University of the Witwatersrand, Johannesburg
School of Electrical and Information Engineering
ELEN7015 Teletraffic Engineering

Tutorial Sheet 1: Introductory Probability and Teletraffic

Introduction

This tutorial is part computer exercise, part modelling and part classical teletraffic calculation.

GNU Octave: The prerequisite is to have GNU Octave available on your computer and to be familiar in the first instance with its command line interface. Later it will be necessary to create function files. The teletraffic-oriented function files available on the course website should be loaded onto your computer in directory `C:\Program Files\GNU Octave 2.1.5\Octave_files`. (Note: The version may not be 2.1.5.)

If you have never encountered Octave before (or MATLAB) here are a few ways of getting up to speed, listed in order of increasing complexity. The good thing about Octave is that one can learn in a just-in-time manner.

- Introduction to Octave: see
  `http://www-mdp.eng.cam.ac.uk/web/CD/engapps/octave/octavetut.pdf`
- Octave has help available from its command line:
  - The Octave help function is invoked by typing `help xxx` into the command line, where xxx is the function name or operator you want information on.
  - Help is also available on operators, for example + - * / .+ .- .* ./ and \.
  - To see all the built in functions and operators in Octave, simply type `help`.
- The Octave download comes with an HTML help file.
- The full GNU Octave Manual (356 pages) is available at many sites: just Goolgle Eaton OctaveManual.pdf.

Probability Theory: Appendix A of the course notes should be consulted for concepts and definitions on probability.

Conventions: Things that you type in the Octave command line are shown in `typewriter font` in the exercises, for example `y = m*x + c` and are all followed by the Enter key.

Exercise 1:

To find out how Octave works, do the following exercises.

1. Type the following into the command line: `x = rand(5,10)` followed by `Enter`.
   Note the matrix `x` of random numbers with 5 rows and 10 columns.
2. Pick out one of the entries in the matrix `x`, for example `x(4,3)`.
   Note that the first subscript identifies the row and the second the column.
3. See how to transpose a matrix: `y = x'`
4. Try `x = rand(5)`.
5. Produce row and column vectors of random numbers by selecting the dimensions.

Exercise 2:

Now let's do things with random numbers.

1. Generate a long sequence of random numbers, for example `x = rand(1,1000)`.
2. Does the sequence have some structure? Try `plot(x)`
   If the graph does not appear, click on the gnuplot graph bar at the bottom of the screen.
3. Look at some properties of `x` and learn some Octave functions, for example:
   The maximum `max(x)`
   The minimum `min(x)`
   The average `mean(x)`
4. Now let's see how the values of `x` are distributed `hist(x)`.
   Here we see the histogram or relative frequencies of values of `x`.
   To get the relative frequencies (and not plot), use `p=hist(x)`.

Exercise 3:

Now let's do some simulation of some familiar games of chance: tossing a coin, rolling dice and playing Roulette.

A function file called throw.m is available that can be used by the command `z = throw(m,n)` that simulates `n` attempts at producing an event that has `m` equally likely outcomes. For example `z = throw(6,1000)` simulates 1000 throws of a fair dice.

We can see how many times a particular number came up by using the histogram function `hist(z,[1:1:6])`. The vector in the second parameter defines the centres of the bins for classifying the throws (if omitted, 10 bins are used).

The function has at its core one statement:

```
zz=fix(rand(m,1)*n)+1;
```

How does this statenment generate the required events?

Exercise 4: Now lets bring some theory to bear.

Lotto has been described a a game for the statistically challenged. Lets look at it theoretically. Considering only the first ball, what is the probability of a given number coming up, say 15? When the second ball is drawn, what is the probability of another number, say 23? What is the probability of getting a particular pair of numbers with the first and second balls, say 15 and 23? What is the probability of getting a particular combination of six numbers for the six balls?

Exercise 5: In many cases we want to generate is a random event that has a given probability of occurring. The function `throw` does this if the first parameter is less than 1 and specifies the required probability, for example `z = throw(0.15,1000)`.

The core code in throw for `n`< 1 is just two statements. The output zz is a vector which is zero valued except the locations where the event with specified probability occurs.

```
zz=zeros(m,1);
zz(find(rand(m,1)<n))=1;
```

How does this code do the necessary computation?

Note: Octave can be used interactively and this is often useful for getting to understand how thing work. For example entering the following sequence of commands does the same thing (Note that omitting the colon at the end of the statement displays the result on the screen).

```
m=100
n=0.15
zz=zeros(m,1)
r = rand(m,1)
k = find(r<n)
zz(k)=1
```

Now do the same thing using row vectors instead of column vectors.

Exercise 6:

Doing some Erlang B state and blocking probability calculations. An m-file called `erlangb` is available for calculating the Erlang B distribution. Find out how it works using the `help` function. Try a few calculations with different offered traffic and serving trunks. Plot the state distribution.

Taking the Erlang B formula

$$p_k = \frac{A^k/k!}{\sum_{k=0}^{N} A^k/k!}$$

one could programme the function as follows, assuming the offered traffic `A` and number of channels `N` are input parameters.

```
aa = [1;cumprod([A*ones(N,1)])]  % Calculate A^k
kf = [1;cumprod((1:N))];         % Calculate k!
aaa = aa./kf                     % Calculate A^k/k!
d = sum(aaa);                    % Calculate denominator
p = aaa./d;                      % Calculate the state probabilities
```

Look at how this function is programmed (to see the internals of the function use `type erlangb`). Why not use the above self-evident way of implementing the Erlang B function?

Exercise 7:

The average holding time for voice calls is 150 s. What is the the probability that a call will last (a) less than half the average holding time, and (b) more than twice the average holding time. What are the 5th and 95th percentiles of the call duration distribution?

Exercise 8:

A concentrator accommodates 1000 subscribers and is connected to the local exchange by four PCM30 time division multiplex lines. How is this concentrator modelled for traffic analysis.

Find the offered traffic per line that can be accommodated if the blocking probability is to be not greater than 0.02.

Exercise 9:

A single stage switch has 30 inlets and 10 outlets. The traffic offered to the switch has the following parameters. The average holding time is 120 seconds and the average number of calls per line per hour per hour is four. Calculate the blocking probability. State all assumptions made.

Exercise 10:

A trunk between a PBX and the PSTN has 30 time slots. At the busy hour, the offered voice traffic (both directions) is 24 Erlang. What is the blocking probability?

Exercise 11:

Consider a 30 channel trunk. Obtain graphs of the state probabilities for offered traffic of 4, 8, 16 and 24 Erl offered traffic. What is the most likely state in each case?

Repeat for 300 timeslots and traffic of 40, 80, 16, 240, 260 and 280 Erl.

Exercise 12:

In the derivation of the Erlang B distribution, we use the normalisation criterion $\sum_{k=0}^{N} p_k = 1$ If we unrealistically assume that we have unlimited servers or channels the corresponding step is $\sum_{k=0}^{\infty} p_k = 1$. Show that, with this form,

$$p_0 = e^{-A} \tag{1}$$

$$p_k = \frac{A^k}{k!} e^{-A} \tag{2}$$

that is, a Poisson distribution of states. This is attractive to use as calculations are simpler than using the Erlang B formula.

Taking a 30 channel multiplex and various levels of traffic, numerically examine the accuracy of Blocking Probabilities calculated from the Poisson form relative to the Erlang B formula.

Exercise 13:

A concentrator requires digit receivers to collect dialled digits. There are M subscribers each offering 0.08 Erl with a speech holding time of 120 s. The average holding time of a digit receiver is 12 s. The design objective is that not more than one call attempt in a thousand should fail because no digit receiver is available. What number of digit receivers is needed for the following numbers of subscribers connected: $M = 1000,\ 2000$ and 5000.

Exercise 14:

A GSM base station is euipped with transceivers, each having a single frequency bamd that accommodates eight TDMA channels. Determine the number of users that can be accommodated in a cell with blocking probability of 2% due to the transceiver capacity when the base station has 1, 2, 3 and 4 transceivers. Justify the model used and state the assumptions.
