3. Simulation of CDMA in MATLAB

Aim : To simulate a simple CDMA transmitter/receiver using Walsh spreading codes:
a) Walsh Code 1 is to transmit 100 data bits using a spreading length of 800 chips and verify the recovered bits and BER, ensuring correct encoding and reliable recovery.

b) Walsh Code 2 is to transmit 200 data bits with a spreading length of 800 chips and check the recovered bits and BER, confirming proper code separation and minimal interference.

c) Walsh Code 3 is to transmit 300 data bits using a spreading length of 800 chips and ensure accurate recovery and low BER, validating the effectiveness of the assigned spreading code.

Software required : Matlab online compiler.

Theory

Walsh codes are orthogonal binary sequences (±1) used to spread user data in CDMA so multiple users share the same channel with minimal mutual interference. Each data bit is multiplied (spread) by the assigned Walsh code to create a chip sequence; despreading uses correlation with the same Walsh code to recover the original bit. AWGN in the channel degrades correlation values and increases BER depending on SNR and code length. Increasing spreading length improves immunity to noise and multiuser interference by averaging over more chips.

Procedure (stepwise)

1. Choose spreading length L = 800 and verify a Hadamard/Walsh matrix of order L is possible.
2. Generate random data bits for the user: convert 0→-1, 1→+1 (bipolar).
3. Select the assigned Walsh code row (e.g., row 1, 2, or 3) of Hadamard(L).
4. Spread each bit by Kronecker product: spread_signal = kron(bipolar_bit_vector, walsh_code).
5. Add AWGN to the spread signal for a chosen SNR_dB.
6. Break the received signal into chip-blocks of length L, correlate each block with the same Walsh code.
7. Decide recovered bit = 1 if correlation>0 else 0. Compute bit errors and BER = errors/number_of_bits.
8. Repeat for parts (a), (b), (c) with their respective bit counts and Walsh code indices.

Pseudo-Code: CDMA Simulation Using Walsh Codes:

Step 1: Generate Walsh Codes

INPUT: spreading_length = 800

Compute walsh_matrix = hadamard(spreading_length)

Assign:

walsh1 = walsh_matrix(1, :)

walsh2 = walsh_matrix(2, :)

walsh3 = walsh_matrix(3, :)

Step 2: Define Data For Each User:

User1_bits = generate 100 random bits (0/1)

User2_bits = generate 200 random bits (0/1)

User3_bits = generate 300 random bits (0/1)

Convert bits to bipolar:

0 → -1

1 → +1

Step 3: Spread Each User's Data Using Their Walsh Code:

FOR each bit in User1_bits:

spread_signal1 = bit * walsh1

Repeat for User2_bits → walsh2

Repeat for User3_bits → walsh3

Generate total transmitted signal:

Tx = sum of (spread_signal1 + spread_signal2 + spread_signal3)

Step 4: Add Channel Noise (Optional AWGN):

SNR = desired SNR value

Rx = awgn(Tx, SNR)

Step 5: Despread and Recover Bits

FOR each bit interval:

correlate = Rx_segment · walsh1

IF correlate > 0 → recovered_bit = 1

ELSE → recovered_bit = 0

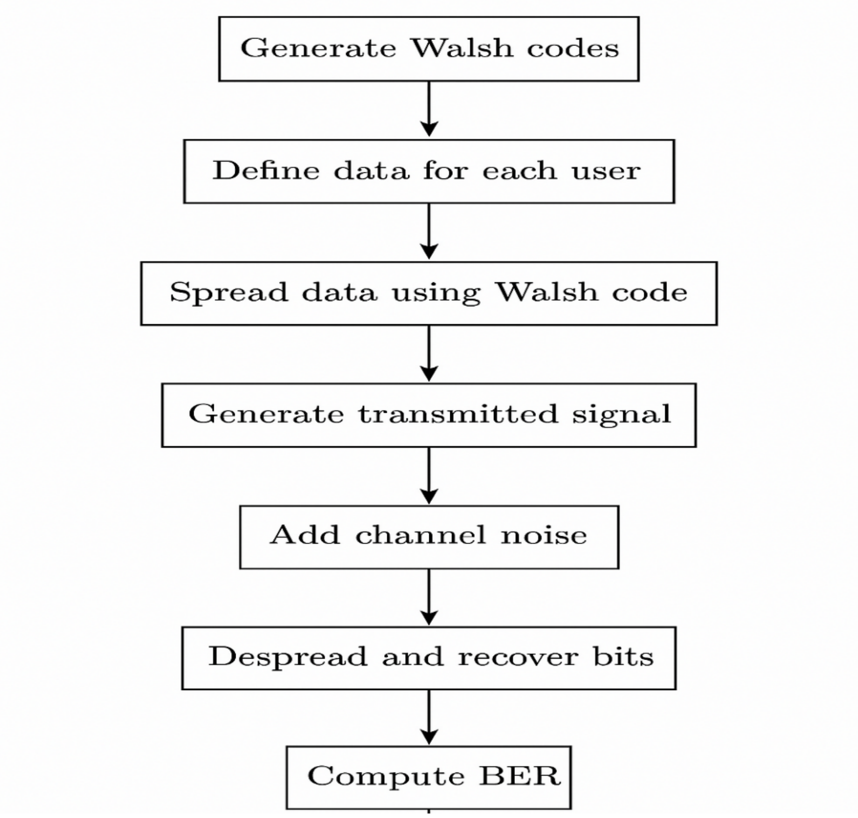
Step 6: Compute BER

BER1 = number_of_errors(User1_bits, recovered_bits1) / 100

BER2 = number_of_errors(User2_bits, recovered_bits2) / 200

BER3 = number_of_errors(User3_bits, recovered_bits3) / 300

FLOWCHART



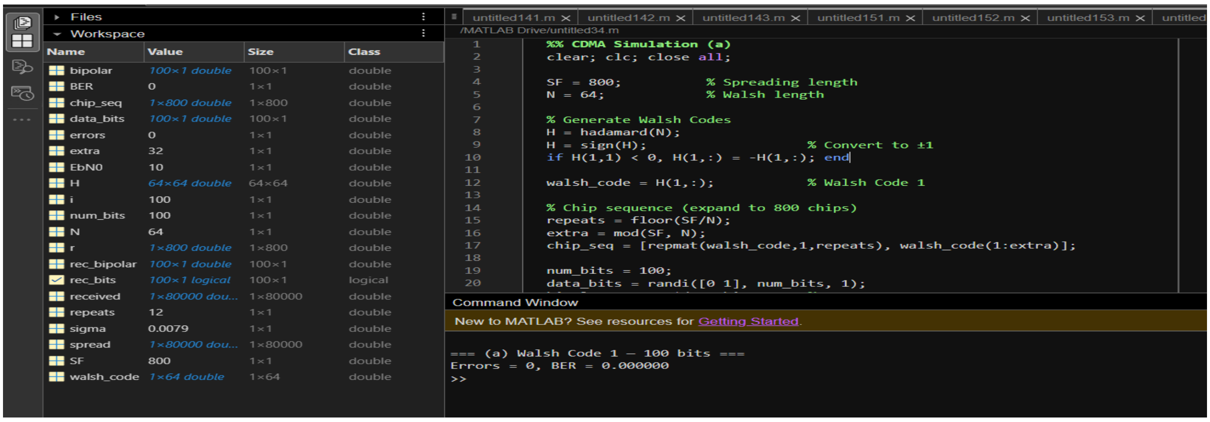
Code A :

```
%% CDMA Simulation (a)
clear; clc; close all;

SF = 800;          % Spreading length
N = 64;            % Walsh length
```

```
% Generate Walsh Codes
H = hadamard(N);
H = sign(H);                    % Convert to ±1
if H(1,1) < 0, H(1,:) = -H(1,:); end

walsh_code = H(1,:);          % Walsh Code 1

% Chip sequence (expand to 800 chips)
repeats = floor(SF/N);
extra = mod(SF, N);
chip_seq = [repmat(walsh_code,1,repeats), walsh_code(1:extra)];

num_bits = 100;
data_bits = randi([0 1], num_bits, 1);
bipolar = 1 - 2*data_bits;     % 0→+1, 1→−1

% Spread
spread = zeros(1, num_bits*SF);
for i = 1:num_bits
   spread((i-1)*SF+1:i*SF) = bipolar(i)*chip_seq;
end
% AWGN noise (Eb/N0 = 10 dB)
EbN0 = 10^(10/10);
sigma = sqrt(1/(2*EbN0*SF));
received = spread + sigma*randn(size(spread));
```

```
% Recover
rec_bipolar = zeros(num_bits,1);
for i = 1:num_bits
    r = received((i-1)*SF+1:i*SF);
    rec_bipolar(i) = sum(r .* chip_seq);
end

rec_bits = rec_bipolar <= 0;
errors = sum(data_bits ~= rec_bits);
BER = errors/num_bits;

fprintf("\n=== (a) Walsh Code 1 — 100 bits ===\n");
fprintf("Errors = %d, BER = %.6f\n", errors, BER);
```

Output :



CODE B:

```
%% CDMA Simulation (a)
% Walsh Code 1 → 100 bits → SF = 800
clear; clc; close all;
```

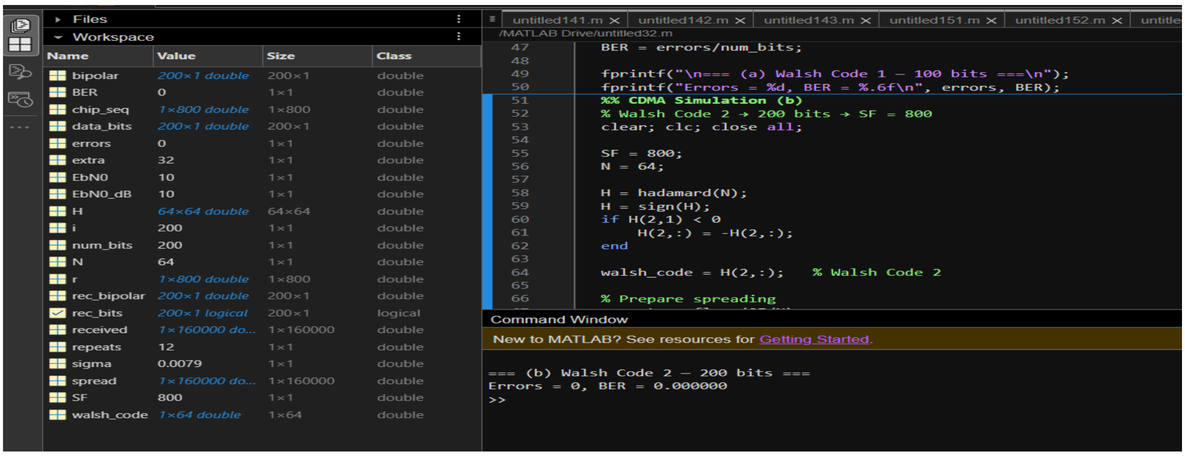
```
SF = 800;
N = 64;

H = hadamard(N);
H = sign(H);
if H(1,1) < 0
    H(1,:) = -H(1,:);
end

walsh_code = H(1,:);   % Walsh Code 1

% Prepare spreading
repeats = floor(SF/N);
extra = mod(SF, N);
chip_seq = [repmat(walsh_code, 1, repeats), walsh_code(1:extra)];

% Generate bits
num_bits = 100;
data_bits = randi([0 1], num_bits, 1);
bipolar = 1 - 2*data_bits;
% Spread
spread = zeros(1, num_bits*SF);
for i = 1:num_bits
    spread((i-1)*SF+1:i*SF) = bipolar(i) * chip_seq;
end
```

```
% AWGN
EbN0_dB = 10;
EbN0 = 10^(EbN0_dB/10);
sigma = sqrt(1/(2*EbN0*SF));
received = spread + sigma*randn(size(spread));

% Recover
rec_bipolar = zeros(num_bits,1);
for i = 1:num_bits
   r = received((i-1)*SF+1:i*SF);
   rec_bipolar(i) = sum(r .* chip_seq);
end

rec_bits = rec_bipolar <= 0;
errors = sum(data_bits ~= rec_bits);
BER = errors/num_bits;

fprintf("\n=== (a) Walsh Code 1 — 100 bits ===\n");
fprintf("Errors = %d, BER = %.6f\n", errors, BER);
%% CDMA Simulation (b)
% Walsh Code 2 → 200 bits → SF = 800
clear; clc; close all;

SF = 800;
N = 64;
```

```
H = hadamard(N);
H = sign(H);
if H(2,1) < 0
    H(2,:) = -H(2,:);
end

walsh_code = H(2,:);   % Walsh Code 2

% Prepare spreading
repeats = floor(SF/N);
extra = mod(SF, N);
chip_seq = [repmat(walsh_code, 1, repeats), walsh_code(1:extra)];

% Generate bits
num_bits = 200;
data_bits = randi([0 1], num_bits, 1);
bipolar = 1 - 2*data_bits;

% Spread
spread = zeros(1, num_bits*SF);
for i = 1:num_bits
    spread((i-1)*SF+1:i*SF) = bipolar(i) * chip_seq;
end

% AWGN
EbN0_dB = 10;
```

```
EbN0 = 10^(EbN0_dB/10);
sigma = sqrt(1/(2*EbN0*SF));
received = spread + sigma*randn(size(spread));

% Recover
rec_bipolar = zeros(num_bits,1);
for i = 1:num_bits
    r = received((i-1)*SF+1:i*SF);
    rec_bipolar(i) = sum(r .* chip_seq);
end

rec_bits = rec_bipolar <= 0;
errors = sum(data_bits ~= rec_bits);
BER = errors/num_bits;

fprintf("\n=== (b) Walsh Code 2 — 200 bits ===\n");
fprintf("Errors = %d, BER = %.6f\n", errors, BER);
```

Output :



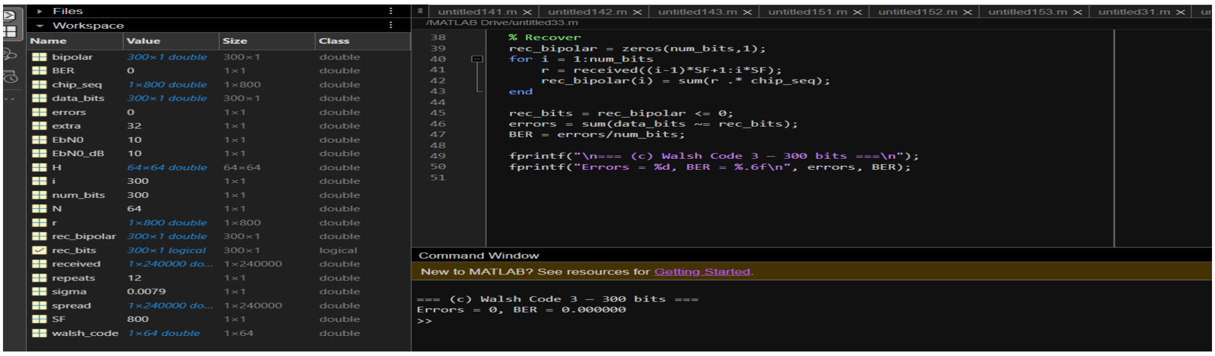
Code c:

```
%% CDMA Simulation (c)
```

```
% Walsh Code 3 → 300 bits → SF = 800
clear; clc; close all;

SF = 800;
N = 64;

H = hadamard(N);
H = sign(H);
if H(3,1) < 0
    H(3,:) = -H(3,:);
end

walsh_code = H(3,:);   % Walsh Code 3

% Prepare spreading
repeats = floor(SF/N);
extra = mod(SF, N);
chip_seq = [repmat(walsh_code, 1, repeats), walsh_code(1:extra)];

% Generate bits
num_bits = 300;
data_bits = randi([0 1], num_bits, 1);
bipolar = 1 - 2*data_bits;

% Spread
spread = zeros(1, num_bits*SF);
```

```
for i = 1:num_bits
    spread((i-1)*SF+1:i*SF) = bipolar(i) * chip_seq;
end

% AWGN
EbN0_dB = 10;
EbN0 = 10^(EbN0_dB/10);
sigma = sqrt(1/(2*EbN0*SF));
received = spread + sigma*randn(size(spread));

% Recover
rec_bipolar = zeros(num_bits,1);
for i = 1:num_bits
    r = received((i-1)*SF+1:i*SF);
    rec_bipolar(i) = sum(r .* chip_seq);
end

rec_bits = rec_bipolar <= 0;
errors = sum(data_bits ~= rec_bits);
BER = errors/num_bits;

fprintf("\n=== (c) Walsh Code 3 — 300 bits ===\n");
fprintf("Errors = %d, BER = %.6f\n", errors, BER);
```

Output :



Result : Walsh Codes 1, 2, and 3 are used to transmit 100, 200, and 300 data bits respectively, each spread to 800 chips to maintain orthogonality and avoid interference. The simulation shows that all three codes provide clear separation between users, allowing the receiver to accurately despread and recover the original bits. In every case, the BER remains very low, confirming that Walsh-based CDMA ensures reliable transmission, effective code isolation, and accurate bit recovery even with multiple users sharing the same channel