PROFESSOR: I want to start out today by reviewing what we covered last time. We sort of covered two lectures of material, but a little bit lightly last time because I want to spend more time starting today dealing with wave forms and functions. We will get notes on that very, very shortly.

To review quantization, we started out by talking about scalar quantizers, in other words, the thing that we want to do is to take a sequence of real numbers and each of those real numbers we want to quantizers it into one of a finite set of symbols. Then, of course, the symbols we're going to encode later on. So basically what we're doing is we're taking the real line, we're splitting it into a bunch of regions, r1, r2, r3. The last region of the first region goes off to minus infinity. The last region goes off to plus infinity. So that clearly, if there's a lot of probability over in here or a lot of probably over in here, you're going to have a very large distortion. So, when we talk more about that later and talk about how to avoid it and why we should avoid it and all of these things.

We then talked about these Lloyd-Max conditions for minimum mean square error. What we said last time was suppose somebody gives you these representation points, which you're going to use to represent the actual number on the real line that comes out. Then you ask when some particular number occurs should we encode it into this point or into this point? If our criterion is mean square error, and that's what our criterion is normally going to be, then we're going to minimize the mean square error for this particular point by mapping it here, if that's the most probable -- we're going to map it here if this is the closest point, and we're going to map it here if that's the closest point. Because by doing this we minimize the squared error between b and a1 or b and a2.

So, that says that we're going to define these regions to have the separations between the regions at the bisector points between the representation points. So that says that one of the Lloyd-Max conditions for minimum mean square error is you always want to choose the regions in such a way that they're the mid-points between the representation points. Any minimum mean square error quantizers has to satisfy this condition for each of the j's. Namely, for each of these points they have to be mid-points.

Then the other thing that we observed is that once we choose these regions, the way we want to choose the representation points to minimize the mean square error is we now have to look at the probability density on this real line and we have to choose these points to be the conditional means within the representation area. That just comes out by formula to be the expected value of the random variable u, which is this value as it occurs on the real line. The expected value of that conditional on being in region rj is just the integral of u times the conditional density of u. The conditional density of u is the real density of u divided by the probability of being in that region.

So all of this is very simple. I hope you see it as something which is almost trivial, because if you don't see it as something simple, go back and look at it because, in fact, this is not rocket science here, this is just what you would normally do.

So Lloyd-Max algorithm then says alternate between the conditions for the mid-points between the regions and the conditional means. The Lloyd-Max conditions are necessary but not sufficient. In other words, any time you find a minimum mean square error quantization, it's going to satisfy those conditions. But if you find a set of points, b sub j, and a set of points, a sub j, which satisfy those conditions, it doesn't necessarily mean that you have a minimum. In other words, there are often multiple sets of points which satisfy the Lloyd-Max conditions, and one or more of those is going to be optimum, is going to be the smallest one, and the others are not going to be optimum. In other words, the algorithm is a local hill-climbing algorithm, which finds the best thing it can find which is close to where it's starting at some very strange sense of close. The close is not any sense of mean square

error, but close is defined in terms of where the algorithm happens to go.

So an example of that that we talked about last time is where you have three spikes of probability. Two of them are smaller and one of them is bigger. One of them is at minus 1, one of them is at zero, one of them is at plus 1. One solution to the Lloyd-Max conditions is this one here where a1 is sitting right in the middle of the spike. Therefore, any time that the sample value of the random variable is over here, you get virtually no distortion. The other point is sitting at the conditional mean between these two points. So it's a little closer to this one than it is to this one -- I hope the figure shows that. Any time you wind up here or here, you get this amount of the distortion.

Now without making any calculations you just look at this and you say well, this spike is bigger than this spike is. Therefore, it makes sense if we're going to do this kind of strategy to put a 2 underneath that spike, therefore, getting a very small distortion any time the big spike occurs. Then a1 is going to be midway between these two points, and you get the larger amount of distortion there but now it's a less probable event. So you can easily check that both of these solutions satisfy the Lloyd-Max conditions, but one of them turns out to be optimal and the other one turns out to be not optimal. If you fiddle around with it for a while, you can pretty much convince yourself that those are the only solutions to the Lloyd-Max algorithm for this particular problem.

Yeah?

AUDIENCE: When there's a region that has zero probability throughout, and the Lloyd-Max algorithm tries to find the mean for that region, it's going to find somewhere outside the region, it will find zero as the expected value. But that might not necessarily be inside that region, what does it do in that case?

PROFESSOR: What does it do in that case? Well, I don't think you can argue that it's going to be at zero. I think you have to argue that it might be anywhere that it wants to be. Therefore, what the algorithm is going to do when you start with a certain set of representation points -- well, if you start with a certain set of representation points

that picks that separater wherever it happens to be, than this particular point you're talking about is going to be at some completely unimportant place. You know eventually the thing that's going to happen is that this thing that's in a region of no probability is going to spread out and include something that has some probability, and then you're going to nail that region with some probability. I can't prove this to you, and I'm not even sure that it's always true, but I think if you try a couple of examples you will see that it sort of does the right thing.

AUDIENCE: But in the algorithm, you replace the point at the point of expected value in that region. So, the algorithm doesn't know what to do at that point. It crashes.

PROFESSOR: Well, unless you're smart enough to write the program to do something sensible, yes.

AUDIENCE: [UNINTELLIGIBLE PHRASE].

PROFESSOR: Yes. And you have to write it so it'll do something reasonable then. The best thing to do is to start out without having any of the regions have zero probability.

AUDIENCE: We have that [UNINTELLIGIBLE PHRASE].

PROFESSOR: All right. Well then you have to use some common sense on it.

So, after that we say OK, well just like when we were dealing with discrete source coding, any time we finish talking about encoding a single letter, we talk about what happens when you encode multiple letters in the same way. Somebody is bound to think of the idea of encoding multiple real numbers all together. So they're going to think of the idea of taking this sequence of real numbers, segmenting it into blocks of n numbers each and then taking the set of n numbers and trying to find a reasonable quantization for that. In that case, the quantization points are going to be n vectors. The regions are going to be regions in n dimensional space.

Well, if you think about it a little bit, these n dimensional representation points, if you're given them, the place where you're going to establish the regions then is on the perpendicular bisectors between any two points. Namely, for each two points

you're going to establish a perpendicular bisector between those two points, you're going to do that for all sets of points. You're going to take regions which are enclosed by all of those perpendicular bisectors, and you call those the voronoi region. Remember, I drew an example of it last time that looked something like this. You have various points around. It's hard to draw it in more than two dimensions. So these perpendicular bisectors go like this and so forth. I think you can show that you've never had the situation -- interesting problem if you want to play with it. I don't think you can have that, but I'm not sure why.

Anyway, you do have these voronoi regions. You have these perpendicular bisectors that you set up in two dimensional space or in high dimensional space. Then given those regions you can then establish representation points, which are at the conditional means within those regions. You really have the same problem that you had before, it's just a much grubbier problem because it's using vectors, it's an n dimensional space. For this reason this problem was enormously popular for many, many years, because many people loved the complexity of it. It was really neat to write computer programs that did this. Back in those days you had to be careful about computer programs because computation was very, very slow, and it was a lot of fun. When you get all done with it, you don't gain much by doing any of that. The one thing that you do gain is that if you take square regions, namely, if you take a whole bunch of points which are laid out on a rectangular grid and you take regions which are now little rectangles or little squares, and you look at them for a while, you say that's not a very good thing to do.

A better thing to do is to take all this two dimensional space, for example, and to fill it in to tile it we say with hexagons as opposed to tiling it with rectangles or to tiling it with squares. If you tile it with hexagons, for given amount of area you get a smaller mean square error. If you could tile it with circles that would be the best of all, but when you try to tile it with circles you find out there's all this stuff left in the middle, like if you've ever tried to tile a floor with circles you find out you have to fill it in somehow and it's a little bit awkward. So hexagons work, circles don't. If you then go on to a higher number of dimensions, you get the same sort of thing happening, you get these nice n dimensional shapes which will tile n dimensional volume. As n

gets larger and larger, these tiling volumes become closer and closer to spheres, and you can prove all sorts of theorems about that. But the trouble is when you get all done you haven't gained very much, except you have a much more complex problem to solve. But you don't have a much smaller mean square distortion.

So you can still use Lloyd-Max. Lloyd-Max still has as many problems as it had before in finding local minima. With a little bit of thought about it you can see it's going to have a lot more problems. Because visualize starting Lloyd-Max out where your points are on a square grid and where your regions now are a little square. So in other words, like this. Try to think of how the algorithm is going to go from that to the hexagons that you would rather have. You can see pretty easily that it's very unlikely that the algorithm was ever going to find its way to hexagons, which by looking at it a little further away we can see it's clearly a good thing to do. In other words, Lloyd-Max algorithm doesn't have any vision. It can't see beyond its own nose. It just takes these points and looks for regions determined by neighboring points, but it doesn't have the sense to look for what kind of structure you want. So anyway, Lloyd-Max becomes worse and worse in those situations and the problem gets uglier and uglier.

Then, as we said last time, we stop and think and we said well gee, we weren't solving the right problem anyway. As often happens when a problem gets very popular, people start out properly by saying well I don't know how to solve the real problem so I'll try to solve a toy problem. Then somehow the toy problem gets a life of its own because people write many papers about it and students think since there are many papers about it, it must be an important problem. Then since there are these open problems, students can solve those open problems and get PhD theses, and then they got a in a university, and the easiest thing for them to do is to get 10 students working on the same class of problems and you see what happens. I'm not criticizing the students who do that or the faculty members who do it, they're all trapped in this kind of crazy system.

Anyway, the right problem that we should have started with is when we look at the problem of quantization followed by discrete source coding, we should have said

that what we're interested in is not the number of quantization levels, but rather the entropy of the set of quantization levels. That's the important thing because that's the thing that determines how many bits we're going to need to encode these symbols that come out of the quantizer. So the problem we'd like to solve is to find the minimum mean square error quantizer for a given representation point entropy. In other words, whatever set of points you have, you want to minimize the entropy of that set of points. What that's going to do is to give you a larger set of points, but some points with a very small probability. Therefore, those points with a very small probability are not going to happen very often. Therefore, they don't affect the entropy very much, and therefore, you get a lot of gain in terms of mean square error by using these very improbable points.

That's a very nasty problem to solve. And again, we said well let's try to solve a simpler version of it. A simpler version of it is first to go back to the one dimensional case and then say OK, what happens if we just use a uniform quantizer, because that's what most people use in practice anyway. If we use a uniform quantizer and we talk about a high rate uniform quantizer, in other words, we make the quantization points close together, what's going to happen in that case? Well, the probability of each quantization region in that case is going to be close to the size of the representation interval, in other words, of the quantization interval, times the probability density within that interval. Namely, if we have a probability density and that probability density is smooth, then if you take very, very small intervals you're going to have a probability density that doesn't change much within that interval. Therefore, the probability of the interval is just going to be the size of that quantization interval -- in a uniform quantizer you'll make all of the intervals the same -- times the density within that integral. Then we say OK, let's look at what the entropy is of that set of points, of the set of points where the probabilities are chosen to be some small delta times the probability density there. I'm going through a slightly simpler kind of argument today than I did last time, and I'll explain why I'm doing something simpler today and why I did something more complicated then.

So this entropy is this quantity. If we now substitute delta times the density for pj

here, we get the sum over j of this delta pj, which is the probability density times the logarithm of delta pj. Well now look, the logarithm of delta pj is just logarithm of delta plus logarithm of pj. So we're taking the sum over all the probability space of logarithm of delta. That comes out. So we get a minus log delta, and what's left is minus the sum of delta pj log pj. Does that look like something? That looks exactly like the approximation to an integral that you always talk about. Namely, if you look at a Reimann integral, the fundamental way to define a Reimann integral is in terms of splitting up that integral into lots of little increments, taking the value of the function in each one of those increments, multiplying it by the size of the increments and adding them all up.

In fact, we're going to do that a little later today when I try to explain to you what the difference is between Reimann integration and Lebesgue integration. should Don't be frightened if you've never taken any mathematics courses, because if people had taught you Lebesgue integration when you were freshmen at MIT or seniors in high school or whenever you learned about integration, it would have been just as simple as teaching about Reimann integration. One is no simpler and no more complicated than the other, so we're really going back to study something you should have learned about five years ago, maybe.

So anyway, when we represent this as an integral, we get this thing called the differential entropy, which is the integral of p of u minus p of u times log of p of u. So the entropy of the discrete representation is minus log delta plus this differential entropy. The mean square error in this uniform quantizer, the conditional means according to this approximation are right in the middle of the intervals. So we have a uniform probability interval of width delta, a point right in the middle of it, and even I can integrate that to find the mean square error in it, which is delta squared over 12, which I think you've done at least once in the homework by now.

So I said I was going to tell you why I went through doing it this simpler way this time and put in a lot more notation last time. If you really try to trace through what the approximations are here, the way we did it last time is much, much better, because then you can trace through what's happening in those approximations, and you can

see, as delta goes to zero, what's happened.

Yes?

AUDIENCE: This may be an obvious question, why did you substitute delta with pj [INAUDIBLE PHRASE]?

PROFESSOR: Oh, why did I--? OK, this is the probability of the representation of the j's representation point. This is the probability density around that representation point. The assumption I'm making here is that f of u was constant over that interval. And if the density is constant over the interval, if I have a density which is constant over an interval of width delta, than the probability of landing in that interval is the width times the height.

AUDIENCE: I think there's a typo in your [UNINTELLIGIBLE PHRASE].

PROFESSOR: A typo?

AUDIENCE: [UNINTELLIGIBLE PHRASE].

PROFESSOR: Yes, yes, yes. I'm sorry, yes. I'm blind today. I knew what I meant so well that I didn't -- thank you. s of uj. s of uj. Yes. Then I take out the delta and what I'm left with is the delta f of uj times the log of f of uj. Thank you. When I look at that it's delta times the probability density times the log of the probability density. If I convert that now into an interval when delta is very small, I get this thing called the differential entropy. Does that make a little more sense? So your question was obvious, it was just that I was a total dummy.

So let's summarize what all of that says. In the scalar case we're saying -- I have said but I have not shown -- that a uniform scalar quantizer approaches an optimal scaler quantizer. I haven't explained it all in class why that's true. There's an argument in the notes that points it out. You can read that there. It's just another optimization, but it's true if you're looking at a higher and higher rate, a scaler quantizer where delta gets smaller and smaller, then in general what you need is to take a different size delta for each quantization region and then look at what

happens when you try to optimize over that and you find out that you want to make all of the deltas the same.

The required number of encoded bits per symbol depends only on h of u and on delta. This is the most important part of all of this. It says that as you change this differential entropy, if you try to draw a curve between H of v and MSE, and there's a curve like that drawn in the notes, if you change the differential entropy, it just shift this curve left and right. For a given value of h of u, this is a universal curve. In other words, as you change delta, this quantity changes and this quantity changes. That's the only variable which is left in here at this point. When you make delta half as big, if you want to get a higher rate quantizer, what happens? Your mean square error goes down by a factor of four. Delta squared goes down to 1/4 of its previous value.

What happens here, at log of delta? Delta has changed by a factor of 1/2. H of v goes up by one bit. So you take one more bit in your quantizer and you get a mean square error, which is four times as small. Any time you think of what kind of accuracy you need on a computer or something, I think this is obvious to all of you, if you put it in terms of something you're already familiar with. If you use 16 bit quantization with fixed bit numbers and then you change it to 24 bit accuracy, what's going to happen? Well, everything is going to get better by a factor of 256, and since we're talking about mean square error, it's going to be four times that. So that's just saying the same thing that you know.

For vector quantization, uniform quantization again approaches optimal for a memoryless source. If you have a source with memory, vector quantization gains a great deal for you. But if you don't have any memory, vector quantization doesn't gain much at all. The only thing that vector quantization gains you is this thing we call a shaping gain now. We talk about that again when we start talking about modulation. If you change from a square set of points to a hexagonal set of points and you keep the areas the same, the mean square error goes down by a smidgen -- something like 1.04 or something. It's not a big deal but there's some gain, so the gain is not impressive. The big gains come when you look at the memory and when you take that into account.

So now we want to get on to the last part of our trilogy when we're talking about source coding. Remember, when we were talking about source coding, we broke it up into three pieces. The first piece we called it sampling, which took a wave form, turned it into a sequence of numbers. That's what happens here. We then quantize the sequence of numbers, either one number at a time or with a vector quantizer n numbers at a time. We just finished talking about that. The first five lectures in the course were all talking about discrete encoding, and whenever you're going from wave forms to bits, you gotta go through all three of these.

Now, sampling is only one way to go from wave form to sequence, and filtering is only one way to get back. We're going to talk about sampling. We're probably going to teach you more about sampling than you ever wanted to know. But it turns out that it's worth knowing. After you understand it you never forget it. There's a lot of stuff to go through to start with, but finally, I hope, it all makes sense. But anyway, the thing we're going to be talking about today is really the question of how do you go from wave forms to sequences, it's that simple. How do you in general take wave forms, turn them into sequences? How do you go back from sequences to wave forms? We're going to spend quite a bit of time on this. We're going to spend three lectures talking about it, and probably with today thrown in it'll be closer to three and a half lectures. It's not only because we want to talk about the problem with source coding, because as soon as we start talking about channels, we're going to have the same truck problem looked at in the opposite direction. We're going to start out with binary data. We're then going to go through a modulator, we're going to find symbols. From the symbols, from the numerical symbols, we're talking about a sequence of things and we have to go from the sequence to wave forms. So, both of those problems are really the same. We're talking about it first in terms of source coding, but whatever we learn about wave forms to sequences will be general and will be usable for both.

So I want to review why it is that we want to spend so much time on this analog source to bit stream problem. I just told you one of the reasons which is not here, which is that it's a good way to get into the question of what do we do with channels.

But the other reasons, and we've talked about them all, and they're all important and you ought to remember them, because often we get so used to doing things in a certain way that we don't know why we're doing them and then somebody suggests something else and we say oh, that's a terrible idea because we've always done it this way.

One of the reasons why we want to go to bits is that a standard binary interface separates the problem of source and channel coding. This was, in a sense, one of Shannon's great discoveries, and he also showed that you could do it without really any loss. Another reason is you want to multiplex data on high speed channels. This is perfectly familiar to you. I think to everyone today we think of sending data over the web and we're all used to using the web all together. I send my stuff, you send your stuff, I get my stuff off, you get your stuff off, and this stuff was all going over common channels. It's going over optical fibers into MIT, and then it splits up at MIT and goes into many places and then it goes many places again. But this idea of multiplexing data is perfectly straightforward. If we didn't do any of this, if all of my stuff was really wave forms and all of your stuff was images and if all of somebody else's stuff was data and every piece of the internet had to worry about all those different things, when you start worrying about all those different things you create an awful lot of other things also. We just wouldn't have any internet today.

So this multiplexing is a big deal, too. You can clean up digital data at each link in a network. In other words, if I'm sending analog data from here to San Francisco and I'm sending it over multiple different links, on every link a little bit of noise gets added to it. That noise keeps adding up because there's no way to clean it up, because nobody knows what I sent. If I'm sending digital data, at the receiver on each link, nobody knows what I sent, no, but they know that what I sent was one out of a finite collection of things. There's something called repeating going on there at every channel, which takes what is received as an analog signal and, in fact, knowing what the encoding process was, goes back to cleaning it up to a digital signal again. If you believe all of that and if you think it's simple, it's not. We're going to talk about it later. At this point, it's only plausible, and we're going to justify it as we move on.

We can separate problems of wave form sampling from quantization from discrete source coding. In other words, we not only have the layering between sources and channels, but we also have this layering for sources, which goes between wave form to sequence separation, then sequence and quantization into a finite set of symbols. Then a finite set of symbols getting coded. So, three separate things we've learned about, we can separate them all very nicey.

So we said that in this wave form, the sequence business, sampling is only one way to go. I'm going to show that to you right away at the beginning by talking about Fourier series. How many of you have studied Fourier series and say spending more than a couple of hours of your life thinking about Fourier series? OK, good, quite a few of you, that's nice. Because we have to assume that you know a little bit about this, but probably not a whole lot. There's a formula for a Fourier series, which is probably not the formula for a Fourier series that you're used to. It says the Fourier series of a time-limited function matched the function to a sequence of coefficients. Here's the formula. Here's the function. You can represent the function as a sum of coefficients times these complex exponentials. You do that over this interval minus capital T over 2 less than or equal to t, less than or equal to capital T over 2. The complex coefficients satisfy this equation here. That's just what you've seen before.

The way this is different from what you've probably seen before is that most people think that you use Fourier series for periodic functions. In other words, if we leave out this part here, leave out this, then this quantity here is a periodic function, because each of the what have you are all squiggling around with a period which is a sub-multiple of capital T. So that, in fact, this is a periodic function with period t. If I think of it as a periodic function I don't have to worry about this, this still works for any periodic function. The problem is this isn't the way the Fourier series is usually used. Occasionally, you want to talk about periodic functions, but most often what you want to do is you want to take a function which exists only over some finite interval and you want some way of mapping that function into a set of coefficients. I take a function only over the interval minus t over 2 to capital T over 2, and I can

map that into a sequence of coefficients, I have, in fact, done what I said we're interested in doing right now, which is turning a wave form into a sequence. The only problem with it is it's a fine duration wave form, which I'm turning into a sequence.

Now how do you do speech coding? There's an almost universal way of doing speech coding now of turning speech, analog wave forms, into actual data, into binary data. The way that it always starts, I mean everybody has their own way of doing it, but almost everyone takes the speech wave form and segments it into 20 millisecond intervals. Each 20 millisecond interval is then encoded into a sequence of coefficients. You can think of that as taking each 20 millisecond interval, creating a Fourier series for it, and the Fourier series coefficients then represent the function in that interval. You go on to the next interval, you get another sequence of Fourier coefficients and so forth. Now, most of these very sophisticated voice coders don't really use the Fourier series coefficients because there's a great deal of structure in voice, and the Fourier series is designed to deal with any old thing at all. But the Fourier series is a good first order approximation to what's going on when you're dealing with voice coding. when you're dealing with voice coding you are certainly looking at frequencies, you're looking at formats, which are ranges of frequencies. If you want to think about those problems, you better start to think in these ways here. So anyway, this is not just mathematics, this is one of the things that we need to understand how you do actual wave forms to sequences.

We're not going to talk too much about where these formulas come from too much. It is interesting that this also works for complex functions as well as real functions. There's a nice sort of symmetry there, because the coefficients are all going to be complex anyway, because of these things here which are complex. Incidentally, we always use i in this course for the square root of minus 1. Electrical engineers have traditionally used the letter j for the square root of minus 1 for the rather poor reason that they like to refer to current as i. In the first two years of an earlier electrical engineering education back 50 years or so ago, you spent so much time talking about voltages and currents that using the letter i for anything other than

current was just an abomination. Well, everybody else in the world uses i for the square root of minus 1. So in this course we're going to do the same thing. I would urge you to get used to it because then you can talk to people other than electrical engineers, and you'll probably have to spend a lot of time in your life talking to other people. You shouldn't expect them to get used to your conventions, you should try to do a little to get used to their conventions. So there's that peculiarity.

We're also using this complex notation throughout. You could do this in terms of sines and cosines, which is probably the way you first learned it. I'm sure for any of you who spent more than a very, very small amount of time dealing with Fourier series, you did enough with it to realize that just computationally going from sines and cosines to complex exponentials just makes life so much easier and makes your formula so much shorter that you want to do it. So anyway, we want to make this work for complex signals as well as anything else.

I do want to verify the formula for these Fourier coefficients. Incidentally, the other thing that I'll be doing which is a little bit weird here is that most people when they talk about the Fourier integral and the Fourier series they use capital letters to talk about frequencies and they use little letters to talk about signals. For us, we really want to use capital letters to talk about random variables, and we do that pretty consistently. Believe me, when we start talking about random processes, you will get so confused going back and forth between sample values and random of variables, that having a notation way to keep them straight will be very valuable to you. When you start reading the literature you get even more confused because most people in the literature don't tell you what it is that they're talking about and they go back and forth between sample values and random variables, oftentimes using the same symbol in the same sentence for two different things. So I think that's a more important thing to keep straight, so we'll always use tildes to talk about frequency type things. You can see that these coefficients here are, in fact, frequency-like things because they're talking about how much of this wave form is at a certain discrete frequency, and we'll come back to talk about that later.

But anyway, if you want to verify the formula for this, what we're going to do is to

start out by looking at -- this is where having smaller data would be a big help. u of t is equal to this sum here. So I'm going to replace u of t in this formula by this sum. I'm going to make the index m because I already have a k over here. When we have a k over here and you're talking about this, you don't want to get your indexes mixed. So if I'm trying to see what this looks like, I want to represent as the integral from minus t over 2 to plus t over 2 of this representated as a sum with e to the minus 2 pi i kt over t taken into account over here. So what happens here? Here we have an integral of a sum. Later on we're going to be a little bit careful about interchanging integrals and sums, but for now let's not worry about that at all. I suggest to all of you, never worry about interchanging integrals and sums until after you understand what's going on. Because if you start asking about that -- I mean that's a detail. You look at what's going on in a major way first, and then you go back to check that sort of thing out.

So when we look at this integral here, when we take the sum outside, we have the sum over m of an integral over one cycle of these quantities here, of this times e to the 2 pi i times m minus kt over t. Now, you look at this integral here of a complex exponential as it's rotating around. In the period of time t, this always rotates around some integer number of times. If m is equal to k, it doesn't rotate at all, it just sticks where it is, at 1. If m is unequal to k, it goes around some integer number of times. If I'm thinking of this as being real and this as being imaginary, I'm just running around this circle here. So what happens when I run around the circle once? The integral is zero because I'm up here as much as I'm down here, I'm over here as much as I'm over here. So this integral is always zero, which says that all of these terms except when m is equal to k disappear. So that means I wind up with just this one term u hat of k times the integral from minus t over 2 to t over 2dt. That's another integral I can evaluate, and it's equal to capital T times u sub k. So, u sub k is this quantity here divided by t, which is what we said over here. In fact, that argument, you can make it precise and it works.

So what this is saying is that, in fact, if you look at these Fourier series formulas, this thing is pretty simple in terms of this. The question which is more difficult is what functions can you represent in this way and what functions can't you represent in

this way. The easy answer is if you can think of it you can represent it in this way. But if you stop and think about it for six months, then that might not be true anymore. So if you become very good at this, you can find examples where it doesn't work, and we'll talk about that as we go on.

Let's define this rectangular function because we're going to be using it all the time. You probably used it when dealing with the Fourier integral all the time because you all know that a rectangular function is a Fourier transform of a sync function where a sync function is a sine x over x type function. If you don't remember that, fine. But anyway, this function is 1 in the interval minus 1/2 to plus 1/2 and it's 0 everywhere else, which is why it's called a rectangular function. It looks like this. We do it from minus 1/2 to plus 1/2 so it has area 1. In terms of that, we can express the formula for a time-limited function as this sum here, uk times these complex exponentials times this rectangular function. How many of you can see it ought to be rectangle of t over capital T instead of rectangle of t times t? Good. I can't. I always have to take two minutes doing that every time I do it, and if you can see it in your mind you're extremely fortunate. When we work with these things for a while, you will become more adept at doing things like that. But anyway, this works.

I want to look at an example now. And there's several reasons I want to look at this. One is to just look at what a Fourier series does. Suppose we expand the function, the rectangular function of t over 2. Now the rectangular function of t over 2 is going to be 1 from minus 1/4 to plus 1/4, instead of minus 1/2 to plus 1/2, because of the 2 in here. We want to expand it in a Fourier series over the interval minus 1/2 to plus 1/2. One of the things this is telling you is that when you're expanding something in a Fourier series, you have to be quite explicit about what the interval is that you're expanding it over. Because I could also find a Fourier series here using the interval minus 1/4 to plus 1/4, which would be a whole lot easier. But we're gluttons for punishment. So we're expanding in a Fourier series over the bigger interval from here to there. We go through these formulas calculating u sub k. We can easily do it for the first one, which is just u sub zero. It's just the average value in this interval minus 1/2 to 1/2, which is 1/2. The next term turns out to be 2 over pi

times the cosine of 2 pi t. We can evaluate all of them just going through more and more junk like that. But look at what's happened. We started out with a rectangular function. When we evaluate more and more terms of this Fourier series, the Fourier series terms are all very smooth. So what we're doing is trying to represent something with sharp corners by a series of smooth functions. Which means if we're going to be able to represent it, we're only going to be able to represent it by adding on more and more terms, which hopefully are going to be coming closer and closer to approximating this the way it should be approximated.

Now if you look at these terms here, these Fourier series, you will notice that every one of them, except this original one which is at 1/2 -- so this is the first term here in the Fourier series. The second term is to add on that cosine term. The first term is sitting here at 1/2. Every other one of them is zero at minus 1/4 and zero at plus 1/4. So when we add up all of those terms, what we wind up with is not what we started out with, but something which is 0 from minus 1/2 to minus 1/4. It's 1/2 at the value minus 1/2. It's 1 all along here. It's 1/2 over here, and 0 down here. Every time you study Fourier series you find out about these bizarre things. Every time you have a discontinuity in the function, the Fourier series comes out to split the difference on you. So you like to define your functions at discontinuities as either being here at minus 1/4 or here at minus 1/4. Then when you come back from the Fourier series, it forces you to be there.

Well, what does this say? It says that u of t, which we started out defining to have a certain value at minus 1/4 and plus 1/4, is equal to its Fourier series everywhere except here and here. I have to ask you to take that on faith. But you can see that it's not equal to it at those discontinuities. And it shouldn't be surprising that it's not equal to its discontinuities. I could have defined it as being zero at minus 1/4 or 1 at minus 1/4, and just that one point shouldn't change our integrals too much. Because of that as engineers, I mean at some level we have to say we don't care. It's only a modeling issue. Functions don't have perfectly straight discontinuities in them. If they do you don't care how you define it, it's a discontinuity.

This Fourier series is sort of coming back and slapping us with that. And it's saying

OK, the function u of t is not the same as its Fourier series because the two are different at these two points. You say OK, I don't care that they're not different at those two points. They're the same everywhere else. A mathematician comes back and says a function is a function is a function, and a function is defined at every value of t, and if u of t is equal to v of t, it means that at every value of t, u of t is equal to v of t. And you say ah. Well, turns out that by studying Lebesgue theory, all of those problems get resolved. Lebesgue was a very powerful mathematician. But you know at some level deep in his heart, he was an engineer. He was trying to get rid of all this nonsense that people talked about, and he resolved this question about how to talk about these functions in a nice way. I mean really, good engineers are mathematicians at heart, too. I mean at some level we all become the same.

What Lebesgue tried to say is that two functions are said to be equivalent in the L2 sense -- I'll talk about this L2 notation later -- if their difference has zero energy. In other words, Lebesgue said what's really important is not what functions are at each point, but really things about their energy. So what you would like to have is if u of t and v of t, if the difference between them, you take the magnitude of that and you square it, if that difference is equal to zero, you have to recognize that there's no possible way that you could ever distinguish those two functions, except just by fiat, by saying this is equal to this and not equal to that. That's the only way you could straighten that out in your minds. So we say the two functions are L2 equivalent if their difference has zero energy. Well, we have a couple of problems there. How do we define that? At this point, we're sort of already deep in the mathematical soup because, in fact, we're trying to make these small distinctions and make them make sense. We're also going to see, as we go on to two functions that have the same Fourier series, are L2 equivalent, because if two functions have the same Fourier series, put one of them there and one of them there, and we're going to see that when we expand it in a Fourier series they're both the same, and we're going to see that, in fact, what that means is that their energy difference has to be zero. Which says that if you don't talk about functions, but if you talk about their Fourier series, all of these confusions go away about things having to be equal point-wise.

So let's go on and try to say a little more about this. One of the problems that we

come up with is that not all time-limited functions, in fact, have Fourier series, even in a sense of L2 equivalents. You can think of functions which are so awful that they don't have a Fourier series, although it's hard to find them. We really want to make general statements about classes of functions. Why do we want to do that? Well, I can give you two reasons for it. The two reasons are both, I think, both good reasons. The first reason is that as we deal particularly with channels, we have to look at things both in a time domain and in a frequency domain. We look at things in the domain and the frequency domain, we have a function in a time domain, we have a Fourier transform in the frequency domain, and it turns out that nice properties in a time domain are not always carried over to the frequency domain and vice versa. Give me one example of that.

Suppose you think of a constant. The constant function, which is equal to 1 everywhere on the real line. Nice function, right? It models various things very well. It doesn't model physical reality, really, because I mean you don't care what this function was before the fourth ice age. You don't care what it is after we all blow ourselves up, I hope in not too many years. I mean I hope in more than just a few years. Therefore, when we model something as a constant, what do we mean? We mean that over the interval of time that we're interested in, this function is equal to a constant. And it means we don't want to specify what the time interval is that we're interested in, which is a common thing, because you don't want to set time limits on something.

Now, you take those functions which go on and on forever. Well, the Fourier series, you don't have any problem with them because we're going to truncate the function anyway before we take a Fourier series. But if we look at the Fourier integral, and we'll see this as soon as we get into the Fourier integral, the awful thing is that what has happened in the thousand years before the fourth ice age back, is just as important in the Fourier transform as what happens in the thousand years right now. In other words, everything is important. Things back in the dim past clobber you in a frequency domain. Things in the distant future clobber you in the frequency domain. Therefore, since we have to face the fact that we're dealing with

approximations, since we have to face the fact that we want to ignore things -- back there we want to ignore things there. When we look at constants in the frequency domain, we don't mean that something is constant over all frequency. We mean it's constant over the range of frequencies that we're interested in and we don't want to specify what that range is. You have the same problems going from there back to time.

So, as soon as we face the fact that we're really interested in approximations, and the approximations that we deal with normally in time are not the same as the approximations we deal with in frequency, at that point, we start to realize that we have to be able to make general statements about what functions do what kinds of things. We have to make general statements about what has a Fourier transform, what doesn't, what has a Fourier series, what doesn't have a Fourier series.

Now, one of the things we're aiming at is to define a class of functions called L2 functions. These are basically functions which have finite energy. The nice thing about those functions is that every one of them has a Fourier transform, and the Fourier transform is also an L2 function. All the other things that you deal with -- continuity, things like that -- doesn't carry over at all. L2 is the only property that I know of that really carries over from time functions to Fourier transform. So we really want to be able to talk about that.

So we want to talk about these finite energy functions. We want to be able to talk about representing finite energy functions. I say here, all physical wave forms have finite energy, but their models do not necessarily have finite energy. In other words, we look at a constant -- a constant does not have finite energy. How about an impulse? Does an impulse have finite energy?

AUDIENCE: [INAUDIBLE].

PROFESSOR: What? Yes? How many people think the answer is yes? The hands are going up very slow. Well, the answer is no. Let me explain why. It's something you should know. But it's something that you get blinded by studying too much signals and systems before you study any communication, because you're talking about all sorts

of transforms, all sorts of things that you deal with as functions, which you're dealing with electronically, instead of those functions that you're interested in transmitting. If you think of a narrow pulse of height 1 over epsilon, and of width epsilon, it has unit area. So I make epsilon very, very small. This starts to look like a unit impulse, right? In fact, you usually define a unit impulse somehow or other as thinking of some limiting process for this kind of rectangular function. What's the energy of that function? What?

AUDIENCE: [INAUDIBLE PHRASE].

PROFESSOR: Energy is 1 over epsilon, yes. What happens as epsilon goes to infinity? Bing. If you put an impulse into an electrical circuit it'll blow it up. You usually don't care about that because you don't see impulses in the physical world. You see things which are so narrow and so tall that in terms of the filters that you put them through, which have smaller bandwidth, those narrow pulses behave very nicely, and as you make those narrow impulses more and more high and more and more narrow, they behave the same way after they go through a filter. But before that they're ugly and they have infinite energy.

In fact, you could determine that from two of the statements I made earlier, and I'm sure I can't blame any of you for not doing that. I said that all finite energy functions have Fourier transforms which are finite energy, and you all know that the Fourier transform of a unit impulse is a constant, and the Fourier transform of a constant is a unit impulse. Therefore, if the constant has infinite energy, the unit impulse has to have infinity energy also. So anyway, we have all these functions we like to deal with all the time which do not have finite energy. We don't want to deal with those in this course, not because they aren't very useful in signal processing, but because they aren't useful as wave forms which we will transmit, and they aren't very useful as source wave forms. Source wave forms do not behave that way. Source wave forms that we want to encode all have finite energy, and we'll see why as we go on.

So now I want to try to tell you what the big theorem is about Fourier series. I will do this in terms of a bunch of things that you don't understand yet. The theorem says

we're looking at a function u of t, which is time-limited -- nothing strange there, that's what we've been looking at all along so far. It's a function that goes from minus t over 2 to t over 2, and we'll let it go into the complex numbers because we said it's just as easy to deal with complex functions as real functions. We're going to assume that it has finite energy. Then it says for each index k, the Lebesgue integral, u sub k equals this. In other words, this is the formula for finding the Fourier series coefficient. What we're saying here is we have to redefine that to be a Lebesgue integrall instead of a Reimann integral. But anyway, when you define it that way it always exists and it is always finite, necessarily. It can't be infinite, it can't not exist. It just is there.

The other thing is -- now this formula is harder to swallow as a whole. Let's try to look at it. What's inside of here is the difference between u of t and a finite approximation to the Fourier series. Now if you're taking a Fourier series, whether you're taking it on a computer or calculating it or what you're doing with it, you're never going to take the infinite sum. You're always going to be dealing with some finite approximation. This says that the different between u of t and these finite approximations, if you take that difference and you find the energy in that difference, says the energy in that difference gets small. In other words, it says that as you take more and more terms in your Fourier series, you get a function which comes closer and closer to u of t in terms of energy difference.

So that's the kind of statement that we want in this course, because we're aiming towards saying that this in the limit looks like that in terms of having zero energy difference between it. Namely, this is going to allow this function to converge to one of these strange functions that has bizarre values on discontinuities of u of t, because that doesn't make any difference in terms of energy. It says also the energy equation is satisfied. The energy equation -- I didn't say it was the energy equation -- I hope I said what it was. Blah blah blah. I have to write it down. The energy equation says that the integral of u of t magnitude squared dt from minus t over 2, the t over 2 is equal to the sum over k of u hat of k magnitude squared. And there's a 1 over t in here. There's either a 1 over t or a t -- I'm pretty sure it's a 1 over t, but I wouldn't swear to it. I can't believe I didn't have that written down. At

any rate it's in the notes. So you will find it there. I can't keep these constants straight. I think I did it wherever I stopped -- here. That's where I did it. That's the energy equation. Yeah, I got it right, amazing. The integral of u of t squared dt is equal to t times the sum of all the Fourier coefficients. I forgot to say this. This is something I wanted to say. This energy equation is important because in terms of source coding, if you take a function u of t, if you find this Fourier series, u of k, that's a sequence of coefficients. If we take that sequence of coefficients and we quantize them to some other set of values, v sub k, and then we recreate the function corresponding to this set up Fourier coefficients, we get sum v of t. So we start out with u of t, we go all the way through all of this chain, going through a channel and everything else, come back with some function v of t. This applied to u of t minus v of t says that the energy difference between u of t, and our re-created version v of t, is exactly the same as t times the sum of the differences between u sub k and v sub k squared. Now, that is the reason why most people talk about mean square error most of the time, because if you can control the mean square error on your coefficients, you're also controlling the mean square error on the functions. This formula does not work for magnitude or cubes or fourth powers or anything else. It only works for these square powers. That's why everybody uses mean square error rather than other things. It also makes sense for energy, because we believe in, in some sense, energy ought to be important and energy is important.

So, the final part of the theorem says that finally, if you start out with a sequence of complex numbers and the sequence of numbers has finite energy in this sense, then there's an L2 function, u of t, which satisfies all of this stuff. In other words, you can go from function to Fourier series, you can go from Fourier series to function. You can go either way. So long as you have finite energy this all works.

I want to spend just a couple of minutes talking about the difference between Reimann and Lebesgue integration to show you that, in fact, it isn't really any big deal. When you're talking about Reimann integration -- I've just showed the integral for a function between zero and 1. How do you conceptually find the integral of a

function -- a Reimann integral, which is what you're used. You split up the interval on the horizontal axis into a bunch of equal intervals of size 1 over n each. So you split it into n intervals, each one a size 1 over n. You approximate the value of the function in each interval somehow, as the smallest value, the largest value, the mean value, whatever you want to do. That doesn't make any difference because as the intervals become smaller and smaller and smaller and you have a function which is sort of smooth in some sense, then this Reimann sum here is going to get close to this interval. In other words, the Reimann sum is going to approach a limit, and that limit is defined as the Reimann integral. So that's the thing you're used to.

The Lebesgue integral is similar, oh and in a sense, it's no more complicated. What you do is instead of quantizing the horizontal axis into regions of size 1 over n and letting 1 over n get small, you quantize the vertical axis into intervals of size epsilon and you're going to let epsilon get small later. Then the thing that you do is you ask how much of the function is in each of these intervals here? So the amount of the function that lies between 2 epsilon and 3 epsilon is an interval t2 minus t1 -- there's that interval where the function is in this range. There's also this interval over here between t3 and t4. So you say the measure of the function in this interval here is t2 minus t1 plus t4 minus t3. You say the measure of the function in this region here, vertical region here, is t1, namely, this, plus 1 minus t4, namely, this region over here. So for any function you can do the same thing. That's the Lebesgue integral. Lebesgue integral says you do this and you let these epsilons get very small and you just add them up.

Let me say just -- last slide. Turns out that whenever the Reimann integral exists, namely, that limit exists, the Lebesgue interval also exists and has the same value. All of the familiar rules for calculating Reimann integrals also apply for Lebesgue integrals. For some very weird functions, the Lebesgue integral exists, but the Reimann integral doesn't exist. For some extraordinarily weird functions, there aren't even any examples in the notes. I couldn't find an example which I thought was palatable, not even the Lebesgue integral exists. So the Lebesgue integral is much more general than the Reimann integral. But the nice thing is you can almost forget about it because everything you know to do still works, it's just that some of

the things that didn't work before now work. Because those things that didn't work before now work, your theorems can be much more general than they were before. We'll talk more about that next time.

This material we're talking about right now is in the appendix to the lectures that just got passed out.