Communication Systems

BPSK Communication system design with Simulink

Farnam Adelkhani

The bit error rate analysis of BPSK, binary phase shift keying, modulated signal detection is executed in company of additive white Gaussian noise. Matlab and Simulink are utilized to implement a simulation of this system. Then analysis and calculation is possible of the probability of error with systems operating with the additive white Gaussian noise and by doing so the performance of modulation techniques can be compared with theoretical values.

Introduction

One of the primary concerns in digital communication system design is to receive the data so it is as similar to the data sent by the transmitter as possible. The system is required to be analyzed to find the probability of error, this gives insight to the systems performance. Different modulation techniques can offer a range of performance pros while dealing with signals that would normally be effected with noise. This paper will focus on comparative performance analysis of basic M-ary PSK modulation schemes, specifically BPSK.

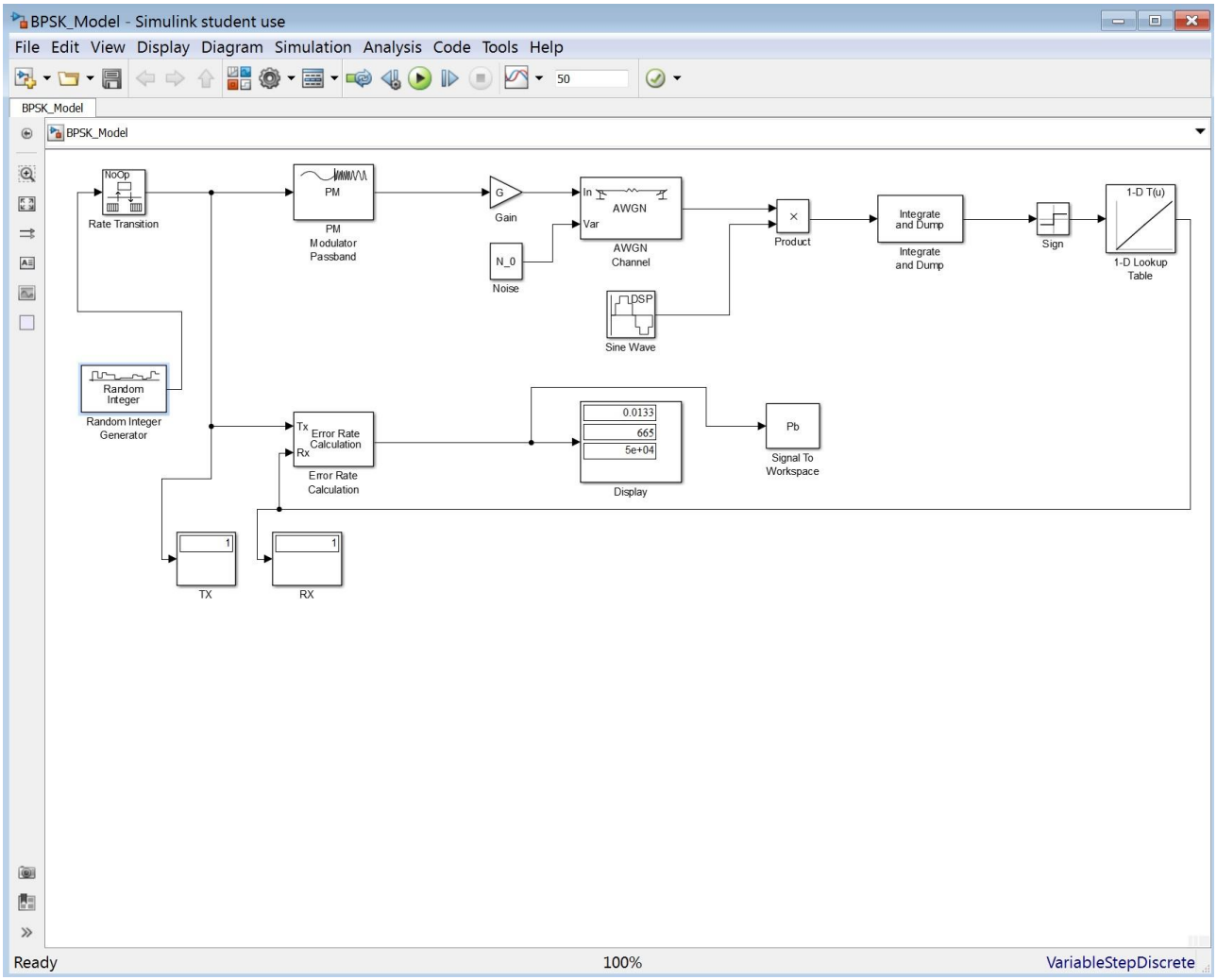
Background Theory

The bit error in digital transmission is the number of received bits from a data stream over a communications channel that has been altered due to interference such as noise , distortion or bit sync errors. The bit error rate is the amount of bits in error divided by the total number of transferred bits during a considered time interval. This is therefore a unit less performance measure, kind of like a benchmark.

Procedure

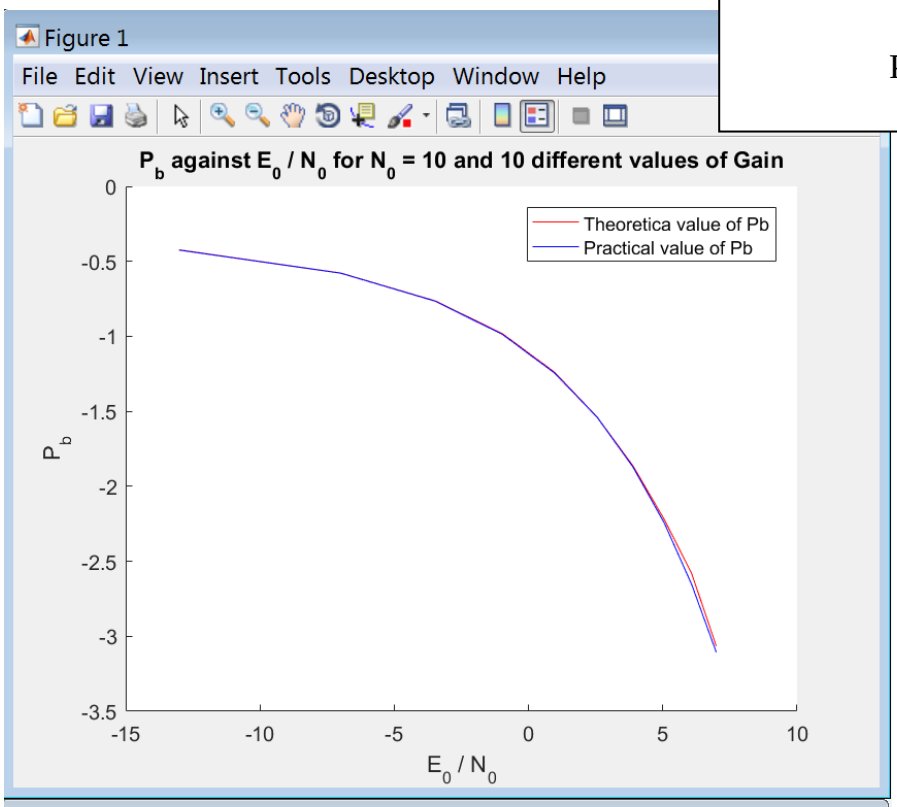
The idea behind the work in this section, is that after I have set the Simulink model, I can then easily and conveniently go back to edit my Matlab script with the goal of setting the variables as desired, then call the model again for simulation. I will call the output from the Simulink system for use again in the script; for plotting purposes. This is just for convenience. The theoretical and practical models of the system are calculated and derived from the simulation for plotting on the same graph. The two systems should lineup and although it may not be visible without zooming into the plot, the data is actually within reasonable range of each other.

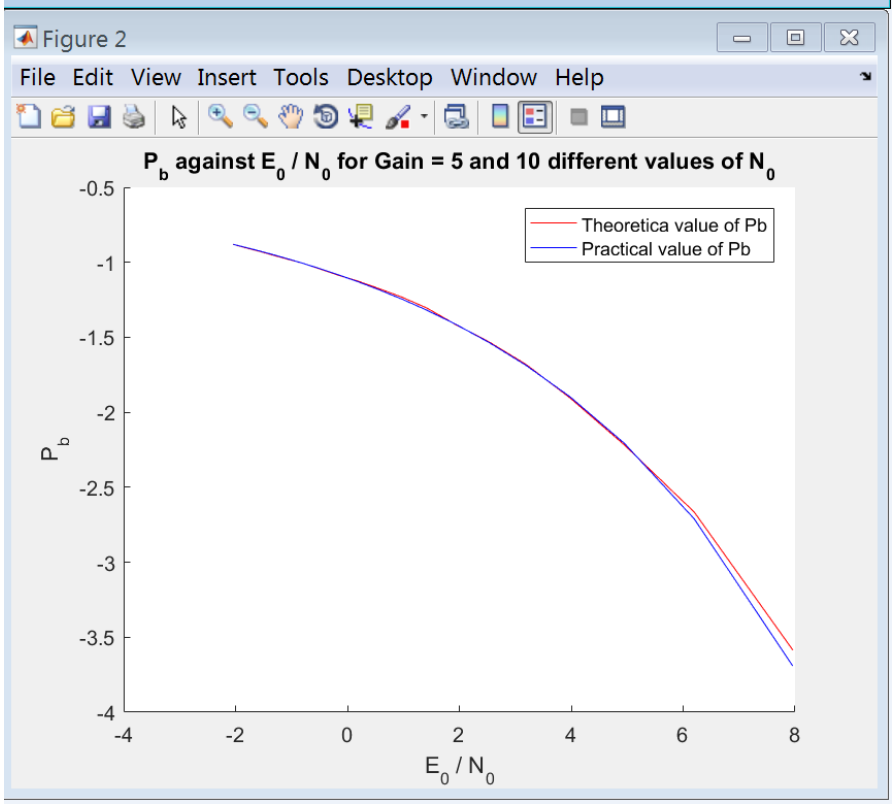
The BPSK Simulink system implemented is shown below:



Results

P_b vs E_0/N_0

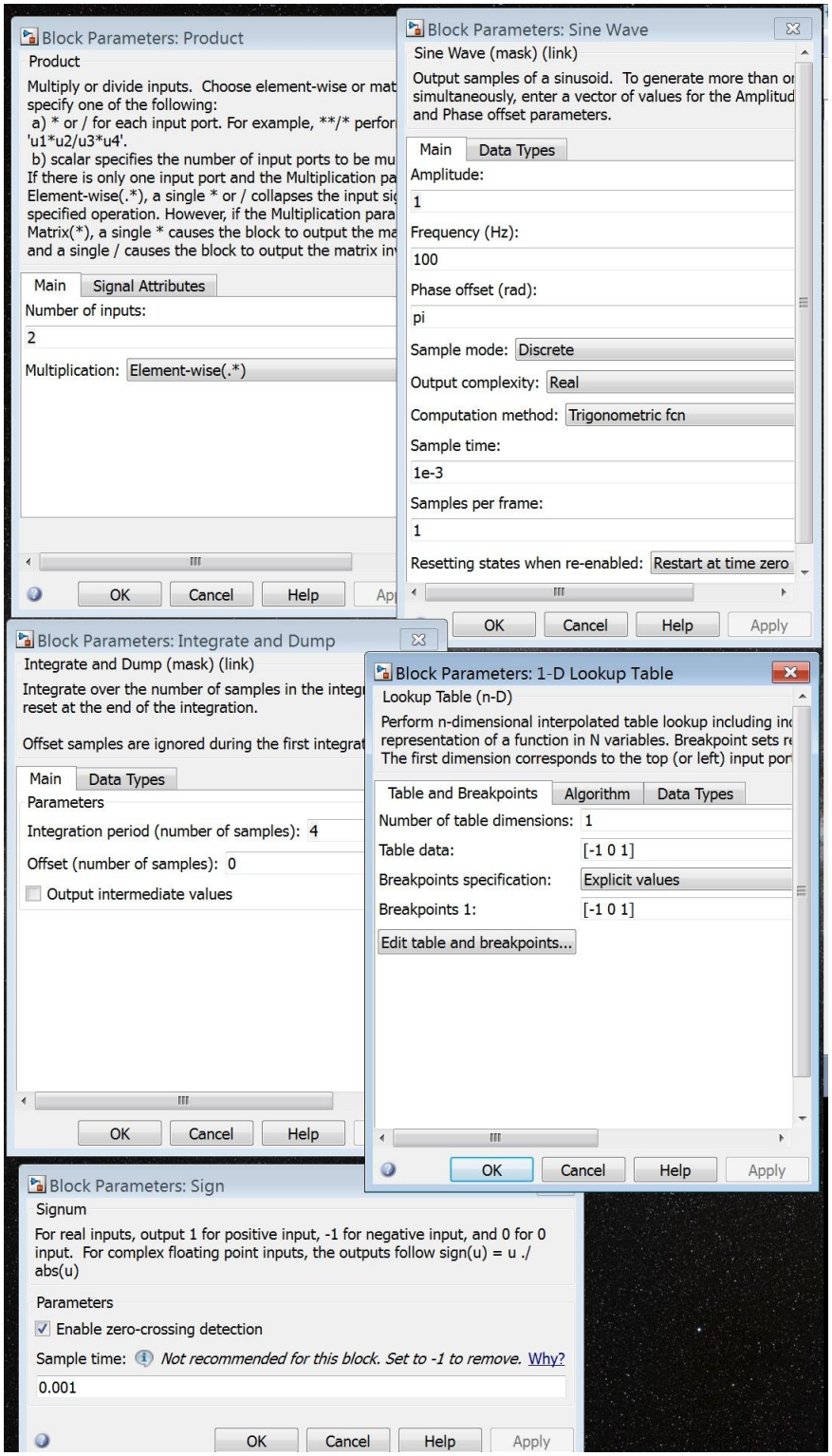




Conclusion

Mathematical analysis and simulations using Simulink tools show that the PSK based digital modulation scheme decreases with escalating values of Eb/No. We can see that theoretical values are in agreement with practical data; which was the goal. So if the data rate is increased this would also increase the SNR, however the increasing bit rate in bits/second will **also** cause more noise.

It is convenient to have a script with the variables from the Simulink available to manipulate, the process becomes more practical and convenient.


Block Parameters: Product
Product
Multiply or divide inputs. Choose element-wise or mat
specify one of the following:
a) * or / for each input port. For example, **/* perfor
'u1*u2/u3*u4'.
b) scalar specifies the number of input ports to be mu
If there is only one input port and the Multiplication pa
Element-wise(.*), a single * or / collapses the input si
specified operation. However, if the Multiplication para
Matrix(*), a single * causes the block to output the ma
and a single / causes the block to output the matrix in
Main
Signal Attributes
Number of inputs:
2
Multiplication: Element-wise(.*)
OK
Cancel
Help
Block Parameters: Sine Wave
Sine Wave (mask) (link)
Output samples of a sinusoid. To generate more than o
simultaneously, enter a vector of values for the Amplitud
and Phase offset parameters.
Main
Data Types
Amplitude:
1
Frequency (Hz):
100
Phase offset (rad):
pi
Sample mode: Discrete
Output complexity: Real
Computation method: Trigonometric fcn
Sample time:
1e-3
Samples per frame:
1
Resetting states when re-enabled: Restart at time zero
OK
Cancel
Help
Apply
Block Parameters: Integrate and Dump
Integrate and Dump (mask) (link)
Integrate over the number of samples in the integ
reset at the end of the integration.
Offset samples are ignored during the first integrat
Main
Data Types
Parameters
Integration period (number of samples): 4
Offset (number of samples): 0
Output intermediate values
OK
Cancel
Help
Block Parameters: 1-D Lookup Table
Lookup Table (n-D)
Perform n-dimensional interpolated table lookup including in
representation of a function in N variables. Breakpoint sets re
The first dimension corresponds to the top (or left) input por
Table and Breakpoints
Algorithm
Data Types
Number of table dimensions: 1
Table data: [-1 0 1]
Breakpoints specification: Explicit values
Breakpoints 1: [-1 0 1]
Edit table and breakpoints...
OK
Cancel
Help
Apply
Block Parameters: Sign
Signum
For real inputs, output 1 for positive input, -1 for negative input, and 0 for 0
input. For complex floating point inputs, the outputs follow sign(u) = u ./
abs(u)
Parameters
Enable zero-crossing detection
Sample time: Not recommended for this block. Set to -1 to remove. Why?
0.001
OK
Cancel
Help
Apply

Code:

```
clear all; clc; close all;
% ***********************************************
mdl = 'BPSK_Model'; % Simulink Model Name
% ***********************************************
```

Open up the simulation and initialize output vars

```
open_system(mdl);      % Open the Simulink Model, if not already open
G_vals = 1:10;         % 1st part -- 1 to 10
N0_vals = 1:20;        % 2nd part -- 1 to 20
n = length(G_vals);    % For part 1 of simulation
E0N0 = zeros(1,n);     % Create Array
Pb_theory = zeros(1,n);
Pb_prac = zeros(1,n);
```

Part 1: Loop and simulate for n different Gain values with N0 = 10

```
for i = 1:n
    N_0 = Simulink.Parameter(10);       % Variable for setting the Noise
    G = Simulink.Parameter(G_vals(i)); % Set block Gain to this variable
    % Now to setup parameters for the simulation
    paramNameValStruct.SimulationMode = 'norm';
    paramNameValStruct.AbsTol         = '1e-5';
    paramNameValStruct.SaveState      = 'on';
    paramNameValStruct.StateSaveName  = 'xoutNew';
    paramNameValStruct.SaveOutput     = 'on';
    paramNameValStruct.OutputSaveName = 'yout';
    paramNameValStruct.StopTime = '50';
    paramNameValStruct.ZeroCross = 'on';
    paramNameValStruct.Solver = 'ode45';
    % Run simulation and store the simulation output results to 'simout'
    simout = sim(mdl,paramNameValStruct);
    E0 = G.Value^2/2; % Theoretical value of E_0
    N0 = N_0.Value;   % The value of N_0 -- 10
    E0N0(i) = E0/N0;  % Store the theoretical value of E_0/N_0 to the array
    Pb_theory(i) = qfunc(sqrt(2*E0/N0)); % Calc/Store Theoretical Pb
    Pb_prac(i) = simout.get('Pb').signals.values(end,1); % Get the final
                                        value of Pb from the simulation
end
```

```
figure(1)
plot(10*log10(E0N0),log10(Pb_theory),'b',10*log10(E0N0),log10(Pb_prac,'r'));
legend('Theoretica value of Pb','Practical value of Pb');
xlabel('{{E_0 / N_0}}');
ylabel('P_b');
title('P_b against E_0 / N_0 for N_0 = 10 and 10 different values of Gain');
grid;
```

Part 2: Loop and simulate through 20 different N0 values for Gain=5

```
n = length(N0_vals);
E0N0 = zeros(1,n);
Pb_theory = zeros(1,n);
Pb_prac = zeros(1,n);
for i = 1:n
    N_0 = Simulink.Parameter(N0_vals(i));
    G = Simulink.Parameter(5);
    paramNameValStruct.SimulationMode = 'norm';
    paramNameValStruct.AbsTol         = '1e-5';
    paramNameValStruct.SaveState      = 'on';
    paramNameValStruct.StateSaveName  = 'xoutNew';
    paramNameValStruct.SaveOutput     = 'on';
    paramNameValStruct.OutputSaveName = 'yout';
    paramNameValStruct.StopTime = '50';
    paramNameValStruct.ZeroCross = 'on';
    paramNameValStruct.Solver = 'ode45';
    simout = sim(mdl,paramNameValStruct);
    E0 = G.Value^2/2;
    N0 = N_0.Value;
    E0N0(i) = E0/N0;
    Pb_theory(i) = qfunc(sqrt(2*E0/N0));
    Pb_prac(i) = simout.get('Pb').signals.values(end,1);
end
```

Plot the data obtained

```
figure(2)
plot(10*log10(E0N0),log10(Pb_theory),'b',10*log10(E0N0),log10(Pb_prac,'r'));
legend('Theoretica value of Pb','Practical value of Pb');
xlabel('{{E_0 / N_0}}');
ylabel('P_b');
title('P_b against E_0 / N_0 for Gain = 5 and 10 different values of N_0');
grid;
```