụng nhất là kiểm tra của Chi-square, kiểm tra của Kolômgorov-Simirnov, và kiểm tra phổ. Việc nghiên cứu các thuật toán này nằm ngoài phạm vi của cuốn sách này. Việc kiểm tra phổ được coi là hữu hiệu nhất. Vì vậy, ở đây mô tả vắt tắt việc kiểm tra phổ, được áp dụng vào thuật toán Wichman-Hill.

Nhiều ứng dụng sau đây, thuộc tính quan trọng nhất phải được đáp ứng là: các phần tử của một chuỗi cho trước là *độc lập, hoặc ít nhất là không tương quan*. Phần này, ta xét 2 kiểm tra đơn giản: biểu đồ tán xạ và kiểm tra Durbin-Watson. Sẽ thấy rõ, các tính chất của chuỗi cho trước áp dụng cho chuỗi đầy đủ (chu kỳ đầy đủ). Nếu chỉ sử dụng một phần của chuỗi thì các tính chất của chuỗi đầy đủ không áp dụng được.

Biểu đồ tán xạ

Biểu đồ tán xạ được minh hoạ tốt nhất bằng một ví dụ.

Ví dụ 7.5: *Biểu đồ tán xạ là vẽ x_{i+1} theo x_i* (nghĩa là *x_{i+1} là hàm của x_i*), nó thể hiện phép đo chất lượng bộ tạo số. Trong ví dụ này, ta xét 2 bộ tạo số ngẫu nhiên được định nghĩa bởi:

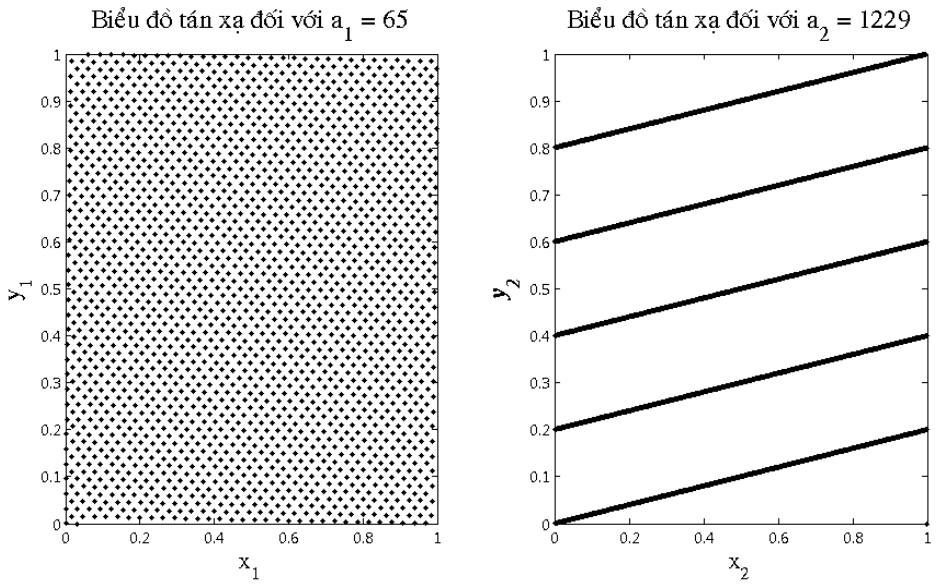
$$x_{i+1} = \left[65x_i + 1\right] mod\left(2048\right) \tag{7.17}$$

và:

$$x_{i+1} = \left[1229x_i + 1\right] mod\left(2048\right) \tag{7.18}$$

Ứng dụng chương trình **NVD7_LCGperiod.m** trong ví dụ 7.2 cho thấy cả 2 bộ tạo số này đều là bộ tạo số chu kỳ đầy đủ. Mã chương trình Matlab để tạo biểu đồ tán xạ cho các bộ tạo số ngẫu nhiên này được cho bởi file **NVD7_LCGSim1.m** (có trong Phụ lục 7A).

Kết quả chạy chương trình **NVD7_LCGSim1.m** nhận được biểu đồ tán xạ được mình hoạ ở hình 7.3. Nó tìm biểu đồ tán xạ trong đó tất cả các kết hợp tung độ x_{i+1} và toạ độ x_i xảy ra. Trường hợp này biểu đồ phân tán không có cấu trúc. Thấy rõ từ hình 7.3 với a = 65 tạo ra bộ tạo số có tính tương quan nối tiếp nhỏ hơn bộ tạo số có a = 1229.



Hình 7.3: Các biểu đồ tán xạ khi $a_1 = 65$ (bên trái) và $a_2 = 1.229$ (bên phải)

Kiểm tra Durbin-Watson

Kiểm tra Durbin-Watson cho tính độc lập được thực hiện bằng cách tính tham số Durbin:

$$D = \frac{\left(\frac{1}{N}\right)\sum_{n=2}^{N}\left(X_{[n]} - X_{[n-1]}\right)^2}{\left(\frac{1}{N}\right)\sum_{n=1}^{N} X^2_{[n]}} \tag{7.19}$$

Trong đó *X[n]* là biến ngẫu nhiên trung bình 0. Ta sẽ thấy rằng, các giá trị của D ở lân cận 2 tương quan giữa *X[n]* và *X[n-1]* là nhỏ.

Để minh hoạ các tính chất của kiểm tra Durbin-Watson, ta coi *X[n-1]* và *X[n]* là tương quan nhau, và *X[n]* là một quá trình *ergodic*. Để đơn giản hoá ký hiệu ta coi *N* đủ lớn sao cho $N\text{-}1 \approx N$ và viết:

$$D = \frac{E\left\{\left(X-Y\right)^2\right\}}{E\left\{X^2\right\}} = \frac{1}{\sigma_X^2} E\left\{\left(X\text{-}Y\right)^2\right\} \tag{7.20}$$

Trong đó *X* ký hiệu cho *X[n]*, *Y* ký hiệu cho *X[n-1]*, và *E{.}* ký hiệu cho phép kỳ vọng. Vì giả định *X[n]* và *X[n-1]* tương quan nhau nên ta đặt:

$$Y = \underbrace{\rho X + \sqrt{1-\rho^2}\,Z}_{\substack{X\&Z\text{: Không tương quan nhau} \\ \rho\text{: Hệ số tương quan X\&Y}}} \tag{7.21}$$

Trong đó *X* và *Z* không tương quan nhau và ρ là hệ số tương quan để quan hệ hóa giữa *X* và *Y*. Lưu ý rằng: *X*, *Y*, *Z* đều có cùng phương sai σ^2. Thay (7.21) vào (7.20) được:

$$D = \frac{1}{\sigma^2} E\left\{ (1-\rho)^2 X^2 - \underbrace{2(1-\rho)\sqrt{1-\rho^2}XZ}_{E[XZ]=0 \text{ vì } X \text{ \& } Z \text{ không tương quan}} + (1-\rho^2) Z^2 \right\} \tag{7.22}$$

Thành phần ở giữa bằng 0 vì X và Z không tương quan nhau và có trung bình 0. Vì X và Z có cùng phương sai:

$$D = \frac{(1-\rho)^2 \sigma^2 + (1-\rho^2)\sigma^2}{\sigma^2} = 2(1-\rho) \tag{7.23}$$

Vì $-1 \le \rho \le 1$ nên tham số Durbin D thay đổi giữa 0 và 4, với $D = 2$ nếu $\rho = 0$. Các giá trị $D < 2$ ngụ ý tương quan dương, trong khi $D > 2$ ngụ ý các giá trị âm của ρ. Mã chương trình Matlab **NVD7_durbin.m** thực hiện tính toán giá trị của tham số Durbin (có ở Phụ lục 7A).

Ví dụ 7.6: Trong ví dụ này ta tính giá trị D cho 2 bộ tạo tạp âm được xét trong ví dụ 7.5. Mã chương trình Matlab được cho bởi **NVD7_ LCDSim2.m** thực hiện bài toán này (có trong Phụ lục 7A).

Chạy chương trình nhận được kết quả:

>> NVD7_ LCDSim2

Giá trị của D_1 là 1,9925 và ρ_1 là 0,0037273.

Giá trị của D_2 là 1,6037 và ρ_2 là 0,19824.

Với $a_1 = 65$ tương quan xấp xỉ bằng 0, trong khi $a_2 = 1{,}229$ thì tương quan xấp xỉ bằng 0,2. Vì vậy từ kiểm tra Durbin-Watson cho thấy $a_1 = 65$ đưa ra các kết quả tốt hơn $a_2 = 1{,}229$. Kết quả này là phù hợp với các biểu đồ tán xạ được cho trong hình 7.3.

7.3.3. Các tiêu chuẩn nhỏ nhất

Nhiệm vụ chính là kiểm tra chất lượng LCG bằng cách chỉ ra rằng thỏa mãn được nhiều phép kiểm tra thống kê về tính ngẫu nhiên. Điều này đặc biệt đúng khi chuỗi được tạo ra là dài. Để giải quyết từng phần bài toán này, nhiều thuật toán được xem là các thuật toán *tiêu chuẩn nhỏ nhất*. Thuật toán tiêu chuẩn tối ưu là thuật toán mà

- Chu kỳ đầy đủ
- Thoả mãn tất cả các kiểm tra thống kê khả dụng về tính ngẫu nhiên
- Dễ dàng truyền tải từ máy tính này đến máy tính khác.

Thuật toán như vậy được biết đến và tư liệu hóa một cách chính đáng, nó trở thành *tiêu chuẩn nhỏ nhất*. Thuật toán có thể được dùng với sự tin tưởng mà không cần kiểm tra thêm. Nếu sử dụng thuật toán tiêu chuẩn nhỏ nhất thì không cần lo lắng về *tính chính xác*, nhưng phải đảm bảo rằng thuật toán phải được *thực thi* một cách chính xác trong môi trường tính toán đã cho. Xét dưới góc độ lập trình thì tất cả các số được tạo ra bởi thuật toán đều có thể biểu diễn một cách *duy nhất* (tính duy nhất của các con số).

Tiêu chuẩn nhỏ nhất Lewis, Goodman, và Miller

Tiêu chuẩn này được định nghĩa bởi:

$$x_{i+1} = [16807 x_i] \bmod (2147483647) \tag{7.24}$$

Trong đó *m* là số nguyên tố Mersenne (nếu $m = 2^k - 1$ thì *m* là số nguyên tố Mersenne) $2^{31} - 1$. Giá trị *m* này đầu tiên được đề xuất bởi Lehmer, người chịu trách nhiệm nhiều trong công việc cơ bản về LCG cách đây hơn nửa thế kỷ. Nó được dùng rộng rãi và được thực hiện dễ dàng đối với số học số nguyên trên các máy tính 32bit, và trong số học dấu phẩy động nếu phần định trị vượt quá 31 bit (từ chương 3 cho thấy, tiêu chuẩn dấu phẩy động IEEE gán 51 bit cho phần định trị).

Thuật toán Wichmann-Hill

Phần trên cho thấy rằng ta mong muốn có các bộ tạo số có chu kỳ dài. Một kỹ thuật hiệu quả để tạo dạng sóng có chu kỳ dài là lấy tổng một số dạng sóng tuần hoàn có chu kỳ khác nhau chút ít. Ví dụ, cos2π (1)t có chu kỳ 1 giây và cos2π (1,0001)t có chu kỳ 10.000/10.001 (nhỏ hơn 1 giây). Dạng sóng tổ hợp được viết ở dạng

$$x(t) = cos\left(2\pi \underbrace{(10000/10000)}_{1} t\right) + cos\left(2\pi \underbrace{(10001/10000)}_{1,0001} t\right) \tag{7.25}$$

Nó có chu kỳ 10000 giây (xấp xỉ 2,78 giờ). Trong chu kỳ này, thành phần đầu tiên trải qua 10000 chu kỳ và thành phần thứ 2 trải qua 10001 chu kỳ.

Cũng kỹ thuật đó có thể được áp dụng cho các LCG bằng cách kết hợp một vài bộ tạo số khác nhau, nhưng có các chu kỳ gần như bằng nhau. Thuật toán Wichmann-Hill có lẽ là một ví dụ minh họa tốt nhất về bộ tạo số số kết hợp. Có thể có nhiều thay đổi khác nhau của thuật toán Wichmann-Hill. Thuật toán ban đầu được mô tả trong bài báo của Coates sử dụng 3 bộ tạo số được định nghĩa là:

$$x_{i+1} = (171 x_i) \bmod (30269) \tag{7.26}$$

$$y_{i+1} = (170 y_i) \bmod (30307) \tag{7.27}$$

$$z_{i+1} = (172 z_i) \bmod (30323) \tag{7.28}$$

Thực sự là 3 bộ tạo chu kỳ đầy đủ. Kết hợp ba bộ tạo này cho ta đầu ra:

$$u_i = \left(\frac{x_i}{30269} + \frac{y_i}{30307} + \frac{z_i}{30323}\right) \bmod (1) \tag{7.29}$$

Thuật toán Wichmann-Hill tương đương với LCG nhân có số nhân *a* và mô đun *m* như sau:

$$a = 16.555.425.264.690 \tag{7.30}$$

$$m = 30269.30307.30323 \approx 2,7817.10^{13} \tag{7.31}$$

Vì *M* không phải là số nguyên tố nên chu kỳ ngắn hơn *m - 1*. Người ta đã chỉ ra rằng, chu kỳ xấp xỉ bằng 7.10^{12}, mặc dù nhỏ hơn *m* nhưng vẫn cực kỳ dài.

Thuật toán Wichmann-Hill, mặc dù khác chút ít về kiến trúc với tiêu chuẩn nhỏ nhất, nhưng được xem là bộ tạo số phân bố đều tiêu chuẩn nhỏ nhất, vì khi mở rộng kiểm tra đã cho thấy rằng nó thoả mãn tất cả các kiểm tra thống kê tiêu chuẩn, và dễ dàng truyền tải từ một máy này tới một máy khác.

7.3.4. Thực hiện trên Matlab

Trước phiên bản Matlab 5, bộ tạo số ngẫu nhiên phân bố đều **rand** trong thư viện Matlab là bộ tạo số tiêu chuẩn nhỏ nhất được định nghĩa bởi (7.24). Bộ tạo số ngẫu nhiên phân bố đều được dùng trong các phiên bản Matlab 5 và 6 dựa trên kỹ thuật được phát triển bởi Marsaglia. Bộ tạo số này có mục đích là tạo ra các số dấu phẩy động chứ không phải là các số nguyên tỉ lệ, được đề cập trong bài báo của Moler. MathWorks khẳng định rằng bộ tạo số này có chu kỳ lớn hơn 2^{1492} và "*khá chính xác*", tạo ra tất cả các số dấu phẩy động giữa *eps* và *1-eps/2*, trong đó hằng số *eps* của Matlab đã được đề cập ở chương 3 là 2^{-52}. Bộ tạo số mới chỉ sử dụng phép cộng và phép trừ. Vì không sử dụng phép nhân hoặc chia nên thuật toán thực hiện nhanh hơn nhiều LCG.

7.3.5. Số gốc và véc tơ

Vì các ví dụ mô phỏng được trình bày trong cuốn sách dựa trên Matlab nên ta xét vắt tắt cách mà Matlab xử lý các số gốc. Bộ tạo số ngẫu nhiên Matlab "cũ" (trước Matlab 5 và được định nghĩa bởi (7.14)) đã sử dụng một con số gốc. Số ngẫu nhiên mới sử dụng số gốc véc tơ được coi là trạng thái của bộ tạo số. Véc tơ này gồm 35 phần tử (32 số dấu phẩy động, hai số nguyên, và một cờ) xác định trạng thái của bộ tạo số. Với Matlab 5 hoặc mới hơn, dùng cả 2 bộ tạo số, trong đó mặc định là bộ tạo số ngẫu nhiên mới. Số ngẫu nhiên cũ được định nghĩa bởi (7.24) sử dụng lệnh RAND('seed',0) hoặc RAND('seed',J). Với mọi lệnh Matlab, ta nên nghiên cứu cẩn thận thông tin được cho bởi lệnh **help**. Ngoài ra, ta nên quan tâm đến các vấn đề sau (thuật ngữ số gốc **seed** được dùng để bao gồm cả các số gốc nguyên và các véc tơ trạng thái):

- Ta có thể sử dụng cả số gốc mặc định hoặc chỉ rõ số gốc.
- Đóng và mở lại Matlab sẽ thiết lập lại số gốc vào giá trị mặc định. Vì vậy, nếu thực hiện gọi bộ tạo số ngẫu nhiên N lần, sau đó đóng và mở lại Matlab và thực hiện gọi một bộ tạo số ngẫu nhiên nhiều hơn N lần, thì sẽ tạo ra cùng N số trong cả 2 trường hợp. Tính chất này là ưu điểm Matlab, vì nó cho phép tạo lại các chuỗi kết quả giống nhau, hữu ích cho các mục đích kiểm tra.
- Đồng hồ hệ thống được dùng để ngẫu nhiên hóa số gốc khởi đầu.
- Các số gốc được lưu trữ trong bộ đệm và không xuất hiện trên không gian làm việc của Matlab vì vậy khi thực hiện lệnh ***clear all*** không có ảnh hưởng.

7.4. Ánh xạ biến ngẫu nhiên phân bố đều thành biến ngẫu nhiên có pdf tuỳ ý

Nhiều phương pháp khác nhau đã được triển khai để ánh xạ biến ngẫu nhiên phân bố đều sang biến ngẫu nhiên có pdf không đều. Xảy ra 3 tình huống khác nhau cơ bản sau:

1. *Biết trước phân bố tích lũy đối với biến ngẫu nhiên đích ở dạng kín*. Ta sẽ thấy rằng nếu biết CDF của biến ngẫu nhiên đích ở dạng kín thì có thể sử dụng phương pháp biến đổi ngược (kỹ thuật rất đơn giản).
2. Biết trước pdf của biến ngẫu nhiên đích dạng kín nhưng CDF không được biết ở dạng kín. Biến ngẫu nhiên phân bố Gausơ thuộc loại này. Tồn tại một số phương pháp đặc biệt cho trường hợp này, và ngoài ra có thể dùng các phương pháp loại bỏ.
3. Cả pdf và CDF đều không được biết ở dạng kín. Thường gặp phải tình trạng này khi phải triển khai bộ tạo số ngẫu nhiên để phù hợp hóa với pdf của dữ liệu được tập hợp từ *thực nghiệm*.

Ta khảo sát các kỹ thuật được dùng trong mỗi trường hợp trên.

7.4.1. Phương pháp biến đổi ngược

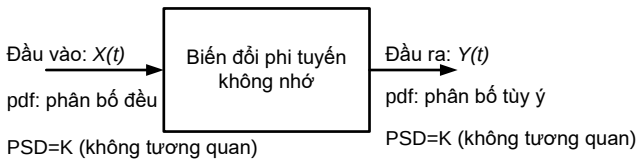
Phương pháp biến đổi ngược cho phép ta chuyển chuỗi ngẫu nhiên không tương quan phân bố đều U thành chuỗi không tương quan (các mẫu độc lập) X có hàm phân bố $F_X(x)$. Việc chuyển đổi dẫn đến sử dụng thiết bị phi tuyến không nhớ như được cho ở hình 7.4. Thực ra thiết bị phi tuyến không nhớ để đảm bảo chuỗi đầu ra không tương quan nhau nếu chuỗi đầu vào không tương quan nhau. Tất nhiên, theo định lý Weiner-Khitchine, một chuỗi các số ngẫu nhiên không tương quan có mật độ phổ công suất PSD không đổi (*trắng*). Kỹ thuật này được thiết lập đơn giản:

$$U = F_X\left(X\right) \tag{7.32}$$

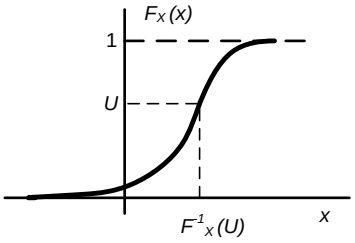
Và tìm x, ta có:

$$X = F_X^{-1}\left(U\right) \tag{7.33}$$

Để áp dụng kỹ thuật biến đổi ngược vào việc tìm x cần phải có hàm phân bố $F_X(x)$ *ở dạng kín*.



Hình 7.4: Phương pháp biến đổi ngược



Hình 7.5: Hàm phân bố tích lũy

Dễ dàng thấy rằng, kỹ thuật biến đổi ngược tạo ra biến ngẫu nhiên có hàm phân bố được yêu cầu. Biết rằng, phân bố $F_X(x)$ là hàm không giảm của đối số x như được minh hoạ trong hình 7.5. Theo định nghĩa:

$$F_X\left(x\right) = Pr\left\{X \le x\right\} \tag{7.34}$$

Đặt $X = F^{-1}(U)$ và thừa nhận rằng $F_X(x)$ là *đơn điệu* ta được:

$$F_X(x) = Pr\left\{\underbrace{F^{-1}(U)}_{X} \le x\right\} = Pr\{U \le F_X(x)\} = F_X(x) \tag{7.35}$$

Là kết quả mong muốn. Ta minh họa phương pháp này thông qua ví dụ đơn giản dưới đây.

Ví dụ 7.7: Biến ngẫu nhiên đều sẽ được chuyển thành biến ngẫu nhiên có phân bố mũ một phía.

$$f_X(x) = \beta e^{-\beta x} u(x), \quad \beta > 0 \tag{7.36}$$

Trong đó $u(x)$ là hàm bước nhảy đơn vị được định nghĩa là:

$$u(x) = \begin{cases} 1, & x > 0 \\ 0, & x < 0 \end{cases} \tag{7.37}$$

Trước hết là tìm hàm phân bố tích lũy CDF:

$$F_X(x) = \int_0^x \beta e^{-\beta y} dy = 1 - e^{-\beta x} \tag{7.38}$$

Phương trình hóa hàm phân bố với biến ngẫu nhiên đều U theo (7.38) và (7.32) ta được:

$$1 - e^{-\beta X} = U \tag{7.39}$$

Khi tìm X ta có:

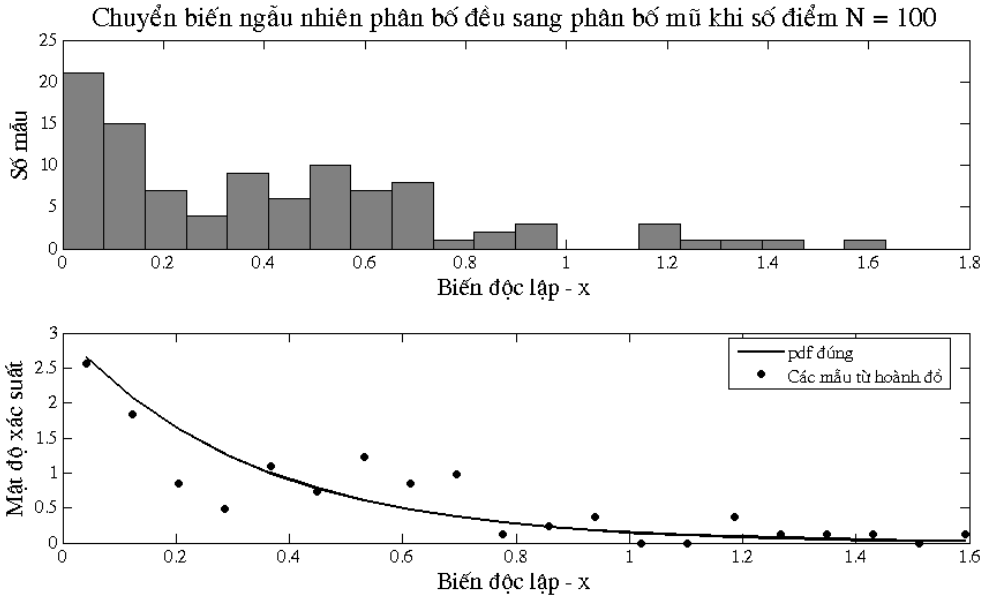
$$e^{-\beta X} = 1 - U \tag{7.40}$$

Vì biến ngẫu nhiên $1\text{-}U$ là tương đương với biến ngẫu nhiên U ($Z = U$ và $Z = 1\text{-}U$ có cùng hàm mật độ xác suất pdf) nên ta có thể viết lời giải cho X là:

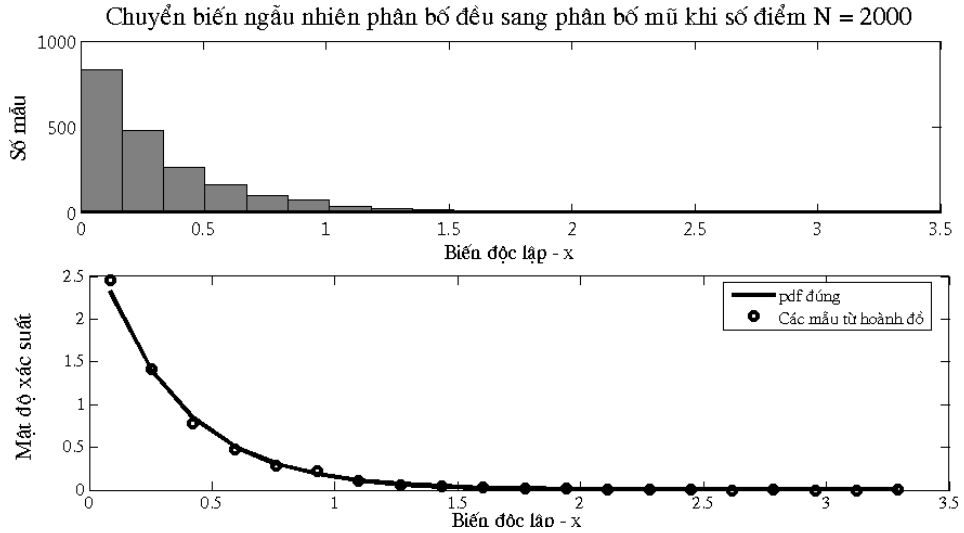
$$X = -\frac{1}{\beta} ln(U) \tag{7.41}$$

Mã chương trình Matlab chuyển đổi phân bố đều sang phân bố luỹ thừa được cho bởi **NVD7_uni2exp.m** dưới đây (có trong Phụ lục 7A).

Kết quả chạy chương trình với $\beta = 3$ và N = 100 được minh hoạ trong hình 7.6. Phần trên của hình cho thấy hoành đồ (*Histogram*). Phần dưới của hình cho thấy pdf lý thuyết và các giá trị "*thí nghiệm-mô phỏng*" với N = 100 mẫu. Các kết quả tương đối kém khi N = 100 dẫn đến cần thực hiện với nhiều mẫu dữ liệu hơn. Kết quả khi N = 2000 được minh hoạ trong hình 7.7 thấy rõ kết quả được cải thiện đáng kể (thực nghiệm tiệm cận đến lý thuyết).



Hình 7.6: Chuyển phân bố đều thành phân bố mũ khi N = 100



Hình 7.7: Chuyển phân bố đều thành phân bố mũ khi N = 2000

Ví dụ 7.8: Xét biến ngẫu nhiên Rayleigh được mô tả bởi pdf:

$$f_R(r) = \frac{r}{\sigma^2}.e^{-\frac{r^2}{2\sigma^2}}.u(r) \tag{7.42}$$

Trong đó dùng hàm bước nhảy đơn vị *u(r)* để xác định pdf là một phía. CDF được cho bởi:

$$F_R(r) = \int_0^r \frac{y}{\sigma^2} e^{-\frac{y^2}{2\sigma^2}}\, dy = 1 - e^{-\frac{r^2}{2\sigma^2}} \tag{7.43}$$

Đặt $F_R(R) = U$ ta có:

$$1 - e^{-\frac{R^2}{2\sigma^2}} = U \tag{7.44}$$

Tương đương với:

$$e^{-\frac{R^2}{2\sigma^2}} = U \tag{7.45}$$

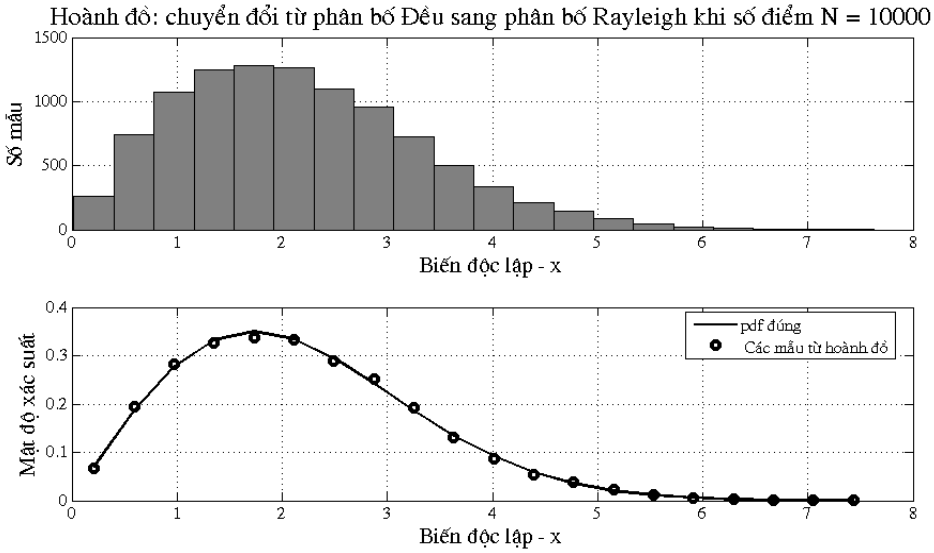
Trong đó một lần nữa ta lại thừa nhận sự tương đương giữa *1 - U* và *U*. Giải tìm *R* ta có:

$$R = \sqrt{-2\sigma^2 \ln(U)} \tag{7.46}$$

Sự chuyển đổi này là bước khởi đầu trong giải thuật Box-Muller là một trong những thuật toán cơ bản để tạo số Gausơ.

Cũng như ở ví dụ 7.7, ta quan tâm thực hiện chuyển đổi và định lượng hiệu năng là *hàm của số điểm được chuyển đổi*.

Chương trình Matlab **NVD7_uni2ray.m** dưới đây (có ở Phụ lục 7A), các kết quả chạy chương trình với *N* = 10000 được cho ở hình 7.8. Bạn đọc nên thực hiện chương trình với các giá trị khác của *N* và so kết quả với hình 7.8.



Hình 7.8: Chuyển phân bố đều thành phân bố Rayleigh với N = 100000

Hai ví dụ trên đã minh hoạ việc ứng dụng phương pháp biến đổi ngược cho các biến ngẫu nhiên liên tục. Tuy nhiên, kỹ thuật đó có thể áp dụng cho các biến ngẫu nhiên rời rạc. Phương pháp hoành đồ là phiên bản số (dữ liệu rời rạc) của phương pháp biến đổi ngược.

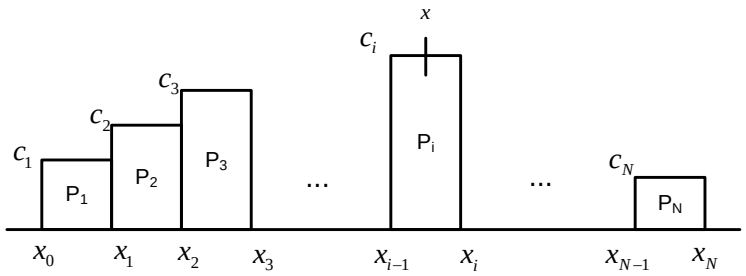
7.4.2. Phương pháp hoành đồ

Hoành đồ (***histogram:*** *hoành đồ - tổ chức đồ - biểu đồ thống kê*) là bộ ước tính hàm mật độ xác suất. Theo đó, giả sử ta có một tập dữ liệu được tập hợp từ thí nghiệm. Trong trường hợp cả pdf và CDF là không được biết, mặc dù pdf có thể được xấp xỉ bởi hoành đồ của dữ liệu. Vấn đề là triển khai một thuật toán để tạo ra một tập các mẫu có pdf xấp xỉ với pdf của dữ liệu thí nghiệm.

Trước hết, ta tạo ra hoành đồ của dữ liệu thí nghiệm. Giả sử hoành đồ được minh hoạ ở hình 7.9 là kết quả. Mỗi khi hoành đồ được tạo ra, thì ta được một xấp xỉ cho pdf và CDF, và vì vậy ta có thể áp dụng phương pháp biến đổi ngược. Kỹ thuật được trình bày ở đây là một sự mở rộng đơn giản của phương pháp biến đổi ngược.

Xác suất mà giá trị mẫu *x* nằm trong biến thứ *i* của hoành đồ là:

$$P_i = Pr\{x_{i-1} < x < x_i\} = c_i\left(x_i - x_{i-1}\right) \tag{7.47}$$



Hình 7.9: Hoành đồ của dữ liệu thí nghiệm

CDF được ước lượng tại điểm *x* ký hiệu là:

$$F_X\left(x\right) = F_{i-1} + c_i\left(x - x_{i-1}\right) \tag{7.48}$$

Trong đó:

$$F_{i-1} = Pr\{X \le x_{i-1}\} = \sum_{j=1}^{i-1} P_{i-1} \tag{7.49}$$

Sau đó, ta đặt $F_X(X) = U$, trong đó *U* là biến ngẫu nhiên đều.

Dẫn đến:

$$F_X\left(X\right) = U = F_{i-1} + c_i\left(X - x_{i-1}\right) \tag{7.50}$$

Giải tìm *X* ta được:

$$X = x_{i-1} + \frac{1}{c_i}\left(U - F_{i-1}\right) \tag{7.51}$$

Thuật toán cho bộ tạo số theo 3 bước sau:

1. Tạo *U* bằng cách lấy ra một mẫu từ bộ tạo số ngẫu nhiên tạo ra các số phân bố đều trong khoảng (0,1).

2. Xác định giá trị *i* sao cho:

$$F_{i-1} < U \le F_i \tag{7.52}$$

Trong đó F_i được xác định theo (7.49).

3. Tạo *X* theo (7.51) và trả *X* về chương trình đang gọi.

Độ tin cậy của bộ tạo số rõ ràng phụ thuộc vào tính chính xác của hoành đồ cơ bản. Tính chính xác của hoành đồ (chi tiết chương sau) phụ thuộc vào số các mẫu khả dụng.

7.4.3. Phương pháp loại bỏ

Kỹ thuật loại bỏ (hay tiếp nhận) để tạo các biến ngẫu nhiên có pdf mong muốn $f_X(x)$ hay $f_X(x)$ "*đích*", về cơ bản bao gồm việc lấy giới hạn pdf đích bởi hàm $Mg_X(x)$, trong đó $g_X(x)$ thể hiện pdf của biến ngẫu nhiên được tạo ra một cách dễ dàng và M là hằng số đủ lớn để đảm bảo:

$$Mg_X(x) \geq f_X(x), \quad \forall x \tag{7.53}$$

Ở dạng đơn giản nhất $g_X(x)$ phân bố đều trên $(0,a)$. Nếu pdf đích $f_X(x) = 0$ ở ngoài khoảng $(0,a)$, thì ta có:

$$Mg_X(x) = \begin{cases} b = \dfrac{M}{a}, & 0 \leq x \leq a \\ 0, & \text{nếu khác} \end{cases} \tag{7.54}$$

Trong đó, do $Mg(x)$ giới hạn $f_X(x)$:

$$b = \frac{M}{a} \geq max\{f_X(x)\} \tag{7.55}$$

Điều này được minh họa ở hình 7.10.

Thuật toán tạo biến ngẫu nhiên X có pdf là $f_X(x)$ được xác định bởi 4 bước sau:

1. Tạo U_1 và U_2 phân bố đều trong khoảng (0,1)

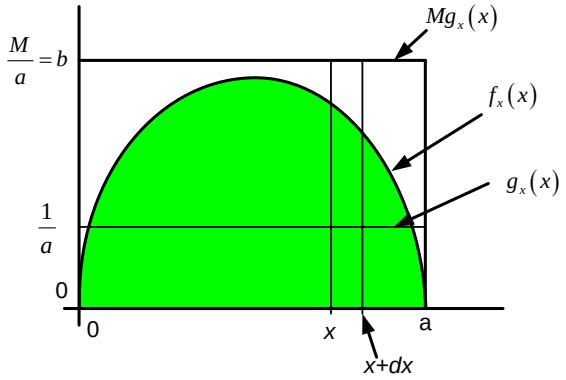
2. Tạo V_1 phân bố đều trong khoảng $(0,a)$, trong đó a là giá trị lớn nhất của X

3. Tạo V_2 phân bố đều trong khoảng $(0,b)$, trong đó b ít nhất là giá trị lớn nhất của $f_X(x)$

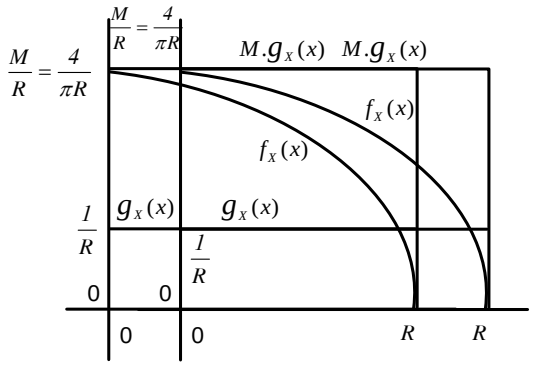
4. Nếu $V_2 \leq f_X(V_1)$, thì đặt $X = V_1$. Nếu không thỏa mãn bất đẳng thức thì hủy bỏ V_1 và V_2 và quá trình được lặp lại từ bước 1.

Dễ dàng chỉ ra rằng thủ tục này tạo ra biến ngẫu nhiên X có pdf đích là $f_X(x)$.

Vì V_1 và V_2 được phân bố đều, nên điểm được tạo ra bởi cặp mẫu (V_1, V_2) có xác suất rơi vào mọi nơi trong diện tích ab là như nhau. Xác suất mà V_1 được chấp nhận là phần diện tích ab nằm dưới pdf $f_X(x)$. Đây là tỉ số giữa diện tích có nền xám trong hình 7.10 với toàn bộ diện tích ab. Vì vậy:



Hình 7.10: Phương pháp loại bỏ



Hình 7.11: Phương pháp loại bỏ cho ví dụ 7.9

$$Pr\{V_1 \text{ được chấp nhận}\} = \frac{\int_0^a f_X(x)dx}{ab} \tag{7.56}$$

Theo định nghĩa, tử số trong biểu thức trên bằng 1, nên:

$$Pr\{V_1 \text{ được chấp nhận}\} = \frac{1}{ab} = \frac{1}{M} \tag{7.57}$$

Vì vậy:

$$P\{x < V_1 \le x+dx \,|\, V_1 \text{ được chấp nhận}\} = \frac{Pr\{x < V_1 \le x+dx \text{ , } V_1 \text{ được chấp nhận}\}}{Pr\{V_1 \text{được chấp nhận}\}}$$

$$= \frac{f_X(x)dx / ab}{1/ab} \tag{7.58}$$

$$= f_X(x)dx = Pr\{x < X \le x+dx\} \tag{7.59}$$

định nghĩa pdf đích.

Ví dụ 7.9: Trong ví dụ này ta áp dụng kỹ thuật đã được đề cập cho pdf sau

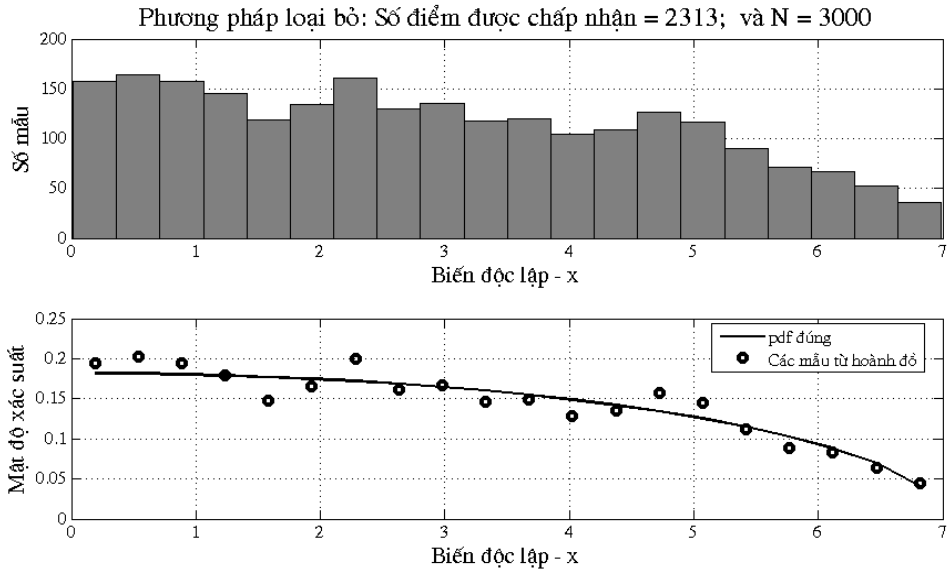
$$f_X(x) = \begin{cases} \frac{4}{\pi R^2}\sqrt{R^2-x^2}, & 0 \le x \le R \\ 0, & \text{nếu khác} \end{cases} \tag{7.60}$$

Ta tìm thuật toán để tạo biến ngẫu nhiên X có pdf $f_X(x)$ bằng cách đặt $a = R$ và $b = M/R$. Hình 7.10 được sửa đổi cho trường hợp cụ thể này được minh hoạ trong hình 7.11.

Mã chương trình Matlab được cho bởi file **NVD7_rejex1.m** (có trong Phụ lục 7A). Kết quả chạy chương trình với N = 3000 điểm được cho ở hình 7.12. Trong 3000 điểm thì 2301 điểm được chấp nhận và 699 điểm bị loại bỏ, hiệu quả 76,70%, nó gần với giá trị lý thuyết (khi $N \to \infty$) là 78,54%.

Mặc dù ta đã minh hoạ phương pháp loại bỏ cho trường hợp trong đó pdf lấy giới hạn là đều, nhưng kỹ thuật được minh hoạ ở đây dễ dàng sửa đổi cho trường hợp trong đó pdf lấy giới

hạn là không đều. Dưới dạng lý tưởng, pdf đích nên được lấy giới hạn chặt sao cho xác suất loại bỏ được giảm thiểu. Phương pháp loại bỏ làm việc tốt trong các trường hợp trong đó pdf đích có sự hỗ trợ hữu hạn (chỉ khác không trên dải hữu hạn). Tuy nhiên, sự hỗ trợ hữu hạn là không cần thiết và kỹ thuật loại bỏ, như được minh hoạ ở đây, có thể dễ dàng được mở rộng cho trường hợp hỗ trợ vô hạn.



Hình 7.12: Kết quả mô phỏng minh họa phương pháp loại bỏ

7.5. Tạo số ngẫu nhiên Gausơ không tương quan

Thấy rõ từ việc nghiên nghiên cứu các hệ thống truyền thông, thường xuyên gặp phải biến ngẫu nhiên Gausơ và tỏ ra là mô hình thích hợp cho tạp âm nhiệt và nhiều hiện tượng khác. Bộ tạo tạp âm Gausơ là khối cơ bản trong nhiều mô phỏng, kết quả là nhiều kỹ thuật đã được triển khai để tạo biến ngẫu nhiên Gausơ. CDF là:

$$F_X(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{y^2}{2\sigma^2}} dy = 1 - Q\left(\frac{x}{\sigma}\right) \tag{7.61}$$

Trong đó Q(.) là hàm Q Gausơ được định nghĩa là:

$$Q(x) = \frac{1}{\sqrt{2\pi}} \int_{x}^{\infty} e^{-\frac{y^2}{2}} dy \tag{7.62}$$

Vì không thể viết hàm Q Gausơ ở dạng kín, nên không dùng được kỹ thuật biến đổi ngược. Có thể áp dụng kỹ thuật loại bỏ nhưng không hiệu quả. Vì vậy, cần phải tìm kỹ thuật khác để tạo biến ngẫu nhiên Gausơ.

7.5.1. Phương pháp lấy tổng các biến ngẫu nhiên phân bố đều

Lý thuyết giới hạn trung tâm (CLT) cho ta một phương pháp tốt để triển khai biến ngẫu nhiên có pdf phân bố Gausơ. CLT phát biểu rằng, dưới các điều kiện khá tổng quát thì pdf của tổng N biến ngẫu nhiên độc lập sẽ hội tụ về biến ngẫu nhiên Gausơ khi $N \to \infty$.

Giả sử có N biến ngẫu nhiên đều độc lập U_i, $i = 1, 2,... N$. Từ N biến ngẫu nhiên đều này tạo thành:

$$Y = B\sum_{i=0}^{N}\left(U_i - \frac{1}{2}\right) \tag{7.63}$$

Trong đó B là hằng số thiết lập phương sai của Y. Từ CLT ta biết rằng Y hội tụ về biến ngẫu nhiên Gausơ khi $N \to \infty$. Vì $E\{U_i\} = \frac{1}{2}$ nên trung bình của Y là:

$$E\{Y\} = B\sum_{i=0}^{N}\left(E\{U_i\} - \frac{1}{2}\right) = 0 \tag{7.64}$$

Tìm được phương sai của Y bằng cách lưu ý đến phương sai của U_i-$1/2$ là:

$$var\left\{U_i - \frac{1}{2}\right\} = \int_{-1/2}^{1/2} x^2 dx = \frac{1}{12} \tag{7.65}$$

Vì các biến ngẫu nhiên thành phần U_i được giả định là độc lập nên:

$$\sigma_y^2 = B^2 \sum_{i=1}^{N} var\left\{U_i - \frac{1}{2}\right\} \tag{7.66}$$

Ta có:

$$\sigma_y^2 = \frac{NB^2}{12} \tag{7.67}$$

Vì vậy khi cho trước giá trị của N, thì phương sai của Y là σ_y^2 có thể được đặt một giá trị mong muốn bất kỳ bằng cách chọn B phù hợp. Việc chọn N là sự dung hoà giữa tốc độ và độ chính xác của các đuôi của pdf kết quả. Giá trị N thường được thiết lập là 12 vì $N = 12$ cho kết quả đơn giản $B = \sigma_y$.

Trong khi thủ tục tạo ra biến ngẫu nhiên Gausơ dựa vào định lý giới hạn tập trung là dễ hiểu, thì xuất hiện một số khó khi áp dụng vào các bài toán truyền tham số thực tế đó là:

Thứ nhất: vì $\left(U_i - \frac{1}{2}\right)$ thay đổi từ $-\frac{1}{2}$ *đến* $\frac{1}{2}$, nên theo (7.63) Y thay đổi từ $-\frac{BN}{2}$ *đến* $\frac{BN}{2}$. Theo đó, mặc dù (7.63) có thể xấp xỉ khá chính xác đến pdf của biến ngẫu nhiên Gausơ ở lân cận giá trị trung bình, thì các đuôi của pdf bị cắt bỏ ở bên ngoài $\pm\frac{BN}{2}$. Nếu mục đích mô phỏng hiệu năng xác suất lỗi ký hiệu của hệ thống truyền thông, thì các đuôi của pdf là rất quan trọng (vì các đuôi của pdf thể hiện các giá trị tạp âm gây ra các lỗi truyền dẫn). Có thể giảm thiểu ảnh hưởng của việc cắt các đuôi pdf của Y bằng cách chọn N đủ lớn.

Cụ thể, từ (7.67) giá trị của *B* là:

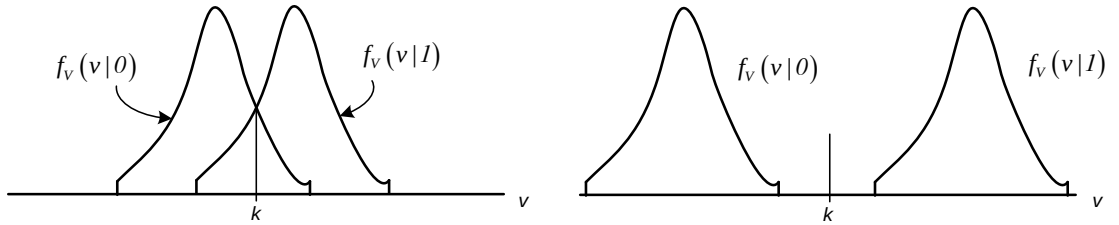
$$B = \sigma_y \sqrt{\frac{12}{N}} \tag{7.68}$$

Vì vậy, pdf "xấp xỉ Gausơ" được cắt sao cho nó chỉ khác 0 trong khoảng:

$$\pm\sigma_y \frac{N}{2}\sqrt{\frac{12}{N}} = \pm\sigma_y \sqrt{3N} \tag{7.69}$$

Với $N = 100$ thì biến ngẫu nhiên Gausơ bị cắt tại $\pm 17{,}32\,\sigma_y$. Đối với một số ứng dụng việc cắt tại 17 lần độ lệch chuẩn có thể vẫn mang lại các lỗi đáng kể. Giá trị phù hợp của *N* là *phụ thuộc ứng dụng*.

Khó khăn của việc cắt các đuôi pdf của tạp âm trong hệ thống truyền thông số được minh hoạ trong hình 7.13, cho thấy các pdf có điều kiện tại đầu ra của máy thu lọc thích hợp khi SNR đầu vào máy thu nhỏ và khi SNR đầu vào máy thu lớn (các đuôi của các pdf thực tế liên tục, hình 7.13 nhấn mạnh ảnh hưởng của cắt xén). Các pdf với điều kiện bit 0 nhị phân được phát đi ký hiệu là $f_V(v|0)$, và với điều kiện bit 1 nhị phân được phát đi ký hiệu là $f_V(v|1)$. Hình 7.13 được xây dựng với giả định phương sai tạp âm là không đổi và SNR được điều chỉnh bằng cách thay đổi công suất tín hiệu. Trường hợp SNR đầu vào máy thu đủ nhỏ như hình 7.13(a) thì các pdf điều kiện chồng lấn nhau đáng kể và xác suất lỗi có thể được xác định với độ chính xác hợp lý. Khi công suất tín hiệu tăng, thì đẩy các pdf điều kiện xa nhau và độ chính xác mô phỏng giảm đi. Do việc cắt các đuôi của các pdf có điều kiện, tăng công suất tín hiệu dẫn đến tình huống các pdf có điều kiện không còn chồng lấn nhau nữa, được minh hoạ ở hình 7.13(b). Nếu các pdf có điều kiện không chồng lấn nhau thì xác suất lỗi bằng 0 độc lập với SNR, rõ ràng là phi thực tế.



(a) pdf đầu ra bộ lọc phối hợp khi SNR đầu vào máy thu nhỏ (b) pdf đầu ra bộ lọc phối hợp khi SNR đầu vào máy thu lớn

Hình 7.13: Ảnh hưởng của việc cắt xén hàm mật độ xác suất pdf đầu ra bộ lọc phối hợp khi SNR đầu vào máy thu thấp và cao

Thứ hai: vì chọn *N* lớn khi sử dụng (7.63) để xấp xỉ biến ngẫu nhiên Gausơ. Vì nó thực hiện *N* lần gọi bộ tạo số ngẫu nhiên phân bố đều để tạo một giá trị của *Y*, nên khi dùng thuật toán (7.63) yêu cầu thời gian CPU vượt quá.

Hai đặc tính bù lại của (7.63): *(i)* nó làm tốt việc xấp xỉ biến ngẫu nhiên Gausơ ở lân cận trung bình của *Y*; *(ii) Y* sẽ xấp xỉ Gausơ cả khi pdf của các biến ngẫu nhiên U_i thành phần là phân bố không đều.

Trong khi ta chủ yếu đề cập trong mô phỏng dùng xử lý nối tiếp bằng một CPU duy nhất, thì điều này cho thấy các thuật toán không phù hợp với các ứng dụng xử lý nối tiếp truyền thống nhưng khá phù hợp với các cơ chế xử lý song song. Ví dụ nếu cơ chế xử lý song song nào đó sử dụng 100 CPU, thì nó có thể tạo 100 giá trị của một biến ngẫu nhiên đều tại cùng một thời điểm. Tổng 100 biến ngẫu nhiên đều dẫn đến xấp xỉ hoàn hảo cho biến ngẫu nhiên Gauso đối với hầu hết các ứng dụng và xử lý song song được thực hiện nhanh chóng. Đây chỉ là một minh chứng cho việc lựa chọn thuật toán phụ thuộc vào môi trường tính toán.

7.5.2. Ánh xạ biến ngẫu nhiên Rayleigh thành biến ngẫu nhiên Gauso

Từ ví dụ 7.8 cho thấy, biến ngẫu nhiên Rayleigh R có thể được tạo ra từ biến ngẫu nhiên đều U bằng quan hệ $R = \sqrt{-2\sigma^2 \ln U}$. Tại đây, ta xét bài toán ánh xạ biến ngẫu nhiên Rayleigh sang biến ngẫu nhiên Gauso.

Giả sử X và Y là 2 biến ngẫu nhiên Gauso độc lập có cùng phương sai σ^2. Vì X và Y là độc lập nên pdf đồng thời là tích của các pdf duyên biên. Vì vậy:

$$\begin{aligned} f_{XY}(x,y) &= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}} \\ &= \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \end{aligned} \tag{7.70}$$

Với x = r$\cos\theta$ và y = r$\sin\theta$ ta có:

$$x^2 + y^2 = r^2 \tag{7.71}$$

$$\theta = tan^{-1}\left(\frac{y}{x}\right) \tag{7.72}$$

pdf đồng thời $f_{R\Theta}(r,\theta)$ tìm được từ $f_{XY}(x,y)$ bởi phép biến đổi:

$$f_{R\Theta}(r,\theta) dA_{R\Theta} = f_{XY}(x,y) dA_{XY} \tag{7.73}$$

Trong đó $dA_{R\Theta}$ và dA_{XY} biểu diễn các diện tích vi phân trong các mặt phẳng R, Θ và X,Y. Theo (7.73) ta có:

$$f_{R\Theta}(r,\theta) = f_{XY}(x,y) \frac{dA_{XY}}{dA_{R\Theta}} \bigg|_{\substack{x=r\cos\theta \\ y=r\sin\theta}} \tag{7.74}$$

Tỉ số của các diện tích vi phân là Jacobian của biến đổi, nó là:

$$\frac{dA_{XY}}{dA_{R\Theta}} = \frac{\partial(x,y)}{\partial(r,\theta)} = \begin{vmatrix} dx/dr & dx/d\theta \\ dy/dr & dy/d\theta \end{vmatrix} \tag{7.75}$$

Cho ta:

$$\frac{dA_{XY}}{dA_{R\Theta}} = \begin{vmatrix} \cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta \end{vmatrix} = r \tag{7.76}$$

Vì vậy thay (7.70), (7.71) và (7.76) vào (7.74) ta được kết quả:

$$f_{R\Theta}(r,\theta) = \frac{r}{2\pi\sigma^2} e^{-\frac{r^2}{2\sigma^2}}, \quad 0 \le r < \infty, \quad 0 \le \theta < 2\pi \tag{7.77}$$

Bây giờ ta khảo sát các pdf duyên biên của R và Θ:

pdf của R là:

$$f_R(r) = \int_0^{2\pi} \frac{r}{2\pi\sigma^2} \mathrm{e}^{-\frac{r^2}{2\sigma^2}} d\theta = \frac{r}{\sigma^2} \mathrm{e}^{-\frac{r^2}{2\sigma^2}}, \qquad 0 \le r < \infty \tag{7.78}$$

pdf của Θ là:

$$f_\Theta(\theta) = \int_0^{\infty} \frac{r}{2\pi\sigma^2} e^{-\frac{r^2}{2\sigma^2}} dr = \frac{1}{2\pi}, \qquad 0 \le \theta < 2\pi \tag{7.79}$$

Do đó *R* là biến ngẫu nhiên phân bố Rayleigh và Θ là biến ngẫu nhiên phân bố đều. Vì biến ngẫu nhiên Rayleigh được tạo ra từ 2 biến ngẫu nhiên Gauso *trực giao***,** theo đó các phép chiếu trực giao của biến ngẫu nhiên Rayleigh tạo ra cặp biến ngẫu nhiên Gauso. Vì vậy, giả sử *R* là biến ngẫu nhiên Rayleigh và Θ là biến ngẫu nhiên đều trên khoảng $(0,2\pi)$, thì các biến ngẫu nhiên Gauso *X* và *Y* được tạo ra bằng các quan hệ sau:

$$X = R\cos\Theta \tag{7.80}$$

$$Y = R\sin\Theta \tag{7.81}$$

Cả *X* và *Y* đều là các biến ngẫu nhiên Gauso trung bình 0 và có cùng phương sai σ^2. Vì chúng là các biến ngẫu nhiên Gauso và không tương quan, nên *X* và *Y độc lập thống kê* (lưu ý rằng, tính chất của biến ngẫu nhiên Gauso là không tương quan thì độc lập thống kê, tính chất này không có được đối với các phân bố khác). Vì vậy, cặp biến ngẫu nhiên Gauso độc lập *X* và *Y* được tạo ra từ cặp biến ngẫu nhiên phân bố đều U_1 và U_2 bằng thuật toán:

$$X = \sqrt{-2\sigma^2 \ln(U_1)}.\cos(2\pi U_2) \tag{7.82}$$

$$Y = \sqrt{-2\sigma^2 \ln(U_1)}.\sin(2\pi U_2) \tag{7.83}$$

trong đó ta đã sử dụng (7.46) cho *R*.

Chương trình Matlab thực hiện thuật toán Box-Muller được cho ở file **NVD7_ boxmu1.m** (có trong Phụ lục 7A).

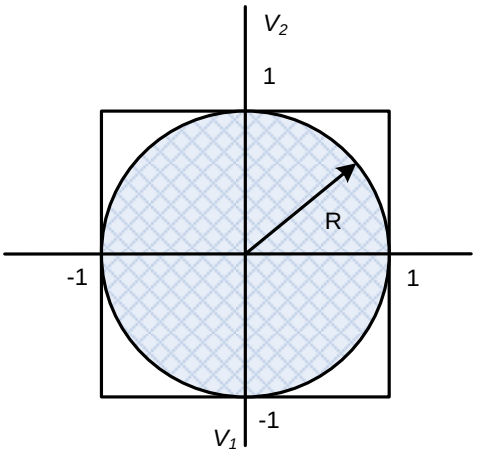
7.5.3. Phương pháp cực

Thuật toán khác để tạo cặp biến ngẫu nhiên Gausơ không tương quan trung bình 0 là phương pháp cực. Thuật toán bao gồm các bước sau:

1. Tạo 2 biến ngẫu nhiên độc lập U_1 và U_2, cả hai đều được phân bố đều trong khoảng (0,1).
2. Đặt $V_1 = 2U_1 - 1$ và $V_2 = 2U_2 - 1$ sao cho V_1 và V_2 là độc lập thống kê nhau và được phân bố đều trong khoảng (-1,1).
3. Tạo $S = \sqrt{V_1^2 + V_2^2}$. Nếu $S<1$ tiếp tục thực hiện bước 4. Nếu $S \geq 1$ huỷ S và quay trở lại bước 1.
4. Tạo $A(S) = \sqrt{\left(-2\sigma^2 \ln S\right)/S}$.
5. Đặt X = A(S)V_1 và Y = A(S)V_2.

Mã chương trình Matlab tạo cặp véc tơ ngẫu nhiên Gausơ sử dụng phương pháp cực được cho bởi file **NVD7_ polar.m** (có trong Phụ lục 7A).

Lưu ý rằng, ta đã dùng hàm **rand** có sẵn trong thư viện Matlab để tạo cặp biến ngẫu nhiên đều trong chương trình **NVD7_ polar.m**.



Hình 7.14: Phương pháp cực

Phương pháp cực là một ví dụ của phương pháp loại bỏ, vì từ bước 3 một số giá trị của S bị loại bỏ. Do đó, sẽ tạo ra ít hơn N biến ngẫu nhiên mỗi khi gọi hàm này. Dễ dàng xác định được xác suất loại bỏ từ hình 7.14. Các biến ngẫu nhiên V_1 và V_2 được phân bố đều trong hộp có diện tích $A_{hộp} = 4$ giới hạn đường tròn bán kính $R = 1$ và diện tích $A_{vòng\ tròn} = \pi$. Vì vậy, xác suất mà S bị loại bỏ là:

$$Pr\{loại\ bỏ\} = 1 - \frac{A_{vòng\ tròn}}{A_{hộp}} = 0,2146 \tag{7.84}$$

Thuật toán cực thường cung cấp các kết quả có các tính chất tương quan tốt hơn thuật toán Box-Muller. Tuy nhiên, có một nhược điểm của phương pháp cực. Lưu ý rằng, N lần gọi đối với thuật toán Box-Muller sẽ tạo N cặp biến ngẫu nhiên Gausơ. Nếu thuật toán cực được gọi N lần, thì số cặp biến ngẫu nhiên Gausơ được tạo ra sẽ là *một biến ngẫu nhiên* có trung bình $(\pi / 4)N$. Thực tế phải thực hiện nhiều lần gọi không được biết trước cho thuật toán cực để tạo một số biến ngẫu nhiên Gausơ cho trước thường làm phức tạp chương trình mô phỏng.

7.5.4. Thực hiện Matlab

Trước phiên bản Matlab 5, con số ngẫu nhiên Gausơ được chứa trong thư viện Matlab, **randn,** sử dụng bộ tạo số tiêu chuẩn nhỏ nhất theo (7.24) và dùng phương pháp cực để ánh xạ các số ngẫu nhiên phân bố đều sang các số ngẫu nhiên có phân bố Gausơ. Bắt đầu từ Matlab 5, một thuật toán hoàn toàn khác không chứa các phép nhân hoặc phép chia. Bộ tạo số ngẫu nhiên

mới trực tiếp tạo ra các số có pdf phân bố Gausơ không cần phải chuyển đổi các số ngẫu nhiên phân bố đều. Vì không dùng phép nhân và phép chia trong thuật toán này, và không cần bước chuyển từ phân bố đều sang phân bố Gausơ nên rất nhanh. Thuật toán được dùng trong các phiên bản sau của Matlab là một phiên bản cải tiến của thuật toán Ziggurt cơ bản.

7.6. Tạo số ngẫu nhiên Gausơ tương quan

Ta mới chỉ tạo ra các số ngẫu nhiên *không tương quan*. Dưới đây, ta xét việc tạo các số ngẫu nhiên có pdf phân bố Gausơ và *tương quan*. *Trước hết*, ta nghiên cứu kỹ thuật đơn giản tạo 2 chuỗi liên quan với nhau thông qua hệ số tương quan cho trước. *Sau đó*, ta xét trường hợp tổng quát hơn ở đó chuỗi được tạo ra có PSD cho trước. Tất nhiên, việc thiết lập PSD cho trước là tương đương với việc thiết lập hàm tự tương quan cho trước.

7.6.1. Thiết lập hệ số tương quan

Ta đã thấy 2 phương pháp tạo cặp biến ngẫu nhiên Gausơ không tương quan. Dễ dàng ánh xạ cặp biến ngẫu nhiên Gausơ không tương quan X và Y thành cặp biến ngẫu nhiên Gausơ có mức độ tương quan cụ thể. Giả sử X và Y có trung bình 0 và không tương quan thì bước tiếp theo là tạo biến ngẫu nhiên thứ 3 được định nghĩa bởi:

$$Z = \rho X + \sqrt{1-\rho^2}\,Y \tag{7.85}$$

Trong đó ρ là tham số với $|\rho| \le 1$. Với thuật toán này, X và Y là các biến ngẫu nhiên trung bình 0 có phương sai bằng nhau. Ta sẽ chỉ ra rằng ρ là hệ số tương quan liên hệ X và Z.

Việc chứng minh là đơn giản. Thấy rõ Z là biến ngẫu nhiên Gausơ bởi lẽ nó là kết hợp tuyến tính của các biến ngẫu nhiên Gausơ. Theo đó, Z cũng có trung bình 0 nếu X và Y có trung bình 0. Phương sai của Z là:

$$\begin{aligned}\sigma_Z^2 &= E\left\{\left[\rho X + \sqrt{1-\rho^2}\,Y\right]^2\right\} \\ &= \rho^2.E\{X^2\} + 2\rho\sqrt{1-\rho^2}.E\{XY\} + (1-\rho^2).E\{Y^2\}\end{aligned} \tag{7.86}$$

Vì $E\{XY\} = E\{Y\}.E\{Y\} = 0$ và $\sigma_X^2 = \sigma_Y^2 = \sigma^2$ nên ta có:

$$\sigma_Z^2 = \rho^2\sigma^2 + (1-\rho^2)\sigma^2 = \sigma^2 \tag{7.87}$$

Hiệp biến (đồng phương sai) E{XZ} là:

$$\begin{aligned}E\{XZ\} &= E\{X[\rho X + (1-\rho)Y]\} = \rho E\{X^2\} + (1-\rho)E\{XY\} \\ &= \rho E(X^2) = \rho\sigma^2\end{aligned} \tag{7.88}$$

Trong đó lưu ý rằng, *X* và *Y* là độc lập và có trung bình 0. Hệ số tương quan ρ_{XZ} là:

$$\rho_{XZ} = \frac{E\{XZ\}}{\sigma_X\sigma_Z} = \frac{\rho\sigma^2}{\sigma^2} = \rho \tag{7.89}$$

Như được mong đợi.

7.6.2. Thiết lập hàm tự tương quan hoặc mật độ phổ công suất tuỳ ý

Kỹ thuật tổng quát để thiết lập chuỗi các số ngẫu nhiên có hàm tự tương quan cho trước, hoặc tương đương với PSD cho trước là *lọc* tập các mẫu không tương quan sao cho thiết lập được PSD đích. Theo định nghĩa, các mẫu không tương quan có PSD là không đổi (hằng số) trong độ rộng băng tần mô phỏng $|f|<f_s/2$. Theo định nghĩa, phương sai là diện tích nằm dưới hàm $S_n(f)$ được minh hoạ trong hình 7.15:

$$\sigma_n^2 = \frac{1}{2} N_0 f_s \tag{7.90}$$

Vì vậy, để thiết lập PSD tạp âm cho trước N_0, thì phương sai bộ tạo số ngẫu nhiên (tạp âm) và tần số lấy mẫu phải thoả mãn:

$$f_s = \frac{2\sigma_n^2}{N_0} \tag{7.91}$$

Vì vậy, phương sai bộ tạo tạp âm khi cho trước PSD là hàm của tần số lấy mẫu.

Việc thiết lập PSD mong muốn cho chuỗi ngẫu nhiên là tương đối dễ hiểu bằng cách sử dụng kết quả cơ bản từ lý thuyết quá trình ngẫu nhiên cơ bản. Ta biết rằng, nếu đưa vào hệ thống tuyến tính một quá trình ngẫu nhiên có PSD là $S_X(f)$, thì PSD đầu ra hệ thống là:

$$S_Y(f) = |H(f)|^2 S_X(f) \tag{7.92}$$

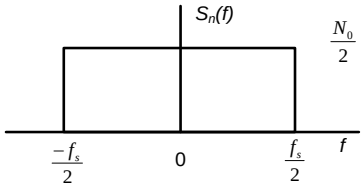
Trong đó $H(f)$ là hàm truyền đạt của hệ thống tuyến tính. Điều này được minh hoạ ở hình 7.16. Nếu các mẫu tạp âm là độc lập thì PSD đầu vào là không đổi tại $K = N_0/2$ *[W/Hz]:*

$$S_Y(f) = |H(f)|^2 K \tag{7.93}$$

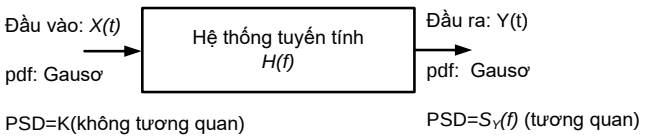
Và cần có $H(f)$ để thiết lập PSD đích là:

$$H(f) = \sqrt{\frac{S_Y(f)}{K}} \tag{7.94}$$

Vì vậy, bài toán định dạng mật độ phổ công suất nhằm đáp ứng yêu cầu cho trước quy về bài toán tìm bộ lọc có hàm truyền đạt $H(f)$ sao cho $H^2(f)$ tạo ra định dạng phổ theo yêu cầu.



Hình 7.15: PSD đối với các mẫu độc lập



Hình 7.16: Tạo chuỗi ngẫu nhiên tương quan

Ví dụ 7.10: Phương pháp thông dụng để tổng hợp một bộ lọc có hàm truyền đạt cho trước là xác định sự phù hợp nhất với hàm truyền đạt sao cho lỗi bình phương trung bình cực tiểu MMSE. Giải các phương trình Yule-Walker bằng hàm Matlab *yulewalk* cho vấn đề này. Cụ thể, *yulewalk* xác định hệ số lọc b_k và a_k sao cho hàm truyền đạt:

$$H(z) = \frac{b_0 + b_1 z^{-1} + \cdots + b_n z^{-n}}{1 + a_1 z^{-1} + \cdots + a_n z^{-n}} \tag{7.95}$$

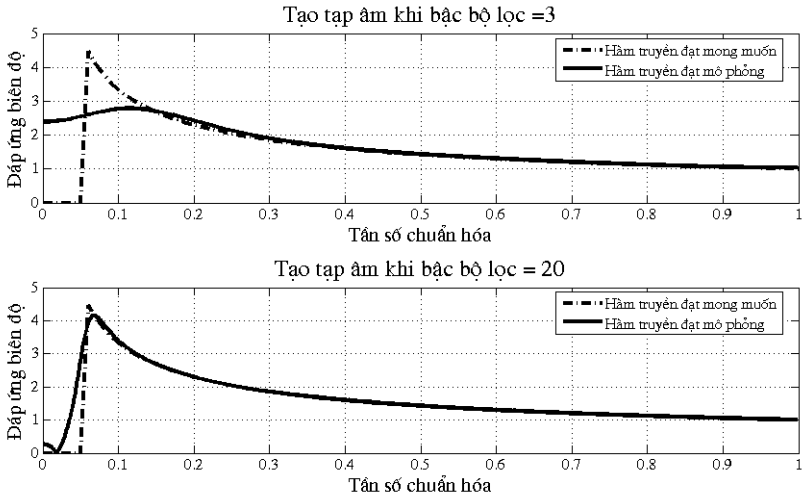
Có lỗi bình phương trung bình cực tiểu MMSE (lưu ý rằng, MSE là lấy trung bình của lỗi được bình phương) phù hợp với hàm truyền đạt cho trước.

Để minh hoạ kỹ thuật này, giả sử cho trước PSD có dạng *S(f) = K/f*, thường được dùng để mô hình hóa tạp âm pha trong các bộ tạo dao động. Để tạo tạp âm này, ta phải tạo một bộ lọc có hàm truyền đạt dạng $H(f) = K/\sqrt{f}$. Cho tạp âm trắng qua bộ lọc này sẽ tạo ra quá trình tạp âm theo yêu cầu. Vì $H(f) \to \infty$ khi $f \to 0$, nên hàm truyền đạt được định nghĩa là:

$$H(f) = \begin{cases} 0, & |f| < f_0 \\ \dfrac{K}{\sqrt{f}}, & |f| > f_0 \end{cases} \tag{7.96}$$

Mã chương trình Matlab **NVD7_ flicker.m** (có trong Phụ lục 7A) tạo ra một xấp xỉ cho *H(f)* trong đó tần số được chuẩn hoá theo tần số Nyquist f_N và $f_0 = f_N/20$:

Kết quả chạy chương trình **NVD7_ flicker.m** được minh hoạ ở hình 7.17, trong đó hàm truyền đạt mong muốn được minh hoạ bằng đường nét đứt. Xấp xỉ bậc 3 được minh hoạ ở phần trên và xấp xỉ bậc 20 được minh hoạ ở phần dưới. Trong vùng tần số ở đó H(f) thay đổi nhanh thì cần có xấp xỉ bậc cao hơn. Giải pháp này là công cụ mạnh để tạo các PSD tùy ý.



Hình 7.17: Tạo tạp âm

Ví dụ 7.11: Mô phỏng các hệ thống vô tuyến trong đó có tính di động, cần phải tạo một quá trình có mật độ phổ công suất PSD:

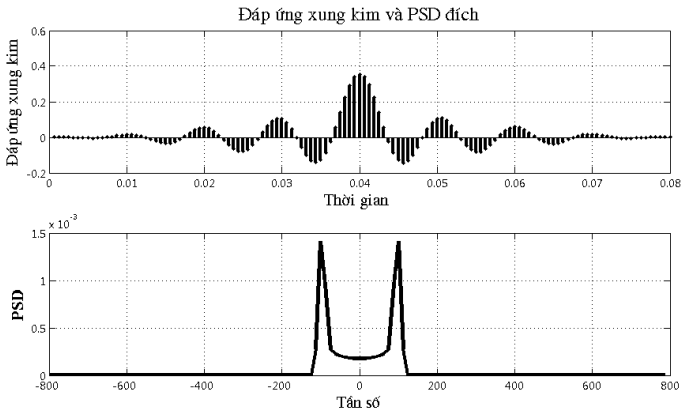
$$S(f)=\begin{cases}\dfrac{1}{\sqrt{1-(f/f_d)^2}}, & |f|<f_d \\ 0, & \text{nếu khác}\end{cases} \tag{7.97}$$

Để thể hiện hiệu ứng dopper. Đại lượng f_d trong (7.97) thể hiện tần số Doppler lớn nhất. Như được minh hoạ trong (7.94), hàm truyền đạt bộ lọc là:

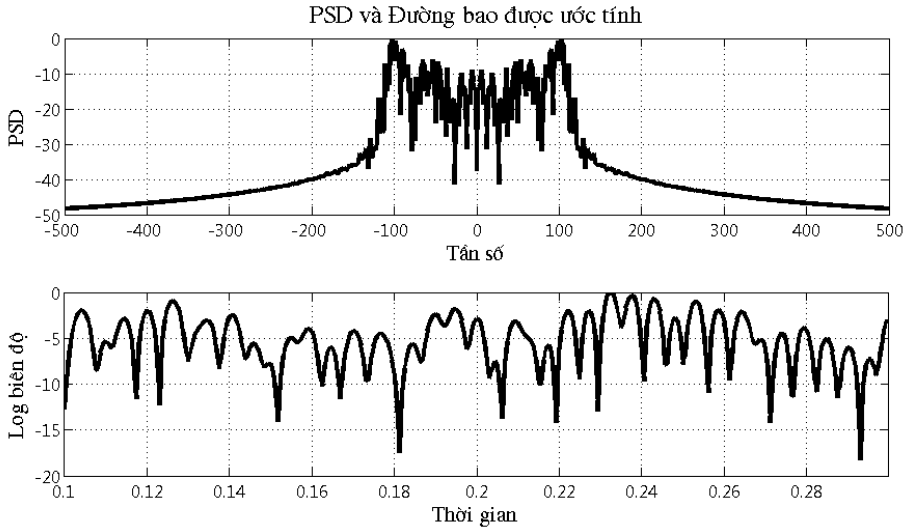
$$H(f)=\begin{cases}\left[1-(f/f_d)^2\right]^{-1/4}, & |f|<f_d \\ 0, & \text{nếu khác}\end{cases} \tag{7.98}$$

Bộ lọc này được thực hiện như bộ lọc FIR có đáp ứng xung đạt được bằng cách thực hiện IDFT lên các giá trị mẫu của (7.98).

Mã chương trình Matlab tạo đáp ứng xung kim của bộ lọc Jakes, và việc lọc tạp âm trắng bằng bộ lọc, được cho bởi file **NVD7_Jakes.m** có trong Phụ lục 7A. Kết quả chạy chương trình **NVD7_Jakes.m** được cho ở các hình 7.18 và 7.19. Hình 7.18 minh hoạ đáp ứng xung và hàm truyền đạt H(f). Hình 7.19 minh hoạ ảnh hưởng của việc cho tạp âm trắng phức qua bộ lọc. Phần trên chỉ ra rằng PSD được ước tính tại đầu ra bộ lọc. Phần dưới minh hoạ độ lớn của đường bao trên thang loga. Trong hệ thống truyền thông vô tuyến, tương ứng với đường bao pha đinh.



Hình 7.18: Đáp ứng xung kim và PSD đích

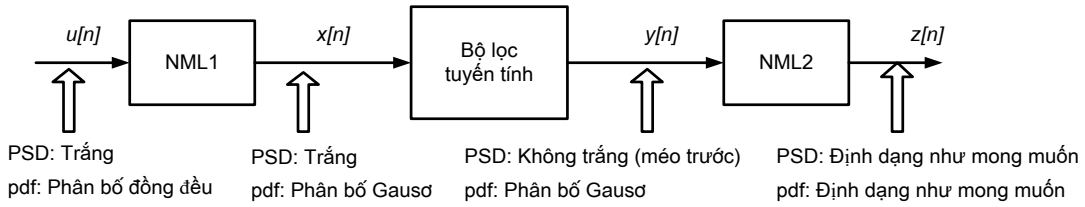


Hình 7.19: PSD được ước tính và hàm đường bao

7.7. Thiết lập hàm mật độ xác suất và mật độ phổ công suất

Ở dạng tổng quát, tạo dạng sóng tạp âm cần phải đáp ứng đồng thời các yêu cầu về pdf và PSD là một nhiệm vụ khó khăn. Tuy nhiên bài toán sẽ đơn giản, nếu pdf đầu ra hệ thống là Gausơ thì chỉ cần tạo một chuỗi mẫu cho đầu vào bộ lọc trong đó các mẫu là Gausơ và độc lập. Vì các mẫu đầu vào là độc lập nên PSD sẽ là hằng số tại K [W/Hz]. PSD có thể được định dạng bằng việc dùng bộ lọc tuyến tính như đã mô tả. Vì bộ lọc định dạng mật độ phổ công suất PSD là tuyến tính nên hàm mật độ xác suất pdf của đầu ra sẽ là Gausơ. Có được kết quả đơn giản bởi lẽ *bất kỳ biến đổi tuyến tính của một quá trình Gausơ sẽ tạo ra một quá trình Gausơ khác*. Tuy nhiên, nhiều bài toán thực tế có thể giải quyết được theo cách này.

Nếu cả pdf và PSD của dạng sóng đích được định rõ và yêu cầu pdf khác với Gausơ thì bài toán khó hơn rất nhiều. Lời giải cho bài toán này được đề xuất bởi Sondhi[12]. Sơ đồ cơ bản để thực hiện thuật toán Sondhi được minh hoạ ở hình 7.20. Như thường lệ, ta bắt đầu bằng một chuỗi các mẫu *u[n]* phân bố đều trong khoảng (0,1). Ngoài ra, chuỗi {*u[n]*} được coi là tương quan delta sao cho PSD là trắng. Nhiệm vụ của phần tử phi tuyến không nhớ đầu tiên MNL1 là ánh xạ chuỗi {*u[n]*} thành chuỗi {*x[n]*}({x[n]}là trắng nhưng có pdf phân bố Gausơo. Ta đã nghiên cứu một số kỹ thuật để thực thi ánh xạ này. Hoạt động lọc là làm méo trước phổ sao cho PSD là $S_Y(f)$. Vì bộ lọc là tuyến tính, nên pdf của chuỗi {*y[n]*} vẫn là Gausơ. Hoạt động cơ bản của phần tử phi tuyến không nhớ thứ hai MNL2 là ánh xạ pdf phân bố Gausơ của chuỗi {*y[n]*} thành pdf mong muốn cuối cùng. Tuy nhiên, tính phi tuyến thứ hai cũng ảnh hưởng lên $S_Y(f)$. Vì vậy, bộ lọc phải sửa đổi PSD đã được làm méo trước $S_Y(f)$ sao cho khi cho chuỗi mẫu qua thiết bị phi tuyến thứ hai làm cho chuỗi {*z[n]*} có cả PSD và pdf mong muốn.

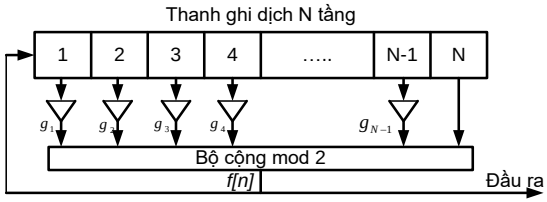


Hình 7.20: Thuật toán Sondhi

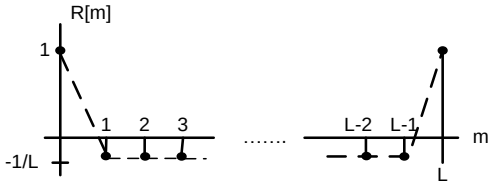
7.8. Các bộ tạo chuỗi PN

Các bộ tạo chuỗi giả ngẫu nhiên PN được dùng trong nhiều ứng dụng, đặc biệt trong lĩnh vực đồng bộ. Chẳng hạn, các chuỗi PN được dùng để xấp xỉ hóa biến ngẫu nhiên có hàm mật độ xác suất phân bố đều. Bộ tạo chuỗi PN có thể có một số dạng, nhưng dạng thông dụng nhất mà ta sẽ tập trung đề cập được minh hoạ trong hình 7.21.

Trong môi trường mô phỏng lý do quan trọng nhất để sử dụng chuỗi PN là mô hình hóa các nguồn dữ liệu. Bằng cách sử dụng bộ tạo chuỗi PN sẽ tạo ra hầu hết các kết hợp bit có chiều dài cho trước trên chiều dài mô phỏng ngắn nhất. Sẽ được dùng để nghiên cứu ảnh hưởng của giao thoa giữa các ký hiệu ISI trong chương 10 khi đó ta xét các kỹ thuật mô phỏng bán giải tích.



Hình 7.21: Bộ tạo chuỗi PN



Hình 7.22: Tự tương quan các nội dung thanh ghi dịch đối với chuỗi PN dài cực đại

Bộ tạo chuỗi PN gồm 3 phần cơ bản: thanh ghi dịch N tầng, bộ cộng mod-2, và véc tơ kết nối (xác định các kết nối giữa các tầng của thanh ghi dịch cụ thể với bộ cộng mod-2). Véc tơ kết nối thiết lập các đặc tính hiệu năng của bộ tạo chuỗi và được định nghĩa bởi đa thức:

$$g(D) = 1 + g_1 D + g_2 D^2 + ... + g_{N-1} D^{N-1} + D^N \tag{7.99}$$

Nếu $g_i = 1$ thì tầng thứ i của thanh ghi dịch được nối với bộ cộng mod-2. Nếu $g_i = 0$ thì không có kết nối. Lưu ý rằng, trong đa thức $g(D)$ cả g_0 và g_N đều bằng 1.

Có thể chỉ ra rằng chu kỳ lớn nhất của đầu ra bộ tạo chuỗi PN là:

$$L = 2^N - 1 \tag{7.100}$$

Và đạt được khi và chỉ khi đa thức $g(D)$ là đa thức nguyên thuỷ. Hàm tự tương quan $R[m]$ tại đầu ra bộ tạo chuỗi PN được minh hoạ trong hình 7.22, trong đó ta đã giả thiết các giá trị dữ liệu là ± 1. Khi $m = 0$ hoặc bội số của L thì hàm tự tương quan bằng 1. Khi $0 < m < L$ thì hàm tự tương quan $R[m] = -1/L$ và xấp xỉ bằng 0 khi L lớn. Theo đó, khi các chuỗi PN có chu kỳ *lớn*, thì hàm tự tương quan tiến dần tới ***xung kim***. Kết quả là, mật độ phổ công suất PSD

xấp xỉ là trắng như mong muốn. Chuỗi PN có nhiều tính chất thú vị, một số tính chất quan tâm trong môi trường mô phỏng gồm:

- Chuỗi gần như được cân bằng. Nói cách khác, trong một chu kỳ của chuỗi thì số bit 1 sẽ lớn hơn số bit 0 là 1. Bằng cách thêm bit 0 bổ sung tại điểm bên phải thì chuỗi được cân bằng.

- Tất cả các kết hợp bit có thể có đều xuất hiện trong một chu kỳ, ngoại trừ sự kiện không có chuỗi *N* bit 0 nhưng có chuỗi *N* bit 1 (lưu ý rằng: nếu tất cả các bộ ghi dịch chứa số không nhị phân, thì bộ tạo sẽ trở nên bị kẹt trong trạng thái toàn không, bằng cách thêm số không ở điểm có *N-1* số không, thì chuỗi trở nên cân bằng. Kết quả sẽ là chuỗi deBruijn).

- Hàm tự tương quan mặc dù tuần hoàn, nhưng gần giống như dạng sóng nhị phân ngẫu nhiên.

Việc thiết kế bộ tạo chuỗi PN dựa vào thanh ghi dịch *N* tầng quy về việc tìm đa thức nguyên thuỷ bậc *N*. Khi *N* lớn, đây có thể là một nhiệm vụ rất khó khăn. Tuy nhiên, các bảng mở rộng của các đa thức nguyên thuỷ có sẵn trong các tài liệu. Một số được biểu diễn phù hợp với (7.99) được cho ở bảng 7.1.

Bảng 7.1: Bảng một số đa thức nguyên thuỷ

N	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}	g_{11}	g_{12}	g_{13}	g_{14}
3	1	0	1											
4	1	0	0	1										
5	0	1	0	0	1									
6	1	0	0	0	0	1								
7	0	0	1	0	0	0	1							
8	0	1	1	1	0	0	0	1						
9	0	0	0	1	0	0	0	0	1					
10	0	0	1	0	0	0	0	0	0	1				
11	0	1	0	0	0	0	0	0	0	0	1			
12	1	0	0	1	0	1	0	0	0	0	0	1		
13	1	0	1	1	0	0	0	0	0	0	0	0	1	
14	1	0	0	0	0	1	0	0	0	1	0	0	0	1

Triển khai chương trình mô phỏng cho bộ tạo chuỗi PN là khá đơn giản. Trước hết, ta thể hiện nội dung thanh ghi dịch và véc tơ kết nối bằng các véc tơ sau:

$$B = \begin{bmatrix} b_1 & b_2 & .... & b_{N-1} & b_N \end{bmatrix} \tag{7.101}$$

$$G = \begin{bmatrix} g_1 & g_2 & .... & g_{N-1} & g_N \end{bmatrix} \tag{7.102}$$

Đầu ra bộ cộng mod-2 như được minh hoạ trong hình 7.21 là ký hiệu hồi tiếp *f[n]*, được định nghĩa bởi:

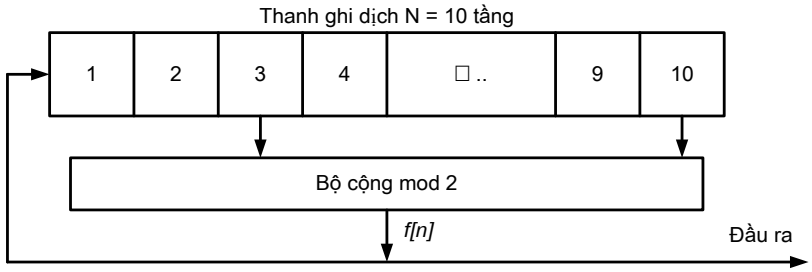
$$f[n] = \sum_{i=1}^{N} b_i g_i = B\left(G^T\right) \tag{7.103}$$

Tín hiệu hồi tiếp cũng là giá trị tiếp theo của b_1. Hiển nhiên, thanh ghi phải được khởi tạo bằng véc tơ B trong đó ít nhất phải có một giá trị $b_j \neq 0$.

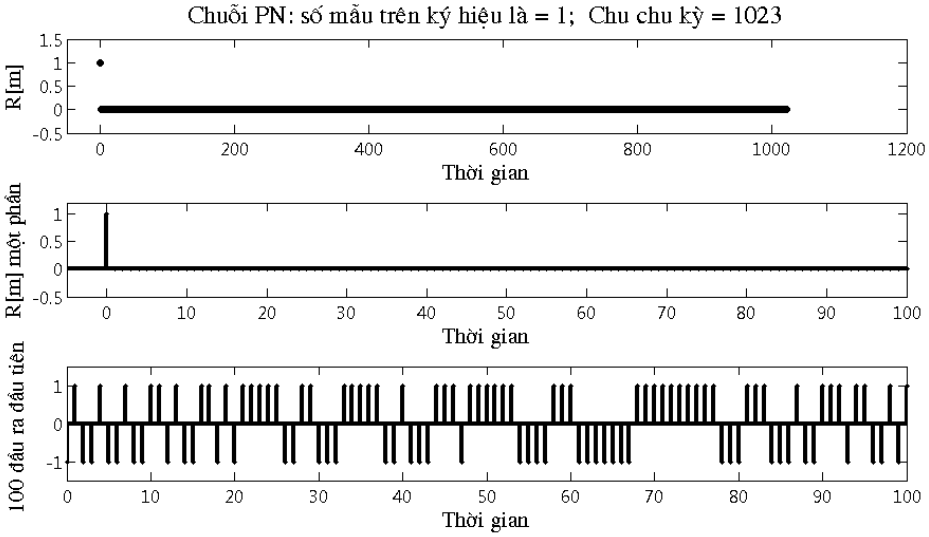
Ví dụ 7.12: Ta thiết kế bộ tạo chuỗi PN với $N = 10$. Với đa thức nguyên thủy $g(D)$ và từ biểu thức (7.100), thì chu kỳ sẽ là $L = 1023$, từ bảng 7.1 véc tơ kết nối là:

$$G = [0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1] \tag{7.104}$$

cho biết kết nối từ tầng thứ ba và tầng cuối cùng của thanh ghi dịch với bộ cộng mod-2, tạo ra cấu hình được cho ở hình 7.23. Mã chương trình Matlab mô phỏng bộ tạo chuỗi PN được cho ở file **NVD7_PNsim** (có trong Phụ lục 7A).

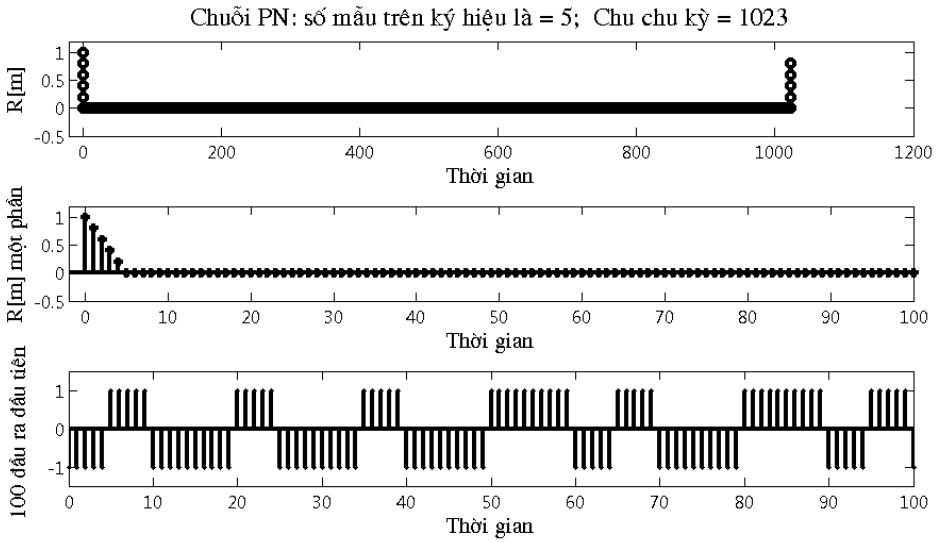


Hình 7.23: Bộ tạo chuỗi PN cho ví dụ 7.5



Hình 7.24: Biểu đồ được tạo khi lấy một mẫu trên một ký hiệu

Lưu ý rằng, chương trình thực hiện một kiểm tra để đảm bảo rằng bộ tạo chuỗi PN thực sự có chu kỳ đầy đủ (trường hợp này, chạy chương trình cho kết quả chu kỳ là 1023, là chu kỳ đầy đủ với $N = 10$). Chương trình cũng tạo ra đồ thị hàm tự tương quan, 101 mẫu đầu tiên của hàm tự tương quan, và 101 mẫu đầu tiên tại đầu ra bộ tạo. Khi lấy mẫu một mẫu/ký hiệu kết quả được minh hoạ trong hình 7.24. Thay đổi tốc độ lấy mẫu lên 5 mẫu cho kết quả trong hình 7.25.



Hình 7.25: Biểu đồ được tạo khi lấy năm mẫu trên một ký hiệu

7.9. Xử lý tín hiệu

Phần này ta nghiên cứu một số quan hệ vào/ra cho các hệ thống tuyến tính khi đầu vào/ra là ngẫu nhiên. Muốn vậy, cần quan tâm các kết quả cơ bản sau:

- Quan hệ giữa trung bình của đầu vào/ra hệ thống
- Quan hệ giữa phương sai của đầu vào/ra hệ thống
- Tương quan chéo vào/ra
- Quan hệ giữa tự tương quan và PSD của đầu vào hệ thống và đầu ra hệ thống.

Các quan hệ này rất hữu hiệu trong nghiên cứu mô phỏng, cho ta những công cụ cơ bản để xử lý tín hiệu cho trường hợp đầu vào hệ thống là một hàm mẫu của một quá trình ngẫu nhiên. Trong các phân tích dưới đây, ta coi rằng hệ thống là tuyến tính và cố định, và đầu vào hệ thống là quá trình dừng theo nghĩa rộng WSS. Bạn đọc nên tham khảo phần Phụ lục 7B1 và 7B2 có trong đĩa CD để được chi tiết hơn về xử lý các tín hiệu ngẫu nhiên.

7.9.1. Trung bình vào/ra

Vì hệ thống được giả định là tuyến tính, nên quan hệ các mẫu đầu ra *y[n]* với các mẫu đầu vào *x[n]* bởi tích chập rời rạc (tổng chập). Vì vậy, ta luôn có:

$$y[n] = \sum_{k=-\infty}^{\infty} h[k].x[n-k] \tag{7.105}$$

Theo định nghĩa, trung bình (giá trị một chiều DC) của đầu ra là:

$$E\{n[n]\} = E\left\{\sum_{k=-\infty}^{\infty} h[k].x[n-k]\right\} = \sum_{k=-\infty}^{\infty} h[k].E\{x[n-k]\} \tag{7.106}$$

Vì kỳ vọng của một tổng là tổng các kỳ vọng, và cũng cần lưu ý rằng với hệ thống cố định thì các thành phần biểu diễn đáp ứng xung kim đơn vị *h[k]* là các hằng số. Từ giả định *dừng* ta có $E\{x[n-k]\} = E\{x[n]\}$. Dẫn đến kết quả đơn giản:

$$m_y = m_x \sum_{k=-\infty}^{\infty} h[k] \tag{7.107}$$

Trong đó m_x và m_y là trung bình của *x[n]* và *y[n]*. Lưu ý rằng, vì $\sum_{k=-\infty}^{\infty} h[k] = H(0)$ nên biểu thức trên được viết là:

$$m_y = H(0).m_x \tag{7.108}$$

Trong đó H(0) là độ lợi dc của bộ lọc.

7.9.2. Tương quan chéo vào/ra

Tương quan chéo vào/ra được định nghĩa bởi:

$$R_{xy}[m] = E\{x[n].y[n+m]\} = E\left\{x[n].\sum_{j=-\infty}^{\infty} h[j].x[n+m-j]\right\} \tag{7.109}$$

$$\Leftrightarrow R_{xy}[m] = \sum_{j=-\infty}^{\infty} h[j]E\{x[n]x[n+m-j]\} \tag{7.110}$$

$$\Leftrightarrow R_{xy}[m] = \sum_{j=-\infty}^{\infty} h[j].R_{xy}[m-j] \tag{7.111}$$

Tương quan chéo vào/ra được dùng để triển khai nhiều bộ ước tính hiệu năng. Một trong số chúng là bộ ước tính tỉ số tín hiệu trên tạp âm SNR tại một điểm trong hệ thống sẽ được triển khai ở chương 8.

7.9.3. Hàm tự tương quan đầu ra

Theo định nghĩa, hàm tự tương quan đầu ra hệ thống tại trễ *m* là $E\{y[n].y[n+m]\}$. Dẫn đến:

$$R_{yy}[m] = E\{y[n].y[n+m]\} = E\left\{\sum_{j=-\infty}^{\infty} h[j].x[n-j].\sum_{k=-\infty}^{\infty} h[k].x[n+m-k]\right\} \tag{7.112}$$

$$\Leftrightarrow R_{yy}[m] = \sum_{j=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} h[j].h[k].E\{x[n-j].x[n+m-k]\} \tag{7.113}$$

Ta lại giả định *x[n]* là *dừng* để tiếp tục. Nếu *x[n]* là *dừng*, thì hàm tự tương quan $E\{x[n-j].x[n-k+m]\}$ chỉ phụ thuộc vào trễ *m-k+j* và:

$$R_{yy}[m] = \sum_{j=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} h[j].h[k].R_{xx}(m-k+j) \tag{7.114}$$

Trong đó R_{xx} *[m]* là hàm tự tương quan của *x[n]* tại trễ *m*. Đáng tiếc, ở dạng tổng quát biểu thức R_{yy} *[m]* không thể đơn giản hơn nữa nếu không biết rõ về đặc tính thống kê của x[n]. Trừ khi x[n] là chuỗi tương quan delta (trắng). Lấy biến đổi Fourier rời rạc cả 2 vế của (7.114) nhận được quan hệ PSD vào/ra là:

$$S_y(f) = S_x(f).\left|H(f)\right|^2 \tag{7.115}$$

Đã được dùng để tạo các chuỗi tương quan.

Nếu chuỗi đầu vào là tương quan delta (nghĩa là tạp âm trắng), thì theo định nghĩa:

$$R_{xx}[m] = E\{x[n].x[n+m]\} = \begin{Bmatrix} \sigma_x^2 & khi\ m = 0 \\ 0 & khi\ \ m \neq 0 \end{Bmatrix} = \sigma_x^2.\delta[m] \tag{7.116}$$

Thay vào (7.114) ta được:

$$R_{yy}[m] = \sigma_x^2 \sum_{j=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} h[j].h[k].\delta[m-k+j] = \sigma_x^2 \sum_{j=-\infty}^{\infty} h[j].h[m+j] \tag{7.117}$$

Trong đó, ta đã dùng tính chất dịch của hàm deta

7.9.4. Phương sai vào/ra

Theo định nghĩa, phương sai đầu ra hệ thống là $R_{yy}[0] = E\{y^2[n]\}$. Trường hợp tổng quát thì phương sai đầu ra tuân theo (7.114) với $m = 0$. Kết quả là:

$$\sigma_y^2 = \sum_{j=-\infty}^{\infty}\sum_{k=-\infty}^{\infty} h[j].h[k].R_{xx}[j-k] \tag{7.118}$$

Nếu *x[n]* là chuỗi tương quan delta (tạp âm trắng), thì tìm được σ_y^2 bằng cách đặt $m = 0$ trong (7.117). Ta có:

$$\sigma_y^2 = \sigma_x^2 \sum_{j=-\infty}^{\infty} h^2[j] \tag{7.119}$$

Kết quả này sẽ được dùng khi nghiên cứu băng thông tạp âm tương đương trong chương 10.