Appendix C

Introduction to MATLAB®

MATLAB® is an efficient simulation tool for various math and engineering disciplines. In addition, MATLAB® is a language for technical computing, where scripts can be written using built-in functions and used for a given simulation problem. Therefore, MATLAB® performs computation, programming and visualization using a user-friendly interface that enables complex design, modeling and simulation problems to be handled efficiently.

The basic element in MATLAB® is the array. Therefore, it handles computations using matrix-based programming where variables, functions or mathematical expressions are dealt with as matrices. Mathematical operations are applied to these matrices and yield new matrices. MATLAB® can perform different types of operations including arithmetic and logical operations, mathematical functions, graphical functions, and input/output operations (Czylwik, 2005). Mathematical operations can be performed using MATLAB® expressions that are based on variables, numbers, operators and functions. These expressions involve matrices and are used to perform computations for a given problem.

In this appendix, an introduction to MATLAB® including the basic MATLAB® tools needed for the simulation of communications systems is presented. These include deterministic and random signals generation and Fourier analysis. The basics of the use of MATLAB® commands and their functionalities will not be discussed here and the readers are referred to different MATLAB® handbooks, MATLAB® manuals and MATLAB® help which are all available on the web site of TheMathWorks™ (MATLAB® Documentation, 2009).

C.1 MATLAB® Scripts

MATLAB® can perform a wide variety of mathematical operations using its commands, which are built-in functions. These functions can be used in the main command menu or by using M-files. Simple problems can be solved efficiently in the command menu but if the problem involves the use of a large number of commands, then an M-file needs to be used. M-files can either be scripts or functions. A script performs a number of commands one-by-one in a similar way to the main command window. On the other

Nonlinear Distortion in Wireless Systems: Modeling and Simulation with MATLAB®, First Edition.
Khaled M. Gharaibeh.

hand, a function accepts a set of input arguments and returns another set of output arguments. The main difference between scripts and functions is that the variables in a function are local to that function and do not appear on the workspace, while all the variables in a script can be recalled in the workspace. On the other hand, the first line in a function needs to start with the declaration of the function, its input arguments and its output arguments. This line appears as

```
function [a,b,c] = MyFunction(x, y, z)
```

where the `MyFunction` is the name of the function, `a,b,c` are the output arguments and `x, y, z` are the input arguments.

C.2 MATLAB® Structures

A structure in MATLAB® is a group of fields that are accessed by the structure name and the field name. For example, the commands:

```
Circuit.InputResistance=50
Circuit.OutputResistance=1e3
Circuit.Capacitance=1e-12
```

establish a MATLAB® struct under the name `Circuit` that has the fields `InputResistance`, `OutputResistance`, and `Capacitance`. When this struct is called in the command menu, the fields are displayed as:

```
>> Circuit
Circuit =
     InputResistance: 50
    OutputResistance: 1000
         Capacitance: 1.0000e-012
```

Stucts are useful in simulations of communication systems because a number of parameters of a system or a process can be grouped in one variable. For example, objects used to measure the performance of a communication system such as `commmeasure` are structs.

C.3 MATLAB® Graphics

MATLAB® is a very efficient tool for visualization of vectors and matrices through its various two- and three-dimensional plotting functions. For example, the `plot` function provides cartesian plots of a given vector with the ability to specify a color, axes, labels, titles, markers, etc. Other 2D plotting functions include `bar`, `stem`, `stairs`, `polar`. On the other hand, 3D plotting can be performed through the various 3D plotting functions such as `mesh`, `surf`, `surfl` and contour. These functions provide 3D plots defined by four matrix arguments (X, Y, Z, C) of 2D functions of the form $Z = f(X, Y)$ where C defines the color map of the plot.

C.4 Random Number Generators

Uniform random numbers can be generated in MATLAB® using the command `rand(1,n)`, which generates an $1 \times n$ vector of uniformly distributed random numbers

in [0,1]. The default random numbers have a zero mean and a unity variance. For example, the command

```
x=rand(1,100)
```

generates a random vector `x` of size 100 whose elements are uniformly distributed within the range [0,1]. To generate random numbers that are uniformly distributed in an arbitrary interval [a,b], a simple linear transformation of the form $y = (a - b)x + b$ can be used and then the random vector y is uniform in [a,b].

```
x=rand(1,100); %Uniform in (0,1)
a=2;
b=5;
y=-3*x+5;  % Uniform in (2,5)
```

To generate discrete uniformly distributed random numbers, such as random bit sequences, the functions: `floor`, `ceil`, `sign` can be used. For example, the command:

```
a=floor(N*rand(1,n)+1);
```

generates an n by 1 vector elements that are uniformly distributed integer numbers within the range [1,..,N]. The command

```
a=sign(2*rand(1,n)-1);
```

generates an n by 1 vector elements that are uniformly distributed integer numbers that take the values of ± 1 only; i.e. the values ± 1 are generated with equal probability. Another example is a random bit generator that takes the values 1 or 0:

```
a=ceil(rand(1,100)-.5)
```

Gaussian random numbers can be generated using the function `randn(n,m)` which generates an n by m matrix of Gaussian distributed random numbers with zero mean and unity variance in a similar way to uniform random numbers.

To generate random numbers of a given distribution, the theory transformation of random variables can be used (Papoulis, 1994). Using the theory of inverse transformation of random variables, a random variable y with density function $f_y(y)$ can be generated from a uniformly distributed random variable u in [0,1] using the transformation $y = F_y^- 1(u)$ where $F_y(y)$ is the distribution function of y.

For example, to generate an exponentially distributed random variable with $f_y(y) = e^{-x}U(x)$ where $U(x)$ is a unit step function, we transform a uniformly distributed random variable u according to $y = F_y^- 1(u)$. Therefore, $F_y(y) = 1 - e^{-y}$ and hence, $y = F_y^- 1(u) = -\ln 1 - u$. The following script generates an exponentially distributed random variable.

```
% Generate a uniformly distributed random vector
x=rand(1,1000);
% Apply transformation to produce an exponentially
% distributed random variable
y=-log(1-x);
% Plot the histogram of x and y
```

```
hist(x,50)
hist(y,50)
```

MATLAB® also has built-in functions in The Statistics Toolbox (The Statistics Toolbox, 2009) that supports the generation of most of the well-known distributions.

C.5 Moments and Correlation Functions of Random Sequences

Table C.1 shows the commands used to compute the moments and correlation functions of random vectors.

C.6 Fourier Transformation

Fourier transform is implemented in MATLAB® in its discrete version, the DFT, which can be computed using the FFT algorithm. Consider a time-domain signal given by an n by 1 vector x, its FFT is computed using the function `FFT(x,N)` which computes the N-point FFT, padded with zeros if n<N and truncated if n>N. The function `FFTSHIFT(FFT(x))` shifts zero-frequency component to the center of the spectrum producing a centered spectrum.

The following script computes the FFT of a signal and plots the spectrum versus frequency.

```
clear all
% Frequency of single tone
f=100;
% Sampling frequency
fs=20*f;
% Sampling time
Ts=1/fs;
% Define a time vector
t=0:Ts:2e4*Ts;
% Single tone signal
xt=cos(2*pi*f*t);
% Find FFT
len=length(xt);
Xf=fftshift(fft(hanning(len)'.*xt/len));
len=length(Xf);
% Define frequency resolution
delf=fs/len;
% Set the frequency vector
m=len/2;
z=-(m):(m-1);
f=delf*z;
% Plot FFT
plot(f,Xf)
```

The PSD of a random process can be found in a similar way from the FFT of the autocorrelation function.

Table C.1 Moments and correlation functions in MATLAB®

Command	Description
mean(x)	mean value of x- or nth power of x
var(x)	variance
std(x)	standard deviation
cov(x,y)	covariance of x and y
corrcoef(x,y)	Correlation coefficient of x and y
xcorr(x,y)	Autocorrelation of x or cross correlation of x and y
xcov(x,y)	Autocovariance of x or cross covariance of x and y

C.7 MATLAB® Toolboxes

In addition to the built-in functions in the MATLAB® environment that represent general-purpose functions, MATLAB® has a number of application-specific packages of functions called "Toolboxes". MATLAB® Toolboxes are software packages that provide specialized functions, plots, simulation algorithms and many other extra capabilities to the built-in components in the MATLAB® environment that serve different fields of math, statistics and engineering. Table C.2 lists the main toolboxes available from TheMathWorks™. In the context of the subject of this book, the most commonly used toolboxes are the RF, Communications and Signal Processing toolboxes.

Table C.2 MATLAB® Toolboxes (MATLAB® Documentation, 2009)

Toolbox	Areas Covered
Communication	Communication System Design
RF	RF Circuit Design
Signal Processing	Signal Processing algorithms and systems
Filter Design	Filter Design
Partial Differential Equations, Statistics	Math & Statistics
System Identification, Optimization, Curve Fitting Statistics Wavelet Spline	System Optimization and Identification
Genetic Algorithm, Neural Network, Fuzzy Logic, Fixed Point	Search Algorithms
Control System, Instrument Control, Robust Control, Model Predictive Control	Control System
Data Base, Data Feed	Data Management
Embedded, Parallel Computing	Computer Engineering
Image Processing Image Acquisition	Image Processing

C.7.1 The Communication Toolbox

The Communication Toolbox provides a large number of functions, plots, graphical user interfaces that serve as tools for the efficient simulation of communication systems. There are a number of features that make this toolbox an efficient tool in the design and analysis of communication system, among which are the following categories (The Communication Toolbox, 2009):

- Libraries of functions for the generation of various communication signals using random data, Table C.3.
- Libraries of functions for simulation of communication channel models that include the commonly used channel models, Table C.4.
- Libraries of communication channel equalizers that aids the design of equalizers to combat the effects of channel impairments, Table C.5.
- Libraries of functions for evaluation of communication system performance, Table C.6.

Furthermore, the toolbox contains a graphical user interface, the BERTool, which can be used for simulation of BER and comparing simulated values with analytical and semianalytical results of most modulation formats.

C.7.2 The RF Toolbox

The RF toolbox provides tools for creation, simulation and visualization of passive and active RF circuits/networks and their characteristics. Table C.7 shows the RF components and networks and the corresponding data and visualization objects supported by the toolbox.

Table C.3 Communication signal generation in The Communication Toolbox (The Communication Toolbox, 2009)

Sublibrary	Functionality	Commonly Used Functions
Signal Sources	Generation of WGN, Random Symbols, Random Integers, BER Patterns	`wgn`, `randsrc`, `randint`, `randerr`
Source Coding	Quantization, Companding, Huffman Coding, PCM	`quantiz compand dpcmenco dpcmdeco`
Error Detection and Correction	Channel Coding: Block, Convolutional, CRC Codes	`encode, decode, gen2par, syndtable hammgen rsgenpoly bchenc, bchdec, rsenc, rsdec`
Special Filters	Pulse Shaping: RC filters	`rcosflt`
Modulation & Demodulation	Analogue Modulation: AM, FM; Digital Modulation: ASK, MPSK, MQAM, DQPSK, OOPSK, MFSK, MSK, GMSK	`ammod`, `pmmod`, `fmmod`; `modem` object

Table C.4 Channel models in The Communication Toolbox (The Communication Toolbox, 2009)

Channel	Commonly Used Functions
AWGN	`awgn`
MIMO	`mimochan` object
Fading	`rayleighchan`, `ricianchan`
Binary Symmetric Channel	`bsc` `rcosflt`

Table C.5 Equalization in The Communication Toolbox (The Communication Toolbox, 2009)

Equalization Technique	Adaptive Algorithm	Commonly Used Functions
Linear	CMA, LMS, RLS	`lineareq`, `equalize`, `lms`, `rls`, `cma`
Decision Feedback	CMA, LMS, RLS	`dfe`, `equalize`, `lms`, `rls`, `cma`
MLSE (Maximum-Likelihood Sequence Estimation)	CMA, LMS, RLS	`mlseeq`, `equalize`, `lms`, `rls`, `cma`

Table C.6 Evaluation of communication system performance in The Communication Toolbox (The Communication Toolbox, 2009)

Performance Metric	Method	Commonly Used Functions
BER	Simulation	`symerr`, `biterr`, `berfit`, `berconfint`
BER	Semianalytic	`semianalytic`
BER	Theoretical	`berawgn`, `bercoding`, `berfading`, `bersync`
Communication Scopes	Eye Diagram, Scatter Plot	`commscope.eyediagram`, `commscope.ScatterPlot`
Communication Measurements	EVM, Modulation Error Rate (MER)	`commmeasure.EVM`, `commmeasure.MER`

The toolbox also contains the RF Analysis GUI that provides a visual interface for creation, analysis and simulation of RF components and networks and enables the visualization of their parameters. Figure C.1 shows the layout of the RF GUI.

C.8 Simulink®

Simulink® has been a very useful tool for analysis, modeling and simulation of dynamic systems in various disciplines of engineering. Through its graphical user interface (GUI), Simulink® enables models of linear and nonlinear systems to be built as block diagrams and simulated in continuous, discrete or hybrid times.

Table C.7 RF circuit analysis, simulation and visualization in The RF Toolbox (The RF Toolbox, 2009)

RF Circuit/Network	Passive networks, Amplifiers and mixers, Amplifiers and mixers, Transmission lines, LC ladder filters, Networks
Creation Object	`rfckt`
Circuit Analysis	`analyse`, `calculate`, `extract`, `listformat`, `listparam`
Visualization	`plotyy`, `smith`
RF Data Object	`rfdata`
Data Extraction	`extract`, `read`, `write getdata`
Type of Data	Network parameters, Spot noise, Noise figure, Third-order intercept point (IP3)
Data Format	SnP, YnP, ZnP, HnP, AMP

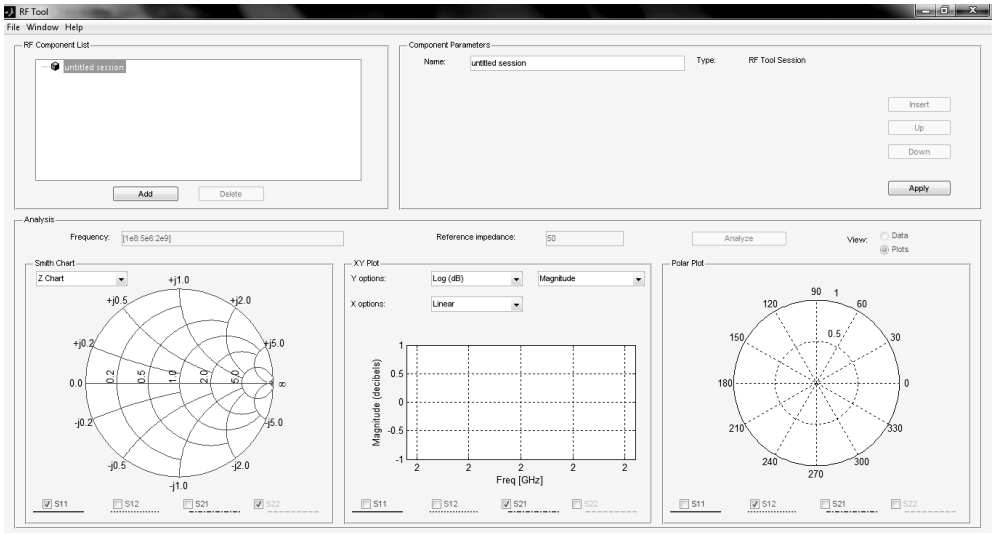

Figure C.1 RF GUI in the RF toolbox (The RF Toolbox, 2009).

Design and modeling with Simulink® is based on model-based design. Complex system models can be built in Simulink® using its comprehensive libraries of blocks which represent submodels and components. These libraries contain a wide variety of sources, sinks, components and connectors. Simulink® blocks are hierarchical, which means that depending on the level of detail required by the design, modeling or simulation problem, these models can be modified by changing their internal design, which can be viewed by looking under their masks. On the other hand, a number of block can be grouped in one block with a certain number of inputs and outputs. New blocks can be built using existing blocks in the library to fit a user requirement for a given design that makes Simulink® unlimited in covering the various engineering problems.

Table C.8 Simulink® Blocksets and specialized software (Simulink® User's Guide, 2009)

Blockset/Toolbox/Software	Areas Covered
Communication	Communication system design
RF	RF Ccircuit/system design
Signal Processing	Signal processing algorithms and systems
Video and Image Processing	Image processing
SimPowerSystems	Power system design
SimElectronics	Electronic system design
SimHydraulics®, SimMechanics, SimDriveline	Mechanical systems
Simscape	Physical systems design and simulation
SimEvents®	Discrete-event simulation
Design Optimization	Model parameter numerical optimization
Real – TimeWorkshop®	Embedded system design and verification
Control Design	Control systems and models

Simulink® software is integrated with the MATLAB® environment where it could use MATLAB® for a number of tasks such as storing model output, calling MATLAB functions within a block, loading a stored signal, etc.

In a similar way to MATLAB®, Simulink® has a number of "Blocksets" and software that contain specialized blocks for various disciplines. Table C.8 shows the main blocksets supported by Simulink®.

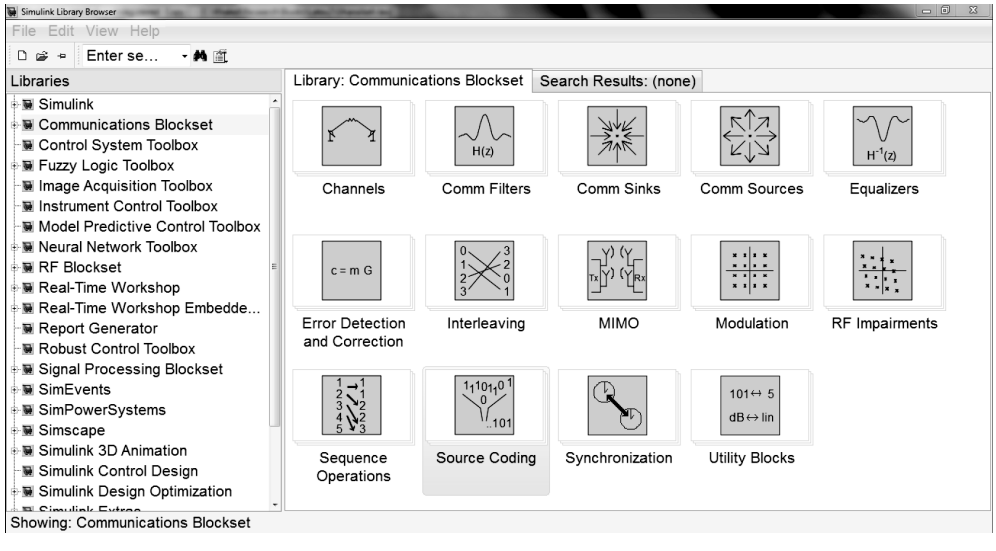

Figure C.2 Sublibraries in the Communication Blockset (The Communication Blockset, 2009).

Table C.9 Sublibraries in the Communication Blockset (The Communication Blockset, 2009)

Sub-Library	Functionality	Blocks
Communication Sources	Random Data Generation	Noise, Sequences BER Patterns
Communication Sinks	Visualization	Error statistics, Scopes
Source Coding	Quantization, Companding	Quantization, Companding
Error Detection and Correction	Channel Coding	Block, Convolutional, CRC Codes
Interleaving	Interleaving	Block, Convolutional
Communication Filters	Pulse Shaping	RC filters
Modulation & Demodulation	Analogue/Digital Modulation	AM, FM; Digital Modulation: ASK, MPSK, MQAM, DQPSK, OOPSK, MFSK, MSK, GMSK
Channels	Channel Models	AWGN, Fading, Binary Symmetric
MIMO	MIMO Systems	Space-Time Coding
RF Impairments	RF Impairments	Nonlinearity and I/Q imbalance, phase/frequency offsets, phase noise, thermal noise and path loss
Equalizers	Adaptive Equalizers	Linear equalizers, Decision-feedback equalizers, MLSE
Sequence Operation	Operations on sequences	Scrambling, puncturing
Synchronization	Timing, Phase Recovery	Timing, Phase and Carrier Recovery, voltage-controlled oscillator (VCO), phase-locked loop (PLL)

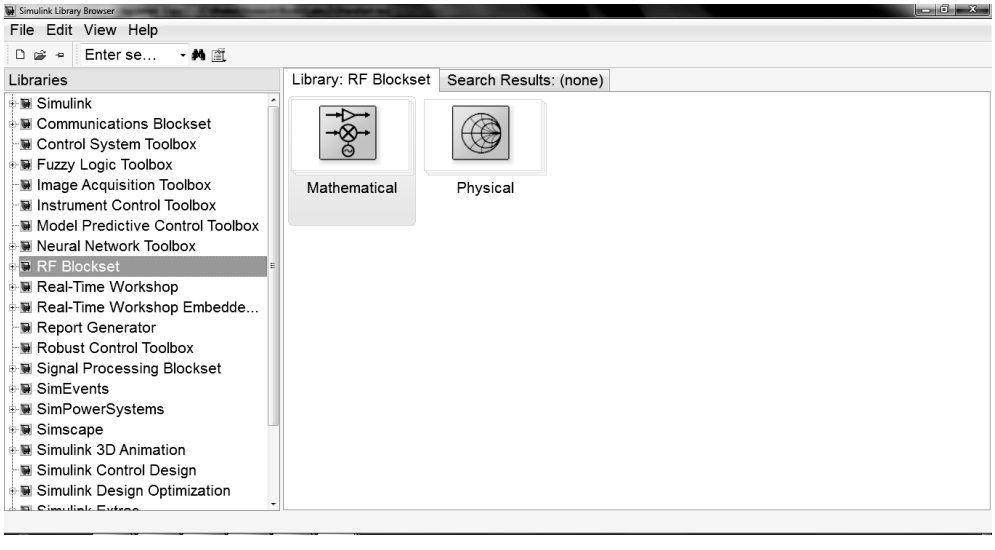

Figure C.3 Sublibraries in the RF Blockset (The RF Blockset, 2009).

Table C.10 Sublibraries in the RF Blockset (The RF Blockset, 2009)

Sub-Library	Blocks
Physical	Amplifiers, Black Box Element, I/O Ports, Mixers, Ladder Filters, RLC Circuits, Transmission Lines
Mathematical	Amplifiers, Mixers, Filters (BPF, BSF, HPF, LPF),

The most important Blocksets in the context of topics covered in this book are The Communication Blockset, The RF Blockset and The Signal Processing Blockset.

C.8.1 The Communication Blockset

The Communication Blockset contains a number of sublibraries that cover various aspects of communication system design and simulation. Figure C.2 shows the sublibraries within The Communication Blockset and Table C.9 shows their description.

C.8.2 The RF Blockset

The RF Blockset contains two sublibraries that cover various aspects of RF system design and simulation. The mathematical sublibrary contains blocks with models based on mathematical representations of the amplifier, mixer, and filter blocks. The physical sublibrary contains models for physical and electrical components and are based on specifying physical properties of components or on importing measured data. Figure C.3 shows the sublibraries within the RF Blockset and Table C.10 shows their description.