# Using MATLAB-based Laboratories to Demonstrate Wireless Communications Systems Principles

**Kathleen A. Kramer**
**University of San Diego**

## Abstract

The usefulness of mathematical simulation to study complex communications concepts, combined with the preparation it provides students for continuing work in the field, make the inclusion of computer simulation exercises or laboratories very appropriate for augmenting undergraduate student learning in these areas of communications. A series of three such laboratories have been developed within a course in wireless communications. Among the communication systems topics investigated in the exercises are: channel allocations to minimize blocking probabilities; the effects of Rayleigh fading on bit-error rate; and the selection of psuedo-random codes to minimize cross-correlation. The paper presents these simulation exercises.

## Introduction

The study of wireless and other advanced communications systems topics at the undergraduate level has developed at some universities, in response to the increased interest resulting from the impressive research and development in such technology over the past several years. Work with digital and wireless communication systems often involves the application of mathematical principles that are themselves complex and or must be applied to a very large numbers of variables. As a result, both industry professionals and academic researchers often use computer simulation, including mathematical simulation programs, such as MATLAB or Mathcad, in their study of these advanced topics.

The exercises are appropriate for use as student laboratory exercises, as a supplement to hardware laboratory exercises, or as outside assignments for courses that do not have a laboratory component. The paper describes the simulation exercises, and reports on the use of these exercises in a university setting to augment a course in wireless communication system principles.

## Course Setting and Motivation

The Electrical Engineering Program at the University of San Diego is a 9-semester undergraduate program requiring 152 semester units. Situated within a primarily liberal arts university, the program culminates in a dual BS/BA degree for its graduates. The program features broad-based course requirements and small class sizes. USD Engineering Programs

were recently ranked 15th nationally amongst non-PhD granting engineering programs by *U.S. News and World Report*.

The extensive requirements and limited size of the program put severe limits on engineering elective offerings. In an effort to meet student demand and to provide students with knowledge relevant to local industry, a special topics course, Wireless and Digital Communications was offered in Fall 1999, only the second of such special elective offering for the program. The course was structured with three hours of lecture and a three-hour laboratory each week.

The primary prerequisite for the course was a senior-level required course in communications systems. The prerequisite course has itself several prerequisites including upper-division mathematics courses, a course in signal and systems analysis, and two courses in electronics.

Despite the rather extensive course background of the students, the theoretical nature of the communication systems topics and the complexity and cost of appropriate hardware systems motivated the development of simulation exercises.

## Simulation Exercises

Three different simulation exercises described here were written to be performed using MATLAB, although other mathematical computation packages such as Mathcad would also provide a suitable platform.

The three exercises described here are:

1. Calculating Auto-Correlations and Cross-Correlations of Binary Sequences
2. Grade-of-Service and Trunking
3. Observing Results of Flat Fading

The first of the exercises builds upon a previous laboratory where students had built a simple PN-code generator using 7400-series ICs and then developed MATLAB modules allowing them to generate an arbitrary PN-code. In this exercise, the students again generate their own pseudo-random codes and then measure auto-correlations and cross-correlations using MATLAB code that they write. Listings for modules are given in Figure 1. The student handout gives students some basic functions, then students write more advanced simulations.

The second exercise has students perform an analysis of the relative trade-offs involved in capacity and trunking efficiency for a communications system, such as a cellular telephone system, where blocked calls are either cleared or queued. Listings for modules are given in Figure 2. In this exercise, students start with some examples the demonstrate use of some of the plotting features of MATLAB and a function to help avoid some difficulties that could arise if computations are not performed in an efficient order.

```
function sr = codeout(poly, shiftreg)
% this function implements a linear-feedback shift register
% poly is a vector specifying which bits of the shift register
%      are tapped.
% shiftreg is a vector specifying the current contents of the register
% sr is the output of "codeout"; it is the updated shift register contents

order = length(shiftreg);
feedbck=0;
for n = 1:order

   feedbck = xor(feedbck,and(shiftreg(n),poly(n+1)));
end

sr = [feedbck shiftreg(1:(order-1)) ];  %perform shift of contents
```

**Listing 1.** MATLAB Code to Implement a Code Generator

```
sr = []      % need to specify initial content of register
taps =[]        % need to specify tap configuration - same size as sr
p = [1 taps]; n=length(sr);
lengthseq=  % need to specify how many bits of sequence to generate
bitseq = zeros(1,lengthseq);  %create sequence variable

for i = 1:lengthseq
   bitseq(i) = sr(n)  % new bit of sequence is last bit of register
   sr = codeout(p,sr)
end
```

**Listing 2.** MATLAB Code to Generate Sequence

```
function out = match(bitu,bitv)
% function outputs +1 when bits match and -1 when they don't
if bitu ==0
   bitu=-1;
end
if bitv ==0
   bitv=-1;
end
out = bitu*bitv;
```

**Listing 3.** MATLAB Code to Compare Bits

```
function correlation = autocorr(a)
% function outputs a vector representing the autocorrelation
% of the vector a
bits = length(a);
correlation = zeros(bits,1);
for i=1:bits
   for i2=1:i  %i2 is where overlap begins
      offset = bits-i;
      correlation(i)=correlation(i)+match(a(i2),a(i2+offset));
   end
end
```

**Listing 4.** MATLAB Code to Generate Auto-Correlation

**Figure 1.** MATLAB listings for Exercise 1

```
function fx = factdiv(m, n)
% this function implements m!/n!, where m>=n
% m!/n!  = m * (m-1) * ...*(n+1)
fx = 1;
for i=(n+1):m
   fx = fx*i;
end
```

**Listing 1.** MATLAB function factdiv.m to compute ratio of two factorials

```
lambda = (1/2)*1/(60*60)   %calls/s each user
H = 3*60 % call duration in seconds
Au = H*lambda;
C=40;
U=(2*C):(4*C):(100*C);  %range on number of Users, U
GOS = zeros(1,length(U));
for i=1:length(U)
   GOS(i)=erlangb(Au*U(i),C); %find GOS for each U value
end

figure   % plot is semilog GOS vs. U with axes and labels specified
semilogy(U,GOS,'-b',U,GOS,'or')
xlabel('Number of Users');
ylabel('Call Blocking Probability');
title('GOS (Call Blocking Prob) vs. Number of Users');
axis([1 100*C 0.00001 1])
```

**Listing 2.** MATLAB code to plot GOS vs number of users

```
function prob = erlangb(A, C)
% this function implements the Erlang-B equation
% A is total offered traffic
% C is number of trunked channels

sum = 0;
for k = 0:C
   sum = sum+ (A^k)*factdiv(C,k);
end
prob = A^C/sum;
```

**Listing 3.** Erlang-B Equation

```
function prob = erlangc(A, C)
% this function implements the Erlang-B equation
% A is total offered traffic
% C is number of trunked channels

temp = 0;
for k = 0:(C-1)
   temp = temp+ (A^k)*factdiv(C,k);
end   %summation times C!

temp = temp*(1-A/C);
```

**Figure 2.** MATLAB listings for Exercise 2

The third laboratory first has the students observe signals experiencing Rayleigh fading by using antennas, a function generator, and a portable oscilloscope and to observe a multipath signal in the time domain. Using MATLAB, students then calculate the effects of fading on the received data under varying conditions that they can control in simulation.

```
function rate = lcr(roh, fm)
% this function finds the average fade duration
% for a given level to rms ratio and max doppler

rate = ((2*pi)^0.5)*fm*roh*exp(-roh*roh);
```

**Listing 1.** Level Crossing Rate function

```
function time = avgfade(roh, fm)
% this function finds the average fade duration
% for a given level to rms ratio and max doppler

time = (exp(roh*roh)-1)/(roh*fm*((2*pi)^0.5))
```

**Listing 2.** Average fade duration function

```
v1=0;
v2 =1:10:120
c=3*10^8;
f=900*10^6; %set modulation freq
lambda = c/f;
fm = v2/lambda; % find max doppler freq

roh = [0.01 0.1 0.5  1];
fadetime= zeros(length(fm),length(roh));
for j=1:4
   for i=1:length(fm)
   fadetime(i,j) = avgfade(roh(j), fm(i))
   end
end
plot( v2, fadetime(:,1),v2, fadetime(:,2),v2, fadetime(:,3),v2,
fadetime(:,4));
legend ('roh = 0.01', 'roh = 0.1', 'roh = 0.5', 'roh =1');
xlabel('Velocity');
ylabel('Avg Fade Duration');
title('Avg Fade Duration vs. Mobile Velocity');
axis([min(v2) max(v2) 0 max(fadetime(:,2)) ]);
%use roh=0.1 for axis scale
```

**Listing 3.** Code to graph Fade Duration vs. Velocity

```
function eprob = ber(bitrate, roh, fm)
% this function calculates bit error prob for given
% data rate, roh, and max doppler freq

fadetime = avgfade(roh,fm);
bitsperfade=ceil(fadetime*bitrate)
Nr = lcr(roh, fm);
errpersec = Nr*bitsperfade;
if (errpersec > bitrate)
   errpersec=bitrate;   %can't have more errors than bits
end
eprob = errpersec/bitrate
```

**Listing 4.** Code to graph BER vs. Velocity

**Figure 3.** MATLAB listings for Exercise 3

## Conclusion

The three exercises were developed to enhance student laboratory experiences within the senior-level elective course in wireless communications systems at USD in the Fall 1999 semester. Students evaluated their experiences very positively. More than half of those students who completed the course went on to accept jobs doing work related to course topics or to graduate studies. The course will be repeated again in Fall 2001 and the instructor is working with individuals from local industry to continue to develop relevant laboratories.

## Acknowledgements

The author would like to acknowledge the contributions of Chuck Pateros and Mark Miller of ViaSat, Inc., Scott Denton of Applied Microcircuits Corporation, and Ricardo Valerdi of Motorola for their contributions to the laboratories developed for the course. She would also like to thank Professor F. Cassara of Polytechnic University for the valuable contributions of the Faculty Enhancement Workshop on Wireless Communications that he led in June 1999.

## References

1. Kathleen Kramer and Thomas F. Schubert, Jr. , “Demonstrating Complex Communication Systems Principles Using Electronic Courseware and a Simple Computer Math Package”, Proceedings of the 1998 American Society of Engineering Educators Conference, June 1998.

2. Theodore S. Rappaport, Wireless communications: Principles and Practices, Prentice-Hall, Englewood, NJ, 1996

3. materials from "Wireless Communications"[NSF-sponsored Faculty Enhancement Workshop directed by Frank Cassara], Polytechnic University, Farmingdale, New York, June 1999

4. Thomas F. Schubert, Jr., "The Use of a Simple Computer Math Package to Demonstrate Complex Communication Systems Principles ", Frontiers in Education Conference Proceedings, November 1994.

KATHLEEN A. KRAMER is an Associate Professor of Electrical Engineering at the University of San Diego. Her teaching and research interests are in the areas of digital systems and communications systems. She received her MS and Ph.D. in Electrical Engineering from the California Institute of Technology, and her BS in Electrical Engineering (with a second major in Physics) from Loyola Marymount University. She is a consulting member of technical staff at ViaSat, Inc. in Carlsbad, CA, where she does research on topics in communication systems and signal processing.