EXP2

SECTORING  
clc;

close all;

clear all;

sectoring\_type = input('Enter the type of sectoring (360, 120, or 60 degrees): ', 's');

if strcmp(sectoring\_type, '360')

N = 12; io = 6; n = 4;

S\_I = (sqrt(3 \* N))^n / io;

fprintf('For 360-degree sectoring:\nS/I in standard scale: %.4f\nS/I in dB scale: %.2f dB\n', S\_I, 10 \* log10(S\_I));

elseif strcmp(sectoring\_type, '120')

N = 7; io = 2; n = 4;

S\_I = (sqrt(3 \* N))^n / io;

fprintf('For 120-degree sectoring:\nS/I in dB scale: %.2f dB\n', 10 \* log10(S\_I));

elseif strcmp(sectoring\_type, '60')

N = 4; io = 1; n = 4;

S\_I = (sqrt(3 \* N))^n / io;

fprintf('For 60-degree sectoring:\nS/I in dB scale: %.2f dB\n', 10 \* log10(S\_I));

else360

disp('Invalid sectoring type. Please enter "360", "120", or "60".');

end

EXP3

Implementation of Free space and Two Ray Model, understanding of nature of Path Loss.

If a transmitter produces 50 watts of power, express the transmit power in units of (a) dBm, and (b) dBW. If 50 watts is applied to a unity gain antenna with a 900 MHz carrier frequency, find the received power in dBm at a free space distance of 100 m from the antenna, What is P (10 km) . Assume unity gain for the receiver antenna.

Input these values:

Pt= 50 W, Gt=1, Gr=1, d=10 Km, f=900 MHz.

1. Let the nature of the model be user defined.

2. Take user defined values of Pt, f, Gt, Gr.

3. Vary distance between Tx and Rx from 10 km to 100 km in steps of 10.

4. Find power received using Free Space model in dBW and dBm.

5. Declare ht=50m and hr=1.5m and find power received in dBW and dBm using Two Ray Gnd Reflection model for same range of distances.

6. Develop understanding of PL vs frequency

Vary the frequency from 900 MHz to 1800 MHz in steps of 1MHz and study the curve obtained for received power for free space.

% Given values

Pt = 50; % Transmit power in Watts

Gt = 1; % Transmit antenna gain

Gr = 2; % Receive antenna gain

f = 900e6; % Frequency in Hz

ht = 50; % Transmit antenna height in meters

hr = 1.5; % Receive antenna height in meters

% Speed of light

c = 3e8;

% Frequency in Hz

lambda = c / f;

% Distances in km

distances\_km = 10:10:100;

distances\_m = distances\_km \* 1000; % Convert to meters

% Initialize vectors for received power

Pr\_free\_space\_dBW = zeros(size(distances\_m));

Pr\_free\_space\_dBm = zeros(size(distances\_m));

Pr\_two\_ray\_dBW = zeros(size(distances\_m));

Pr\_two\_ray\_dBm = zeros(size(distances\_m));

% Loop over distances

for i = 1:length(distances\_m)

d = distances\_m(i);

% Free Space Model

Pr\_free\_space = Pt \* Gt \* Gr \* (lambda / (4 \* pi \* d))^2;

Pr\_free\_space\_dBW(i) = 10 \* log10(Pr\_free\_space);

Pr\_free\_space\_dBm(i) = 10 \* log10(Pr\_free\_space \* 1e3);

% Two Ray Ground Reflection Model

% Breakpoint distance

d\_break = 4 \* pi \* ht \* hr / lambda;

if d <= d\_break

% Free space equation before breakpoint distance

Pr\_two\_ray = Pr\_free\_space;

else

% Two ray ground reflection model after breakpoint distance

Pr\_two\_ray = Pt \* Gt \* Gr \* (ht \* hr / d^2)^2;

end

Pr\_two\_ray\_dBW(i) = 10 \* log10(Pr\_two\_ray);

Pr\_two\_ray\_dBm(i) = 10 \* log10(Pr\_two\_ray \* 1e3);

end

% Display results

disp('Distance (km) Free Space (dBW) Free Space (dBm) Two Ray (dBW) Two Ray (dBm)');

for i = 1:length(distances\_km)

fprintf('%10.1f %16.2f %16.2f %13.2f %13.2f\n', distances\_km(i), Pr\_free\_space\_dBW(i), Pr\_free\_space\_dBm(i), Pr\_two\_ray\_dBW(i), Pr\_two\_ray\_dBm(i));

end

% Plotting

figure;

subplot(2, 1, 1);

plot(distances\_km, Pr\_free\_space\_dBW, '-o', 'DisplayName', 'Free Space Model');

hold on;

plot(distances\_km, Pr\_two\_ray\_dBW, '-x', 'DisplayName', 'Two Ray Ground Reflection Model');

xlabel('Distance (km)');

ylabel('Received Power (dBW)');

title('Received Power (dBW) vs Distance');

legend show;

grid on;

subplot(2, 1, 2);

plot(distances\_km, Pr\_free\_space\_dBm, '-o', 'DisplayName', 'Free Space Model');

hold on;

plot(distances\_km, Pr\_two\_ray\_dBm, '-x', 'DisplayName', 'Two Ray Ground Reflection Model');

xlabel('Distance (km)');

ylabel('Received Power (dBm)');

title('Received Power (dBm) vs Distance');

legend show;

grid on;

Part B

Plot the graph for received power at the mobile using the 2-ray ground reflection model assuming the height of the transmitting antenna is 50 m and the receiving antenna is 1.5 m above ground. Vary the frequency from 900 MHz to 1800 MHz in steps of 1MHz and study the curve obtained for received power for two ray ground reflection model.

PART B

% Given values

Pt = 50; % Transmit power in Watts

Gt = 1; % Transmit antenna gain

Gr = 2; % Receive antenna gain

d = 10e3; % Distance in meters (10 km)

% Speed of light

c = 3e8;

% Frequencies in Hz

frequencies\_MHz = 900:1:1800;

frequencies\_Hz = frequencies\_MHz \* 1e6;

% Initialize vector for received power

Pr\_free\_space\_dBW = zeros(size(frequencies\_Hz));

Pr\_free\_space\_dBm = zeros(size(frequencies\_Hz));

% Loop over frequencies

for i = 1:length(frequencies\_Hz)

f = frequencies\_Hz(i);

lambda = c / f;

% Free Space Model

Pr\_free\_space = Pt \* Gt \* Gr \* (lambda / (4 \* pi \* d))^2;

Pr\_free\_space\_dBW(i) = 10 \* log10(Pr\_free\_space);

Pr\_free\_space\_dBm(i) = 10 \* log10(Pr\_free\_space \* 1e3);

end

% Plotting

figure;

subplot(2, 1, 1);

plot(frequencies\_MHz, Pr\_free\_space\_dBW, '-o');

xlabel('Frequency (MHz)');

ylabel('Received Power (dBW)');

title('Received Power (dBW) vs Frequency');

grid on;

subplot(2, 1, 2);

plot(frequencies\_MHz, Pr\_free\_space\_dBm, '-o');

xlabel('Frequency (MHz)');

ylabel('Received Power (dBm)');

title('Received Power (dBm) vs Frequency');

grid on;

% Display results for first few frequencies for verification

disp('Frequency (MHz) Free Space (dBW) Free Space (dBm)');

for i = 1:10

fprintf('%15.1f %16.2f %16.2f\n', frequencies\_MHz(i), Pr\_free\_space\_dBW(i), Pr\_free\_space\_dBm(i));

end

Part C:

A mobile is located 5 km away from a base station and uses a vertical monopole antenna with a gain of 2.55 dB to receive cellular radio signals. The E-field at 1 km from the transmitter is measured to be 10 ^-3 V/m.The carrier frequency used for this system is 900 MHz.

(a) Find the length and the effective aperture of the receiving antenna.

(b) Find the received power at the mobile using the 2-ray ground reflection model assuming the height of the transmitting antenna is 50 m and the receiving antenna is 1.5 m above ground.

f = 900e6;

c = 3e8;

Gt = 1.791;

d = 5e3;

d\_o = 1e3;

E\_o = 1e-3;

h\_t = 50;

h\_r = 1.5;

lambda = c / f;

L = lambda / 4;

Ae = Gt \* (lambda^2) / (4 \* pi);

Ep\_d = 2 \* E\_o \* d\_o \* (2 \* h\_t \* h\_r \* pi) / (d^2 \* lambda);

Pr = (Ep\_d^2) \* Ae / 377;

disp(['Length(L): ', num2str(L), 'meters']);

disp(['Effective aperture (Ae): ', num2str(Ae), ' square meters']);

disp(['Electric field at distance d, Ep(d): ', num2str(Ep\_d), ' V/m']);

disp(['Received power (Pr): ', num2str(Pr), ' Watts']);

EXP4

**a. To predict the type of small scale fading channel according to coherence bandwidth.**

**b. To predict the type of small scale fading channel according to Doppler spread.**

clc;

clear all;

close all;

% Part A: Coherence Bandwidth and Channel Type Prediction

% Step 1: Take user-defined values of RMS delay spread (τ\_rms) and frequency correlation (fc)

tau\_rms = input('Enter the RMS delay spread (in seconds): ');

fc = input('Enter the frequency correlation (fc): ');

% Validate inputs

if tau\_rms <= 0

fprintf('Error: RMS delay spread must be a positive number.\n');

elseif fc < 0.5 || fc > 1

fprintf('Error: Frequency correlation (fc) must be between 0.5 and 1.\n');

else

% Step 2: Find coherence bandwidth (Bc)

if fc >= 0.9

Bc = 1 / (50 \* tau\_rms);

elseif fc >= 0.5 && fc < 0.9

Bc = 1 / (5 \* tau\_rms);

end

fprintf('The coherence bandwidth (Bc) is: %.2f Hz\n', Bc);

% Step 3: Take user-defined values of signal bandwidth (Bs) and symbol period (Ts)

Bs = input('Enter the signal bandwidth (in Hz): ');

Ts = input('Enter the symbol period (in seconds): ');

% Predict the type of channel based on coherence bandwidth and signal bandwidth

if Bs <= Bc

fprintf('The channel is a Flat Fading Channel.\n');

else

fprintf('The channel is a Frequency Selective Fading Channel.\n');

end

end

% Part B: Doppler Shift, Coherence Time, and Channel Type Prediction

% Step 1: Take user-defined velocity (v) in m/s and carrier frequency (f\_c)

v = input('Enter the velocity (in m/s): ');

f\_c = input('Enter the carrier frequency (in Hz): ');

% Validate inputs

if v < 0

fprintf('Error: Velocity must be a non-negative number.\n');

elseif f\_c <= 0

fprintf('Error: Carrier frequency must be a positive number.\n');

else

% Step 2: Calculate maximum Doppler shift (f\_d)

c = 3e8; % Speed of light in m/s

f\_d = (v / c) \* f\_c;

fprintf('The maximum Doppler shift (f\_d) is: %.2f Hz\n', f\_d);

% Step 3: Calculate coherence time (Tc)

Tc = (0.423 / f\_d);

fprintf('The coherence time (Tc) is: %.2f seconds\n', Tc);

% Step 4: Take user-defined values of signal bandwidth (Bs) and symbol period (Ts)

Bs = input('Enter the signal bandwidth (in Hz): ');

Ts = input('Enter the symbol period (in seconds): ');

% Predict the type of channel based on coherence time and symbol period

if Ts <= Tc

fprintf('The channel is a Slow Fading Channel.\n');

else

fprintf('The channel is a Fast Fading Channel.\n');

end

end

EXP5

Part A: The total spectrum allocation is 12.5 MHZ and the guard band allocated at the edge of the spectrum is 10KHz for a AMPS system. Determine the number of channels available for a FDMA system.

Code: clc; clear all; close all; Bt = 12.5\*10e6; Bg = 10\*10e3; Bc = 30\*10e3; N = (Bt-2\*Bg)/Bc; fprintf('Number of available channels: %d\n', floor(N));

Consider Global System for Mobile, which is a TDMA system that uses 25 MHz for the forward link, which is broken into radio channels of 200 kHz. If 8 speech channels are supported on a single radio channel, and if no guard band is assumed, find the number of simultaneous users that can be accommodated in GSM.

Code: clc; clear all; close all; Bt = 25\*10e6; Bc = 200\*10e3; speech\_channels = 8; N\_Radio\_Channels = Bt/Bc; Total\_Users = N\_Radio\_Channels\*speech\_channels; fprintf("Total number of simultaneous users: %d\n", Total\_Users);

ii) If GSM uses a frame structure where each frame consists of 8 time slots, and each time slot contains 156.25 bits, and data is transmitted at 270.833 kbps in the channel, find (a) the time duration of a bit, (b) the time duration of a slot, (c) the time duration of a frame, (d) how long must a user occupying a single time slot must wait between two simultaneous solutions. Code:

clc; clear all; close all; data\_rate\_kbps = 270.833\*10e3; bits\_per\_slot = 156.25; slots\_per\_frame = 8; time\_duration\_of\_a\_bit = 1 / data\_rate\_kbps; time\_duration\_of\_a\_slot = bits\_per\_slot\*time\_duration\_of\_a\_bit; time\_duration\_of\_a\_frame = slots\_per\_frame\*time\_duration\_of\_a\_slot; wait\_time\_between\_transmissions = time\_duration\_of\_a\_frame-time\_duration\_of\_a\_slot; fprintf('Time Duration of a Bit: %.8f seconds\n', time\_duration\_of\_a\_bit); fprintf('Time Duration of a Slot: %.8f seconds\n', time\_duration\_of\_a\_slot); fprintf('Time Duration of a Frame: %.8f seconds\n', time\_duration\_of\_a\_frame); fprintf('Wait Time Between Two Simultaneous Transmissions: %.8f seconds\n', wait\_time\_between\_transmissions);

iii) If a normal GSM time slot consists of 6 trailing bits, 8.25 guard bits, 26 training bits, and 2 traffic bursts of 58 bits of data, find the frame efficiency. Code: clc;

clear all; close all; trailing\_bits = 6; guard\_bits = 8.25; training\_bits = 26; data\_bits\_per\_burst = 58; number\_of\_bursts = 2; total\_data\_bits = number\_of\_bursts \* data\_bits\_per\_burst; total\_bits\_in\_time\_slot = trailing\_bits + guard\_bits + training\_bits + total\_data\_bits; frame\_efficiency = total\_data\_bits / total\_bits\_in\_time\_slot; fprintf('Frame Efficiency: %.4f\n', frame\_efficiency \* 100);

EXP6

function exp6()

% Main script to take user input and generate PN code

disp('Enter a 4-bit binary array (e.g., [1 0 1 0]):');

array = input('', 's'); % Read user input as a string

array = str2num(array); % Convert string to numeric array

% Debugging output to check the input array

disp('Input array:');

disp(array);

% Validate input

if length(array) != 4

error('Input must be a 4-bit binary array.');

endif

% Validate if all elements are binary

if any(array < 0 | array > 1)

error('Array must contain only binary digits (0 or 1).');

endif

% Initialize PN code and shift register

pn\_code = [];

shift\_register = array;

% Generate PN code (length = 15 for a 4-bit shift register)

for i = 1:15

% Append the leftmost bit to the PN code

pn\_code = [pn\_code, shift\_register(1)];

% Calculate the feedback bit using XOR of specific bits

feedback\_bit = xor(shift\_register(1), shift\_register(2)); % Simple feedback for example

% Shift the register

shift\_register = [shift\_register(2:4), feedback\_bit];

endfor

% Display the PN code and its length

disp('Generated PN code:');

disp(pn\_code);

disp(['Length of PN code: ', num2str(length(pn\_code))]);

endfunction

EXP7   
Task 1.

1. Let N be the number of users in a system, who are allotted Walsh codes.

2. Initialize a variable w=1.

Construct a 2x2 Walsh matrix H. (Hint: H=[w,w,w,~w]

3. Let the number of times this construction be repeated for N users be m. Note that m=ceil(log2(N)).

4. Ok Repeat construction of H m times to accommodate N users.

5. Display H.

CODE:

N = 16;

w = 1;

m = ceil(log2(N));

H = w;

for i = 1:m

H = [H, H; H, -H];

end

disp('Walsh Matrix H:');

disp(H);

TASK 2.

1. Generate the orthogonality test matrix ‘O’ in which the result of orthogonality is stored. For example, the matrix should look as follows for 4 users:

𝑂=[1001000000001001]

Code :

%continued below previous code

O = zeros(N, N);

for i = 1:N

for j = 1:N

O(i, j) = dot(H(i, :), H(j, :)) / N;

end

end

disp('Walsh Matrix H:');

disp(H)

disp('Orthogonality Matrix O:');

disp(O);