**ENTS 656 Introduction to Cellular Communications**

**Fall 2021 Project**

**Due By Email: Part1: November 29 11:59pm, Part2: Dec 09, 11:59pm**

This project will explore the effects of downtilting antennas on cell performance, both from a coverage and an interference/throughput point of view. You are to implement a Python application which will simulate the downlink behavior of two basestations along a road communicating with mobiles travelling the road. The basestations can be considered to be 3 sectored basestations but you will only model one sector of each, and that sector will point down the road toward the other basestation (as is common for basestations along busy roads). The following picture should help illustrate this:

Picture not to scale

( ( (

) ) )

The basestations will use panel antennas which have a horizontal and vertical pattern. Since the road is directly in front of the antennas, there is no need for the horizontal pattern. The vertical pattern will be discussed in more detail in the vertical antenna discrimination section below.

**Basic and Basestation Parameters**

Your simulation will have the following basic parameters, some of which may be varied:

* **Road length**: 6 km or more
* **Simulation Time Step Size**: ΔT = 1 second
* **Total Simulation Time:** 1 or more hours.

The basestations will have the following properties, some of which may be varied:

* **Height**: hB = 50 m (This is the basestation antenna height)
* **Location**: 15 m orthogonal to the road at each end of the road as pictured
* **TX Power:** PTX= 43 dBm
* **Line/connector Losses:** L= 1 db
* **Antenna Gain:** GTX= 14.8 dBi (at boresight)
* **Antenna Tilt:** α = 2° typically, but can be between 0° and 10° for this project
* **Number of Traffic Channels:** NCH = 15 per sector
* **Frequency:** Both basestations will operate at fMHZ = 800 MHz
* **Shadowing:** log normal with μ = 2 dB and σ = 2 dB (**see channel properties below**)
* **Shadowing Resolution:** 20 meters

The mobiles will have the following properties, some of which may be varied:

* **Mobile Height:** hm = 1 meter (users are assumed to be in a car)
* **Handoff margin**: HOm = 3 dB (this will be used to reduce ping-ponging between basestations)
* **Mobile Rx Threshold**: RSLTHRESH = -100 dBm

**User Parameters**

We will assume all of the users for these basestations are on the road within the prescribed length. Users will appear along the road when they try to make a call. Assume the users’ initial positions when they appear are uniformly distributed along the road with the following characteristics, some of which may be varied:

* **Number of Users:** U = 150
* **Call Rate: λ =**2 calls per hour (on average)
* **Average Call Duration:** H=3 minutes/call (= 180 seconds/call)
* **User Speed:** Constant but Gaussian distributed with mean μ = 12 m/s and std σ = 3 m/s
* **Direction:** Users appearing on the road will head to the most distant basestation.

Your simulation should run with discrete time steps equal to one second each. At each second you will need to determine if each user, who does not currently have an active call, is going to make a call; this can be derived from the call rate, assuming users may make a call at any time with the same probability. (NOTE: we will assume our users are good citizens who are using hands-free devices while they are driving; therefore it would be impossible for them to have more than one call active at a time, or to make conference calls, texts, etc.) When the call is established, your program should determine the length of the call in seconds, along with the position, speed, and direction of the user’s car. Call lengths will be exponentially distributed with mean equal to the average call duration (this will be the scale factor needed by the exponential routine in numpy.random). Users will appear randomly along the road with a uniform distribution and will head in the direction of the furthest basestation; that is, if a user appears near the basestation at the beginning on the road, they will head toward the basestation at the other end of the road and vice versa. The users speed will be a Gaussian distributed random variable with mean and standard deviation given above. Users are assumed to establish a call on the basestation which appears with the highest RSL when the call is initiated. The RSL will be determined by computing the EIRP in the direction of the user from each basestation and subtracting the path loss obtained by modelling the communications channel for that basestation.

**Channel Properties and Path Loss**

The channel properties will determine the total path loss, which will include the propagation loss, shadowing, and fading as follows:

* **Propagation Loss**: Use the Okamura-Hata model adjusted for a small city. Include the mobile height term as required by the model. Note that you will need to compute the distance from the mobile to the basestation to get the propagation loss.
* **Shadowing**: Shadowing normally follows a log-normal distribution (i.e. a normal distribution in dB) with a zero mean and a standard deviation as high as 7 dB or so. However for this problem we are assuming the basestations are near the road so the mean will be larger (meaning less loss than PL50 would indicate) and the standard deviations will be smaller. You should take the mean μ = 2 dB and σ = 2 dB.
  + **NOTE!**: Shadowing is location dependent, NOT time dependent. Thus your simulation should construct shadowing values for each 20m section of the road and save these for use with all mobiles, depending on the mobile location on the road.
  + **NOTE!!**: The shadowing from one basestation will have NOTHING TO DO with the shadowing from the other. Thus you must create TWO shadowing arrays, one to use with each basestation.
* **Fading:** Use a Rayleigh distribution: the magnitude of a complex Gaussian distribution such that the real and complex parts have zero mean and unit variance (or use numpy.random.rayleigh).
  + Your simulation will be computing RSL values by subtracting the results of propagation from the EIRP and adding the shadowing and fading results. In your program, this will be compared to the mobile threshold value. It is unreasonable to design a system which drops a call as soon as the RSL falls below the threshold. Most real systems can survive an isolated deep fade. To model this in your simulation, each time you sample the Rayleigh distribution, generate 10 samples, throw away the deepest fade value and report the second deepest value as the fade value. In this way, the simulated system will only drop the call if 2 out of 10 of the values would end up below threshold. (NB: fading is a short time effect: you should assume the same values for propagation and shadowing will apply for all 10 fading samples.)
  + Note that the Rayleigh distribution described here (and also in the fourth Python homework assignment) gives **linear** values for the fading **multiplier**. You will need to convert to dB before **subtracting from** the other components to get a path loss value (that is, the fading value will “add” to the RSL). Also, remember that the fading value generated by either the magnitude of a complex Gaussian or by the Rayleigh routine in numpy will give a linear amplitude value that must be squared to give a power adjustment before converting to dB.
  + NB: This is a simple way to simulate the effects of fading in this system. A more accurate method would take account of the mobile speed/Doppler (e.g. Clark/Gans/Smith). However in our simulation, the simple method will be good enough since the speed of the mobiles and the environment is relatively slow.

**Vertical Antenna Discrimination**

Once you compute the total path loss, you must subtract it from the EIRP in the direction of the mobile to get the RSL. However, the EIRP in the direction of the mobile will NOT simply be the EIRP of the basestation at boresight. In general, this is a complicated problem (given horizontal and vertical antenna patterns, how do you interpolate to find the gain in a specific direction? Answer: this is hard to do accurately. TIA ANSI Standard 10 has details). In your case, however, things are greatly simplified. Since the antennas of both basestations point down the road, you do not need to worry about the horizontal pattern (i.e. take the horizontal discrimination to be 0, so if you were 50m off the ground and the antenna wasn’t tilted, you would be at boresight ). So to find the EIRP in the direction of the mobile, you simply need to take the boresight EIRP of the basestation and subtract the vertical pattern discrimination value for angle between the boresight direction and the mobile position. The following diagram should help illustrate the situation.

α = tilt angle

NOTE: α + β = γ

NOTE: α + β = γ

β = discrimination

γ

Boresight direction

In the diagram, γ is easy to determine as it is the angle from the mobile to the top of the antenna, which can be computed as an arctangent given that you know the distance to the antenna and the antenna height. If the antenna is downtilted by the angle α, then this must be subtracted from γ to get the discrimination angle β. The antenna discrimination values are given in dB and supplied by the manufacturer for each antenna model. The discrimination values for the antenna you will use are in vertical\_pattern.txt.

* **NOTE!**: It is entirely possible for discrimination angles to be negative. Thus you will find there is a need for the values at angles like 359°, etc.
* **NOTE!!**: Since your angles are not likely to be integers, you will need to interpolate between the values in the vertical pattern table to get the proper discrimination. Recall that given (x1,y1) and (x2,y2) in a table and x1<x<x2, with x2 – x1 =1 then y= (x-x1)\*y2 +(x2-x)\*y1 will be the interpolated value. WARNING: though python’s indexing makes it easier, you should still be careful when you interpolate points with negative angle values.

**RSL Values and S/I**

Mobiles which have a call up will need to evaluate the RSL from both basestations at each time step. This will be needed to determine if the call gets dropped or if the mobile attempts to hand off from one basestation to the other. This information can also be used to track the S/I and estimate throughput performance for data sessions. In addition to using the RSL information to determine what happens to the mobile, your script should keep track of mobile location, the serving basestation, and the S/I (that is, RSLSERVER –RSLOTHER). These values should only be tracked for users with a call up; users setting up a call will not be transferring substantial amounts of user data so you should not track those values. Once the simulation is completed, the values can be used to determine the number/percentage of packets in each section of roadway with high, medium, or low S/I. Areas with high S/I can use almost all their bits to carry data. As the S/I declines, more bits will be required for error correction which will reduce the data rate. In very low S/I areas, a lot of error correction will be required and the data rates will be low. Note that the calculation of the two RSL values will be different: each basestation will have a different EIRP (because of the angles), different propagation loss, different shadowing array and different fading.

**Simulation Details**

Your simulation should keep track of each call attempt and all calls. It should keep a table of active calls to update as the simulation time progresses. It should also keep track of various conditions that may arise such as hand off attempts, successful hand offs, hand off failures, capacity blocked calls, dropped calls due to signal strength, dropped calls due to capacity, completed calls, etc. It should also keep track of the number of active calls on each basestation (which you will need to know to determine if a call attempt blocks). Initialize the simulation by assuming that none of the users have active calls. Your simulation should calculate various initial settings that do not need to be calculated at every step such as the EIRP at boresight for each basestation and the two arrays of shadowing values for each position along the road (in 20m increments). The remaining steps are executed for every time step, ΔT, for the duration of the simulation time, Ttotal. Note that for easier understanding, the instructions for what to do if a user does not have a call up are listed first. When your program executes, however, it is important to first service the users who have active calls up before dealing with those who do not. This guarantees that requests for handoffs between basestations will take precedence over new call requests, which is as it should be. It also guarantees that if user 1 terminates a call, user 2 may claim the newly freed channel as a free channel (which may be a little optimistic, but that’s ok).

It would be a good idea to ask the user to enter the number of users (U), the tilt (α) and the total simulation time (Ttotal) as parameters, since the questions later will vary these values.

1. For each user that does not have a call up:
   1. Determine if the user makes a call request. Each user without an active call will have a probability = λ\*ΔT of making a call during that time step (NOTE: make sure the units match!!).
   2. If the user does not make a call request, we are done: go on to the next user.
   3. If the user DOES make a call request:
      1. Determine the user’s location along the road. Users are assumed to be uniformly distributed along the length of road (anywhere from 0 to 6 km for a 6 km road). The user’s location should be selected using floating point arithmetic.
      2. Determine the user’s direction (toward the furthest basestation.)
      3. Find the RSL at the mobile from each basestation. This will require the distance between the mobile and the basestation (for the Okamura-Hata model) as well as the vertical angle off boresight (for the antenna vertical discrimination). Pick the basestation with the highest RSL to be the one that will service the call request. This will be called the serving basestation. NOTE: we do NOT use this data to compute S/I at this point because the connection has not been established.
      4. Determine if the RSL from the serving basestation, RSLSERVER, is greater than or equal to the RSL threshold, RSLTHRESH, for the mobile. If not, that is, if RSLSERVER < RSLTHRESH then the call attempt fails and this is recorded as a dropped call due to signal strength for that basestation. We are done with this user and can go on to the next one.
      5. If the RSLSERVER ≥ RSLTHRESH then we can attempt to establish the call. Your program should check to see if there is an available channel on the basestation. If not, then the call is blocked; this is recorded as a blocked call due to capacity for this basestation. At this point you should see if the other basestation has sufficient signal strength to be the serving basestation.
         1. If the second best basestation has RSLSERVER ≥ RSLTHRESH, then check if a channel is available on that basestation. If so, proceed to then establish the call on that basestation as in vi below
         2. If the call cannot be established on the other basestation, then record this as a dropped call due to capacity for THE ORIGINAL BASESTATION ONLY. We don’t want to penalize the other basestation for trying to help out.
      6. If RSLSERVER ≥ RSLTHRESH and there is an available channel then the call is established. Your simulation should make an entry for this user in the table of active calls. Your program will need to determine the length of the call so it can keep a counter of the time left. As explained above, call lengths will be exponentially distributed. Compute a value for the speed for this user; this speed will be constant until the call terminates (successfully or unsuccessfully). You will need to keep track of the user’s location, direction of motion and speed, serving basestation, call time left, etc. This entry will be updated at each future time step until the call completes (either successfully or unsuccessfully as described below)
2. For each user who DOES have a call up:
   1. Update the user’s location. We assume users will continue in the same direction and same speed for the duration of their call.
   2. Update the call time. If the user’s call timer has run out, then the call completes normally. Record this as a successful call on that basestation. You can delete the entry in the active call list (or move it to an archive list). This will also free a channel on the serving basestation. We are done with this user for now.
   3. Check if the user’s location has moved beyond the ends of the road. If so, then we assume the user has handed off to another sector and everything is well. Record this as a successful call on this basestation (since we are not modeling the handoff procedure to other sectors, we will not record this as a successful handoff). This will also free a channel on the serving basestation. We are done with this user for now.
   4. Calculate the RSLSERVER and RSLOTHER. This will be at a new location, with new EIRP, path loss, shadowing and fading values. If RSLSERVER < RSLTHRESH the call drops. Record this as a dropped call due to signal strength for the serving basestation. The entry should be deleted from the active call list (or archived). The channel is now free and available for other users. We are done with this user for now.
   5. If RSLSERVER ≥ RSLTHRESH, compute the RSL for the other basestation, RSLOTHER. At this point save the location, serving basestation and S/I values in a list (or whatever) for later use. If RSLOTHER > RSLSERVER +HOm, then there is a potential handoff. NOTE: HOm >0 forces the call to stay on a weaker basestation a little longer, which limits excessive hand offs due to fading.
      1. Record this as a hand off attempt for this basestation
      2. If the other basestation has a channel available, then there will be a handoff. The other basestation becomes the new serving basestation, so the active call entry for this call will need to be updated. This is recorded as a successful handoff out of the old serving basestation. The call timer is not reset by the handoff and will continue to count in the next time step. A channel is freed on the old server and a channel is now in use on the new server for this call.
      3. If the other basestation does NOT have a channel available, then there will be a handoff failure. This is recorded as a failed handoff out of the old basestation. The call will NOT drop: it simply continues on the old basestation in the next time step. (it will likely try to hand off again in the next time step, but that is ok… it might succeed!)
3. Once all users have been processed, your simulation should collect statistics and update information about the basestations. After each hour of simulation time, your program should produce a report for the basestation. For each basestation your program should give
   1. the number of channels currently in use
   2. the number of call attempts
   3. the number of successful calls
   4. the number of successful handoffs
   5. the number of handoff failures into and out of each basestation
   6. the number of call drops due to low signal strength and also due to capacity
   7. the number of blocks due to capacity
   8. any other information you think will be useful

Your program should also produce a summary report at the end for the entire simulation time. Once your simulation is complete, your summary output will give a good overview of the coverage performance of the basestation. To estimate the throughput, look at the values stored in the S/I list. For the first basestation, for each 100m section of roadway, count the number of points which have S/I >= 10 dB (green points), the number with 10 dB> S/I >=5 dB (magenta points) and the number with S/I < 5 dB (red points). (There may be a way to use hist to do this, but I found it easier just to do it in a loop). Make a bar chart or plot of the green, magenta, and red values on the same graph so they can be compared. Do the same for the second basestation (a separate graph)

**Deliverables**

This is a complex project. To make it easier (and hopefully foster better performance) I am breaking it into 2 parts with 2 due dates.

**Part1**:

Good programming practice dictates that many parts of this should be broken out as functions and (perhaps) stored in a module. For the first part, write the functions required to compute propagation loss, EIRP in the direction of the mobile, shadowing from each basestation, and fading (as described above). Write a function to compute the RSLs from the basestations (which will probably call the other functions…). Test and debug each of these.

Write a short script that creates an array of points from 0 to 6000m (or 6 km) and computes the output of each of the functions above for a mobile at that distance along the roadway. For the first basestation, on a single graph, plot each of the following in different colors:

* EIRP at boresight (14.8) – propagation loss
* EIRP in the direction of the mobile, with tilt=0, – propagation loss
* EIRP in the direction of the mobile, with tilt=2, – propagation loss
* EIRP in the direction of the mobile, with tilt=5, – propagation loss
* EIRP in the direction of the mobile, with tilt=10, – propagation loss

Do the same for the second basestation (on a separate graph).

For tilt=2, make a single graph with the following in different colors:

* EIRP in the direction of the mobile – propagation loss for the first basestation
* EIRP in the direction of the mobile – propagation loss for the second basestation
* RSL including shadowing and fading for the first basestation
* RSL including shadowing and fading for the second basestation

Submit your code and your graphs by **November 29 11:59pm.**

**Part2:** Using your functions from part1, create the rest of the simulation.

Q1: Run your simulation for four hours with the parameters given. Use a downtilt of 2°. How many problems (e.g. drops, blocks, hand off failures) occur? What percentage of call attempts have a problem so severe that they drop? What is the major cause of drops (e.g. signal strength, capacity, etc)? Does this basestation perform well?

Q2: Rerun the simulation with tilt = 5° for both basestations. What is the effect on drops and performance? Rerun again with tilt=10°. What is the effect on drops and performance?

Q3: Change the tilt back to2°. Increase the number of users from 150 to 400. Rerun your simulation for 4 hours. How many problems occur now, and what is the new percentage of call attempts that end up dropped? Compared to Q1, what is the main cause of the additional problems (e.g. drops from signal strength? Or blocks from capacity? Or Hand off failures? Etc.)

While the structure of the simulation is for you to decide/develop, I am insisting that you create the functions you will use and make sure they run correctly before you start the main simulation (Part1). This way, as you work on the simulation, you can at least be confident that things are being correctly computed (even if your code is doing something bizarre with the results). However the rest of the program, data structures and general implementation are entirely up to you.

Please submit the Python source code implementing the simulator, all the functions, and scripts to run it. In addition, **write and submit a report** giving the results generated by your code and the answers to the questions given in Q1, Q2, and Q3. The report should include the output from your code for all the runs (typically as an appendix) All material should be submitted ELECTRONICALLY as email attachments to [mdellomo@umd.edu](mailto:mdellomo@umd.edu).

PLEASE NOTE: in addition to correctness and functionality, the code will also be evaluated for style. Thus, you should pay attention to software design and engineering issues such as: code modularization, code block organization, variable naming, comments, etc. Your report will also be evaluated.

A word about academic integrity: Copying code (or rewriting code or copying algorithms or sharing code etc.) from other students or from the web is **strictly prohibited** and will not be tolerated. You may discuss the project ONLY IN GENERAL TERMS with other students. If you are having problems, please see either the TA or me during our office hours. Please submit a signed honor pledge form with your code (below)

Lateness: For Part1, you will lose one point for each day late. If you have not submitted it by class time on Thursday Dec. 02, I will want to talk to you... For Part2, while the project is due on Thursday Dec 09, you will not be penalized for lateness until Sunday Dec 12 at 11:59pm. Projects submitted on Monday, Dec 13 will lose 20 points. Projects submitted on Tuesday Dec 14 will lose 40 points. Projects submitted on Wednesday Dec 15 will lose 60 points. No projects will be accepted after 11:59pm on Dec 15. Note: 20 points will cost you ~½ a letter grade in the course (e.g. A => B+). MAKE SURE YOUR PROJECT ARRIVES AT MY INBOX BY 11:59PM SUNDAY DEC 12! NOTE: This is a hard project. Start it NOW… or else!

HONOR PLEDGE: I pledge on my honor that I have not given or received any unauthorized assistance on this assignment.

Signature\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_