# Network Analysis and Simulation - Homework 4

Michele Polese, 1100877

June 11, 2015

## Exercise 1 - A discrete event queue simulator in MATLAB

The discrete event queue simulator is based on the simulator proposed by Law in [1]. It is a simplified version of a more general DE simulator since it has to handle just one server and the events may be only arrivals or departures. The objective is to measure the delay that users suffers when entering in the system for different utilization factors $\rho$ and for different interarrival/service times. The simulator is designed to be as more general as possible, given that the code is written in MATLAB. In particular the simulator uses the following counters:

- `clock` the current system time
- `next_arr` the time of the next arrival. It is updated whenever there is an arrival, by extracting a random interarrival time according to the specifics of the queue
- `next_dep` the time of the next departure. It is updated by extracting a random service time if an user arrives and finds the queue empty or if a user departs and there is at least one other user waiting in the queue. Instead, when the system is empty the `next_dep` has no meaning and it is set to `next_arr` + 1.
- `time_of_last_event` is the time at which the previous event happened
- `server_status` set to 0 if the server is free, 1 if it is busy
- `number_in_queue` number of users in the queue (= number of users in the system - 1)
- `times_of_arrival` a FIFO queue that collects the time of arrival of each user
- `number_of_events` and `renewal_instants` are 2 counters that could set the stopping time of the simulation. Generic simulations stop when the number of events (either arrivals or departures) has reached a given maximum. Instead, if the queue has memoryless arrivals (Poisson or Bernoulli iid), the simulator counts a renewal instant when an arrival finds the queue empty, and the simulation stops when this counter reaches a given threshold, which is lower than the maximum number of events. Indeed a renewal interval is completely representative of the statistics of the queue and since for small utilization factors $\rho$ they are short, then it is sufficient a smaller number of events than the maximum allowed.

There's an initial phase of initialization, with the extraction of the first interarrival time (and the first departure time set to the previous plus 1), and the counters all to 0. Then at each step the simulator checks whether the next event is an arrival or a departure by comparing `next_arr` with `next_dep`. This explains why when the system is empty `next_dep` is set to `next_arr` + 1: the next event will be surely an arrival.
If the event is an arrival, the simulator extracts the next interarrival time $t_a$ and stores `clock` + $t_a$ in `next_arr`. It checks if the queue is empty, in this case it schedules immediately a departure for the user just arrived. If not, the user is backlogged in the queue. In both cases the arrival time is stored in `times_of_arrival`.
If the event is a departure, the simulators performs the following operations. At first it checks if there is someone left in the queue. If not, it sets the next departure time as at the beginning at `next_arr` + 1, instead if a user is ready enters in service it computes a random service time. Since a user has just left, the head of the queue `times_of_arrival` is read and popped out. The delay for that user is computed as the difference between current clock value and its arrival time, and added to a cumulative metric $D_t$. The number of departed users $u_d$ is increased by one.
The interarrival and service times are generated with 2 dedicated functions, that can be easily extended in order to support any kind of slotted or continuous interevent time. As for now, they can handle

- Poisson arrivals/departures, with exponential interevent times

- Arrivals/departures with probability $b$ in each slot, i.e. interevent times are geometric of mean $1/b$
- Interevent times of 1 or 2 slot with probability 0.5 each

The slot length can be specified.

## Exercise 2 - Read logs and analyze data

In this second exercise the goal was to extract a look-up table from a log file of a simulator and use it to compute some measurements on useful quantities in underwater optical communications. In particular, the log files are obtained from the simulations of the ambient light irradiance $E_0$ with the simulator HYDROLIGHT. From the 3 different dump files (each for a different value of $c \in [0.15, 0.4, 2.19]$ with $c$ the attenuation coefficient) I recovered the values of $E_0$ and the corresponding depth $z$ in meters using a `perl` script. It uses a regular expression to identify the right rows of the log and selects the columns with the desired values. In Figure 1 $E_0$ is plotted as a function of the depth $z$.

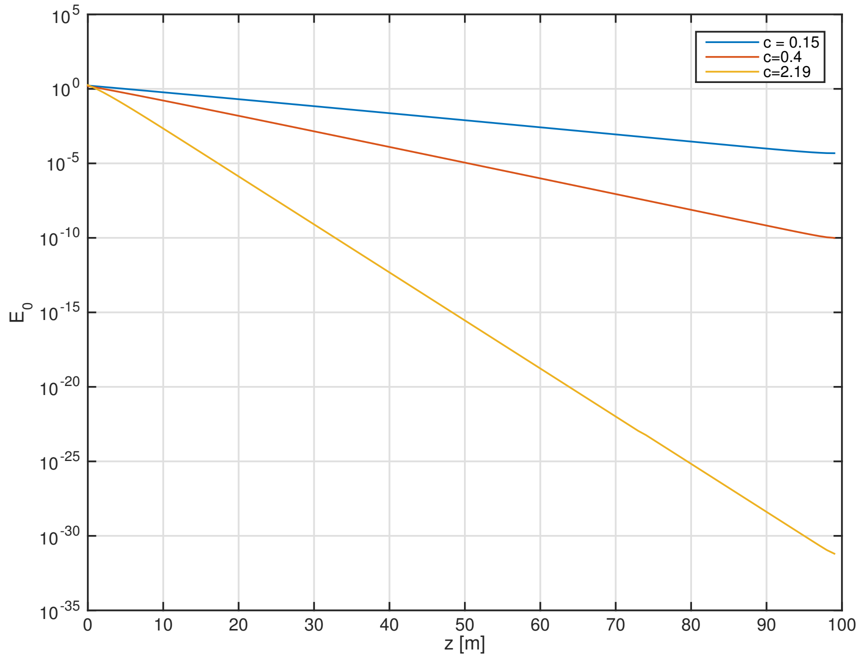


Figure 1: Irradiance $E_0$ vs. $z$

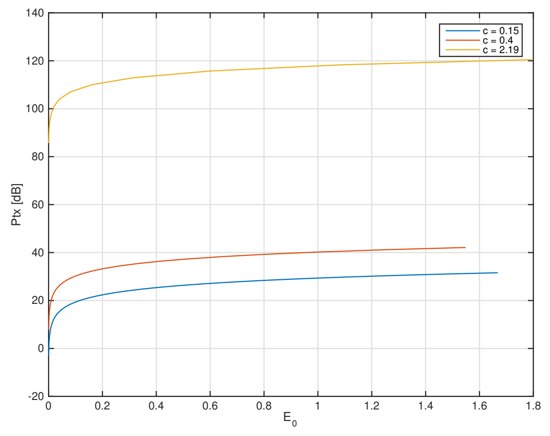
Some evaluation on the propagation are then carried out with a MATLAB script that uses the provided parameters for receiver and transmitter. The propagation model used the one proposed in [3] and assumes that there is a successful transmission if the SNR is over a certain threshold. In particular from the irradiance and the parameters $S$ (sensitivity) and $A_r$ (area of the receiver) of the considered modem it is possible to compute the power of ambient light noise, which is $N_A = (SE_0A_r)^2$. Let the received power $P$ be

$$P = \frac{2P_{tx}A_r\cos(\beta)}{\pi d^2(1-\cos(\theta)) + 2A_t}e^{-cd} \tag{1}$$

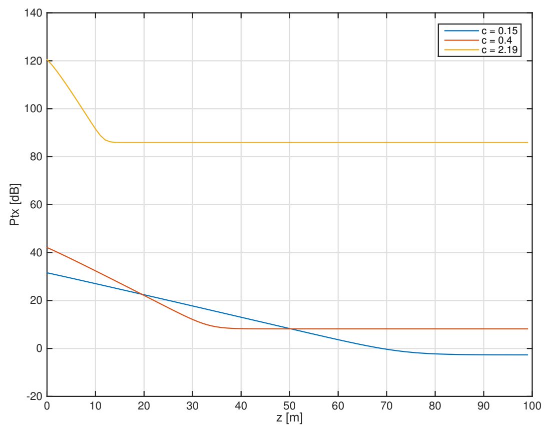
then the SNR is

$$\Gamma = \frac{(SP)^2}{2q(I_D + I_L)BW + \frac{4KTBW}{R} + N_A} \tag{2}$$

In the following, if the SNR is above the threshold of 20 dB then there is a successful transmission. In particular, by setting the threshold and a distance $d = 10$ m, by inverting (2) in order to get $P$ and (1) in order to compute $P_{tx}$ it is possible to know which is the minimum transmission power that can be used to reach a distance $d$. In Figure 2[a] there is the plot of the minimum $P_{tx}$ against the irradiance $E_0$, while in Figure 2[b] the x axis is the depth $z$ considered.



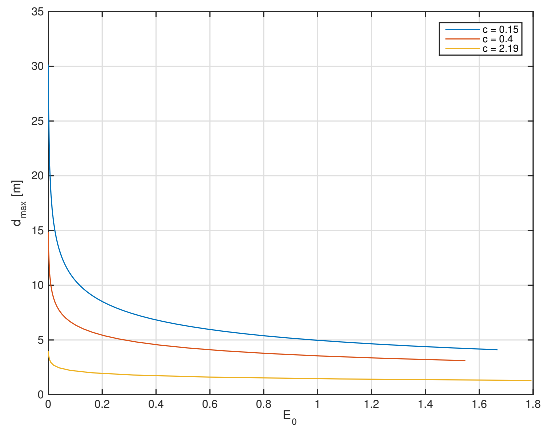
(a) $P_{tx}$ to reach $d = 10$ m as a function of $E_0$



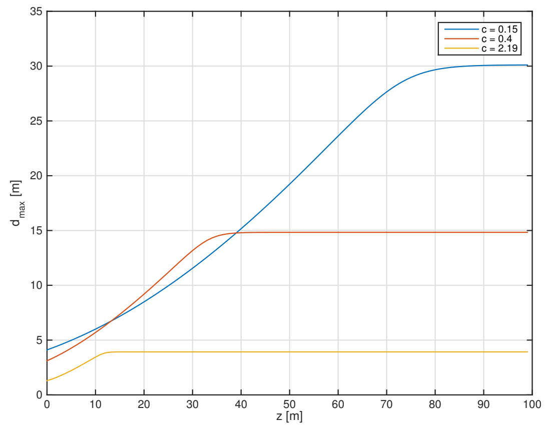
(b) $P_{tx}$ to reach $d = 10$ m as a function of $z$

Figure 2: Transmitted power $P_{tx}$ required to transmit at a distance of $d = 10$ meters

Finally, the transmit power is set to $P_{tx,max} = 100$ W and the maximum distance at which is possible to transmit is evaluated. Note that (1) is not invertible with respect to the distance $d$, therefore the received power $P$ given by (1) is precomputed for $d \in [0, 50]$ m and stored in a vector `P_d`. Then, given a certain $E_0$, from (2) the minimum received power to guarantee an SNR of 20 dB is computed and the closest $P$ in `P_d` is found. The corresponding value of $d$ is chosen as the maximum distance $d_{max}$ at which it is possible to transmit with $P_{tx,max} = 100$ W and that $E_0$ - or the corresponding $z$.
The results are in Figure 3.



(a) $d_{max}$ as a function of $E_0$, for $P_{tx,max} = 100$



(b) $d_{max}$ as a function of $z$, for $P_{tx,max} = 100$

Figure 3: Maximum distance $d_{max}$ with a transmit power of $P_{tx,max} = 100$

## References

[1] A.M. Law, Simulation Modeling and Analysis, 4th ed., McGraw Hill, 2006

[2] Y. Le Boudec, Performance Evaluation of Computer and Communications Systems, EPFL, 2015

[3] D. Anguita et al., Optical wireless underwater communication for AUV: Preliminary simulation and experimental results, in Proc. IEEE/OES Oceans, Santander, Spain, Jun. 2011