BIT ERROR RATIO TESTING: HOW MANY BITS ARE ENOUGH?

Andy Baldman
UNH InterOperability Lab
March 18, 2003

Abstract:

The purpose of this article is to discuss the concept of Bit Error Ratio testing, and will explain, using a minimum amount of math, the answer to a fairly common question:

"How many bits do I need to send in order to verify that a device exceeds a given BER?"

The analysis will provide the same answer as the more formal approach[1], which employs the statistical concepts of Confidence Intervals and Hypothesis Testing, however this analysis is designed to provide an intuitive, easy-to-understand explanation that does not require the reader to have any prior knowledge of statistics or probability.

Introduction:

The concept of the Bit Error Ratio (BER[2]) is ubiquitous in the communications industry. It is commonly used as the fundamental measure of performance for digital communication systems. The BER can be thought of as the average number of errors that would occur in a sequence of *n* bits. Consequently, when *n* is equal to 1, we can think of the BER as the probability that any given bit will be received in error. Formally, the BER is expressed in terms of the following ratio:

$$\frac{1 \text{ error}}{n \text{ bits transmitted}}$$

So, if a device's specified BER is 1/*n*, we can expect to see, *on average*, 1 bit error for every *n* bits sent. The fact that the BER is defined as an average quantity is an important fact, for reasons that will soon become apparent.

While the definition of BER may be fairly straightforward, verifying that a device meets or exceeds a certain BER is not as simple. If a device's BER claims to be 1/*n*, and we send *n* bits and see one error, can we say that the BER is 1/*n*? What happens if we see no errors? What if there are multiple errors? What if we see a different number of errors each time we repeat the test? Is there anything that we can say for certain?

Well, the answer is… maybe. It depends on the circumstances. To begin with, let's start with what we *do* know: If at least one error is seen, we can say with 100 percent certainty that the BER is definitely not zero. Also, as long as at least one bit was

successfully received, we can additionally state that the BER is most definitely less than 1. We've already narrowed it down to somewhere between 0 and 1! How difficult can this be, right?

Obviously, while these observations are technically correct, they really don't help us that much. We need a better way of analyzing this problem, so we'll do what all good engineers and scientists do: we'll make a model of the system, and try to infer answers from the model, based on the characteristics of the model and the assumptions we make about its behavior.

Before we do that though, allow me to make one statement, (which happens to be correct, as we will see by the end of this article.)

"In reality, if a device's BER is $1/n$ and n bits are sent, we can actually expect to see 0 errors during those n bits approximately **35** percent of the time."

"35 percent? Are you crazy? That makes no sense!", I hear you cry.

Yes, it sounds odd. However, it's actually not that difficult to understand why this is the case, and to see where this number comes from. To do this however, we'll need to explain a couple of basic concepts from the study of Probability. Don't panic, it's nothing complicated. In fact it's as simple as flipping a coin…

Bernoulli Trials: Heads or Tails?

If you flip a coin, everyone knows the probability of getting either heads or tails is 50/50 (as long as the coin isn't rigged in some way, of course.) However, it wouldn't be terribly shocking to flip a coin 3 times and get 3 heads or 3 tails in a row. Even though the chance of getting heads for any particular flip is 50%, it is entirely possible to get *any* number of heads or tails for any given number of consecutive flips. Although these events are less likely to happen than usual, they *do* happen, and it's not too hard to say something about how often they happen, using some basic assumptions.

In the study of Probability, one of the simplest and most important concepts is known as the "Bernoulli Trial". Basically, this is just a fancy term to describe flipping a coin. In general, a "Bernoulli Process" is comprised of a series of Bernoulli Trials, where the trials are described by the following 3 characteristics:

1) Each "trial" has two possible outcomes, generically called *success* and *failure*.

2) The trials are independent. Intuitively, this means that the outcome of one trial has no influence over the outcome of any other trial.

3) For each trial, the probability of success (however it may be defined) is $p,$ where p is some value between 0 and 1. The probability of a failure (defined as the only other

alternative outcome to success) is ***1-p***.

That being said, let's go back to coin flipping. Although a normal coin may have a 50/50 chance of landing heads up for any given flip (technically called a **fair coin**), we know that it's fairly easy to flip a coin twice and get 2 heads, or 3 times and get 3 heads. In reality, for the case of three flips, we see that there are only 8 possible combinations that can occur (H = heads, T = tails):

<div align="center">

H-H-H
H-H-T
H-T-H
H-T-T
T-H-H
T-H-T
T-T-H
T-T-T

</div>

Because there are only 8 possible outcomes, *and the chances of getting a head are the same as getting a tail*, then the chances of any one *specific combination* of outcomes occurring is always 1 in 8, or .125. Note that we can also get this value using the probability for each coin flip. If the probability of getting one head is .5, then the probability of getting 3 heads in a row is just:

$$.5 * .5 * .5 = .125 \text{ (or } 1/8)$$

Again, because the chances of getting a head are the same as getting a tail, the probability of any particular one of the eight combinations above is always just $.5*.5*.5 = .125$.

"But how can they all be the same? Shouldn't the chances of getting all heads be smaller?"

Yes, and it is, which is easier to see if we look at the list of the 8 possible outcomes in a different way. If I ask the question, "What are the chances of getting *all heads* for three consecutive flips?" there is only one way to do it, and so the answer is 1/8. However, if I change the wording of the question to ask, "What are the chances of getting *only one* head in three flips?" we see that the answer is now 3/8. This is because here (unlike the all-heads case) there are now *three different* ways of getting what we want (HTT, THT, TTH) and each one has a 1 in 8 chance of occurring. Thus, we can add the probabilities together. To better see why the probabilities add, just look at the table of the 8 possible outcomes for 3 flips. The chances of getting only one head makes up three of the eight total combinations, thus you're three times more likely to get one of those combinations than you are to get the *single* case of all three heads.

Note that we can ask other questions here, too. I could word the question to ask, "What is the probability of getting *up to* 2 heads?" In this case, we need to find the sum of the following:

	Condition	Possible Outcomes		Total Probability
	The probability of getting 0 heads:	TTT	=	1/8
plus,	The probability of getting 1 head:	HTT, THT, TTH	=	3/8
plus,	The probability of getting 2 heads:	HHT, HTH, THH	=	3/8

				7/8

Observe that in order to get this result, we had to find the sum of the probabilities for 7 out of the 8 possible outcomes. Since for any experiment, the sum for all of the total possible outcomes must always add up to 1, in this case we also could have found the answer by computing 1 *minus* the probability of *not* getting up to 2 heads, which is simply 1 minus the probability of getting all three tails, which is 1 - 1/8 = 7/8. Simple!

"OK, so what does all of this have to do with BER testing?"

We can use this same basic coin-flipping model to describe our process of sending and receiving bits, with a few minor tweaks. First, we have to realize that we will no longer have a fair coin. If we equate a "tail" with a bit error, then we now have a coin that flips heads almost all of the time, and which flips tails according to the BER (say 1 tail in 100 flips for a BER of 1E-2 for example.) The other thing to realize is that the math gets a little more complicated. To see why, suppose I ask the question, "What is the total probability of getting *exactly* 3 tails in 100 flips?" To find this probability, one of the things we need to know is the *number of ways* we can get 3 tails in 100 flips, e.g.,

TTT + 97 Heads
THTT + 96 Heads
TTHT + 96 Heads
HTTT + 96 Heads
HHTTT + 95 Heads
Etc…

It doesn't take long to see that the number of combinations quickly gets far too cumbersome to determine by hand. Have no fear, though, since there exists an easy formula that can provide the value much easier.

ENTER THE BINOMIAL COEFFICIENT!

Without getting into the history of it, someone along the way figured out a handy little formula that tells you the total possible number of combinations that k specific outcomes can occur out of a total of n trials. It is sometimes referred to as "n-choose-k", and is technically called the Binomial Coefficient. The formula is as follows:

$$C_{(n,k)} = \frac{n!}{k! * (n-k)!} \qquad \text{[EQ. 1]}$$

Plugging in some real values to this formula, if we want to know the *number of combinations* for getting 2 heads in a total of 3 flips, we get:

$$\frac{3!}{2! * (3-2)!} = \frac{6}{2 * 1} = 3 \text{ possible ways. (HHT, THH, and HTH)}$$

Similarly, for 2 heads in 4 flips, we get:

$$\frac{4!}{2! * (4-2)!} = \frac{24}{2 * 2} = 6 \text{ possible ways. (HHTT, TTHH, THTH, HTHT, THHT, HTTH)}$$

We can easily see the power of this formula, as it allows us to quickly find the number of combinations for examples that would be far too complicated to figure out by hand. For example, take 12 heads in 50 flips:

$$\frac{50!}{12! * (50-12)!} = 121,399,651,100 \text{ possible combinations!}$$

Granted, there is still a problem with trying to figure out the value of 50! by hand, however we can easily do this with a computer. (Note that in MATLAB, you can also use the 'nchoosek' function to compute the Binomial Coefficient directly.)

Armed with the power of the Binomial Coefficient, we now have everything we need to create our analytical model of BER testing.

A Simple BER Model:

We are now ready to study this problem in depth. First, let's take a look back at the bold statement that was made a while back about the 35% probability of seeing zero errors when the BER is $1/n$, and n bits are sent.

Expanding on our coin-flipping example, if we were to say that we now have a coin that flips tails only 10% of the time (instead of the normal 50%), we can begin to

create our model. From this, we can directly compute the probability of getting zero tails (i.e., getting ten heads) in 10 consecutive flips. This would then be:

$$P_{(no\ tails)} = (\#\ of\ ways\ to\ get\ 10\ heads)\ \ x\ \ P_{(10\ heads)}$$

$$=\ \ \ \ \ \ \ \ \ 1\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ x\ \ \ [P_{(1\ head)}]^{10}$$

(Note that I have begun to use the notation "$P_{(1\ head)}$", which is just shorthand for saying, "The probability of getting 1 head." We will continue to use this notation for the remainder of this discussion.)

Note that this is just one specific example of a much more general case. In general, we can write the formula for finding the probability of getting any *k* number of tails out of *n* total flips as the following:

$$P_{tot} = C_{(n,k)} * [p^k * (1\text{-}p)^{(n\text{-}k)}] \qquad\qquad [EQ.\ 2]$$

"Whoa. What the heck is that? You said this wouldn't be difficult."

Hang on a minute. It's not as bad as it looks. First, let's look at the part inside the brackets. When we say we want the probability of getting exactly 1 tail in 10 flips, what we are actually saying is that we want the probability of getting 1 tail, *and nine heads*. This is what the bracketed expression represents. If we want *k* tails, we imply that the remaining number of flips *(n-k)* must be heads. When figuring in these heads we must account for the probability of getting a head, however recall that $P_{(head)}$ is actually just $1\text{-}P_{(tail)}$.

As an example, take two tails in ten flips, where $P_{(tail)} = .1$:

$$[p^k * (1\text{-}p)^{(n\text{-}k)}]\ =\ (P_{(tail)})^2 * (P_{(head)})^8$$

$$=\ (.1)^2\ \ *\ \ (.9)^8$$

$$=\ .0043$$

Now, realize that this number only represents the probability for *one way* of getting 2 tails in 10 flips. The other part of EQ 2 above, (the $C_{(n,k)}$ part) is the Binomial Coefficient, the part that tells you *the total number of possible combinations* by which you can get 2 tails out of 10 total flips. Multiplying the .0043 result from above by the proper Binomial Coefficient will give you the *total probability* of getting two tails in ten flips for $P_{(tail)} = 1/10$:

$$P_{tot} = C_{(n,k)} * [p^k * (1\text{-}p)^{(n\text{-}k)}]$$

$$= C_{(10,\ 2)} * [.1^2 * .9^8]$$

$$= 45 * [.0043]$$

$$= .1937$$

So, this means that if $P_{(tail)}$ is 1/10, then for ten flips, you will have a 19.37% chance of getting exactly *two* tails. Pretty neat, huh?

OK, so let's do more with this. What then, would be the probability of getting just *one* tail in ten flips? Let's see:

$$P_{tot} = C_{(n,k)} * [p^k * (1-p)^{(n-k)}]$$

$$= C_{(10, 1)} * [.1^1 * .9^9]$$

$$= 10 * [.03874]$$

$$= .3874$$

We see that the result is higher than before, in that for $P_{(tail)} = 1/10$, we're 38.74% likely to see 1 tail in 10 flips, compared to the 19.37% chance we have of seeing 2 tails. (But it makes sense that we'd expect to see 1 tail more often than 2, since our coin has a known $P_{(tail)}$ equal to 1/10, right? Right…)

Even more interesting is the case for 0 tails (i.e., 10 heads):

$$P_{tot} = C_{(n,k)} * [p^k * (1-p)^{(n-k)}]$$
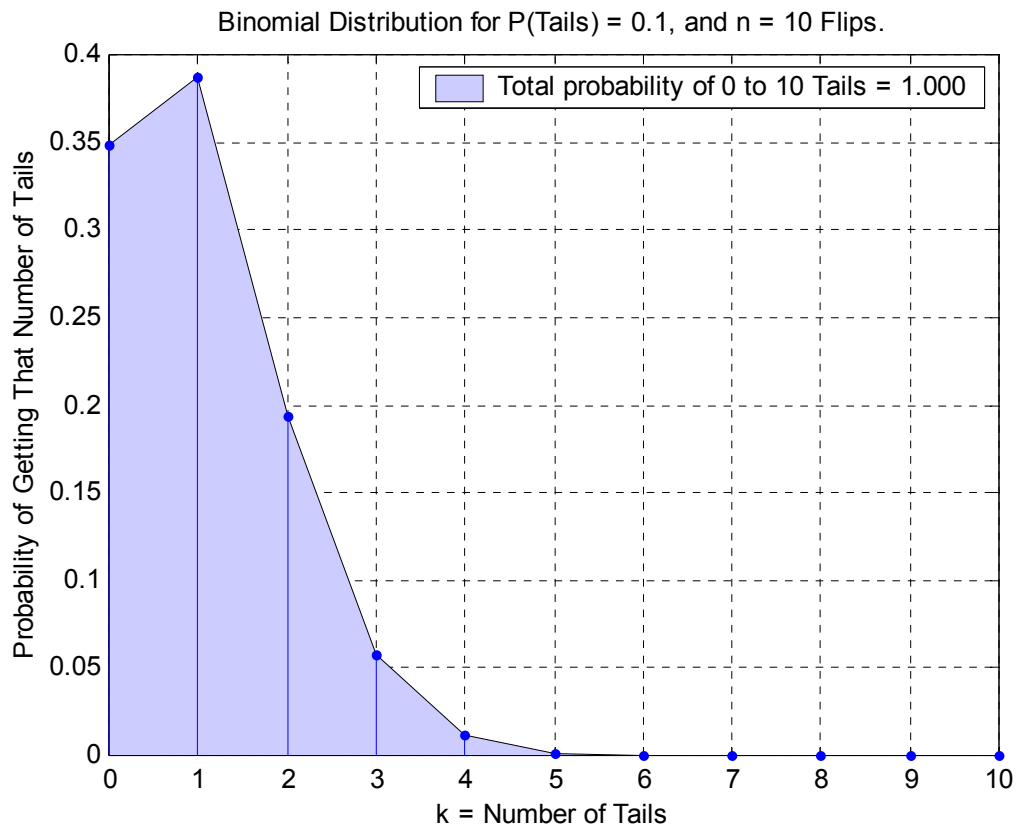
$$= C_{(10, 0)} * [.1^0 * .9^{10}]$$

$$= 1 * [.3487]$$

$$= .3487$$

So, if $P_{(tail)}$ is 1/10, and I perform 10 flips, then there's an approximately 35% chance that I will see *zero* heads in those 10 flips. Amazing! It almost seems counter-intuitive, but the math spells it out pretty clearly. Note that seeing 0 tails is still *slightly* less likely to occur than seeing 1 tail (34.8% for 0 tails, vs. 38.7% for 1 tail), although they're pretty close.

Since we've now calculated the probabilities of getting 0, 1, or 2 tails in 10 flips, we can now write a little program to go and calculate the probabilities for all the possible number of tails we could possibly see for 10 total flips (up to a maximum of 10, of course.) (Note that the MATLAB code to do this will be included at the end of this article.) Here's what the plot looks like for all possible numbers of tails given 10 total flips:

Binomial Distribution for P(Tails) = 0.1, and n = 10 Flips.

Total probability of 0 to 10 Tails = 1.000

Probability of Getting That Number of Tails

k = Number of Tails

Note that in the plot above, the stem plot represents the discrete values 0 through 10, for which the probabilities were calculated. (The shaded region is only shown to highlight the overall shape of the plot.) Note also that the sum of the probabilities for 0-10 tails adds up to 1.000, as it should, since every time you do 10 flips, you *must* get some number of tails between 0 and 10 inclusive.

Now, let's make things interesting. Let's compute the distribution from 0 to 10 tails again, however this time let's assume that $P_{(tails)} = 2/10$, or twice what it was in the last example. Here's what we get:

Binomial Distribution for P(Tails) = 0.2, and n = 10 Flips.
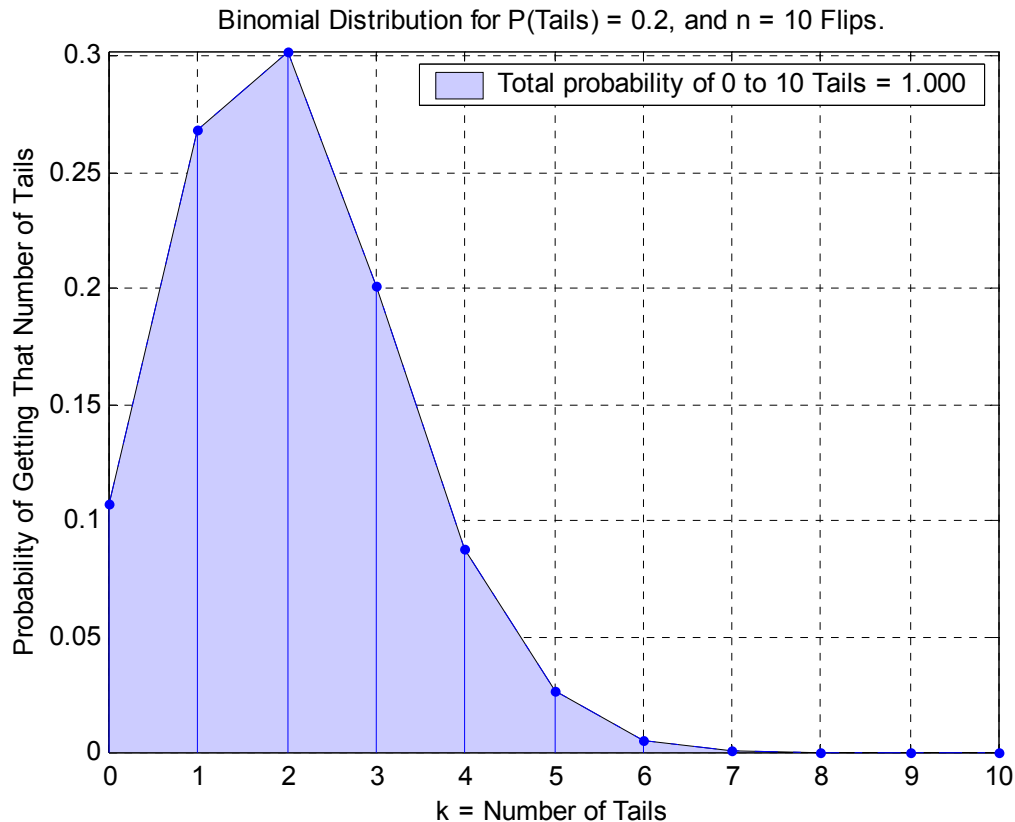
Look closely. Even though $P_{(tails)}$ was 2/10, we see that approximately 10% of the time when we do the 10 flips, we will see *zero* tails. Also, approximately 27% of the time, we can expect to see just one tail. We're still *most likely* to see 2 tails (again, because P(tails) is 2/10), however the chances of seeing 0 or 1 aren't that far behind.

"OK, so why does all of this matter?"

The reason all of this matters goes back to the idea of BER testing. In this model, if we consider a "tail" to be a "bit error", and replace "total number of flips" with "total number of bits sent", then we have our simple BER model.

If we designed our BER test such that we attempt to verify that a particular device's BER is equal to or better than 1/10 by sending 10 bits and allowing up to 1 error to occur (in order to be considered "passing"), then we run into the case whereby a device with a BER that is *clearly* sub-standard (2/10), could theoretically slip by, and actually PASS the test approximately 37% of the time! If we change the threshold and only allow

0 errors to be seen in order to pass, we still will have 'bad' devices slipping by and passing approximately 10% of the time.

   Note that while this may seem like a good thing to the person who built the device, it is most certainly a bad thing from the perspective of the person trying to verify the actual BER. This type of error is what is known as a "Type II Error" in the formal terminology of Hypothesis Testing, whereby the device is given a passing result, even though it truly does have a failing BER. This is arguably worse than the opposite case (Can anyone guess what that might be called?) A Type I Error describes the opposite case, where the device actually *does* have a valid BER, however for that particular set of flips it just happened to get *really unlucky* and have multiple bit errors, thus causing it to fail.

   To see the impact of a "Type II Error", let's go back to the original case where $P_{(tail)}$ (a.k.a. $P_{(error)}$) was 1/10:

Binomial Distribution for P(Error) = 0.1, and n = 10 Bits.
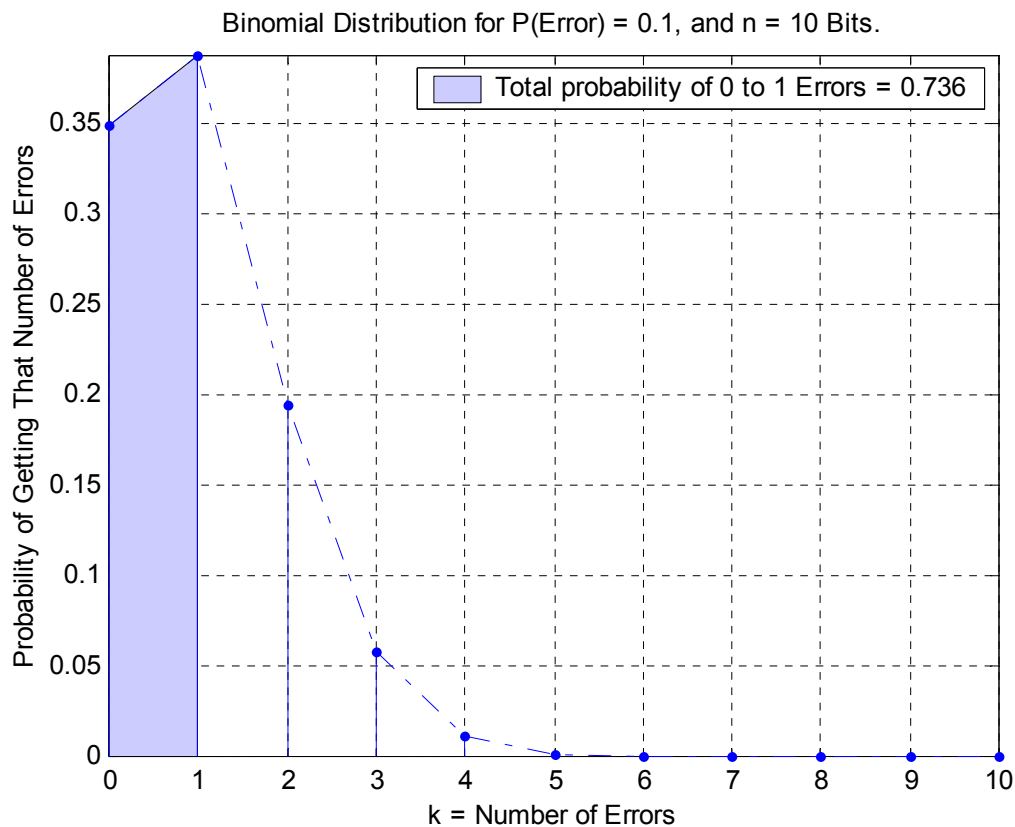
Here we see the same plot as before, however this time "total probability" only shows the sum of the first 2 values of k, up to and including 1 error (crudely approximated by the shaded "area under the curve", however keep in mind that because the plot is a set of *discrete* values, the sum really is just the sum of the 0 and 1 error probabilities, and not a true area, as would be the case if the plot were continuous.) The point of showing the plot this way is that we now see that even though the BER is really

1/10, approximately (1 - .736) = **26.4%** of the time, we will actually see 2 or more errors and erroneously assign a *failing* result, even though the device's BER truly was conformant.

"OK, that all makes sense. So, how do we deal with this?"

Well, we do just about the only thing we can do - send more bits. However, it is important to realize the fact that as we send more and more bits, we are simultaneously *increasing* the overall chance that we *will* see bit errors. Thus, we may have to adjust the cutoff value for the number of errors that we will allow and still assign a passing result to the device being tested.

Let's see what happens when we *triple* the number of bits we send:

Binomial Distribution for P(Error) = 0.1, and n = 30 Bits.
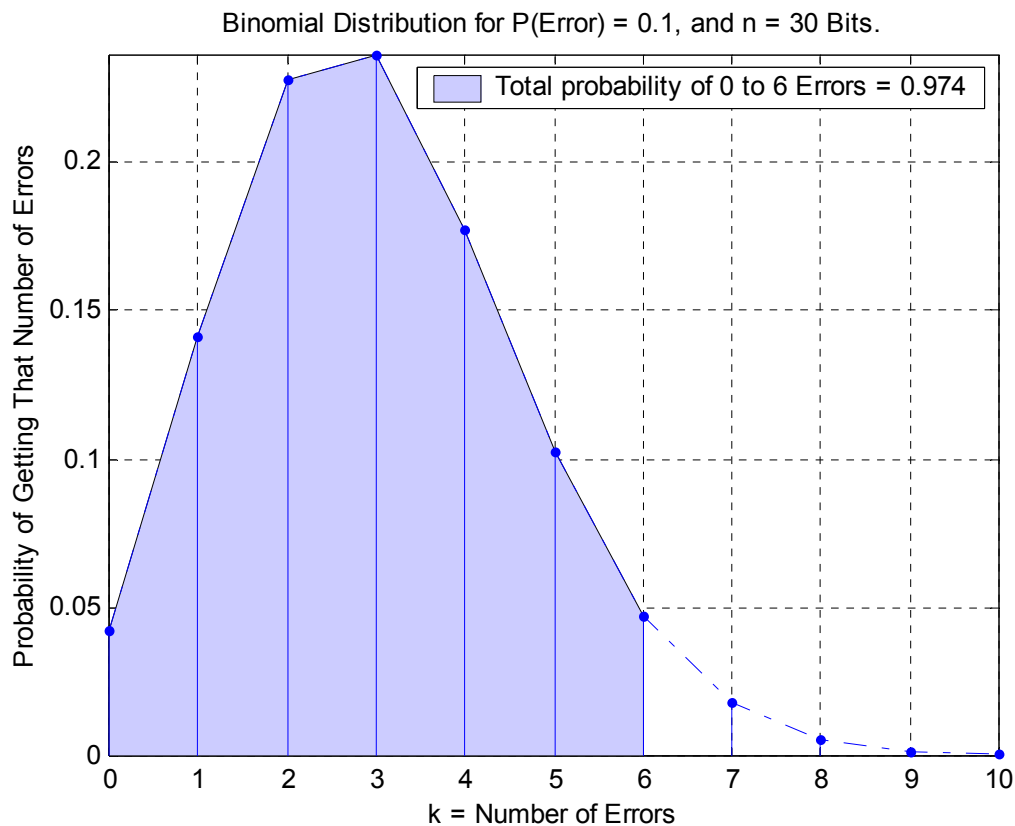
Here we see that when the BER is 1/10 and 30 bits are sent, we are *most* likely to see 3 errors (again, which makes sense, intuitively). More significant is the fact that 97.4% of the time, we should expect to see 6 errors or less. Also, perhaps the most important observation is that we can expect to see 0 errors less than 5% of the time. To see why this is meaningful, consider the following: Suppose that we are trying to verify that the BER of an actual device is 1/10 or better. If we send it 30 bits, and see *zero* errors, this means either one of two things is true:

1) The actual BER is equal to or *worse* than 1/10, and the device just got *really* lucky.

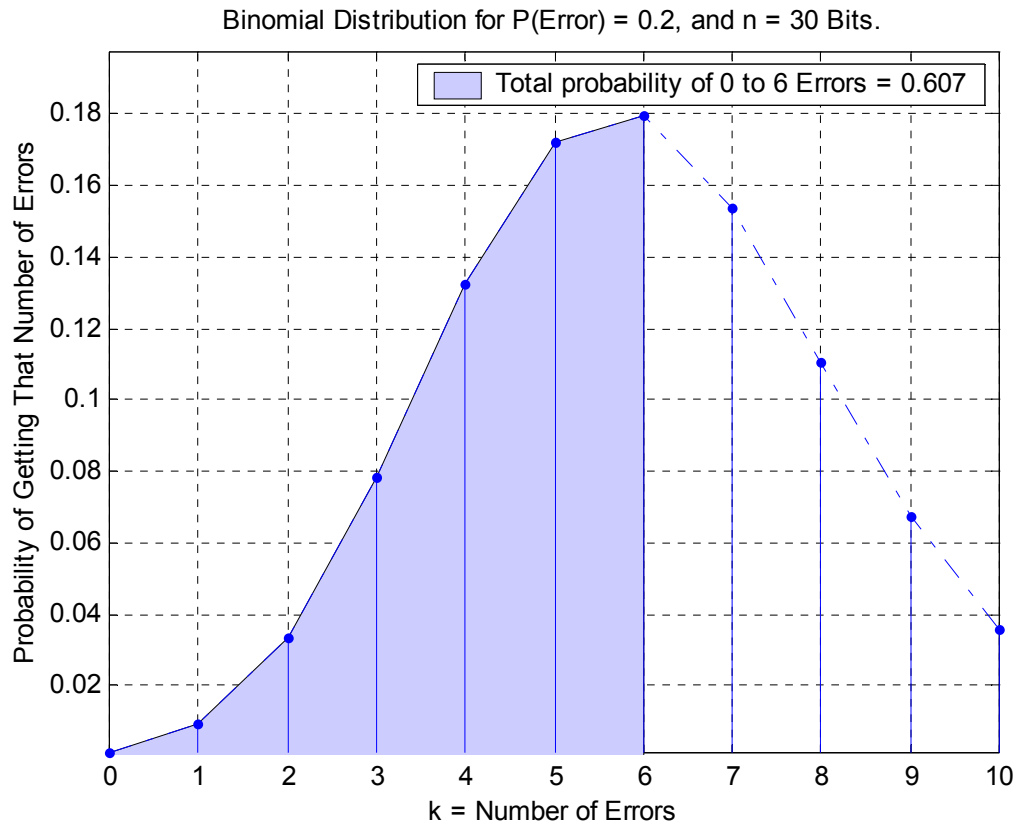<div align="center">OR</div>

2) The actual BER is better than 1/10.

For the same experiment, if *7 or more* errors are seen, we can also come to one of two conclusions:

1) The actual BER is equal to or *better* than 1/10, but for this particular run, the device just was especially *unlucky*, and landed in the ~2.6% chance region of having 7 or more bit errors.
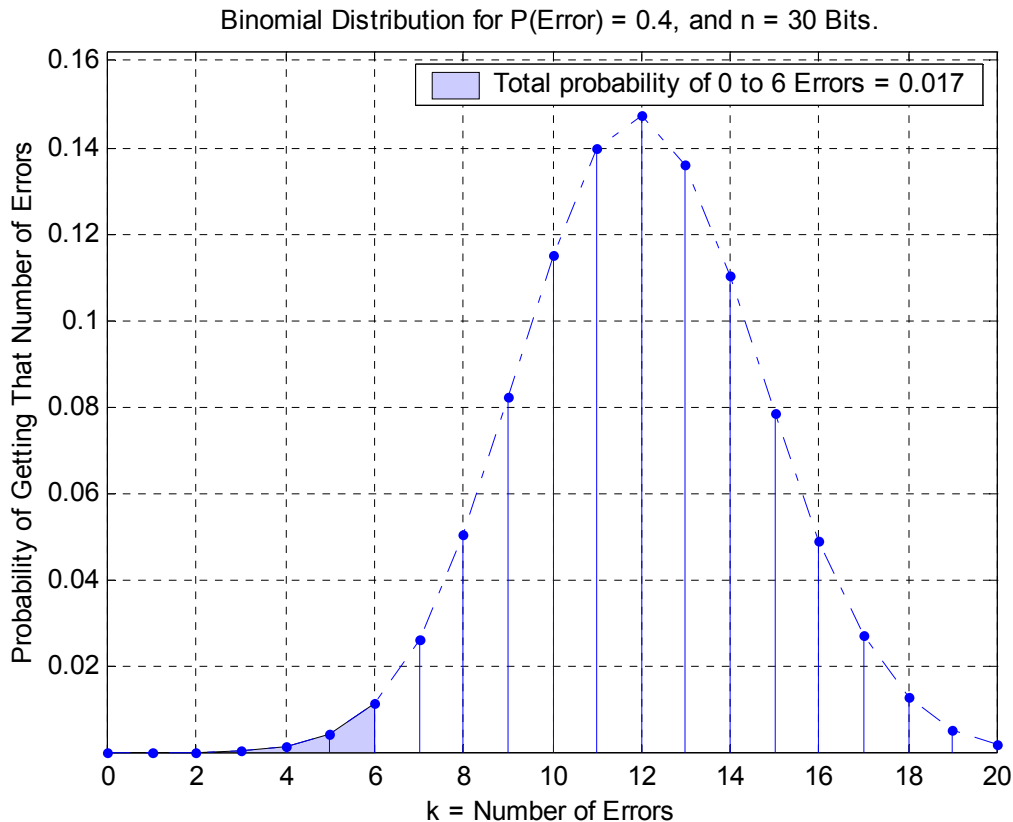
<div align="center">OR</div>

2) The actual BER is *worse* than 1/10.

So what happens if we see 1 to 6 errors? Well, we can't be sure exactly (at least not as sure as for the other two cases), however we can use our model to get an idea of where we lie. It is possible that a device could have a worse BER than what we're testing for, and still produce 6 or fewer errors with a fair probability, however those chances drop off quickly. To see how this affects the error distribution, let's see what the curve looks like if the actual BER is just one bit error worse (2/10), given the same number of bits sent: [n = 10;ber(2/n, 3*n, 10, 6);]

Binomial Distribution for P(Error) = 0.2, and n = 30 Bits.

Total probability of 0 to 6 Errors = 0.607

Probability of Getting That Number of Errors

k = Number of Errors

Here we see that the device has a 60.7% chance of producing 6 errors or less and 'passing' the test, even though the actual BER is failing. If we go just two bit errors further, (4/10), we get: [n = 10;ber(4/n, 3*n, 20, 6);]

Binomial Distribution for P(Error) = 0.4, and n = 30 Bits.

Total probability of 0 to 6 Errors = 0.017

Probability of Getting That Number of Errors

k = Number of Errors

Now, we see that the chance of getting 6 errors or less also falls to about 2%. In general, we see that as the actual BER increases, the 'peak' of the distribution moves further and further to the right as a function of *3 times* the actual BER. In other words, depending on whatever the actual BER is (in terms of errors per *n* bits), the peak of the distribution will be shifted further and further to the right by a factor of 3, because we are sending 3*n* bits. The larger the actual BER, the smaller the area under the tail will be, which represents the chances of getting up to 6 bit errors.

(Also note that as the actual BER gets worse, the chances of seeing 7 or more errors increases rapidly, strengthening our confidence in assigning a failing result to devices that show 7 or more errors.)

Thus, the question still remains as to how to interpret the results when 1 to 6 errors are observed. In actuality, this is exactly the same "Type II Error" situation as was seen before when we were sending only *n* bits (instead of 3*n*), and looking for the probability of seeing zero errors. Practically speaking, any test design we adopt will have a range in which the results will be effectively inconclusive. In these cases, as was the case for 0 errors, the only real solution is to design a more thorough test in which more bits are sent. In the absence of this option, the result basically comes down to semantics. Using a courtroom analogy, if the DUT is on trial as the defendant, we must ask the following:

1) Is it up to *us* (the prosecution) to prove "beyond a reasonable doubt" that the BER is failing?

<center>OR</center>

2) Is it up to the DUT, in its own defense, to prove that its BER is passing, with the same degree of confidence?

If the burden of proof lies with the person performing the test, then we have failed to produce conclusive proof "beyond a reasonable doubt" that the BER is failing (i.e., 7 errors or more), and thus must assign a passing result by default. Alternately, if the burden is shifted to the DUT, then *any* number of observed errors other than zero would be considered "reasonable doubt", and would constitute a fail. In practice, the interpretation tends to favor the former scenario, whereby the DUT is "innocent until proven guilty" so to speak, and the burden of proof is placed on the tester to see 7 or more errors.

Note that for case where 1 to 6 errors are seen, the option also exists to make a statement to the effect that the device meets or exceeds a *lower* BER, with a higher degree of confidence. This is generally not desirable, since we are now interpreting the test based on the observed outcome (instead of the other way around), which defeats the purpose of the original test that was defined. In this case, we can view the test as effectively "sending more bits", albeit relative to a *lower* target BER.

Validity of the Model: Scaling

All of the examples so far have used BER values of $1/n$, $2/n$, $3/n$, etc., and trial lengths of n or $3n$, where n was always equal to 10. This was done solely to keep things simple and intuitive. We have seen that we can design a BER test whereby if we want to verify a BER of $1/n$, we send $3n$ bits, and interpret the results as follows:

1) 0 errors seen:
   We are 95% sure that the actual BER is equal to or better than $1/n$.

2) 7 or more errors seen:
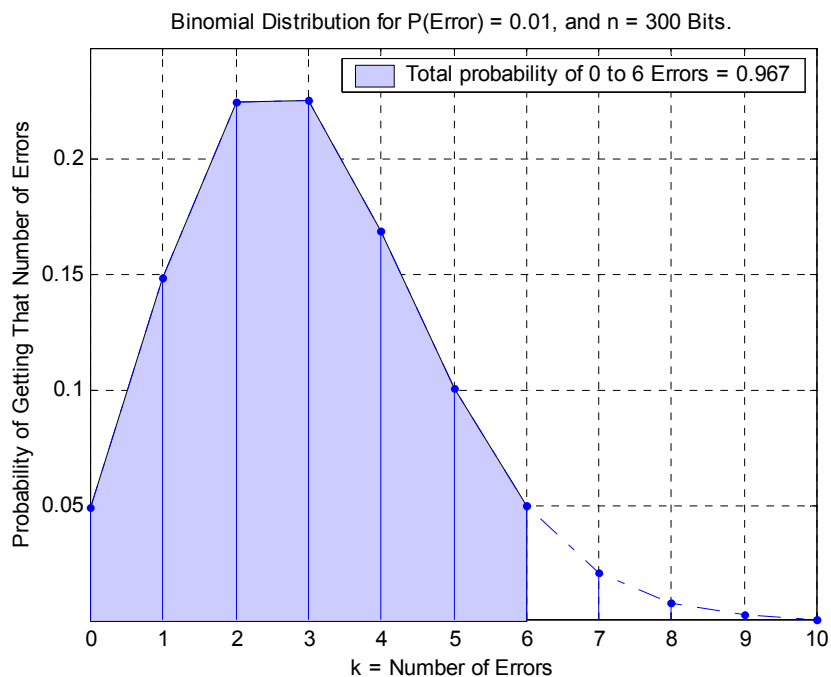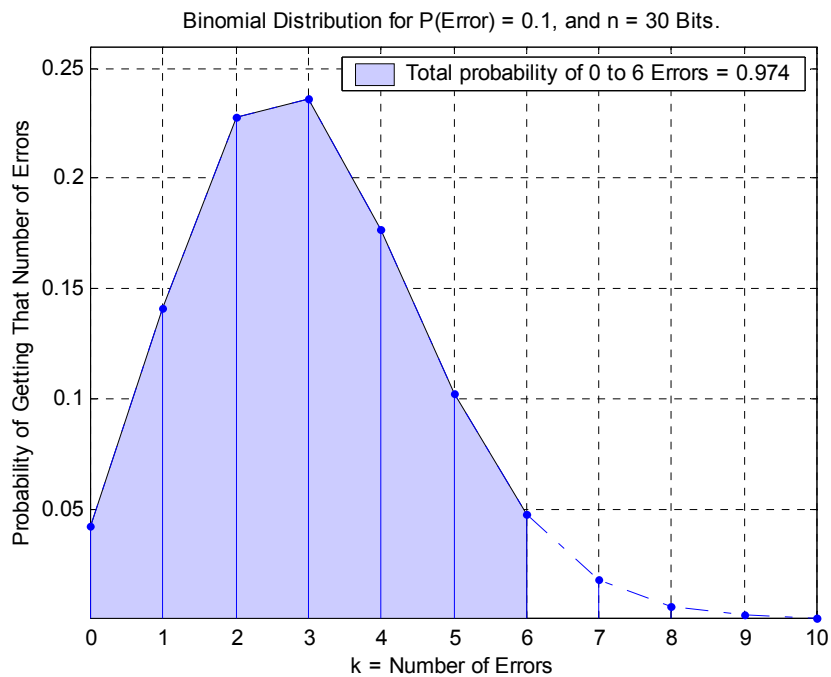   We are 97% sure that the actual BER is *worse* than $1/n$.

3) 1 to 6 errors seen:
   Can't say anything with high confidence, so the device passes by default.

The question then arises,

"Do these values still apply for cases when n is greater than 10?"

Well, the quick answer is basically, yes. Let's see what happens to the numbers when we change n from 10 to 100:

Binomial Distribution for P(Error) = 0.1, and n = 30 Bits.

Binomial Distribution for P(Error) = 0.01, and n = 300 Bits.

We see that the total probability for up to 6 errors has decreased slightly, by less than 1%, while the probability of just 0 errors seems to have increased to almost exactly 5%. What happens when we try n = 1000? Here's the plot:

Binomial Distribution for P(Error) = 0.001, and n = 3000 Bits.
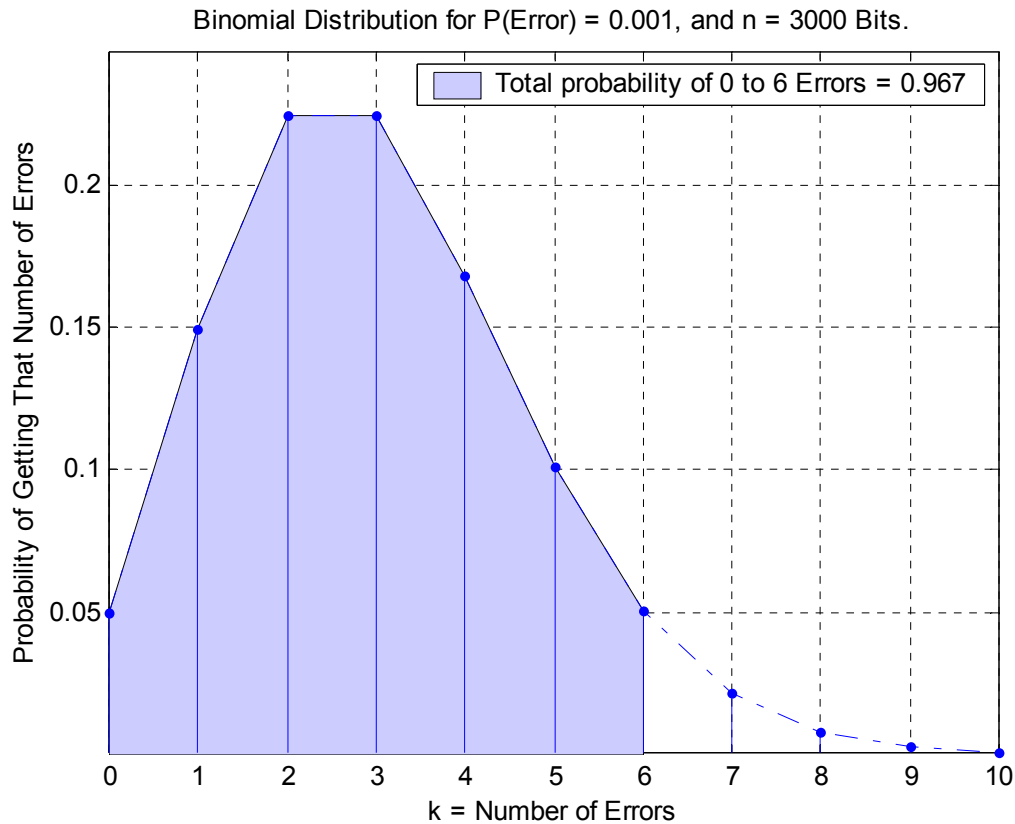
We see that the results haven't changed at all when we go from $n = 100$, to $n = 1000$ (at least within the displayed number of significant digits.) This is the behavior we see for values of n greater than about 100. To formally prove that this is the case as n goes to infinity requires a bit more math than we'd like to get into here, but as a quick "feel good" check, we can easily just look at the behavior of the case for 0 observed errors, as n gets large.

As we computed before, for a BER of 1/n, the probability of seeing 0 bit errors (or "all heads") in n trials is just:

$$P_{tot} = [P_{(head)}]^n$$

$$= [1 - P_{(tail)}]^n$$

$$= [1 - 1/n]^n$$

We can easily plot this quantity as a function of n:

Probability of 0 Errors Given P(e) = 1/n and n Trials
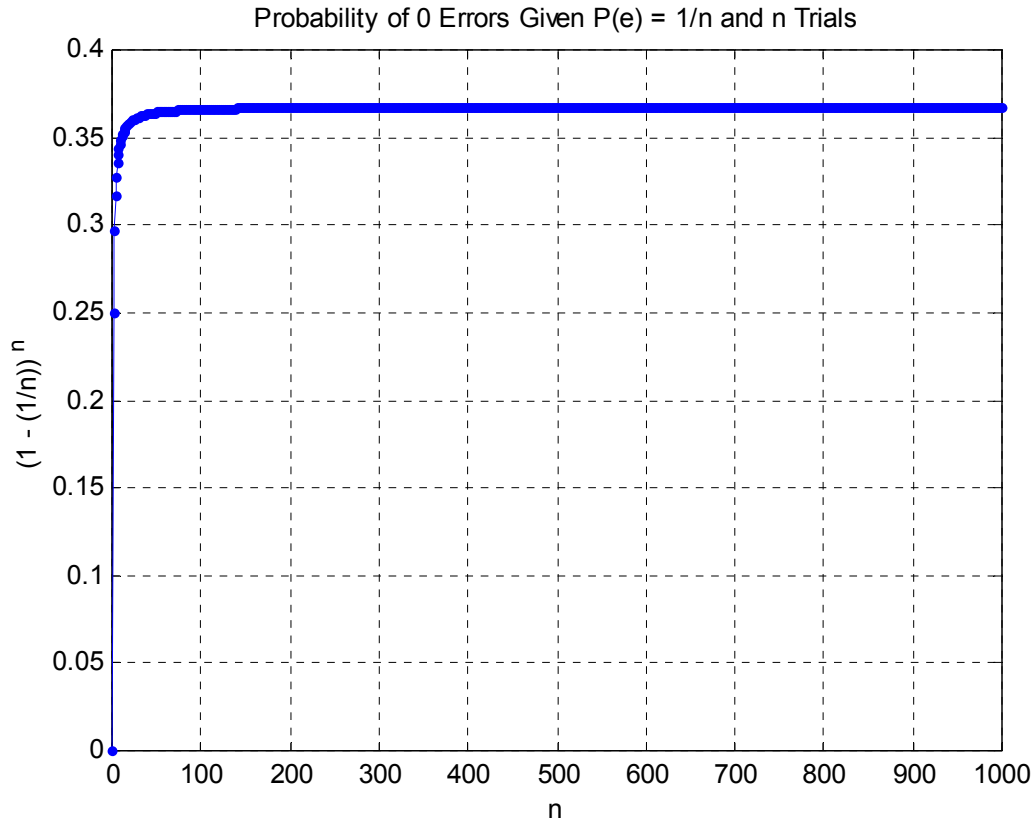
We see that the value asymptotically approaches some particular limit value, and for *n* greater than 100, it is practically level.

(As an aside, the actual limit of $(1-1/n)^n$ as n -> ∞ is actually 1/e, (~.3679), but the proof of this is left up to the reader. (Hint: Look it up in Google. That's what I did!))

Conclusion:

In this paper, we have attempted to de-mystify some of the statistical analysis surrounding the notion of Bit Error Ratio testing. Beginning with the basic Probabilistic concepts of Bernoulli Trials and the Binomial Distribution, a model was developed that allows for the visualization of such statistical concepts as Confidence Intervals and Hypothesis Testing. Whenever possible, analogies and examples were used to help simplify these concepts and present them in a manner that appeals ideally to the intuition of the reader, without requiring mathematical "leaps of faith", or prior training in Statistics or Probability Theory. A small MATLAB implementation of the model is included (see Appendix A). This code was the exact code used to generate all of the error distribution plots in this document, and is included here for the benefit of anyone wishing to further explore this subject.

APPENDIX A:
MATLAB CODE FOR CREATING ERROR-RATE DISTRIBUTIONS

```matlab
function dist = ber(p, n, e2calc, nsum)
%BER  Compute binomial distribution for zero to n errors
%     given a target BER and number of bits sent (trials).
%
%     User provides the probability of error, number of trials,
%     number of error values to calculate out to, and the subset
%     of that number to include in the 'total probability' sum.
%
%     The plot is the binomial error distribution.
%
%dist = ber(p, n, e2calc, nsum);
%dist = ber(1E-8, 3E8, 20, 7);

%--- Revision History ----------------------------------------------
% Version 2.0 (06Feb2003) (aab)
%     + Created by Andy Baldman, UNH InterOperability Lab.
%-------------------------------------------------------------------

% Label the plots in terms of bits/errors, or flips/tails?
s = {'Errors', 'Bits'};
%s = {'Tails', 'Flips'};

% (Since the numbers involved here can get really big, the results of nchoosek
% are only exact to 15 places in some cases.  (Good enough for our purposes.)
% MATLAB normally notifies the user when this happens.  Temporarily turn off
% the displaying of warning messages.)
warnstate = warning;
warning off

for i = 1:e2calc+1
   k = i-1;
   C = nchoosek(n,k);
   p_of_num = C * p^k * (1-p)^(n-k);
   x(i) = k;
   dist(i) = p_of_num;
end

% Plot.
figure;
hl = area(x(1:nsum+1),dist(1:nsum+1),'facecolor',[.8 .8 1]);
hold on;stem(x,dist,'b.');  % Make the stem plot.
hold on;plot(x,dist,'b-.'); % Add a dash plot to highlight the overall shape.
xlabel(sprintf('k = Number of %s', s{1}));
ylabel(sprintf('Probability of Getting That Number of %s', s{1}));
title(sprintf('Binomial Distribution for P(%s) = %.3g, and n = %d %s.', s{1}(1:end-1), p,
n, s{2}));
legend(hl, sprintf('Total probability of 0 to %d %s = %.3f', nsum, s{1},
sum(dist(1:nsum+1))), 1);
zoom on;grid on;
axis([min(x) max(x) min(dist) 1.1*max(dist)]); % Adjust axes to look pretty.
warning(warnstate);  % Reset warning state
```