CH5230: System Identification

Response-based Description

Okay, very good morning. Before we, so today we plan to learn what are the other ways of representing LTA systems, particularly the parametric form. Until now we have discussed the non parametric forms. Mainly, the impulse responses, step-response and frequency response functions. And we have spent considerable time on understanding the frequency response functions, primarily because if you look at from a learner's viewpoint, you're not generally exposed and comfortable with frequency domain description, so we spend considerable time on that. I just want to add a few things to this frequency response function, especially the ETF, where we use the DFT. Now, you should slowly make it a habit to be comfortable with computing the DFT as I said, the DFT is computed using an algorithm known as a Fast Fourier Transform Algorithm. Although, it reset for as Fast Fourier Transform, it is not a new transform. It is simply a computationally efficient implementation of the DFT. Now, there was among the many questions that were asked after yesterday's class. There was one question that was asked as to what is the connection between the number of observations that we have and the frequency in DFT. Remember, we assume that we are given a signal and just using a generic signal x[k] here, it could be output or input. So given this, in observations of a signal, generally we know that the DFT is computed at the end points. So that the grid spacing is 2 pi by n. And or if you look at it in cyclic frequency, it's 2 over n. So what is the connection between this spacing and the number of observations? Why there should be a connection in the first place? Well, we will not go into detail proof, but I will give you a very quick way of understanding this. Remember, that eventually when you compute the DTF and you do all these calculations, it amounts to assuming that the infinitely long signal with we have not observed is a periodic extension of this, right? So I have x[k] starting from 0, let's say up to n minus 1, here.

This is n minus 1. We have not observed x[k] to the left of zero or to the right of n minus one. We don't know what it is. And the DFT assumes that the infinitely long signal is simply a periodic extension of whatever you have observed over this, those endpoints. That means it assumes that, the signal that you're analyzing is periodic with a period of, how much? N, correct. So what you are effectively doing is you are constructing what is known as a Fourier Series Representation of a discrete time periodic signal, whose spirit is n. Remember, when we talk of periodic signals, we talk a Fourier series representation. When we talk of a periodic signals satisfying some conditions, then only we talk of transforms. So, although, we set out to compute the Fourier transform, what we are actually doing is we are constructing a Fourier series representation. That is, the implicit message that you have to keep in mind. And when it comes to full year series representation, which is applicable for periodic signals, we know that whether it's a continuous time periodic signal or a discrete time periodic signal, the frequency components into which you're breaking the signal, now are not any set of frequencies. They are going to be specific sets of frequencies starting with the fundamental and then the harmonics. Because, intuitively, if you think of it, if I have a periodic signal of period n, a discrete time periodic signal. Then obviously, you will include only those sine and cosine which are the same period. They may not have the same fundamental period. So the first signal that would go into sinusoid or the first frequency competent that would go into explaining this would be of zero frequency. That is to take care of the DC component. Then, the next frequency that you would consider in explaining x[k]. Now we'll remember that we have assumed x[k] to be periodic. Okay.

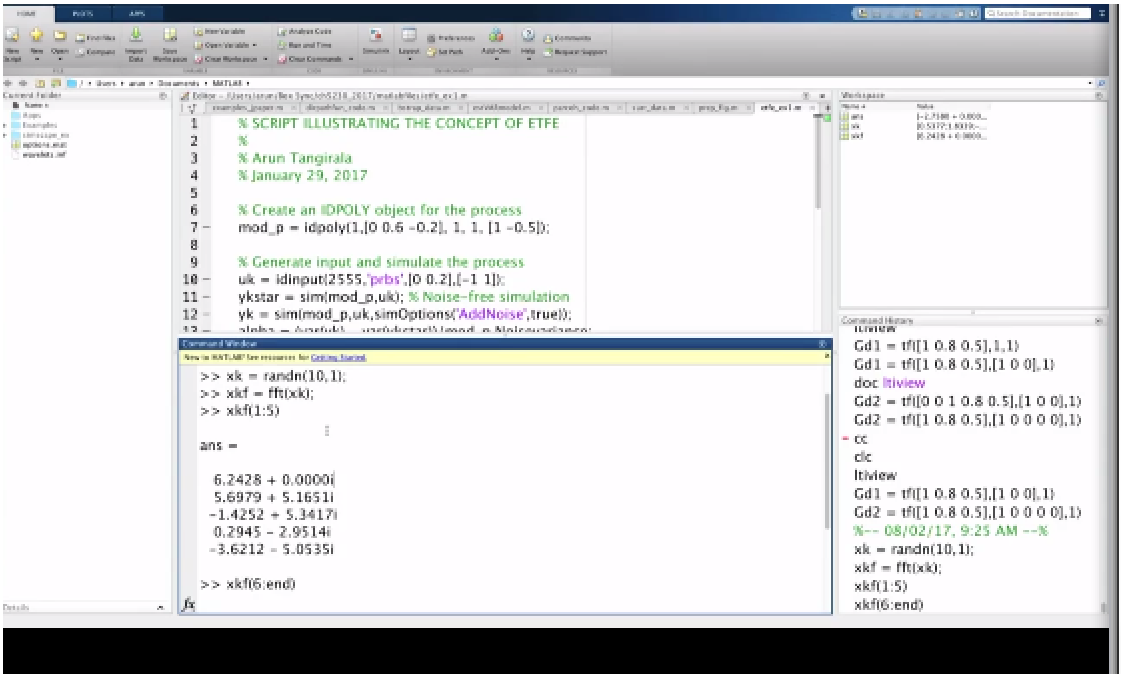
So the next frequency competent under consideration should have the same period as the signal and that would be the fundamental frequency. And that would be a sinusoid of frequency 1 over. There's one, very important thing that you have to understand. Whenever we look at the discrete time signals of this form, let's say, I'm looking at discrete time sine wave of some frequency f, then this cyclic frequency, we know has a certain restriction for it to be unique, right? We know that f has to be between minus 0.5 and 0.5. Or if you're right in terms of omega, minus pi 2 pi. So when I look at this any discrete time sine or cosine, not all sinusoids are periodic. Discrete time sinusoids are period unlike continuous time sinusoids. Compare this with the continuous time case. I use the capital F to distinguish. The difference between a discrete time sine wave and a continuous time sine wave is that, any discrete, sorry, any continuous time sine wave is periodic. What is its period, fundamental period? For the continuous time signal, what is the fundamental period? 1 by f. Very good. On the other hand, on the other hand, the discrete time sine wave, if you look at it, first of all, how is a period defined for a discrete time signal, sine wave or any signal? How do you define a periodic discrete time signal? Or when do I say discrete time signal is periodic? When do I say continuous time signal is periodic? When there [7:46 inaudible] exists a time interval after which it repeats itself. A time instant after which it repeats itself, right? Sometime. Apply this definition to discrete time. Correct. So samples are the important. So, I should find some sample after there is a discrete time signal repeats itself, which means, that the period of a discrete time sine wave is an integer. That's a key difference and a very, very important difference between a continuous time periodic signal and discrete time periodic signal. Always, if they are periodic, they have an integer period. Unlike your continuous time counterparts, which means, I simply cannot say for example here that the period of the discrete time sine wave with frequency small f is one over f. Can I say that? Suppose f is, small f is 0.4. What would be the period? Some example here, if f 0.4, I have discrete time sine wave of frequency point 0.4 cycles per sample. Units of small f are cycles per sample. What is it's period. Fundamental period. Sorry? 25? Why? 42? 2.5. But that violates this definition. You can't [9:51 inaudible] you can have a two point fifth sample. Two and a half sample. Right. It is the first sample after it you see the repetition. It's completely different from the continuous time case. And you're not used to it, because, generally when you learn Fourier transforms in your early math courses, you are looking at continuous time functions, continuous time signals and so on. Discrete time signals come into picture only when you take courses on signal processing or system identification and so on, time series analysis. So the period here, fundamental period here is five. On the other hand, if here, I had a sine wave of frequency 0.4 cycles per time, not samples. Then what would be the period? 2.5 There's no doubt here. Straight away, I can say it's 1 over f. But I can't do that here and you shouldn't do that. So how does 1 arrive at the period of a discrete time sine wave? This in rational form. F as for 4 by 10, you write in the simplest, most, you know, simple rational form such that the numerator and denominator are co primes. What this means is that, this signal completes actually two cycles in five samples. But you cannot do better than that. In the sense, if the signal maybe compute a completing two cycles, but you wouldn't know when it completes one cycle. Because it complete one cycle in two and a half samples which you don't have an opportunity to observe. So to the user, the first observation is that, that it has taken five samples to repeat. Okay? So the fundamental period of the signal is 5, the frequencies is 0.4. Now, if you now extend this argument here in general, a discrete time sine wave is periodic if and only if, you can express the frequency in rational form. That means, sinusoids, discrete time sinusoids with irrational frequencies are not periodic. Suppose, the small f here was, let's say square root of 0.5 or let us say the square root of 2 divided by 10. Then, it's not periodic. You'll never find a sample after which it'll repeat itself. Strange right? But that is the consequence of sampling a signal. You have chosen to observe at specific instance in time, that means, you'd, you have lost the opportunity to observe every periodic signal. Only some subset, a subset of periodic signals in continuous time can be observed to be periodic in the discrete time. That is something that we have to live with. Okay. So coming back to the point here, I am assuming x[k] to be periodic with period n. So what are all the frequencies that I can consider, what are all the sinusoids that I can consider in explaining x[k]? What would be those frequencies? You understood the definition of a periodic discrete time signal, right? Now what remains to be understood is, what sinusoids should I include in my bank to explain a discrete time signal whose period is m. What would be those frequencies? We'll leave aside the zero frequency that anyway is there by default. Apart from that.

Multiple of 1 by 1.

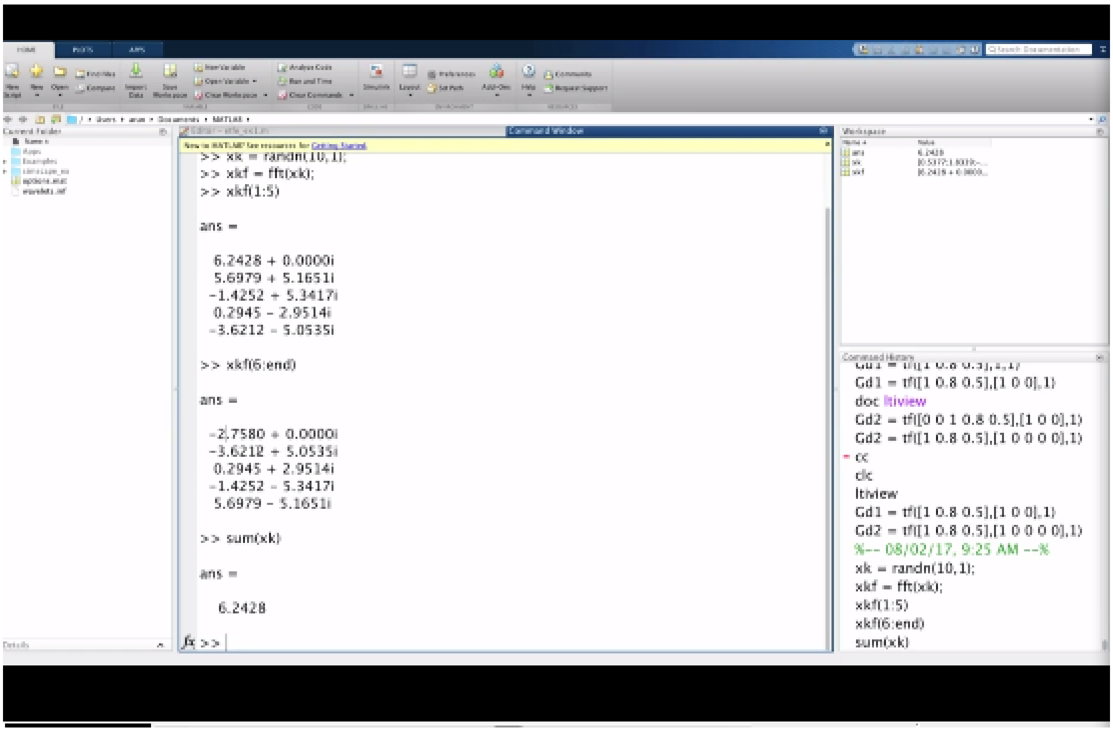
So 1 by n, 2 by n, all of them have a fundamental period of n. Up to n minus 1 by n. Why not n by n? Because n by n would mean 1 which again amounts to 0 frequency because of the repetitive nature of the discrete time sine waves. So, the unique set of sinusoids that I consider in explaining a discrete time periodic signal with period n are those starting from 0 to n minus 1 by n, in spacing of 1 over n. So, that is the connection between the frequency spacing and the number of observations that you have. I must say that I have actually used the interpretation of DFT to explain the spacing. There are, there are and there must be other ways of explaining why there is a connection between this frequency spacing and the number of observations. This is one of the simple ways of remember the connection. It doesn't mean that you cannot compute the DTF at other points. That means that other frequencies. You are, you should, you can, you can do many things, no problem. But that won't get you any new information. Okay. I can compute DTF at larger, on a final grid. No problem, but that would just amount to interpolation. If you want better display of the Fourier transform or the magnitude of Fourier transform, [15:55 inaudible] you can do it. But otherwise, it won't get you any new information. And by the way, this is called endpoint DFT. Okay. In literature, you will find, come across endpoint DFT and I should caution you that the notations can be considerably different across the literature, so do not get confused. For example, I have used k for the, to keep track of time or the sampling instance and n to keep track of frequency points. In literature, the situation may be reverse. Okay? So just make sure that you're aware of the notation and very often you will find this notation as well. Where now, because x now at discrete frequency function, you can now, just like we kept track of, we denoted a discrete time signal by x[k] where k is a [16:50 inaudible] k eight sampling instant. I can keep track of the DTF coefficient. This is called the DFT coefficient. By this index n, so both X, n [17:03 inaudible] square brackets and x omega n and [17:07 inaudible] parentheses are the same. Unless otherwise mentioned, it is always an endpoint DFT.

Okay. And then one more thing, these DFT coefficients, you know, of course, you'll compute in the FFT algorithm which was conceived by Cooley and Tukey in mid '60s. And since then, FTTs have been used everywhere, you will not find a single domain in which Fourier transforms have not been used. Now, the nice thing about this DFT coefficient is that they have a certain symmetry, remember? We we're computing n coefficients, right? Capital N coefficients. So, I just want to illustrate the symmetry and then we will move on to the difference equation forms. So, for this purpose, I have just generated. I hope you can see from the back, the font size is good enough. Okay. So what I have done here is, I have generated, I just generated bunch of numbers, 10 numbers of some sequence, okay? It's a random sequence that is immaterial as to where it comes from, because we want to illustrate the symmetry property. Next I take the Fourier transform using FFT. And I call that as x[k]. What is show you, so how many coefficients should you expect in x[k]? What should the length of x[k]? 10 right? Because it's an endpoint DFT. And so here, I'm showing you the first five coefficients. What are these corresponding to n equals zero, n equals 1, up to n equals 4. Although I say 1 to 5, remember, MATLAB's indexing begins from 1, but our indexing begins from 0. So you should always distinguish between the theoretical index and the programming index. So the value at n equals 0, what does it represent? It says the DC component, right? It's simply the, in fact, if you were to write the DFT, by the way, here, in all the algorithms, typically, you would not have the 1 over root n. So, sorry. X of n would be X[k] e to the minus j omega k, omega n. Where omega n is two pi. Which means that x of zero, what would that be? Simply, some of the numbers. Right? Is that clear?

(Refer Slide Time: 20:07)



So shall we check if that is indeed the case, then only we'll accept the result here. Correct? Compare this answer with this, right? Oops. All right. So you can see that it matches. The next is n equals 1. And then, 2, 3, and 4. And then the bottom here, I'm showing you the remaining five coefficients. You must notice quickly that there is some kind of a conjugate symmetry there. What I mean by conjugate symmetry here is, look at x, the big x at one. That is the coefficient, DFT coefficient that N equals 1. And the DTF coefficient at n equals 10. What is it that you notice? Their conjugates of each other, but they are flipped conjugates. So it is not that at n equals any n equals four, you have a mirror image. It's a flip mirror image. All right? So what are the unique coefficients here? From n equals zero, up to five, up to this point. Remaining at N equal six, seven, eight and nine are simply conjugate symmetry.

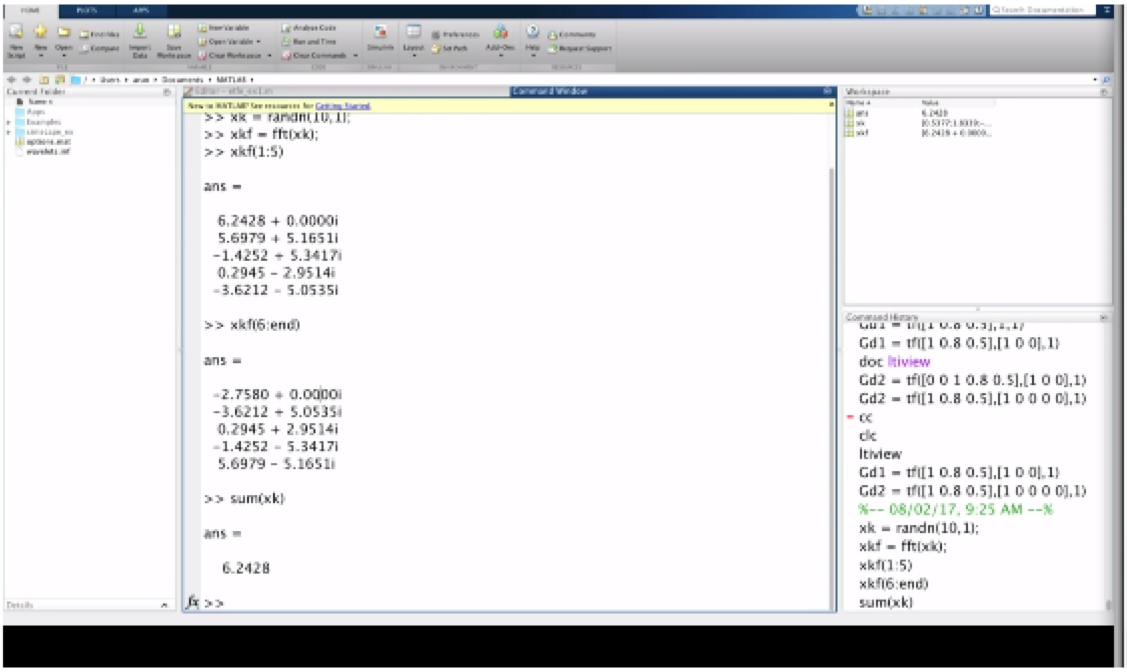


(Refer Slide Time: 21:28)

Why does this happen? It happens because of the property of the Fourier transform. Okay, actually we are supposed to compute from minus pi, two pi. But we are complete computing from zero to two pi. So you can think of it this way, the minus pi up to zero excluding zero, has been actually taken and flipped around and appended from 0 to the right of 0 two pi, right. So Frequency accessed, you can say, even if you say minus 0.5, so here is 0 in terms of cyclic frequency I have here minus 0.5 to 0.5. We don't have to consider both endpoints. So I would compute at frequencies here and also frequencies here. However, the DFT is not computing at negative frequencies per se, directly. What it is actually doing equal entry is, it's taking this instead of taking this interval, it looks at the interval. Doesn't matter, you can pick any interval of the length one in cyclic frequency, or an angular frequency of length 2 pi. But because of the nature of your complex exponential, when you compute a frequency point here that actually amounts to which one? Right. So it's actually being flipped around. And that is, that is why in many software packages you have the opportunity till you pass the FFT and it'll actually rearrange the FFT for you, so that you see from minus 0.5 to 0.5. Even in MATLAB, there is a way of doing it.

We will not get into that right now but there is a way of rearranging your DTF, so that you have coefficients lined up in this order. You understand, we are supposed to compute from minus 0.5 to 0.5 but we are computing from 0 to 1. And in doing so, the coefficients beyond 0.5 up to 1 are essentially flipped versions of minus 0.5 up to 0. And that is why you have a special conjugate symmetry property. It is not just symmetric, just like that. Ideally, if it was just conjugate symmetric in the usual sense, you should expect it this value and this value to be the same, right? But it isn't. In fact, what is happening is the value that I see here, it's actually equivalent to, so this value here. So what is happening is if I were to rearrange, if I were to rearrange then I'll have to flip and rearrange so that I get this coefficients. But without worrying about any of that, all you have to remember is the DTF coefficients are unique, only up to half. Depending, I mean, if it is n is even then it's half plus one. If n is odd, there is a number of observations is odd. It's up to the half interval. Okay, so here, disregard [25:03 inaudible] the zero coefficient, zero frequency, usually we remove the zero frequency before computing Fourier Transform, the remaining five coefficients are unique. This four here and the fifth one.

(Refer Slide Time: 25:21)



Any questions? Why some of you look confused. You can write for yourself what happened? So what all you have to do is go home and actually ask yourself, what is the relation between x at n by two plus 1? Okay, x at n by 2 plus 1 which coefficient would that be for us? The n equal 6, n equal 6 would mean this one here. Don't confuse the programming index with this and the other coefficient that you see which is, so this is equal into the conjugate of this part here. So you can ask what is the relation between this coefficient and x off n by 2, minus 1. How do you obtain a relation between these two? Just go back to this definition here, DFT definition plug-in, in place of n, you plug-in n by 2, 2 plus 1 and see if you can rewrite this in terms of n by 2 minus 1. You'll find that there is a conjugate relation. Clear? What this means is that computationally, I do not have to compute all the coefficients. I can exploit this conjugate symmetry property and only calculate up to the half or up to the unique ones. So I can save a lot on computation time. For 10 you may not see an advantage, but suppose I have 10,000 points, I can save considerable time and computational effort.

Now, what this also means for us is the empirical transfer function that we have computing g of e to the j omega. See, this is the property of DFTs. What is our ETF? ETF is simply the ratio of DFTs. I mean, one way if you look, if you look at the definition, G (ej omega n) is simply Y (omega n) over u of omega that is y[n] over u[n]. Doesn't matter whether you have a 1 or root 10 or not, they're going to get cancelled out. So what can you say about the ETF? It is also going to be conjugated symmetric. Right? In fact, what can you say about the magnitude? Same. So the only the phase will be different, but otherwise, magnitudes are the same. And that's why you will see in all the bode plots, in fact, in the example that I showed you yesterday, I didn't show you for the negative minus 5 to 0. I only showed you from 0 to pi. Because it is symmetric with respect to the left half. Any questions?

And finally, when you are computing the inverse Fourier transform, for some reason if you have to compute the inverse Fourier transform you have to be careful. Any software that you use for computing Fourier, look up what definition it is using. MATLAB does not use a 1 over root n. Okay. In fact, it has an IFFT. So if you pull up the help on inverse discrete Fourier transform, what does the IFFT do? It is the endpoint inverse transform. And you should pull up the reference page for IFFT and see what it uses. Sometimes the formula is given, sometimes it isn't. You should check if it does use a 1 over n in its inverse Fourier transform. Ideally, it should, because it's not using a 1 over root n in the forward transform. But in some software packages, that 1 over and n, maybe be missing because the same algorithm which is FFT can we use for inverse Fourier transform and forward transform. Why? Because after all, if you look at the Fourier transform, barring the multiplication factor, what is the expression sigma x[n] of course there is a 1 over n, if you don't have an n 1 over, 1 over root n here, you have j omega n k, but this time the summation is on n. So if you look at this, as I pointed out yesterday, they look almost identical except this conjugating, but that doesn't matter. As well as algorithm is concerned that doesn't matter. So you may have the same routine doing both except that you say that compute with the conjugate complex exponential and [30:30 inaudible] is one software which uses the same routine. If you say FFT, if you ask for FFT of x in r, it will give you the forward transform, if you ask for FFT of x, minus 1.

Then it would give you the inverse Fourier transform. But then it's your duty to take care of the 1 over n. Right. So when you are performing inverse Fourier transform of a sequence, keep in mind, several, at least two points. One, the multiplication factor and two, whether you have arranged the coefficients in the order that FFT returns. See, somebody may give you coefficients from here to here. But when you pass these coefficients to inverse Fourier transform, it expects to be in the same order as what FFT returns. By the way, earlier, I said that there is a command in MATLAB which rearrange is the coefficients for you. And that command is a FFT shift. So you can see the difference here. Let's do that here. So I have here, sorry x kf and ask for an FFT shift of x[k]. Look at what it has done. What is the difference between these two? Correct. So in x[k]f the counting begins from 0 frequency. Right? You can see here. This corresponds to n equals 0, and then small n equals 1 up to n minus 1. What about the counting here, the DC component is at the middle here, more or less, because n is even, we can't find what's the middle value, but more or less at the middle, which means it has arranged in this way for you. Whereas, when computation is being performed it compute from 0, 0 to this. So you have to be aware of all of this when you are using FFT. And that's a minimum, definitely you should be aware, there's no doubt about is no escape to it. You can't say, it's okay, I can forget it. But it's very important that you remember these two, the multiplication factor and the arrangement of the coefficients. So you can write your own routine to calculate ETF. All you have to do is use FFT. That's all. Compute FFT of y, FFT of u, take the ratio at the respective frequencies that is your ETF. You should try and do this and see if you get the same output that you get from the ETFE, routine in [33:13 inaudible SysID] toolbox. Of course, ETFE in SysID toolbox is much more comprehensive because it can do a few more things which probably your code doesn't do. But at the core, all it is doing is it's implementing that definition of ETF.

Whatever you give input-output data, it's going to compute the DFT, take the ratio and give it to you. So check if they both match. If they don't, then your tasks, why? Then only will get to know what are the other things that are pre-written codes, some of [33:46 inaudible] written by someone else is doing. And this is something that I caution everyone, not only in this course, but in every course where you use a software package to do some things using routines written by someone else, always, the person who wrote the code must have built in some features made some assumptions, you should be completely aware of that. For a long time, we have been trained, even during our grad-studies that we should write our own codes. Unless, of course, you're fully aware of what a particular routine that has been pre written does. So this is the story of DFT for you. I've talked about the theoretical aspects and the practical aspects, keep that in mind, we will return to this frequency response function estimation again, after we have reviewed the theory of random processes. Because then, as I said yesterday, we have to fuse these two and then come back and ask the question. How do we estimate the FRF now? Does this route give me good estimates, even when Y has measurement error and so on these are the questions that we'll ask? Or is there an alternate way of estimating the FRF. Fine? Any questions? Because now we're going to move to the parametric word. Good.