```
%
% Output argument
% inhibit    : 0: idle   1: busy
%
% Programmed by M. Okita
% Checked by H. Harada
%

function [inhibit] = inhibitsense(no,now_time)

global Mnum Mstime Ttime Dtime
  % definition of the global variable

delay   = Dtime * Ttime;  % calculation of delay time
inhibit = 0;

idx = find((Mstime+delay) <= now_time ...
  & now_time <= (Mstime+delay+Ttime));% inhibit sensing
if length(idx) > 0
  idx = find(idx~=no);           % except itself
  if length(idx) > 0             % inhibit is detected
     inhibit = 1;                % channel is busy
  end
end

%%%%%%%%%%%%%%%%%%%% end of file %%%%%%%%%%%%%%%%%%%%
```

7

Cellular Telecommunications Systems

7.1 Introduction

In Chapters 1–6, the performance of point-to-point and point-to-multipoint transmissions was evaluated by using several transmission schemes. However, in a mobile communication system, a base station communicates with many users. This means point-to-point transmission is just one contributing factor in a mobile communication system. Thus, we need to evaluate not only the performance of point-to-point transmission, but also the per- formance of multipoint access schemes—or how performance is affected by several independent users. Particularly in a mobile communication system, the cellular telecommunication model is frequently used as the basic access scheme, because it can obtain high utilization efficiency of frequencies. This chapter will discuss a cellular telecommunication system model that is actually in use and achieving high system capacity [1, 2]. It is necessary to use computer simulations to evaluate for total system performance, such as system capacity, and blocking probability or forced termination probability of a call.

One advantage of using an access scheme model based on a cellular system is that it can be combined with some of the transmission schemes presented in previous chapters, enabling advanced evaluation. Indeed, it is often the case that a cellular system simulation is conducted by combining other complex factors of the communication process such as transmission power control and antenna array techniques, making primitive theoretical analysis of the system performance nearly impossible.

We therefore introduce the fundamental model of cellular system simulations. Several techniques used in our simulation are essential to real evaluations,

and at the same time are easily applicable to the above-mentioned probable situations.

The basic concept of a cellular telecommunication system and intercell interference is explained in Section 7.2. Then, in Section 7.3, some key techniques to construct the computer simulation model for a cellular system using specific software are introduced. Moreover, a technique to achieve the DCA algorithm, a well-known channel assignment scheme, is presented there. We present our simulation results using that algorithm in Section 7.4. Section 7.5 describes the model for a cellular system using an array antenna. Section 7.6 concludes the chapter. All programs related to the chapter are presented in Appendix 7A and in the CD-ROM accompanying the book.

7.2 Concept of Cellular System and Channel Assignment Algorithm

The basic concept of a cellular system is shown in Figure 7.1. In this system, the entire service area is divided into several small areas called cells. A base station located in the center of each cell allocates several traffic channels for mobile terminals within the cell when they initiate their calls. One of the most common features of this cellular system is that the same traffic channel can be concurrently allocated in different cells if they are sufficiently spaced apart. If the same channels are allocated in closer cells, each cell causes cochannel interference to the other cells and degrades the transmission quality. However, if the cells with the same channels are adequately spaced from the others, that interference is diminished by the distance between the cells; transmission quality is never corrupted by such interference, and the radio resources for the entire system can be utilized effectively.

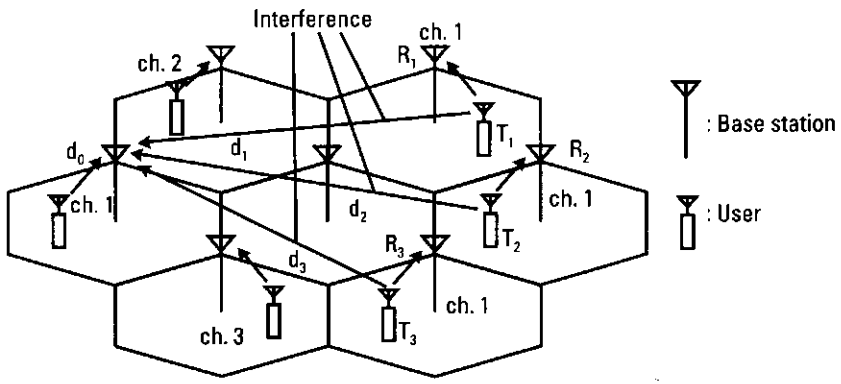

Figure 7.1 Cellular system concept.

In Figure 7.1, the four base stations are R_0, R_1, R_2, and R_3, and the four users are T_0, T_1, T_2, and T_3. Here, it is assumed that user T_i is connected to base station R_i, and that all of four users are allocated the same traffic channels. On this assumption, we can derive the carrier-to-noise plus interference ratio (C/(N + I)) R_{cni} for user T_0 as follows,

$$R_{cni} = \frac{AP_0 d_0^{-\alpha} 10^{\frac{\xi_0}{10}}}{N + \sum_{i=1}^{3} AP_i d_i^{-\alpha} 10^{\frac{\xi_i}{10}}} \tag{7.1}$$

Here, α is a path loss factor, A is a proportional coefficient, P_i is the transmitted power of user T_i, d_i is the distance between user T_i and base station R_0. In addition, ξ_i is the distortion caused by shadowing between T_i and R_0, the value of which is denoted by decibels in (7.1). From (7.1), we can see that the longer distances of d_1, d_2, d_3 result in a higher C/(N + I) ratio, which is equivalent to a higher quality of transmission. Therefore, to achieve an adequate C/(N + I) ratio for preliminary service quality, several channel assignment algorithms are being used in the current systems.

Two major examples of such channel assignment algorithms are FCA and DCA [3]. FCA is an algorithm that conducts a fixed assignment of channels per base station. An example of FCA is shown in Figure 7.2, where four channels are allocated to each base station so that users in a cell suffer from minimum

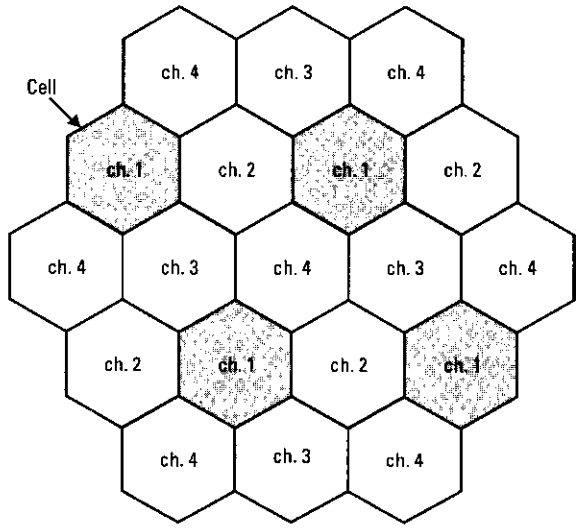

Figure 7.2 FCA algorithm concept.

cochannel interference from other cells. On the other hand, in the case of DCA, a base station can handle all the channels for the system and dynamically allocates suitable channels to minimize interference conditions. A concept of DCA is shown in Figure 7.3. The DCA algorithm can achieve higher capacity and cope more flexibly with fluctuations of traffic than FCA can, because all the channels are adaptively allocated according to users' demands. In the case of FCA, as shown in Figure 7.3(a), the second user in a cell faces call blocking, because the channels per base station are fixedly restricted. On the other hand, in case of DCA shown in Figure 7.3(b), the base stations all share the channels, which are adaptively allocated, thereby decreasing incidents of call blocking that occur with FCA. However, DCA needs a more complicated control scheme than FCA does, such as interference measurement and channel searching.

7.3 Simulation Program for Dynamic Channel Assignment System

This section presents a simulation program to realize a cellular system using the DCA algorithm. The main program is shown as Program 7.1, and subprograms used by the main program are shown as Programs 7.2–7.7. Table 7.1 summarizes the functions of each program.

7.3.1 Performance Measures

Before giving a detailed explanation about our simulation and programs, we will describe the performance measures in our simulation. We are focusing on

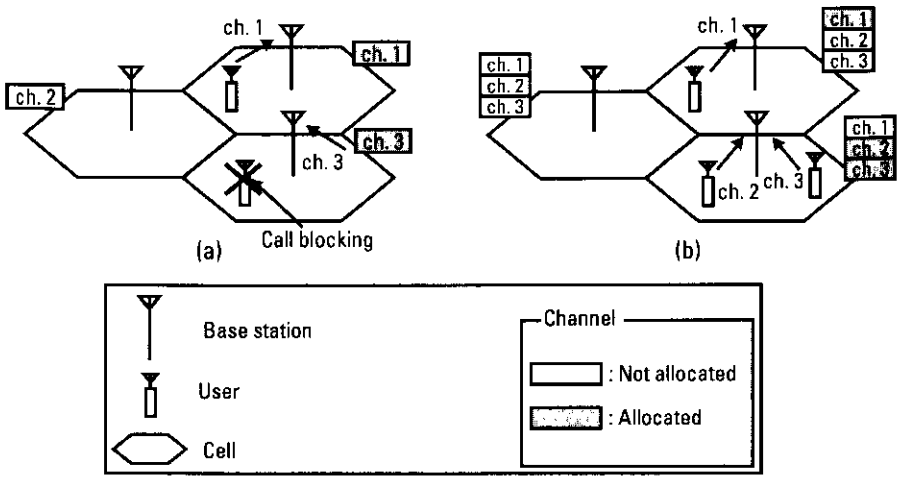

Figure 7.3 DCA algorithm concept: (a) FCA and (b) DCA.

Table 7.1
Functions of Programs 7.1–7.7

Program	Name in the Simulation	Function
Program 7.1	dcamain.m	Main program
Program 7.2	basest.m	Determine location of each base station
Program 7.3	wrap.m	Determine relationship of base station positions introducing cell-wrapping
Program 7.4	cellmesh.m	Distribute a cell into meshes and store the data about each mesh coordinate
Program 7.5	holdtime.m	Generates call holding time, which is exponentially distributed
Program 7.6	shadow.m	Generates an attenuation due to shadowing, which has log-normal distribution
Program 7.7	dist.m	Generates a path loss due to distance

two measures, blocking probability and forced termination probability. The blocking probability is defined as the statistical probability that a new call will fail to find suitable channels that satisfy the C/(N + I) ratio condition mentioned in Section 7.2. Although the blocking probability is the measure pertaining to new calls, a connected call can be interrupted before it finishes due to rapid degradation of the C/(N + I) ratio condition. Thus, we define forced termination probability as the statistical probability that a connected call will be interrupted before its conclusion. If we further define `callnum`, `blocknum`, and `forcenum` as the number of generated calls, blocked calls, and forced terminated calls, respectively, we truly introduce the variables of the same names in the simulation as shown in following sections. Blocking probability P_{bl} and forced termination probability P_{fo} are given as follows:

$$P_{bl} = \frac{blocknum}{callnum} \tag{7.2}$$

$$P_{fo} = \frac{forcenum}{callnum - blocknum} \tag{7.3}$$

Introducing those two performance measures enables several potential evaluations of cellular systems.

7.3.2 Flowchart of Entire Simulation

Figure 7.4 shows a flowchart of the entire simulation. Our simulation consists of three parts: the preparation part, the main loop part, and the output part. In the preparation part, several pieces of information needed for the simulation are introduced, such as the cell layout or traffic parameters, before the main loop is started.

The main loop of our simulation is activated by the `while` loop with preliminary finish time "`timeend`." Throughout the entire loop, the status of present users is checked and, if necessary, renewed in a short time interval, the

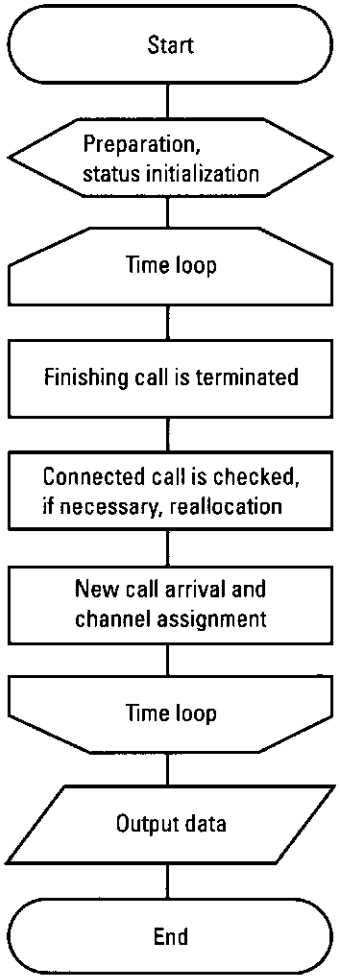

Figure 7.4 Simulation flowchart.

period "`timestep`." Several status indicators are successively stored in matrix "`userinfo`." Table 7.2 shows the contents of matrix "`userinfo`."

In every time period in the loop, each user causes several events, such as call initiation, channel searching, channel allocation, channel reallocation, and call termination based on the status matrix. In a time period, the following events are considered in turn.

1. Calls of connected users are terminated if they finish in this period.
2. Calls of still-connected users are examined. If a desirable interference condition is not satisfied, reallocation is attempted by searching for new channels.
3. With a preliminary probability, users that are not connected start new calls and search for channels that satisfy the interference conditions.
4. Return to (1).

Several types of numerical data are also measured in every period.

Finally, in the output part, measured and accumulated data in the main loop are organized into output, in the form of output matrix "`check`," or "`check2`," or "`output`," or text data "`data.txt`."

7.3.3 Cell Layout and Cell-Wrapping Technique Used

Figure 7.5 shows the cell layout employed in our simulation. We used 19 hexagonal cells having a cell radius of 1. Such cells are determined by the position of the 19 base stations. The base station positions are determined by Program 7.2, "`basest.m`," and this information is stored in the 19 × 2 matrix "`baseinfo`." For example, "`baseinfo(5, 1)`" and "`baseinfo(5, 2)`" respectively reveal x and y coordinates of the fifth base stations. In our simulation,

Table 7.2
Information Stored in "`userinfo`"

Column	Stored Information
userinfo(:, :, 1)	X-coordinates of the user
userinfo(:, :, 2)	Y-coordinates of the user
userinfo(:, :, 3)	Path loss of the user (if connected)
userinfo(:, :, 4)	Usage; 0:not connected, 1:connected
userinfo(:, :, 5)	Call termination time
userinfo(:, :, 6)	Allocated channel number (if connected)

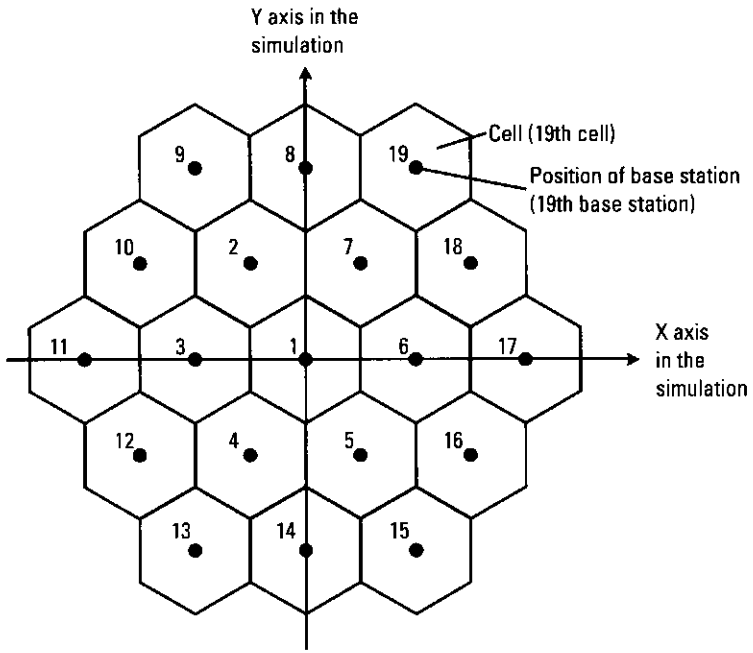

Figure 7.5 Cell layout.

regulated numbers of users are scattered in each of the 19 cells from which data are taken.

In the case of a cellular system using the DCA algorithm, we should take into account not only one sample cell, but also neighboring cells, because cochannel interference from neighboring cells has a significant effect on the performance of the sample cell. For example, the 5th cell is subject to interference from the 1st, 4th, 14th, 15th, 16th, and 6th cells. Cells located farther away, such as the 2nd or 3rd ones, could interfere with the 5th cell. However, it is assumed that such interference is decreased enough by the distance that it can be ignored, and we take into account only the six immediate neighboring cells in the simulation. On the other hand, in the case of the 9th cell, which is located on the boundary of the cell layout, it has only three neighboring cells, the 10th, 2nd, and 8th. Such a "boundary cell" has different performance than an "inner-located cell," for example the 9th cell, which would show better performance than the 2nd cell, because fewer cells cause interference in the 9th cell. Consequently, taking user activity in the boundary cells as well as that in an inner cell into account does not adequately evaluate DCA performance.

Thus, to avoid such a problem, two solutions can be used. One is to take data only from inner cells such as the 1st, 2nd, 3rd, 4th, 5th, 6th, and 7th cells

in Figure 7.5, and exclude boundary cells. These inner cells are all subject to interference from six neighboring cells and are expected to reveal effective performances of the DCA algorithm. However, because boundary cells do not contribute to output data, we need a larger number of cells to construct the entire cell layout to obtain well-averaged data, making the simulation burden heavy.

The other solution is to use a cell-wrapping technique. Figure 7.6 shows a concept of this technique. In this technique, boundary cells are regarded as neighbors of the boundary cells located almost directly opposite the cell layout. In Figure 7.6, only the 19 shaded cells are cells that really exist, and the other cells are copies of the real cells having the same number. As a result, the 9th cell suffers from interference not only from the 10th, 2nd, and 8th cells, but also copies of the 13th, 17th, and 14th cells in the neighboring positions. On this assumption, every cell in the cell layout can be regarded as being an "inner-located cell" having six neighbors.

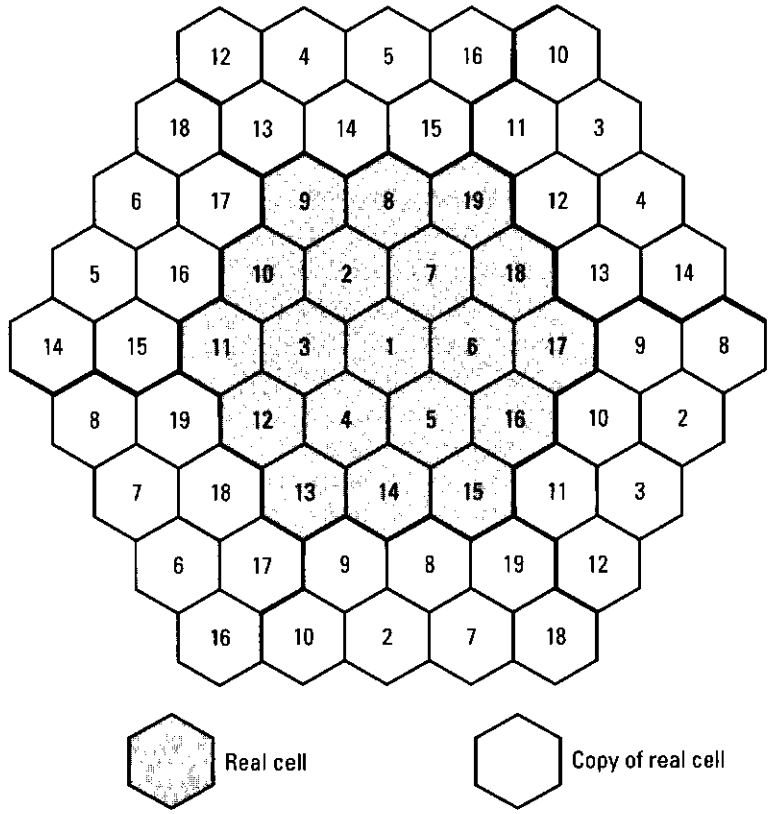

Figure 7.6 Cell-wrapping technique concept.

To realize cell wrapping, a 19 × 19 matrix `wrapinfo` is generated in the program `wrap.m`, and introduced in the main program. The `wrapinfo` matrix reveals the relationship among the 19 cells, including cell wrapping. Thus, for example, `wrapinfo(5, :)` stores information for fifth cell about other cells, especially `wrapinfo(5, 2)` to `wrapinfo(5, 7)`, which are the numbers of its six neighbors. (See Program 7.3 with Figure 7.6.) The concrete usage of this matrix will be explained later.

7.3.4 Cell Mesh Construction

In our simulation, user distribution is considered to be uniform over one cell as well as over the entire cell layout. Such a condition is realized by cells distributed into a lot of small meshes. Figure 7.7 shows a cell and meshes used in our simulation. Program `cellmesh.m` plays the role of a mesh generator in our simulation and outputs the number of meshes in a cell as a variable `cellnum` and outputs the `cellnum` × 2 matrix `meshposition`, which stores the position of each mesh. We can also see a result of `cellmesh.m` in the command line of MATLAB as follows:

```
>>[meshnum meshposition] = cellmesh;
>>plot (meshposition(:, 1),meshposition(:, 2),'.').
```

Figure 7.8 shows the results, where "`Fineness`" is a parameter for mesh fineness and is usually set to "`Fineness = 50`" in our simulation. Here, discrete

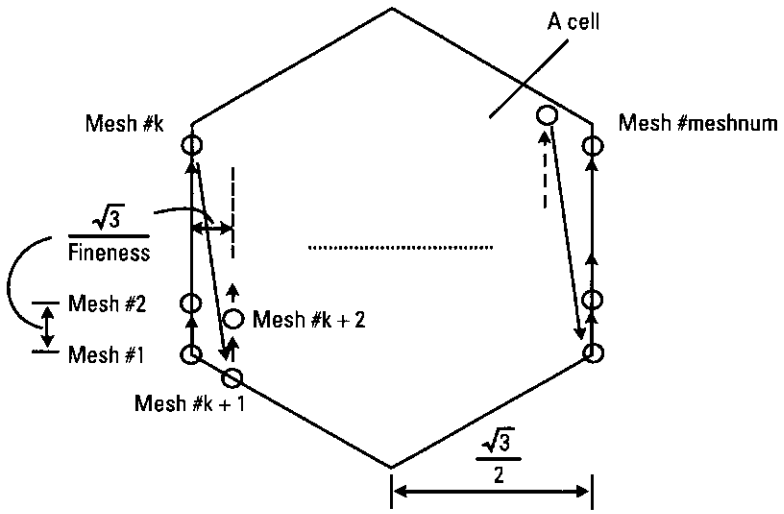

Figure 7.7 Meshes constructed in cell.

meshes are used to allocate a certain amount of traffic into the specially shaped area such as a hexagonal cell. If we introduce a larger value of "`Fineness`," such a discrete mesh structure gets close to the continuous distribution model, as expected in Figure 7.8.

In the simulation, when a user initiates a call, we generate a random integer "`mesh`" whose value uniformly fluctuates from 1 to "`meshnum`" and locates a user on a location of "`meshposition(mesh, :)`" in a cell. As a result, each user is scattered in a certain point of "`meshnum`" points in a cell with equal probability at its call initiation, and we can obtain uniform user distribution over a cell.

7.3.5 Traffic Parameters of a Call

Each call generation is subject to the Poisson process with its mean arrival rate for each user of "lambda" (calls/hour). To realize such an arrival rate, we examine each user that is not connected in every time period "`timestep`" using a random function "`rand`." If a value of "`rand`" is not more than "`lambda * timestep / 3600`," which presents an average arrival rate during "`timestep`," then the user is regarded as having started a call, and the number of generated calls is counted as a variable "`callnum`."

On the other hand, every initiated call has its own call holding time, and the initiated call is terminated after such a time. The holding time of each call is subject to exponential distribution with a mean value of "`ht`" (second). In our simulation, we obtain the value of the holding time as the output of the function "`holdtime(ht)`," which outputs a random value subject to exponential distribution with an average value "`ht`." The "`holdtime(ht)`" is provided by program "`holdtime.m`," and Figure 7.9 shows a *probability distribution function* (pdf) of the "`holdtime(ht)`" output, which is measured by MATLAB.

Moreover, the number of users is also a significant traffic parameter. We can set up the number of users existing in a cell and investigate the effects of such fluctuations of the user numbers. In the simulation results as shown later, we evaluate the performance of DCA mainly according to such numbers of users per cell, which is revealed by the variable "`user`."

7.3.6 Propagation Conditions

The strength of the received signal, regardless of the desired wave, or interference wave, is one of the most important issues in our simulation. The transmitted signal suffers from attenuation caused by such factors as distance and obstruction. In the simulation, we introduce path loss and shadowing as such attenuation factors.

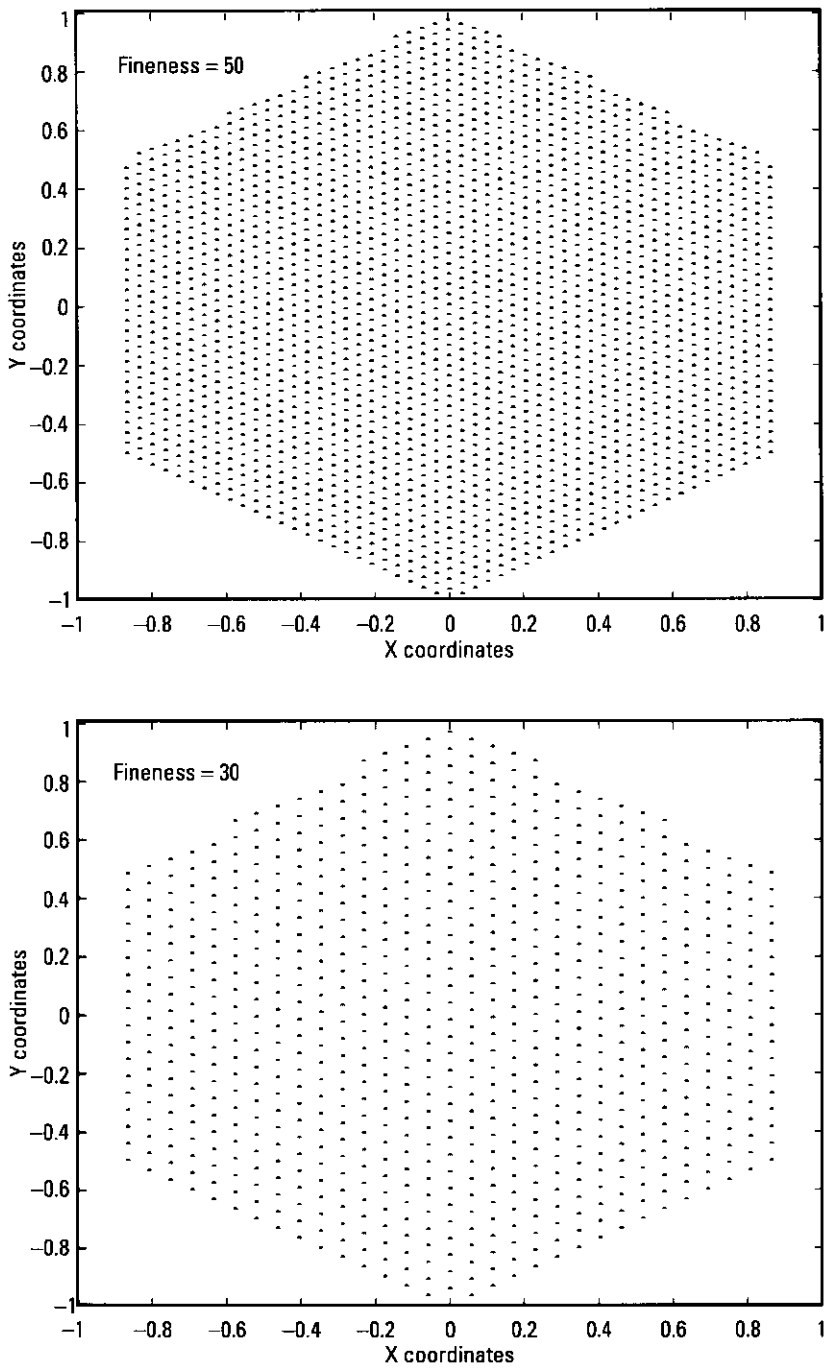

Figure 7.8 Mesh pattern examples.

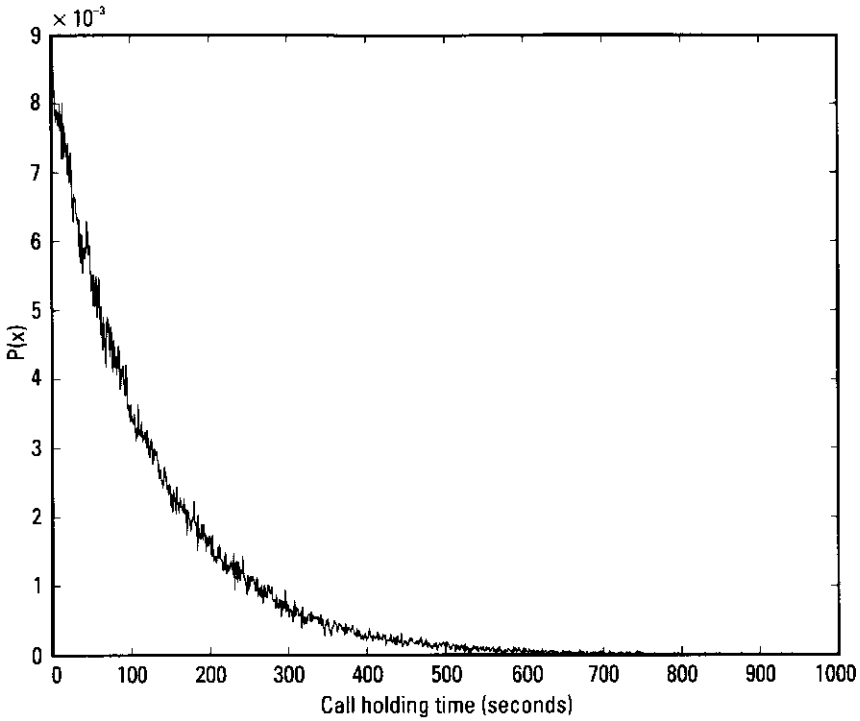

Figure 7.9 Call holding time pdf.

We assume the transmitted signal is subject to path loss with a decay factor of α as in (7.1), which is provided as variable "`alpha`" and set "`alpha = 3.5`" in the simulation. Calculating this path loss is conducted using the program "`dist.m`." This program provides a function "`dist(a, b, alpha),`" where "`a`" and "`b`" are a 1×2 matrix and respectively denote coordinates of two points. Then, value of "`dist(a, b, alpha)`" reveals the path loss between these two points, a and b. We describe a more practical usage of this function later.

Moreover, shadowing is assumed to be subject to log-normal distribution with a standard deviation of "`sigma,`" corresponding to ξ_i in (7.1). This "`sigma`" is set to "`sigma = 6.5`" in the simulation. We also obtain the value of shadowing as the output of the function "`shadow(sigma),`" which outputs a random value subject to log-normal distribution with its standard deviation "`sigma.`" The symbol "`shadow(sigma)`" is provided by program "`shadow.m,`" and Figure 7.10 shows a pdf of the "`shadow(sigma)`" output, which is measured by MATLAB.

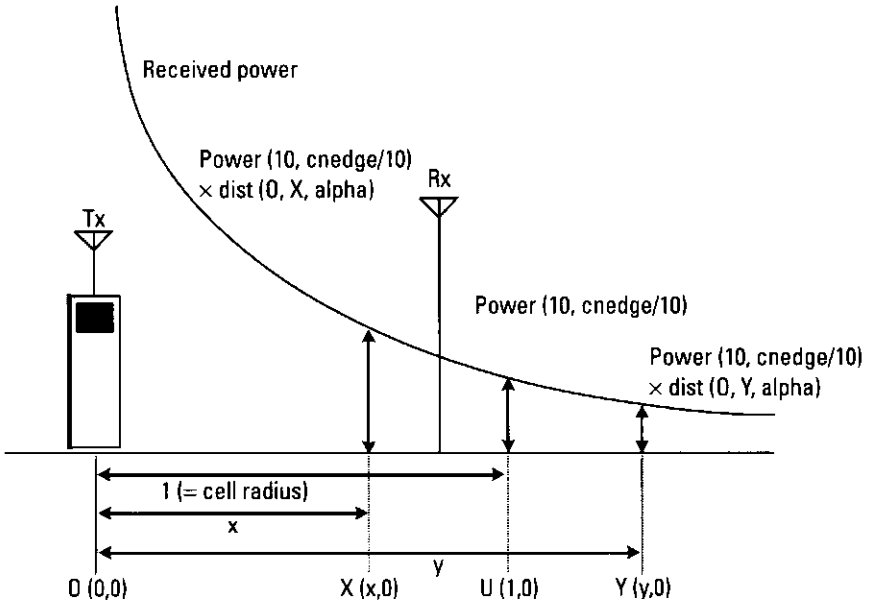

Figure 7.10 Path loss conditions.

7.3.7 Channel Assignment

The channel assignment process in the simulation is explained in this section. The routine is activated in the main program, "`dcamain.m`."

First of all, uplink power strength from a call-initiating user to the base station is examined. This power is provided by the C/N. We present this C/N using the simulation parameter "`cnedge`" (decibels), which reveals the C/N when the distance between the transmitter and receiver is 1. Figure 7.11 shows these conditions, where only path loss is considered.

In the program, we first input the coordinates of the user's position and the base station, obtain the path loss value as "`dist(here, there, alpha),`" then add shadowing attenuation generated by "`shadow(sigma)`" to that value. After that we obtain the attenuation value of the desired signal, which we call "`dwave`" in the program. Using the "`dwave`" value, we can obtain the C/N of the signal received by the base station. This C/N corresponds to "`power(10.0, cnedge/10.0) * dwave,`" and is represented by the "`cn`" variable.

After calculating "`cn,`" we search for an available channel that is not in use, and that satisfies the interference conditions of the C/(N + I). The channel search is conducted in the loop of channel number "`ch,`" so that the provided number ("`chnum`") of channels is examined in its entirety. If another user is allocated the same channel "`ch,`" we calculate the interference from that

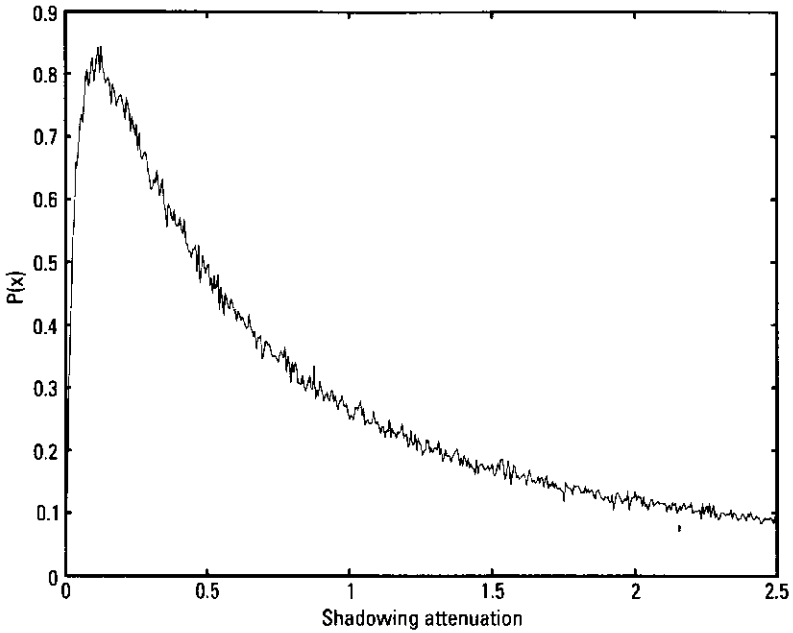

Figure 7.11 The pdf of shadowing attenuation.

user. In a loop of "`for around = 2:7,`" the variable "`othercell = wrapinfo(numcell, around)`" means the cell that is equivalent to the neighboring cell of the "`numcell`"th cell. We also calculate the attenuation value of the interference signal from this neighboring cell, the sum of which is called "`uwave,`" in the same manner as "`dwave.`" However, please note that one of the two coordinates matrix, in this case, the matrix "`there`" is provided somewhat differently, as follows:

```
>> userposi(1, 1:2) = userinfo(othercell, other, 1:2);
>> here = baseinfo(numcell, :);
>> there = userposi - baseinfo(othercell, :) ...
        + baseinfo(around, :) + baseinfo(numcell, :);
```

compared with "`dwave`" case below:

```
>> userposi(1, 1:2) = userinfo(numcell, numuser, 1:2);
>> here = baseinfo(numcell, :);
>> there = userposi;
```

In the case of "`uwave,`" there are three additive terms. These three terms play the role of compensating for a miscalculation that could be caused by cell wrapping.

Figure 7.12 shows the compensation by using two such terms. In the figure, interference from a user in the 15th cell to the 19th base station in 19th cell is evaluated. In this case the, the values are "`numcell = 19`," "`othercell = 15`," and "`around = 2`." (See `wrap.m`.) If we use the matrix "`there`" with the three terms as follows,

```
>> userposi(1, 1:2) = userinfo(15, other, 1:2);
>> here = baseinfo(19, :);
>> there = userposi - baseinfo(15, :) + baseinfo(2, :) ...
     + baseinfo(19, :);
```

we can obtain a suitable calculation of path loss "`dist(here, there, alpha)`" including the cell wrapping, shown as in Figure 7.12(a). However, if we ignore the three terms as follows,

```
>> userposi(1, 1:2) = userinfo(15, other, 1:2);
>> here = baseinfo(19, :);
>> there = userposi;
```

we obtain an incorrect value of path loss shown as in Figure 7.12(b). We calculate interference from every user that can cause interference and finally add them up as the total interference "`dwave`."

Now, as a goal, we have to obtain the C/(N + I) ratio R_{cni} which can be described as follows:

$$R_{cni} = \frac{C}{N+I} = \frac{1}{N/C + I/C} = \frac{1}{(C/N)^{-1} + (C/I)^{-1}} \tag{7.4}$$

From (7.2), we can see that the C/(N + I) is revealed by using the C/N and C/I. We have already provided the value of the C/N as "`power(10.0, cnedge/10.0) * dwave`," and the value of the C/I corresponds to "`dwave/ uwave`." By using such values, we can calculate the value of the C/(N + I) ratio "`cnirdb`" (decibels) in the simulation.

Finally, we examine if the achieved value of the C/(N + I) can satisfy the interference condition, that is, if it is greater than the C/(N + I) threshold "`cnirth`." If the achieved C/(N + I) exceeds the threshold, a call is accepted, and a channel is allocated to the user. In this case, the function "`holdtime()`" is activated and the output is also allocated to the user as its call holding time,

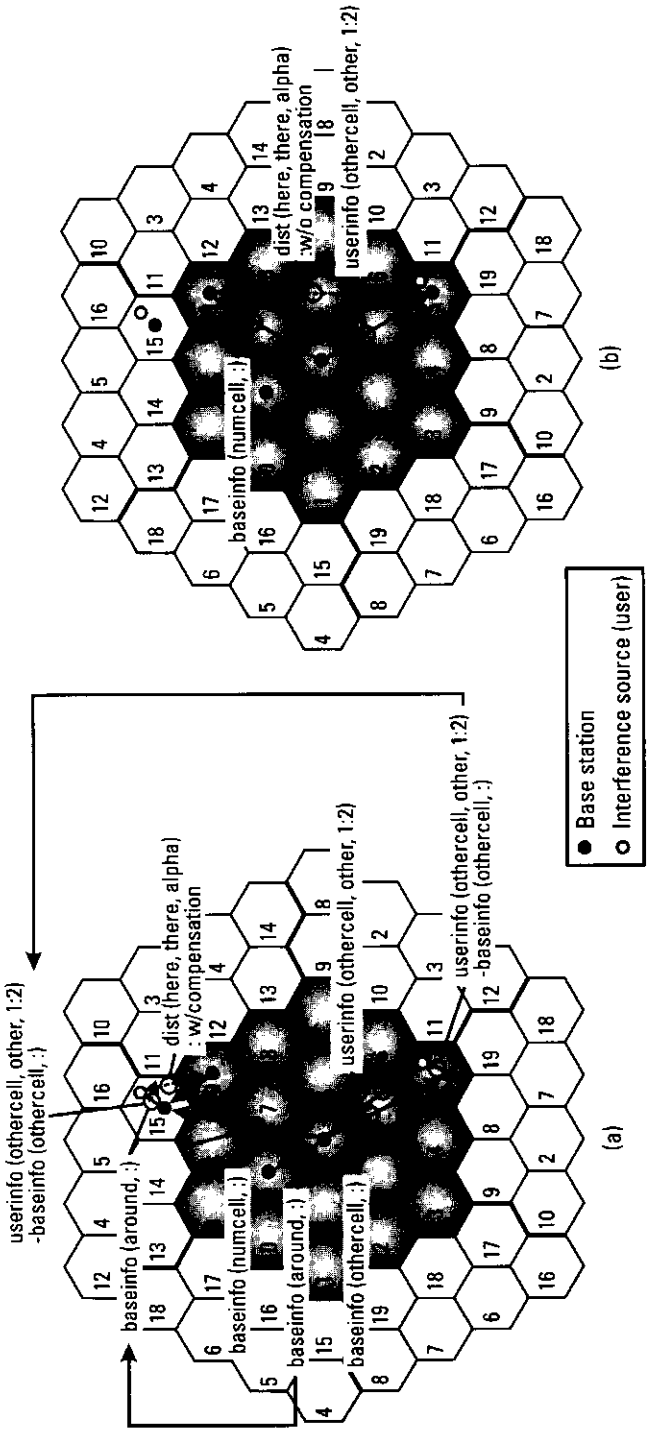

Figure 7.12 Compensation for cell-wrapping: (a) with compensation and (b) without compensation.

which provides the time the call is finished. If the interference condition is not satisfied, we regard such a call as being blocked, and the number of blocked calls is counted as the variable "blocknum."

7.3.8 Channel Check and Reallocation

We also check the interference conditions for connected users in every time period. This routine is conducted in a similar way to channel assignment, and the C/(N + I) of allocated channel for each connected user is examined. If the C/(N + I) of the channel is not satisfied, the user releases a currently allocated channel and tries to find an alternative channel in just the same way as channel assignment. At this point, if a channel condition is not satisfied, such an event is not regarded as blocking but as the forced termination of a call. Therefore, in that case, the number of forced terminations of calls is counted as the variable "forcenum."

7.3.9 Output

Finally, after the entire main loop part is executed, the accumulated data are calculated and stored in the output matrix "output." We have already counted the essential numerical values such as "callnum," "blocknum," and "forcenum," thereby simply calculating the previously defined blocking probability P_{bl} and forced termination probability P_{fo} as in the manner of (7.2) and (7.3).

Moreover, in Program 7.1, for convenience, we provide five simulations using different parameters for "usernum." Measured data in every "usernum" is all stored in matrix "output," which is also written as a text file "data.txt." Contents of the matrix "output" are shown in Table 7.3.

We also introduce other output matrices "check" and "check2" in the simulation. In every time period we store the current value of blocking probability and forced termination probability in "check" and "check2,"

Table 7.3
Contents of "output"

Column	Contents
output(1, :)	The number of calls
output(2, :)	The number of blocked calls
output(3, :)	Blocking probability
output(4, :)	Forced termination probability

respectively. Consequently, "check" and "check2" give transitional data in every time period, so we can determine suitable loop lengths for our simulation.

7.4 Simulation Results of Cellular System with DCA Algorithm

This section will evaluate the suitable performance of a cellular system using DCA.

7.4.1 Simulation Results

Using the same output as in previous sections, we can evaluate the results of our simulation. For example, we can see the blocking probability performance according to the number of users in the following manner:

```
>> semilogy (usernum, output(3, :))
```

Figure 7.13 shows the blocking probability versus the number of users. In the figures, "cnedge" [Figure 7.13(a)] and "chnum" [Figure 7.13(b)] are evaluated as parameters. We can see that a higher received power and a greater number of channels result in a lower blocking probability.

Figure 7.14 shows that the transitional blocking probability obtained by "check," which means the length of the simulation (we use "timeend = 5000"), is adequate for the evaluation.

7.4.2 Theoretical Analysis

This section evaluates the theoretical result for blocking probability. For simplicity, we make the assumption that the received power of each user is sufficiently high that the interference from other users can be ignored. By this assumption, we can denote the blocking probability by using Engset's loss formula [4] as follows:

$$P_{bl-theo} = \frac{\binom{n-1}{s}(vh)^s}{\sum_{i=0}^{s}\binom{n-1}{i}(vh)^i} \tag{7.5}$$

Here, n and s are the number of users and channels, respectively, and v and h are the average call arrival rates per nonconnected user and the average call holding time, which respectively correspond to "lambda" and "ht" in our simulation.

Figure 7.15 shows the comparison between the simulation results and theoretical value. We confirmed that our simulation results fit the theoretical value.

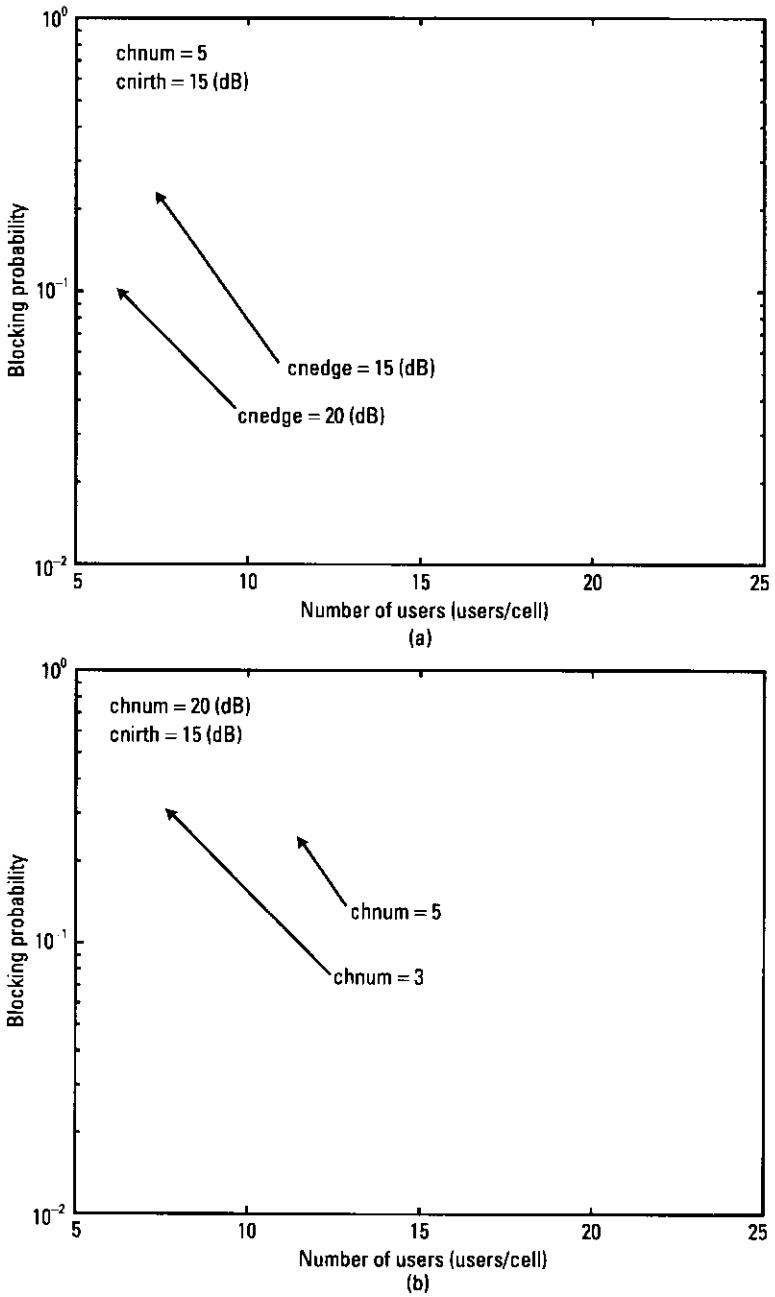

Figure 7.13 Blocking probability versus number of users: (a) parameter: cnedge, and (b) parameter: chnum.

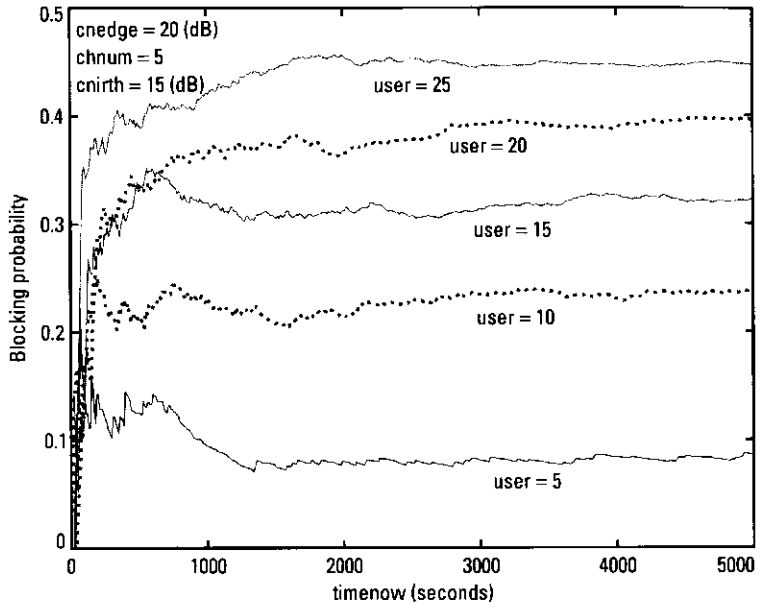

Figure 7.14 Transitional performances of blocking probability.

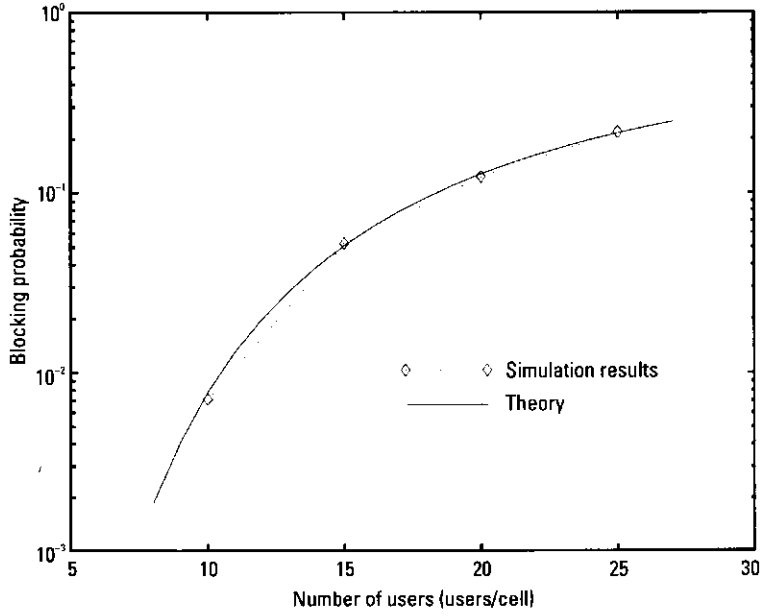

Figure 7.15 Theoretical analysis.

7.5 Beamforming Technique Using Array Antennas for Cellular Systems

The beamforming technique is widely used in mobile communication systems to optimize the use of frequency resources in the space domain [5]. By implementing this technique at the base station, the base station can change the shape of zones dynamically according to the locations of the mobile stations [6]. The cochannel interference and the reception of excess multipath fading from undesired directions are reduced at the base station, because it can issue high antenna gain in the desired direction and low antenna gain in the undesired one. Similarly, from the viewpoint of the mobile station, interference signals from other base stations are reduced. This is because base stations can avoid transmitting signals in undesired directions. As a result, the quality of service improves at both the base station and the desired mobile station. System capacity also increases because channel-reuse distances can be shortened. Naturally, using the beamforming technique at the mobile station is necessary for further improvement. The ratio of the desired signal (carrier) power to total cochannel interference signal power (CIR) are the key parameters that determine the capacity in mobile communication systems. Because the beamforming technique can greatly suppress interference, it is a powerful tool for improving system capacity.

Beamforming control provides several benefits to all conventional access schemes for wireless communication systems. These systems include FDMA, *time division multiple access* (TDMA), and CDMA. Moreover, *space division multiple access* (SDMA) has been proposed and studied as a scheme for increasing channel capacity within a limited frequency resource. In SDMA, a base station can allocate individual beams for each mobile station, and the mobile stations communicate with the base station by using the same frequency within a cell.

There are several measures for evaluating the effect of beamforming control, and these depend on the access scheme. This section focuses only on the CIR because it is a common parameter for all schemes. Here, we disregard thermal noise, because the CIR is far more influential in mobile communication systems.

For simplicity, this section explains the evaluation of the uplink improvement at the base station. Figure 7.16 shows a base station using the beamforming technique. In Figure 7.16, there are three cells using the same frequency, with the frequency reuse distance D. This distance is set to satisfy the required *quality of service* in the service model. Three base stations (B_0, B_1, and B_2) and three mobile stations (M_0, M_1, and M_2) are shown. Here, we assume that M_0, M_1, and M_2 transmit signals to B_0, B_1, and B_2, respectively, using the same

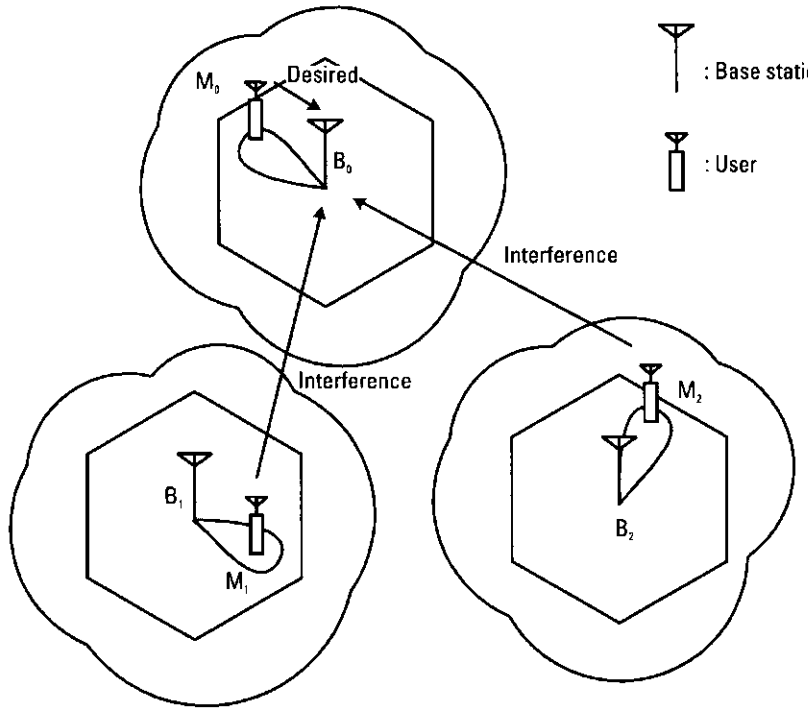

Figure 7.16 Use of beamforming technique at base stations in cellular systems.

frequency. When we notice the B_0, the ratio of the power of the desired signal from M_0 to the power of interference signals from M_1 and M_2 is given as follows:

$$R_{CI} = \frac{AP_0 d_0^{-\alpha} P 10^{\frac{\xi_0}{10}} \times 10^{\frac{G_{HB_0}(\theta_{HM_0})}{10}} \times 10^{\frac{G_{VB_0}(\theta_{VM_0})}{10}} \times 10^{\frac{G_{HM_0}(\theta_{HB_0})}{10}} \times 10^{\frac{G_{VM_0}(\theta_{VB_0})}{10}}}{\sum_{i=1}^{2} AP_i d_i^{-\alpha} 10^{\frac{\xi_i}{10}} \times 10^{\frac{G_{HB_0}(\theta_{HM_i})}{10}} \times 10^{\frac{G_{VB_0}(\theta_{VM_i})}{10}} \times 10^{\frac{G_{HM_i}(\theta_{HB_0})}{10}} \times 10^{\frac{G_{VM_i}(\theta_{VB_0})}{10}}} \tag{7.6}$$

where α is the path loss factor, A is a proportional coefficient, P_j is the transmitted power at M_j, d_j is the distance between M_j and B_0, and ξ_j is the distortion due to shadowing between M_j and B_0 ($j = 0, 1, 2$). The symbol $G_{HB_0}(\theta_{HM_j})$ represents the horizontal antenna gain of B_0 toward θ_{HM_j}, where θ_{HM_j} is the direction of M_j from B_0 in the horizontal direction. The symbol $G_{VB_0}(\theta_{VM_j})$ represents the vertical antenna gain of B_0 toward θ_{VM_j}, where θ_{VM_j} is the vertical direction between M_j and B_0. In the same way, $G_{HM_j}(\theta_{HB_0})$ represents the

horizontal antenna gain of M_j toward θ_{HB_0}, where θ_{HB_0} is the horizontal direction between B_0 and M_j, the symbol $G_{VM_j}(\theta_{VB_0})$ represents the vertical antenna gain of M_j toward θ_{VB_0}, where θ_{VB_0} is the vertical direction between B_0 and M_j. The relationship between θ_{HM_j} and θ_{VM_j} is shown in Figure 7.17. The terms θ_{HB_0} and θ_{VB_0} correspond to $(360\text{-}\theta_{HM_j})$ and $(90\text{-}\theta_{VM_j})$, respectively.

7.5.1 Simulation Programs for Evaluating CIR Improvement

This section uses a simulation program to evaluate CIR improvement in a mobile cellular system using the beamforming technique and an array antenna. The main program is shown as Program 7.8, and subprograms used in the main program are shown in Programs 7.9–7.11. Table 7.4 summarizes the function of each program. Figure 7.18(a) shows a simulation model. It is assumed that there are 19 cells using the same frequency. The distances between all base stations are

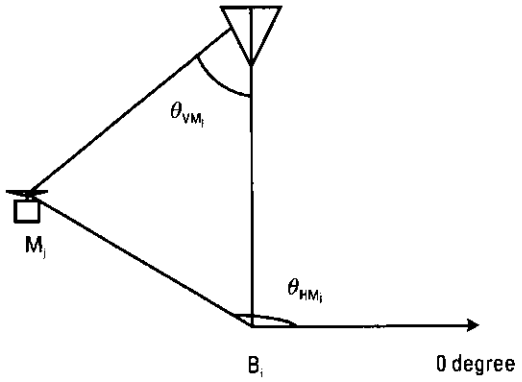

Figure 7.17 The relationship between a base station and a mobile station.

Table 7.4
Function of Programs 7.8–7.11

Program	Name in Simulation	Function
Program 7.8	main.m	Main program
Program 7.9	set_D.m	Determines distance between base stations
Program 7.10	stationinit.m	Determines position of base station
Program 7.11	antgain.m	Determines characteristics of the antenna gain
Program 7.6	shadow.m	Generates an attenuation due to shadowing that has log-normal distribution

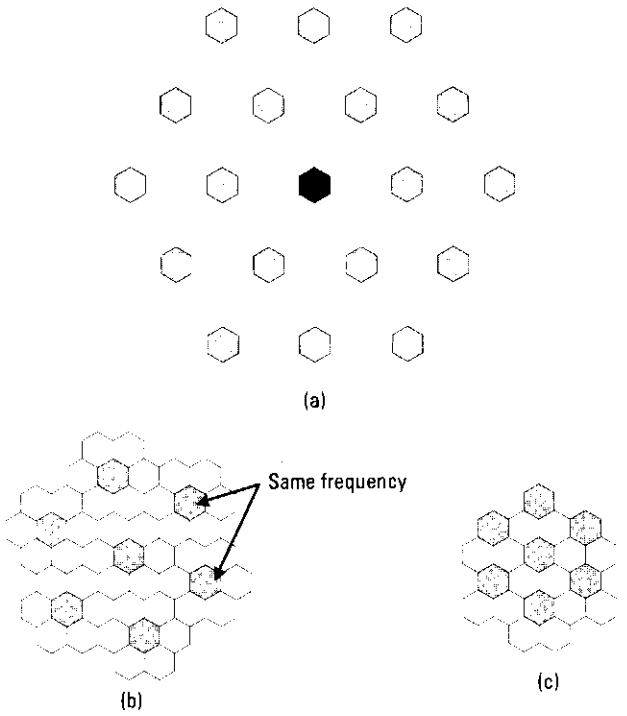

Figure 7.18 The arrangement of cells in simulation: (a) the arrangement of 19 cells using the same frequency, (b) the cell arrangement when the cluster size is 7, and (c) the cell arrangement when the cluster size is 3.

defined by the size of the clusters. Figure 7.18(b) and (c) shows the relationship between the cell arrangement and the size of the clusters. Here the centered cell is numbered 0 and the surrounding cells are numbered 1 through 18.

The program calculates the CIR for an uplink at a central base station and outputs the difference in the CIR statistics for two cases:

- Case 1: beamforming control is applied.
- Case 2: beamforming control is not applied.

The simulation process is as follows:

- Define the positions of the base stations.
- Define the position of the mobile stations in each cell i ($i = 0 \ldots 18$).

- Calculate the following:
  - The distance and angle between the mobile station in the ith cell and the target ith base station;
  - The distance and angle between the mobile station in the ith cell and the centered base station;
  - The antenna gain for the direction of the mobile station in the ith cell at the central base station;
  - The antenna gain for the direction of the centered base station at the mobile station in the ith cell;
  - The effect of shadowing.
- CIR calculation for cases 1 and 2.

Figure 7.19 shows a simulation flowchart. The following is an outline of the main program.

```
[Status initialization]
```

First, the cluster size is determined by setting parameters I and J. Here, cluster size is given by the following equation:

$$I^2 + J^2 + I \cdot J \tag{7.7}$$

To illustrate, when $I = 1$ and $J = 1$, the cluster size is 3. Next, based on the cluster size, we determine the arrangement of the base stations. We also decide on the height of the base station and the radius of the cell. If we set the height of the base station to 0, we can approximate the macrocell model.

Second, we determine the characteristics of the antenna gain at both the base station (BS) and the mobile station (MS). Here we define BS of the ith cell as B_i. We also define MS in the ith cell as M_i. In this simulation, we assume $G_{HB_i}(\phi) = G_{HM_i}(\phi) = G_{VB_i}(\phi) = G_{VM_i}(\phi) = 0$ $(i = 0 \ldots 18)$ for all direction ϕ in case 2. On the other hand, we define specific values for these antenna gain parameters in case 1. We assume that the characteristics of antenna gain can be approximated such that,

$$G_{HB_i}(\phi) = \begin{cases} 10\log_{10}\cos^n(\phi), & -\pi/2 \le \phi \le \pi/2 \\ x, & \text{otherwise} \end{cases} \tag{7.8}$$

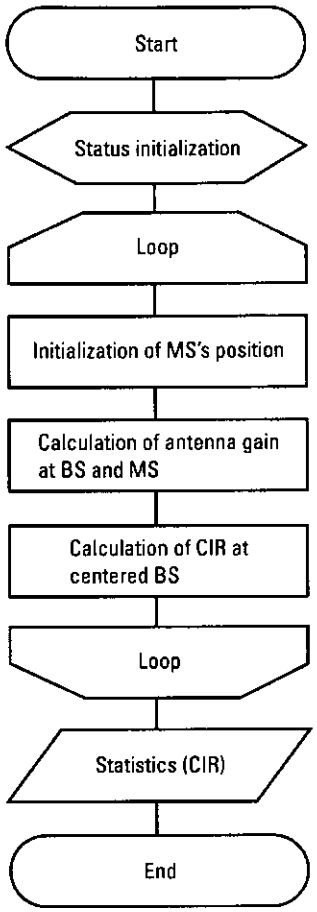

Figure 7.19 Simulation flowchart.

where x is the antenna gain for the back direction. The parameter n is used to determine the beamwidth θ. This width is within 3 dB of the peak. The value of n is determined by

$$n = \frac{\log_{10}\sqrt{1/2}}{\log_{10}\cos(\theta\pi/180)} \tag{7.9}$$

We can determine the characteristics of $G_{HB_i}(\phi)$, $G_{HM_i}(\phi)$, $G_{VB_i}(\phi)$, and $G_{VM_i}(\phi)$, using (7.8). Of course, we can determine them freely in order to evaluate the CIR improvement using another characteristic of antenna gain.

```
[Loop]
```

The positions of the mobile stations in each cell are determined randomly.

First, we calculate the distances between M_i and B_i to find the required transmission power P_i of M_i ($i = 0 \ldots 18$). By controlling P_i, we can implement a power control technique. This power control technique is also an effective way to improve CIR and has real relevance in beam control. Using this technique, each mobile station can control the transmission power dynamically in accordance with the distance between the mobile station itself and the desired base station.

Second, we calculate the distance between the central B_0 and M_i to find the path loss $d_i^{-\alpha}$.

Third, we calculate the horizontal and vertical angles θ_{HB_i} and θ_{VB_i} for the target B_i at M_i. In case 1 each M_i sets the peak antenna gain for the target B_i. Namely, since G_{HM_i} has a peak when $\phi = 0$ as shown in (7.8), $G_{HM_i}(0)$ is set at the direction θ_{HB_i} ($i = 0 \ldots 18$). In the same way, the peak direction of the vertical antenna gain, $G_{VM_i}(0)$, is set for direction θ_{VB_i}. Next, we calculate the horizontal and vertical angles θ_{HB_0} and θ_{VB_0} for the centered B_0 M_j ($j = 1 \ldots 18$). The difference between θ_{HB_i} and θ_{HB_0} and the difference between θ_{VB_i} and θ_{VB_0} are used to calculate the antenna gains $G_{HM_j}(\theta_{HB_0})$ and $G_{VM_j}(\theta_{VB_0})$, respectively.

In the same way, we calculate the horizontal and vertical angles θ_{HM_0} and θ_{VM_0} for M_0 from the centered B_0, and set the peak direction of the antenna gains $G_{HB_0}(0)$ and $G_{VB_0}(0)$ for θ_{HM_0} and θ_{VM_0}, respectively. Next, we calculate the horizontal and vertical angles θ_{HM_j} and θ_{VM_j} for M_j ($j = 1 \ldots 18$) from the centered B_0. By using the difference between θ_{HM_j} and θ_{HM_0} and the difference between θ_{VM_j} and θ_{VM_0}, we obtain the antenna gains $G_{HB_0}(\theta_{HM_j})$ and $G_{VB_0}(\theta_{VM_j})$.

Finally, based on (7.6), the CIR of the uplink in the centered base station for cases 1 and 2 are calculated.

```
[statistics (CIR)]
```

After repeating the `[Loop]`, the CIR statistics for cases 1 and 2 are calculated.

7.5.2 Simulation Results

Figure 7.20 shows the results for the CIR improvement of the uplink at the centered base station, highlighting the difference between cases 1 and 2. The parameters are as follows:

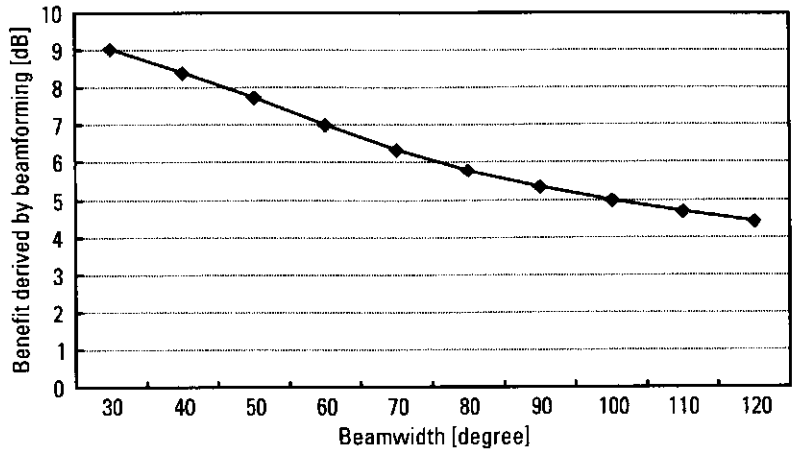

Figure 7.20 CIR improvement.

- $I = 1, J = 1$;
- $G_{HB_i}(\phi)$ is defined by (7.8) where $x = -100$ [dB], $n = 2.41$ ($\theta = 60$);
- $G_{VB_i}(\phi) = G_{HM_i}(\phi) = G_{VM_i}(\phi) = 0$ for all direction ϕ;
- Standard deviation of shadowing sigma = 0;
- Path loss factor alpha = 3.5;
- There is no power control.
- The height of the base station is 0[m] (the macrocell situation was assumed).

In Figure 7.20, the horizontal axis shows the beamwidth at the base station, and the vertical axis shows the difference in CIR between cases 1 and 2 at the center base station. The average CIR with beamforming control shows an improvement of more than 8 dB as compared to that without beamforming control.

Naturally, arranging the program to calculate the downlink improvement is easy. Note that the parameters between the base station and the mobile station are replaced. Note, as well, that we can arrange the program so that multiple users can use the same frequency in a cell. In this case, we must consider the interference calculation from intercell users. When we arrange the program, we must pay close attention to the application of the model and a target user or a base station that is focused in the CIR calculation.

7.6 Conclusions

This chapter focuses on an access scheme simulation to evaluate the total system performance considering multipoint situations. After explaining the concept of a cellular system and a channel assignment algorithm, we introduced an example of a simulation program for a cellular system having DCA, as well as several key techniques that are very useful for conducting these kinds of simulations. Moreover, we also introduced DCA with an array antenna and described the effectiveness of constructing dynamic zones.

References

[1] Balston, D. M., and R. C. V. Macario, *Cellular Radio Systems,* Norwood, MA: Artech House, 1993.

[2] Sampei, S., *Applications of Digital Wireless Technologies to Global Wireless Communications,* Upper Saddle River, NJ: Prentice Hall, 1997.

[3] Gelenbe, E., and G. Pujolle, *Introduction to Queueing Networks,* New York: John Wiley & Sons, 1987.

[4] Lee, W. C. Y., *Mobile Cellular Telecommunications Systems,* New York: McGraw-Hill, 1989.

[5] Godara, L. C., "Applications of Antenna Arrays to Mobile Communications, Part I: Performance Improvement, Feasibility, and System Considerations," *Proc. of IEEE,* Vol. 85, No. 8, August 1997, pp. 1195–1245.

[6] Litva, J., and T. K. Lo, *Digital Beamforming in Wireless Communications,* Norwood, MA: Artech House, 1996.

Appendix 7A

Program 7.1

```
% Program 7-1
%
% dcamain.m
%
% Simulation program to realize DCA algorithm
%
% Programmed by F. Kojima
%

%%%%%%%%%%%%%%%%%%% preparation part %%%%%%%%%%%%%%%%%%%

cnedge = 20.0;  % CNR on cell edge (dB)
cnirth = 15.0;  % CNIR threshold (dB)
lambda = 6.0;   % average call arrival rate (times/hour)
ht = 120.0;     % average call holding time (second)
timestep = 10;  % time step of condition check (second)
timeend = 5000; % time length of simulation (second)
chnum = 5;      % number of channels per each base station

alpha = 3.5;           % path loss factor
sigma = 6.5;           % standard deviation of shadowing

usernum = [5,10,15,20,25]; % number of users per cell

output = zeros(4,5); % output matrix

check = zeros(5,floor(timeend/timestep));
% matrix for transient status

for parameter = 1:5

  rand('state',5);
  randn('state',1);

  user = usernum(parameter); %number of users per cell

  baseinfo = zeros(19, 2);
  %baseinfo(cell #, informations)
  %%%%%baseinfo(:, 1): x coordinates
  %%%%%baseinfo(:, 2): y coordinates

  userinfo = zeros(19, user, 15);
  %userinfo(cell #, user #, informations)
  %%%%%userinfo(:, :, 1): x axis
  %%%%%userinfo(:, :, 2): y axis
  %%%%%userinfo(:, :, 3): attenuation
  %%%%%userinfo(:, :, 4): usage 0->non-connected
  %%%%%1->connected
```

```
%%%%%userinfo(:, :, 5): call termination time
%%%%%userinfo(:, :, 6): allocated channel #

[baseinfo] = basest;
[wrapinfo] = wrap;

[meshnum, meshposition] = cellmesh;

timenow = 0;
blocknum = 0;
forcenum = 0;
callnum = 0;
users = 0; % number of connected users

%%%%%%%%%%%%%%%%%% main loop part %%%%%%%%%%%%%%%%%%

while timenow timeend

    callnumold = callnum;
    blocknumold = blocknum;
    forcenumold = forcenum;

    %finished calls
    for numcell = 1:19
        for numuser = 1:user
            if userinfo(numcell, numuser, 4) == 1 &
                userinfo(numcell, numuser, 5) < timenow
                userinfo(numcell, numuser, 4) = 0;
                users = users -1;
            end
        end
    end

    %reallocation check
    for numcell = 1:19
        for numuser = 1:user
            if userinfo(numcell, numuser,4) == 1
                reallo = 0; % flag
                cnirdb1 = 0.0;
                dwave = userinfo(numcell, numuser, 3);
                cn = power(10.0, cnedge/10.0) * dwave;
                uwave = 0.0;
                ch = userinfo(numcell, numuser, 6);
                for around = 2:7
                    othercell = wrapinfo(numcell, around);
                    for other = 1:user
    if userinfo(othercell, other, 4) ...
        == 1 & userinfo(othercell,...
        other, 6) == ch
        userposi(1,1:2) = userinfo...
            (othercell, other, 1:2);
        here = baseinfo(numcell, :);
        there = userposi - baseinfo...
```



```
            (othercell, :) + baseinfo
            (around, :) + baseinfo
            (numcell, :);
    uwave = uwave + dist(here,
        there, alpha)*shadow
        (sigma);
        end
            end
        end % around loop
        if uwave == 0
            cnirdb = 10.0*log10(cn);
        else
            cnirdb = 10.0*log10(1/(uwave/
                dwave+1/cn));
        end
        if cnirdb < cnirth
            reallo = 1;
        end

        if reallo == 1
            userinfo(numcell, numuser, 4) = 0;
            users = users -1;
            succeed = 0;
            cnirdb = 0.0;
            for ch = 1:chnum
                available = 1;
                for other = 1:user
                    if userinfo(numcell, other, 4)
                        == 1 & userinfo(numcell,
                        other, 6) == ch
                    available = 0;
                end
            end
            if available == 1
                uwave = 0.0;
                for around = 2:7
                    othercell = wrapinfo(numcell,
                        around);
    for other = 1:user
    if userinfo(othercell,
        other, 4) == 1 &
        userinfo(othercell,
        other, 6) == ch
        userposi(1,1:2) =
        userinfo(othercell,
        other, 1:2);
        here = baseinfo
            (numcell, :);
        there = userposi -
            baseinfo(othercell,
            :) + baseinfo
            (around, :) + baseinfo
            (numcell, :);
```

```
                uwave = uwave + dist...
                    (here, there, alpha)...
                    *shadow(sigma);
                end
                    end
                        end % around loop
                        if uwave == 0
                            cnirdb = 10.0*log10(cn);
                        else
                            cnirdb = 10.0*log10(1/(uwave/...
                                dwave+1/cn));
                        end
                    else
                        cnirdb = 0.0;
                    end
                    if cnirdb >= cnirth
                        succeed = 1;
                        users = users + 1;
                        userinfo(numcell, numuser, 4) = 1;
                        userinfo(numcell, numuser, 6) = ch;
                        break
                    end
                end % ch loop
                if succeed == 0
                    forcenum = forcenum + 1;
                end
            end % reallo == 1
        end % connected (need to be checked)
    end % user loop
end % cell loop

%new call arrival
for numcell = 1:19
    for numuser = 1:user
        if userinfo(numcell, numuser, 4) == 0 & rand <=...
            lambda*timestep/3600
            callnum = callnum + 1;
            mesh = floor(meshnum.*rand) +1;
            while mesh > meshnum
                mesh = floor(meshnum.*rand) +1;
            end
            userinfo(numcell, numuser, 1:2) = baseinfo...
                (numcell, :) + meshposition(mesh, :);
            succeed = 0; % flag
            cnirdb = 0.0;
            userposi(1,1:2) = userinfo(numcell,...
                numuser, 1:2);
            here = baseinfo(numcell,:);
            there = userposi;
            dwave = dist(here, there, alpha) * shadow...
                (sigma);
            cn = power(10.0, cnedge/10.0) * dwave;
            for ch = 1:chnum
```



```
                available = 1;
                for other = 1:user
                    if userinfo(numcell, other, 4) == 1 &
                        userinfo(numcell, other, 6) == ch
                        available = 0;
                    end
                end
                if available == 1

                    uwave = 0.0;
                    for around = 2:7
                        othercell = wrapinfo(numcell,...
                            around);
for other = 1:user
    if userinfo(othercell, other,....
        4) == 1 & userinfo
        (othercell, other, 6) == ch...
        userposi(1,1:2) = userinfo...
            (othercell, other, 1:2);
        here = baseinfo(numcell, :);
        there = userposi - baseinfo...
            (othercell, :) + baseinfo...
        (around, :) + baseinfo
            (numcell, :);
        uwave = uwave + dist(here,...
            there, alpha)*shadow...
            (sigma);
        end
    end
                        end % around loop
                        if uwave == 0
                            cnirdb = 10.0*log10(cn);
                        else
                            cnirdb = 10.0*log10(1/(uwave/...
                                dwave+1/cn));
                        end
                    else
                        cnirdb = 0.0;
                    end
                    if cnirdb >= cnirth
                        succeed = 1;
                        users = users + 1;
                        userinfo(numcell, numuser, 3) = dwave;
                        userinfo(numcell, numuser, 4) = 1;
                        userinfo(numcell, numuser, 5) =...
                            timenow + holdtime(ht);
                        userinfo(numcell, numuser, 6) = ch;
                        break
                    end
                end % ch loop
                if succeed == 0
                    blocknum = blocknum + 1;
                end
```

```
        end % new call
      end % user loop
    end % cell loop


    fprintf('%d\t%d\t%d\t%d\t%e\n',parameter,timenow,callnum...
        -callnumold,blocknum-blocknumold,blocknum/callnum);
        check(parameter,timenow/timestep+1) = blocknum/callnum;
        check2(parameter,timenow/timestep+1) = forcenum/...
            (callnum-blocknum);

        timenow = timenow + timestep;
    end %while loop

    %%%%%%%%%%%%%%%%%%% output part %%%%%%%%%%%%%%%%%

    output(1,parameter) = callnum;
    output(2,parameter) = blocknum;
    output(3,parameter) = blocknum/callnum;
    output(4,parameter) = forcenum/(callnum-blocknum);
end %parameter loop

fid = fopen('data.txt','w');
fprintf(fid,'UserNumber\t');
fprintf(fid,'%g\t%g\t%g\n', usernum(1,:));
fprintf(fid,'CallNumber\t');
fprintf(fid,'%g\t%g\t%g\n', output(1,:));
fprintf(fid,'BlockNumber\t');
fprintf(fid,'%g\t%g\t%g\n', output(2,:));
fprintf(fid,'BlockingProb. \t');
fprintf(fid,'%g\t%g\t%g\n', output(3,:));
fprintf(fid,'ForcedTerminationProb. \t');
fprintf(fid,'%g\t%g\t%g\n', output(4,:));
fclose(fid);

%********************* end of file ********************
```

Program 7.2

```
% Program 7-2
%
% basest.m
%
% This function sets positions of the base stations.
%
% Programmed by F. Kojima
% Checked by H. Harada
%

function [out] =  basest()
```



```
baseinfo(1, 1) = 0.0;
baseinfo(1, 2) = 0.0;
baseinfo(2, 1) = -0.5*sqrt(3.0);
baseinfo(2, 2) = 1.5;
baseinfo(3, 1) = -sqrt(3.0);
baseinfo(3, 2) = 0.0;
baseinfo(4, 1) = -0.5*sqrt(3.0);
baseinfo(4, 2) = -1.5;
baseinfo(5, 1) = 0.5*sqrt(3.0);
baseinfo(5, 2) = -1.5;
baseinfo(6, 1) = sqrt(3.0);
baseinfo(6, 2) = 0.0;
baseinfo(7, 1) = 0.5*sqrt(3.0);
baseinfo(7, 2) = 1.5;
baseinfo(8, 1) = 0.0;
baseinfo(8, 2) = 3.0;
baseinfo(9, 1) = -sqrt(3.0);
baseinfo(9, 2) = 3.0;
baseinfo(10, 1) = -1.5*sqrt(3.0);
baseinfo(10, 2) = 1.5;
baseinfo(11, 1) = -2.0*sqrt(3.0);
baseinfo(11, 2) = 0.0;
baseinfo(12, 1) = -1.5*sqrt(3.0);
baseinfo(12, 2) = -1.5;
baseinfo(13, 1) = -sqrt(3.0);
baseinfo(13, 2) = -3.0;
baseinfo(14, 1) = 0.0;
baseinfo(14, 2) = -3.0;
baseinfo(15, 1) = sqrt(3.0);
baseinfo(15, 2) = -3.0;
baseinfo(16, 1) = 1.5*sqrt(3.0);
baseinfo(16, 2) = -1.5;
baseinfo(17, 1) = 2.0*sqrt(3.0);
baseinfo(17, 2) = 0.0;
baseinfo(18, 1) = 1.5*sqrt(3.0);
baseinfo(18, 2) = 1.5;
baseinfo(19, 1) = sqrt(3.0);
baseinfo(19, 2) = 3.0;
out = baseinfo;

%********************* end of file *********************
```

Program 7.3

```
% Program 7-3
%
% wrap.m
%
% This function gives informations about cell-wrapping
%
% Programmed by F. Kojima
```

```
% Checked by H. Harada
%

function [out] =  wrap(inputmat)

inputmat( 1,1:19) = 1:19;
inputmat( 2,1:19) = [ 2, 9,10, 3, 1, 7, 8,14,13,17,16,...
  11,12, 4, 5, 6,18,19,15];
inputmat( 3,1:19) = [ 3,10,11,12, 4, 1, 2, 9,17,16,15,...
  19,18,13,14, 5, 6, 7, 8];
inputmat( 4,1:19) = [ 4, 3,12,13,14, 5, 1, 2,10,11,19,...
  18,17, 9, 8,15,16, 6, 7];
inputmat( 5,1:19) = [ 5, 1, 4,14,15,16, 6, 7, 2, 3,12,...
  13, 9, 8,19,11,10,17,18];
inputmat( 6,1:19) = [ 6, 7, 1, 5,16,17,18,19, 8, 2, 3,...
  4,14,15,11,10, 9,13,12];
inputmat( 7,1:19) = [ 7, 8, 2, 1, 6,18,19,15,14, 9,10,...
  3, 4, 5,16,17,13,12,11];
inputmat( 8,1:19) = [ 8,14, 9, 2, 7,19,15, 5, 4,13,17,...
  10, 3, 1, 6,18,12,11,16];
inputmat( 9,1:19) = [ 9,13,17,10, 2, 8,14, 4,12,18, 6,...
  16,11, 3, 1, 7,19,15, 5];
inputmat(10,1:19) = [10,17,16,11, 3, 2, 9,13,18, 6, 5,...
  15,19,12, 4, 1, 7, 8,14];
inputmat(11,1:19) = [11,16,15,19,12, 3,10,17, 6, 5,14,...
  8, 7,18,13, 4, 1, 2, 9];
inputmat(12,1:19) = [12,11,19,18,13, 4, 3,10,16,15, 8,...
  7, 6,17, 9,14, 5, 1, 2];
inputmat(13,1:19) = [13,12,18,17, 9,14, 4, 3,11,19, 7,...
  6,16,10, 2, 8,15, 5, 1];
inputmat(14,1:19) = [14, 4,13, 9, 8,15, 5, 1, 3,12,18,...
  17,10, 2, 7,19,11,16, 6];
inputmat(15,1:19) = [15, 5,14, 8,19,11,16, 6, 1, 4,13,...
  9, 2, 7,18,12, 3,10,17];
inputmat(16,1:19) = [16, 6, 5,15,11,10,17,18, 7, 1, 4,...
  14, 8,19,12, 3, 2, 9,13];
inputmat(17,1:19) = [17,18, 6,16,10, 9,13,12,19, 7, 1,...
  5,15,11, 3, 2, 8,14, 4];
inputmat(18,1:19) = [18,19, 7, 6,17,13,12,11,15, 8, 2,...
  1, 5,16,10, 9,14, 4, 3];
inputmat(19,1:19) = [19,15, 8, 7,18,12,11,16, 5,14, 9,...
  2, 1, 6,17,13, 4, 3,10];

out = inputmat;

%********************* end of file **********************
```

Program 7.4

```
% Program 7-4
%
% cellmesh.m
```



```
%
% This function sets meshes in a cell.
%
% Programmed by F. Kojima
% Checked by H. Harada
%

function [meshnum, meshposition] =  cellmesh()

Fineness = 50; % parameter for mesh fineness
k = 1;
j = 0;
ds = sqrt(3.0) / Fineness;
xmesh = -0.5 * sqrt(3.0);
while xmesh 0.5 * sqrt(3.0)
  xmesh = (k - 1) * ds - 0.5 * sqrt(3.0);
  if xmesh 0.0
     ymin = -xmesh/sqrt(3.0) - ds - 1.0;
        % lower line of a cell
     ymax = xmesh/sqrt(3.0) + 1.0;
        % upper line of a cell
  elseif xmesh == 0.0

     ymin = -1.0;
     ymax = 1.0;
  else
     ymin = xmesh/sqrt(3.0) - ds -1.0;
        % lower line of a cell
     ymax = -xmesh/sqrt(3.0) + 1.0;
        % upper line of a cell
  end
  k= k + 1;
  ymesh = ymin;
  while ymesh ymax
     ymesh = ymesh + ds;
     if ymesh  ymax
        break
     end
     j = j + 1;
     posi(j,1) = xmesh;
     posi(j,2) = ymesh;
  end
end
meshnum = j;
meshposition = posi;

%********************* end of file **********************
```

Program 7.5

```
% Program 7-5
%
```

```
% holdtime.m
%
% This function generates holding time
%
% Programmed by F. Kojima
% Checked by H. Harada
%

function [x] =  holdtime(ht)

para = rand;
while para >= 1
  para = rand;
end
x = ht.*(-log(1-para));

%********************* end of file **********************
```

Program 7.6

```
% Program 7-6
%
% shadow.m
%
% This function generates attenuation of shadowing
%
% Programmed by F. Kojima
% Checked by H. Harada
%

function [x] =  shadow(sigma)

anoz = randn;
db = sigma * anoz;
x = power(10.0, 0.1*db);

%********************* end of file **********************
```

Program 7.7

```
% Program 7-7
%
% dist.m
%
% This function generates attenuation due to distance
%
% Programmed by F. Kojima
% Checked by H. Harada
%

function [x] =  dist(a,b,alpha)
```



```
% a,b: position of two points
% size(a) = size(b) = (1,2)

x = sqrt((a-b)*(a-b)');
x = power(x, -1.*alpha);

%********************* end of file **********************
```

Program 7.8

```
% Program 7-8
%
% main.m
%
% Programmed by A. Kanazawa
% Checked by H. Harada
%

clear
%%%%%%%%%%%%%%%% Status initialization
I = 1;       % The cluster size is determined from I and
             % J. (n = I*I + J*J + I*J)
J = 2;
r = 100;     % the radius of the cell[m]
h = 0;       % the height of the BS[m]

D = set_D(I,J,r);
station = stationInit(D);
xbs = real(station);   % The x axis of the BS
ybs = imag(station);   % The y axis of the BS

sigma = 6.5;        % standard deviation of shadowing

alpha = 3.5;        % path loss factor
% margin = 0;       % The parameter for power control

% Characteristics of antenna gain decision for BS
w_HBS = 60;      % [horizontal]: beam width at BS for the
                 % target direction [degree]
backg_BS = -100;% [horizontal]: antenna gain at BS for the
                 % opposite direction [dB]
w_VBS = 360;     % [vertical]: beam width at BS [degree]

% Characteristics of antenna gain decision for MS
w_HMS = 360;     % [horizontal]: beam width at MS for the
                 % target direction [degree]
backg_MS = -100;% [horizontal]: antenna gain at
                 % MS for the opposite direction [dB]
w_VMS = 360;     % [vertical]:beam width at MS [degree]

if h == 0,        % In the case of macro cell situation,
  w_VBS = 360; w_VMS = 360;    % the effect of beam tilt
```

```
                                % becomes less.
end

% Antenna gain calculation of each BS
g_HBS = antgain(w_HBS, backg_BS);
g_VBS = antgain(w_VBS, 0);
g_HMS = antgain(w_HMS, backg_MS);
g_VMS = antgain(w_VMS, 0);

%%%%%%%%%%%%%% Loop
%-----Initialization of MS positions
N=1000;                 % The number of repeat
for num = 1:N,
  Rx = rand(1,19);      % the random values: [0-1]
  Ry = rand(1,19);      % the random values: [0-1]
  X = r*Rx;
  Y = Ry.* sqrt ( r ^2 - X.^2 );

  tx = 2*((rand(1,19)0.5) -0.5);    % the random values:
                                    % -1 or 1
  ty = 2*((rand(1,19)0.5) -0.5);    % the random values:
                                    % -1 or 1
  x = X.* tx;       % The x axis of the MS when we regard
                    % the position of each BS as (0,0)
  y = Y.* ty;       % The y axis of the MS when we regard
                    % the position of each BS as (0,0)

  x2 = x+xbs.';     % The x axis of the MS when we regard
                    % the position of central BS as (0,0)
  y2 = y+ybs.';     % The y axis of the MS when we regard
    the position of central BS as (0,0)

% The complex expression of MS when we regard the position
% of each BS as (0,0)
  z(1,:) = x + i * y;
% The complex expression of MS when we regard the position
% of central BS as (0,0)
  z(2,:) = x2+ i * y2;

  d(1,:) = abs(z(1,:));   % The distance between BS_i
                          % and MS_i in horizontal axis
  d(2,:) = abs(z(2,:));   % The distance between central
                          % BS and MS_i in horizontal axis
  d2 = sqrt(d.^2 + h^2);  % The distance

  phai(1,:) = angle(z(1,:)); % The angle difference
                             % between BS_i and MS_i [rad]
  phai(2,:) = angle(z(2,:)); % The angle difference
                             % between central BS and
                             % MS_i [rad]
  deg = phai*180/pi;         % the conversion of radian
                             % to degree
```



```
  if h ==0, degH = 90*ones(1,19);
     else
        % the elevation angle between central BS and MS_i
        phaiH = atan(d(2,:)/h);
        % the conversion of radian to degree
        degH = phaiH*180/pi;
     end

%-------- shadowing --------
  for m = 1:19
     g(m) = 10*log10(shadow(sigma));
  end

%-------- propagation loss --------
  Loss(1,:) = 10 * log10(d2(1,:).^alpha);
     % The propagation loss from MS_i to BS_i [dB]
  Loss(2,:) = 10 * log10(d2(2,:).^alpha);
     % The propagation loss from MS_i to BS_0 [dB]
  Loss_max = 10 * log10(r.^alpha);
     % The propagation loss from the cell boundary to
     % BS [dB]

% Free space loss
% wl = 0.1;
% Loss(1,:) = 10 * log10((4*pi*d2(1,:)/wl).^2);
  % The propagation loss from MS_i to BS_i [dB]
% Loss(2,:) = 10 * log10((4*pi*d2(2,:)/wl).^2);
  % The propagation loss from MS_i to BS_0 [dB]
% Loss_max = 10 * log10((4*pi*r/wl).^2);
  % The propagation loss from the cell boundary to BS [dB]

%----- Transmission power level of each MS [dB] -----

  Ptm_0= Loss_max*ones(1,19);       % no power control
% Ptm_0= Loss(1,:) + margin;        % power control
                                    % (with margin [dB])

%--Calculation of antenna gain for the target direction

% the angle difference between the MS_0 and MS_i from
% central BS
  deg_B = deg(2,1)-deg(2,:);
% the angle difference between the BS_0 and BS_i from MS_i
  deg_M = deg(1,:)-deg(2,:);

  degHBS = mod(round(deg_B),360);
  degHMS = mod(round(deg_M),360);
  degVBS = round(degH-degH(1)); % the angle difference
     % in vertical direction between MSs and central BS
  degVMS = degVBS;     % the angle difference in
     % vertical direction between MSs and central BS
```

```
    %----- Calculation of CIR at centered BS -----
    %Control
    CIdB_a= Ptm_0(1:19)+g_HBS(degHBS(1:19)+1) + g_VBS...
        (degVBS(1:19)+1) + g_HMS(degHMS(1:19)+1) + g_VMS...
        (degVMS(1:19)+1) - Loss(2,1:19)-g(1:19);
       % Received level at central BS (beam forming)
    CIw_a = 10 .^ ( CIdB_a ./ 10 );       % dB → W
    isum_a = sum( CIw_a(2:19));
    CIR_a(num) =   CIw_a(1) / isum_a;
    %No Control
    CIdB_o= Ptm_0(1:19) - Loss(2,1:19)-g(1:19);
       % Received level at centered BS (Omni)
    CIw_o= 10 .^ ( CIdB_o ./ 10 );        % dB → W
    isum_o = sum( CIw_o(2:19));
    CIR_o(num) =  CIw_o(1) / isum_o;

    %----- Calculation of CIR under various w_HBS
    %ii = 1;
    %for w_HBS2=30:10:180,
    %   g_HBS2 = antgain(w_HBS2, backg_BS);
    %   CIdB_a2= Ptm_0(1:19)+g_HBS2(degHBS(1:19)+1) + ...
    %   g_VBS(degVBS(1:19)+1) + g_HMS(degHMS(1:19)+1) + ...
    %   g_VMS(degVMS(1:19)+1) - Loss(2,1:19)-g(1:19);
        % Received level at central BS (beam)
    %   CIw_a2 = 10 .^ ( CIdB_a2 ./ 10 );     % dB → W
    %   ciw_a2 = sum( CIw_a2(2:19));
    %   CIR_a2(num,ii) =   CIw_a2(1) / ciw_a2;
    %   ii = ii+1;
    %end

end
%---- statistics

CA = 10 * log10(sum(CIR_a)/N);
CO = 10 * log10(sum(CIR_o)/N);

%---- result
CA-CO                                % Improvement

%---- Calculation of CIR under various w_HBS
% CA2= 10 * log10(sum(CIR_a2)/N);
% CA2-CO
% plot(30:10:/20,CA2-CO)

%********************* End of file **********************
```

Program 7.9

```
% Program 7-9
%
% set_D.m
%
% Programmed by A. Kanazawa
% Checked by H. Harada
%

function D = set_D(x,y,R)

% The funstion to determine the distance between BSs
i_R = sqrt(3) * R;
j_R = i_R / sqrt(2);

D2 = 0+0j;
for k = 1:x
  D2 =  D2 + i_R;
end

for k = 1:y
  D2 = D2 + j_R + j * j_R;
end

D = abs(D2);

%********************** end of file *********************
```


Program 7.10

```
% Program 7-10
%
% stationInit.m
%
% Programmed by A. Kanazawa
% Checked by H. Harada
%

function STATION = stationInit(d)

% The function to determine the position of BS
STATION = zeros(19,1);

for i = 0:5
  STATION(i + 2, 1) = d * cos(i * pi / 3.0) + j * d...
     * sin(i * pi / 3.0);
end

for i = 0:11
  STATION(i + 8, 1) = 2 * d * cos(i * pi / 6.0) + j * 2...
     * d * sin(i * pi / 6.0);
end

%********************** end of file*********************
```

Program 7.11

```
% Program 7-11
%
% antgain.m
%
% Programmed by A. Kanazawa
% Checked by H. Harada
%

function gain = antgain(width, back)

theta = [0:359];
gain = zeros(1,360);

if width ~= 360,

  n = log10(sqrt(1/2))./log10(cos(width*pi/360));
     % the coefficient n
  gain(1:89) = 10*log10(cos(theta(1:89)*pi/180).^n);
     % [dB] Antenna gain
  gain(272:360) = 10*log10(cos(theta(272:360)*pi/180).^n);
     % [dB] Antenna gain

  if back ~= 0,     % definition of antenna gain for
                    % horizontal direction[dB]
     gain(90:271) = back;              %[dB]
  end
end

%********************** end of file**********************
```

8

Software Radio Communication Systems

8.1 Introduction

As the number of systems that users can employ increased, there have been increasing demands for the coexistence of several mobile telecommunication services (e.g., GSM, IS-54, IS-136, IS-95, Japanese PDC or PHS, or the IMT-2000 system). This is because there are a lot of air interfaces in the world, and most mobile phone users disapprove of carrying more than one mobile terminal.

Readers of this book should already know that any communication system—whether for radio transmission schemes or multiple access schemes—can be written in programming languages such as MATLAB or C. These computer simulation languages have a good relationship with software languages that configure DSPH such as DSP and/or FPGAs and/or ASIC. Typical software languages for DSPH are VHDL and Verilog-HDL. DSPH has been utilized to configure the mobile terminals and base stations of mobile communication systems. The DSPH is general-purpose, and the configuration can be programmed by downloading *DSP software* (DSPS). This means that users can download DSPS describing the desired elemental components into the DSPH of only one terminal. This is the basic concept of a software radio communication system.

If software radio is realized in the near future, mobile terminal manufacturers will be able to make products that are independent of the specifications of a particular telecommunication device. This means that users can select providers as they like (e.g., if a user finds that the capacity of one provider