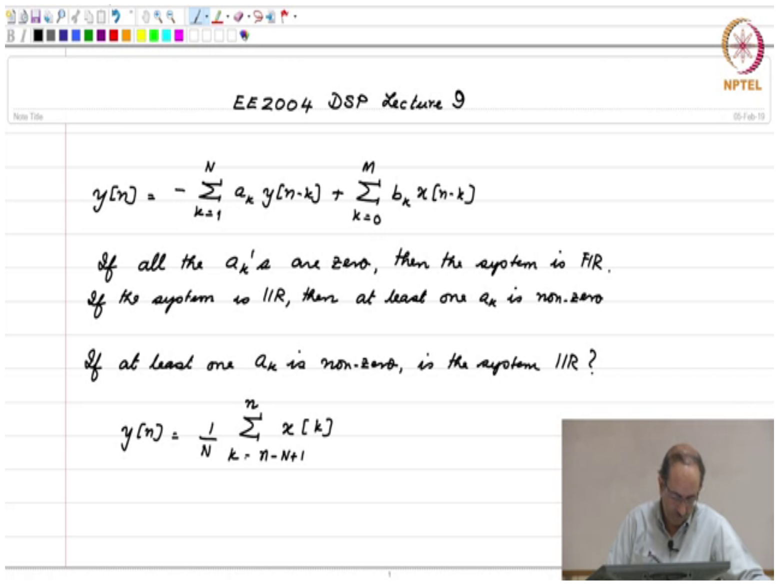
Digital Signal Processing

Prof. C.S. Ramalingam
Department Electrical Engineering
Indian Institute of Technology, Madras

Lecture 19:
Karplus-Strong Algorithm, Z-transform (1)
-Recursive implementation of FIR filters
-Karplus Strong Algorithum

We were looking at general systems and then focused on the subclass, namely LTI. And within LTI, we are interested in the class of systems described by linear constant coefficient difference equations. And one of the reading exercises for you is to look up Oppenheim and Schaeffer's Signals and Systems Section 2.4 to see, under what conditions does an LCCDE represent an LTI system, under what condition is it also causal.

(Refer Slide Time: 00:50)


EE2004 DSP Lecture 9

$$y[n] = -\sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k]$$

If all the a_k's are zero, then the system is FIR.
If the system is IIR, then at least one a_k is non-zero

If at least one a_k is non-zero, is the system IIR?

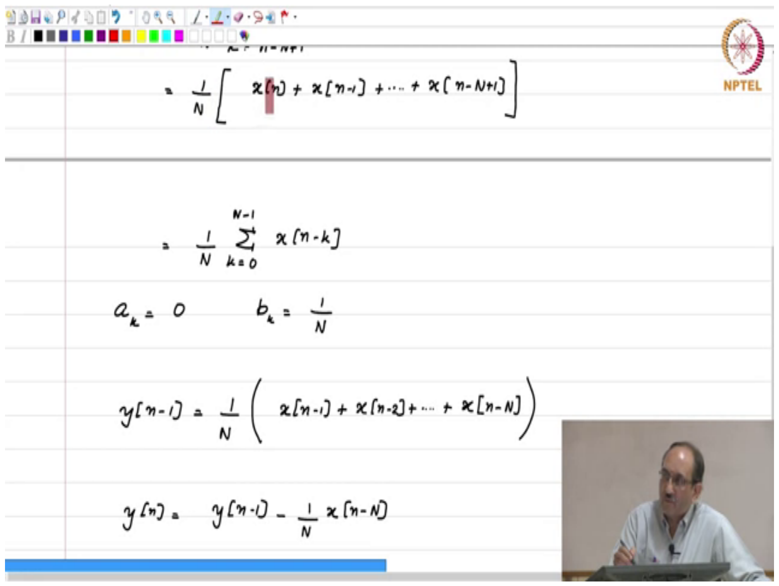
$$y[n] = \frac{1}{N} \sum_{k=n-N+1}^{n} x[k]$$

And we classified systems in terms of the duration of the impulse response, we looked at FIR (Finite Impulse Response) and then IIR (Infinite Impulse Response). And let us continue that discussion. So, the general LCCDE is of the form, $y[n] = -\sum_{k=1}^{N} a_k y[n-k] + \sum_{k=0}^{M} b_k x[n-k]$ and then we made the point that if all the a_ks are 0, then the system is FIR.

Then we also made the point, if the system is IIR, then at least one a_k is non-zero. And then the very last question that we were left with at the end of the last lecture was if at least one a_k is non-zero, is the system IIR? So, if at least one a_k is non-zero, is the system IIR? is the question. If the system is IIR, then surely you need at least one a_k to be non-zero.

Now, to answer this question, let us consider this particular example. So, let us consider $y[n]$ and what this system is going to do is, its going to take n contiguous samples, average them and then give out one number. Therefore, $y[n] = \frac{1}{N}\sum_{k=n-N+1}^{n} x[k]$. So, this is what the system is, the input output relationship.

(Refer Slide Time: 03:50)



$$= \frac{1}{N}\left[x[n] + x[n-1] + \dots + x[n-N+1] \right]$$

$$= \frac{1}{N}\sum_{k=0}^{N-1} x[n-k]$$

$$a_k = 0 \qquad b_k = \frac{1}{N}$$

$$y[n-1] = \frac{1}{N}\left(x[n-1] + x[n-2] + \dots + x[n-N] \right)$$

$$y[n] = y[n-1] - \frac{1}{N}x[n-N]$$

So, you can see that this is nothing, but $\frac{1}{N}\left[x[n] + x[n-1] + \ldots + x[n-N+1]\right]$. So, all it does is, it takes n contiguous samples, averages them and then puts out one number.; so $y[n]$ is given by this. So, clearly the above equation can also be written as $\frac{1}{N}\sum_{k=0}^{n-N+1} x[n-k]$. So, now, in this form, this is recognizable as the standard form of an LCCDE and here, what about a_k?

Student: (Refer Time 05:02).

Say that again.

Student: (Refer Time 05:09).

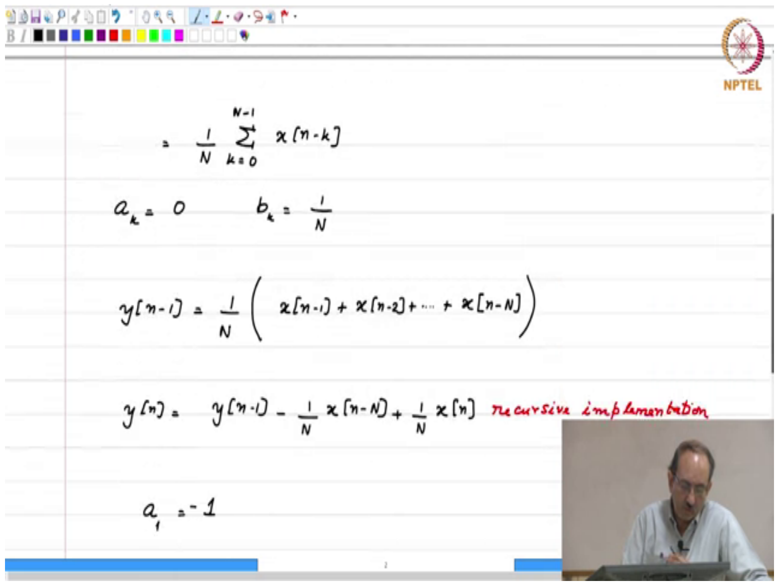
Yeah, I am sorry, $k = 0$ to $N-1$, like $x[n-k]$, $y[n] = \frac{1}{N}\sum_{k=0}^{N-1} x[n-k]$. you are right. Thank you. So, now, what about a_k? They are all 0 and $b_k = \frac{1}{N}$. So, clearly this system is FIR, alright. Now let us rewrite this system in a slightly different manner.

What is $y[n-1]$? $y[n-1]$ is nothing, but $\frac{1}{N}\left[x[n-1] + x[n-2] + \ldots + x[n-N]\right]$. All I have done is in this equation, wherever n is there, I have replaced n by $n-1$. So, this is what $y[n-1]$ is.

Now, let us express $y[n]$ in a slightly different manner. So, $y[n]$ can be also be written as $y[n-1]$. But if you look at the expression of $y[n-1]$, you have this term $\frac{1}{N}x[n-N]$, the last term that is there is not present in $y[n]$. Therefore, we need to get rid of this extra term, so, you can subtract $\frac{1}{N}x[n-N]$.

But, what is missing is the very first term, that is $\frac{1}{N}x[n]$ is missing. So, we need to put that back in. So, this is $y[n] = y[n-1] - \frac{1}{N}x[n-N] + \frac{1}{N}x[n]$.

(Refer Slide Time: 07:46)



$$= \frac{1}{N}\sum_{k=0}^{N-1} x[n-k]$$

$$a_k = 0 \qquad b_k = \frac{1}{N}$$

$$y[n-1] = \frac{1}{N}\left(x[n-1] + x[n-2] + \cdots + x[n-N] \right)$$

$$y[n] = y[n-1] - \frac{1}{N}x[n-N] + \frac{1}{N}x[n]$$ recursive implementation

$$a_1 = -1$$

Now, the intuition behind rewriting like this is that, all you are doing is you are averaging n samples and when a new sample comes, you do not have to recompute the average from scratch, all you need to do is update the average. So, this is updating of the average. So, this the old average, you throw out $1/N$ times the last sample and then add in $1/N$ times the latest sample.

So, this is updating the average without having to recompute from scratch. Now, in this form, you see that $a_k = 1$. Remember, a_ks are the coefficients of terms like this, $y[n-k]$. So, here actually, I should call it as a_1; $a_1 = 1$.

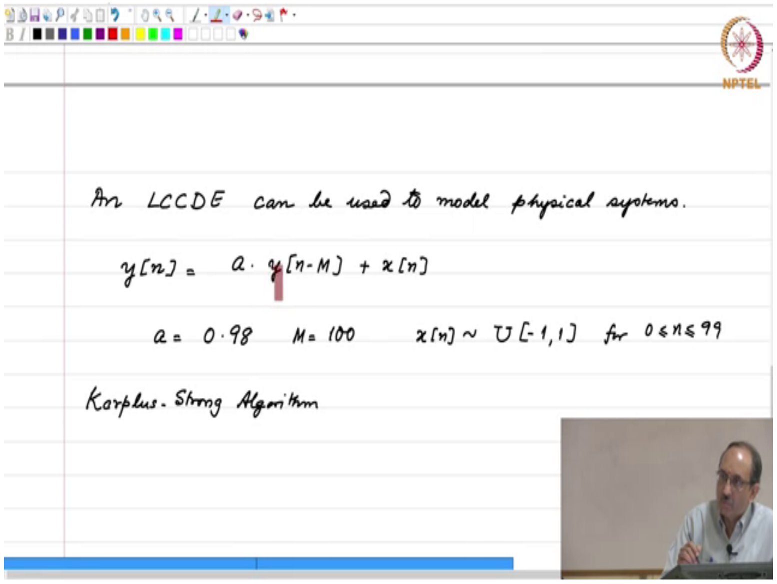
Student: (Refer Time 09:06).

a_1 is 1, right. So, this is -1. Was that the question? Was the? Yeah, because in the form, we have minus; so $a_1 = -1$. So, this is what is called recursive implementation. It is recursive because you take $y[n]$ delayed by 1 sample to get $y[n-1]$ and that is fed back into the system. So, this implementation, what is called as a recursive implementation because, the current output $y[n]$ in this implementation also depends on the previous output. But, this does not convert the system which was FIR into an IIR system. The system continues to be exactly the same as before and we convinced ourselves that this system is indeed FIR.

But, in this implementation, it has feedback and $a_1 = -1$, but this does not make the system IIR, that is the point. Therefore, the question that we asked, if at least one a_k is non-zero, is the system necessarily IIR? Here is an example that shows that, that need not be the case that there can be systems in which you may have some aks to be non-zero and yet the system can be FIR.

We will revisit this later, we will talk about poles and zeros and we will talk about uncancelled non trivial poles and later we will make the statement that if the system has an uncancelled non trivial pole, it necessarily has to be IIR. So, we will revisit exactly the same example, moment we introduce the z transform and then we will talk about poles and zeros and then we will talk about trivial and non trivial poles and then we will make the statement that this system will be definitely IIR, if there is an uncancelled nontrivial pole.

(Refer Slide Time: 12:11)



An LCCDE can be used to model physical systems.

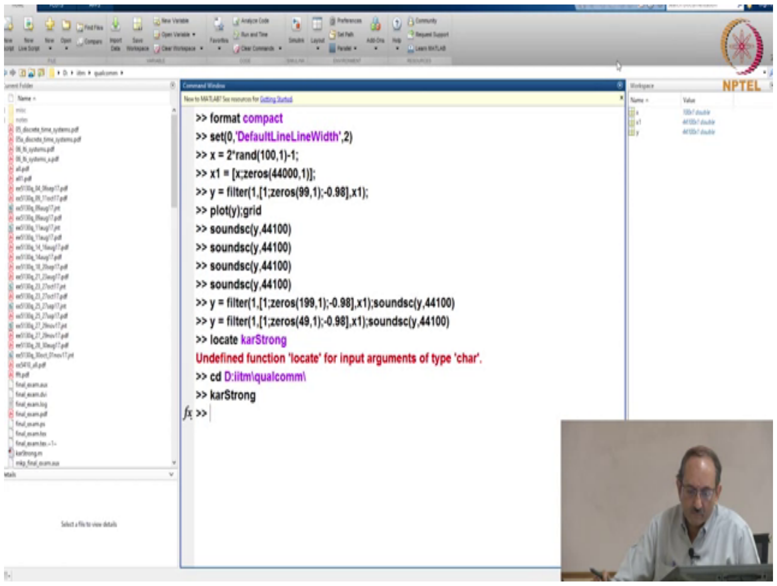
$y[n] = a \cdot y[n-M] + x[n]$

$a = 0.98 \quad M = 100 \quad x[n] \sim U[-1,1] \;\; \text{for} \;\; 0 \leq n \leq 99$

Karplus-Strong Algorithm

An LCCDE can be used to model physical systems. Let us consider this particular example, $y[n] = a.y[n-M] + x[n]$n, this is a very simple difference equation. So, we will let a to be 0.98, $M = 100$ and then $x[n]$, we will make it as a uniform random variable between -1 and $+1$ for $0 \leq n \leq 99$. Then, we look at what y is and then see whether it reminds us of something that we have seen or heard before.

(Refer Slide Time: 14:04)



```
>> format compact
>> set(0,'DefaultLineLineWidth',2)
>> x = 2*rand(100,1)-1;
>> x1 = [x;zeros(44000,1)];
>> y = filter(1,[1;zeros(99,1);-0.98],x1);
>> plot(y);grid
>> soundsc(y,44100)
>> soundsc(y,44100)
>> soundsc(y,44100)
>> soundsc(y,44100)
>> y = filter(1,[1;zeros(199,1);-0.98],x1);soundsc(y,44100)
>> y = filter(1,[1;zeros(49,1);-0.98],x1);soundsc(y,44100)
>> locate karStrong
Undefined function 'locate' for input arguments of type 'char'.
>> cd D:\iitm\qualcomm\
>> karStrong
fx >>
```

So, let me implement this in matlab. Remember the input has to be uniform random variable for 100 samples and that uniform random variable is between -1 and 1. So, x is, I am going to use the matlab's rand function and rand function gives me a uniform random variable between 0 and 1 Therefore, I need to convert that to be between -1 and 1 therefore, what I am going to do is very simple here.

So, rand gives me uniform, randn will give me Gaussian normal distributed random variables. So, you need to look up rand and randn in matlab. So, here I am having 100 samples. So, if x is uniform between 0 and 1, $2x - 1$ will be uniform between -1 and 1. And then, after all, I am going to let only the first 100 samples to be uniform, the rest I will make it as 0. Therefore, I will form a vector of 44100 samples.

The first 100 samples are x, which are the random variables we have just now generated. Then, I am going to implement this difference equation. So, you need to look up the matlab command *filter*, a filter has the form $filter(\mathbf{b}, \mathbf{a}, x)$. So, $\mathbf{b}$ are the b coefficients of the LCCDE that we have seen, a vector of coefficients, $\mathbf{a}$ again a vector of coefficients corresponding to the a coefficients in the LCCDE and x of course, is the input, ok. So, filter and then the b coefficients is 1 as far as a is concerned. So, this is the filtered output.

Now, let me plot y. So, this is how it looks, this is the output of the difference equation with the first 100 samples being random between -1 to 1 and then the rest are all zeros and this input has been fed to this system with these coefficients.

Now, what I am going to do is, I am going to play this, *soundsc* will take the vector y and *soundsc* will actually scale it. Matlab expects the vector to be within a certain range, if we just use the *sound* command, if you have not scaled it properly, you may either exceed the matlab's expected range or the scaling. If its too small, the volume will not be loud enough whereas, if you exceed the range, you will start to get clipping. So, *soundsc* will take care of the normalization so that, its within the maximum range possible. So, y is a vector and then the sampling frequency is 44100 is what I am going to use.

Let me play it again, what does it remind you of? It reminds you of a plucked string, right. So, it certainly reminds you of a plucked string. So, now, remember this parameter M was there. So, you can play with this, you can either make it larger or smaller. So, now, let me make it larger and then see what effect that has on the quality of the output.

So, now I am again going to filter it. Rather than using this particular value, I am going to now give it a longer delay. So, that has the frequency increased or decreased? Decreased, alright. So, now, you can expect what will happen if instead of increasing the delay if I reduce it. So, this is a shorter delay. So, now, it looks like a plucked string except with that, it has a higher frequency.

Now, let me see if I can play this. So, now, Rupak Vignesh was a student in computer science some time ago when I taught the class, he was there and then this kind of excited him. So, what he did was, take this difference equation and then he varied the parameters and then created this sound file, repeated applications of this difference equation. That is all, nothing more, nothing less. The only thing he did was, vary the amplitude and delay.

So, clearly this looks like a stringed instrument that is being played. So, it kind of gives you an idea as to what you can do with such a simple minded thing as a simple difference equation. This you need to look up, what is called the Karplus strong algorithm. If you are curious to learn more, please look up the Karplus strong algorithm.

So, in music synthesis, signal processing is used very heavily and this even though it sounds good very impressive and all that, it is still not quite what you can use to sell commercial instruments, right. So, this needs to be refined and other things have to be added, so that the quality is of commercial standards.

I mean, you cannot just using this simple minded thing, you cannot hope to make a synthesizer and hope to make money, you need to refine this, plus this software is only a stringed instrument.

So, this needs to be improved upon, plus you also need other things like percussion instruments and so on. So, you need to model those. So, those are modeled and then they are generated using in hardware, right. So, DSP algorithm generates all these things. So, in music synthesis, signal processing is very heavily used and this kind of gives you feel for what can be achieved.