23-APRIL-2024

Spread Spectrum

Data Communication

KAUSHAL VALA: 22BCE376

Why is spreading important?

→ For some applications it is necessary for the system to resist **external interferences** and to make it difficult for unauthorized receivers to receive the message being transmitted.

→ A type of communications is called as secure communication such that **noise interference** and **unwanted receivers** should not detect the message.

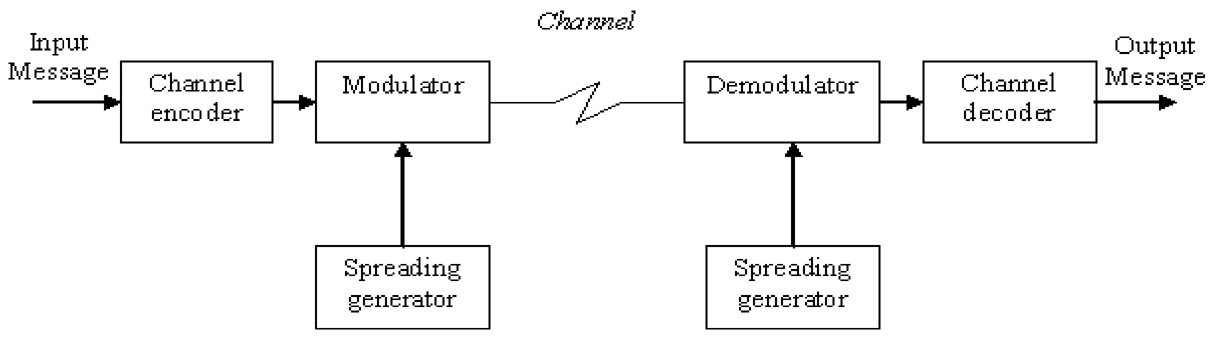
→ The interference may intentional or unintentional

→ **Unintentional**: due to transmission by other user on same channel.

→ **Intentional**: due to hostile transmitter to JAM the transmission

Introduction:

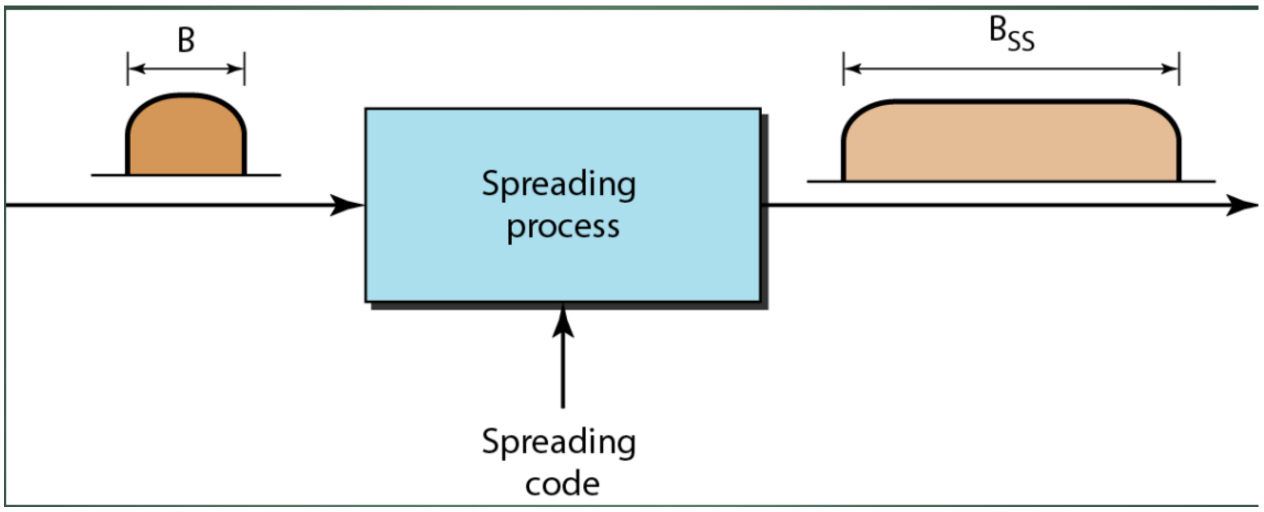
- Spread-spectrum radio communications is a favorite technology of the military because it resists jamming and is hard for an enemy to intercept, Just as they are unlikely to be intercepted by a military opponent



Binary information sequence is input to the channel encoder which encodes the message according to some coding technique.

Coded sequence is then given to modulator which gets pseudo-noise sequence from the pseudo-random pattern generator which spreads the sequence over wide frequency band.

At receiver side reverse process is to be done. Pseudorandom pattern generator at both the sides operate in synchronization. For an arbitrary receiver it is difficult to know the pseudo-noise sequence since it appears like noise.

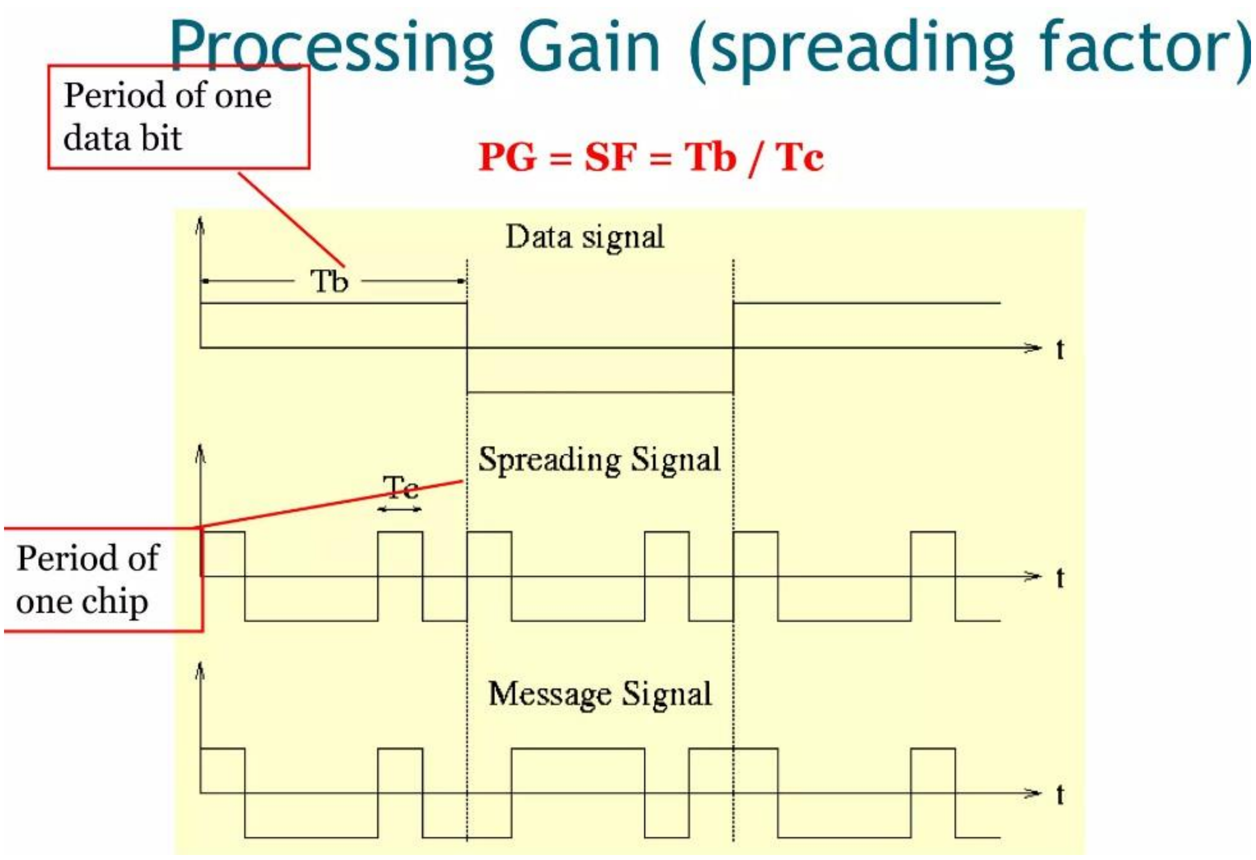


Types of Spread Spectrum:

1) FHSS (Frequency-hopping spread spectrum)

2) DSSS (Direct Sequence spread spectrum)

DSSS (Direct Sequence spread spectrum):



CODE:

```
clc;
clear;
close all;
prompt = {'enter bit sequence (52 bits max):'};
ititle = 'give input bits';
dims = [1 50];
temp1 = inputdlg(prompt,ititle,dims);
temp2 = bin2dec(temp1{1});
bit_sequence = str2num(dec2bin(temp2,length(temp1{1})).');
in_vect = bit_sequence.';
t1 = length(temp1{1});
psn = randi([0 1],1,t1);
Xor_out = xor(psn,in_vect);
```

```
f = 10^6;
T = 1\f;
t = T/99:T/99:T;
BPSK = [];
for i=1:length(bit_sequence)
if Xor_out(1,i) == 0
BPSK = [BPSK sin(2*pi*f*t)];
else
BPSK = [BPSK -sin(2*pi*f*t)];
end

end

tt = T/99:T/99:T*t1;
%GRAPHS
figure(1)
subplot(4,1,1);
stairs(in_vect,'linewidth',3), grid on;
title('Input Bit Sequence');
xlabel('time(s)');
ylabel('amplitude(v)');
axis([1 t1 0 1]);
subplot(4,1,2);
stairs(psn,'r','linewidth',3), grid on;
title('Psuedo Random Noise Code');
xlabel('Time(s)');
ylabel('Amplitude(v)');
axis([1 t1 0 1]);
subplot(4,1,3);
stairs(Xor_out,'m','linewidth',3), grid on;
title('X OR');
xlabel('Time(s)');
ylabel('Amplitude(V)');
axis([1 t1 0 1]);
subplot(4,1,4);
plot(tt,BPSK,'k','linewidth',3), grid on;
title('BPSK');
xlabel('Time(s)');
ylabel('Amplitude(V)');
axis([1 T*t1 -1 1]);
```

Output:

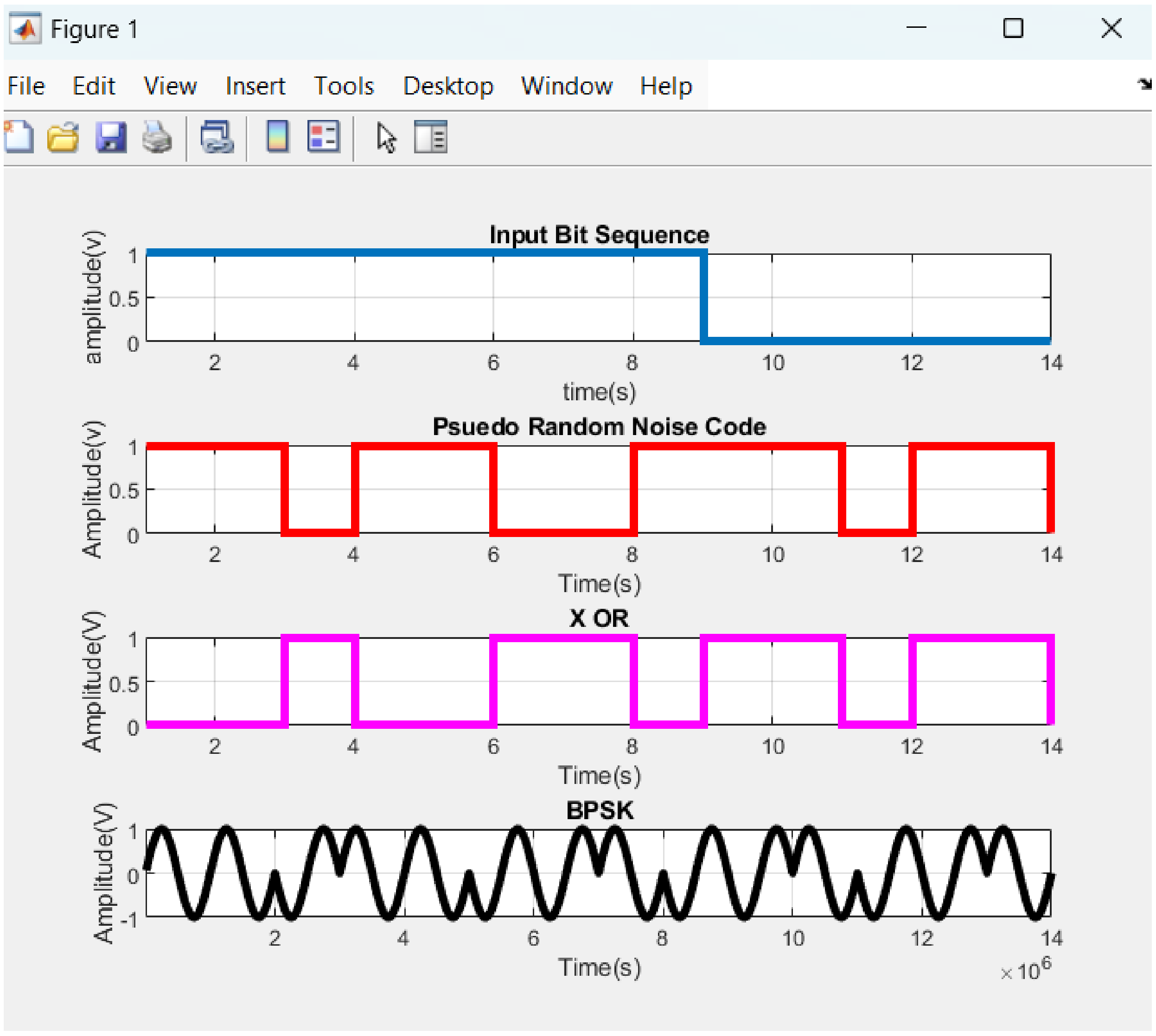
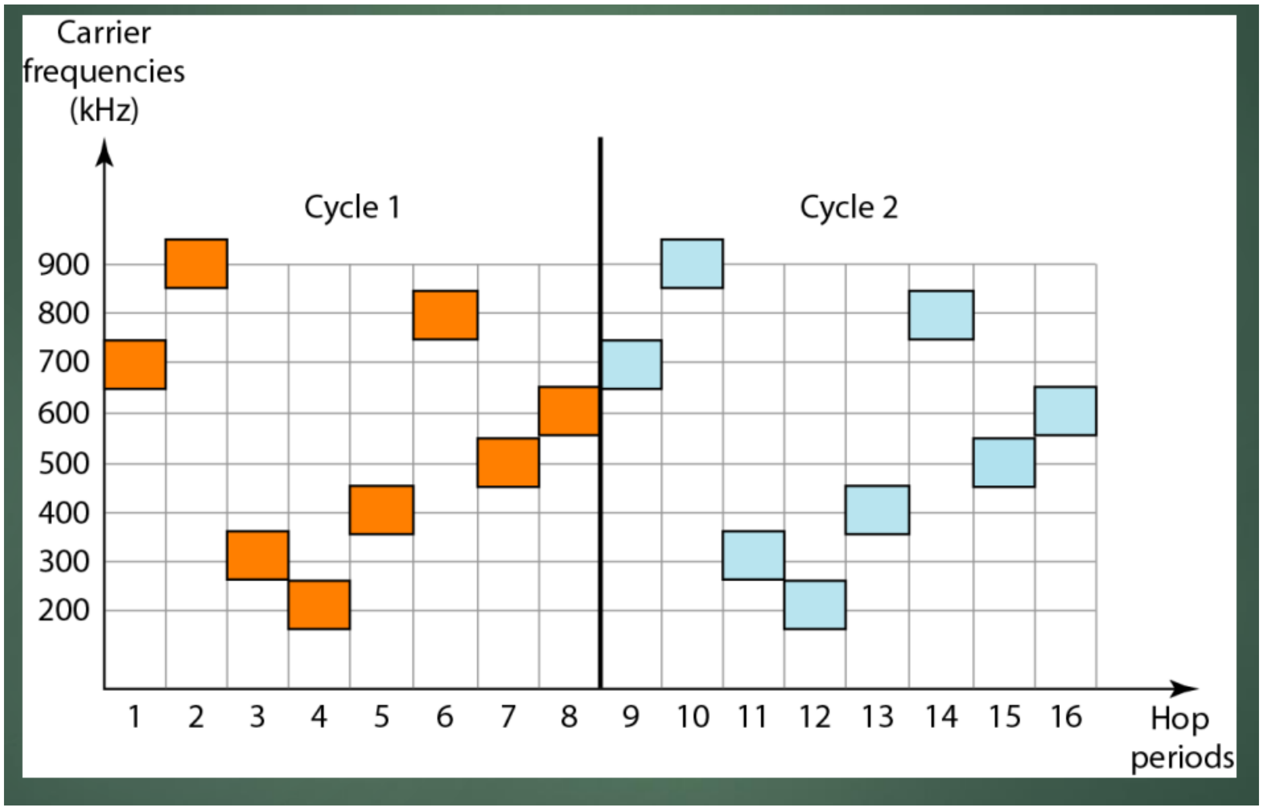


Figure 1
File Edit View Insert Tools Desktop Window Help
Input Bit Sequence
amplitude(v)
time(s)
Psuedo Random Noise Code
Amplitude(v)
Time(s)
X OR
Amplitude(V)
Time(s)
BPSK
Amplitude(V)
Time(s)
$\times 10^6$

Frequency-hopping spread spectrum (FHSS):



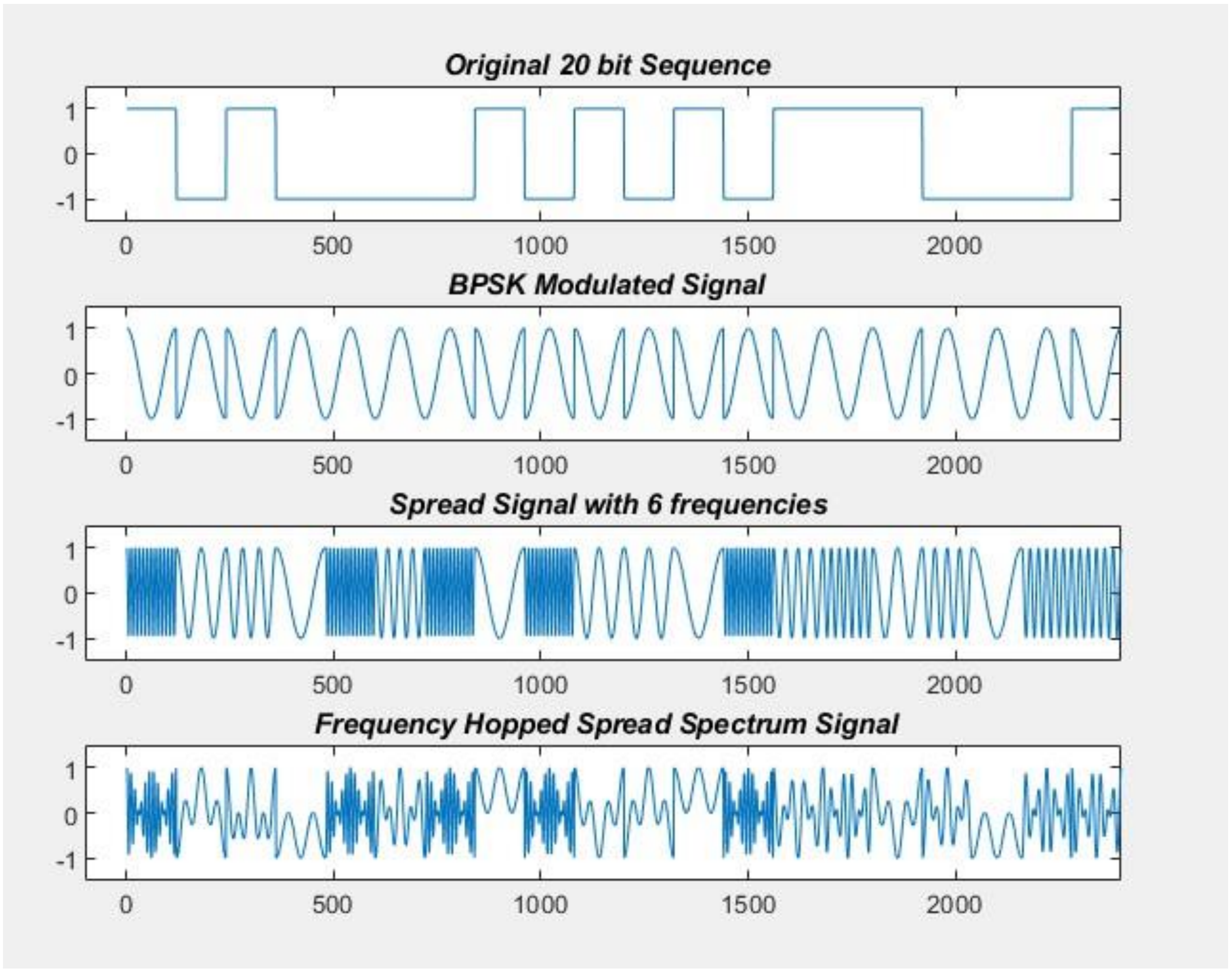
Code:

```
clc
clear
% Generation of Sample bits
sequence=round(rand(1,20));
input_signal=[];
carrier_signal=[];
time=[0:2*pi/119:2*pi];
for k = 1 :20
if sequence(1,k)==0
sig=-ones(1,120);
else
sig=ones(1,120);
end
c=cos(time);
carrier_signal = [carrier_signal c];
input_signal = [input_signal sig];
end
figure(1)
```

```
subplot(4,1,1);
plot(input_signal);
axis([-100 2400 -1.5 1.5]);
title('\bf\it Original 20 bit Sequence');
bpsk_mod_signal=input_signal.*carrier_signal;
subplot(4,1,2);
plot(bpsk_mod_signal);
axis([-100 2400 -1.5 1.5]);
title('\bf\it BPSK Modulated Signal');
time1=[0:2*pi/9:2*pi]; %[0:0.698:6.28]
time2=[0:2*pi/19:2*pi]; %[0:0.331:6.28]
time3=[0:2*pi/29:2*pi]; %[0:0.217:6.28]
time4=[0:2*pi/39:2*pi]; %[0:0.161:6.28]
time5=[0:2*pi/59:2*pi]; %[0:0.106:6.28]
time6=[0:2*pi/119:2*pi];%[0:0.052:6.28]
carrier1=cos(time1);
carrier1=[carrier1 carrier1 carrier1 carrier1 carrier1 carrier1 carrier1 carrier1
carrier1 carrier1 carrier1 carrier1];
carrier2=cos(time2);
carrier2=[carrier2 carrier2 carrier2 carrier2 carrier2 carrier2];
carrier3=cos(time3);
carrier3=[carrier3 carrier3 carrier3 carrier3];
carrier4=cos(time4);
carrier4=[carrier4 carrier4 carrier4];
carrier5=cos(time5);
carrier5=[carrier5 carrier5];
carrier6=cos(time6);
spread_signal=[];
for n=1:20
c=randi([1 6],1,1);
switch(c)
case(1)
spread_signal=[spread_signal carrier1];
case(2)
spread_signal=[spread_signal carrier2];
case(3)
spread_signal=[spread_signal carrier3];
case(4)
spread_signal=[spread_signal carrier4];
case(5)
spread_signal=[spread_signal carrier5];
case(6)
spread_signal=[spread_signal carrier6];
end
end
subplot(4,1,3)
plot([1:2400],spread_signal);
```

```
axis([-100 2400 -1.5 1.5]);
title('\bf\it Spread Signal with 6 frequencies');
% Spreading BPSK Signal
freq_hopped_sig=bpsk_mod_signal.*spread_signal;
subplot(4,1,4)
plot([1:2400],freq_hopped_sig);
axis([-100 2400 -1.5 1.5]);
title('\bf\it Frequency Hopped Spread Spectrum Signal');
% Demodulation of BPSK Signal
bpsk_demodulated=freq_hopped_sig./spread_signal;
figure(2)
subplot(2,1,1)
plot([1:2400],bpsk_demodulated);
axis([-100 2400 -1.5 1.5]);
title('\bf Demodulated BPSK Signal from Wide Spread');
original_BPSK_signal=bpsk_demodulated./carrier_signal;
subplot(2,1,2)
plot([1:2400],original_BPSK_signal);
axis([-100 2400 -1.5 1.5]);
title('\bf Transmitted Original Bit Sequence');
```

Output:



Applications:

Military

- Spread-spectrum signals are highly resistant to deliberate jamming unless the adversary has knowledge of the frequency-hopping pattern. Military radios generate the frequency-hopping pattern under the control of a secret Transmission Security Key (TRANSEC) that the sender and receiver share in advance.

It avoids intentional interference such as jamming effectively.