IK2219/IK3506
Homework #2: Random Waypoint Simulation

Due on 23:55, September 21st (Sunday), 2014

Jeong-woo Cho

Rules

Each student (no grouping) should carry out this homework. Although you are encouraged to discuss with each other, you have to write programming codes and solutions **in your own words**. You can hand in solutions either in a **single** Acrobat PDF file or Microsoft Word file. The length of the submission should not be more than 5 pages (font size: 10-12, paper size: A4/Letter).

If you **miss** the deadline specified on the front page of this document and hand in your solutions within exactly one week after the deadline, **30% of the total points will be taken off**, regardless of your actual score. Solutions handed in more than one week after the deadline will not be graded. Most importantly, if your solutions are incorrect, you are highly likely to receive 0 points for the corresponding questions. In this light, you should double-check if **your solutions are unequivocally correct**.

Note again that you are allowed to use only built-in functions in MATLAB and its toolboxes, *e.g.*, Statistics Toolbox. It is not allowed to use SIMULINK.

Instructions

By doing this homework, you are expected to learn how to implement a basic simulator in MATLAB. Being able to write program codes for your simulations will be very helpful when you need to assess various scenarios in communication networks. First of all, as introduced in the textbook (Chapter 6), there are two representative ways to implement a simulation program, *i.e.*, (i) *Discrete Event Simulation* (aka DES), and (ii) *Stochastic Recurrence*, among which you can choose one according to your personal preference. However, it is of utmost importance to implement your simulation program *correctly* because an *incorrect* simulation program will lead to a completely wrong result, which will in turn deduct your points considerably.

In particular, as discussed in Chapter 6.2.1 of the textbook, the core of *Discrete Event Simulation* approach lies in the so-called *event scheduler*. Once you know how to implement and handle an event scheduler, you will be able to write a number of simulation programs for various cases on top of the event scheduler module. An event scheduler is basically a list of events, often in order of increasing time. To implement such a list, you must be able to modify the size of the list dynamically because, generally speaking, the maximum size of the list cannot be known in advance and the processed events must be deleted from the list so as to cope with the memory scalability issue.

There can be various ways to implement an event scheduler in MATLAB. One of the easiest ways which might not be very inefficient is to use the data type called *Structure Array*. For example, the following commands create 1×3 struct array with fields (i) name, (ii) billing and (iii) test:

```
patient.name = 'John Doe';
patient.billing = 127.00;
patient.test = [79, 75, 73; 180, 178, 177.5; 220, 210, 205];

patient(2).name = 'Ann Lane';
patient(2).billing = 28.50;
patient(2).test = [68, 70, 68; 118, 118, 119; 172, 170, 169];

patient(3).name = 'New Name';
```

In order to delete the the second element of the struct array, *i.e.*, `patient(2)`, just type:

```
patient(2) = [];
```

In the meantime, in order to add a new element at the end of the struct array, type the following:

```
patient(end+1).name = 'Pooh';
```

```
patient(end).billing = 223.9;
patient(end).test = [23, 35, 34; 100, 99, 91; 162, 131, 200];
```

where the command in the first line allocates memory for the new element `patient(3)` (`end` indicates last array index). Since the value of `end` changes from 2 to 3 due to the allocation of the new element after the execution of the first line, you have to use simply `end` (not `end+1`) in the remaining two lines. Note however that the above method provides you a list of events, *not* in order of increasing time. Note also that the above rather inefficient method is intended for grasping how simulations can be conducted because you are unlikely to use MATLAB for heavy simulations occurring in real research projects. You can spare yourself from troubles by adopting this simple method but you are free to choose your own ways.

Lastly, though the simulation in this homework is simplistic, writing the simulation program might take much more time than you would predict in case you have not ever written such a program code. You would better start this homework as soon as possible.

Problem 1: Simulator

Write a MATLAB program to simulate the random waypoint model defined in Chapter 6. Use the following parameters.

- Number of users: $N = 100$.
- The area is a rectangle of dimensions $l \times L$ with $l = L = 1000$ meters.
- $v_{min} = 1$ m/sec, $v_{max} = 5$ m/sec.
- The simulation terminates at (simulated) time $T_s = 86400\ sec = 1$ day.

Do **not** attach your MATLAB program here. In the last problem of this homework, you will be asked to attach a *single* MATLAB program code which generates solutions to all the problems in this homework.

Display the trajectory (use `plot()` with **solid lines**) of one user on the area ($l \times L$). Display the waypoints (use `plot()` with **dots**) of one user on the area. Can you observe that the waypoints are distributed uniformly over the area? How much real time did your simulation take for one day of simulated time?

Problem 2: Different Viewpoints

In this problem, we investigate how two different viewpoints (or sampling methods) affect the metrics.

EVENT AVERAGE (PALM) VIEWPOINT

(a)

Display the positions of all users sampled at transition epochs (when state transition occur). To confirm that these positions are distributed uniformly on the entire area, show the histogram of positions of all users sampled at transition epochs (in other words, waypoints). To this end, you have to divide the whole area into square bins and show the frequencies of positions falling into each square bin. To spare yourself trouble, you are recommended to use `hist3()` function in Statistics Toolbox. To make the bars of this histogram colored according to height (frequencies), use the following commands:

```
hist3(data);
set(get(gca,'child'),'FaceColor','interp','CDataMode','auto');
colormap(gray);
```

(b)

Show the histogram of speeds of all users sampled at transition epochs.

TIME AVERAGE VIEWPOINT

(c)

Now sample the positions of all users every 10 seconds. Now we repeat (a) based on these samples. Do you still think that the positions are uniformly distributed over the area? Describe how the positions are distributed.

(d)

Now sample the speeds of all users every 10 seconds. Show the histogram of speeds of all users based on these samples. Do you still think that the sampled speeds follow a uniform distribution?

COMPARISON

(e)

Now compute the mean of speeds of all users for the following two cases: (i) when speeds are sampled at transition epochs, and (ii) when speeds are sampled every 10 seconds. Show the two means (in m/sec) such that the number of digits to the right of the decimal point is two. For example, π is shown as 3.14. Which sampling method leads to a relatively smaller mean of speeds?

(f)

Attach here a single MATLAB program code which generates all solutions (in particular, figures) to all problems in this homework. You don't need to commentate the code.

Before finishing this homework, make sure that there are eight figures in total in your submission.