MWC Experiment 8

Shivani Bhat

BE EXTC A
2022200013

Aim:

To understand and simulate Direct Sequence Spread Spectrum Modulation using MATLAB

Theory:

Direct Sequence Spread Spectrum (DSSS) is a modulation and transmission technique used in spread spectrum communications. In this method, the original information signal is *spread* over a much wider bandwidth than the minimum required. This spreading is achieved by combining the data signal with a high-rate **Pseudo-Random Noise (PN) code**. The result is a wideband signal that appears noise-like to unintended receivers, providing enhanced **security**, **interference rejection**, and **multi-user access** capabilities.

1. Principle of DSSS

In DSSS, each bit of the original data is represented by multiple shorter-duration bits called **chips**. A unique **spreading code** or **PN sequence** determines how each bit is spread. The multiplication (or XOR operation in digital logic) of the data and the PN code produces a *spread signal* with a bandwidth several times greater than the original data bandwidth.

Mathematically,
[
s(t) = b(t) * c(t)
]
where:

- (b(t)) = baseband data signal (+1 or –1)
- (c(t)) = spreading code (PN sequence)
- (s(t)) = spread signal

This process effectively distributes the signal energy across a wide frequency spectrum, reducing power spectral density and making the signal more resistant to interference and jamming.

2. Spreading Using PN Code

- The **PN sequence** is a deterministic binary sequence that appears random but can be reproduced exactly by both transmitter and receiver.
- The **chip rate (Rc)** — the rate of the PN sequence — is much higher than the **data rate (Rb)**, leading to a **spreading factor (SF)** defined as:
  SF=Rc/Rb

- Each data bit is represented by *SF chips*, thereby increasing the signal bandwidth by a factor of *SF*.

This process makes the transmitted waveform appear noise-like, preventing easy detection or interception.

3. Signal Despreading at the Receiver

At the receiver end, the received signal initially appears as random noise. However, if the receiver uses the **same PN sequence**, synchronized in phase and timing, it can **multiply** the incoming signal by the same code. This operation "compresses" the wideband signal back into its original narrowband form, recovering the transmitted data.

If the incorrect PN code or phase is used, the output remains noise, providing an inherent level of security.

4. System Characteristics

Parameter	**Description**
Data Rate (Rb)	Rate of transmission of the original message bits
Chip Rate (Rc)	Rate of PN code bits per second
Processing Gain (Gp)	Ratio of spread bandwidth to data bandwidth, (G_p = 10 \log_{10}(R_c / R_b))
PN Code	A pseudo-random binary sequence used for spreading and despreading
Chips	Individual bits of the PN code representing each data bit

5. Applications of DSSS

1. **Wireless Communication (e.g., IEEE 802.11b Wi-Fi):**
   DSSS provides robustness against interference and enables multiple users to share the same frequency band using different PN codes.
2. **Global Positioning System (GPS):**
   DSSS allows simultaneous transmission from multiple satellites using unique spreading codes, ensuring accurate position determination even in noisy environments.
3. **Military and Secure Communication:**
   The noise-like nature of DSSS makes interception or jamming difficult, making it ideal for tactical communication systems.

4. **Anti-Jamming and Interference-Resistant Systems:**
   DSSS minimizes the impact of narrowband interference, as unwanted signals are “despread” and averaged out during demodulation.

6. Advantages of DSSS

- **Enhanced Security:**
  The use of pseudo-random spreading codes makes unauthorized interception difficult.
- **Interference Resistance:**
  Narrowband interference affects only a small portion of the spread spectrum, which is easily filtered out.
- **Multipath and Fading Resistance:**
  Spreading provides time and frequency diversity, reducing the effect of multipath propagation.
- **Robustness in Noisy Environments:**
  DSSS maintains reliable performance even under significant noise or jamming conditions.

Code and Results:

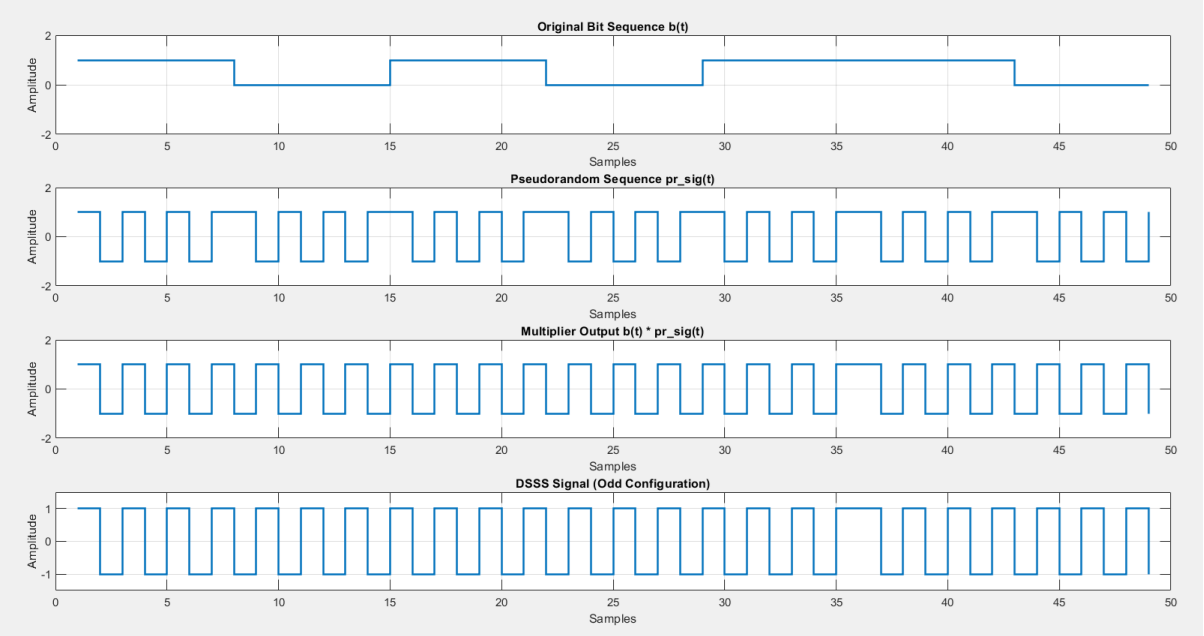
Our task was to simulate DSSS for an odd configuration of bits.

1. DSSS without BPSK

```
% Direct Sequence Spread Spectrum (DSSS) without BPSK

% Odd configuration version


clc; clear; close all;


% Parameters

data = [1 0 1 0 1 1 0]; % Odd number of data bits

chip_rate = 7; % Odd chip rate for spreading

pn_seq = [1 -1 1 -1 1 -1 1]; % Odd-length pseudorandom sequence


% Repeat the PN sequence to match the spread length

spread_length = length(data) * chip_rate;

pn_seq = repmat(pn_seq, 1, ceil(spread_length / length(pn_seq)));

pn_seq = pn_seq(1:spread_length); % Trim to match exact spread length
```

```
% Spread the signal (no BPSK modulation)
data_repeated = repelem(2*data - 1, chip_rate); % Convert 0→-1 and repeat each bit
spread_signal = data_repeated .* pn_seq; % DSSS signal (chip spreading)

% Plotting
figure;
subplot(4, 1, 1);
stairs(repelem(data, chip_rate), 'LineWidth', 1.5);
title('Original Bit Sequence b(t)');
xlabel('Samples');
ylabel('Amplitude');
ylim([-2 2]);
grid on;

subplot(4, 1, 2);
stairs(pn_seq, 'LineWidth', 1.5);
title('Pseudorandom Sequence pr\_sig(t)');
xlabel('Samples');
ylabel('Amplitude');
ylim([-2 2]);
grid on;

subplot(4, 1, 3);
stairs(data_repeated .* pn_seq, 'LineWidth', 1.5);
title('Multiplier Output b(t) * pr\_sig(t)');
xlabel('Samples');
ylabel('Amplitude');
ylim([-2 2]);
grid on;
```

```
subplot(4, 1, 4);
stairs(spread_signal, 'LineWidth', 1.5);
title('DSSS Signal (Odd Configuration)');
xlabel('Samples');
ylabel('Amplitude');
ylim([-1.5 1.5]);
grid on;
```

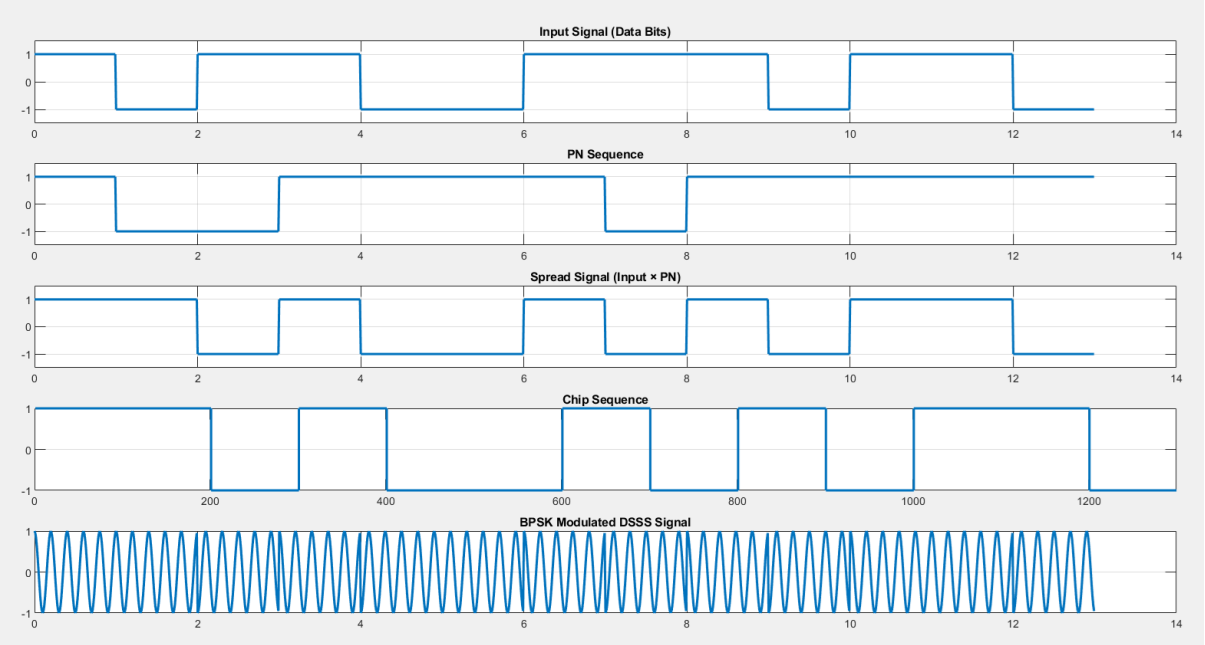


2. DSSS with BPSK:

```
clc; clear; close all;


% ----- PARAMETERS -----
num_bits = randi([5 15], 1, 1);     % Randomly choose number of bits between 5 and 15
if mod(num_bits, 2) == 0
    num_bits = num_bits + 1;        % Ensure odd number of bits
end


Tb = 1;         % Bit duration
fs = 100;       % Sampling frequency
fc = 5;         % Carrier frequency
```

```
t = 0:1/fs:Tb-1/fs;   % Time vector for one bit duration

% ----- INPUT BIT GENERATION -----
input_bits = randi([0 1], 1, num_bits);     % Random binary bits
input_signal = [];                          % Initialize signal

for i = 1:num_bits
   bit_value = 2*input_bits(i) - 1;       % Convert 0→-1, 1→+1
   input_signal = [input_signal bit_value * ones(1, length(t))];
end

% ----- PN SEQUENCE GENERATION -----
pn_sequence = randi([0 1], 1, num_bits);    % Random PN bits
pn_sequence(pn_sequence == 0) = -1;         % Convert 0→-1

pn_signal = [];
for i = 1:num_bits
   pn_signal = [pn_signal pn_sequence(i) * ones(1, length(t))];
end

% ----- SPREADING -----
spread_signal = input_signal .* pn_signal;

% ----- BPSK MODULATION -----
tt = 0:1/fs:Tb*num_bits - 1/fs;            % Total time vector
carrier = cos(2*pi*fc*tt);                 % Carrier wave
bpsk_signal = spread_signal .* carrier;

% ----- PLOTTING -----
figure;
```

```
subplot(5,1,1);
plot(tt, input_signal, 'LineWidth', 1.5);
title('Input Signal (Data Bits)');
ylim([-1.5 1.5]); grid on;

subplot(5,1,2);
plot(tt, pn_signal, 'LineWidth', 1.5);
title('PN Sequence');
ylim([-1.5 1.5]); grid on;

subplot(5,1,3);
plot(tt, spread_signal, 'LineWidth', 1.5);
title('Spread Signal (Input × PN)');
ylim([-1.5 1.5]); grid on;

subplot(5,1,4);
stairs(1:length(spread_signal), spread_signal, 'LineWidth', 1.5);
title('Chip Sequence');
xlim([0 length(spread_signal)]); grid on;

subplot(5,1,5);
plot(tt, bpsk_signal, 'LineWidth', 1.5);
title('BPSK Modulated DSSS Signal');
grid on;

% ----- INFO DISPLAY -----
disp(['Number of bits used (odd): ', num2str(num_bits)]);
disp('Input bits: ');
disp(input_bits);
disp('PN sequence: ');
disp((pn_sequence+1)/2); % Convert back to 0/1 for clarity
```



Conclusion:

In this experiment, we simulated the Direct Sequence Spread Spectrum (DSSS) system using an odd number of bits to ensure an asymmetric bit pattern, which helps in clearly observing spreading and despreading behavior. The experiment was performed in two stages — first using only DSSS (baseband spreading), and then with BPSK modulation applied to the spread signal.

1. DSSS without BPSK

- In this stage, the input data bits were multiplied by a pseudo-noise (PN) sequence of equal (odd) length.
- The result was a **spread signal** where each data bit was represented by multiple faster "chips," increasing the signal's bandwidth.
- The spreading operation distributed the signal energy over a wider frequency range, improving its resistance to narrowband interference and noise.
- The odd configuration of bits ensured that the PN alignment was unambiguous — the first and last chips didn't pair symmetrically, making correlation peaks clearer during despreading.

2. DSSS with BPSK

- Here, the spread signal was modulated using **Binary Phase Shift Keying (BPSK)** by multiplying it with a cosine carrier.
- The BPSK step converted the baseband spread sequence into a bandpass signal, suitable for RF transmission.
- The BPSK-modulated DSSS signal showed **phase reversals corresponding to data bit transitions**, but the overall power and bandwidth characteristics were similar to DSSS.

- The modulated DSSS signal can be transmitted efficiently through a wireless channel and later demodulated and despread to recover the original bits.

Thus we conclude that DSSS alone enhances signal security and interference resistance, while DSSS combined with BPSK provides both robustness and transmission suitability — a fundamental concept in CDMA and secure wireless systems.