


## Mô phỏng tín hiệu và quá trình thu phát

- Giới thiệu
- Mô phỏng nguồn tín hiệu
- Mã hóa
- Điều chế và giải điều chế
- Quá trình lọc
- Quá trình đồng bộ


105



## Giới thiệu

- *Mô phỏng tín hiệu băng gốc và thông dải:*
  - Tín hiệu băng gốc (baseband): có phổ tần tập trung quanh tần số 0.
  - Tín hiệu thông dải (passband): có phổ tần tập trung quanh một tần số sóng mang  $f_c$ .
  - Tín hiệu băng gốc có thể được chuyển đổi thành tín hiệu thông dải qua quá trình đổi tần lên (up-conversion)
  - Tín hiệu thông dải có thể được chuyển đổi thành tín hiệu băng gốc qua quá trình đổi tần xuống (down-conversion)
  - Tín hiệu thông dải  $s_p(t)$  được xây dựng từ hai tín hiệu băng gốc  $s_I(t)$  và  $s_Q(t)$  (trong điều chế số)




106




## Giới thiệu

- **Mô phỏng tín hiệu bằng gốc và thông dải:**
  - Tín hiệu thông dải có thể được viết:
 
$$s_P(t) = \sqrt{2} \cdot A_{s_I}(t) \cdot \cos(2\pi f_c t) + \sqrt{2} \cdot A_{s_Q}(t) \cdot \sin(2\pi f_c t).$$
  - Định nghĩa tín hiệu  $s(t)$ :  $s(t) = s_I(t) - j \cdot s_Q(t)$ ,  $\rightarrow$  tín hiệu  $s_P(t)$  có thể viết lại
 
$$s_P(t) = \sqrt{2} \cdot A \cdot \Re\{s(t) \cdot \exp(j2\pi f_c t)\}.$$
  - Tín hiệu  $s(t)$ :
    - Được gọi là tín hiệu tương đương băng gốc hoặc lớp vỏ phức của tín hiệu thông dải  $s_P(t)$
    - Chứa cùng thông tin như  $s_P(t)$
    - $s(t)$  là tín hiệu phức

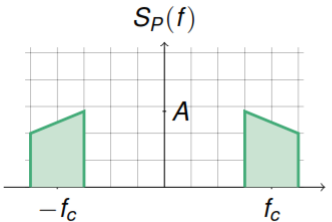
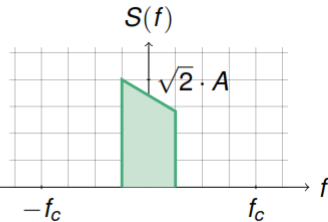
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

107



## Giới thiệu

- **Mô phỏng tín hiệu bằng gốc và thông dải:**
  - Trong miền tần số:
 
$$S(f) = \begin{cases} \sqrt{2} \cdot S_P(f + f_c) & \text{for } f + f_c > 0 \\ 0 & \text{else.} \end{cases}$$
    - Hệ số  $\sqrt{2}$  đảm bảo cả hai loại tín hiệu có cùng mức công suất.

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

108

**Giới thiệu**

- Mô phỏng tín hiệu băng gốc và thông dải:
  - Mô hình thông dải:

Passband model

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

109

**Giới thiệu**

- Mô phỏng tín hiệu băng gốc và thông dải:
  - Mô hình tương đương thông thấp:

Baseband model

$$r_{RF}(t) = \text{Re}\{r(t) e^{j2\pi f_c t}\} \quad c_{RF}(\tau; t) = 2\text{Re}\{c(\tau; t) e^{j2\pi f_c t}\}$$

$$s_{RF}(t) = \text{Re}\{s(t) e^{j2\pi f_c t}\} \quad n_{RF}(t) = \text{Re}\{n(t) e^{j2\pi f_c t}\}$$

- Thu được hệ thống tương đương băng gốc:  
sử dụng các tín hiệu băng gốc:
  - Tín hiệu phát tương đương băng gốc:
  - Kênh tương đương băng gốc với đáp ứng xung kim giá trị phức  $h(t)$  với  $h_P(t) = \Re\{h(t) \cdot \exp(j2\pi f_c t)\}$
  - Tín hiệu thu tương đương băng gốc:  $R(t)$
  - Nhiều Gaussian cộng giá trị phức:  $N(t)$

$$s(t) = s_I(t) - j \cdot s_Q(t).$$

$$R(t) = R_I(t) - j \cdot R_Q(t).$$

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

110




## Giới thiệu

- **Mô phỏng tín hiệu bằng gốc và thông dải:**
  - Mô hình thông dải:
    - Các tín hiệu là thực
    - Sát với hệ thống thực
    - Tần số lấy mẫu cao hơn
  - Mô hình tương đương thông thấp:
    - Các tín hiệu là phức
    - Mô hình gọn và đơn giản hơn
    - Tần số lấy mẫu thấp hơn
    - Trong các trường hợp thực tế, xử lý tín hiệu số được thực hiện trên tín hiệu được chuyển đổi bằng gốc.
  - Mô hình tương đương bằng gốc là thuận tiện hơn trong mô phỏng hệ thống.
    - Hệ thống tuyến tính:  $R(t) = s(t) * h(t) + n(t)$  and  $R(f) = S(f) \cdot H(f) + N(f)$ .

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

111



## Giới thiệu

- **Quá trình lấy mẫu và nội suy:**
  - Trong mô phỏng hệ thống truyền tin trên hệ thống máy tính số đòi hỏi sự chuyển đổi mô hình thời gian liên tục thành mô hình rời rạc về thời gian.
  - Theo định lý lấy mẫu Nyquist (hoặc Shannon): nếu  $B_s$  là độ rộng băng tần của tín hiệu băng gốc  $s(t) \rightarrow$  tần số lấy mẫu  $f_s \geq 2B_s$ .
  - Quá trình lấy mẫu:  $s(t) \rightarrow s_s(t) = s(nT_s)$ 

$$s_s(t) = s(t)p(t) \text{ với } p(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT_s) \rightarrow s_s(t) = \sum_{n=-\infty}^{\infty} s(nT_s)\delta(t - nT_s)$$
 trong đó:  $T_s$  – chu kỳ lấy mẫu,  $f_s = 1/T_s$  – tần số lấy mẫu
  - Tần số lấy mẫu được lựa chọn phù hợp để giảm thiểu lỗi chồng phổ mà tránh tăng thời gian mô phỏng.

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

112




## Giới thiệu

- **Quá trình lấy mẫu và nội suy:**
  - Trong một số trường hợp mô phỏng hệ thống trên các độ rộng băng tần khác nhau → chuyển đổi tốc độ mẫu
    - Tăng mẫu (upsampling): tại biên giữa phần tín hiệu băng hẹp và băng rộng  
 $s(kT_s) \rightarrow s(kT_u) = s(kT_s/M)$
    - Giảm mẫu (downsampling): tại biên giữa phần tín hiệu băng rộng và băng hẹp  
 $s(kT_s) \rightarrow s(kT_d) = s(kMT_s)$
  - Quá trình nội suy: quan trọng trong kỹ thuật đa tốc độ
    - Bộ nội suy hàm sinc
    - Bộ nội suy tuyến tính

Trong MATLAB sử dụng hàm *interp*:

$y = \text{interp}(x,r)$  thực hiện lấy lại mẫu giá trị trong vector  $x$  tại  $r$  lần tốc độ lấy mẫu ban đầu.

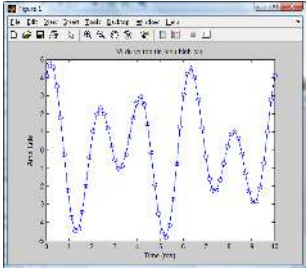
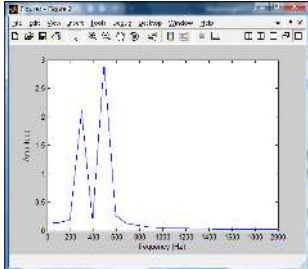

113




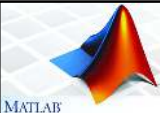
## Mô phỏng nguồn tín hiệu

- **Nguồn tín hiệu tương tự:**
  - Tín hiệu đơn tần:
 
$$x(t_k) = A \cos(2\pi f_0 t_k + \varphi) \quad \text{hoặc} \quad \tilde{x}(k) = A \exp(2\pi j k f_0 / f_s) \exp(j\varphi)$$
  - Tín hiệu đa tần:
 
$$x(t_k) = \sum_{n=1}^M x_n(t_k) \quad \text{với} \quad x_n(t_k) = A_n \cos(2\pi f_n t_k + \varphi_n)$$

$$\tilde{x}(k) = \sum_{n=1}^M A_n \exp(2\pi j k f_n / f_s) \exp(j\varphi_n)$$


114




## Mô phỏng nguồn tín hiệu

- **Nguồn tín hiệu số:**
  - Nguồn thông tin rời rạc thường có giá trị trong bảng alphabet.
  - Tín hiệu số: là dạng sóng mang thông tin số.
  - Có 3 tham số chính:
    - Kiểu nguồn (alphabet): danh sách các ký hiệu thông tin có thể mà nguồn tạo ra.
      - VD:  $A = \{0, 1\}$ ; các ký hiệu được gọi là bit
      - Với  $M$  ký hiệu ( $M = 2^n$ ):  $A = \{0, 1, \dots, M-1\}$  hoặc  $A = \{\pm 1, \pm 3, \dots, \pm(M-1)\}$
      - Các symbol có thể có giá trị phức:  $A = \{\pm 1, \pm j\}$
    - Xác suất ưu tiên phát: tần số xuất hiện tương đối của mỗi ký hiệu mà nguồn tạo ra.
      - VD: nguồn sinh ra các bit 0 và 1 có xác suất bằng nhau  $\pi_0 = \pi_1 = \frac{1}{2}$ .
    - Tốc độ ký hiệu (symbol rate): số lượng ký hiệu thông tin mà nguồn sinh ra trong một đơn vị thời gian (baud rate)
 
$$R_b = R \cdot \log_2(M).$$

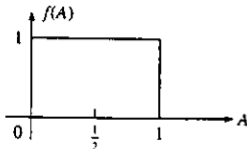
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

115

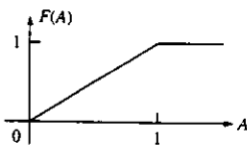


## Mô phỏng nguồn tín hiệu

- **Nguồn tín hiệu ngẫu nhiên:**
  - Các nguồn tin trong thực tế là ngẫu nhiên  $\rightarrow$  tạo các tín hiệu ngẫu nhiên trong mô phỏng.
  - Tạo biến ngẫu nhiên phân bố đều: .



(a)



(b)

- Sử dụng hàm *rand* trong MATLAB

```
>> x = rand(5,10) - Tạo ma trận 5x10 các số ngẫu nhiên phân bố đều trong khoảng [0,1]
Tạo các số ngẫu nhiên phân bố đều trong khoảng [a, b] :
>> x = a + (b-a) * rand(m,n)
Tạo các số nguyên ngẫu nhiên phân bố đều trên tập 1:n :
>> x = ceil(n.*rand(100,1));
```

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

116



## Mô phỏng nguồn tín hiệu

- **Nguồn tín hiệu ngẫu nhiên:**
  - Tạo biến ngẫu nhiên phân bố đều: .

Vi dụ: Tạo vector hàng 1000 số ngẫu nhiên phân bố đều trong khoảng [0,1], hiển thị 10 số đầu tiên

```
>> x = rand(1,1000);
>> x(1:10)
```

ans =


0.4330 0.8424 0.1845 0.5082 0.4522 0.3256 0.3801 0.8865 0.7613 0.8838

```
>> hist(x,10)
```



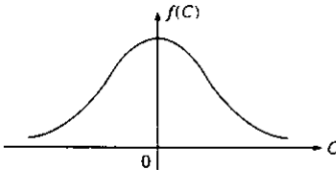
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**  
 Posts & Telecommunications Institute of Technology

117

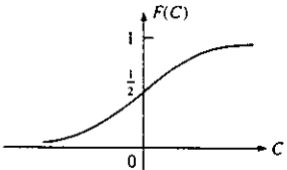


## Mô phỏng nguồn tín hiệu

- **Nguồn tín hiệu ngẫu nhiên:**
  - Tạo biến ngẫu nhiên phân bố chuẩn:



(a)



(b)

- Sử dụng hàm *randn* trong MATLAB .

Tạo các số ngẫu nhiên phân bố chuẩn có trung bình bằng 0 và độ lệch chuẩn bằng 1 :

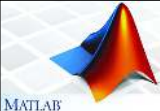
```
>> x = randn(m,n)
```

Tạo các số ngẫu nhiên phân bố chuẩn có trung bình bằng m và phương sai v :

```
>> x = m + sqrt(v) * randn(m,n)
```

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**  
 Posts & Telecommunications Institute of Technology

118



## Mô phỏng nguồn tín hiệu

- **Nguồn tín hiệu ngẫu nhiên:**
  - Tạo biến ngẫu nhiên phân bố chuẩn:
 

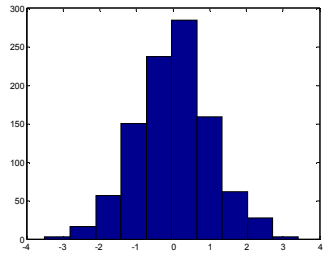
Vi dụ: Tạo vector hàng 1000 số ngẫu nhiên phân bố chuẩn có trung bình 0 và độ lệch chuẩn bằng 1, hiển thị 10 số đầu tiên

```
>> x = randn(1,1000);
>> x(1:10)
```

ans =

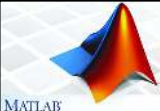
-0.6028 -0.9934 1.1889 2.3880 2.2655 2.3011 -0.2701 0.5028 -0.1192 -0.0019

```
>> hist(x,10)
```



HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

119



## Mô phỏng nguồn tín hiệu

- **Nguồn tín hiệu ngẫu nhiên:**
  - Tạo số nguyên ngẫu nhiên phân bố đều:
    - Sử dụng hàm *randint* trong MATLAB:

Tạo ma trận mxn các số 0 và 1 có xác suất bằng nhau

```
>> x = randint(m,n);
```

Vi dụ:

```
>> x = randint(1,10)
```

x =

0 0 1 1 1 0 1 1 0 0

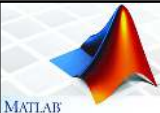
Tạo ma trận mxn có các giá trị phân bố đều trong dải từ 0 đến 7

```
>> x = randint(m, n, [0, 7]); hoặc
>> x = randint(m,n, 8);
```

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

120





## Mô phỏng nguồn tín hiệu

- **Nguồn tín hiệu ngẫu nhiên:**
  - Tạo symbol ngẫu nhiên theo danh sách alphabet định trước:
    - Sử dụng hàm `randsrc` trong MATLAB:

Tạo ma trận mxn các số -1 và 1 có xác suất bằng nhau

```
>> x = randsrc(m,n);
```

Ví dụ:

```
>> x = randsrc(1,10)
```

x =

```
-1  1  1 -1 -1 -1 -1  1  1  1
```

Tạo ma trận mxn có các giá trị phân bố đều trong tập {-3,-1,1,3}


```
>> x = randsrc(10,10,[-3 -1 1 3]); hoặc
```

```
>> x = randsrc(10,10,[-3 -1 1 3; .25 .25 .25 .25]);
```

- Tạo nguồn lỗi ngẫu nhiên:
  - Sử dụng hàm `randerr` trong MATLAB

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

121



## Mã hóa

- **Mã hóa nguồn:**
  - Quá trình chuyển đổi A/D:
 

Analog  
signal

→

Sampler

→

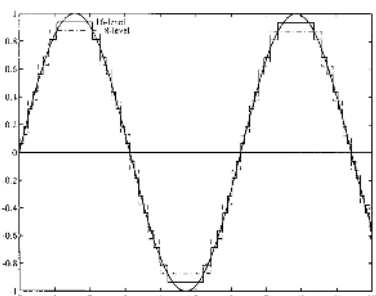
Quantizer

→

Coder

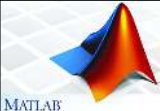
→ Digital  
signal

sampling
quantization
coding
  - Quá trình PCM:
    - Lượng tử hóa đều
    - Lượng tử hóa không đều



HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

122



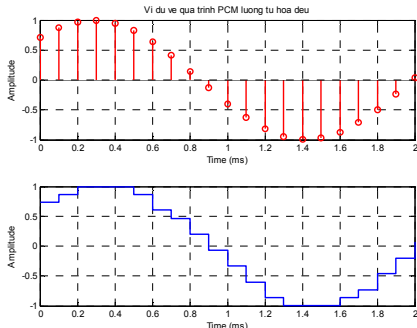
# Mã hóa

- **Mã hóa nguồn:**
  - MATLAB code cho quá trình PCM lượng tử hóa đều:
 

```
function [code,xq,sqnr] = uniform_pcm(x,M)
% Uniform PCM encoding of a sequence
% x = input sequence
% M = number of quantization levels
% code = the encoded output
% xq = quantized sequence before encoding
% sqnr = signal to quantization noise ratio in dB
% Written by Nguyen Duc Nhan - 2012

Nb = log2(M);
Amax = max(abs(x));
delta = 2*Amax/(M-1);
Mq = -Amax:delta:Amax;
Ml = 0:M-1;

xq = zeros(size(x));
xcode = xq;
for k = 1:M
    ind = find(x > Mq(k)-delta/2 & x <= Mq(k)+delta/2);
    xq(ind) = Mq(k);
    xcode(ind) = Ml(k);
end
sqnr = 20*log10(norm(x)/norm(x-xq));
code = de2bi(xcode,Nb,'left-msb');
```



code =

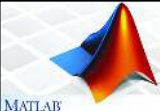
1	1	0	1
1	1	1	0
1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

sqnr = 25.4136 dB

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

Posts & Telecommunications Institute of Technology

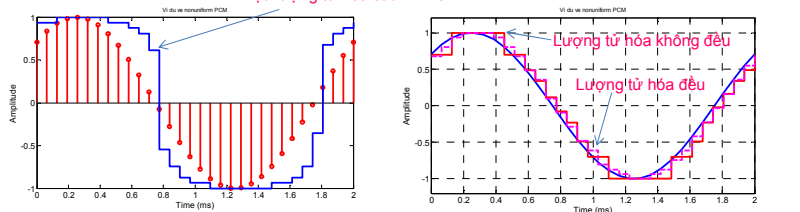
123



# Mã hóa

- **Mã hóa nguồn:**
  - Quá trình PCM lượng tử hóa không đều: Ví dụ theo luật  $\mu$ 

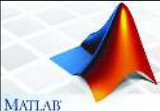
```
%% nonuniform PCM process
mu = 255;
M = 32; % number of quantization levels
[y,amax] = mulaw(x,mu); % compress the signal
[code,yq,sqnr] = uniform_pcm(y,M); % coding
xq = invmulaw(yq,mu); % expand the signal
xq = xq*amax;
sqnr = 20*log10(norm(x)/norm(x-xq)); % in dB
```



HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG


Posts & Telecommunications Institute of Technology

124



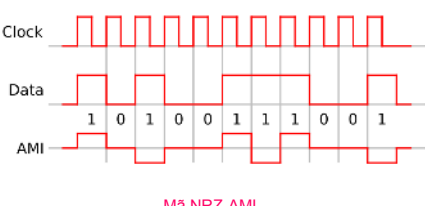
# Mã hóa

- **Mã đường:**
  - Kiểu mã hóa để tạo dạng phổ và một số đặc tính xác định của xung tín hiệu hỗ trợ cho quá trình đồng bộ.
  - Gồm 2 bước:
    - Sắp xếp logic
    - Chuyển đổi thành dạng sóng
  - Ví dụ:
 




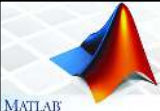
Mã non-return-to-zero (NRZ)

Mã Manchester



Mã NRZ-AMI

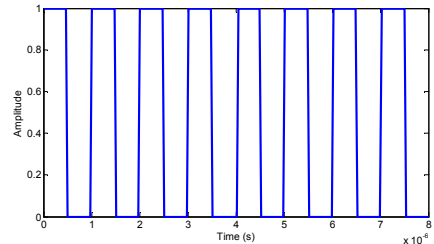

125



# Mã hóa

- **Mã đường:**
  - Tạo chuỗi xung vuông:
    - Trong mỗi chu kỳ xung, hàm xung vuông được định nghĩa như sau:
 
$$u(t) = \begin{cases} 1, & t \leq T_p \\ 0, & t > T_p \end{cases}$$


```
>> [t,y]=rectpulse(0.5e-6,1e6,256,8);
>> plot(t,y);
```

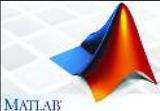


```
function [t,y] = rectpulse(Tw,Rp,Ns,Np)
% Chương trình ví dụ tạo chuỗi xung vuông
% Tw - the pulsewidth
% Rp - the repetition rate of pulse Tp < 1/Rp
% Ns - the number of samples
% Np - the number of pulses (the length of pulse train)
% t - the time vector output
% y - the vector output of the pulse samples
% written by Nguyen Duc Nhan

Tp = 1/Rp; % pulse period
Timewindow = Np*Tp; % time window
ts = Timewindow/(Ns-1); % sampling time
t = 0:ts:Timewindow; % time vector
Nsp = round(Tp/ts); % number of samples within Tp

y = zeros(size(t));
for k = 1:Ns
    if mod(t(k),Nsp*ts) <= Tw
        y(k) = 1;
    else
        y(k) = 0;
    end
end
```


126

 **Mã hóa**

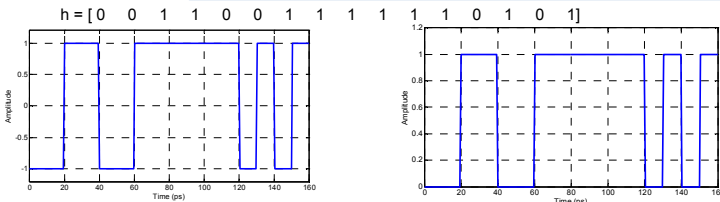
- Mã đường:**
  - Mã NRZ:**

```
function [t,y,code] = nrzcode(d,R,Ns,type)
% Chương trình ví dụ vẽ mã đường truyền NRZ
% d - the data sequence
% R - the data rate
% Ns - the number of samples
% t - the time vector output
% y - the vector output of the pulse samples
% type - the type of code (unipolar - 'unipol' or polar - 'pol')
% written by Nguyen Duc Nhan

Tb = 1/R;           % bit period
Nb = length(d);     % number of bits
Timewindow = Nb*Tb; % time window
ts = Timewindow/(Ns-1); % sampling time
t = 0:ts:Timewindow; % time vector
y = zeros(size(t));
code = [];

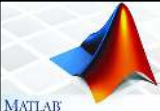
if nargin <= 3
    type = 'unipol';
end
for k = 1:Ns
    n = fix(t(k)/Tb)+1;
    if n >= Nb
        n = Nb;
    end
    switch (type)
        case 'unipol'
            y(k) = d(n);
            code(n) = d(n);
        case 'pol'
            y(k) = 2*d(n)-1;
            code(n) = 2*d(n)-1;
        end
    end
end
```

h = [0 0 1 1 0 0 1 1 1 1 1 0 1 0 1]



HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

127

 **Mã hóa**

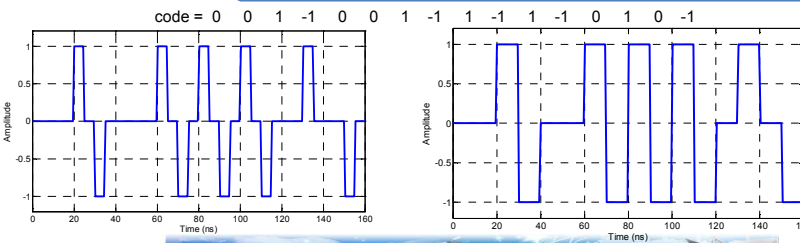
- Mã đường:**
  - Mã AMI:**

```
function [t,y,code] = amicode(d,R,Ns,type)
% Chương trình ví dụ vẽ mã AMI
% d - the data sequence
% R - the data rate
% Ns - the number of samples
% t - the time vector output
% y - the vector output of the pulse samples
% type - the type of code (NRZ - 'NRZ' or RZ - 'RZ')
% written by Nguyen Duc Nhan
...

y = zeros(size(t));
code = [];
...

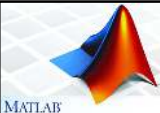
s = 1;
for k = 1:Nb
    if d(k) == 0
        code(k) = 0;
    else
        s = s+1;
        if mod(s,2)==0
            code(k) = 1;
        else
            code(k) = -1;
        end
    end
end
end
```

code = 0 0 1 -1 0 0 1 -1 0 1 -1 0 1 0 -1



HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

128




# Mã hóa


- **Mã hóa kênh:**
  - Tăng hiệu năng của kênh truyền:
    - Phát hiện lỗi
    - Sửa lỗi
  - Gồm 2 loại chính:
    - Mã khối
    - Mã xoắn
  - Ví dụ mã khối:
    - Mã hóa: Từ mã  $c = uG$

Trong đó:  $u$  – chuỗi dữ liệu có  $k$  bit,  $G$  – ma trận tạo mã cỡ  $k \times n$ ,  $c$  – từ mã được mã hóa có  $n$  bit ( $n > k$ )

- Khoảng cách Hamming tối thiểu:  $d_{\min} = \min_{i \neq j} d_H(c_i, c_j)$
- Đối với mã khối tuyến tính: khoảng cách tối thiểu bằng với trọng số nhỏ nhất của mã

$$w_{\min} = \min_{c_i \neq 0} w(c_i)$$


129




# Mã hóa

- **Mã hóa kênh:**
  - Ví dụ mã khối:

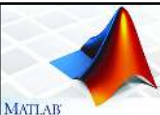
```

% Chương trình ví dụ về mã hóa kênh
% Block coding
k = 4;
for i=1:2^4
    for j=k:-1:1
        if rem(i-1,2^(-j+k+1))>=2^(-j+k)
            u(i,j)=1;
        else
            u(i,j)=0;
        end
    end
end
% Define G, the generator matrix
g = [1 0 0 1 1 1 0 1 1 1;
     1 1 0 0 0 1 1 1 0;
     0 1 1 0 1 1 0 1 0 1;
     1 1 0 1 1 1 0 0 1];
% generate codewords
c = rem(u*g,2);
% find the minimum distance
w_min = min(sum((c(2:2^k,:))));
  
```

u =	c =
0 0 0 0	0 0 0 0 0 0 0 0 0 0
0 0 0 1	1 1 0 1 1 1 1 0 0 1
0 0 1 0	0 1 1 0 1 1 0 1 0 1
0 0 1 1	1 0 1 1 0 0 0 1 1 0 0
0 1 0 0	1 1 1 0 0 0 0 1 1 1 0
0 1 0 1	0 0 1 1 1 1 0 1 1 1
0 1 1 0	1 0 0 0 1 1 1 0 1 1
0 1 1 1	0 1 0 1 0 0 0 0 0 1 0
1 0 0 0	1 0 0 1 1 1 0 1 1 1
1 0 0 1	0 1 0 0 0 0 0 1 1 1 0
1 0 1 0	1 1 1 1 0 0 0 0 0 1 0
1 0 1 1	0 0 1 0 1 1 1 0 1 1
1 1 0 0	0 1 1 1 1 1 1 0 0 1
1 1 0 1	1 0 1 0 0 0 0 0 0 0 0
1 1 1 0	0 0 0 1 0 0 0 1 1 0 0
1 1 1 1	1 1 0 0 1 1 0 1 0 1 0



130



## Điều chế và giải điều chế

- **Điều chế tín hiệu tương tự:**
  - Điều chế biên độ AM:
    - Tín hiệu bản tin:  $m(t) = M \cdot \cos(\omega_m t + \phi)$ .
    - Tín hiệu sóng mang:  $c(t) = A \cdot \sin(\omega_c t + \phi_c)$ ,
    - Quá trình điều chế: là một quá trình nhân
 
$$y(t) = [1 + m(t)] \cdot c(t),$$

$$= A \cdot [1 + M \cdot \cos(\omega_m t + \phi)] \cdot \sin(\omega_c t).$$
    - Độ sâu điều chế (modulation index):
 
$$h = \frac{\text{peak value of } m(t)}{A} = \frac{M}{A},$$

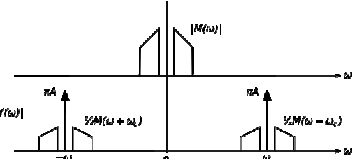
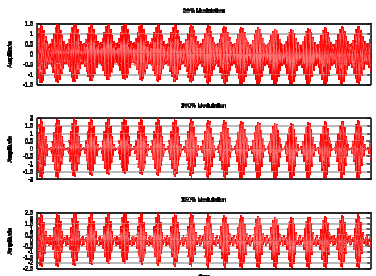
$$m(t) \xleftrightarrow{F} M(\omega)$$

$$\sin(\omega_c t) \xleftrightarrow{F} i\pi \cdot [\delta(\omega + \omega_c) - \delta(\omega - \omega_c)]$$

$$A \cdot \sin(\omega_c t) \xleftrightarrow{F} i\pi A \cdot [\delta(\omega + \omega_c) - \delta(\omega - \omega_c)]$$

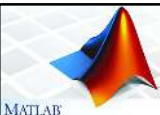
$$m(t) \cdot A \cdot \sin(\omega_c t) \xleftrightarrow{F} \frac{A}{2\pi} \cdot \{M(\omega) * [i\pi \cdot (\delta(\omega + \omega_c) - \delta(\omega - \omega_c))]\}$$

$$= \frac{iA}{2} \cdot [M(\omega + \omega_c) - M(\omega - \omega_c)]$$

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**  
 Posts & Telecommunications Institute of Technology

131

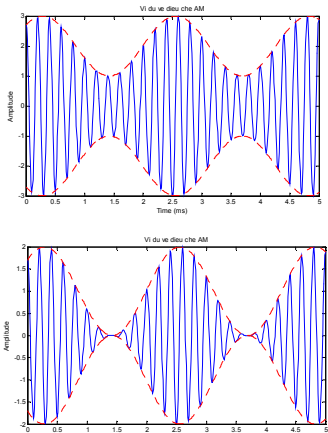


## Điều chế và giải điều chế

- **Điều chế tín hiệu tương tự:**
  - Điều chế biên độ AM:
 

M = 50%

M = 100%



```

% Chương trình ví dụ về điều chế AM
%% Set parameters
% Message
A = 1;           % amplitude
f = 440;         % frequency [Hz]
phi = -pi/4;     % Phase [rad]

% Carrier
m = 0.5;         % modulation index
Ac = A/m;        % amplitude
fc = 5e3;        % frequency [Hz]
phi_c = 0;       % Phase [rad]

N = 2^9;         % number of samples
T0 = 0;          % start time [s]
Tf = 5e-3;       % end time [s]
Ts = (Tf-T0)/(N-1); % sampling period
fs = 1/Ts;       % sampling frequency [Hz]

%% Amplitude Modulation
% Generate sinusoid
t = T0:Ts:Tf;    % time vector
x = A*cos(2*pi*f*t+phi); % message signal
xc = Ac*cos(2*pi*fc*t+phi_c); % carrier signal

% Modulation
y = (1+x/Ac).*xc;
          
```

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**  
 Posts & Telecommunications Institute of Technology

132



## Điều chế và giải điều chế

- Điều chế tín hiệu tương tự:
  - Điều chế biên độ AM:
    - Điều chế SSB (Single side band): Sử dụng bộ lọc hoặc dùng khai triển Hilbert

```

%% SSB Modulation
% Generate sinusoid
t = T0:Ts:Tf;
x = A*cos(2*pi*f*t+phi);
% Modulation
y = ssbmod(x,fc,fs,phic);
% Demodulation
xr = ssbdemod(y,fc,fs,phic);

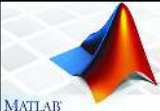
```



Ví dụ về điều chế SSB

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**  
 Posts & Telecommunications Institute of Technology

133



## Điều chế và giải điều chế

- Điều chế tín hiệu tương tự:
  - Điều chế tần số FM:
 

$$y(t) = A_c \cos \left( 2\pi \int_0^t f(\tau) d\tau \right)$$

$$= A_c \cos \left( 2\pi \int_0^t [f_c + f_\Delta x_m(\tau)] d\tau \right)$$

$$= A_c \cos \left( 2\pi f_c t + 2\pi f_\Delta \int_0^t x_m(\tau) d\tau \right)$$

$$y(t) = A_c \cos \left( 2\pi f_c t + \frac{f_\Delta}{f_m} \cos(2\pi f_m t) \right)$$

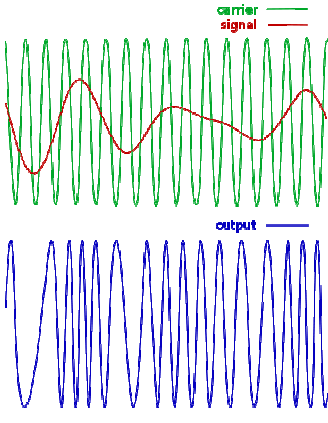
Chỉ số điều chế:  $\beta = \frac{\Delta f}{f_m} = \frac{f_\Delta |x_m(t)|}{f_m}$

Độ rộng băng tần:  $B_T = 2(\Delta f + f_m)$

```

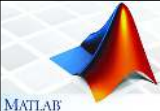
% Modulation
y = pmmod(x,fc,fs,phic);
% Demodulation
xr = pmdemod(y,fc,fs,phic);

```



**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**  
 Posts & Telecommunications Institute of Technology

134



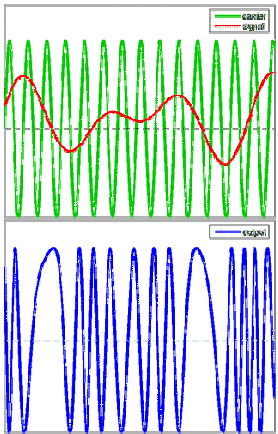
## Điều chế và giải điều chế

- **Điều chế tín hiệu tương tự:**
  - Điều chế pha PM:
    - Sóng mang:  $c(t) = A_c \sin(\omega_c t + \phi_c)$ .
    - Tín hiệu điều chế:  $y(t) = A_c \sin(\omega_c t + m(t) + \phi_c)$ .

Chỉ số điều chế:  $2(l+1)f_M$   
 $f_M = \omega_m / 2\pi$   
 $f_k = \Delta\theta$


- Trong MATLAB sử dụng các hàm *pmmmod* và *pmdemod* cho điều chế PM.

```
%% Phase Modulation
t = T0:Ts:Tf;
x = A*cos(2*pi*f*t+phi);
% Modulation
y = pmmmod(x,fc,fs,phic);
% Demodulation
xr = pmdemod(y,fc,fs,phic);
```



HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

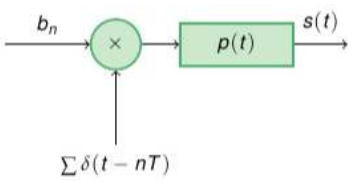
135



## Điều chế và giải điều chế

- **Điều chế tín hiệu số:**
  - Điều chế tuyến tính:
    - Được biểu diễn bởi:
 
$$s(t) = \sum_{n=0}^{N-1} b_n \cdot p(t - nT)$$

$s(t)$  là tuyến tính với  $b_n$

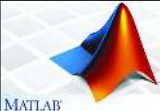


- Các định dạng điều chế khác nhau được xây dựng bằng việc chọn các danh sách alphabet phù hợp
  - ▶ **BPSK:**  $b_n \in \{1, -1\}$
  - ▶ **OOK:**  $b_n \in \{0, 1\}$
  - ▶ **PAM:**  $b_n \in \{\pm 1, \dots, \pm(M-1)\}$ .
- Hàm  $p(t)$  xác định dạng xung đầu ra tín hiệu được điều chế
  - $p(t)$  có thể là hàm xung vuông hoặc hàm xung sinc

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

136



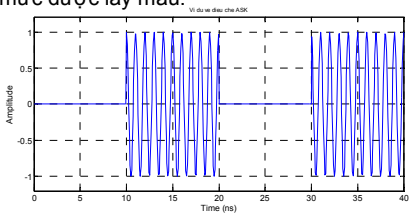


## Điều chế và giải điều chế

- **Điều chế tín hiệu số:**
  - Điều chế biên độ ASK:
    - Sử dụng hàm *pammod* và *pamdmod* trong MATLAB → tạo ra ký hiệu phức tương đương bằng gốc.
 

```
% Signal generator
dm = randint(1,1000,4);
% PAM modulation
s = pammod(dm,4);
% PAM demodulation
r = pamdmod(s,4);
```
    - Để biểu diễn dạng sóng điều chế có sóng mang có thể sử dụng hàm *ammod* trong điều chế AM với các mức được lấy mẫu:
 


```
% Data sequence
h = [0 1 0 1];
% NRZ modulation
[t,y,code]=nrzcode(h,1e6,512);
% AM demodulation
ts = t(2)-t(1); % sampling time
fs = 1/ts; % sampling freq.
fc = 10e6; % carrier freq.
yd = ammod(y,fc,fs);
```



Vẽ tín hiệu điều chế ASK

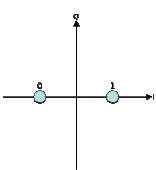
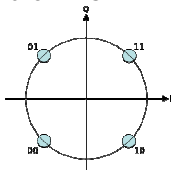
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
 Posts & Telecommunications Institute of Technology

137



## Điều chế và giải điều chế

- **Điều chế tín hiệu số:**
  - Điều chế pha PSK:
    - Điều chế M-PSK:
 

BPSK

Tín hiệu BPSK:

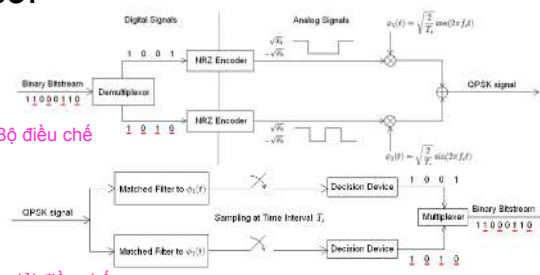
$$s_n(t) = \sqrt{\frac{2E_b}{T_b}} \cos(2\pi f_c t + \pi(1-n)), n = 0, 1.$$

QPSK

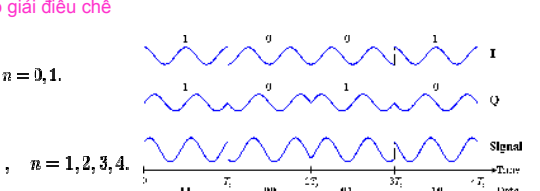
Tín hiệu QPSK:

$$s_n(t) = \sqrt{\frac{2E_b}{T_b}} \cos\left(2\pi f_c t - (2n-1)\frac{\pi}{4}\right), n = 1, 2, 3, 4.$$

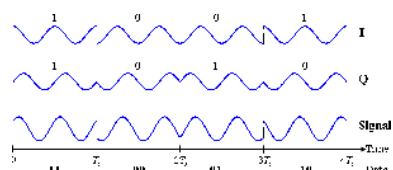
Bộ điều chế



Bộ giải điều chế



Dạng sóng QPSK:



HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
 Posts & Telecommunications Institute of Technology

138

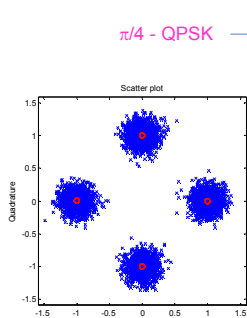
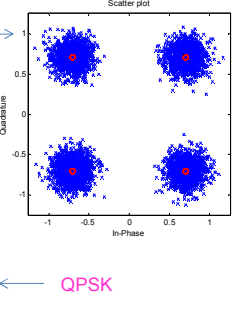
## Điều chế và giải điều chế

- **Điều chế tín hiệu số:**
  - Điều chế pha PSK:
    - Trong MATLAB sử dụng hàm *pskmod* và *pskdemod* cho điều chế PSK kết hợp với *modem* để xây dựng object của bộ điều chế và giải điều chế:

```
% Create a random digital message
M = 4; % Alphabet size
x = randint(5000,1,M); % Message generator
% Use QPSK modulation to produce y.
h = modem.pskmod(M,pi/4);
h.symbolorder = 'gray';
y = modulate(h,x);

% Transmit signal through an AWGN channel.
ynoisyy = awgn(y,15,'measured');

% Create scatter plot from noisy data.
h = scatterplot(ynoisyy,1,0,'xb');
hold on;
scatterplot(y,1,0,'or',h);
% Demodulate ynoisyy to recover the message.
h = modem.pskdemod(M,pi/4);
h.symbolorder = 'gray';
z = demodulate(h,ynoisyy);
```

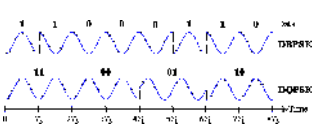
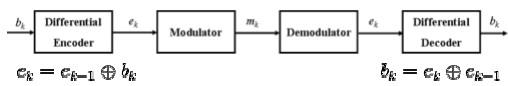



HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

139

## Điều chế và giải điều chế

- **Điều chế tín hiệu số:**
  - Điều chế pha PSK:
    - Điều chế PSK mã hóa vi sai: sử dụng hàm *dpskmod* và *dpskdemod*

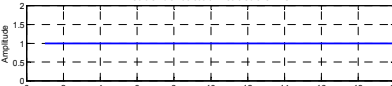

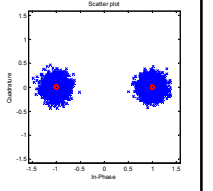
```
% Create a random digital message
M = 2; % Alphabet size
x = randint(5000,1,M); % Message
% Use DPSK modulation to produce y.
y = dpskmod(x,M);

% Demodulate to recover the message.
z = dpskdemod(y,M);
```

$$c_k = c_{k-1} \oplus b_k$$

$$\hat{b}_k = c_k \oplus c_{k-1}$$

x = 0 1 1 0 0 0 1 0 0 0 0 1 0 1 1 1 0 1 0 1

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

140

**Điều chế và giải điều chế**

- **Điều chế tín hiệu số:**
  - Điều chế QAM: kết hợp giữa ASK và PSK

**BỘ ĐIỀU CHẾ**

**BỘ GIẢI ĐIỀU CHẾ**

**Rectangular QAM**

**Circular QAM**

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

141

**Điều chế và giải điều chế**

- **Điều chế tín hiệu số:**
  - Điều chế QAM:
    - Sử dụng hàm *qammod* và *qamdemod* trong MATLAB

```
close all;
% Create a random digital message
M = 16; % Alphabet size
x = randint(5000,1,M);

% Use 16-QAM modulation to produce y.
y = modulate(modem.qammod(M),x);

% Transmit signal through an AWGN channel.
ynoisyy = awgn(y,15,'measured');

% Create scatter plot from noisy data.
h = scatterplot(ynoisyy,1,0,'xb');
hold on;
scatterplot(y,1,0,'or',h); hold off;
```

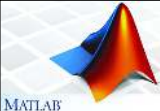
**Biểu diễn biên độ của tín hiệu điều chế 16-QAM**

**Biểu diễn biên độ của tín hiệu điều chế 16-QAM**

**Biểu diễn biên độ của tín hiệu điều chế 16-QAM**

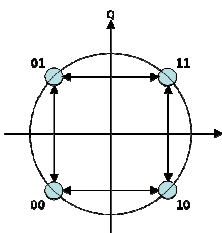
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

142



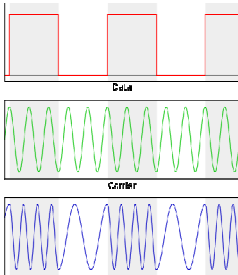
## Điều chế và giải điều chế

- Điều chế tín hiệu số:
  - Điều chế FSK:
    - Điều chế FSK
    - Điều chế FSK pha liên tục (CPFSK)
    - Điều chế MSK: một kiểu CPFSK



$$s(t) = \cos[2\pi f_c t + b_k(t) \frac{\pi t}{2T} + \phi_k]$$

$$s(t) = a_I(t) \cos\left(\frac{\pi t}{2T}\right) \cos(2\pi f_c t) - a_Q(t) \sin\left(\frac{\pi t}{2T}\right) \sin(2\pi f_c t)$$



- Sử dụng hàm *fskmod* và *fskdemod* cho điều chế FSK
- Sử dụng hàm *mskmod* và *fskdemod* cho điều chế MSK

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

143



## Điều chế và giải điều chế

- Điều chế tín hiệu số:
  - Điều chế FSK:






```

% Chương trình ví dụ về MSK
close all;
% Parameters
Ns = 8; % number of samples per symbol

x = randint(1000,1); % Random signal

% Use MSK modulation to produce y.
y = mskmod(x,Ns,[],pi/2);

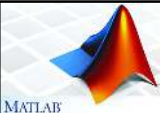
h = scatterplot(y,1,0,'xb');
hold on;
scatterplot(y,Ns,0,'or',h); hold off;

% Transmit signal through an AWGN channel.
yn = awgn(y,25,'measured');

% Plot eyediagram
eyediagram(yn,16);
          
```

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

144




# Quá trình lọc


- **Tạo dạng phổ:**
  - Mật độ phổ công suất  $S_Y(f)$  của tín hiệu ra  $y(t)$ :
 
$$S_Y(f) = S_X(f) |H(f)|^2$$

Trong đó  $S_X(f)$  là PSD của tín hiệu vào  $x(t)$  và  $H(f)$  là hàm truyền của hệ thống (hay bộ lọc)
  - Bằng việc lựa chọn cẩn thận  $H(f) \rightarrow$  tăng cường hoặc khử các thành phần phổ chọn lọc của tín hiệu vào.
  - Khi  $S_X(f)$  và  $H(f)$  xác định  $\rightarrow$  xác định được  $S_Y(f)$
  - Ngược lại: biết  $S_Y(f)$  là PSD đầu ra mong muốn của PSD đầu vào  $S_X(f) \rightarrow$  xác định được  $H(f)$
  - Ví dụ bộ lọc butterworth:
 
$$|H(f)|^2 = \frac{1}{1 + (\omega / \omega_b)^{2n}}$$

$n$  – bậc của bộ lọc

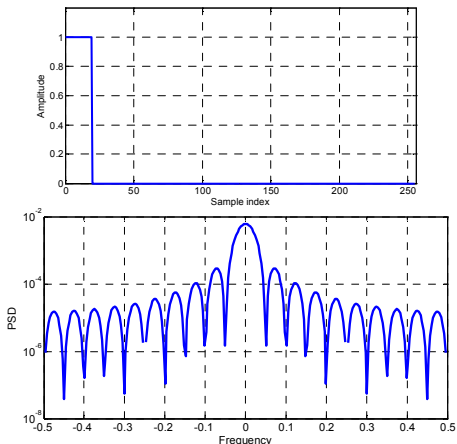
$\omega_b$  – độ rộng băng tần 3 dB


145



# Quá trình lọc

- **Tạo dạng phổ:**
  - Tính toán PSD của một tín hiệu:




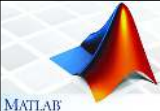
```
function [f,Pf] = spectrocal(t,x)
% Ví dụ chương trình tính toán spectrum
% t - time vector
% x - input samples
% f - frequency vector
% Pf - estimated PSD of x
% written by Nguyen Duc Nhan

Ns = length(x);
Ts = t(2)-t(1);

f = (-Ns/2:Ns/2-1)/(Ns*Ts); % freq. vector
Pf = fft(x,Ns);
Pf = fftshift(Pf)/Ns;
Pf = abs(Pf).^2;

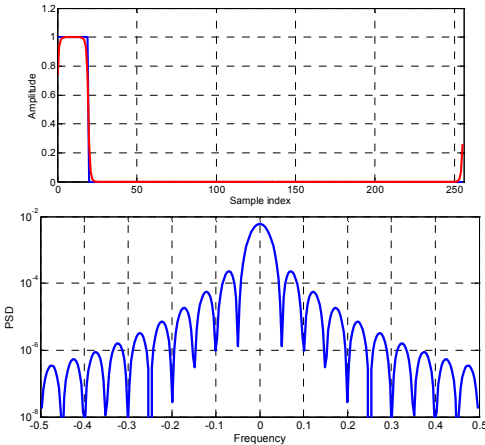
>> t = 0:0.255;
>> x = zeros(1,length(t));
>> x(1:20) = 1;
>> plot(t,x);grid;
>> [f,Xf] = spectrocal(t,x);
>> semilogy(f,Xf);grid;
```


146



## Quá trình lọc

- **Tạo dạng phổ:**
  - Tính toán PSD của một tín hiệu:

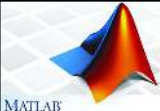


```
function y = butterwflt(x,n,B,Ts)
% Function bo lọc butterworth
% B - filter bandwidth
% Ts - sampling time
% n - filter order
% y - filtered output
Ns = length(x);
% Frequency domain
f = [0:Ns/2-1 -Ns/2:-1]/(Ns*Ts);
Xf = fft(x);
Hf = 1./(1+(f./B).^(2*n)); % transfer func.
Yf = Xf.*Hf;
% Convert into time domain
y = ifft(Yf);
```

```
>> y = butterwflt(x,1,0.2,1);
>> [f,Xf]=spectral(t,y);
>> semilogy(f,Xf); grid;
>> figure(2);
>> plot(t,x,t,y); grid;
```

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**  
Posts & Telecommunications Institute of Technology

147



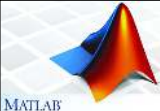
## Quá trình lọc

- **Tạo dạng xung:**
  - Mỗi xung đơn dạng sóng trong các hệ thống truyền dẫn số thường được yêu cầu thỏa mãn 2 điều kiện quan trọng về:
    - Băng thông
    - Chuyển tiếp qua 0 (zero crossings)
  - Nyquist đã chứng minh rằng: một xung  $p(t)$  có zero crossing mỗi chu kỳ  $T_b = 1/R_b$  nếu khai triển  $P(f)$  đáp ứng các ràng buộc sau:
 
$$\sum_{k=-\infty}^{\infty} P(f + kR_b) = T_b \quad |f| < R_b/2$$
  - Một họ  $P(f)$  đáp ứng tiêu chuẩn Nyquist là họ *raised cosine*:
 
$$P(f) = \begin{cases} T_b, & |f| \leq R_b/2 - \beta \\ T_b \cos^2 \frac{\pi}{4\beta} \left( |f| - \frac{R_b}{2} + \beta \right), & R_b/2 - \beta < |f| \leq R_b/2 + \beta \\ 0, & |f| > R_b/2 + \beta \end{cases}$$

Trong đó  $\beta$  là tham số băng tần trội (hệ số rolloff) và  $P(f)$  bị giới hạn bằng tới  $\beta + (R_b/2)$ .

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**  
Posts & Telecommunications Institute of Technology

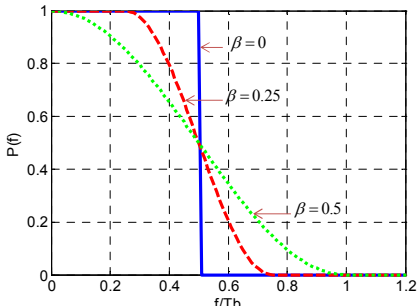
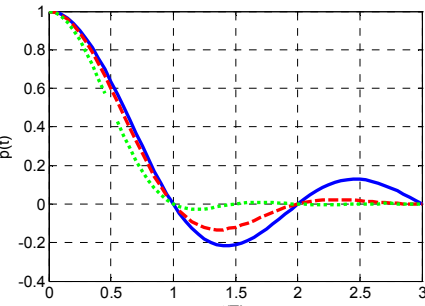
148



## Quá trình lọc


- Tạo dạng xung:**
  - Đáp ứng xung kim của họ *raised cosine*:

$$p(t) = \frac{\cos 2\pi\beta t}{1 - (4\beta t)^2} \left( \frac{\sin \pi R_b t}{\pi R_b t} \right)$$

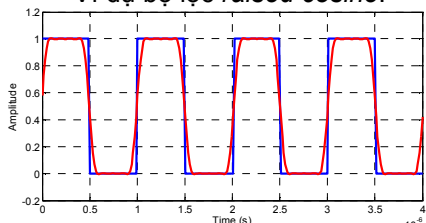
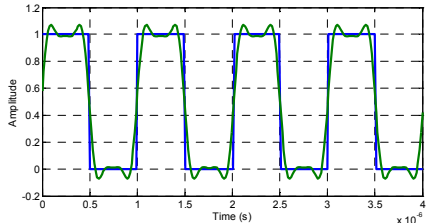
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

149



## Quá trình lọc

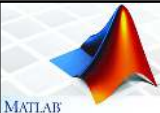
- Tạo dạng xung:**
  - Ví dụ bộ lọc *raised cosine*:

```
function y = raisedcosflt(x,Rb,Ts,beta)
% Function bộ lọc raised cosine
% x - input samples
% Rb - filter bandwidth
% Ts - sampling time
% beta - rolloff factor
% y - filtered output
Ns = length(x);
Tb = 1/Rb;
beta = beta*Rb;
% Frequency domain
f = [0:Ns/2-1 -Ns/2:-1]/(Ns*Ts);
Xf = fft(x);
Yf = zeros(size(Xf));
ind = (abs(f)<=(Rb/2-beta));
Yf(ind) = Xf(ind).*Tb;
ind = (abs(f)<=(Rb/2+beta)&&abs(f)>(Rb/2-beta));
Yf(ind) = Xf(ind).*(Tb*cos(pi/(4*beta))*...
    (abs(f(ind))-Rb/2+beta).^2);
ind = (abs(f)>(Rb/2+beta));
Yf(ind) = Xf(ind).*0;
% Convert into time domain
y = ifft(Yf)/Tb;
```

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

150




## Quá trình lọc

- **Tạo dạng xung:**
  - Trong MATLAB có thể sử dụng các hàm cho quá trình lọc trong signal processing toolbox:
  - Ví dụ sử dụng hàm filter:
 
$$y = \text{filter}(b,a,x) \quad \text{Trong đó } a, b \text{ là các vector chứa các hệ số của bộ lọc}$$

$$y(n) = b(1)*x(n) + b(2)*x(n-1) + \dots + b(nb+1)*x(n-nb) - a(2)*y(n-1) - \dots - a(na+1)*y(n-na)$$
  - Một số kiểu bộ lọc sẵn có:
    - Bộ lọc butterworth:  $[b,a] = \text{butter}(n,Wn)$
    - Bộ lọc bessel:  $[b,a] = \text{besself}(n,Wo)$
    - Bộ lọc chebyshev:  $[b,a] = \text{cheby1}(n,R,Wp)$  và  $[b,a] = \text{cheby2}(n,R,Wst)$
    - Bộ lọc raised cosine:  $y = \text{rcosfilt}(x,Fd,Fs)$

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

151



## Quá trình lọc

- **Bộ lọc phối hợp (matched filters):**
  - Xét bài toán thu được hoặc không thu được một xung có dạng  $p(t)$  từ tín hiệu quan sát  $y(t)$ :
 
$$y(t) = \begin{cases} p(t) + N(t) \\ \text{or} \\ N(t) \end{cases} \quad N(t) - \text{nhiều cộng có trung bình 0.}$$
  - Xử lý  $y(t)$  để quyết định có hay không có  $p(t)$  trong khoảng chu kỳ  $T$ .
  - Xử lý  $y(t)$  thu được:  $z = \int_0^T y(\tau)h(T-\tau)d\tau$
  - Xác suất lỗi trung bình nhỏ nhất khi bộ lọc có hàm truyền:
 
$$H(f) = KP^*(f)\exp(j2\pi fT)$$
  - đáp ứng xung:  $h(t) = p(T-t) \rightarrow z = \int_0^T y(\tau)h(T-\tau)d\tau = \int_0^T y(\tau)p(\tau)d\tau$

Đáp ứng xung phối hợp với  $p(t)$ .

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

152






## Quá trình lọc

- **Bộ lọc phối hợp (matched filters):**
  - Bộ lọc phối hợp cho một tín hiệu điều chế tuyến tính sử dụng dạng xung  $p(t)$ :
 

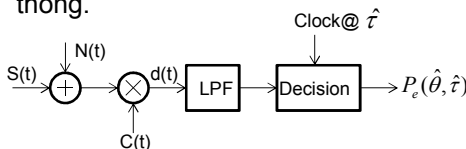

  - Khi  $p(t)$  có dạng xung vuông đơn vị  $\rightarrow z$  là tích phân của tín hiệu đầu vào.
  - Quá trình lấy tích phân được bắt đầu thực hiện tại đầu mỗi khoảng chu kì  $\rightarrow$  bộ lọc integrate-and-dump (I&D).

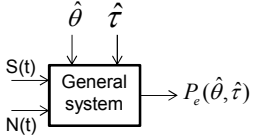
153



## Quá trình đồng bộ

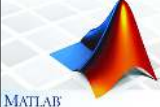
- **Quá trình đồng bộ trong mô phỏng:**
  - Xem xét sự ảnh hưởng của đồng bộ đến hiệu năng của hệ thống.





- Có các cách tiếp cận khác nhau:
  - Dạng cấu trúc
  - Dạng mô tả thống kê
- Tại bộ thu, quá trình đồng bộ bao gồm:
  - Khôi phục sóng mang
  - Khôi phục đồng hồ (tín hiệu định thời)

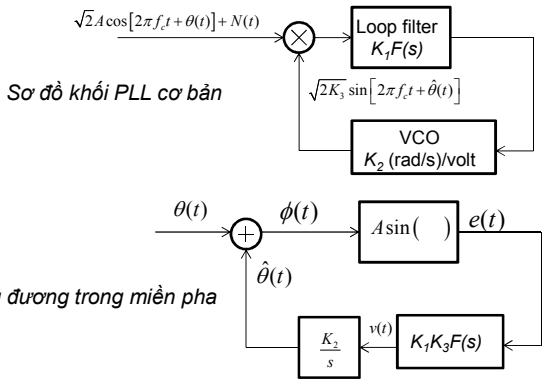
154



## Quá trình đồng bộ

- Mô phỏng mạch vòng khóa pha PLL:**
  - PLL được mô tả bởi ptr. vi phân phi tuyến
  - Bộ lọc vòng bậc 2:

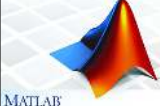
*Sơ đồ khối PLL cơ bản*



*Mô tả tương đương trong miền pha*

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology

155



## Quá trình đồng bộ

- Mô phỏng mạch vòng khóa pha PLL:**
  - Hàm truyền bộ lọc vòng:

$$F(s) = \frac{s\tau_2 + 1}{s\tau_1 + \alpha_1} \quad \text{Với} \quad \alpha_1 = \begin{cases} 1 & \text{Bộ lọc thụ động} \\ 0 & \text{Bộ lọc tích cực} \end{cases}$$

- PLL mở rộng:
 
$$\begin{aligned} v(t) &= (K_1 K_3 / \tau_1) [\tau_2 \dot{y}(t) + y(t)] \\ \hat{\theta}(t) &= (K_1 K_2 K_3 / \tau_1) [\tau_2 \dot{y}(t) + y(t)] \\ \phi(t) &= \theta(t) - \hat{\theta}(t) = \theta(t) - (K_1 K_2 K_3 / \tau_1) [\tau_2 \dot{y}(t) + y(t)] \\ e(t) &= A \sin \{ \theta(t) - (K_1 K_2 K_3 / \tau_1) [\tau_2 \dot{y}(t) + y(t)] \} \\ \ddot{y}(t) &= e(t) - (\alpha_1 / \tau_1) \dot{y}(t) \end{aligned}$$

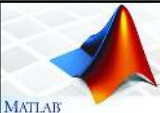
Đặt các hằng số:

$$c_1 = \frac{-K_1 K_2 K_3 \tau_2}{\tau_1} \quad c_2 = \frac{-K_1 K_2 K_3}{\tau_1} \quad c_3 = -\frac{\alpha_1}{\tau_1}$$

→ 
$$\ddot{y}(t) = A \sin [\theta(t) + c_1 \dot{y}(t) + c_2 y(t)] + c_3 \dot{y}(t)$$

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG  
Posts & Telecommunications Institute of Technology


156




## Bài tập chương 4

31. Viết chương trình MATLAB tạo chuỗi bit ngẫu nhiên phân bố đều có độ dài 128 bit. Sau đó thực hiện chuyển đổi chuỗi bit này thành các giá trị thập phân trong dải từ [0,15].
  - Gợi ý: Chuyển đổi vector hàng thành ma trận mx4, sau đó dùng hàm *bi2de* để chuyển đổi sang dạng thập phân.
32. Xây dựng các *function* để nén và giải nén tín hiệu theo luật A có dạng cấu trúc sau:
 

$\text{function [y,amax] = alaw(x,A) và function x = invalaw(y,A)}$
33. Viết chương trình mã hóa tín hiệu  $x = 2\cos(4\pi t)$  tại tần số lấy mẫu  $f_s = 20$  Hz sử dụng quá trình PCM theo luật A với 8 mức lượng tử. Hãy xác định từ mã đầu ra của 5 mẫu đầu tiên. Vẽ biểu diễn tín hiệu gốc ban đầu, tín hiệu được lấy mẫu và tín hiệu được lượng tử hóa trên cùng một hình. (Sử dụng các function đã xây dựng của bài tập trên)
34. Xây dựng function để chuyển đổi chuỗi bit dữ liệu đầu vào thành mã đường RZ đơn cực.  
 Sử dụng function này để mã hóa RZ và biểu diễn dạng sóng của chuỗi bit [0 1 1 0 1 0 1 0] tại tốc độ 1 Mbit/s.


157



## Bài tập chương 4


35. Viết chương trình MATLAB xác định các từ mã đầu ra của bộ mã hóa khối có tốc độ 4/7. Biết ma trận tạo mã có dạng:
 

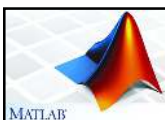
1	0	0	0	1	1	1
0	1	0	0	1	1	0
0	0	1	0	1	0	1
0	0	0	1	0	0	1
36. Viết function để tạo chuỗi xung tam giác đầu ra. Biết trong một chu kỳ dạng xung được biểu diễn bởi hàm sau:
 

$$p(t) = \begin{cases} 1 - |(t - T_w)/T_w|, & 0 \leq t \leq T_p \\ 0, & t \text{ khác} \end{cases} \quad \begin{matrix} \text{Với } T_w = T_p/2 \\ T_p - \text{chu kỳ xung} \end{matrix}$$
37. Cho tín hiệu tương tự được mô tả bởi công thức sau:
 

$$s(t) = 2\cos(20\pi t + \pi/4) + \cos(30\pi t)$$

 Viết chương trình thực hiện điều chế biên độ tín hiệu bằng sóng mang  $f_c = 300$  Hz. Vẽ dạng sóng tín hiệu bản tin ban đầu và tín hiệu được điều chế.  
 Giải điều chế tín hiệu trên bằng kỹ thuật phù hợp và vẽ dạng sóng tín hiệu sau khi được giải điều chế.


158



## Bài tập chương 4

38. Tạo chuỗi bit ngẫu nhiên có độ dài 5000 bits. Chuyển đổi chuỗi bit này thành dạng sóng mã đường NRZ lưỡng cực tại tốc độ 100 Mbit/s. Sử dụng bộ lọc *raised cosine* có độ rộng băng tần 300 MHz và hệ số rolloff bằng 0.5 để lọc chuỗi tín hiệu NRZ này. Vẽ biểu diễn dạng sóng tín hiệu trên 10 chu kỳ bit trước và sau khi lọc tín hiệu. Vẽ biểu đồ mắt của tín hiệu sau khi lọc trên cửa sổ 2 chu kỳ bit. (Sử dụng hàm *eyediagram* để vẽ mẫu mắt).
39. Tương tự bài tập 38, sử dụng bộ lọc *butterworth* có tần số cắt khoảng 250 MHz để lọc tín hiệu. (Sử dụng các hàm *butter* và *filter* trong signal processing toolbox).
40. Tạo chuỗi ký tự ngẫu nhiên có độ dài 1000 ký tự để thực hiện điều biến BPSK. Hãy viết chương trình biểu diễn dạng sóng đường bao phức của tín hiệu điều biến BPSK tại tốc độ 10 Mb/s bởi các xung *raised cosine* được mô tả bởi hàm sau:

$$p(t) = \left(1 - \cos\left(\frac{2\pi t}{T}\right)\right) \quad 0 \leq t \leq T$$

Vẽ dạng phổ của tín hiệu được điều biến.