Performance Evaluation of

Binary Communication Systems

Part I of Final Project for

ECE 475 Fall, 2006

by John O'Flaherty

Contents

Introduction

Bit error probability is a quantitative method of estimating the quality of a binary communications channel. All communications take place in an environment of some degree of noise, and the noise causes distortions or outright errors. In the case of binary communications, the definition of an error is as simple as the definition of a data bit: if the data bit is interpreted incorrectly, there was an error. Bit error probability, or BEP or BER (bit error rate), is simply the probability that a bit will be transmitted incorrectly through the channel. It is expressed as the ratio of the number of error bits to the number of test bits in a particular test. It may be evaluated analytically by characterizing the system mathematically and evaluating the resulting equations, or by creating a simulation based on the mathematical model, and running the simulation for enough cycles to get enough error bits to give a reliable probability. This method may not be practical for very low error rates, because the very large number of bits to be simulated may make computation impractical. Moore's law will make this frontier ever more remote, but it is still a consideration.

Methods

In this report, BEPs of several system types will be examined, each by both analytical and Monte Carlo methods. The analytical methods aren't purely analytical, but are actually simulations based on analytical models that are evaluated numerically. The types of systems to be analyzed include on-off keying (OOK), phase shift keying (PSK), and frequency shift keying (FSK) with coherent detection, and OOK and FSK with non-coherent detection. For brevity, further references to these systems will be by the following abbreviations, whose mappings are obvious: COHOOK, COHFSK, COHPSK, NCOOK, NCFSK. There is no PSK evaluation for non-coherent detection since PSK depends in essence on phase coherence.

The analytical tables and graphs are produced by numerical integration of the applicable formulas for a range of signal to noise ratios of 0 to 12 db, with signal amplitude assumed to be 1 (or ± 1) for all cases. For the single case of the non-coherent OOK receiver, the threshold is a variable that changes the BEP, and the optimum value changes with different signal to noise ratios; to find the attainable BEP at each SNR, the threshold level must be optimized at each SNR level. This means that no single receiver could achieve these BEPs under varying conditions unless it had an adaptive threshold.

Theory: derivations of formulas for binary PCM systems

Coherent Systems

OOK and PSK: Bandpass and baseband systems

With OOK, or on-off keying, the carrier is on for a one-bit, and off for a zero bit. With PSK, the carrier is always on, but its phase is shifted 180° between one and zero bits. The structure of the receiver is the same for both, up to the value of the threshold in the decision circuit, which is different.

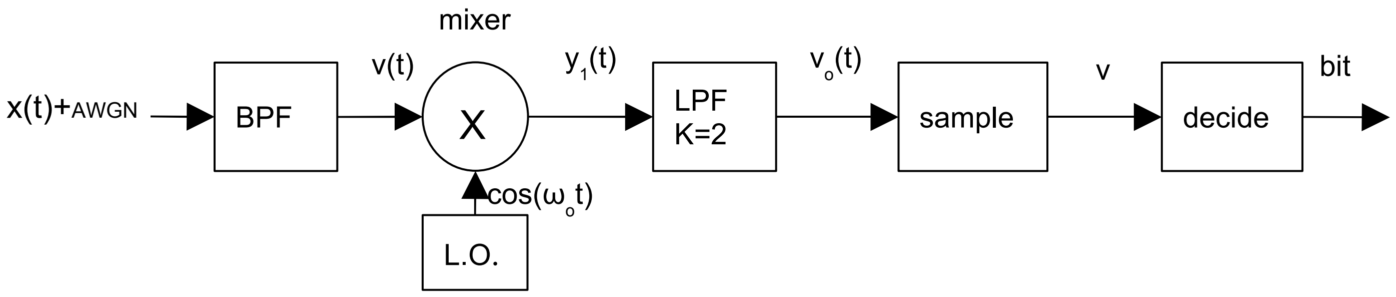


Figure 1: Block diagram for OOK and PSK coherent receivers.

This is a series of formulas that show that coherent bandpass systems can be analyzed as if they were baseband systems, if the bandpass analog systems operate perfectly. They are explained on the next page.

1. $input = x(t) + AWGN$
2. $x(t) = g(t)\cos(\omega_0 t) + n(t)$
3. $n(t) = X(t)\cos(\omega_0 t) - Y(t)\sin(\omega_0 t)$
4. $v(t) = g(t)\cos(\omega_0 t) + X(t)\cos(\omega_0 t) - Y(t)\sin(\omega_0 t)$
5. $v(t) = \left(g(t) + X(t)\right)\cos(\omega_0 t) - Y(t)\sin(\omega_0 t)$
6. $y_1(t) = v(t)\cos(\omega_0 t) = \left(g(t) + X(t)\right)\cos^2(\omega_0 t) - Y(t)\sin(\omega_0 t)\cos(\omega_0 t)$
7. $y_1(t) = \left(g(t) + X(t)\right)\frac{1}{2}\left(1 + \cos(2\omega_0 t)\right) - Y(t)\frac{1}{2}\left(\sin(2\omega_0 t)\right)$
8. $g(t) = \begin{Bmatrix} +A \\ -A \end{Bmatrix} RECT\left(\frac{t-\frac{T}{2}}{T}\right) \overset{F}{\Leftrightarrow} \begin{Bmatrix} +A \\ -A \end{Bmatrix} T\,\mathrm{sinc}\left(-\right) e^{-j\omega\frac{T}{2}}$ for PSK,

   or subsitute $\begin{Bmatrix} A \\ 0 \end{Bmatrix}$ for OOK.
9. $v_o(t) = g(t) + X(t)$

The input to the receiver's BPF is shown in equation 1. The output of the BPF is in equation 2, consisting of the signal plus noise. The last term, n(t), can be modeled as narrowband noise, as shown in equation 3. In equations 4 and 5, we collect terms to isolate the term of interest. The signal is mixed with (multiplied by) a local oscillator which is phase-locked to the transmitter, or, more realistically, to the average phase of the received signal. This multiplies the carrier already present, giving the expression in equation 6. By trigonometric identity, this is equal to equation 7. Note that equation 7 has only one baseband term, the one resulting from multiplying 1 by g(t). The signal now goes through a low-pass filter whose gain is set to 2 to compensate for the ½ term, and its cutoff frequency is set to about $2\pi/T$, the first null of the frequency domain sinc function corresponding to a pulse of width T. This eliminates the passband terms, leaving only (g(t)+X(t)), the original baseband modulating signal, plus the in-phase noise X(t), which, like the input AWGN, has a mean of zero and a variance of the same σ^2. This means that the BEP may be modeled by simply adding the anticipated channel AWGN to the original baseband signal, as if the modulation had never occurred. (This is ignoring some details like phase noise of the local oscillator, distortion in the RF stages, etc. That is, we are assuming perfect operation of all the analog components, so the BEPs to be arrived at are best-case figures.)

The recovered baseband signal has the form of equations 8 and 9, that is, the binary modulating signal plus the in-phase noise. The noise, after all the transformations, has the same mean and variance as it had at the receiver input. The sampler in the circuit above takes a single sample of the random process at the LPF output. This sample has the same statistical characteristics as the whole process. The decision circuit must decide, based on the sample voltage which includes instantaneous noise, whether a one or zero was sent.

The distribution of voltages seen at the decision circuit input has the form shown in figure 2. The two normal PDFs seen are the distributions of receiver voltage for zero and one inputs, left to right. The center voltages are ±A, and the PDFs are the characteristic of the noise, with its mean shifted by the signal amplitudes. The decision voltage level is shown as d, which will be zero for a system with amplitudes of ±A and equal probability of ones and zeros. As can be seen from the figure, for each input to the system, zero or one, there are values of noise that will cause the bit to be misinterpreted. These are the shaded areas of the tails of the PDFs. We can estimate BEP by integrating the areas of these tails and multiplying the results by the probability of one or zero.

10. $Pe = P[0]\int_{d}^{\infty} fV(v|0)dv + P[1]\int_{-\infty}^{d} fV(v|1)dv$

The expression *fV(v)* is the PDF of the received signal plus noise. For purposes of this project, P[0] and P[1] are taken as ½ each. For the PSK system, *d* is optimum at zero volts, centered between the plus and minus amplitudes. The formula reduces to the familiar normal distribution functions, and for the PSK system, becomes

11. $Pe = Q\left(\frac{A}{\sigma}\right)$

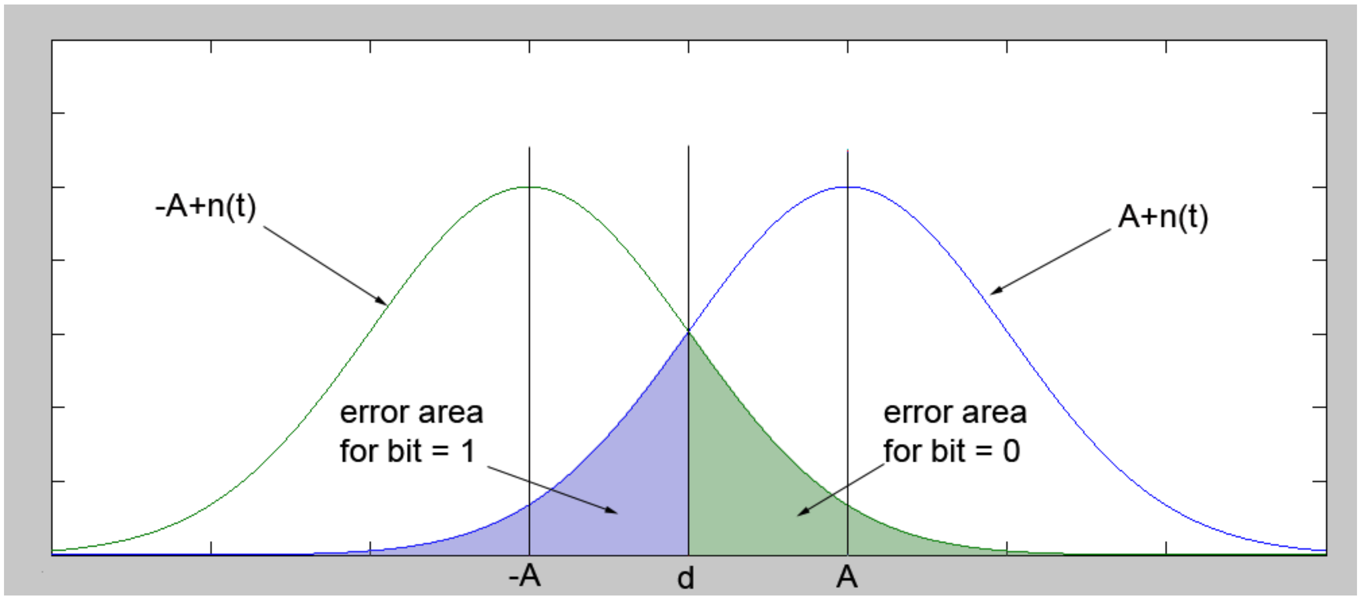


Figure 2. Distribution of decision circuit input voltage.

A similar diagram applies to the OOK system, except the –A becomes zero volts, and optimum d becomes A/2. The probability of error for this system reduces to

12. $Pe = Q\left(\frac{A}{2\sigma}\right)$

which shows that the BEP (or Pe) will be higher for an OOK system than for a PSK system. These are the analytical formulas to be used for BEP calculations for these two systems.

FSK system

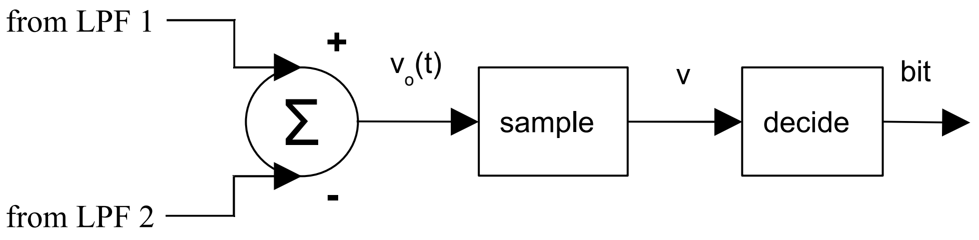


Figure 3: detector diagram for FSK coherent receiver.

13. $g(t) = \begin{Bmatrix} +A \\ -A \end{Bmatrix} + (X_1 - X_2) = \begin{Bmatrix} +A \\ -A \end{Bmatrix} + X_3$

14. $E[X_3{}^2] = E[(X_1 - X_2)^2] = E[X_1{}^2 - 2X_1X_2 + X_2{}^2]$

15. $= E[X_1{}^2] - 2E[X_1X_2] + E[X_2{}^2] = E[X_1{}^2] + E[X_2{}^2] = 2\sigma^2$

The FSK coherent receiver is the same as two PSK coherent receivers, at two slightly different frequencies, one for a one bit and one for a zero bit. This is true up to the output of the low-pass filters. The part that follows the LPFs is shown in figure 3. Here the signals are subtracted, which forms ±*A*, and subtracts the noises from the two channels, (equation 13) which, since they are uncorrelated, is the same as adding them. The means of the noise in both channels are zero, so the output noise has zero mean as well. The variances for the two channels are of equal magnitude but uncorrelated. This means the cross product term in the variance equation disappears, so as equations 11 and 12 show, the result is just a signal with twice the variance of the input AWGN. The upshot of this is that the FSK system output has the same mathematical form as that of the PSK system, except that the variance, or power, of the noise is doubled. The same decision curves apply as for PSK, and the same analysis applies except that the BEP formula is now

16. $Pe = Q\left(\frac{A}{\sqrt{2}\sigma}\right)$

due to the doubling of the noise power at the adder. This means that the FSK system has worse noise performance than PSK, but better than OOK. The reason to use FSK rather than PSK in some cases is that PSK may be impossible due to the phase distortion in a multi-path situation. Both FSK and PSK have the advantage of being constant power systems, which don't have to handle the contrast between full and zero power that occurs in an OOK system.

Non-coherent Systems

OOK

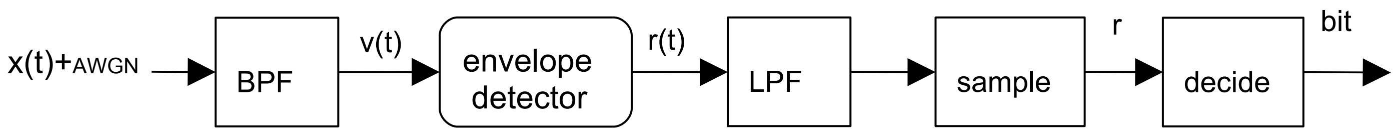


Figure 3: Block diagram for OOK non-coherent receiver.

The received signal distribution for the non-coherent OOK system has the same general form as that for the coherent system, and the BEP formula is the same, except that the distribution curves are quite different, since there is no local carrier to create the baseband signal. This must be created by an envelope detector, which is basically an absolute value circuit followed by a low pass filter. Such a detector can't distinguish between signal envelope and noise envelope.

The applicable formulas when the input bit is zero (transmitter is off),are these:

17. $v(t) = n(t) = X(t)\cos(\omega_0 t) - Y(t)\sin(\omega_0 t)$

18. $\sqrt{X(t)^2 + Y(t)^2}\cos(\omega_0 t + \theta(t))$

19. $r = \sqrt{X(t)^2 + Y(t)^2}$

20. $fR(r) = \dfrac{r \times e^{\frac{-r^2}{2\sigma^2}}}{\sigma^2}, \quad r \geq 0$

The input signal voltage consists only of noise. The envelope detector discards phase information, and the low pass filter discards the rest of the high frequency information, leaving the expression for *r* in equation 19. Although X(t) and Y(t) are Gaussian, their squared sum is not a linear combination of them, and its distribution is not Gaussian, but Rayleigh (equation 20).

When the input is one, the detected signal voltage distribution has *A* added, as seen in equation 21. This converts the distribution into a Rician distribution, as seen in equation 22. The Rayleigh distribution is the same as the Rician with A set to zero. The I_0 term is the modified Bessel function of the first kind of order zero. The form of the received signal is seen in figure 4.

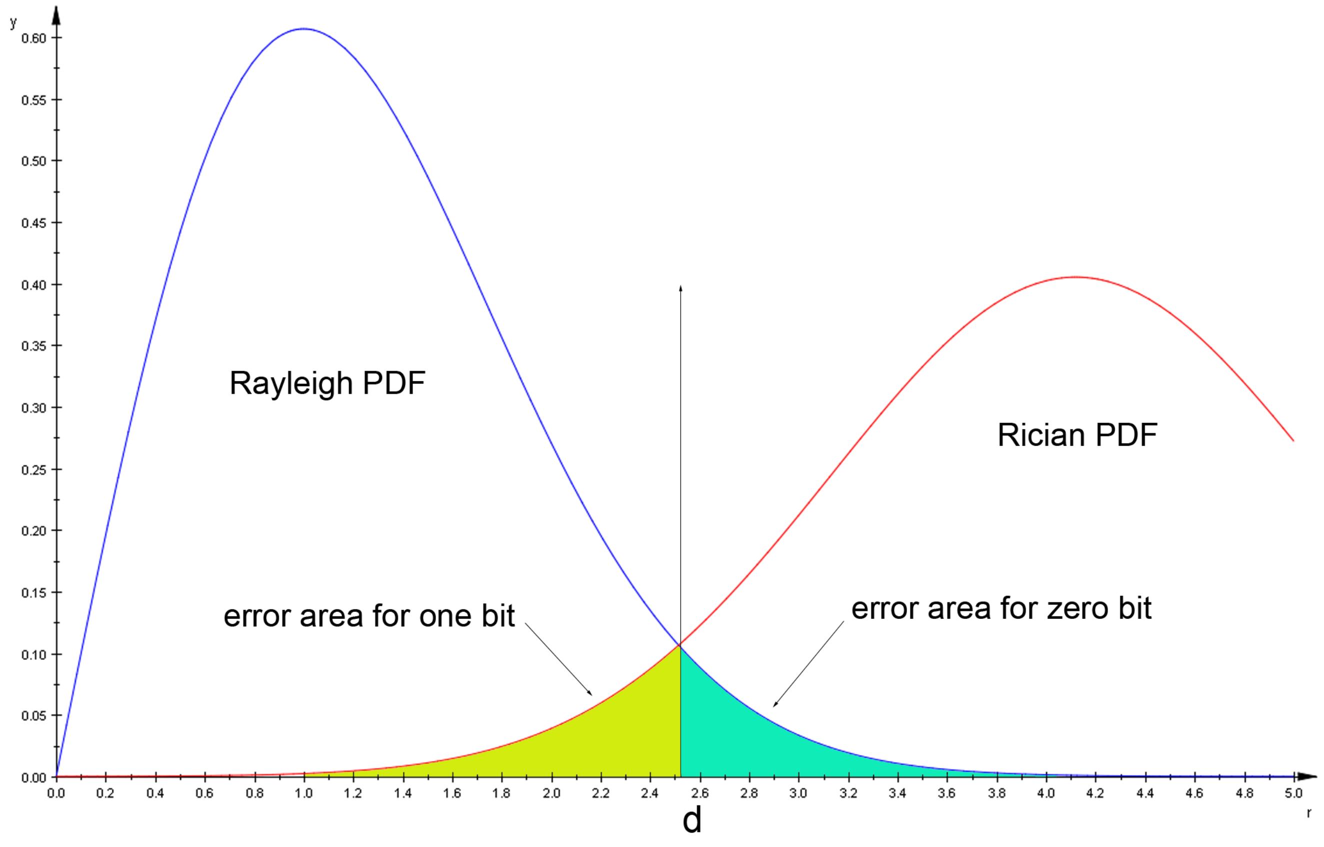


Figure 4

21. $r = \sqrt{\left(A + X(t)^2\right) + Y(t)^2}$

22. $fR(r) = \dfrac{r \times e^{\left(\frac{-\left(r^2 + A^2\right)}{2\sigma^2}\right)}}{\sigma^2} \times I_0\left(\dfrac{rA}{\sigma^2}\right), \quad r \geq 0$

The result is that the same global analytical formula is used to evaluate BEP, but with a Rayleigh PDF on the zero side and a Rician PDF on the one side.

PSK is not considered for non-coherent detection because phase information is by definition unavailable in a non-coherent system.

FSK

The non-coherent FSK system doubles the recipe for the non-coherent OOK system, with a detector like the FSK subtractor in the coherent FSK system, producing r_1 and r_2 instead of just r. This subtractor makes the output bit equal one when $r_1 > r_2$, and zero otherwise, so no other threshold is in effect. However, the noise and signals are combined in such a way that a joint probability distribution must be used to calculate the error probabilities. The formulas are

23. $Pe\,|\,1 = P[r_2 > r_1] = \int_0^\infty \int_{r_1}^\infty f(r_1, r_2)\,dr_2\,dr_1$

24. $f(r_1, r_2) = f(r_1)\,f(r_2)$

25. $Pe = \frac{1}{2} e^{\frac{-A^2}{4\sigma^2}}$

Formula 23 shows the BEP due to the joint probability distribution for a 1 bit transmitted. The joint PDF in the double integral is equal to the product of the Rician and Rayleigh PDFs because the variables r_1 and r_2 are independent. This expression simplifies to that in formula 25. By the symmetry of the receiver structure, the error probabilities for both one and zero are the same, so the same formula applies to the entire system.

Programs to determine BEP

Analytical

Matlab programs were written to analytically determine the bit error rates of the following types of system: COHOOK, COHFSK and COHPSK, and NCOOK and NCFSK. The programs in all but the last case work by integration of the probability curve error tails shown in the theoretical diagrams above, rather than by using summary formulas. The results were compared at the 8 db point to the summary formula results to be sure that the program was operating accurately enough, however. The non-coherent FSK system, which requires a double integration of the joint Rician-Rayleigh probability distribution, proved too difficult to implement with `trapz`, so the summary formula was used for the Matlab table and graph for this system. The actual double integration was done in a computer algebra system (CAS), for all the required SNR levels, and the results were identical to those from the summary formula given in class. The output of the CAS program for the non-coherent FSK system is included as Appendix F.

The programs for analytical BEP were divided into two parts, those for non-coherent systems, and a function program for coherent systems called by the first program. This was done so all the graphs would be plotted together, giving a basis for comparison. The two programs appear as Appendix C and Appendix D, respectively.

In general, the analytical routines divide the SNR axis linearly into 1200 parts, to assure a smooth graph. The SNR powers are calculated into another vector, so all plotting can be done against the vector that is linear in SNR, and everything can be ordered by a linear SNR. For all cases, signal amplitude is taken to be 1 V, providing 1 W into 1 Ω. The modulation of SNR is done by changing the noise power. The exception in SNR resolution is the non-coherent OOK determination.

NCOOK

This is the worst-performing and the most difficult to analyze, since the optimum threshold varies with SNR. To keep execution time reasonable, the NCOOK is done only at integer db SNRs, for a total of 13 determinations (0-12). Though the program output only requires steps of two, it was convenient to do steps of one and skip every other one when making the printout. The integration of the error tails in this system requires Rayleigh (for zeros) and Rician (for ones) PDFs. A single function was written to provide these; it appears at the end of the program in Appendix C. Calling it with the amplitude parameter set to zero makes it Rayleigh instead of Rician.

To demonstrate the change in optimum threshold, a graph of changing BEP for different threshold values was created, and the optimum threshold found was marked for each curve on this graph. The threshold value vector over the SNR range was saved for use in the Monte Carlo simulations, where a separate determination of optimum threshold was not made.

The graph for BEP vs. SNR is started in this program, using the `semilogy` function since the abscissa is already in db. Here the BEPs found are printed out for 0 to 12 db in steps of two as required, along with the associated threshold voltage.

At this point in the program, it was found convenient to call the program for coherent BEPs, to get all curves on the same graph.

NCFSK

This program section uses the 1200 part SNR axis. As mentioned above, the summary formula was used instead of `trapz` double integration, because the latter proved too difficult in terms of getting the matrix sizes and indexes correct. The CAS results for double integration are in Appendix F, and they agree perfectly with those from the summary formula. The graph and table for this part work as for NCOOK, except that no threshold value is printed, since it is zero.

COHOOK

The coherent program appears as Appendix D. It uses `trapz` integration of the Gaussian error tail. It takes advantage of the symmetry available for coherent systems to integrate only one side; it uses a threshold of 0.5, half of the amplitude $A = 1$.

COHFSK

This section uses the same integration as for coherent PSK, but the noise power is doubled, reflecting the subtraction that occurs before the decision circuit. Since the noise in the two sections of the receiver is taken as uncorrelated, a subtraction is the same as an addition, and as derived above, the variance or noise power doubles.

COHPSK

This section uses the same integration as NCOOK, but the amplitude is double to $\pm A$, and the threshold is zero.

For all sections, the integration range is from the appropriate threshold value to 10 V on the receiver input. With a signal power/variance of 1, the signal standard deviation σ is 1, and 10 V amounts to 10σ. The normal PDF at 10σ amounts to 10^{-22}, so this level is equivalent to positive infinity for all practical purposes. The voltage vector used for integration is of size 2000, going from 0 in steps of .005 to 10. It was found that this provided quite enough accuracy in the result to agree with the analytical formulas to the fifth or sixth decimal place.

Since all tables in this part use the same format, a function appearing at the end of the program is use to print them.

Monte Carlo

The Monte Carlo section uses code to simulate the operation of the detector circuits. It works by creating signal and noise vectors of 1s and 0s, and adding Gaussian noise vectors to them. The Gaussian noise vectors are produced by the instruction `wgn`(). Successive invocations of this function produce uncorrelated noise vectors, which is a requirement for several sections of the program. All sections have in common that the BEP is estimated by comparing the signal-plus-noise vectors to a threshold value, and finding the percentage of bits that are in error. The SNR axis is divided only into 120 parts for this program, to keep execution time reasonable. The number of test bits, `nBits`, is a variable for the program. The final value used for the number of bits was 10^6 , except for the COHPSK detector, where the low BEP justified using 10^7. Since the program runs pretty quickly, the required number of bits wasn't estimated by formula, but was determined empirically by the smoothness of the error curves produced. The program automatically multiplies the selected number of bits by ten for the COHPSK system. The specific details for each detector follow.

COHOOK

This detector creates separate signal plus noise vectors for 1s and 0s, and uses a threshold of 0.5 V. Only a single noise vector is needed since the 1s and 0s occur at different times.

COHFSK

This detector uses two noise vectors and four signal vectors to simulate two independent channels, where the noise is the same for 1s and 0s in a particular channel, but different across channels. The resulting signal plus noise vectors are subtracted, and the decision threshold of zero is applied to arrive at BEP.

COHPSK

This system is like the COHOOK, except that the threshold is zero and the 1s vector is plus ones, while the 0s vector is at minus one.

NCOOK

This detector creates vectors for 1s by adding noise to a signal vector of 1s, and combining this with an independent noise vector, and then finding the square root of the sum of the squares, as an envelope detector would do. The zeros vector is produced by doing the same to two independent noise vectors, with no signal added. For this section, the threshold is set separately at each SNR level by using the threshold levels determined in the analytical integration section. This may give a better Monte Carlo estimation of BEP than if the threshold itself were determined by Monte Carlo, but in fact it can't make very much difference; since the thresholds are at minimums, the slope of the BEP vs. threshold curve is zero at that point, and a slight change in threshold can't produce much change in BEP.

NCFSK

This type uses four independent noise vectors to simulate the operation of two independent NCOOK channels, each with I & Q noise, which will be subtracted from each other as in COHFSK. Advantage is taken of the symmetry of the system to have channel 1 carry only 1s, and channel 2 only 0s.

Summary and results.

The programs all worked well, and the agreement between the analytical and Monte Carlo results was satisfying for both the numerical results and the graphs. Figure 5 is the graph of BEP for all five system types produced by integration of the analytical formulas, except as noted previously.

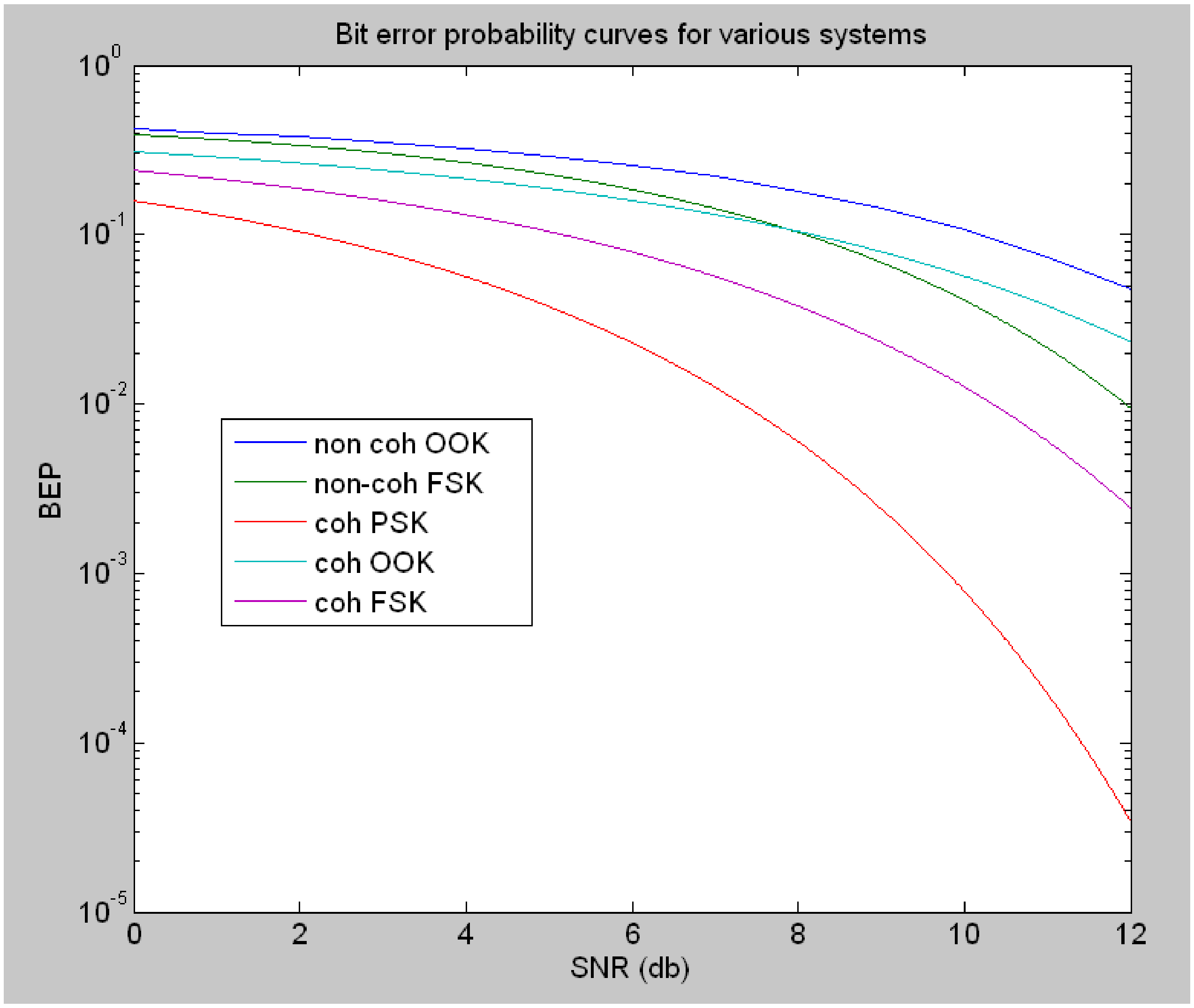


Figure 5

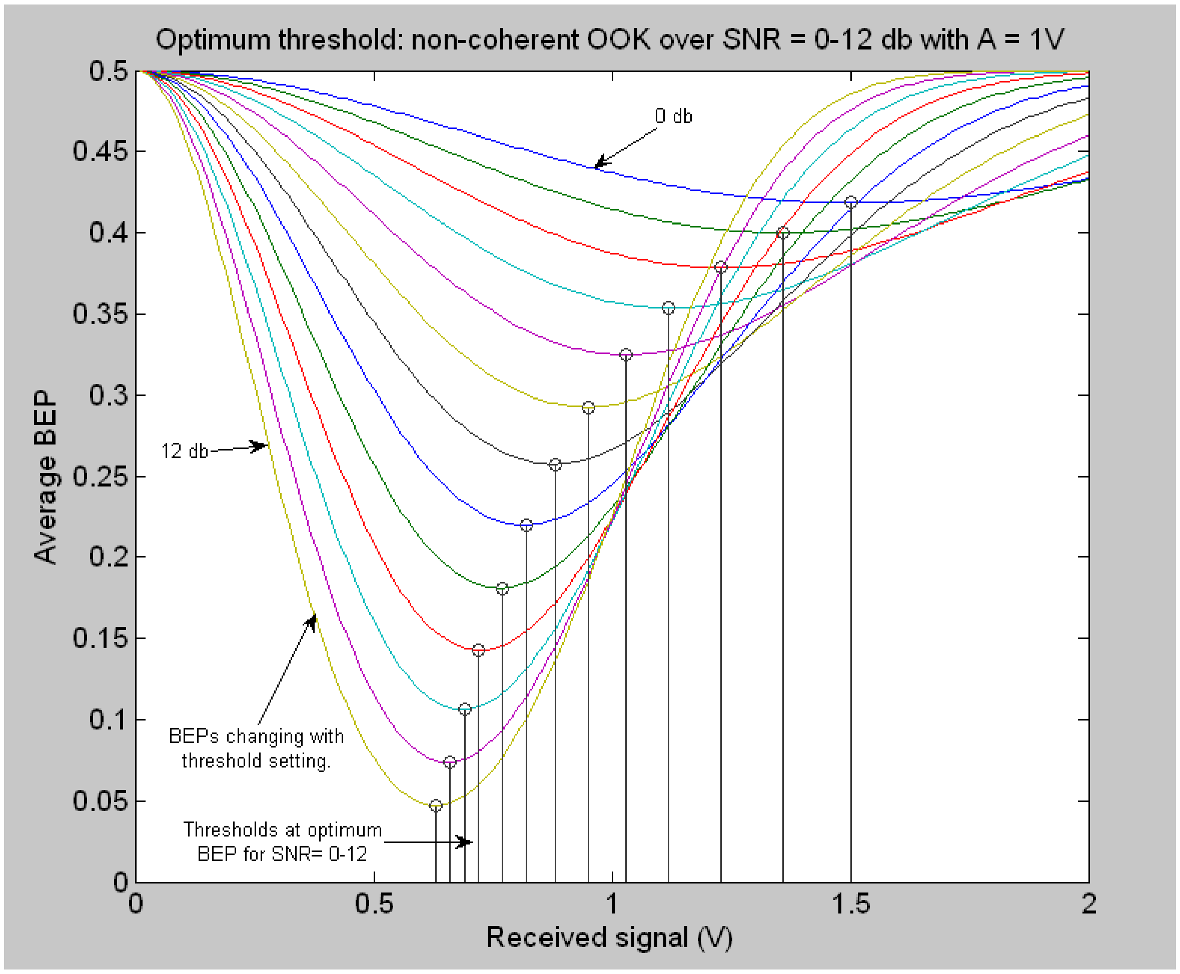


Figure 6

Figure 6 is a family of curves of optimum thresholds vs. BEPs as SNR is changed from 0 to 12 db. This graph was interesting, as it seems to progress towards a 0.5 V optimum threshold for very high SNRs.

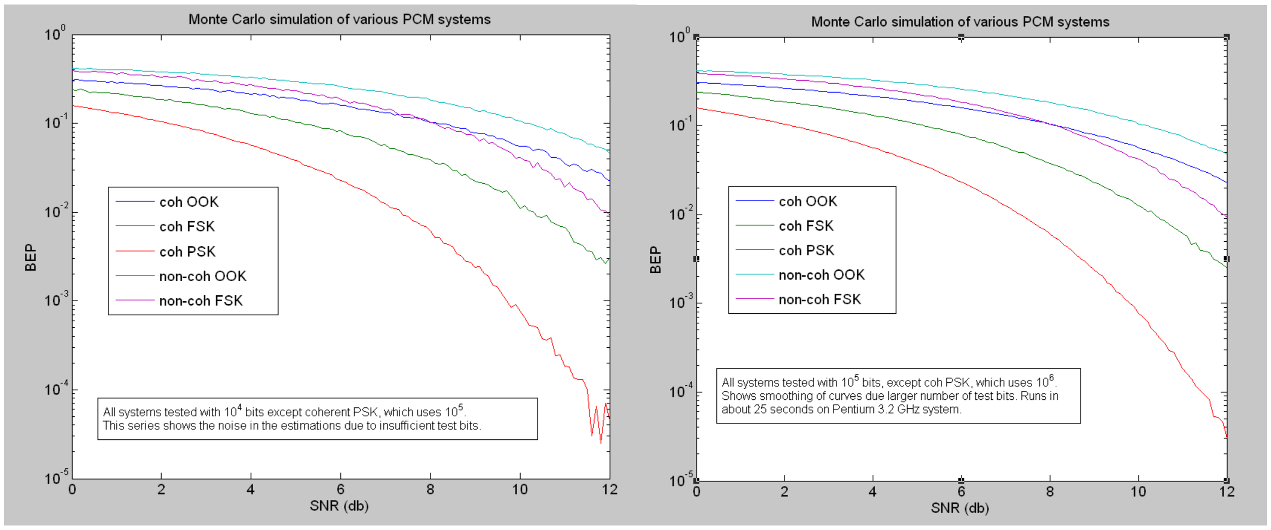


Figure 7

Figure 8 shows the final Monte Carlo BEP curves, done with 10^6 bits per graph point, except for COHPSK, which used 10^7. These graphs are barely distinguishable from the analytical graphs, except for a very little bumpiness. To see the effect of smaller numbers of test bits, see the two graphs of Figure 7, which show simulations with 10 and 100 times fewer bits.

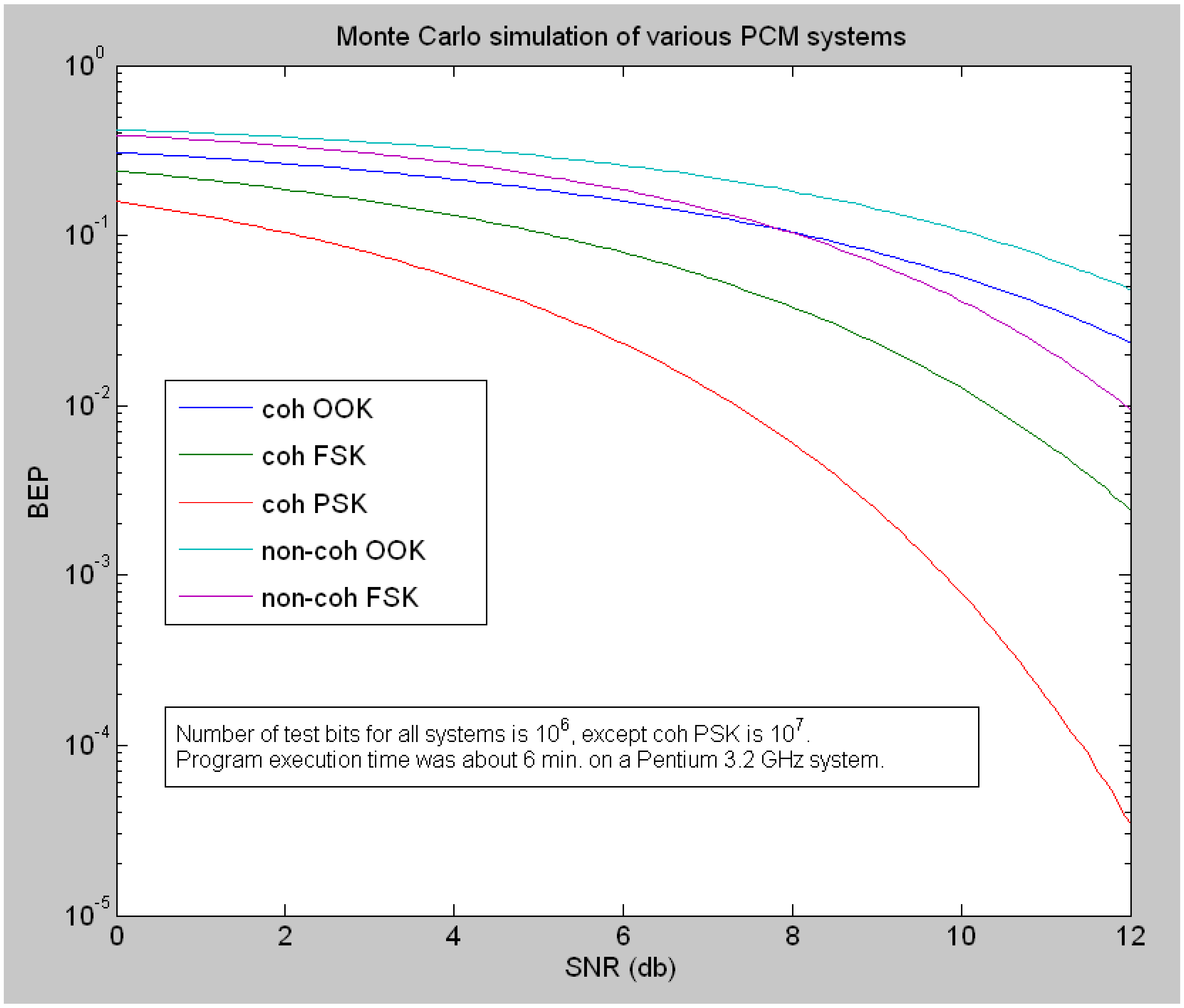


Figure 8

Conclusions

The table results are in Appendix A on the next page. Table 1 shows analytically calculated BEPs, and Table 2 shows BEPs measured by Monte Carlo simulation. Table 3 compares the results from tables 1 and 2, showing the close agreement.

This close agreement between the analytical and Monte Carlo results, provided enough bits are used in the latter, shows that Monte Carlo is an excellent tool to analyze systems. It will be especially useful for systems that can't be analyzed mathematically because of non-linearity or other problems, but for which a computer simulation model can be constructed.

Appendix A

Tabular results

Analytical BEP values for various systems as SNR varies from 0 - 12 db

SNR	Coherent systems			Non-coherent systems		
(db)	PSK	OOK	FSK	FSK	OOK	@ d =
0	0.1586558	0.3085379	0.2397503	0.389400	0.418305	1.50
2	0.1040294	0.2645238	0.1866811	0.336428	0.378533	1.23
4	0.0564962	0.2140514	0.1312108	0.266837	0.324802	1.03
6	0.0230080	0.1592311	0.0791433	0.184812	0.257404	0.88
8	0.0060049	0.1045713	0.0378533	0.103256	0.181082	0.77
10	0.0007829	0.0569269	0.0126744	0.041042	0.106298	0.69
12	0.0000343	0.0232696	0.0024389	0.009510	0.047186	0.63

Table 1

Monte Carlo BEP values for various systems as SNR varies from 0 - 12 db

SNR	Coherent systems			Non-coherent systems		
(db)	PSK	OOK	FSK	FSK	OOK	@ d =
0	0.1586029	0.3085685	0.2398385	0.3890200	0.4180085	1.50
2	0.1038738	0.2646030	0.1868370	0.3365900	0.3786275	1.23
4	0.0564300	0.2137940	0.1312165	0.2667520	0.3247245	1.03
6	0.0229811	0.1593740	0.0791470	0.1847780	0.2578755	0.88
8	0.0059928	0.1047610	0.0377825	0.1033010	0.1810810	0.77
10	0.0007785	0.0572420	0.0127145	0.0406410	0.1062330	0.69
12	0.0000346	0.0231800	0.0024095	0.0094270	0.0470710	0.63

Table 2

Fractional differences: analytical and Monte Carlo determinations: (parts per thousand or tenths of 1%)				
0.33	-0.10	-0.37	0.98	0.71
1.50	-0.30	-0.84	-0.48	-0.25
1.17	1.20	-0.04	0.32	0.24
1.17	-0.90	-0.05	0.18	-1.83
2.02	-1.81	1.87	-0.44	0.01
5.62	-5.54	-3.16	9.77	0.61
-8.75	3.85	12.05	8.73	2.44

Table 3

Note: The threshold values for the Monte Carlo simulations of non-coherent FSK systems are those determined in the analytical calculations, and were not determined independently in Monte Carlo. This provides a more reliable threshold value for the Monte Carlo BEPs.

Appendix B

Output from Matlab screen for the BEP programs.

Analytical:

```
Coherent OOK system
SNR   BEP
0   0.3085379
2   0.2645238
4   0.2140514
6   0.1592311
8   0.1045713
10   0.0569269
12   0.0232696

Coherent FSK system
SNR   BEP
0   0.2397503
2   0.1866811
4   0.1312108
6   0.0791433
8   0.0378533
10   0.0126744
12   0.0024389

Coherent PSK system
SNR   BEP
0   0.1586558
2   0.1040294
4   0.0564962
6   0.0230080
8   0.0060049
10   0.0007829
12   0.0000343

Non-coherent OOK system
SNR   BEP       threshold
0   0.418305      1.50
2   0.378533      1.23
4   0.324802      1.03
6   0.257404      0.88
8   0.181082      0.77
10   0.106298      0.69
12   0.047186      0.63

Non-coherent FSK system
SNR   BEP
0   0.389400
2   0.336428
4   0.266837
6   0.184812
8   0.103256
10   0.041042
12   0.009510

execution time: 9.7 sec.
```

Monte Carlo:

```
Coherent OOK
SNR   BEP
0   0.3085685
2   0.2646030
4   0.2137940
6   0.1593740
8   0.1047610
10   0.0572420
12   0.0231800

Coherent FSK
SNR   BEP
0   0.2398385
2   0.1868370
4   0.1312165
6   0.0791470
8   0.0377825
10   0.0127145
12   0.0024095

Coherent PSK
SNR   BEP
0   0.1586029
2   0.1038738
4   0.0564300
6   0.0229811
8   0.0059928
10   0.0007785
12   0.0000346

Non-coherent OOK
SNR   BEP
0   0.4180085
2   0.3786275
4   0.3247245
6   0.2578755
8   0.1810810
10   0.1062330
12   0.0470710

Non-coherent FSK
SNR   BEP
0   0.3890200
2   0.3365900
4   0.2667520
6   0.1847780
8   0.1033010
10   0.0406410
12   0.0094270

execution time: 259.3 sec.
```

Appendix C Non-coherent analytical program

```
function ncoh
% Program to plot and make a table of BEP for non-coherent FSK and OOK
%  systems by the analytical formula. This program finds the optimum threshold
%  for the OOK system at SNRs from 0 to 12 db in steps of two, and supplies
%  the BEP at those optimum values.
% Note: SNR is (A^2 / sigma^2) in all that follows,
%  and A is set to 1 volt, so the SNR is adjusted by modifying the
%  noise power or variance (sigma^2).
% The common assumption is that P[0] = P[1] = 1/2.

% Set up common parts of program- begin with a vector of SNR in db,
%  to provide a convenient index for graph and table.

strt = cputime;
SNRdb = 0:12;                % Use a zero to 12db snr, unit steps.
SNR = 10 .^ (SNRdb / 10);    % Precalculate the power ratio for 0-12 db
sigmaSQ = 1 ./ SNR;          % Get the noise power = the variance.
                             % The numerator A is taken as a constant 1.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% BEP vs SNR for NON-COHERENT OOK
BEPmin = ones(1, length(SNR));
dMin = ones(1, length(SNR));
for k = 1:length(SNR);
dLim = 2; % highest threshold d to try.
res = 100; % resolution of d search
  BEP=ones(1, dLim * res);
  dd = ones(1,length(BEP));
for index=1 : dLim * res
    d = index / res;
    dd(index)=d;
    r1 = 0 : .01 : d;  % integration vector for Rician PDF fr(r|1)
    r2 = d : .01 : 20; % integration vector for Rayleigh PDF fr(r|0)
    s = sigmaSQ(k);
    % function RicPDF(s,A,r) s=noise variance, A = sig ampl, r = rcvr volt.
    %                     fr(r|1)= Rician                    fr(r|0)= Rayleigh
     BEP(index) = 0.5 * (trapz(r1,RicPDF(s,1,r1))) + 0.5 * (trapz(r2,RicPDF(s,0,r2)));
     if BEP(index) < BEPmin(k)
        BEPmin(k) = BEP(index);
        dMin(k) = d;
     end
end
save('dMin','dMin');
plot(dd,BEP) % For each threshold search, plot the BEP curve found.
hold all     % all 13 (0-12db) on the same graph.
end
pltVec = 0 : dLim / (length(SNR) - 1) : dLim;
stem(dMin,BEPmin) % Gaudify the graph with optimum threshold markers.
title('Optimum threshold: non-coherent OOK over SNR = 0-12 db with A = 1V')
xlabel('Received signal (V)');
ylabel('Average BEP');
hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure

prj475coh;  % Run all the coherents now to get them in the same graph

semilogy(SNRdb, BEPmin);
hold all

disp('Non-coherent OOK system')
disp('SNR   BEP       threshold')
for cnt=0:2:12
 disp(sprintf('%d   %.6f      %.2f',cnt,BEPmin(cnt+1),dMin(cnt+1)))
end
disp(' ')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NON-COHERENT FSK
% The non-coherent FSK BEP is done by reduced formula (not by double integration of
%  the joint PDFs - this proved too difficult for me in Matlab using trapz,
%  so that task was moved to a CAS system).

SNRdb = 0 : .01 : 12;            % Use a zero to 12db snr, resolve to 1201 different steps.
SNR = 10 .^ (SNRdb / 10);        % Precalculate the power ratio for 0-12 db
sigmaSQ = 1 ./ SNR;              % Get the noise power = the variance.
                                 % The numerator A is taken as a constant 1.


BEP = zeros(1,length(SNR));      % Declare a BEP matrix for the results.
for k = 1:length(SNR);
    BEP(k) = 1/2 * exp(-1 / (4 * sigmaSQ(k)));
end

semilogy(SNRdb, BEP);

disp('Non-coherent FSK system')
disp('SNR   BEP')
for cnt=0:2:12
 disp(sprintf('%d   %.6f', cnt, BEP(cnt * 100 + 1)))
end
disp(' ')

% Add tinsel to the graph...
ylabel('BEP'); xlabel('SNR (db)');
title('Bit error probability curves for various systems')
legend('non coh OOK','non-coh FSK','coh PSK','coh OOK','coh FSK' )

hold off

disp(sprintf('execution time: %.1f sec.', cputime - strt))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function returns the Rician or Rayleigh PDF
% as a function of s = noise power (sigma^2),
% A = signal amplitude, r = received voltage.
function rtn=RicPDF(s,A,r)
rtn=r/s.*exp(-(r.^2+A^2)/(2*s)).*(besseli(0,r*A/s));
```

Appendix D Coherent analytical program

```
function coherentBEPs
% Program to plot and make a table of BEP for coherent PSK, OOK, and FSK
%  systems by integration of the analytical formula.
% Note: SNR is (A^2 / sigma^2) in all that follows,
%  and A is set to 1 volt, so the SNR is adjusted by modifying the
%  noise power or variance (sigma^2).
% The common assumption is that P[0] = P[1] = 1/2.
% All of these systems exhibit symmetry between 1 and 0 signals
%  around the decision point. This fact and the equal probability of 0 and 1
%  allow calculation of the BEP by a single integration as if the
%  transmitted signal were always zero.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Set up common parts of program- begin with a vector of SNR in db,
%  to provide a convenient index for graph and table.

SNRdb = 0:.01:12;               % Use a zero to 12db snr, resolve to 1201 different steps.
SNR = 10 .^ (SNRdb / 10);       % Precalculate the power ratio for 0-12 db
sigmaSQ = 1 ./ SNR;             % Get the noise power or variance.
                                % The numerator A is taken as a constant 1.
BEP = zeros(1,length(SNR));     % Declare a BEP matrix for the results.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% BEP vs SNR FOR COHERENT OOK
x = 0.5 : .005 : 10;             % Use a different integration range, from A/2 = 0.5 to
for k = 1:length(SNR);          % infinity, and use a mu of zero.
 BEP(k) = 1 / sqrt(2 * pi * sigmaSQ(k)) * trapz(x, exp(-(x).^2/(2 * sigmaSQ(k))));
end
semilogy(SNRdb,BEP);
disp('Coherent OOK system')
tablePrint(BEP)
BEP8OOK = BEP(801);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% BEP vs SNR FOR COHERENT FSK
x = 0 : .005 : 10;              % Use the same integration range and mu as for PSK.
                                % The noise power doubling is reflected in the extra
for k = 1:length(SNR);          % factor of 2 wherever sigma^2 appears in the PDF.
 BEP(k) = 1 / sqrt(4 * pi * sigmaSQ(k)) * trapz(x, exp(-(x + 1).^2/(4 * sigmaSQ(k))));
end
semilogy(SNRdb, BEP);
disp('Coherent FSK system')
tablePrint(BEP)
BEP8FSK = BEP(801);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% BEP vs.SNR FOR COHERENT PSK
x = 0 : .005 : 10;               % Set up an integration vector from zero to infinity for coh. PSK
for k = 1 : length(SNR);
 BEP(k) = 1 / sqrt(2 * pi * sigmaSQ(k)) * trapz(x, exp(-(x + 1) .^2 / (2 * sigmaSQ(k))));
end
semilogy(SNRdb, BEP);           % Plot results with x axis linear in db, and the log of y
hold all                        % Use the same graph for easy comparison of systems.
disp('Coherent PSK system')
tablePrint(BEP)
BEP8PSK = BEP(801);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Compare integration results to already derived formulas at 8 db SNR:
% The formulas are for PSK, OOK and FSK respectively
% BEP = Q(A/sigma), Q(A/2sigma), Q(A/sqrt(2)sigma)
sgm = sqrt(sigmaSQ(801));
d1 = (BEP8PSK - (1-normcdf(1/sgm)))/BEP8PSK;
d2 = (BEP8OOK - (1-normcdf(1/(2*sgm))))/BEP8OOK;
d3 = (BEP8FSK - (1-normcdf(1/(sqrt(2)*sgm))))/BEP8FSK;
% The results show agreement to within 1 part in 10,000.
% This can be improved by making the integration vector x finer
%  but that slows the program, and it's accurate enough as it is.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
hold all
function tablePrint(BEP)
disp('SNR   BEP')
for cnt=0:2:12
 disp(sprintf('%d   %.7f',cnt,BEP(cnt * 100 + 1)))
end
disp(' ')
```

Appendix E Monte Carlo program

```
function monte
% Signal assumed to be 1 volt amplitude, or 1 watt into 1 ohm
% The wgn function generates white Gaussian noise assuming 1 ohm load
%  at the power level specified in the third function parameter.
global res;
strt = cputime;
res = 10;
SNRdb = 0 : 1/res : 12;
NPWR = 0: -1/res : -12; % Assuming a signal of 1 volt/watt
nBits = 1e6;
BEP = ones(1,length(SNRdb));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% COH OOK system
for k = 1 : length(SNRdb)
    thr = 0.5;               % Detector threshold voltage.
    nb = nBits;              % Number of bits in test.
    n = wgn(1, nb, NPWR(k));    % A white Gaussian noise vector.
    s1 = ones(1, nb);        % Make a vector of ones.
    s0 = zeros(1, nb);       % Make a vector of zeros.
    s1n = s1 + n;            % Add noise to the set of ones.
    s0n = s0 + n;            % Add noise to the set of zeros.
    lz = length(s1n(s1n < thr)) / length(s1n); % Calculate BEP for ones.
    l1 = length(s0n(s0n > thr)) / length(s0n); % Calculate BEP for zeros.
    BEP(k) = 0.5 * lz + 0.5 * l1 ;             % Average and record the BEP
end
semilogy(SNRdb,BEP)
hold all
disp('Coherent OOK');
tablePrint(BEP);
disp(' ');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% COH FSK system
for k = 1 : length(SNRdb)
    thr = 0.0;               % Detector threshold voltage.
    nb = nBits;              % Number of bits in test.
    n1 = wgn(1, nb, NPWR(k));    % A white Gaussian noise vector.
    n2 = wgn(1, nb, NPWR(k));    % Another independent one for the other channel.
    s11 = ones(1, nb);       % A vector of ones for ch1 sending 1
    s10 = zeros(1,nb);       % A vector of zeros for ch1 sending 0
    s21 = ones(1,nb);        % A vector of ones for ch2 sending 1
    s20 = zeros(1,nb);       % A vector of zeros for ch2 sending 0
    s11n = s11 + n1;         % Add noise to channel 1.
    s10n = s10 + n1;         %  ditto
    s21n = s21 + n2;         % Add noise to channel 2.
    s20n = s20 + n2;         %  ditto
    sx1 = s11n - s20n;         % the subtractor output for ones sent.
    sx0 = s10n - s21n;         % the subtractor output for zeros sent.
    lz = length(sx1(sx1 < thr)) / length(sx1); % Calculate BEP for ones.
    l1 = length(sx0(sx0 > thr)) / length(sx0); % Calculate BEP for zeros.
    BEP(k) = 0.5 * lz + 0.5 * l1 ;             % Average and record the BEP
end
semilogy(SNRdb,BEP)
disp('Coherent FSK');
tablePrint(BEP);
disp(' ');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% COH PSK system
for k = 1 : length(SNRdb)
    thr = 0.0;               % Detector threshold voltage.
    nb = nBits * 10;         % Number of bits in test. For PSK,
                             %  multiply # bits times 10, since only PSK goes
                             %  low enough in error rate to need that many bits.
    n = wgn(1, nb, NPWR(k)); % A white Gaussian noise vector.
    s1 = ones(1, nb);        % Make a vector of ones.
    s0 = -ones(1, nb);       % Make a vector of zeros.
    s1n = s1 + n;            % Add noise to the set of ones.
    s0n = s0 + n;            % Add noise to the set of zeros.
    lz = length(s1n(s1n < thr)) / length(s1n); % Calculate BEP for ones.
    l1 = length(s0n(s0n > thr)) / length(s0n); % Calculate BEP for zeros.
    BEP(k) = 0.5 * lz + 0.5 * l1 ;             % Average and record the BEP
```

```
end
semilogy(SNRdb,BEP)
disp('Coherent PSK');
tablePrint(BEP);
disp(' ');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NON-COH OOK system
% Use threshold set determined in the analytical OOK series
%  to be sure of getting the optimum BEP from the full Monte.
    load('dMin');
for k = 1 : 1 : length(SNRdb)
    thr = dMin(ceil(k / res)); % Detector threshold voltage.
    nb = nBits;                 % Number of bits in test.
    n1 = wgn(1, nb, NPWR(k));   % A white Gaussian noise vector.
    n2 = wgn(1, nb, NPWR(k));   %  and another, uncorrelated.
    s1 = ones(1, nb);           % Make a vector of ones.
    s1n = s1 + n1;               % Add noise to the set of ones.
    Ric = sqrt(s1n.^2 + n2 .^2); % Simulate the envelope detector operation.
    Ray = sqrt(n1 .^2 + n2 .^2); % Make a vector of zeros with noise included.
    lz = length(Ric(Ric < thr)) / length(Ric); % Calculate BEP for ones.
    l1 = length(Ray(Ray > thr)) / length(Ray); % Calculate BEP for zeros.
    BEP(k) = 0.5 * lz + 0.5 * l1 ;             % Average and record the BEP
end
semilogy(SNRdb,BEP)
disp('Non-coherent OOK');
tablePrint(BEP);
disp(' ');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% NON-COH FSK system
% The threshold is just zero (r1 > r2 --> 1).
%   Simulate two identical channels as in coh FSK, but
%   the form of the channels is as in non-coh OOK.
% On this system, we make use of the fact that the
%  channels are identical: the situation is analyzed
%  for channel 1 carrying ones, and channel 0 carrying
%  zeros. The overall BEP will be the same for the
%  the average of this case with the opposite case.
% This takes four uncorrelated noise vectors: I & Q for each channel.

    load('dMin'); %Use the thresholds determined in analytical series.
for k = 1 : 1 : length(SNRdb)
    thr = 0;                    % Detector threshold voltage.
    nb = nBits;                 % Number of bits in test.
    n1 = wgn(1, nb, NPWR(k));   % In-phase WGN for the ones channel.
    n2 = wgn(1, nb, NPWR(k));   % Uncorrelated quadrature WGN.
    s1 = ones(1, nb);           % Make a vector of ones.
    s1n = s1 + n1;              % Add noise to the set of ones.
    Ric = sqrt(s1n.^2 + n2.^2);   % Simulate chan 1 envelope detector.

    n3 = wgn(1, nb, NPWR(k));   % In-phase WGN for the zeros channel.
    n4 = wgn(1, nb, NPWR(k));   % Uncorrelated quadrature WGN for the zeros channel.
    Ray = sqrt(n3 .^2 + n4 .^2); % Simulate chan 2 envelope detector.
    rv = Ric - Ray;             % Channel 1 - channel 2.
    BEP(k) = length(rv(rv < thr)) / length(rv); % Calculate & record the overall BEP.
end
semilogy(SNRdb,BEP)
disp('Non-coherent FSK');
tablePrint(BEP);
disp(' ');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
legend('coh OOK', 'coh FSK', 'coh PSK', 'non-coh OOK','non-coh FSK')
title('Monte Carlo simulation of various PCM systems')
xlabel('SNR (db)')
ylabel('BEP')
%hold off
disp(sprintf('execution time: %.1f sec.', cputime - strt))
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function tablePrint(BEP)
global res;
disp('SNR   BEP')
for cnt=0:2:12
 disp(sprintf('%d   %.7f',cnt,BEP(cnt * res + 1)))
end ;    disp(' ')
```

Appendix F   Analytical BEP for non-coherent FSK system

(Using CAS, not Matlab)

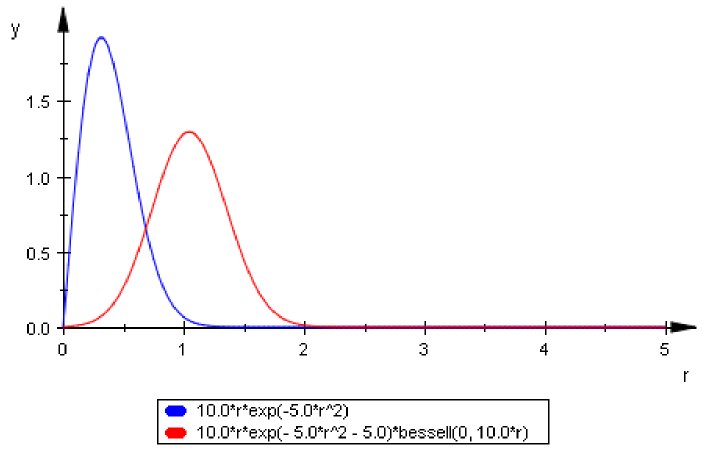
Create a Rician PDF function (with input argument A = 0, it is a Rayleigh PDF)
(Here, s is noise variance, sigma squared).

```
RicianPDF:=(s,A,r)-->r/s*exp(-(r^2+A^2)/(2*s))*(besselI(0,r*A/s))
```

$$(s, A, r) \rightarrow \frac{r \cdot e^{-\frac{A^2+r^2}{2 \cdot s}} \cdot I_0\left(\frac{A \cdot r}{s}\right)}{s}$$

Example curves showing RicianPDF function with A=0 (Rayleigh) and A=1 (Rician)
and noise power = 0.1, noise voltage = sqrt(0.1) = 0.31623.
(To show the function works as expected.)

```
plotfunc2d(RicianPDF(.1,0,r),RicianPDF(.1,1,r),r=0..5)
```



Make a table from 0 to 12 db SNR, double integrating the joint PDF, which is the product of the Rician and Rayleigh PDFs:

```
for s from 0 to 12 do
ssq :=10.0^(-s/10.0):
print(s,float(int(int(RicianPDF(ssq,1.0,r1)*RicianPDF(ssq,0.0,r2),r2=r1..inf
inity),r1=0..infinity)))
end
```

```
0, 0.3894003915
1, 0.3649924781
2, 0.3364282157
3, 0.3036247365
4, 0.2668366011
5, 0.2267932213
6, 0.1848121996
7, 0.1428278437
8, 0.1032563506
9, 0.06863396382
10, 0.04104249931
11, 0.0214837013
12, 0.00951029711
```