

Scilab Manual for
Wireless Communications
by Dr V. A. Sankar Ponnappalli
Electronics and Telecommunication
Engineering
Icfai Foundation For Higher Education¹

Solutions provided by
Dr V. A. Sankar Ponnappalli
Electronics and Telecommunication Engineering
Icfai Foundation For Higher Education

June 26, 2026

¹Funded by a grant from the National Mission on Education through ICT, <http://spoken-tutorial.org/NMEICT-Intro>. This Scilab Manual and Scilab codes written in it can be downloaded from the "Migrated Labs" section at the website <http://scilab.in>

Contents

List of Scilab Solutions	3
1 Study of Free Space Propagation-Path Loss Model	5
2 Study of Outdoor Propagation – Okumura Model	9
3 Study of Outdoor Propagation – Hata Model	14
4 Study of Multipath Fading Channel	18
5 Modulation and Demodulation of Binary Phase Shift Keying over Additive White Gaussian Noise (AWGN) channel	23
6 Modulation and Demodulation of Direct-Sequence Spread Spectrum	27
7 Simulation of TDMA Technique	32
8 Simulation of FDMA Technique	37
9 Simulation of CDMA Technique	41

List of Experiments

Solution 1.0	Path Loss Model	5
Solution 2.0	Okumura Model	9
Solution 3.0	Hata Model	14
Solution 4.0	Multipath Fading Channel	18
Solution 5.0	BER OF BPSK	23
Solution 6.0	DSSS	27
Solution 7.0	TDMA	32
Solution 8.0	FDMA	37
Solution 9.0	CDMA	41

List of Figures

1.1	Path Loss Model	6
2.1	Okumura Model	10
3.1	Hata Model	15
4.1	Multipath Fading Channel	19
5.1	BER OF BPSK	24
6.1	DSSS	28
7.1	TDMA	33
8.1	FDMA	38
9.1	CDMA	42

Experiment: 1

Study of Free Space Propagation-Path Loss Model

Scilab code Solution 1.0 Path Loss Model

```
1 // Study of Free Space Propagation-Path Loss Model (
    FSPL)
2 // OS – Windows 10
3 // Scilab 6.1.0
4 // Course Instructor Name: Dr. V. A. Sankar
    Ponnappalli
5 // Institute Name: ICFAI Foundation for Higher
    Education Hyderabad
6
7 clc;
8 clear;
9 close();
10
11 disp("Enter frequency in Hz (Recommended: 800e6 to
    2.5 e9):");
12 f = input("Frequency (Hz): ");
13
```

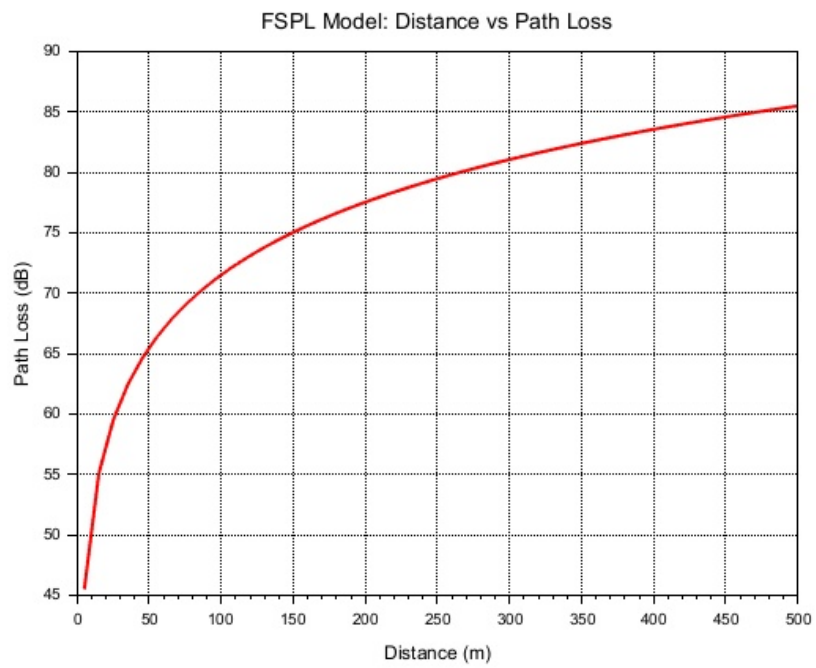


Figure 1.1: Path Loss Model

```

14 disp("Enter minimum distance in meters (Recommended:
      1 to 10 m):");
15 d_min = input("Minimum distance (m): ");
16
17 disp("Enter maximum distance in meters (Recommended:
      100 to 10000 m):");
18 d_max = input("Maximum distance (m): ");
19
20 disp("Enter number of distance samples (Recommended:
      50 to 500):");
21 n = input("Number of points: ");
22
23 c = 3e8; // Speed of light in m/s
24 lambda = c / f; // Wavelength in meters
25 d = linspace(d_min, d_max, n);
26
27 // Compute Free Space Path Loss (FSPL)
28 path_loss = 20*log10(d) + 20*log10(f) + 20*log10(4 *
      %pi / c);
29
30 scf();
31 plot(d, path_loss, 'r', 'LineWidth', 2);
32 xlabel('Distance (m)');
33 ylabel('Path Loss (dB)');
34 title('FSPL Model: Distance vs Path Loss');
35 h = gca();
36 h.grid = [1 1]; // Turn on both axes grid
37
38 // Description of the Figure
39 // User Inputs:
40 // Enter frequency in Hz (Recommended: 800e6 to 2.5
      e9):
41 // Frequency (Hz): 900e6
42 // Enter minimum distance in meters (Recommended: 1
      to 10 m):
43 // Minimum distance (m): 5
44 // Enter maximum distance in meters (Recommended:
      100 to 10000 m):

```

```
45 // Maximum distance (m): 500
46 // Enter number of distance samples (Recommended: 50
    to 500):
47 // Number of points: 50
48
49 // The figure titled "Interactive FSPL Model:
    Distance vs Path Loss" illustrates how free space
    path loss (in dB)
50 // increases logarithmically with distance in a line
    -of-sight communication environment.
```

Experiment: 2

Study of Outdoor Propagation – Okumura Model

Scilab code Solution 2.0 Okumura Model

```
1 // Study of Outdoor Propagation      Okumura Model
2 // OS: Windows 10
3 // Scilab Version: 6.1.0
4 // Course Instructor: Dr. V. A. Sankar Ponnappalli
5 // Institute: ICFAI Foundation for Higher Education,
  Hyderabad
6
7 clc;
8 clear;
9 close;
10
11 disp("Enter distance between T x Rx in km (Valid
    range: 1 100 km):");
12 d = input("Distance (km): ");
13
14 disp("Enter base station antenna height in meters (
    Valid range: 30 1000 m):");
```

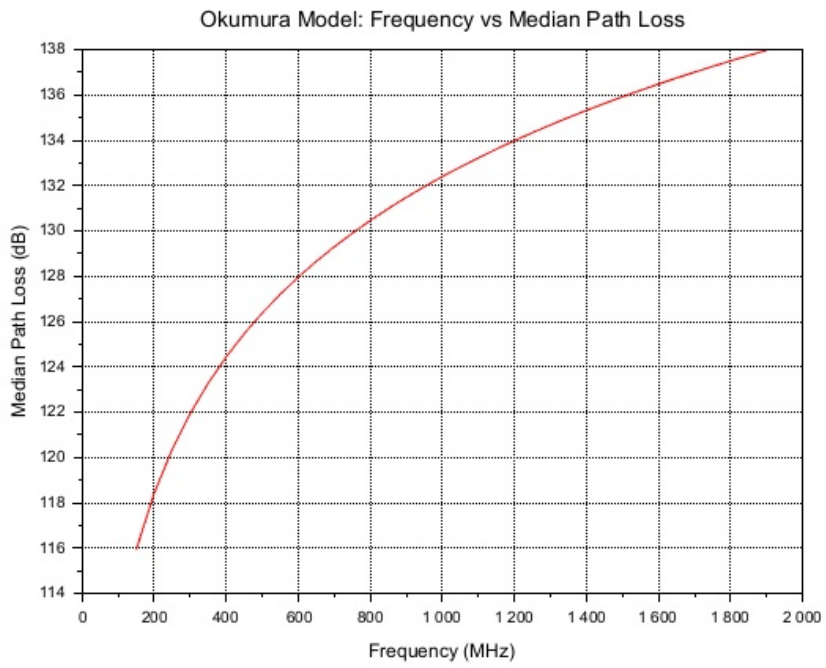


Figure 2.1: Okumura Model

```

15 ht = input("Base Station Height (m): ");
16
17 disp("Enter mobile station antenna height in meters
      (Valid range: 1 10 m):");
18 hr = input("Mobile Station Height (m): ");
19
20 disp("Choose Environment:");
21 disp("1. Urban");
22 disp("2. Suburban");
23 disp("3. Rural");
24 choice = input("Enter your choice (1 3 ): ");
25
26 select choice
27     case 1 then
28         environment = "urban";
29     case 2 then
30         environment = "suburban";
31     case 3 then
32         environment = "rural";
33     else
34         disp("Invalid choice. Defaulting to urban.")
35         ;
36         environment = "urban";
37 end
38 f_start = 150;      // MHz
39 f_end = 1920;      // MHz
40 f_step = 50;       // MHz
41 frequencies = f_start:f_step:f_end;
42
43 medianLoss = zeros(frequencies);
44
45 // _____ Path Loss Calculation
46
46 for i = 1:length(frequencies)
47     f = frequencies(i);
48
49     // Free space path loss

```

```

50     Lf = 32.4 + 20*log10(f) + 20*log10(d);
51
52     // Antenna gains
53     Gt = 20*log10(ht / 200);
54     Gr = 10*log10(hr / 3);
55
56     // Median path loss
57     Am = Lf - Gt - Gr;
58
59     // Apply environment correction
60     select environment
61         case "suburban" then
62             Am = Am - (2 * (log10(f / 28))^2) - 5.4;
63         case "rural" then
64             Am = Am - (4.78 * (log10(f))^2) + (18.33
65                 * log10(f)) - 40.94;
66
67     end
68     medianLoss(i) = Am;
69 end
70 scf(0);
71 plot(frequencies, medianLoss, 'r-');
72 xlabel("Frequency (MHz)");
73 ylabel("Median Path Loss (dB)");
74 title("Okumura Model: Frequency vs Median Path Loss"
75     );
76 xgrid();
77 // Description of the Figure
78 // User Inputs:
79 // "Enter distance between Tx Rx in km (Valid range
80 // : 1 100 km):"
81 // Distance (km): 100
82 // "Enter base station antenna height in meters (
83 // Valid range: 30 1000 m):"
84 // Base Station Height (m): 200
85 // "Enter mobile station antenna height in meters (

```

```
Valid range: 1 10 m):”
84 //Mobile Station Height (m): 3
85 //”Choose Environment:”
86 //”1. Urban”
87 //”2. Suburban”
88 //”3. Rural”
89 //Enter your choice (1 3 ): 1
90 //The figure titled ”Okumura Model: Frequency vs
Median Path Loss” illustrates how path loss
behaves with varying frequencies in an urban
setting and how the system parameters (distance ,
a//ntenna heights , and frequency) influence the
signal strength.
```

Experiment: 3

Study of Outdoor Propagation – Hata Model

Scilab code Solution 3.0 Hata Model

```
1 // Study of Outdoor Propagation      Hata Model
2 // OS: Windows 10
3 // Scilab: 6.1.0
4 // Course Instructor: Dr. V. A. Sankar Ponnappalli
5 // Institute: ICFAI Foundation for Higher Education
   Hyderabad
6
7 clc;
8 clear;
9 close;
10
11 disp("Enter Transmitter Height in meters (Valid
      range: 30 200 m):");
12 ht = input("Transmitter Height (m): ");
13
14 disp("Enter Receiver Height in meters (Valid range:
      1 10 m):");
```

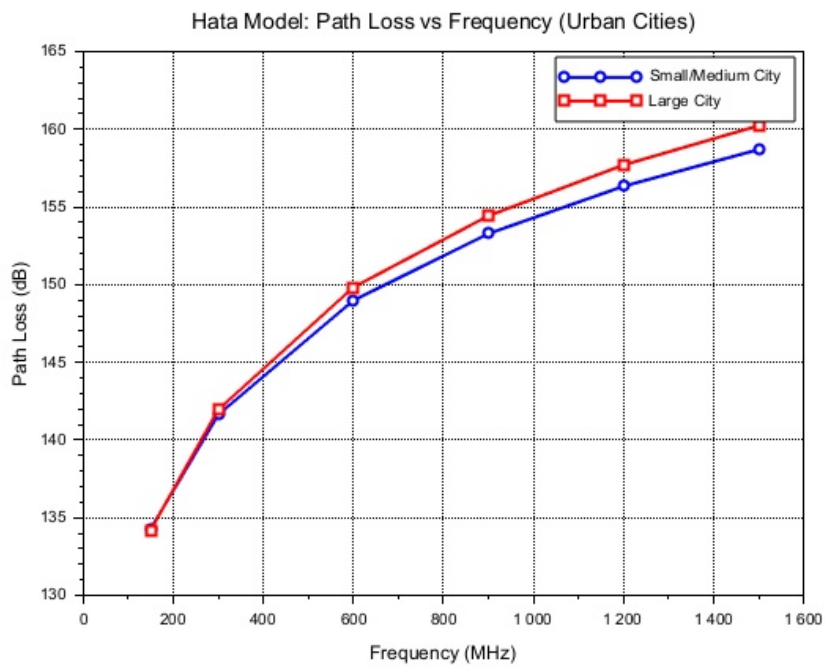


Figure 3.1: Hata Model

```

15 hr = input("Receiver Height (m): ");
16
17 disp("Enter Distance between T x Rx in km (Valid
    range: 1 20 km):");
18 d = input("Distance (km): ");
19
20 f_values = [150, 300, 600, 900, 1200, 1500]; // in
    MHz
21
22 Lp_small = zeros(1, length(f_values)); // Path loss
    for small/medium cities
23 Lp_large = zeros(1, length(f_values)); // Path loss
    for large cities
24
25 // _____ HATA MODEL CALCULATION
    _____
26
27 for i = 1:length(f_values)
28     f = f_values(i);
29
30     // Correction factor for small/medium cities
31     ah_small = (1.1 * log10(f) - 0.7) * hr - (1.56 *
        log10(f) - 0.8);
32
33     // Correction factor for large cities
34     if f <= 200 then
35         ah_large = 8.29 * (log10(1.54 * hr))^2 -
            1.1;
36     else
37         ah_large = 3.2 * (log10(11.75 * hr))^2 -
            4.97;
38     end
39
40     // Path Loss Calculations
41     Lp_small(i) = 69.55 + 26.16 * log10(f) - 13.82 *
        log10(ht) - ah_small + (44.9 - 6.55 * log10(
            ht)) * log10(d);
42     Lp_large(i) = 69.55 + 26.16 * log10(f) - 13.82 *

```

```

        log10(ht) - ah_large + (44.9 - 6.55 * log10(
        ht)) * log10(d);
43 end
44
45 scf(0);
46 plot(f_values, Lp_small, 'b-o', "LineWidth", 2);
47 plot(f_values, Lp_large, 'r-s', "LineWidth", 2);
48
49 xlabel("Frequency (MHz)");
50 ylabel("Path Loss (dB)");
51 title("Hata Model: Path Loss vs Frequency (Urban
        Cities)");
52 legend("Small/Medium City", "Large City", "location"
        , "upper_left");
53 xgrid();
54
55 // Description of the Figure
56 // User Inputs:
57 // "Enter Transmitter Height in meters (Valid range:
        30 200 m):"
58 // Transmitter Height (m): 50
59 // "Enter Receiver Height in meters (Valid range: 1
        10 m):"
60 // Receiver Height (m): 3
61 // "Enter Distance between T x Rx in km (Valid
        range: 1 20 km):"
62 // Distance (km): 10
63 // The figure titled "Hata Model: Path Loss vs
        Frequency (Urban Cities)" illustrates the
        variation of path loss with frequency for urban
        environments using the Hata propagation model.
64 // It compares path loss in two types of urban areas
        .

```

Experiment: 4

Study of Multipath Fading Channel

Scilab code Solution 4.0 Multipath Fading Channel

```
1 //Study of Multipath Fading Channel
2 // OS-Windows 10
3 // Scilab 6.1.0
4 // Course Instructor Name: Dr. V. A. Sankar
   Ponnapalli
5 // Institute Name: ICFAI Foundation for Higher
   Education Hyderabad
6
7 clc;
8 clear;
9 close;
10
11 printf("Enter number of frequency points N (power of
      2, between 128 and 4096):\n");
12 N = input("N = ");
13 if ~((floor(log2(N)) == log2(N)) & N >= 128 & N <=
      4096) then
```

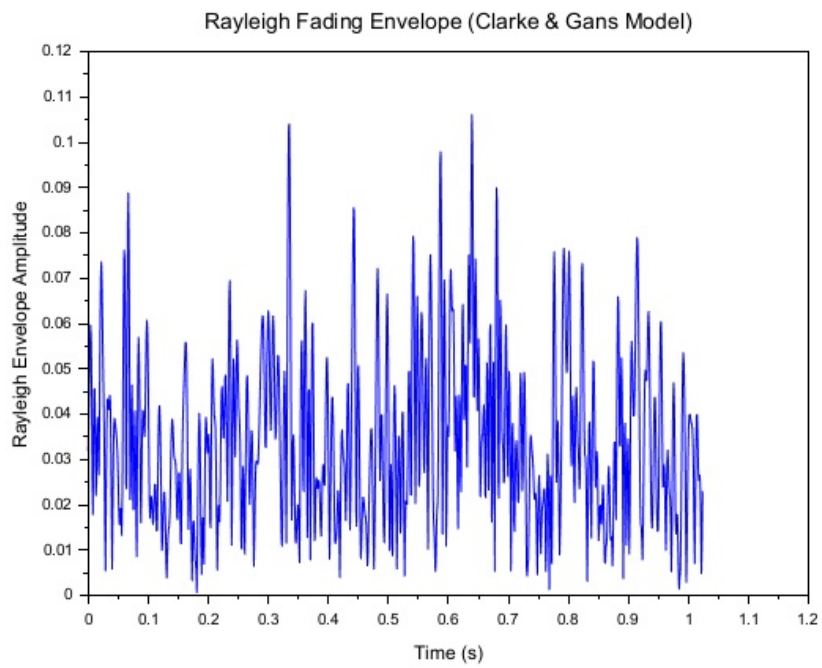


Figure 4.1: Multipath Fading Channel

```

14     error("Error: N must be a power of 2 between 128
        and 4096.");
15 end
16
17 // Input for max Doppler frequency fm
18 printf("Enter maximum Doppler frequency shift fm in
        Hz (e.g., between 10 and 500 Hz):\n");
19 fm = input("fm = ");
20 if fm <= 0 | fm > 1000 then
21     error("Error: fm must be between 10 and 1000 Hz.
        ");
22 end
23
24 // Input for sampling frequency fs
25 printf("Enter sampling frequency fs in Hz (must be >
        2*fm):\n");
26 fs = input("fs = ");
27 if fs <= 2*fm then
28     error("Error: fs must be greater than 2 * fm.");
29 end
30
31 // Derived values
32 df = fs / N;           // Frequency resolution
33 T = 1 / df;           // Duration of fading
        waveform
34 f = (-N/2:N/2-1)*df;   // Frequency axis
35
36 // ——— GENERATE DOPPLER SPECTRUM S_Eg(f) ———
37 Se = zeros(1, N);
38 for i = 1:N
39     f_norm = f(i) / fm;
40     if abs(f_norm) < 1 then
41         Se(i) = 1.5 / (%pi * fm * sqrt(1 - f_norm^2)
            );
42     else
43         Se(i) = 0;
44     end
45 end

```

```

46
47 // ——— GENERATE COMPLEX GAUSSIAN NOISE ———
48 rand("normal");
49 gauss_real = rand(1, N, "normal");
50 gauss_imag = rand(1, N, "normal");
51 Gf = (gauss_real + %i * gauss_imag) / sqrt(2);
52
53 // ——— SHAPE NOISE WITH DOPPLER SPECTRUM ———
54 Sqrt_Se = sqrt(Se);
55 Hf = Gf .* Sqrt_Se;
56
57 // ——— IFFT TO TIME DOMAIN ———
58 Hf_shifted = fftshift(Hf);
59 ht = ifft(Hf_shifted);
60 ht = ht * sqrt(N);           // Normalize
61
62 // ——— COMPUTE RAYLEIGH ENVELOPE ———
63 rayleigh_envelope = sqrt(real(ht).^2 + imag(ht).^2);
64
65 t = (0:N-1)/fs;
66 scf(1);
67 plot(t, rayleigh_envelope);
68 xlabel("Time (s)");
69 ylabel("Rayleigh Envelope Amplitude");
70 title("Rayleigh Fading Envelope (Clarke & Gans Model
       )");
71
72 // Description of the Figure
73 // User Inputs:
74 //Enter number of frequency points N (power of 2,
       between 128 and 4096):
75 //N = 1024
76 //Enter maximum Doppler frequency shift fm in Hz (e.
       g., between 10 and 500 Hz):
77 //fm = 100
78 //Enter sampling frequency fs in Hz (must be > 2*fm)
       :
79 //fs = 1000

```

80 // The figure titled "Rayleigh Fading Envelope (Clarke & Gans Model)" exhibits rapid variations typical of Rayleigh fading with many fades per second, consistent with a Doppler frequency //of 100 Hz.

Experiment: 5

Modulation and Demodulation of Binary Phase Shift Keying over Additive White Gaussian Noise (AWGN) channel

Scilab code Solution 5.0 BER OF BPSK

```
1 //Modulation and Demodulation of Binary Phase Shift
   Keying over Additive White Gaussian Noise (AWGN)
   channel
2 // OS-Windows 10
3 // Scilab 6.1.0
4 // Course Instructor Name: Dr. V. A. Sankar
   Ponnappalli
5 // Institute Name: ICFAI Foundation for Higher
   Education Hyderabad
6
7
8 clear;
9 clc;
```

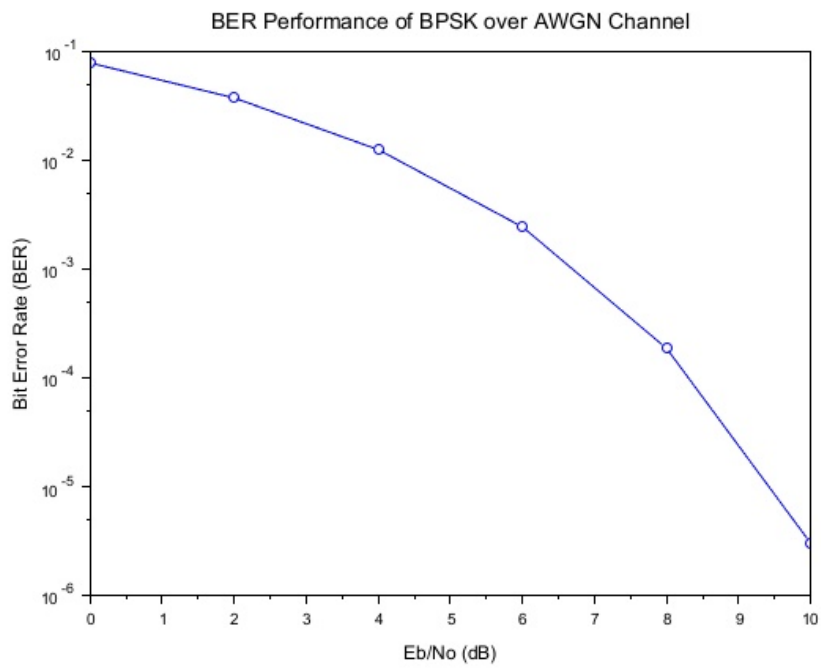


Figure 5.1: BER OF BPSK

```

10 close;
11
12 l = input("Enter number of bits to transmit (Choose
           between: 100000 to 1000000): ");
13 while l < 100000 | l > 1000000
14     l = input("Invalid input. Enter number of bits
           between 100000 and 1000000: ");
15 end
16
17 EbNodB = 0:2:10;           // Range of Eb/No in
           dB
18 BER = zeros(1, length(EbNodB)); // Initialize BER
           array
19
20 // Pick example Eb/N0 for printing outputs (middle
           of range)
21 example_idx = floor(length(EbNodB)/2);
22 example_EbNodB = EbNodB(example_idx);
23
24 for n = 1:length(EbNodB)
25     // Generate random BPSK symbols: -1 or 1
26     s = 2*(round(rand(1, 1))-0.5);
27
28     // AWGN noise with variance based on Eb/N0
29     noise_std = 1/sqrt(2*10^(EbNodB(n)/10));
30     w = noise_std * rand(1, 1, 'normal');
31
32     // Received signal
33     r = s + w;
34
35     // Demodulation: sign detector
36     s_est = sign(r);
37
38     // BER calculation
39     BER(n) = sum(s ~= s_est)/l;
40
41     // Save example bits for display
42     if n == example_idx then

```

```

43         bits_sent = (s + 1)/2;          // convert from
         -1/1 to 0/1 for clarity
44         bits_demod = (s_est + 1)/2;
45     end
46 end
47
48 // Display example output for the chosen Eb/N0
49 disp("Example Eb/N0 (dB): " + string(example_EbNodB)
      );
50 disp("First 50 generated bits (0/1):");
51 disp(bits_sent(1:50));
52 disp("First 50 demodulated bits (0/1):");
53 disp(bits_demod(1:50));
54
55 // Plot BER vs Eb/N0
56 scf();
57 semilogy(EbNodB, BER, 'o-');
58 xlabel('Eb/No (dB)');
59 ylabel('Bit Error Rate (BER)');
60 title('BER Performance of BPSK over AWGN Channel');
61
62 //// Description of the Figure
63 // User Inputs:
64 // Enter number of bits to transmit (Choose between:
        100000 to 1000000): 1e6
65 // The figure titled "BER Performance of BPSK over
        AWGN Channel" illustrates how BPSK performs over
        AWGN.

```

Experiment: 6

Modulation and Demodulation of Direct-Sequence Spread Spectrum

Scilab code Solution 6.0 DSSS

```
1 // Direct-Sequence Spread Spectrum (DSSS) Modulation
  and Demodulation
2 // OS: Windows 10
3 // Scilab Version: 6.1.0
4 // Course Instructor: Dr. V. A. Sankar Ponnappalli
5 // Institute: ICFAI Foundation for Higher Education
  Hyderabad
6
7 clc;
8 clear;
9 close;
10
11 N_bits = input("Enter number of data bits (e.g., 10
  to 100): ");
12 PG = input("Enter processing gain (e.g., 5 to 20): ")
```

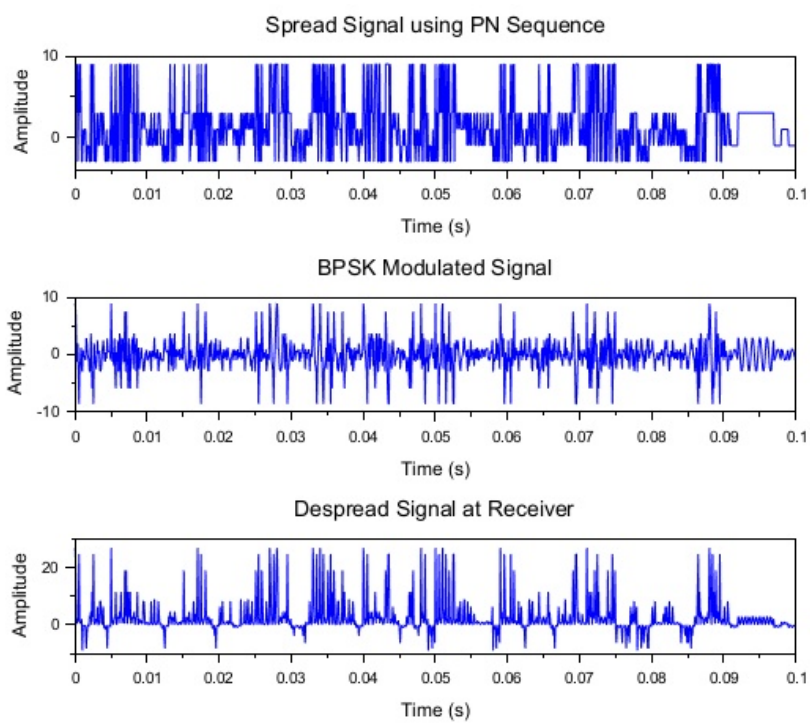


Figure 6.1: DSSS

```

    );
13 fc = input("Enter carrier frequency in Hz (e.g.,
    1000): ");
14 fs = input("Enter sampling frequency in Hz (must be
    > 10*fc): ");
15
16 // Validations
17 if fs <= 10 * fc then
18     error("Sampling frequency must be greater than
    10 times the carrier frequency.");
19 end
20
21 // Timing and signal parameters
22 Tb = 1e-3; // Bit duration (1 ms)
23 Tc = Tb / PG; // Chip duration
24 Ns = int(fs * Tb); // Samples per bit
25 Nc = int(fs * Tc); // Samples per chip
26 total_samples = N_bits * Ns; // Total number of
    samples
27 t = (0:total_samples - 1) / fs; // Time vector
28
29 // Generate random data bits (+1 or -1)
30 data_bits = 2 * grand(1, N_bits, "uin", 0, 2) - 1;
31 data_signal = matrix(ones(Ns, 1) * data_bits, 1, -1)
    ;
32
33 // Generate PN sequence long enough
34 num_chips = N_bits * PG;
35 pn_seq = 2 * grand(1, num_chips, "uin", 0, 2) - 1;
36 pn_signal_full = matrix(ones(Nc, 1) * pn_seq, 1, -1)
    ;
37
38 // Ensure PN signal has enough samples
39 if length(pn_signal_full) < total_samples then
40     // Pad if too short
41     extra_needed = total_samples - length(
        pn_signal_full);
42     pn_signal_full = [pn_signal_full, ones(1,

```

```

        extra_needed)];
43 end
44 pn_signal = pn_signal_full(1:total_samples); //
    Trim to exact length
45
46 // Spread signal
47 spread_signal = data_signal .* pn_signal;
48
49 // Carrier
50 carrier = cos(2 * %pi * fc * t);
51
52 // Modulate
53 modulated_signal = spread_signal .* carrier;
54
55 // Receiver: Demodulate
56 received = modulated_signal .* carrier;
57 despread = received .* pn_signal;
58
59 // Recover bits
60 recovered_bits = zeros(1, N_bits);
61 for i = 1:N_bits
62     idx_start = (i - 1) * Ns + 1;
63     idx_end = i * Ns;
64     recovered_bits(i) = sign(sum(despread(idx_start:
        idx_end)));
65 end
66
67 // Plot
68 clf();
69 subplot(3,1,1);
70 plot(t, spread_signal);
71 xtitle("Spread Signal using PN Sequence");
72 xlabel("Time (s)");
73 ylabel("Amplitude");
74
75 subplot(3,1,2);
76 plot(t, modulated_signal);
77 xtitle("BPSK Modulated Signal");

```

```

78 xlabel("Time (s)");
79 ylabel("Amplitude");
80
81 subplot(3,1,3);
82 plot(t, despread);
83 xtitle("Despread Signal at Receiver");
84 xlabel("Time (s)");
85 ylabel("Amplitude");
86
87 // Display bits
88 disp("Original bits: ");
89 disp(data_bits);
90 disp("Recovered bits: ");
91 disp(recovered_bits);
92
93 // Description of the Figure
94 // User Inputs:
95 // Enter number of data bits (e.g., 10 to 100): 100
96 // Enter processing gain (e.g., 5 to 20): 10
97 // Enter carrier frequency in Hz (e.g., 1000): 1000
98 // Enter sampling frequency in Hz (must be > 10*fc):
    11000
99 // The figure depicts Direct-Sequence Spread
    Spectrum Modulation and Demodulation.

```

Experiment: 7

Simulation of TDMA Technique

Scilab code Solution 7.0 TDMA

```
1 //Simulation of TDMA (Time Division Multiplexing and
   // Demultiplexing) Technique)
2 // OS-Windows 10
3 // Scilab 6.1.0
4 // Course Instructor Name: Dr. V. A. Sankar
   // Ponnappalli
5 // Institute Name: ICFAI Foundation for Higher
   // Education Hyderabad
6
7 clear;
8 clc;
9 close;
10
11 disp("==== TDMA System Simulation ====");
12
13 N_r = input("Enter number of traffic bursts per
   // frame (N_r) [1 100 ]: ");
```

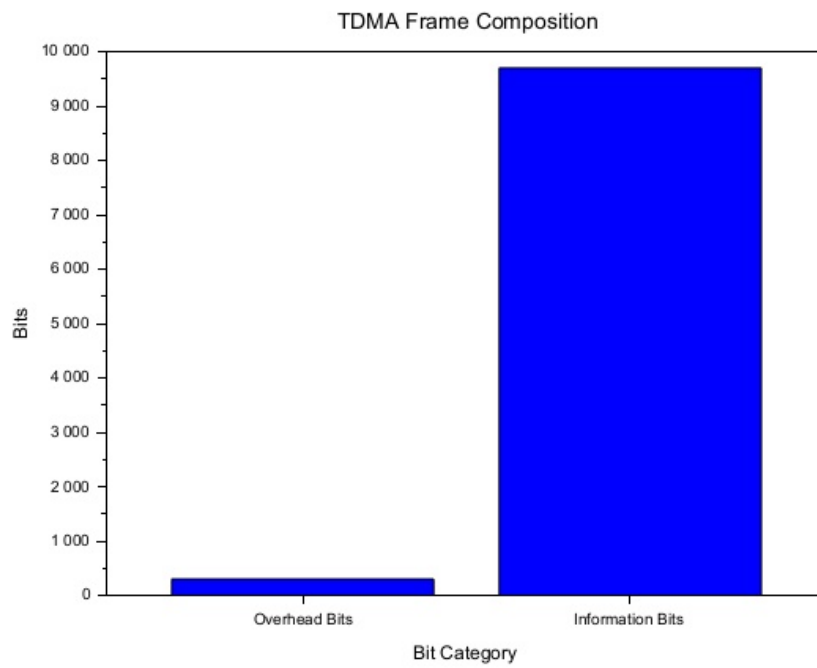


Figure 7.1: TDMA

```

14 b_r = input("Enter number of bits per traffic burst
      (b_r) [100 1000 ]: ");
15 N_p = input("Enter number of preambles per frame (
      N_p) [1 10 ]: ");
16 b_p = input("Enter bits per preamble (b_p) [10 100
      ]: ");
17 N_g = input("Enter number of guard intervals per
      frame (N_g) [1 20 ]: ");
18 b_g = input("Enter bits per guard time interval (b_g
      ) [1 50 ]: ");
19 T_f = input("Enter frame duration T_f in seconds [e.
      g., 0.01]: ");
20 R_b = input("Enter bit rate R_b in bits/sec [e.g., 1
      e6]: ");
21
22 //Overhead bits per frame
23 b_OH = N_p*b_p + N_g*b_g;
24 disp("Total overhead bits per frame (b_OH): " +
      string(b_OH));
25
26 //Total bits per frame
27 b_T = T_f * R_b;
28 disp("Total bits per frame (b_T): " + string(b_T));
29
30 //Frame Efficiency
31 eta_f = (1 - (b_OH / b_T)) * 100;
32 disp("Frame Efficiency ( eta_f ) in %: " + string(eta_f
      ));
33
34 // Number of TDMA Channels
35 m = input("Enter max users per channel (m) [1 100
      ]: ");
36 B_tot = input("Enter total system bandwidth B_tot in
      Hz [e.g., 200e3]: ");
37 B_guard = input("Enter guard band on one side (
      B_guard) in Hz [e.g., 10e3]: ");
38 B_c = input("Enter bandwidth per channel B_c in Hz [
      e.g., 25e3]: ");

```

```

39
40 N_channels = m * (B_tot - 2*B_guard) / B_c;
41 disp("Number of TDMA channels (N): " + string(
    N_channels));
42
43 x = [b_OH, b_T - b_OH];
44 bar(x);
45 xtitle("TDMA Frame Composition", "Bit Category", "
    Bits");
46 a = gca(); // get current axes
47 a.x_ticks = tlist(["ticks", "locations", "labels"],
    [1; 2], ["Overhead Bits"; "Information Bits"]);
48
49 // Description of the Figure
50 // User Inputs:
51
52 //Enter number of traffic bursts per frame (N_r) [1
    100 ]: 50
53 //Enter number of bits per traffic burst (b_r) [100
    1000 ]: 150
54 //Enter number of preambles per frame (N_p) [1 10
    ]: 5
55 //Enter bits per preamble (b_p) [10 100 ]: 50
56 //Enter number of guard intervals per frame (N_g) [1
    20 ]: 10
57 //Enter bits per guard time interval (b_g) [1 50 ]:
    5
58 //Enter frame duration T_f in seconds [e.g., 0.01]:
    0.01
59 //Enter bit rate R_b in bits/sec [e.g., 1e6]: 1e6
60 //Enter max users per channel (m) [1 100 ]: 50
61 //Enter total system bandwidth B_tot in Hz [e.g.,
    200e3]: 200e3
62 //Enter guard band on one side (B_guard) in Hz [e.g
    ., 10e3]: 10e3
63 //Enter bandwidth per channel B_c in Hz [e.g., 25e3
    ]: 25e3
64 //The left bar labeled    Overhead    Bits

```

represents preambles and guard intervals that
support synchronization and prevent interference.
65 //The right bar labeled Information Bits
represents actual user data.

Experiment: 8

Simulation of FDMA Technique

Scilab code Solution 8.0 FDMA

```
1 //Simulation of FDMA (Frequency Division
  Multiplexing and Demultiplexing) Technique
2 // OS-Windows 10
3 // Scilab 6.1.0
4 // Course Instructor Name: Dr. V. A. Sankar
  Ponnappalli
5 // Institute Name: ICFAI Foundation for Higher
  Education Hyderabad
6
7 clear;
8 clc;
9 close;
10
11 disp("=== FDMA System Capacity Simulation ===");
12
13 // Total bandwidth B_t
14 B_t = input("Enter total system bandwidth B_t in Hz
  [1e6      25e6]: ");
15 while B_t < 1e6 | B_t > 25e6
```

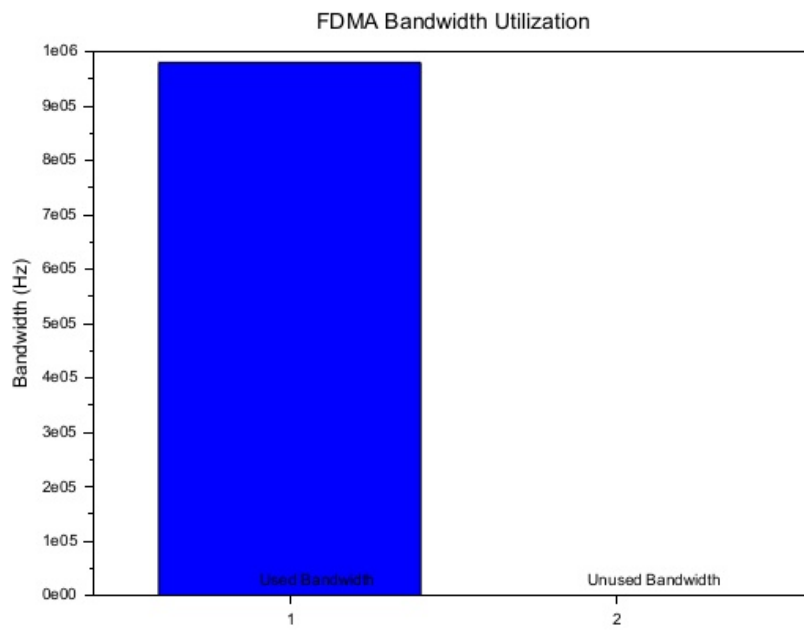


Figure 8.1: FDMA

```

16     B_t = input("Invalid! Enter B_t between 1e6 and
                25e6 Hz: ");
17 end
18
19 // Guard band per side
20 B_guard = input("Enter guard band on one side
                B_guard in Hz [1e4      5e5]: ");
21 while B_guard < 1e4 | B_guard > 5e5
22     B_guard = input("Invalid! Enter B_guard between
                1e4 and 5e5 Hz: ");
23 end
24
25 // Channel bandwidth B_c
26 B_c = input("Enter bandwidth per FDMA channel B_c in
                Hz [1e4      5e4]: ");
27 while B_c < 1e4 | B_c > 5e4
28     B_c = input("Invalid! Enter B_c between 1e4 and
                5e4 Hz: ");
29 end
30
31 //-----
32 // Calculation
33 //-----
34 if B_t <= 2*B_guard then
35     error("Total bandwidth must be greater than
            twice the guard band.");
36 end
37
38 B_available = B_t - 2 * B_guard;
39 N_channels = floor(B_available / B_c);
40 B_used = N_channels * B_c;
41 B_unused = B_available - B_used;
42 efficiency = (B_used / B_t) * 100;
43
44 mprintf("Total Bandwidth (B_t): %.2f Hz\n", B_t);
45 mprintf("Guard Band per side (B_guard): %.2f Hz\n",
        B_guard);
46 mprintf("Channel Bandwidth (B_c): %.2f Hz\n", B_c);

```

```

47 mprintf(" Available Bandwidth after guard bands: %.2 f
      Hz\n", B_available);
48 mprintf(" Number of FDMA Channels (N): %d\n",
      N_channels);
49 mprintf(" Total Used Bandwidth: %.2 f Hz\n", B_used);
50 mprintf(" Unused Bandwidth: %.2 f Hz\n", B_unused);
51 mprintf(" Bandwidth Utilization Efficiency: %.2 f %%\n
      ", efficiency);
52
53 scf(0);
54 bar([1 2], [B_used B_unused], "stacked");
55 xt = [1, 2];
56 labels = ["Used Bandwidth", "Unused Bandwidth"];
57 for i = 1:2
58     xstring(xt(i) - 0.1, 0, labels(i));
59 end
60 ylabel(" Bandwidth (Hz)");
61 title("FDMA Bandwidth Utilization");
62
63 // Description of the Figure
64 // User Inputs:
65 // Enter total system bandwidth B_t in Hz [1e6
      25e6]: 1e6
66 // Enter guard band on one side B_guard in Hz [1e4
      5e5]: 1e4
67 // Enter bandwidth per FDMA channel B_c in Hz [1e4
      5e4]: 1e4
68 // The figure titles "FDMA Bandwidth Utilization"
      shows how efficiently the bandwidth is utilized
      in an FDMA system given user-defined channel
      width and guard bands.

```

Experiment: 9

Simulation of CDMA Technique

Scilab code Solution 9.0 CDMA

```
1 //Simulation of CDMA (Code Division Multiplexing and
    Demultiplexing) Technique
2 // OS-Windows 10
3 // Scilab 6.1.0
4 // Course Instructor Name: Dr. V. A. Sankar
    Ponnappalli
5 // Institute Name: ICFAI Foundation for Higher
    Education Hyderabad
6
7 clear;
8 clc;
9 close;
10
11
12 W = input("Enter total RF bandwidth W in Hz (e.g.,
    1.25e6 for 1.25 MHz): ");
13 R = input("Enter data rate R per user in Hz (e.g.,
```

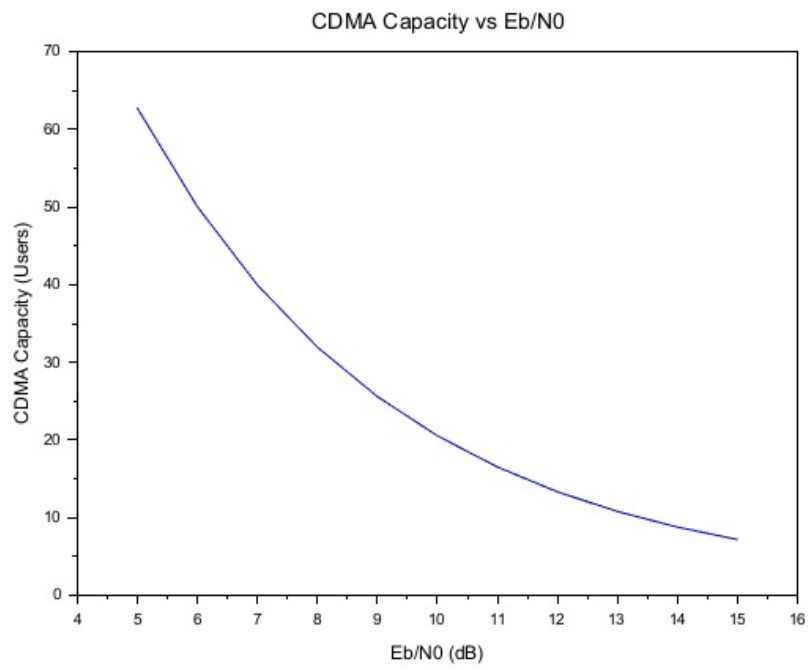


Figure 9.1: CDMA

```

    4800, 9600, 19200): ");
14 Eb_NO = input("Enter Eb/N0 in dB (suggested range: 5
    to 15 dB): ");
15 eta = input("Enter background noise fraction    (
    suggested range: 0.5 to 1): ");
16 alpha = input("Enter voice activity factor    (
    suggested range: 0.3 to 1): ");
17 sectorization_factor = input("Enter antenna
    sectorization factor (1 for omni, 3 for 120
    beamwidth): ");
18
19 // Convert Eb/N0 from dB to linear
20 Eb_NO_linear = 10^(Eb_NO/10);
21
22 S = 1; // Assume desired signal power S = 1 unit
23 N_basic = 1 + (W/R) / (Eb_NO_linear * (eta/S));
24
25 // Apply sectorization and voice activity factor
26 N_sector = (N_basic - 1) * alpha + (eta/S);
27 Eb_NO_sector = (W/R) / N_sector;
28
29 // Final capacity with sectoring and voice activity
30 N_final = 1 + (W/R) / Eb_NO_sector;
31
32 disp("===== CDMA Capacity Results =====");
33 disp("Processing Gain (W/R): " + string(W/R));
34 disp("Eb/N0 (linear): " + string(Eb_NO_linear));
35 disp("Estimated basic user capacity (no interference
    control): " + string(N_basic));
36 disp("Eb/N0 within sector with interference control:
    " + string(Eb_NO_sector));
37 disp("Estimated final user capacity (with
    sectorization and voice activity): " + string(
    N_final));
38
39 eb_range = 5:1:15;
40 N_plot = [];
41 for eb = eb_range

```

```

42     EbN0_lin = 10^(eb/10);
43     N_temp = 1 + (W/R) / (EbN0_lin * (eta/S));
44     N_plot($+1) = N_temp;
45 end
46
47 scf(0);
48 plot(eb_range, N_plot);
49 xlabel("Eb/N0 (dB)");
50 ylabel("CDMA Capacity (Users)");
51 title("Effect of Eb/N0 on CDMA Capacity");
52 xtitle("CDMA Capacity vs Eb/N0");
53
54 // Description of the Figure
55 // User Inputs:
56 // Enter total RF bandwidth W in Hz (e.g., 1.25e6
    for 1.25 MHz): 1.5e6
57 // Enter data rate R per user in Hz (e.g., 4800,
    9600, 19200): 9600
58 // Enter Eb/N0 in dB (suggested range: 5 to 15 dB):
    8
59 // Enter background noise fraction (suggested
    range: 0.5 to 1): 0.8
60 // Enter voice activity factor (suggested range:
    0.3 to 1): 0.5
61 // Enter antenna sectorization factor (1 for omni, 3
    for 120 beamwidth): 3
62 // The figure titled "CDMA Capacity vs Eb/N0"
    represents how the user capacity of a CDMA system
    changes with respect to the energy-per-bit to
    noise power spectral density ratio.

```
