PROFESSOR: So the idea behind the encoder is increase the minimum distance at the cost of spectral efficiency. The idea behind the decoder was the following. You have a received signal, y. And the job of the decoder is to find x hat, which belongs to the signal set so that you minimize your probability of error. And the probability of error x is not equal to x-hat. That's the probability of error for your decoder.

And we saw a bunch of equivalence rules -- we said that the minimum probability of error rule, which is this rule here, is the same as the maximum a posteriori rule. And this just follows by the definition. This was the same as the maximum likelihood rule. And here we made one assumption that all the signals in your signal set A are equally likely. And this was shown to be equivalent to your minimum distance decision rule. And in order to show this equivalence, we use the fact that the noise was White and Gaussian. So this assumes equi-probable signals. And this assumes AWGN channel. OK.

In fact, the minimum distance decisions has very nice geometrical properties. We basically said that the decision regions had a particular shape. They were convex polytopes. And they were also known as Voronoi regions. However, finding the exact expression for the probability of error seemed quite tedious so we decided to settle with bounds. And we developed some bounds for the probability of error. And the main bounds were -- we had a strict upper bound, which was the summation. OK. This is a strict upper bound. Well, the inequality here, say it's a strict upper bound.

We also had a union bound estimate. Because this is a sum of various Q functions, this is still a very cumbersome expression to evaluate. So we wanted a simpler expression. So we then approximated this by K_min of aj times Q of d_min over 2 sigma. OK, and now if we take the average on both sides, what we get is the probability of error is approximately equal to K_min of the constellation, which is the average number of nearest neighbors, times Q of d_min over 2 sigma. And this expression is known as the union bound estimate. We call it an estimate because it's not a bound anymore. We have made an approximation going from here to here. So this is what we came up with last time.

Throughout the analysis today, whenever we have a probability of error expression, we will be estimating it by the union bound estimate. So this union bound estimate is quite important. Are there any questions on this?

OK, so today we will start by introducing the notion of effective coding gain. And we'll use the symbol gamma sub f for that. What do we mean by effective coding gain? First you have to decide what region you're operating in. It's defined for both bandwidth-limited regime and power-limited regimes. So let us start with the power-limited regime. Remember in the power-limited regime, the trade-off we care about is the probability of bit error versus Eb over N_0. The baseline scheme is 2-PAM. And for 2-PAM, we showed two lectures ago that the probability of bit error is given by Q of root 2 Eb N_0. OK, it should be -- so now the idea is to plot your probability of bit error as a function of Eb N_0. Eb N_0 is always plotted in dB on the x-axis. The probability of bit error we also plot on a logarithmic scale. So this is ten to the minus six. This is ten to the minus five, ten to the minus four and so on. The Shannon limit is at -- this is the Shannon limit -- is at minus 1.59 dB. OK?

For the power-limited regime, the performance of 2-PAM, which is given by this expression, is here. And we basically go to say that when the probability of bit error is ten to the minus five, the EbN _0 that you require is 9.6 dB. This is a nine. So what's the idea behind the effective coding gain? Well, suppose I give you a certain constellation, OK, a certain signal set. And you find the probability of bit error for that, and you plot it. And it comes out to be something like this. So this is the performance for some given signal set A. This is your 2-PAM.

And now you want to quantify how much better is the signal set over the baseline system. Now, so in other words you want to look at the gap between the two curves. Now clearly the gap is going to be a function of the probability of bit error, right? But the basic idea now is you want to fix a certain probability of bit error because if you're designing a system, the probability of bit error is something that the system tolerates and is going to be fixed throughout the analysis.

So in this course, we'll be fixing it at ten to the minus five. You fix this probability of bit error to ten to the minus five, and you look at how much a gap you have between the two curves. And this gap is going to be your effective coding gain in dB. So in other words, effective coding gain, I can write here, is the gap between a given signal set and concordant system, 2-PAM, at a fixed probability of bit error. OK?

So now let us try to quantify this effective coding gain using the union bound estimate that we have here. So if I have a certain signal set, A, then I know from the union bound estimate that the probability of error is given by K_min of A times approximately Q of d_min over 2 sigma. So since I want that expression in terms of probability of bit error, I will use the fact that probability of bit error is probability of error per symbols or the number of bits per symobl. Since I have endpoints in my signal set, I have logM bits per symbol. And this is then K_min of A over logM base 2 times Q of d_min over 2 sigma.

So I'm going to rewrite my probability of bit error expression now in this -- in fact the right side in this quad. K_b of A times Q of root 2 gamma_c of A times Eb over N_0. So I'm just going to define the right hand side in this way. And I'm going to relate gamma_c of A and K_b of A to the parameters I have on the right hand side there. And why do I want to do it this way? Because I want to get an expression as close as possible to the performance of an uncoded 2-PAM system because that way things will be easier to do if I want to calculate the effective coding gain, OK?

So how do the parameters relate? Well, K_b of A is K_min of A over logM to the base 2. And this is the average number of nearest neighbors per information bit. If I compare the arguments inside the Q function, what I have is 2 gamma_c of A times Eb N_0 is the argument d_min over 2 sigma squared, which I will write as d_min squared over 4 sigma squared.

So remember sigma squared is N_0 over 2. So if I simplify this expression, what I will end up getting is gamma_c of A is d_min squared over 4 Eb. Gamma_c of A is known as the nominal coding gain. It's not the same as the effective coding gain, which is the distance between the two curves. OK, are there any questions so far?

So far what I have done is given a constellation A, I can find two parameters. I can find the nominal coding gain, and I can find the average number of nearest neighbors per information bit. And using these two parameters, we will try to get a handle over the effective coding gain. So that's the idea.

So how can we plot -- so given this constellation A, we want to plot the probability of bit error curve like this. But instead of plotting the exact curve we will be plotting this curve here, which is the union bound estimate of your probability of bit error. So we will always be plotting the union bound estimate.

So how do we do that? So we'll again start with a curve like that. Probability of bit error as a function of Eb N_0. This is in dB. The y-axis is also in dB. Sorry, not in dB but on the log scale. And so on. This is the Shannon limit at minus 1. 59 dB. This is the 2-PAM performance.

So if we want to plot this union bound estimate, you will be doing two steps. First, we'll be plotting this Q function, a curve corresponding to the Q function. And then we'll be scaling it. So if I just want to plot this Q function, how will it compare to this curve here? Well, you're just scaling the argument inside the Q function, right? But now because we are plotting the x-axis on a dB scale, it simply means that we are translating to the left by an amount of gamma_c in dB. Is that clear?

Well, let's see. What we are doing is we are going to scale the x-axis by a certain constant. So this means that we will have to scale x-axis here. But, our Eb N_0 is being plotted on a dB scale, right? So multiplying in the linear scale corresponds to an addition on the logarithmic scale. An addition on the logarithmic scale simply implies a shift on the x-axis so that's the constant shift on the x-axis. I'm going to plot it here. This curve is going to be parallel to the original 2-PAM curve to the best of my abilities. OK, so this is what we get as your Q function.

Now we are going to scale the y-axis by this factor here, K_b of A. But remember the y-axis is also on a logarithmic scale, rather than the linear scale. So multiply [INAUDIBLE] by a factor of K_b of A implies a vertical shift by a certain factor. So I'm going to do a vertical shift here. So let me make that darker now. So this is what I end up getting.

OK, now note that the distance between these two curves is not fixed anymore. If the slope is steeper, then this distance is small, meaning that this distance here is large. On the other hand, if the slope is going to be smaller here, then this distance is large. So this means that a distance between these two curves is no longer fixed. And basically what we're saying here is that if the probability of error is large, then the number of nearest neighbors matters much more. But if the probability of error is going to be quite small, then the more important factor is your exponent, how fast the slope decays as opposed to the number of nearest neighbors, OK?

This distance here is your effective coding gain. And this constant horizontal distance here is your nominal coding gain in dB. Are there any questions on this curve?

AUDIENCE: [INAUDIBLE] on the right of four 2-PAM because these are the logM.

PROFESSOR: So so what I'm really assuming is that gamma_c of A is greater than 1. So if I'm adding it, then I'm going to shift to the left, right? OK, so a couple of remarks. I'm just summarizing a few remarks I made while plotting the curve. First is the log-log scale is very convenient because any multiplicative factor simply involves a translation along the x and y directions.

The second point is that the accuracy we have in the effective coding gain in this way is determined by the union bound estimate because that's all we are plotting. We are not really plotting the exact curve. And the third point is that we have a rule of thumb. Because the distance between the curves is not fixed, it's still too tedious to find the exact numerical value of effective coding gain. So what we can find is the value of the nominal coding gain and K_b of A exactly, given a constellation. So we want to have a rule of thumb for the effective coding gain.

And the rule of thumb is that the effective coding gain is given by the nominal coding gain in dB minus 0.2 log2 of K_b of A. This is in dB. What this means is every factor of two increase is K_b results in a loss of 0.2 dB in the effective coding gain. OK, and this is our probability of bit error of ten to the minus five. So that's the region that we will be doing our analysis in. So this rule of thumb is quite helpful. Are there any questions?

AUDIENCE: [INAUDIBLE]

PROFESSOR: This rule of thumb? Well, it works quite well in practice so you want to use that rule.

AUDIENCE: [INAUDIBLE]

PROFESSOR: Right. If things are clear, let us do some examples. So let us start with a 2-PAM for examples. The first is 2-PAM system. What's the effective coding gain going to be for this constellation? It's going to be 0 dB, right? It just follows from the definition. I mean the effective coding gain tells us how much gap we have between this new constellation and 2-PAM. If your new constellation is 2-PAM itself, we don't have any effective coding gain.

What about the 4-QAM? It's going to be still 0, because a 4-QAM is a Cartesian product of two 2-PAM constellations. And we argued last time that there is no coding gain if you use a Cartesian product. If you want to verify it using the framework that we have just developed, say we have four points. This point is alpha, alpha. This point is minus alpha, alpha, minus alpha, minus alpha, and alpha minus alpha. The nominal coding gain is given by d_min squared over 4 Eb. The minimum distance is 3 alpha between any two points. So we have 4 alpha squared.

Then what's the energy per bit? Well, the energy per symbol is 2 alpha squared. Then we have two bits per symbol, so the energy per bit is alpha squared. So we have 4 alpha squared, and so that's 1. So we do not have any nominal coding gain. K_b of A, what's that going to be? Well, we have two nearest neighbors per symbol, but we also have two bits per symbol. So the number of nearest neighbors per bit is going to be one. So your nominal coding gain is 1, K_b of A is going to be 1. And so we do not see any coding gain for the 4-QAM system.

Now let me do a slightly different case. Let me start with a 4-QAM, but I remove two points. I remove this point, and I remove this point. And I only keep the two points here. So I have a constellation, say, A prime, which only has two points, one point here and one point here. So this point is alpha, alpha, and this point is minus alpha, minus alpha. Any idea what the effective coding gain will be for this case?

AUDIENCE: I think it's the same.

PROFESSOR: It's still zero, right?

AUDIENCE: Yeah, because it's saying that's 2-PAM.

PROFESSOR: Exactly. All this is a 2-PAM constellation embedded in two dimensions. It's 2-PAM along this line, right? So I cannot hope to have a higher coding gain for this constellation over the original one. I mean I told you the story that we can take a subset of points, and we can increase the minimum distance by throwing away some points. But one has to be careful in that analysis. There could be examples where you're strictly improving the minimum distance. Note that the minimum distance here is going to 8 alpha squared now. OK, so the minimum distance is strictly improved.

But what happens to be your energy per bit? Well, the energy per symbol is 2 alpha squared. But you only have one bit per symbol, so the energy per bit is 2 alpha squared. So your nominal coding gain is d_min squared over 4 Eb, which follows from the relation we have here. And d_min squared is 8 alpha squared. 4 Eb is also 8 alpha squared, so the nominal coding gain is 1, or 0 dB.

The average number of nearest neighbors, well, we only have one nearest neighbor, and we have one bit per symbol, so that's -- and this is A prime by the way -- is going to be 1. And so the effective coding gain is still 0 dB.

AUDIENCE: [INAUDIBLE]

PROFESSOR: So probability of symbol error as a function of alpha has definitely decreased, right. But the trade-off we care about is probability of bit error as a function of Eb N_0. In other words, this curve is still here. It's not changed. I just shifted right by reducing my spectral efficiency. OK?

So let's do one last example just to prove that it's not always the case that you get nothing by removing points. OK, so the contellation that I have in mind is the one I introduced last time. We'll start with a 3, 4 Cartesian product of 2-PAM. OK, so in three dimensions it's a cube, and the 3,4 Cartesian product will have a point on each vertex of the cube. And we only keep four points. So the points we keep are these four points. OK?

So I write B, the coordinates are alpha, alpha, alpha, minus alpha, minus alpha, alpha, minus alpha, alpha, minus alpha, and alpha, minus alpha, minus alpha. So these are the vertices. These are the coordinates of the four points here where we have the standard x, y and z-axis, which I'm not drawing because it will just look confusing.

So now we'll see that for this particular signal set, we do have a coding gain, OK? Let's find the minimum distance. The minimum distance here is what? It's the distance along a diagonal. And the diagonal is of length to -- each side of the cube is off-length to alpha. So the diagonal is of length to alpha. So d_min squared is 8 alpha squared.

What is the energy per bit going to be in this case? Well, what's the energy per symbol now?

AUDIENCE: [INAUDIBLE]

PROFESSOR: 3 alpha squared over 2 is the Eb, right? We have 3 alpha squared units of energy per symbol. We have four points, so we have two bits per symbol. So the energy per bit is 3 alpha squared over 2. So now if I look at my nominal coding gain, that's going to be d_min squared over 4 Eb. d_min squared is 8 alpha squared. 4 dB is 6 alpha squared. So that's 4/3. On a linear scale and on a dB scale, this is going to be? You guys should remember your dB tables.

AUDIENCE: 0.2?

PROFESSOR: 12. You have to multiply by ten. So actually I think it's 1.3, more like 1.3 dB if you look at the last significant digit. OK, so what's K_b of A going to be in this case? Or K_b of B I should say. I'm calling the signal set B here. So what is K_b if B going to be?

AUDIENCE: [INAUDIBLE]

PROFESSOR: 3 by 2, right? You have three nearest neighbors per point, for each point has three nearest neighbors. They're all equidistant. And you have two bits per symbol. OK? So if you work out your effective coding again using this rule of thumb here, take gamma_c of A in dB and subtract 0.2 log2 you will see that the effective coding gain is approximately 1 dB. Are there any questions?

OK, so it might seem that this is still not a very impressive number. Yes, you had a question.

AUDIENCE: What was it about this constellation that gave us coding gain? Was is the face that we went to higher dimensions?

PROFESSOR: Right, so usually you do get a coding gain if you trade-off minimum distance with spectral efficiency. So that's usually the case. So that's why I presented you an example of that case. It's just that this was a very special case where you end up with a 2-PAM constellation to start with.

OK, so the comment I was going to make is that 1.2 dB still might not be like a very impressive number. I mean after all the hard work, you just get 1 dB effective coding gain. So the question to ask at this point is are there any constellations that have much higher coding gains? In particular, other constellations that come close to the Shannon limit at all.

And we'll look at one interesting class of signal sets, which are known as orthogonal signal sets. And you probably saw these examples back in 6.450, but we're just reviewing it again here. And these signal sets have a property that as you increase the number of dimensions, you come arbitrarily close to the Shannon limit. So the definition of the signal sets is your A -- it's a collection of signals aj, where j goes from 1 to M such that the inner product between ai and aj is E A times delta_i,j, meaning that if i is not equal to j, the two signals are orthogonal. And if i equals j, the energy is E(A).

Geometrically, we have two points. The two points can be thought of as two points on the two axes. So this is a case when M equals 2. When M equals 3, the three points will lie on those three corresponding axes. So this is the case when M equals 3 and so on. So generally when you have M points, your signal spatializes in M dimensions, and each point can be thought of as a point on one of the axes.

So if you look at the distance between any two points, it's the norm of ai minus aj squared. I could simplify this simply as the norm of ai squared plus the norm of aj squared because the end product is going to be zero. OK, now ai squared is going to be E(A). aj squared is going to be E(A). So this is 2E(A). So what we're saying here is that every point is equidistant from every other point, and the square of the distance is 2 E(A). OK? So in other words, the average number of nearest neighbors is always going to be a M minus 1.

So let us calculate the nominal coding gain for this constellation. That's going to be d_min squared over 4 Eb. d_min squared is 2E(A) over 4 Eb. Well, the energy per bit is the energy per symbol, which is E(A) over the number of bits per symbol, which is log M to the base 2. And this I'm going to write here is 1/2 log M to the base 2. So as I increase my M, my number of dimensions, my nominal coding gain goes to infinity. OK? What do we expect for the effective coding gain? Will it also go to infinity?

AUDIENCE: It approaches the Shannon limit.

PROFESSOR: Well, we don't know whether it approaches the Shannon limit as of yet, but we know it cannot go to infinity. It's upper bounded by the Shannon limit. Right? So the effective coding gain cannot really become unbounded, and what's really happening here?

AUDIENCE: [INAUDIBLE]

PROFESSOR: Right, but why does the effective coding gain stay bounded and the nominal coding gain blow up?

AUDIENCE: [INAUDIBLE]

PROFESSOR: Right, the number of nearest neighbors also increases with M, right? Not just the nominal coding gain. So let's write that down. If I look at the number of nearest neighbors per information bit -- yes?

AUDIENCE: These don't have 0 mean, right?

PROFESSOR: Right, they don't have 0 mean. That's a good point.

PROFESSOR: Let's see, what happens to the mean as M goes to infinity? If we look at what happens to the mean, it will be 1 over M. You find the mean of this guy, it's 1 over 2 times the distance here times distance here is 1 over 2 E(A), E(A). OK? If you look at three dimensions, it's 1 over 3 E(A), E(A), E(A). So in M dimensions, it's going to be 1 over M times all these coordinates. If we look at the mean squared, it goes to 0 as M goes to infinity. So that's a good point. So the mean does go to 0, and so we approach the Shannon limit.

So we have K_b of A over logM What's the number of nearest neighbors? We saw it was M minus 1 over log M. And that goes to infinity as M goes to infinity. OK, so the average number of nearest neighbors per information bit also go to infinity as M goes to infinity. And if we look at the mean -- just to write down the comment that was made. If we look at the mean of this constellation, that's going to be 1 over the number of points times root E(A) root E(A) and root E(A). So there are M coordinates. So that's going to be the mean if you believe this is the coordinate axis here.

And if you we look at the norm of M(A) squared, then it's going to be E(A) over M. And as M goes to infinity, this goes to 0. And because the mean goes to 0, it's not unreasonable to expect that the performance does approach to the ultimate Shanon limit.

OK, so so far we have looked at K_b of A and gamma_c of A. If you look at the effective coding gain, it is still bounded. It's always bounded. It is not clear at this point what really happens to it. And in fact, it's not quite straightforward to actually show what happens to the effective coding gain. If you have taken 6.450, then Professor Bob Gallager really goes into the details of proving how the effective coding gain for this signal set indeed does approach the Shannon limit.

But it can be shown that the effective coding gain does approach 11.2 dB as M goes to infinity. The main trick here is that union bound is usually quite weak if the probability of error is small. So we replace the probability of error just by 1, instead of the union bound, at some point. And if you do that change, then things work out nicely. But in this course, we will just assert that the orthogonal signal sets do achieve the ultimate Shannon limit. And you can see 6.450 notes for the proof. So that's the assertion that we are making right now.

So let's step back and take a look at what's happening so far. We have a class of signal sets which are conceptually quite easy to describe. They are just orthogonal signal sets. They are not too hard to analyze, and they asymptotically approach the ultimate Shannon limit. So this seems quite nice. So why not we just implement them?

Now it turns that these are not very common in practice. They are not implemented as often as you might think when you first look at them and because they have some drawbacks. And there are a couple of drawbacks which make them very less appealing than what you might otherwise think. And first of all, what's the spectral efficiency for these signal sets?

AUDIENCE: 0, right?

PROFESSOR: It goes to 0. And how does it go to 0? It's 2log M to the base 2 over M. As M goes to infinity, this goes to 0. And in fact, it goes to 0 quite sharply. Now that's fine, but if you want to approach Shannon limit, our spectral efficiency should indeed go to 0. But if you're looking at the system design problem, what do you really want?

You have a certain effective coding gain that you have in mind, and suppose you are presented with a list of several different signal sets that achieve this effective coding gain. And your job is to pick one of these possible signal sets. Which one will you pick?

Well, there are two things you will look for, right? You will look for the spectral efficiency that these different signal sets require or achieve in order to get this effective coding gain. And the higher the spectral efficiency the better. And secondly, you will also want to look at the implementation complexity. You do not want something which really takes a lot of time to decode.

OK? So it turns out that because the spectral efficiency goes to 0 so sharply, it's not very surprising that there are other codes that have the same effective coding gain but higher spectral efficiency. So there are binary codes that have a higher spectral efficiency for same effective coding gain. And so they would be a natural choice over the orthogonal signal set. Yes?

AUDIENCE: [INAUDIBLE] high-spectrum efficiency to mean bandwidth?

PROFESSOR: Right, because in practice, bandwidth is never really infinite, right? So if you are achieving the same -- the first objective is to have a certain effective coding gain. If you do have that, you want to pick something which has a higher spectral efficiency. And indeed, the subject of chapters six and eight is to come up with binary codes that have a much better performance than the orthogonal signal set for a given effective coding gain. Chapter six deals with the the block codes, whereas chapter eight deals with convolutional codes. And chapter seven basically develops all the finite field theory that you require to study convolutional codes. So the point is, in the subsequent lectures we will be focusing a lot on improving the performance over orthogonal signal sets.

The second point I mentioned was implementation complexity. Now one particular way of implementing orthogonal codes, which you saw in the first homework, was using this idea of Hadamard transforms, right? You have a Hardamard matrix, which is an M by M matrix. The rows of the Hadamard matrix are your orthogonal signal sets, which are elements aj.

And at the receiver, what you would do is when you get the signal y, you simply multiply by the Hadamard matrix and look at the coordinate which is the largest. So this is something you did in the first homework exercise. And how many computations did you require for an orthogonal signal set of size M?

AUDIENCE: [INAUDIBLE]

PROFESSOR: Sorry? OK, so I meant M being the size of the orthogonal signal set.

AUDIENCE: Yeah, [INAUDIBLE]

PROFESSOR: Right.

AUDIENCE: 2 to the M by 2 to the M.

PROFESSOR: If it is 2 to the M by 2 to the M, it is M times 2 to the M. But there are M orthogonal signal sets, then it will be M log M, where M is the number of signal sets. So that's the number of computations that you would require.

Now, the question to ask is, is this fast or is this slow? Is this too many computations or is this too little computations? And if you're looking in the power-limited regime, what we really want is to see the complexity per information bit because we normalize things per information bit. Now, how many bits are sent when you have orthogonal signal set of size M? We have log M bits, right? So this quantity is actually exponential in the number of transmitted bits that we are sending, and so in fact this is quite slow. Or in other words we are saying it is we require M times 2 to the M computations, where M is the number of information bits that we are sending.

AUDIENCE: So if M is the size of your [INAUDIBLE] then you have log M bits?

PROFESSOR: Right.

AUDIENCE: So M log M over log M is M.

PROFESSOR: Why are you adding by M?

AUDIENCE: You need M log M operations to decode a symbol.

PROFESSOR: To decode a symbol.

AUDIENCE: And each symbol gives you log M bits. So you have M operations per bit.

PROFESSOR: Well, when you send, say, log M bits, right, you can only send one of the M possible symbols. And when you want to decode the symbol, you would end up recovering log M bits. So you should not be dividing by M here.

AUDIENCE: You're dividing by log M because you want to do it per bit.

PROFESSOR: All right. You're right. So you're dividing by log M. You have M log M bits. You're dividing by log M, so you still have --

AUDIENCE: You have M calculations per bit. [INAUDIBLE]

PROFESSOR: So we are sending -- so I need this many computations to get log M bits. So is this expression exponential in log M or not? So that's all I was saying.

AUDIENCE: I just put it another way that you could divide by log M.

PROFESSOR: M for information, right. OK, are there any questions? I mean we are both agreeing to the fact that this is too many computations than you would want. And there exist other decoding algorithms for which the number of computations is much smaller than exponential.

OK, so let's next look at the bandwidth-limited regime. Well, actually before that I wanted to mention a couple of other signal sets as well. So there are two other important class of signal sets which are related to the orthogonal signal sets. The first is this class of simplex signal sets. And the other class is bi-orthogonal signal sets.

So the idea behind simplex signal sets goes back to the observation that was made that we have a non-zero mean for this signal set here. So if we do, we can indeed subtract of the mean and get a performance which is better than the orthogonal signal set. So in particular if I subtract of the mean here, I get a signal set, which is like this. This is the simplex signal set when M equals 2. When M equals 3, I'm going to subtract of the mean from this plane here. The points lie on this particular plane. And what I end up with is an equilateral triangle. So this is the case when M equals 3.

What do you think M equals 4 would look like? A tetrahedron right here. That's a simplex signal set. So basically, it's also not too hard to write an algebraic expression for these signal sets. E of A prime is going to be E of A minus M of A. You subtract off the mean from your orthogonal signal set. If I want to write it in terms of inner products, the inner product between ai and aj is given by E(A) if i equals j. And it's I believe minus of 1 over M minus 1 times E(A) if i is not equal to j. So this is the inner product between ai and aj. And I mean there are also many properties of this simplex signal set, but I believe that that's going to be in your next homework exercise so I'm not going into too many details.

What' the spectral efficiency in this case going to be?

AUDIENCE: I have a question. So what is a simplex? Is a simplex just a reference to anything with [INAUDIBLE].

PROFESSOR: Simplex signal set is derived from the orthogonal signal set by subtracting off its mean.

AUDIENCE: Right, but in general simplex means anything [INAUDIBLE]?

PROFESSOR: I'm not sure about that. What's the spectral efficiency going to be?

AUDIENCE: [INAUDIBLE]

PROFESSOR: Well, almost.

AUDIENCE: You use one dimension.

PROFESSOR: Right, you just use one dimension. OK, and there are other properties that you'll be looking at it in the homework. Now the idea behind this --

AUDIENCE: [INAUDIBLE]

PROFESSOR: This is 2 log M over [INAUDIBLE]. So the idea behind the bi-orthogonal signal set is you start with an orthogonal signal set, and for each point, you also put the negative signal of it. So in addition to this, you will have a point here, and you will have a point here. So the case in two dimensions, M equals 2, you will have four points like this. So in this case you are increasing the number of signal points by a factor of two. But this does come at a price, right? Because if you're increasing the number of signal points, the number of nearest neighbors is also going to increase by a factor of two. In this case, you have two nearest neighbors now. So you have both of the effects going on. I mean ultimately, all the signal sets, as M goes to infinity, are going to approach the Shannon limit. So again, but they do suffer from the same kind of drawbacks that we saw for the orthogonal signal sets. So these are more of theoretical interest as opposed to real, practical implementation points. Any questions?

AUDIENCE: Minus [INAUDIBLE]

PROFESSOR: It's a bit involved to show. I think you'll be showing it in the homework exercise or for an exam problem at one point. You basically start with the orthogonal signal sets here, look at the inner product here, and use the relation between the simplex signal sets and the orthogonal signal sets to do a subtraction of the mean. And then you can derive in a product expression.

OK, so let's now move on to the bandwidth-limited regime. OK, so the main difference between -- yes?

AUDIENCE: [INAUDIBLE]

PROFESSOR: You mean here? For each --

AUDIENCE: How far can you [INAUDIBLE]?

PROFESSOR: A factor of two. I said a factor right? So if you have M points in the orthogonal signal set, you have 2M points in the bi-orthogonal signal set because for each --

AUDIENCE: In the orthogonal sets you have 3.

PROFESSOR: And in this you have 6. So let's look at the bandwidth-limited regime. The trade-off we care about is the probability of error for two dimensions as a function of SNR norm. OK? The baseline scheme here is your M-PAM scheme. And we showed a couple of lectures ago that the probability of error for two dimensions is approximately 4 Q times root 3 SNR norm. OK?

So let's plot the performance graph. So on the x-axis I plot SNR norm. On the y-axis I plot Ps of E again on a log-log scale. This is ten to the negative six, ten to the negative five, ten to the negative four, ten to the negative three and so on. The intercept is at 0 dB, this point here. So this is your Shannon limit in the bandwidth-limited regime.

Now your performance curve is going to be something like this for the M-PAM system. And we want to basically quantify now the effective coding gain in this regime. So it's the same story as in the power-limited regime. We will start off with the probability of error using the union bound estimate, which is K_min of A times Q of d_min over 2 sigma. Now Ps of E is the probability of error for two dimensions. Assuming we have N dimensions here, it is two times the probability of error over N. So this is probability of error per symbol. We are dividing by the number of dimensions and multiplying by two because it is for two dimensions. And this is going to be 2 times K_min of A over N, times Q of d_min over 2 sigma.

So now let's do the same trick that we did in the power-limited regime. We will write Ps of E be approximately -- and instead of the right hand side, I will write it as Ks of A times Q of root 3 SNR norm. But because I have a factor of four up there, I will also multiply and divide by 4. That's not going to change anything.

AUDIENCE: [INAUDIBLE] per two dimensions or by symbol.

PROFESSOR: Per two dimensions. And for bandwidth-limited regime, we normalize everything by two dimensions.

AUDIENCE: But for N band we used it for symbols, right? The calculations represent --

PROFESSOR: No, it was for two dimensions. OK?, so now let's compare the two expressions. We have Ks of A, which we'll define by two times K_min of A over N. And this is the number of nearest neighbors per two dimensions. And I can write 3 times SNR norm is d_min squared over 4 sigma squared, OK? Actually, I was missing this factor of gamma. I knew I was missing something. It's not just 3 SNR norm. It's 3 times this nominal coding gain times SNR norm. So 3 times SNR norm times gamma_c of A here. So this is d_min squared over 2N_0. So what do I have for the gamma_c c o A is d_min squared over 6 N_0 times SNR norm. Remember SNR norm is SNR over 2 to the rho minus 1. It's normalized signal-to-noise ratio.

So this is the d_min squared times 2 to the rho minus 1 over 6 times N_0 times SNR. But N_0 times SNR is just Es. So this is 6 times Es. So this is the expression you have for the nominal coding gain.

So now again, given a signal set A, I can find those two parameters, the nominal coding gain and the number of nearest neighbors per two dimensions. And I can use those to plot on the Ps of E versus SNR norm curve.

Again, the exact same story. If I have a certain nominal coding gain, I will simply shift my curve around the x-axis by that factor as a pure translation. And then because I have a factor of Ks A over 4, I'm going to plot -- how did that expression go? This expression on the top. Remember this curve here is 4 times root 3 SNR norm, right? So the multiplicative factor I have is Ks(A) over 4. So I will shift my curve up by that factor. And I will get something which is like this. And that's my union bound estimate for this new constellation A.

The distance here is gamma effective. This distance here is gamma_c of A. Are there any questions? So now we also have a similar rule of thumb as in the power-limited regime. I will write that rule of thumb here. We have that gamma effective is going to be gamma_c in dB minus 0.2 times log2 times Ks of A over 4 in dB. OK, so again, a factor of 2NKs will decrease your nominal coding gain by a factor of 0.2 in dB. Are there any questions?

Well, so I mean this is the end of chapter five. We still have like ten minutes remaining, so I thought I would give you a preview of the next chapter. Professor Forney will be coming next class, and he will be starting chapter six all over I believe. This chalk is much nicer to erase than that other one.

So in chapter six, we'll be looking at the binary block codes. OK, and what is the main architecture of these codes? Well, for an encoder, remember we have K bits coming in. And we pass it now through a binary block code. And what we get out is a binary sequence of length N. OK? So x belongs to c, where c is a subset of binary sequences of length N.

So we start with K bits, and we convert them to N bits, where N is going to be greater than or equal to K. Once we have this sequence of N bits, what we end up doing is we pass it through a binary signal constellation. So I'm writing in as binary signaling. And what we get out is S of x, which is a sequence of N symbols. Each symbol can either take value minus alpha or alpha.

So this binary signalling is actually quite a trivial step. Basically, the relation is S of 0 is going to be alpha, S of 1 is going to be minus alpha. If I have a 0 coming in, I will produce an alpha. if I have a 1 coming in, I will produce a minus alpha. So this part here is trivial. All our energy will be focused on will be to find a good code. Given a sequence of input bits, we want to find a good binary code in order to optimize the minimum distance and so on. So that's the architecture that we will be using for the binary linear codes. Both chapters six and eight will be only focusing on this part. And this binary signalling scheme, for the most part, will be implicit in our architecture.

So at this point, one might ask is there anything to lose in having imposed this type of architecture? Remember, the nice thing about this architecture is we have coding, which is going on here, and we have modulation that is happening here. And the modulation step is quite simple. So we have a separation between coding and modulation. OK?

And the question is, is this the right thing to do? Now, it turns out that if we are going to live in this binary world where we are only constrained ourselves to sending binary signals over the channel, this is in fact an economical structure. There isn't much improvement we can get, and that is quite obvious, right? But it turns out that if you are going to go for non-binary signaling, particularly in the bandwidth-limited regime, there is a cost to be paid for this kind of separation. In fact, in the 1980's there was this whole idea of turbo-coded modulation or trellis-coded modulation. And the idea there was to combine coding and modulation in one step. So this is not optimal for non-binary signalling, but's it's OK for binary. So we will be focusing on this type of an architecture.

The other issue is OK, nobody told us that you can only send binary signals over the channel. If you want to achieve the Shannon limit, when we derived the capacity expression -- or we just stated it but you can derive it. We said that the Shannon limit is always less than log2 1 plus SNR. And we never said that we can only send binary signals over the channel. So it could be that this is some inherence of optimality in this type of architecture. Why send binary signals in the first place?

Again, it turns out that the regime that we are interested in here is the power-limited regime because the spectral efficiency can never be more than two bits per two dimensions, right? Remember, rho is 2K over N here, and K is going to be less than that.

So we are inherently in the power-limited regime. And so a natural thing to do in order to understand how much fundamental loss we have in imposing this type of binary constraint over the signals is to not look at this expression, but to find the best possible performance we can have if we constrain ourselves to binary signals over the channel. If we impose a certain signaling constraint, we can find another fundamental limit on the spectral efficiency. So the point being, the best possible binary code can only achieve this upper bound here. Now, this upper bound has to be less than log2 1 plus SNR, right? Because the Shannon code assumes the binary signaling is a special case.

So it turns out that this expression is actually quite tedious to write out. I'm not even sure if the closed-form expression exists, but you can calculate this numerically. It's just some kind of a convex optimization problem.

And now, to understand how much loss we have, we can compare the two expressions in the power-limited regime. So I'm going to plot the curves for the two cases. I'm going to plot the spectral efficiency on the y-axis as a function of Eb N_0. You can easily convert from SNR to Eb N_0 using the relations we discussed in chapter four.

Now we first plot the Shannon limit. The ultimate Shannon limit is minus 1.59 dB. This is going to be 0 dB here. And if I'm plot the Shannon limit, it will be some code like this. It increases with Eb N_0. And minus 1.59 dB is 0. This point here is 1. Some point here is 2. So actually I want to draw this horizontal asymptote at two when rho Shannon equals 2.

So this is my rho Shannon. Now, I can now also plot rho binary from this expression here. I can never exceed two bits per two dimensions. So I had plotted this horizontal line. So as Eb N_0 goes to infinity, the most I can get is two bits per two dimensions. If Eb N_0 is smaller, I cannot do better so I will be doing worse and worse, and I will always be below this line. Ultimately, it can be shown that I will have the same Shannon limit here. So this is rho binary.

And the main observation here is if I'm in the power-limited regime, which is say, in this part of the curve -- like, say around one bit per two dimension -- the gap here is quite small. In fact, this gap can be shown to be at most 0.2 dB. So if I want to achieve a certain spectral efficiency, if I impose a binary signaling constraint, the additional Eb N_0 that I require when the spectral efficiency is 1 dB, one bit per two dimensions, is at most 0.2 dB. So there is not much to lose by imposing a binary signaling constraint.

Said in different words words if you had the best possible binary linear code, followed by a binary signaling constellation, the most you can lose is 0.2 dB. And so this type of an architecture does make sense. There are a lot of details that the next powerpoint will be to kind of formalize the notion of a binary block code, then specialize it to the case of binary linear codes and so on.

Unfortunately, the first thing to do is to start with some finite field theory because there's a lot of finite field algebra involved in this algebraic block codes. So how many of you have seen finite field theory before? OK, if you haven't seen it, don't worry about it. We'll be starting right from the basics. But I think I will be stopping here for today because I don't want to go into those details today.