Mathematical Aspects of Biomedical Electronic System Design
Professor Rathin Joshi
Department of Electronics Systems Engineering
Indian Institute of Science Bangalore
Lecture 13
MATLAB Live Demo on Moving average and signal acquisition

Hello everyone, welcome to the course on mathematical aspects of biomedical electronic system design. I am Rathin Joshi, TA of the course, our PhD student at Department of Electronics System Engineering, Indian Institute of Science. In the previous TA class, what we saw is what are the different kinds of signals, what are the different types of systems.

Also, we have seen our demonstration on moving average per system; how it smoothens and we remove the high frequency burst, and some of the unwanted noise in biomedical signals. In response to that video, I have received multiple queries regarding showing how I have generated the sinusoid functions; how I have added noises, which I have already shown there.

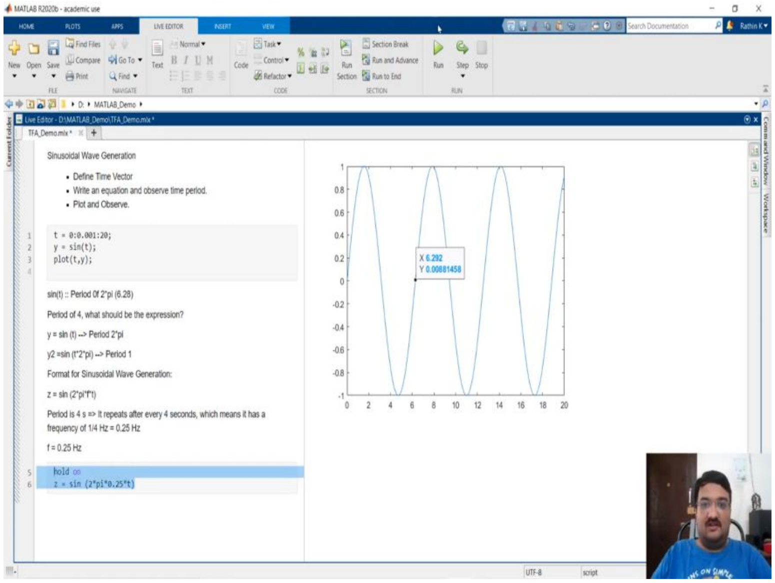
But many of you have asked how I have generated the sinusoidal function. So, it is a very basic thing I believed at that time that most of you are aware. But anyways, we are going to see this in today's lecture. We will start from a scratch how to generate a signal, how to generate a sinusoidal signal; how to fabricate or how to get signal which is near to what a physiological or biomedical signals, in order to understand the further processing of that.

So, we will see that in this lecture; we will also see how Fourier transform is useful in biomedical signal processing; we will see how it is being calculated, how Fourier transform is calculated. As per the definition, we will also see how FFT is calculated, why it is called a Fast Fourier Transform. And which is more feasible or why FFTs are more frequently used; also, we will map it with the corresponding time domain waveform.

And see by changing a time domain waveform, how it is reflecting in frequency spectrum. All these things we will see in this TA class; so let us get started. I will try to add as much demonstration as possible; so, it will make things easier for you people to understand some of this fundamental concept.

One more important thing is if you are working on any biomedical signal processing project, if you want me to cover some of the key concepts, or if you want me to demonstrate some of the concepts; feel free to write to me to my email id. Or in the forum, we will try to include that topic in the course, which might help you in your, with your project or for your better learning. We will start the demonstration using MATLAB.

(Refer Slide Time: 03:36)


MATLAB R2020b - academic use
HOME PLOTS APPS LIVE EDITOR INSERT VIEW
Search Documentation
Rathin K
New Open Save Find Files Compare Print Go To Find Text Normal Code Task Control Refactor Run Section Section Break Run and Advance Run to End Run Step Stop
FILE NAVIGATE TEXT CODE SECTION RUN
D: MATLAB_Demo
Live Editor - D:\MATLAB_Demo\TFA_Demo.mlx *
TFA_Demo.mlx *
X 6.292
Y 0.00881458
1 0.8 0.6 0.4 0.2 0 -0.2 -0.4 -0.6 -0.8 -1
0 2 4 6 8 10 12 14 16 18 20
UTF-8 script

Sinusoidal Wave Generation

- Define Time Vector
- Write an equation and observe time period.
- Plot and Observe.

```
t = 0:0.001:20;
y = sin(t);
plot(t,y);
```

sin(t) :: Period Of 2*pi (6.28)

Period of 4, what should be the expression?

y = sin (t) --> Period 2*pi

y2 =sin (t*2*pi) --> Period 1

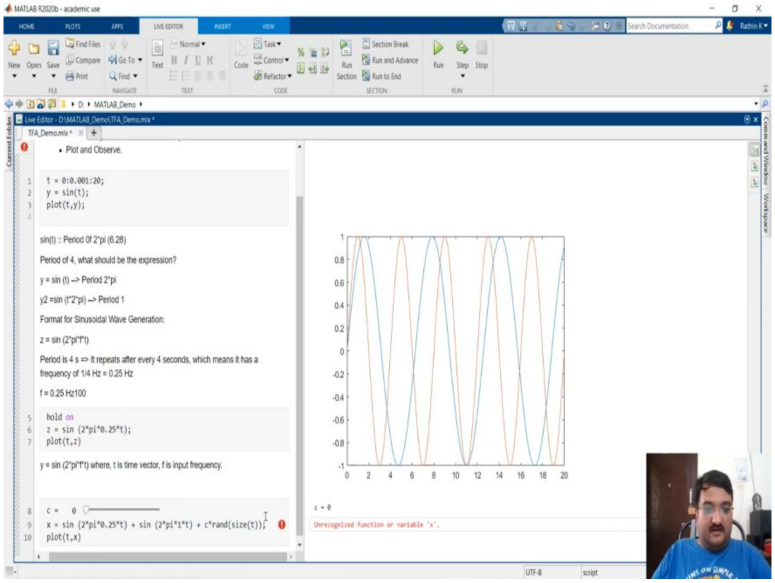
Format for Sinusoidal Wave Generation:

z = sin (2*pi*f*t)

Period is 4 s => It repeats after every 4 seconds, which means it has a frequency of 1/4 Hz = 0.25 Hz

f = 0.25 Hz

```
hold on
z = sin (2*pi*0.25*t)
```


MATLAB R2020b - academic use
HOME PLOTS APPS LIVE EDITOR INSERT VIEW
Search Documentation
Rathin K
New Open Save Find Files Compare Print Go To Find Text Normal Code Task Control Refactor Run Section Section Break Run and Advance Run to End Run Step Stop
FILE NAVIGATE TEXT CODE SECTION RUN
D: MATLAB_Demo
Live Editor - D:\MATLAB_Demo\TFA_Demo.mlx *
TFA_Demo.mlx *
1 0.8 0.6 0.4 0.2 0 -0.2 -0.4 -0.6 -0.8 -1
0 2 4 6 8 10 12 14 16 18 20
UTF-8 script

- Plot and Observe.

```
t = 0:0.001:20;
y = sin(t);
plot(t,y);
```

sin(t) :: Period Of 2*pi (6.28)

Period of 4, what should be the expression?

y = sin (t) --> Period 2*pi

y2 =sin (t*2*pi) --> Period 1

Format for Sinusoidal Wave Generation:

z = sin (2*pi*f*t)

Period is 4 s => It repeats after every 4 seconds, which means it has a frequency of 1/4 Hz = 0.25 Hz

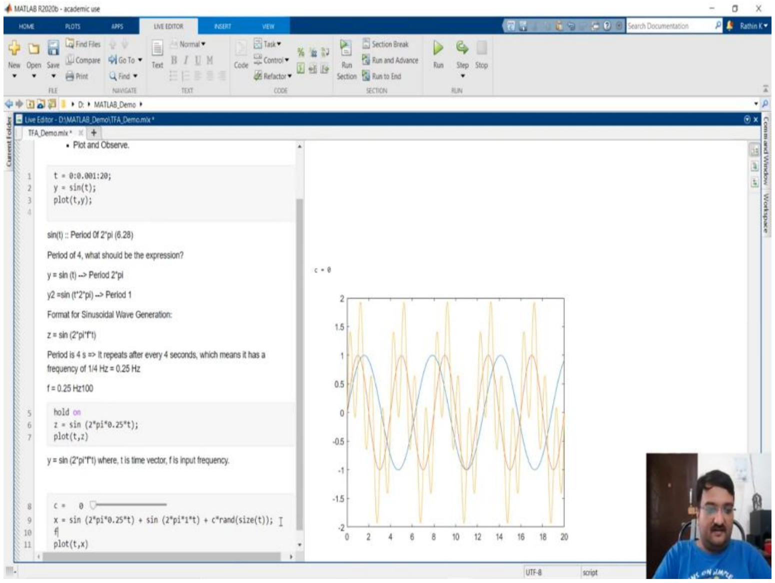
f = 0.25 Hz100

```
hold on
z = sin (2*pi*0.25*t);
plot(t,z)
```

y = sin (2*pi*f*t) where, t is time vector, f is input frequency.

```
c =   0
x = sin (2*pi*0.25*t) + sin (2*pi*1*t) + c*rand(size(t));
plot(t,x)
```

```
c = 0
Unrecognized function or variable 'x'.
```



- Plot and Observe.

```
t = 0:0.001:20;
y = sin(t);
plot(t,y);
```

sin(t) :: Period Of 2*pi (6.28)

Period of 4, what should be the expression?

y = sin (t) --> Period 2*pi

y2 =sin (t*2*pi) --> Period 1

Format for Sinusoidal Wave Generation:

z = sin (2*pi*f*t)

Period is 4 s => It repeats after every 4 seconds, which means it has a frequency of 1/4 Hz = 0.25 Hz

f = 0.25 Hz100

```
hold on
z = sin (2*pi*0.25*t);
plot(t,z)
```

y = sin (2*pi*f*t) where, t is time vector, f is input frequency.

```
c =   0
x = sin (2*pi*0.25*t) + sin (2*pi*1*t) + c*rand(size(t));
f
plot(t,x)
```

So, this is a live editor, in which I can code and type the text which might be useful for your understanding. So, I have chosen for this instead of only demonstration, let us quickly generate a sine wave; or I should say sinusoidal wave sinusoidal wave generation.

So, what are the steps, I will just list down the key steps. First, we will define a time vector; we will write an equation for sine wave, whichever frequency and all you want. And observe the time period, because this is mostly the case most of you are not aware that how can I generate a particular sine wave with a desired frequency plot and observe. So, these are the basic things and basic steps which you are going to follow.

Now, let us realize this, in order to realize this, we will convert this into a port; as discussed we will first define the time vector t. Now, let me define in such a way that it would be from 0 to let us say, I want it till 10 minutes or 10 seconds; but I wanted at every millisecond. So, this is my resolution. I need an increment of 0.001 second; that is 1 millisecond and I want it for 10 seconds. So, this is how my time vector is defined.

Now, what I want to get? I want to just calculate the sine wave; I do not know what is value and all. But I just want to get the sine wave. I am more interested in getting the relationship between the frequency of this wave, and how I can change it; need a controllability on the frequency, which most of you have asked. And then I can simply plot it with respect to y; so let us quickly run this. So, you will get an idea how it will look like; then we will play with this. And we will identify how frequency can be obtained and all.

So, you can see that we have defined time from 0 to 10, which is here and sine wave; so, it starts with 0. How to how we can identify the time period of this? So we need to we can we know that for sine wave behavior, it will repeat like this. Suppose the good thing about live editor is see I just have to increase here 20. If I run the same thing; I will get corresponding result quite easily.

So, in order to give you an idea, what is the time period, this is getting repeated? Right? Three times almost in 20 seconds. So, we can say that it is $\frac{20}{3}$, which is around six point something would be the one time period something. But how to get it precisely? We can get the data tips from here, when exactly it is equal to 0 somewhere here.

Let me just try somewhere here; so around 6.3, around 6.3, it is 0. If I can little drag it to get the more precise. If you can see 6.29; so all this thing we all know that sine waves, period, time period is 2π. So, 2×3.14 blah blah will comes around this. But we have written an expression as y=sin(t), just given a period of 6.28. Some of you wants certain more better order; or you know, you want to keep the sine frequency as our sine waves as to how to get that.

So there are, there is a format, okay; for sin(t), I will just write it here, for your better understanding text switch to text. Of course sin(t) corresponds to period of 2π, which is nothing but 6.28 something like this. Whereas, you need a period period of; how much let us say I need a period of 4. Then what should be the expression or equation which we have written?

Now, you see, the period of 2π is obtained by writing a ‘t’ here. Now, if you want the same thing to get repeated after 4 second. So, for that, there is a formula or there is a structure sin(2π); what if, if 2sin(2πt), 2πt instead of t, what if it is sin(2π)? 2π we are already writing. So, in a way, what we are doing? We are scaling this entire scale, instead of t now we are multiplying with 2π. So, for in between 0 to 1 only, you will get the one time period.

Again, I will repeat y=sin(t); it has a period 2π. Now, what I am doing is I am changing the signal y_2=sin(t); I am keeping as it is, just multiplying it by 2π. What it does is it will multiply your x-axis, which was moving here from 0 to 2π for one iteration; now it has a multiple of 2π. So, instead of 0 to 2π, it will get all the value in between 0 and 1. Little bit difficult to understand, but once you understand, you can clearly have an idea that why this period is now not 2π, but 1.

In other words, the format for sinusoidal wave generation, which most of you know; we write it without completely understanding that why it is like this. Suppose let us name the signal as z; I will keep the time vector as it is, but the formation is like 2π, some number. I will just put here f into your time vector. What you are doing is you are multiplying it, you are multiplying your time axis, which was earlier just time variable with by 2π in order to get integer frequency, in order to get a integer time period; so let us quickly realize this again in terms of code.

So, what I will do is I want to plot it here only, it will be better. So, first I will hold this graph that I do not want to plot it in another separate graph; and I will plot it as z = sin(2π). Now, as we mentioned, we want a period of 4. Now, period of 4 means, in 20 seconds if I say a period of 4; it will get repeated 5 times. So, a period I can right here period is 4 seconds, which implies that it repeats after every 4 seconds. And which means it has a frequency of $\frac{1}{4}$ hertz, which is nothing but 0.25 hertz.

So, your frequency, or how rapidly you want this sine wave to repeat is 0.25 hertz. So, I will just write this 0.25 hertz here as it is; and I will put the 0.25 hertz here, which is nothing but your f. So, and your time vector will remain as it is. Here what you are saying is, I want this particular waveform to get repeated in how much period of time? 4 seconds; and let us quickly see that whether the same holds or not. So, let me run this section, so you will get an idea that how it is getting repeated, and how you are getting a value though. I have to plot it as well, so p,z; these are the values; if I plot it should be z only, auto suggestions are on.

Now if I plot this z; now if I plot you will see another trace here once it is run, see the trace here. So it is little; we can focus on this red color line, and identify the time period. Now, where it is about to get 0? So it is zero at around 4, near about 4. So, you can see that what we wanted is we want a period of 4. So, it should repeat 5 times within 20 seconds, we get it. So, what is the format now? The main thing to understand here is the format you should be able to generate all kinds of desired waveform by your own. So, format is y = sin(2πft), where ‘t’ is time vector, ‘f’ is input frequency.

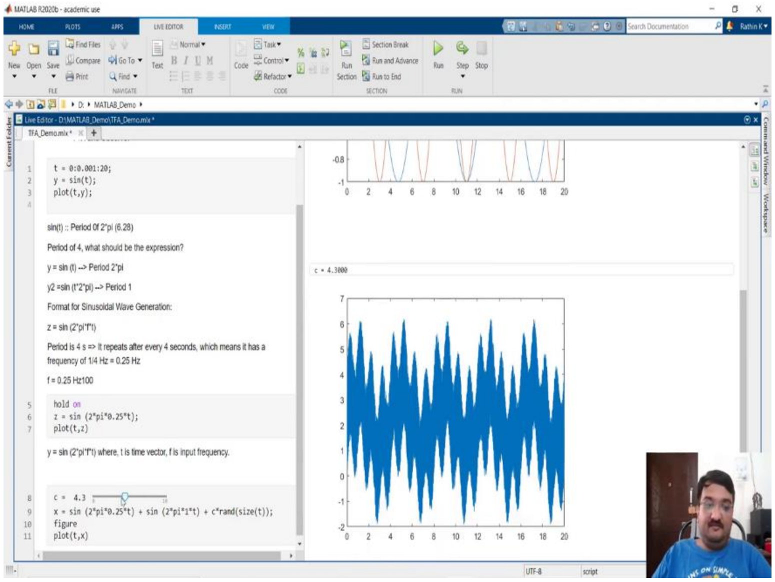
So, I use this in order to get the desired waveform. And while explaining the moving average to you. Let us quickly see how that how this moving average helps and getting how I can quickly control that shift. That is that would be easier here to understand; so let us just make one signal. I am just generating a random signal here; I will switch to code in that case. So, x as name it as x, x is equal to; I will copy this. So, we will say over time, x is equal to. This is already the signal which we have; I am adding some more components to it, similar kind of components 2π; let us make a frequency 1.

So, it repeats at one second; and I will also add a random noise here; random noise of size of x. I need to delete this; you can even scale if you want to. If you want, you can put here c; c is nothing but your constant, which is getting multiplied by this random function. One important point is you can add your own controllability here, you can add, you can decide how much you want your c.

So, I will quickly show that to you, we can add a slider here. Now, many of you would have seen a slider in your while accessing a website or something. What you can do is you can set up the maximum minimum limit; you can also define a step, how much you want to pause it or move; so, this is your c. Accordingly, it will get multiplied by here your sin; how what I am trying to say with this, let us quickly visualize it.

Let us quickly plot t and x; this is a separate plot, not with the existing plot. Let us just quickly plot it, unrecognized is the function or variable; because here it is not defined. Let me just give a variable as time; it is small runtime issue, but can be (multi) easily resolved by adding this thing. So now you can see three different waveforms; what I can do is if I just want to see our obtained graph, I just right here figure. And if I then run it, I will get a separate graph like this.

(Refer Slide Time: 17:27)



```
t = 0:0.001:20;
y = sin(t);
plot(t,y);
```

sin(t) :: Period Of 2*pi (6.28)

Period of 4, what should be the expression?

y = sin (t) --> Period 2*pi

y2 =sin (t*2*pi) --> Period 1

Format for Sinusoidal Wave Generation:

z = sin (2*pi*f*t)

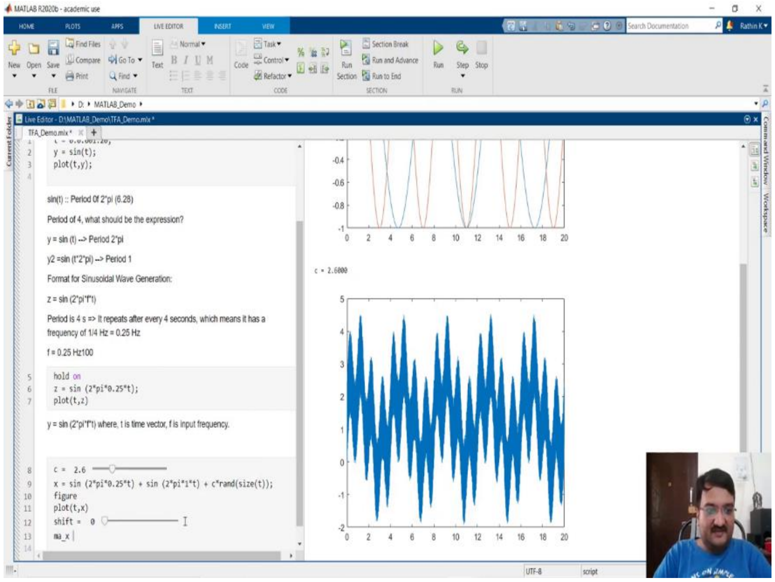
Period is 4 s => It repeats after every 4 seconds, which means it has a frequency of 1/4 Hz = 0.25 Hz

f = 0.25 Hz100

```
hold on
z = sin (2*pi*0.25*t);
plot(t,z)
```

y = sin (2*pi*f*t) where, t is time vector, f is input frequency.

```
c =  4.3
x = sin (2*pi*0.25*t) + sin (2*pi*1*t) + c*rand(size(t));
figure
plot(t,x)
```



```
y = sin(t);
plot(t,y);
```

sin(t) :: Period Of 2*pi (6.28)

Period of 4, what should be the expression?

y = sin (t) --> Period 2*pi

y2 =sin (t*2*pi) --> Period 1

Format for Sinusoidal Wave Generation:

z = sin (2*pi*f*t)

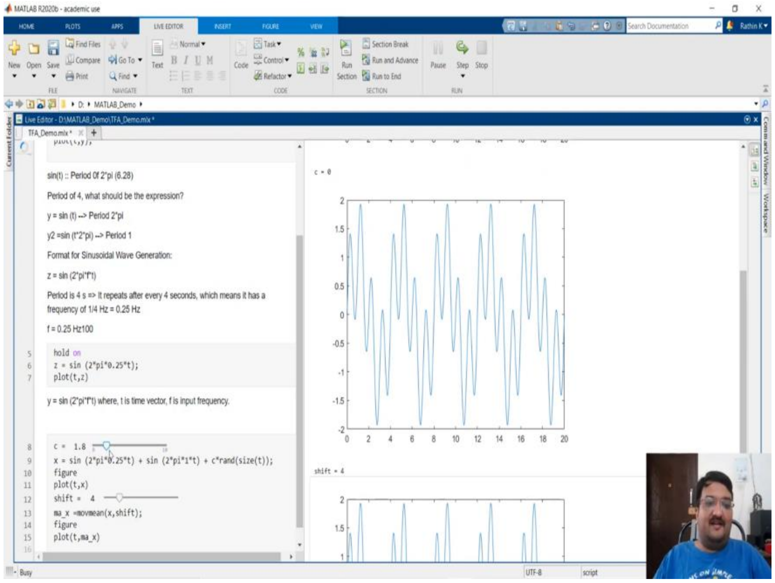
Period is 4 s => It repeats after every 4 seconds, which means it has a frequency of 1/4 Hz = 0.25 Hz

f = 0.25 Hz100

```
hold on
z = sin (2*pi*0.25*t);
plot(t,z)
```

y = sin (2*pi*f*t) where, t is time vector, f is input frequency.

```
c =  2.6
x = sin (2*pi*0.25*t) + sin (2*pi*1*t) + c*rand(size(t));
figure
plot(t,x)
shift =  0
ma_x
```



sin(t) :: Period Of 2*pi (6.28)

Period of 4, what should be the expression?

y = sin (t) --> Period 2*pi

y2 =sin (t*2*pi) --> Period 1

Format for Sinusoidal Wave Generation:

z = sin (2*pi*f*t)

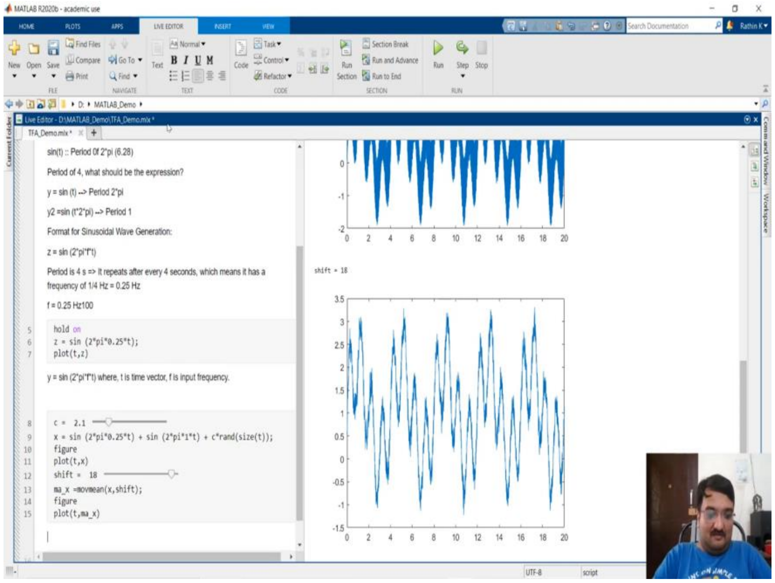
Period is 4 s => It repeats after every 4 seconds, which means it has a frequency of 1/4 Hz = 0.25 Hz

f = 0.25 Hz100

```
hold on
z = sin (2*pi*0.25*t);
plot(t,z)
```

y = sin (2*pi*f*t) where, t is time vector, f is input frequency.

```
c =  1.8
x = sin (2*pi*0.25*t) + sin (2*pi*1*t) + c*rand(size(t));
figure
plot(t,x)
shift =  4
ma_x =movmean(x,shift);
figure
plot(t,ma_x)
```



sin(t) :: Period Of 2*pi (6.28)

Period of 4, what should be the expression?

y = sin (t) --> Period 2*pi

y2 =sin (t*2*pi) --> Period 1

Format for Sinusoidal Wave Generation:

z = sin (2*pi*f*t)

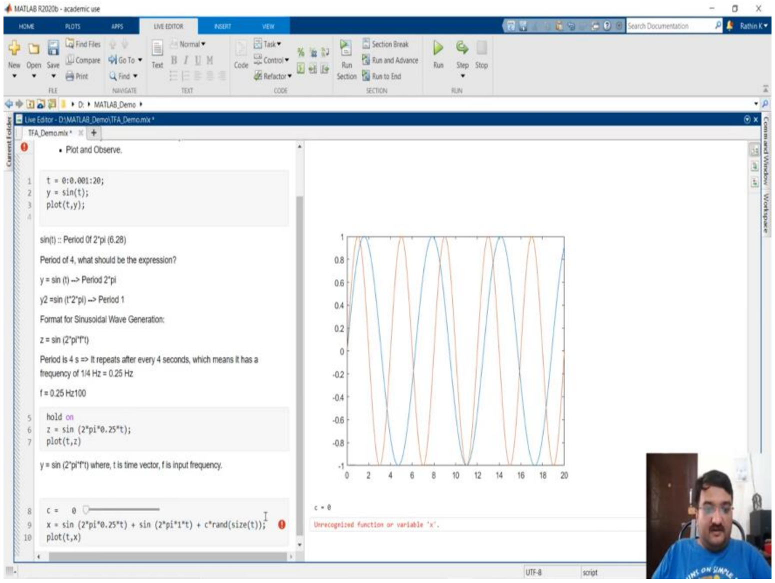
Period is 4 s => It repeats after every 4 seconds, which means it has a frequency of 1/4 Hz = 0.25 Hz

f = 0.25 Hz100

```
hold on
z = sin (2*pi*0.25*t);
plot(t,z)
```

y = sin (2*pi*f*t) where, t is time vector, f is input frequency.

```
c =  2.1
x = sin (2*pi*0.25*t) + sin (2*pi*1*t) + c*rand(size(t));
figure
plot(t,x)
shift =  18
ma_x =movmean(x,shift);
figure
plot(t,ma_x)
```



- Plot and Observe.

```
t = 0:0.001:20;
y = sin(t);
plot(t,y);
```

sin(t) :: Period Of 2*pi (6.28)

Period of 4, what should be the expression?

y = sin (t) --> Period 2*pi

y2 =sin (t*2*pi) --> Period 1

Format for Sinusoidal Wave Generation:

z = sin (2*pi*f*t)

Period is 4 s => It repeats after every 4 seconds, which means it has a frequency of 1/4 Hz = 0.25 Hz

f = 0.25 Hz100

```
hold on
z = sin (2*pi*0.25*t);
plot(t,z)
```

y = sin (2*pi*f*t) where, t is time vector, f is input frequency.

```
c =   0
x = sin (2*pi*0.25*t) + sin (2*pi*1*t) + c*rand(size(t));
plot(t,x)
```

```
c = 0
Unrecognized function or variable 'x'.
```

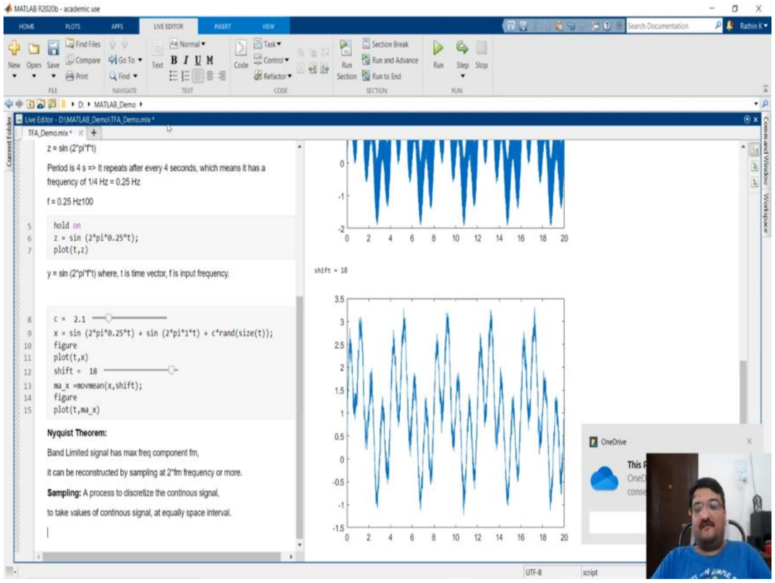
Now, what is the impact of this, c, you can see here; you can see clearly by increasing this thing, this waveform is getting distorted accordingly. Basically, we are adding more and more noise. It is the same thing what we obtained before, but added this noise; you can see that here clearly. This was our original without any kind of noise, it is looking like this. But if you add noise, little noise; so how to remove this noise, I showed it to you that we can use moving average. So, let us get the moving average. Again, you can define a moving average shift accordingly.

Shift, we can put it again; we will put a slider for the time being just give them a value from 0 to 20 with an increment of 1. Define a slider here, this will be used as a shift for moving average; moving average when version of x, let us define it ma_x is equal to movemean. This is the same function what we have used of x with shift; that is what we have defined. So, now let us plot this on a separate graph as well. Agenda here is to show you how controllability can be achieved; so just a quick illustration. So, now if I run this your c is defined which is your constant again. The shift should be a finite number that is what they are trying to say.

But you can see that this was the original waveform; this was the original waveform. You can add we have added little bit noise that is 2.6 for now. It has altered your sinusoidal some of sinusoidal waveform like this. And then by adding more and more shift in moving average; you will be able to get better and better result, in a way you are kind of recovering your original signal from the noisy signal. So, by just modifying this parameters, I believe you will get an idea by reducing your scalar to the noise and increasing the shift, you are getting more and more closer to your original waveform.

So, I believe now, this is clear this just to show you with these two scalar parameters. What we have seen last time with any three of, any three values 11.0 and 7.0 and 3.0. But here if you go further, you will get better at better signal. So, your signal fidelity, your signal quality will increase by having more and more shift in terms of moving average. So, this is just a quick recap of what we have seen in the last lecture.

(Refer Slide Time: 21:00)



z = sin (2*pi*f*t)

Period is 4 s => It repeats after every 4 seconds, which means it has a frequency of 1/4 Hz = 0.25 Hz

f = 0.25 Hz100

```
hold on
z = sin (2*pi*0.25*t);
plot(t,z)
```

y = sin (2*pi*f*t) where, t is time vector, f is input frequency.

```
c =  2.1
x = sin (2*pi*0.25*t) + sin (2*pi*1*t) + c*rand(size(t));
figure
plot(t,x)
shift =  18
ma_x =movmean(x,shift);
figure
plot(t,ma_x)
```

shift = 18

Nyquist Theorem:

Band Limited signal has max freq component fm,

it can be reconstructed by sampling at 2*fm frequency or more.

Sampling: A process to discretize the continous signal,

to take values of continous signal, at equally space interval.

Now, let us see something called as Nyquist theorem, very important when it comes to signal acquisition. We will not go line by line what exactly the theorem says and all; but I just wanted to give you people an brief idea what is it. Suppose your band limited signal you have a band limited signal. Now, band limited signal means we are not covered the entire Fourier transform yet; but your signal will have a finite frequency component. So, let say your band limited signal has maximum frequency component; I will just write freq component-fm; then it can be reconstructed by sampling at 2 × fm frequency or more.

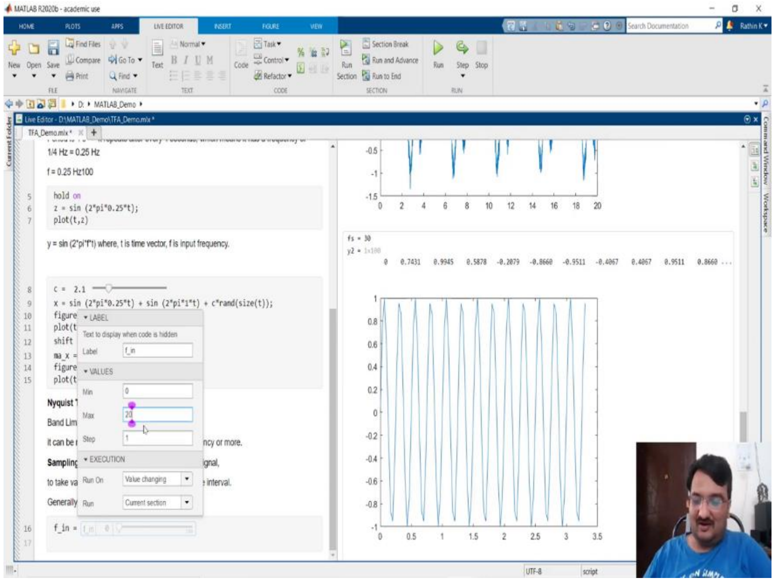
What does it says is, you have any signal which has a frequency fm; then there is something called sampling. Now, sampling means it is a; I should not say conversion, but it is a process to discretize. Or discretize the continuous signal in simpler terms to take values of continuous signal at equally space intervals.

So, I most, most of the modern day system use digitized discretized signal processing. Your computer or your mobile wherever you buy your computer or mobile; you will first ask what is the frequency supported or what is the crystal frequency, which is in terms of gigahertz. Which means, per second it supports x gigahertz operations.

So, in other words, what it says is whenever your continuous data you are taking, per seconds it operates or it partitions your continuous data into this many samples and process it. So, that is basically funda of sampling, very important when it comes to signal processing and all. And even more important to acquire the value at a proper sample data rate.

And why it is important and how it is related to your biomedical system? Again whenever you are taking any kind of biomedical signal, it has its own frequency. Physiological signals, if you are recording your EEG, if you are recording your ECG, if you are recording any other signal; it has its own frequency.

(Refer Slide Time: 24:10)



1/4 Hz = 0.25 Hz

f = 0.25 Hz100

```
hold on
z = sin (2*pi*0.25*t);
plot(t,z)
```

y = sin (2*pi*f*t) where, t is time vector, f is input frequency.

```
c =  2.1
x = sin (2*pi*0.25*t) + sin (2*pi*1*t) + c*rand(size(t));
figure
plot(t
shift
ma_x =
figure
plot(t
```

Nyquist

Band Lim

It can be ... ncy or more.

Sampling ... gnal,

to take va ... e interval.

Generally

LABEL

Text to display when code is hidden

Label: f_in

VALUES

Min: 0

Max: 20

Step: 1

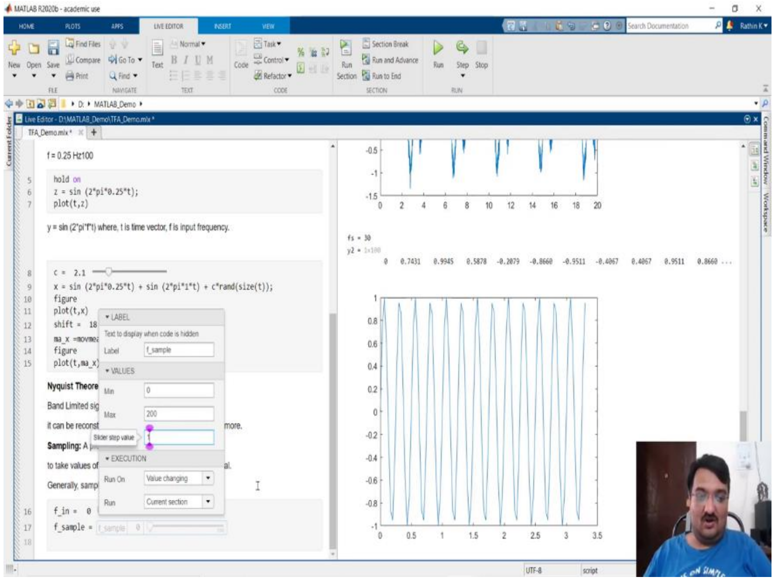
EXECUTION

Run On: Value changing

Run: Current section

```
f_in =
```

```
fs = 30
y2 = 1×100
      0    0.7431    0.9945    0.5878   -0.2079   -0.8660   -0.9511   -0.4067    0.4067    0.9511    0.8660 ...
```


MATLAB R2020b - academic use
HOME
PLOTS
APPS
LIVE EDITOR
INSERT
VIEW
Live Editor - D:\MATLAB_Demo\TFA_Demo.mlx *
TFA_Demo.mlx *

f = 0.25 Hz100

```
hold on
z = sin (2*pi*0.25*t);
plot(t,z)
```

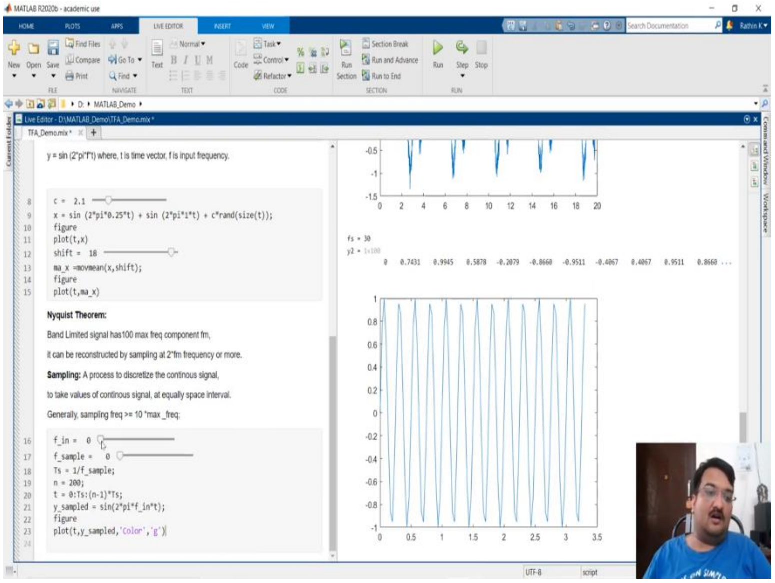
y = sin (2*pi*f*t) where, t is time vector, f is input frequency.

```
c =  2.1
x = sin (2*pi*0.25*t) + sin (2*pi*1*t) + c*rand(size(t));
figure
plot(t,x)
shift =  18
ma_x =movmea
figure
plot(t,ma_x)
```

LABEL
Text to display when code is hidden
Label: f_sample
VALUES
Min: 0
Max: 200
Slider step value: 1
EXECUTION
Run On: Value changing
Run: Current section

```
f_in =  0
f_sample =
```

```
fs = 30
y2 = 1×100
     0    0.7431    0.9945    0.5878   -0.2079   -0.8660   -0.9511   -0.4067    0.4067    0.9511    0.8660 ...
```


MATLAB R2020b - academic use
HOME
PLOTS
APPS
LIVE EDITOR
INSERT
VIEW
Live Editor - D:\MATLAB_Demo\TFA_Demo.mlx *
TFA_Demo.mlx *

y = sin (2*pi*f*t) where, t is time vector, f is input frequency.

```
c =  2.1
x = sin (2*pi*0.25*t) + sin (2*pi*1*t) + c*rand(size(t));
figure
plot(t,x)
shift =  18
ma_x =movmean(x,shift);
figure
plot(t,ma_x)
```

Nyquist Theorem:

Band Limited signal has100 max freq component fm,

it can be reconstructed by sampling at 2*fm frequency or more.

Sampling: A process to discretize the continous signal,

to take values of continous signal, at equally space interval.

Generally, sampling freq >= 10 *max _freq;

```
f_in =  0
f_sample =  0
Ts = 1/f_sample;
n = 200;
t = 0:Ts:(n-1)*Ts;
y_sampled = sin(2*pi*f_in*t);
figure
plot(t,y_sampled,'Color','g')
```

```
fs = 30
y2 = 1×100
     0    0.7431    0.9945    0.5878   -0.2079   -0.8660   -0.9511   -0.4067    0.4067    0.9511    0.8660 ...
```

So, generally, sampling frequency is at least 10 times more than your maximum occurring frequency; we will see why it is like this. But in order to just reconstruct is the word, if you want to get a better signal fidelity, if you want to get it with a proper quality. Generally, it can be 10 and even more than that; we will quickly see that. So, now all of you knows how to get a sine wave for desired frequency.

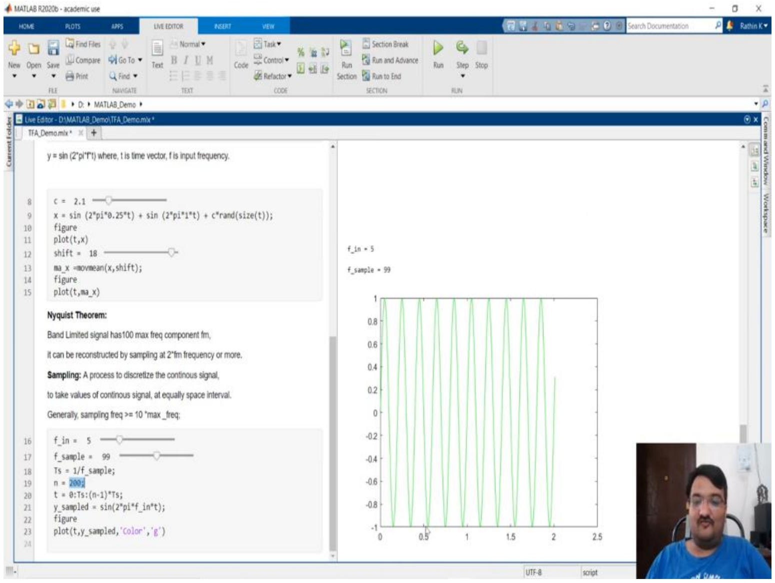
Let us just define it as f in, that is your input frequency. Instead of writing anywhere, you will put a slider here; which will give you a better idea of how frequency reads the frequency. So, it cannot be negative here; I will just increment it by 1 and let us just keep it to that limited to 20 or a 20 is fine for now, for illustration 20 is fine. Let us take a sampling one more frequency, which is your sampling frequency.

So, here again I am repeating input frequency is the frequency of your signal; sampling frequency is the frequency at which you are sampling that signal. Again, will put a control slider and it is a frequency; but earlier we kept input frequency 0 to 20. So, just to illustrate the other thing, I will keep the maximum frequency as 200. You will get an idea how we are able to get better fidelity by sampling at higher frequency rate. So, based on your sampling frequency, you can get your duration sample duration; nothing just an inverse of your sampling frequency.

Also you can get your output defined by how much finite sequence you want. So, let us take 200 samples and now we need to get a time vector. So, time vector can be obtained by 0 to t as at every Ts interval, sampling interval, it will record till how much time. You need 200 samples, so (n-1)Ts; and finally the signal. So, it will be your sampled signal, sampled at Fs frequency. I will just name it as y sampled; y sampled = sin(2πft).

And finally, we would like to plot it as well with respect to time variable, this y sample; I think that is fine. Also we will plot it in another variable another figure. So, I just added figure here and let us draw it in some other color; let us put it in green color, so will get an idea. Now, we need to set some input frequency some sampling frequency. So, let us just keep the input frequency as 5.

(Refer Slide Time: 28:02)


MATLAB R2020b - academic use
HOME PLOTS APPS LIVE EDITOR INSERT VIEW
Live Editor - D:\MATLAB_Demo\TFA_Demo.mlx *
TFA_Demo.mlx *

y = sin (2*pi*f*t) where, t is time vector, f is input frequency.

```
c =  2.1
x = sin (2*pi*0.25*t) + sin (2*pi*1*t) + c*rand(size(t));
figure
plot(t,x)
shift =  18
ma_x =movmean(x,shift);
figure
plot(t,ma_x)
```

Nyquist Theorem:

Band Limited signal has100 max freq component fm,

it can be reconstructed by sampling at 2*fm frequency or more.

Sampling: A process to discretize the continous signal,

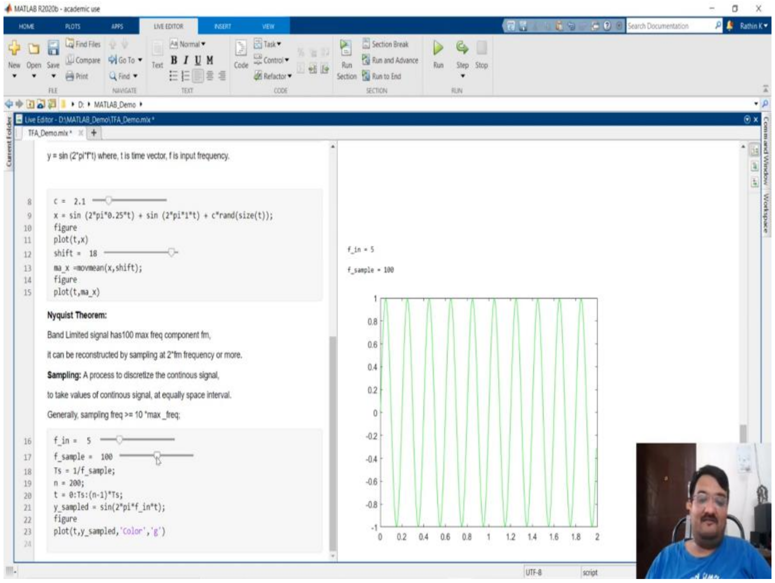
to take values of continous signal, at equally space interval.

Generally, sampling freq >= 10 *max _freq;

```
f_in =  5
f_sample =  99
Ts = 1/f_sample;
n = 200;
t = 0:Ts:(n-1)*Ts;
y_sampled = sin(2*pi*f_in*t);
figure
plot(t,y_sampled,'Color','g')
```

```
f_in = 5
f_sample = 99
```

1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.5
1
1.5
2
2.5
UTF-8
script


MATLAB R2020b - academic use
HOME PLOTS APPS LIVE EDITOR INSERT VIEW
Live Editor - D:\MATLAB_Demo\TFA_Demo.mlx *
TFA_Demo.mlx *

y = sin (2*pi*f*t) where, t is time vector, f is input frequency.

```
c =  2.1
x = sin (2*pi*0.25*t) + sin (2*pi*1*t) + c*rand(size(t));
figure
plot(t,x)
shift =  18
ma_x =movmean(x,shift);
figure
plot(t,ma_x)
```

Nyquist Theorem:

Band Limited signal has100 max freq component fm,

it can be reconstructed by sampling at 2*fm frequency or more.

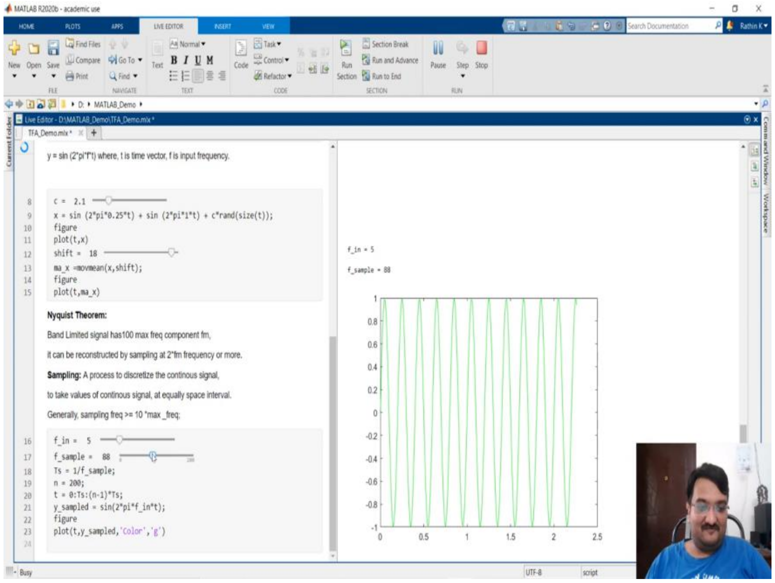
Sampling: A process to discretize the continous signal,

to take values of continous signal, at equally space interval.

Generally, sampling freq >= 10 *max _freq;

```
f_in =  5
f_sample =  100
Ts = 1/f_sample;
n = 200;
t = 0:Ts:(n-1)*Ts;
y_sampled = sin(2*pi*f_in*t);
figure
plot(t,y_sampled,'Color','g')
```

```
f_in = 5
f_sample = 100
```



y = sin (2*pi*f*t) where, t is time vector, f is input frequency.

```
c =  2.1
x = sin (2*pi*0.25*t) + sin (2*pi*1*t) + c*rand(size(t));
figure
plot(t,x)
shift =  18
ma_x =movmean(x,shift);
figure
plot(t,ma_x)
```

Nyquist Theorem:

Band Limited signal has100 max freq component fm,

it can be reconstructed by sampling at 2*fm frequency or more.

Sampling: A process to discretize the continous signal,

to take values of continous signal, at equally space interval.

Generally, sampling freq >= 10 *max _freq;

```
f_in =  5
f_sample =  88
Ts = 1/f_sample;
n = 200;
t = 0:Ts:(n-1)*Ts;
y_sampled = sin(2*pi*f_in*t);
figure
plot(t,y_sampled,'Color','g')
```

```
f_in = 5
f_sample = 88
```

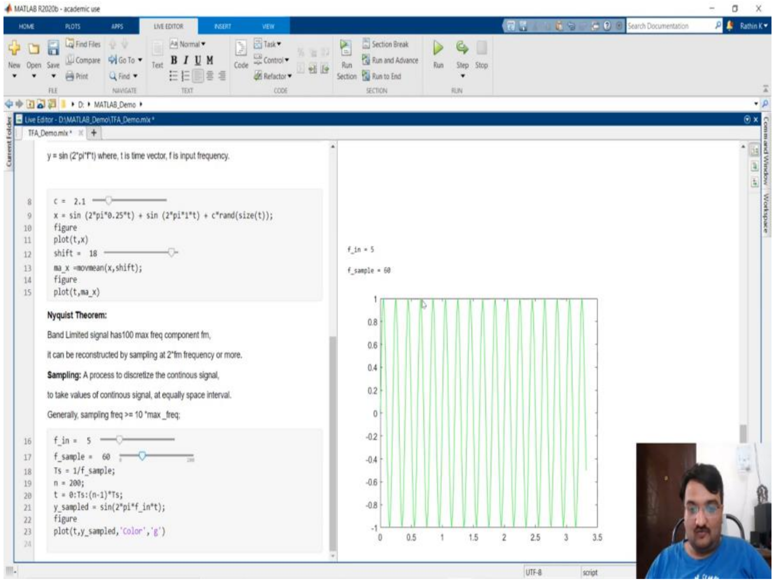
And let us give the sample frequency, which should be at least greater than that. For now, we will keep it very high; just to give you people an idea we will keep it 100; so, which is almost 20 percent more than maximum occurring frequency. Here the signal has only one frequency that is 5; so maximum frequency is 5.

And this value we can put it as a second, 99 is also fine. So, you can see here your input frequency is 5, the sampling frequency is 99 which almost 20 percent than that. And you are able to get a signal which is like input frequency file. So, we find input frequency is 5, sampling is around 100. So, in 200 sample, you will get a data of 2 seconds.

So, that is what you are almost getting a data of two second which is your duration; and also you can see that in 1 second, it is almost repeating five times. So, it is able to reconstruct the 5 hertz signal using frequency of 99; you can change this 100 as well by giving a manual value typing it, so this is fine. We are okay with this.

Now let us quickly decreasing the sampling frequency and observe the behavior; that is that will give you an idea how decreasing sampling frequency will affect the signal fidelity. This is a very simple signal sinusoidal waveform with only one component; if you have more components, it will even affect more. So, you can see here at 88. Also you are able to see the proper sine wave. Let us further decrease it, make it 60; which is almost 12 percent, 12 times more than your input frequency. And let us see how it goes.

(Refer Slide Time: 29:55)


MATLAB R2020b - academic use
HOME
PLOTS
APPS
LIVE EDITOR
INSERT
VIEW
Live Editor - D:\MATLAB_Demo\TFA_Demo.mlx *
TFA_Demo.mlx *
UTF-8
script

y = sin (2*pi*f*t) where, t is time vector, f is input frequency.

```
c = 2.1
x = sin (2*pi*0.25*t) + sin (2*pi*1*t) + c*rand(size(t));
figure
plot(t,x)
shift = 18
ma_x =movmean(x,shift);
figure
plot(t,ma_x)
```

Nyquist Theorem:

Band Limited signal has100 max freq component fm,

it can be reconstructed by sampling at 2*fm frequency or more.

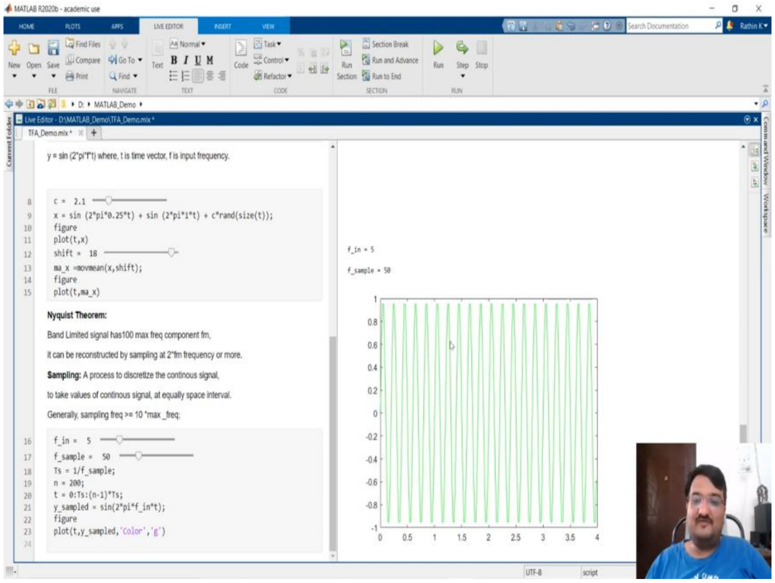
Sampling: A process to discretize the continous signal,

to take values of continous signal, at equally space interval.

Generally, sampling freq >= 10 *max _freq;

```
f_in = 5
f_sample = 60
Ts = 1/f_sample;
n = 200;
t = 0:Ts:(n-1)*Ts;
y_sampled = sin(2*pi*f_in*t);
figure
plot(t,y_sampled,'Color','g')
```

```
f_in = 5
f_sample = 60
```


MATLAB R2020b - academic use
HOME
PLOTS
APPS
LIVE EDITOR
INSERT
VIEW
Live Editor - D:\MATLAB_Demo\TFA_Demo.mlx *
TFA_Demo.mlx *
UTF-8
script

y = sin (2*pi*f*t) where, t is time vector, f is input frequency.

```
c = 2.1
x = sin (2*pi*0.25*t) + sin (2*pi*1*t) + c*rand(size(t));
figure
plot(t,x)
shift = 18
ma_x =movmean(x,shift);
figure
plot(t,ma_x)
```

Nyquist Theorem:

Band Limited signal has100 max freq component fm,

it can be reconstructed by sampling at 2*fm frequency or more.

Sampling: A process to discretize the continous signal,

to take values of continous signal, at equally space interval.

Generally, sampling freq >= 10 *max _freq;

```
f_in = 5
f_sample = 50
Ts = 1/f_sample;
n = 200;
t = 0:Ts:(n-1)*Ts;
y_sampled = sin(2*pi*f_in*t);
figure
plot(t,y_sampled,'Color','g')
```

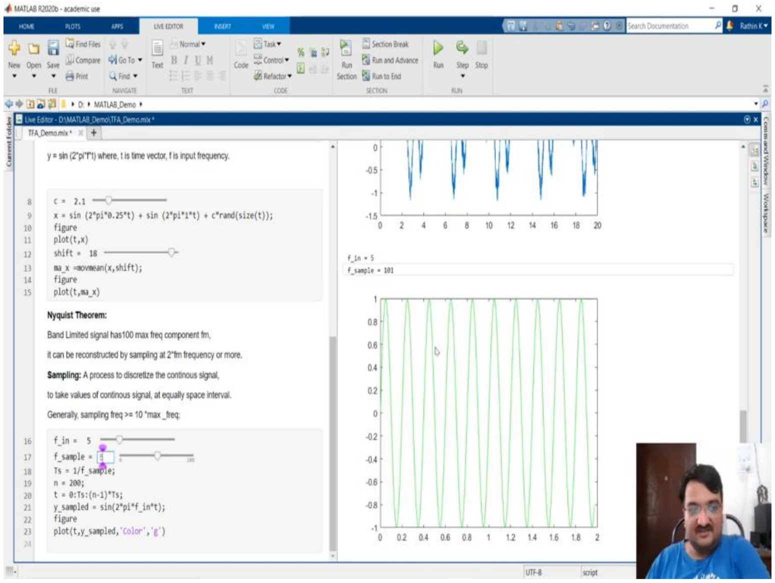
```
f_in = 5
f_sample = 50
```

So again, if you can see here, it is still processing. But, if you can see here, you are able to see the sine wave. One important point to mention here is values are also preserved as well as the frequency of the input signal has not changed at all. You see it is starting at 0, 1 sec, 2, 3, 4, 5 or 6, 5; around 5 it is stopped.

So, 5 means your input frequency is preserved. So, for input because if 5 were at 60 hertz is fine; let us go even beyond. So, I told you around 10 is okay to get your desired waveform with a higher fidelity. So, now I have changed it to 50. So, can you see little bit here, it is little tapered, or not proper inconsistency, it is not exactly the sine wave which we want.

So, when your sampling frequency or you can compare it, when your sampling frequency; that is why I kept a slider here. It is very easy to analyze when your sampling frequency is 100. What you are getting? You are getting a proper edges here; proper sinusoidal, just a second. You can see the exact behavior of how sinusoidal input looks like.

(Refer Slide Time: 31:21)



y = sin (2*pi*f*t) where, t is time vector, f is input frequency.

```
c =  2.1
x = sin (2*pi*0.25*t) + sin (2*pi*1*t) + c*rand(size(t));
figure
plot(t,x)
shift =  18
ma_x =movmean(x,shift);
figure
plot(t,ma_x)
```

Nyquist Theorem:

Band Limited signal has100 max freq component fm,

it can be reconstructed by sampling at 2*fm frequency or more.

Sampling: A process to discretize the continous signal,

to take values of continous signal, at equally space interval.

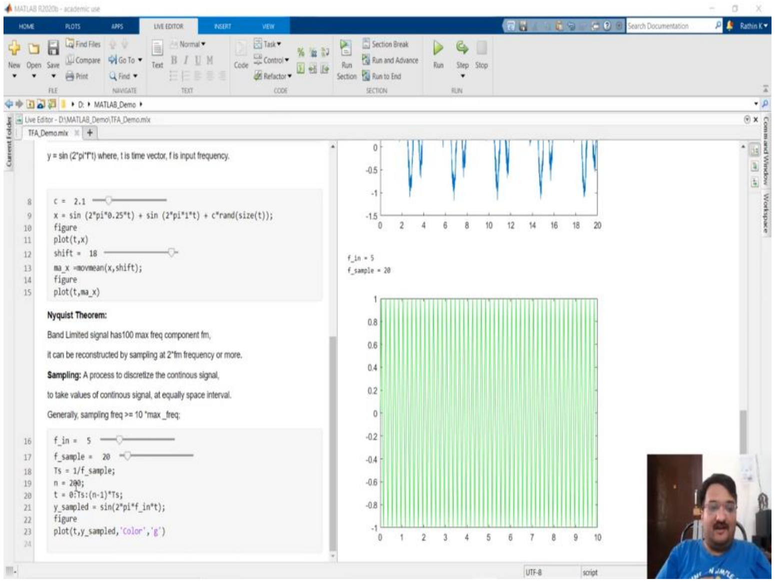
Generally, sampling freq >= 10 *max _freq;

```
f_in =  5
f_sample = 5
Ts = 1/f_sample;
n = 200;
t = 0:Ts:(n-1)*Ts;
y_sampled = sin(2*pi*f_in*t);
figure
plot(t,y_sampled,'Color','g')
```

```
f_in = 5
f_sample = 101
```

Whereas, when I change this quickly to 50, you just focus on this shape, where I just change this to 50; you will clearly see the tapered waveform or non ideal sinusoidal waveform. Now this is around 50, also you are not able to get exactly the kind of waveform. What Nyquist theorem suggests is about 10 you can reconstruct; reconstruct as in; you can approximate or you can vaguely get an idea about how your original signal look like. Let us just make it 20 to make things there to keep the sampling rate reducing even further.

(Refer Slide Time: 32:04)



y = sin (2*pi*f*t) where, t is time vector, f is input frequency.

```
c =  2.1
x = sin (2*pi*0.25*t) + sin (2*pi*1*t) + c*rand(size(t));
figure
plot(t,x)
shift =  18
ma_x =movmean(x,shift);
figure
plot(t,ma_x)
```

Nyquist Theorem:

Band Limited signal has100 max freq component fm,

it can be reconstructed by sampling at 2*fm frequency or more.

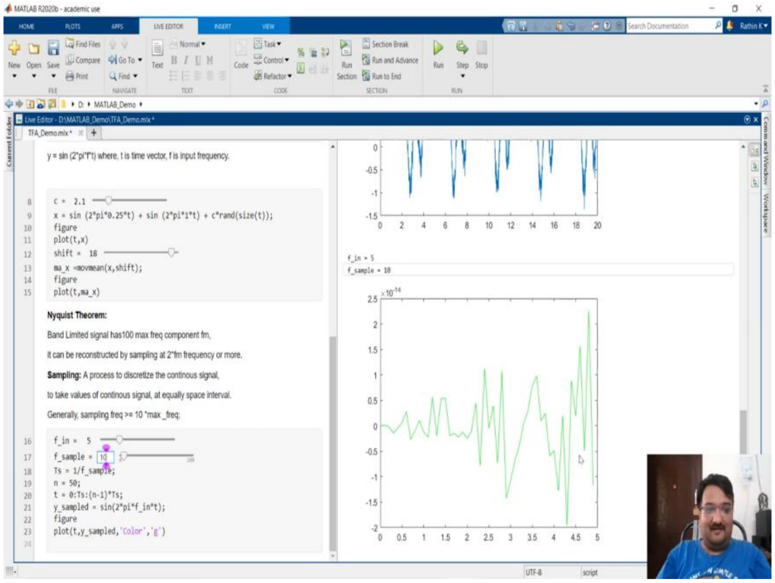
Sampling: A process to discretize the continous signal,

to take values of continous signal, at equally space interval.

Generally, sampling freq >= 10 *max _freq;

```
f_in =  5
f_sample =  20
Ts = 1/f_sample;
n = 200;
t = 0:Ts:(n-1)*Ts;
y_sampled = sin(2*pi*f_in*t);
figure
plot(t,y_sampled,'Color','g')
```

```
f_in = 5
f_sample = 20
```



y = sin (2*pi*f*t) where, t is time vector, f is input frequency.

```
c =  2.1
x = sin (2*pi*0.25*t) + sin (2*pi*1*t) + c*rand(size(t));
figure
plot(t,x)
shift =  18
ma_x =movmean(x,shift);
figure
plot(t,ma_x)
```

Nyquist Theorem:

Band Limited signal has100 max freq component fm,

it can be reconstructed by sampling at 2*fm frequency or more.

Sampling: A process to discretize the continous signal,

to take values of continous signal, at equally space interval.

Generally, sampling freq >= 10 *max _freq;

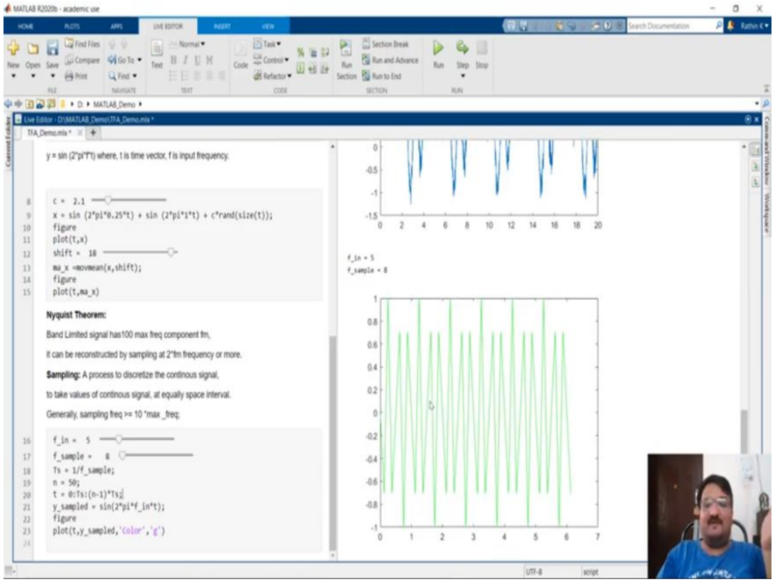
```
f_in =  5
f_sample =  10
Ts = 1/f_sample;
n = 50;
t = 0:Ts:(n-1)*Ts;
y_sampled = sin(2*pi*f_in*t);
figure
plot(t,y_sampled,'Color','g')
```

```
f_in = 5
f_sample = 10
```

So, you can see that it is. One more thing to notice here is when your sampling rate is 20, and your sample number of samples are 200; you are getting 10 seconds here, as you can see here. Additionally, your input frequency is 5; so, but see this as the input frequency is maintained in one second only five times it repeats. But you are getting almost a spike kind of waveform; let us decrease it even further.

But before that, I will just reduce the number of samples. So, we will get to see the how many repetitions and all vary, clearly. We have now, as the sampling frequency is less than 20 around 10 or something; I think 50 samples should also fine. And let me just decrease the sampling frequency; we will go to 10, which is exactly what Nyquist as suggested, like at least 10. But better the sampling frequency, the better the signal fidelity; but the computation power would be more. So, if you see that here around 10, you are not getting the kind of waveform we should be getting.

(Refer Slide Time: 33:18)



y = sin (2*pi*f*t) where, t is time vector, f is input frequency.

```
c =  2.1
x = sin (2*pi*0.25*t) + sin (2*pi*1*t) + c*rand(size(t));
figure
plot(t,x)
shift =  18
ma_x =movmean(x,shift);
figure
plot(t,ma_x)
```

Nyquist Theorem:

Band Limited signal has100 max freq component fm,

it can be reconstructed by sampling at 2*fm frequency or more.

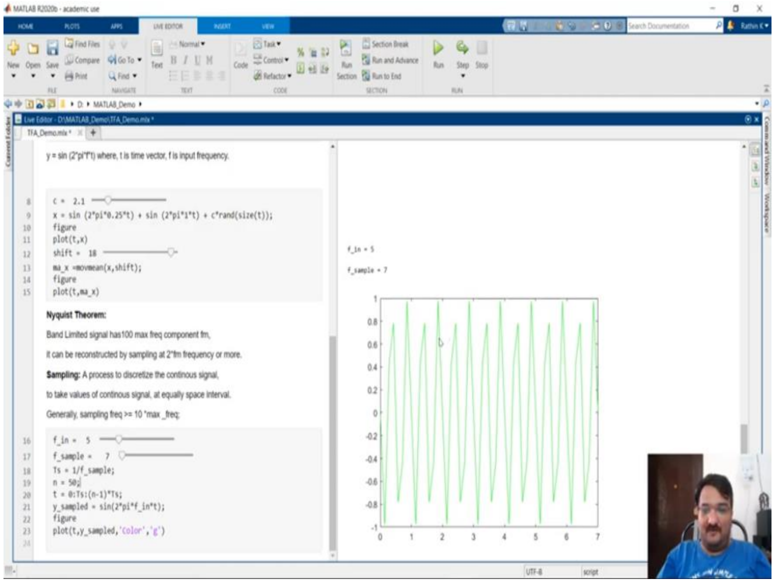
Sampling: A process to discretize the continous signal,

to take values of continous signal, at equally space interval.

Generally, sampling freq >= 10 *max _freq;

```
f_in =  5
f_sample =   8
Ts = 1/f_sample;
n = 50;
t = 0:Ts:(n-1)*Ts;
y_sampled = sin(2*pi*f_in*t);
figure
plot(t,y_sampled,'Color','g')
```

```
f_in = 5
f_sample = 8
```



y = sin (2*pi*f*t) where, t is time vector, f is input frequency.

```
c =  2.1
x = sin (2*pi*0.25*t) + sin (2*pi*1*t) + c*rand(size(t));
figure
plot(t,x)
shift =  18
ma_x =movmean(x,shift);
figure
plot(t,ma_x)
```

Nyquist Theorem:

Band Limited signal has100 max freq component fm,

it can be reconstructed by sampling at 2*fm frequency or more.

Sampling: A process to discretize the continous signal,

to take values of continous signal, at equally space interval.

Generally, sampling freq >= 10 *max _freq;

```
f_in =  5
f_sample =   7
Ts = 1/f_sample;
n = 50;
t = 0:Ts:(n-1)*Ts;
y_sampled = sin(2*pi*f_in*t);
figure
plot(t,y_sampled,'Color','g')
```

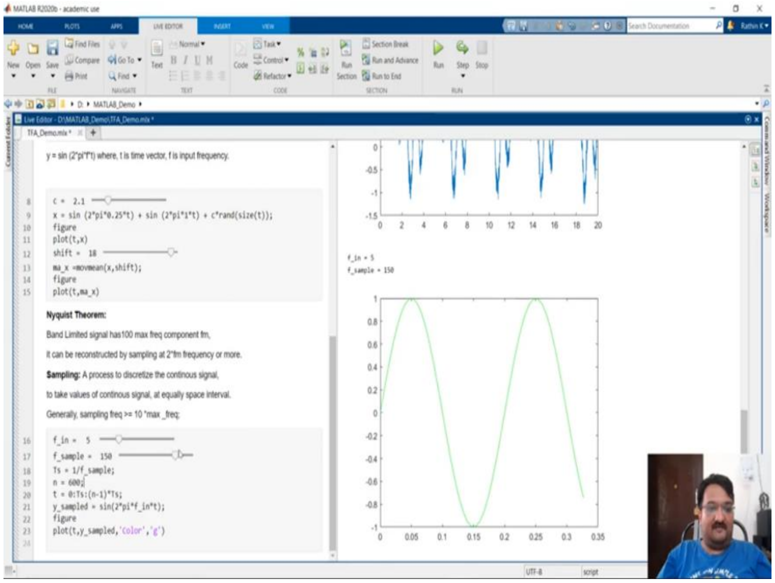
```
f_in = 5
f_sample = 7
```

So, greater than 10 if I go 11, then also you see almost like polarity is somehow maintained. But if you have to keep it greater than as I mentioned greater than or equal to 10; or sampling frequency. Let us keep it at least more than a 2 × fm, which we have kept; so you see the kind of shape is preserved.

Further if you go down what happened? You will not be able to reconstruct signal; you will not be able to get the waveform. You will be able to get the waveform; you will be able to plot it as well. But the shape of your original sine wave will not be preserved; this is not the kind of sine wave if you want to see.

Further if you go, it depends also on your exact value of sampling frequency, and how it is related to your input frequency et-cetera. See, it is almost tapered, your values are not also reaching up to the where it should reach ±1 and all. So, the moral of the story idea here of this illustration is to give you people an idea, the higher the sample frequency; let us keep it around 150.

(Refer Slide Time: 34:26)



y = sin (2*pi*f*t) where, t is time vector, f is input frequency.

```
c =  2.1
x = sin (2*pi*0.25*t) + sin (2*pi*1*t) + c*rand(size(t));
figure
plot(t,x)
shift =  18
ma_x =movmean(x,shift);
figure
plot(t,ma_x)
```

Nyquist Theorem:

Band Limited signal has100 max freq component fm,

it can be reconstructed by sampling at 2*fm frequency or more.

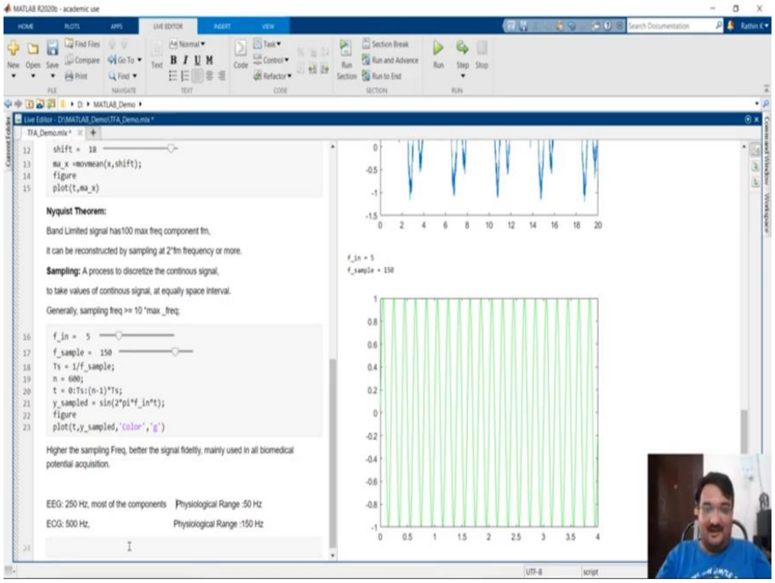
Sampling: A process to discretize the continous signal,

to take values of continous signal, at equally space interval.

Generally, sampling freq >= 10 *max _freq;

```
f_in =  5
f_sample =  150
Ts = 1/f_sample;
n = 600;
t = 0:Ts:(n-1)*Ts;
y_sampled = sin(2*pi*f_in*t);
figure
plot(t,y_sampled,'Color','g')
```

```
f_in = 5
f_sample = 150
```



```
shift =  18
ma_x =movmean(x,shift);
figure
plot(t,ma_x)
```

Nyquist Theorem:

Band Limited signal has100 max freq component fm,

it can be reconstructed by sampling at 2*fm frequency or more.

Sampling: A process to discretize the continous signal,

to take values of continous signal, at equally space interval.

Generally, sampling freq >= 10 *max _freq;

```
f_in =  5
f_sample =  150
Ts = 1/f_sample;
n = 600;
t = 0:Ts:(n-1)*Ts;
y_sampled = sin(2*pi*f_in*t);
figure
plot(t,y_sampled,'Color','g')
```

Higher the sampling Freq, better the signal fidelity, mainly used in all biomedical potential acquisition.

EEG: 250 Hz, most of the components	Physiological Range :50 Hz
ECG: 500 Hz,	Physiological Range :150 Hz

```
f_in = 5
f_sample = 150
```

You will get a very nice waveform; but in this case, you need to increase this number of samples to see a better visualization, to see the more repetitions; you will get an idea. But you can clearly see this thing here. You are getting an excellent sine wave here; because your sampling frequency is way higher. So, concluding of what we have seen that higher the sampling frequency, better the signal fidelity. Mainly used in all biomedical potential acquisition EEG. Generally, for EEG 250 hertz are fine; because most of the components lies in first most of the components lies in 50 hertz.

And then if you want even further components, you can increase the frequency ECG heart rate. Most of you knows it will be around maximum around 100 hertz or 150 hertz; not more than that; that is what ECG’s physiological range. That is why if you are making a device to record ECG, you can put it as a 1 kilo hertz; a sampling frequency 1 kilohertz or even 500 should be good enough to reconstruct your input ECG signal.

Also there is something called aliasing and all that also depends on the selection of your sampling frequency and input frequency. We will not go into the detail in that, but that is also an important thing. And in the upcoming lectures, what we have seen is frequencies. This is as I mentioned, physiological range, physiological range 100 hertz for ECG; or 100 or 150 ranges in maximum frequency.

There is some cardiac arrhythmia; some disorders which might force your heart pumping; or frequency of hearts even further. Here again, ECG if I say a physiological range, which means that it is around 100 hertz or let say 50 hertz; which means it is a frequency range in normative condition. There are disorders like say let say epilepsy or something, where it might go beyond that as well.

But the moral of the story is based on the physiological range; you should be able to at least approximate what should be the sampling frequency accordingly, you can able to reconstruct the original signal. So, I believe, you got some idea, next time when we will meet; we will see how to map your time domain waveform with frequency using Fourier transform. And how the computation complexity differs when you perform Fourier transform as per definition; why FFT is called fast Fourier transform. We will see in the next lecture. Bye. Take care.