

Tools in Scientific Computing
Prof. Aditya Bandopadhyay
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

Lecture - 39
Audio analysis – determine motor RPM

Hello everyone. Welcome to lecture 39, we have moved into the last week where we are going to utilize some of the tools and learn new tools, in order to analyze some experimental data. And many of you I am sure take interest in experiments and during experiments you obviously, collect a lot of data, data collection is not what it used to be what it was in your school days or something.

It is now more electronic, there is data loggers, all sorts of sensors, temperature sensors, pressure sensors things like that. And so, in this electronic age it is expected that once you have data you are going to analyze the data and get something useful out of it right. So, in this particular lecture we are going to touch upon a bit of audio processing.

Well I call it audio processing, but more generally it is signal processing. So, few days back I was tuning the carburetor of my bike and once I switch it on I can see the meter on the bike it says something close to 1500 RPM and it is something which you can make out clearly and it is sort of noise it is not too fast it is not too slow ok.

So, then that gives me an idea whether we can predict or tell what the RPM of a certain devices with some audio clues. And what is audio? I mean whatever you perceive as sound is really, how a pressure fluctuation is reaching your microphone or your ear right, and whenever you have various machines that that whiz or were that go around you.

For example, the fridge compressor or your fan or some high frequency sound of a; of an insect sort of associated with some kind of periodic motion that particular device or insect is making right. So, in this lecture we are going to see how we can predict the RPM of two machines that I had I mean in the workshop.

(Refer Slide Time: 03:20)

The image shows a split-screen view. On the left is a browser window displaying a Wikipedia article titled 'Synchronous speed'. The article includes the following text:

Synchronous speed [edit]

The synchronous speed of a synchronous motor is given^[c] in RPM, by:

$$N_s = 60 \frac{f}{P} = 120 \frac{f}{P}$$

and in rad s⁻¹, by:

$$\omega_s = 2\pi \frac{f}{P} = 4\pi \frac{f}{P}$$

where:

- f is the frequency of the AC supply current in Hz,
- p is the number of magnetic poles,
- P is the number of pole pairs (rarely, planes of commutation), $P = p/2$.

Examples [edit]

A single-phase, 4-pole (2-pole-pair) synchronous motor is operating at an AC supply frequency of 50 Hz:

$$N_s = 60 \times \frac{50}{2} = 1500 \text{ rpm}$$

A three-phase, 12-pole (6-pole-pair) synchronous motor is operating at an AC supply frequency of 50 Hz:

$$N_s = 60 \times \frac{50}{6} = 600 \text{ rpm}$$

The number of magnetic poles, p , is equal to the number of coil groups per phase. To determine the number of poles, which is 3. The coils may span several slots in the stator core, making it tedious to count the number of poles. The number of magnetic poles in the rotor is equal to the number of magnetic poles in the stator.

Construction [edit]

The principal components of a synchronous motor are the stator and the rotor.^[c] The stator is a cylindrical shell with several slots in its inner surface. Each slot contains a coil of wire. The rotor is a cylindrical shell with several slots in its inner surface. Each slot contains a coil of wire. The rotor is a cylindrical shell with several slots in its inner surface. Each slot contains a coil of wire.

On the right is a handwritten note on a whiteboard. It lists:

1. Benchgrinder DC, Universal
2. Pedestal fan

Below the list, there is a diagram of a pedestal fan with a label 'AC induction motor'. To the right of the diagram, it says 'Synchronous speeds'. Below the diagram, there is a calculation: '50 Hz AC mains frequency' with an arrow pointing to '1500 rpm' and '75 25 1500 8 60 Hz'. Below this, it says '4 pole motor' and '120 f / P = 120 x 50 / 4 = 1500 rpm'. At the bottom, it says 'How? -> Tachometer oscilloscope'.

So, the 1st machine is the bench grinder and the other machine is a simple pedestal fan. And well everyone knows what a pedestal fan is it is one of those fans, which has a base it has a long stem then from the side it looks something like this, this is shroud and this three blades like this and this goes to mains. So, it is a device which runs of mains and it does have an AC induction motor ok.

A bench grinder is also something similar how it looks like is there is a central bearing holder and then there is two shafts and there is two wheels connected. So, these are centered grits of aluminium oxide carborundum whatever it is. It is a; it is a device where this rotate these two discs rotate and you can sharpen various things so, it is called as a bench grinder. It is also AC induction motor, but whether or not when they have the same frequency well, AC machines they run at synchronous speeds.

(Refer Slide Time: 05:07)

Audio of a pedestal fan

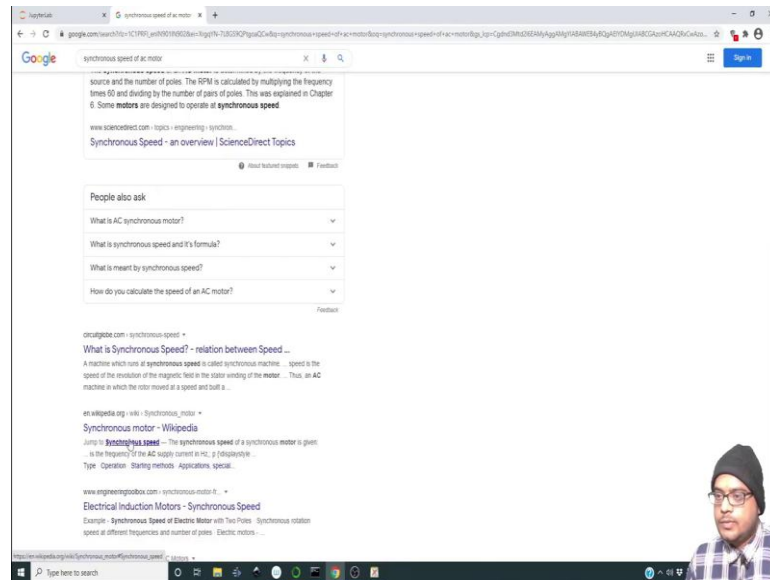
(Refer Slide Time: 05:30)

Audio of a bench grinder

This is quite different from how other kinds of motors, for example, DC motors or universal motors run ok. So, induction motors have a synchronous speed well not all kinds of induction motors. This different kinds like shaded pole motors which do not run at in synchronous speeds, DC motors have a large speed regulation depending on the load, universal motors tend to run at significantly higher RPM's and that also depends on the load.

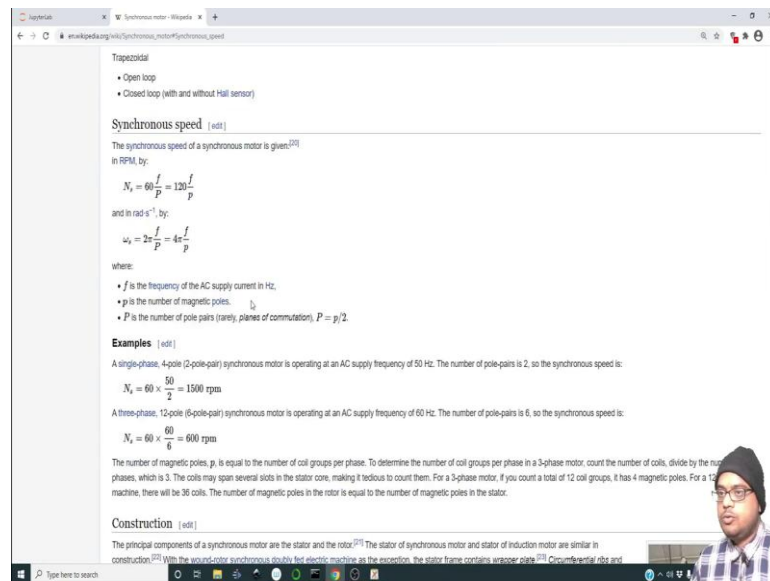
But, AC induction motors is the one you find in your ceiling fan and all these kinds of appliances, there are synchronous speeds. The synchronous speed is dependent on the number of poles the frequency ok. So, if you have 50 hertz as the AC mains frequency, so the synchronous speed will be something like 1500 rpm and the formula is quite.

(Refer Slide Time: 07:11)



So, let me just show that formula.

(Refer Slide Time: 07:27)



So, it is simply equal to $120 \frac{f}{p}$ ok. So, for a single phase 4 pole, so this is for 4 pole motor and the formula is $60 \frac{f}{p}$ rather $120 \frac{f}{p}$ alright. So, these are the number of pairs ok. So, $120 \times \frac{50}{4}$ total number of poles, so there is 2 pairs, but there is 4 poles.

So, 30 and this gives you something like 1500 right. So, that is the rpm that we have of the motor and that rpm of the motor when converted to hertz. So, you cannot just say that this is the hertz that this is the motor is running at because the number of poles are different.

And so, the corresponding hertz or per second calculation cycles per second is 1500 by 60 right. So, that is the hertz of the device. So, what do we get 75/ 3, 25. So, we have 25 hertz as the frequency of a 4 pole induction motor. And let us see whether we can predict that the rpm of this induction motor is something which we can get from an audio source.

(Refer Slide Time: 09:20)

The image shows a Jupyter Notebook window with the following code and output:

```

import numpy as np;
import matplotlib.pyplot as plt;
plt.rcParams.update({"text.usetex": True});
import scipy.io.wavfile as sw

sam, d = sw.read("tablefan.wav");
print(sam)
print(np.size(d))

16000
322560

```

The whiteboard contains handwritten notes and calculations:

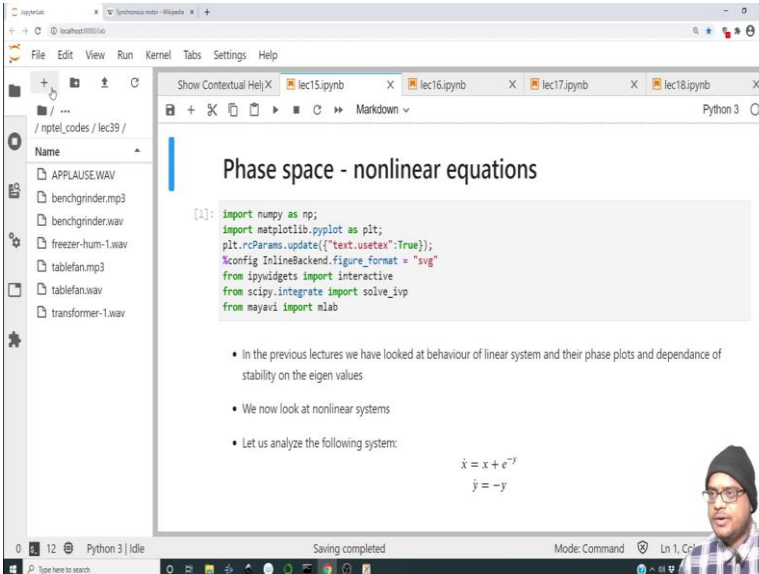
- 50 Hz AC mains frequency
- 1500 rpm 4 pole motor
- Calculation: $120 \frac{f}{P} = 120 \times \frac{50}{4} \approx 1500$
- 1500 Hz
- 16 kHz d: data
- 16,000 samples / 4 = 1 cycle
- 20.8
- Handwritten text: "How? Tachometer, Stroboscope"

And how do you actually find the frequency? How do you actually do it? Well you use a tachometer there is various kinds of meters you can attach it to the spindle it rotates or you can use a strobe ok. So, in your basic electrical machines lab you might have done something like this, but we do not bother with all this these are sort of.

So, this is a non contact method yeah it requires you to have a stroboscope this requires you to have a tachometer there is laser tachometers as well, the old kind it involves you mating the spindle of the tachometer with the rotating shaft. But, let us see whether we can do something even simpler. So, what I did, I is I took my mobile phone I kept it over here and I turned on sound recording when the fan was on. And I took my mobile phone and I kept it on the casing of this bench grinder and I recorded a bit of audio.

How that audio can give me insight into the frequency of rotation of the motor is what we are going to study ok. So, we are going away from the traditional techniques something which is quite cheap you can do it at your home as well and at the end of the lecture I do hope that you will take this technique and try to figure out various the running frequencies of various motors or things around your house ok.

(Refer Slide Time: 11:09)



The screenshot shows a Jupyter Notebook window with the following content:

Phase space - nonlinear equations

```
[1]: import numpy as np;
import matplotlib.pyplot as plt;
plt.rcParams.update({"text.usetex": True});
%config InlineBackend.figure_format = "svg"
from ipynbwidgets import interactive
from scipy.integrate import solve_ivp
from mayavi import mlab
```

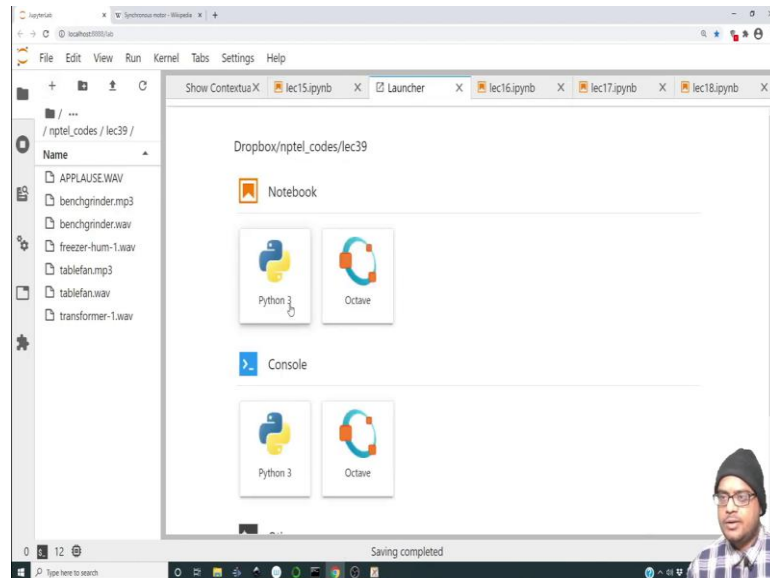
- In the previous lectures we have looked at behaviour of linear system and their phase plots and dependence of stability on the eigen values
- We now look at nonlinear systems
- Let us analyze the following system:

$$\begin{aligned}\dot{x} &= x + e^{-y} \\ \dot{y} &= -y\end{aligned}$$

The interface also shows a file explorer on the left with files like APPLAUSE.WAV, benchgrinder.mp3, benchgrinder.wav, freezer-hum-1.wav, tablefan.mp3, tablefan.wav, and transformer-1.wav. A small video inset of a person is visible in the bottom right corner.

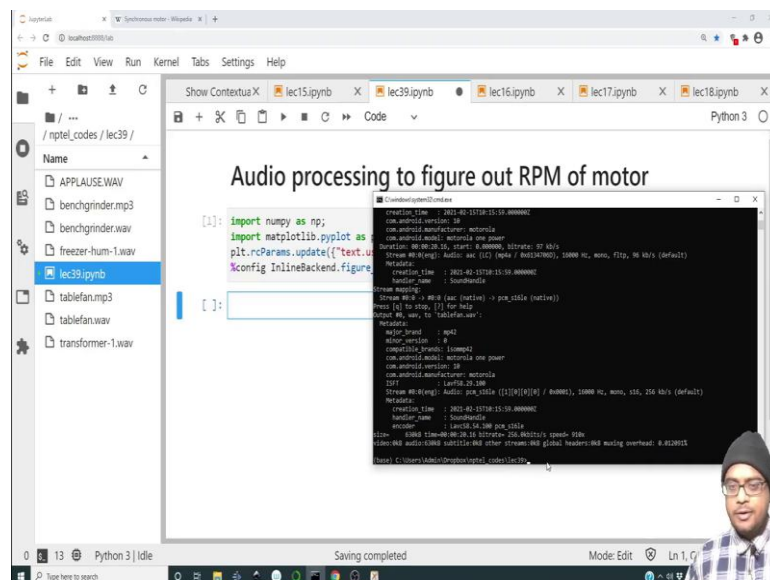
So, let me go to JupyterLab. So, what so this is these are the old files. So, let me create a new file it is Python 3, I do not know why there is some bug.

(Refer Slide Time: 11:11)



But I need to select it again from here.

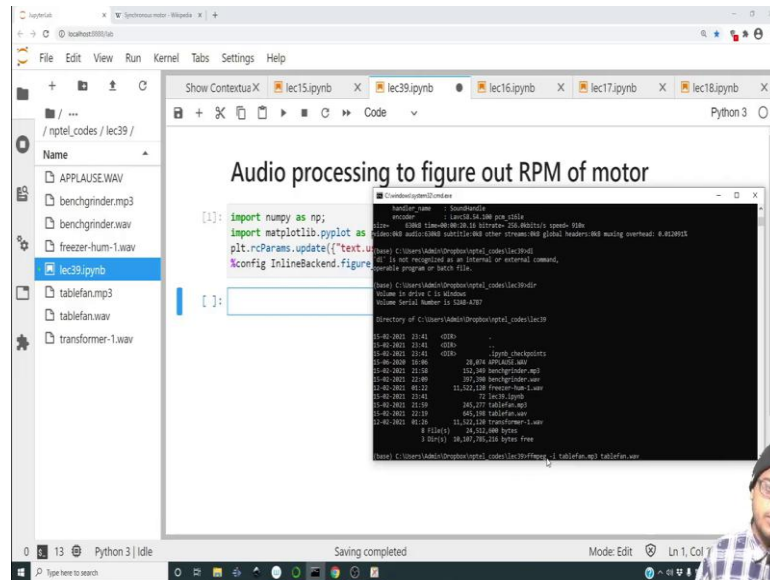
(Refer Slide Time: 11:21)



. So, I have created a new file let me rename it as lec39 alright. So, first things first, let us declare what this program is going to be about it is going to be about audio alright. So, let us import the usual things we will require numpy and all this. So, let me run this and what I did was I recorded this mp3.

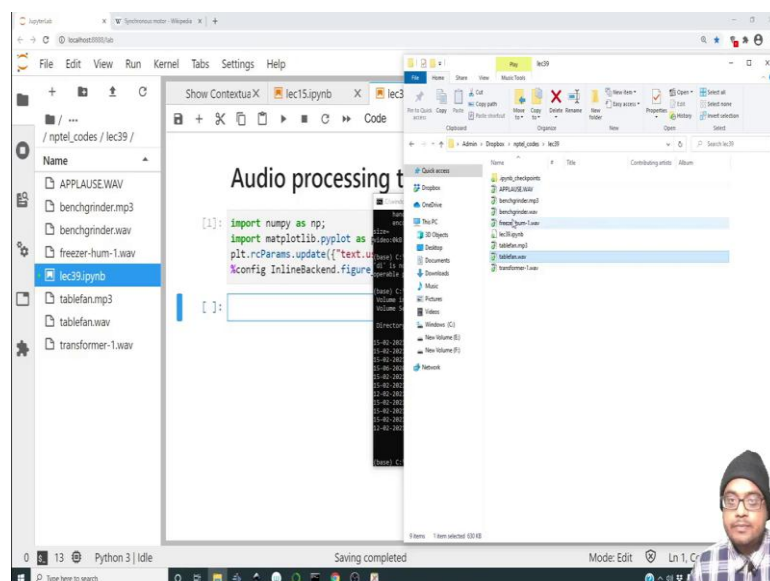
Because my audio recorder the audio recorder I have in my mobile phone it gives me an output in mp3. In order to process the audio files I need to have it in dot wav format and how I did it was I went to the command prompt I navigated to the folder and I executed a command in ffmpeg.

(Refer Slide Time: 12:38)

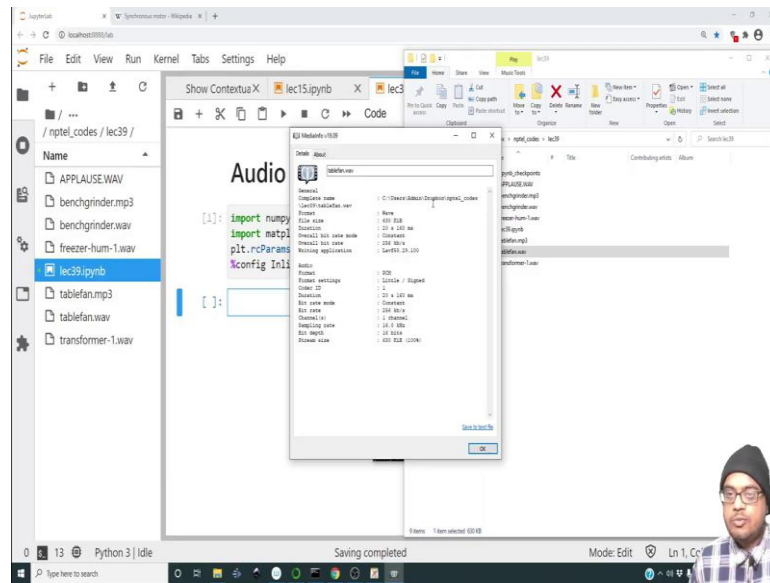


So, ffmpeg is a set of encoders and decoders very efficient and I gave an input file as tablefan. mp3 and the output file has tablefan.wav.

(Refer Slide Time: 12:52)

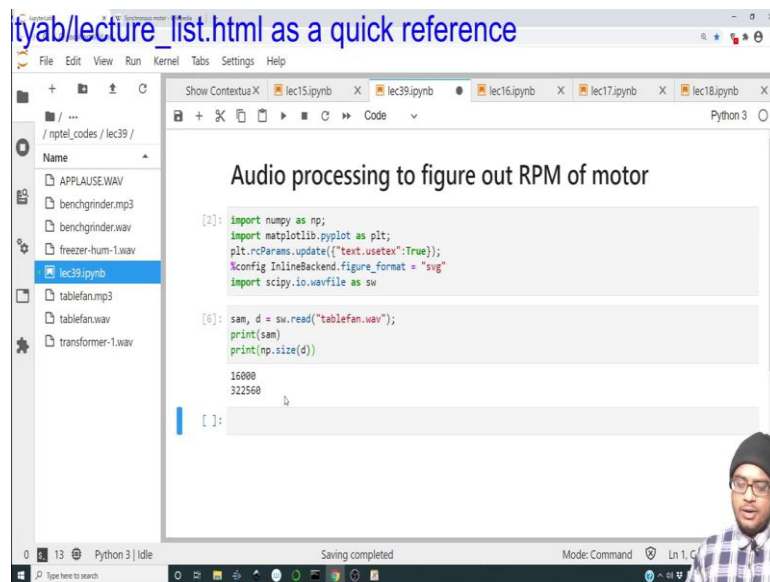


(Refer Slide Time: 12:58)



So, dot wav file let me go over here and get the media info maybe it is a bit too small to read, but yeah this gives you all the information, but we can fetch all these information from Python as well.

(Refer Slide Time: 13:14)



So, before going to that we need to do import the module which is going to help us to import a .wav file ok. So, **import scipy.io.wavfile as sw** and let me run this cell. So, let us do it. So, `sam, d = sw.read("tablefan.wav")`, we are going to give the name of the file. So, the first file we are going to analyze is the table fan.

So, tablefan.wav alright. So, let me run this. So, we have loaded the wav file and what do the sam and d contain? So, let me print out what sam is, it is 1600 and let me print out what d is, but it is a big; it is a big array, but we can query the size. So, it is an array of size 32000 no its 322560. So, it is 322560 alright. So, what is the 16000, it is the sampling rate of the file meaning that audio file has in 1 cycle, it has 16,000 samples ok.

So, it is samples per second. So, it is 16 kilo hertz basically. And so, the number of the size of d that is, so d contains the data. So, d is it looks roughly 20 times what this number is, meaning that the file we expect to be of 20 seconds duration. So, let us see whether it is really 20 seconds. Let us go over here let me query the media info and it is 20 seconds great. So, what it is table fan.

(Refer Slide Time: 15:43)

The screenshot shows a Python IDE with a Jupyter notebook. The code cell contains the following Python code:

```
[1]: sam, d = sv.read("tablefan.wav");
N = np.size(d);
t = np.arange(N)*1.0/sam;
plt.plot(t, d); plt.xlabel("Time")
```

Below the code is a plot of the audio signal. The y-axis is labeled "Audio signal near fan" and ranges from -15000 to 20000. The x-axis is labeled "Time" and ranges from 0.0 to 5.0. The plot shows a noisy signal that fluctuates between approximately -10000 and 10000.

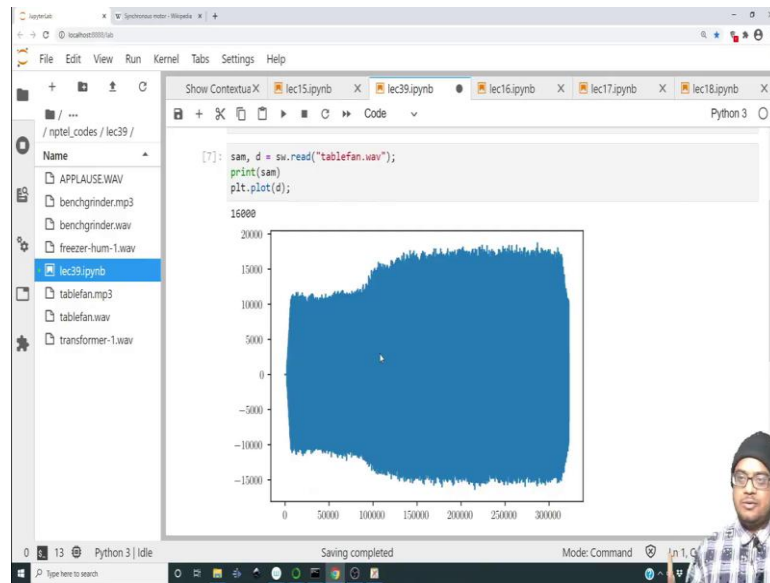
Handwritten notes on the right side of the slide include:

- tablefan.wav : 20s
- 1s : 16000 sample
- DFT FFT
- $x[n]$ $n = 0 \dots N-1$
- $y(k) = \sum_{n=0}^{N-1} e^{-2\pi i k n / N} x[n]$ $i = \sqrt{-1}$
- $x(n) = \frac{1}{N} \sum_{k=0}^{N-1} e^{2\pi i k n / N} y(k)$
- $2\pi \times \frac{0}{N} \rightarrow 2\pi \frac{N-1}{N}$
- $x = \sin(5 \cdot 2\pi t)$

A small video inset in the bottom right corner shows a person wearing a beanie and glasses, likely the presenter.

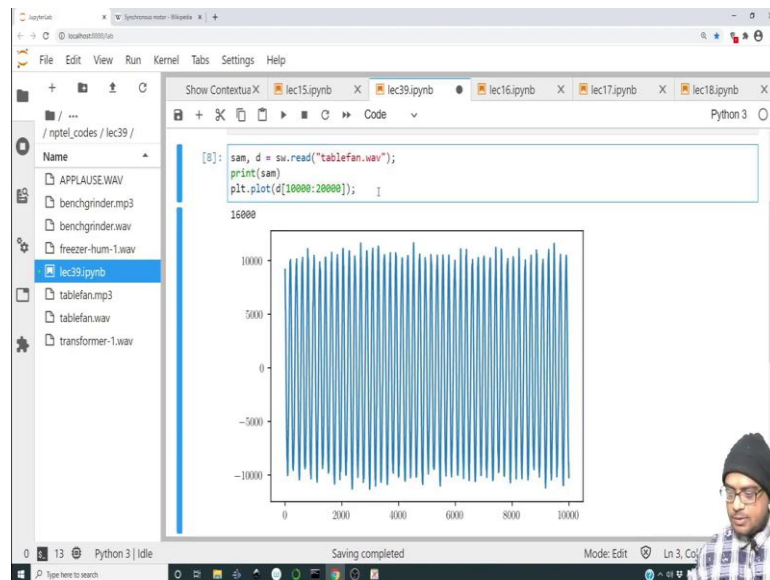
So, tablefan.wav is a file of 20 seconds, 1 second contains 16000 samples and that is what the information we have and d is obviously, the data. So, what we can do is we can plot the data.

(Refer Slide Time: 16:02)



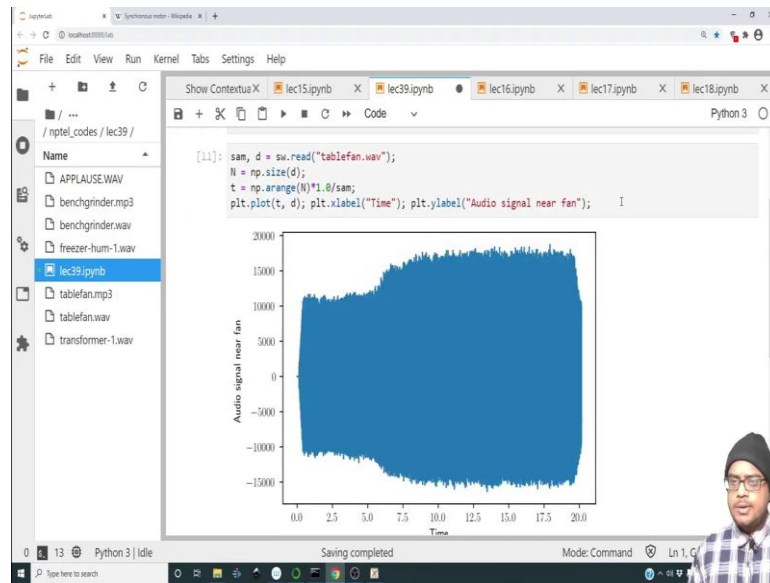
So, `plt.plot` and we can simply plot `d`, this is the data alright and it is just a bunch of waveforms.

(Refer Slide Time: 16:22)



So, let me pick out a small waveform. So, that let me pick out something between a million and this to something like this, it is just a waveform cannot really understand anything from a waveform can you.

(Refer Slide Time: 16:43)



That is where Fourier transforms come into the picture. So, well let me declare $N = \text{np.size}(d)$ that is the total number of samples right. So, on the one hand we have d on the other hand we have the sample rate and N . So, we can actually create an array. So, we can create $t = \text{np.arange}(N)*1.0/\text{sam}$, just to make the time axis we can `plt.plot(t, d)`. So, that gives us 20 seconds.

So, basically what we have over here is it is the Audio signal near a fan let me suppress this right, so far so good, but how do I make sense of this data I mean it is too much data in this I do not know what frequencies it contains. And by now you might have guess that ok he is going to take a Fourier transform of it and find out the frequencies and if you thought this you are absolutely correct, that is exactly what we are going to do just to give you a small background.

So, we are going to do a discrete Fourier transform and it is called as a fast Fourier transform. And what this is going to contain do is do the following. So, it is going to take a signal $x(n)$ where n goes from 0 to $N - 1$. It is going to transform it to the from time domain to frequency domain.

So, we have $y(k) = \sum_{n=0}^{N-1} e^{-2\pi i \frac{kn}{N}} x(n)$ and you may recognize this as being the discrete form

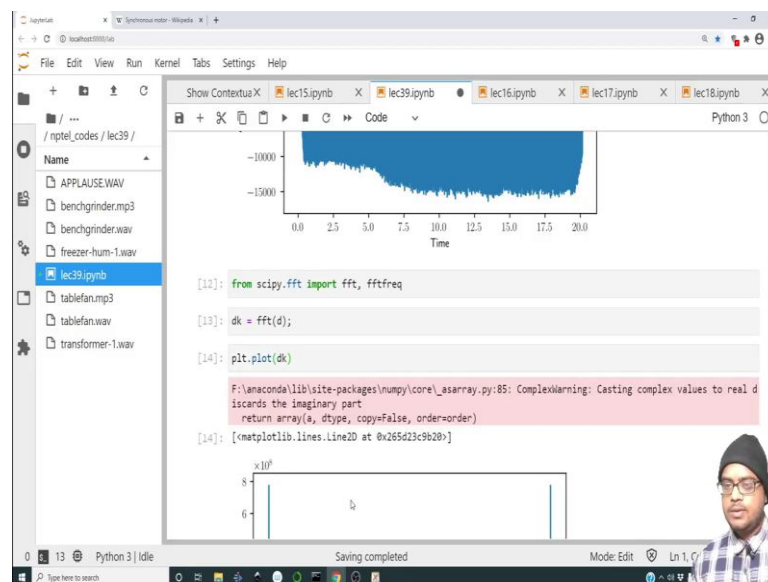
of the continuous Fourier transform. And similarly the inverse transform is nothing, but

$$x(n) = \sum_{k=0}^{N-1} e^{2\pi i \frac{kn}{N}} y(k), \text{ where } i \text{ is imaginary number, } i = \sqrt{-1}.$$

So, what this does it is going to pick out the amplitude of that particular frequency. So, this is an indicator that given a certain frequency n what is the amplitude of that particular wave? Imagine you have $x = \sin(5 \times 2\pi t)$ alright. So, when $n = 5$ it is going to be 1, for all the other ends it is going to be 0, right.

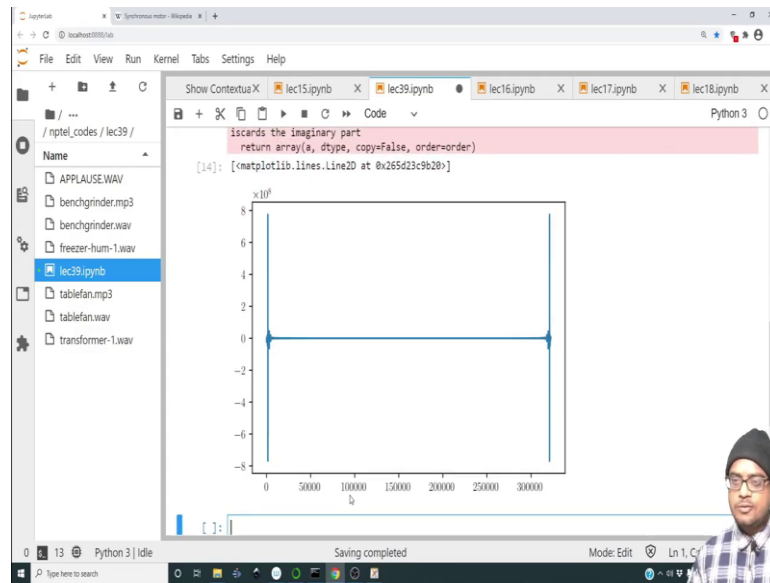
So, this is what the interpretation is going to be it is going to isolate each wave component, but it is not going to go over all the frequencies. It is going to go from $2\pi \times \frac{0}{N}$ to $2\pi \times \frac{N-1}{N}$. So, it is going to go over the discrete frequencies alright. So, we are going to take a Fourier transform of all this. So, for that we have to import the Fourier. So, let us import all those. So, we have to import `fft`.

(Refer Slide Time: 21:37)



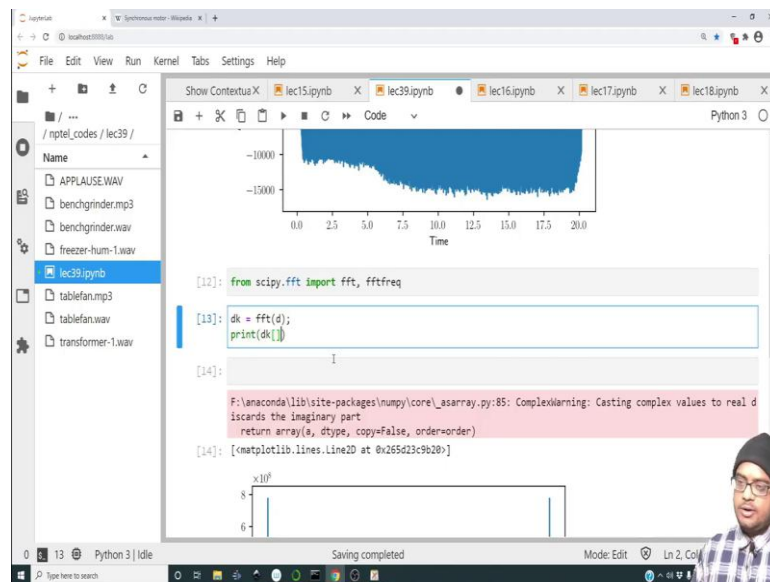
So, from `scipy dot transform` sorry import `fft` comma `fftfreq` and I will tell you what these two functions are. So, after importing this we are going to take a Fourier transform of the data. So, the data is `d`. So, let me call `dk` as the transform data. So, `dk` will be `fft` of `d` alright.

(Refer Slide Time: 22:28)



Let us see what how dk looks like. So, plt.plot dk it is look it looks very weird ok, but there is a certain way in which we must plot this, but before even plotting let me show you what dk has.

(Refer Slide Time: 22:41)



So, let me first of all print dk well. Let me tell you the ordering of what dk is.

(Refer Slide Time: 22:52)

The screenshot shows a Jupyter Notebook on the left and a handwritten slide on the right. The Jupyter Notebook code includes:

```
[12]: from scipy.fft import fft, fftfreq
[15]: dk = fft(d);
      xk = fftfreq(N, 1.0/sam);
[14]: [matplotlib.lines.Line2D at 0x26...
```

The handwritten slide contains the following content:

$$y(k) = \sum_{n=0}^{N-1} e^{-2\pi i k n} x(n) \quad i = \sqrt{-1}$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} e^{2\pi i k n} y(k)$$

$$x = \sin(5 \cdot 2\pi t)$$

$$k = 0, 1, 2, \dots, N/2 - 1, N/2, \dots, 1$$

Below the frequency list, it says "16000 samples per cycle" and shows a calculation: $\frac{1}{16000}$ cycle/sample.

So, once you take a Fourier transform it arranges the terms in a very peculiar fashion it goes from the following sequence of K it goes from 0, 1, 2 all the way to $\frac{N}{2} - 1$ for the positive frequencies. For the negative frequencies, it goes in the reverse order ok, it goes in the reverse order and this particular sequence you can get using the fftfreq function as well. So, once you have dk it is customary to also get the frequency points.

(Refer Slide Time: 23:37)

The screenshot shows a Jupyter Notebook on the left and a handwritten slide on the right. The Jupyter Notebook code includes:

```
[12]: from scipy.fft import fft, fftfreq
[13]: dk = fft(d);
      xk = fftfreq(N, 1.0/sam);
[14]: [matplotlib.lines.Line2D at 0x26...
```

The handwritten slide contains the following content:

$$y(k) = \sum_{n=0}^{N-1} e^{-2\pi i k n} x(n)$$

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} e^{2\pi i k n} y(k)$$

$$x = \sin(5 \cdot 2\pi t)$$

$$k = 0, 1, 2, \dots, N/2 - 1, N/2, \dots, 1$$

Below the frequency list, it says $\Delta t = \frac{1}{16000}$.

So, let me call them as x_k as `fftfreq` and this has the total number of sample points and an additional input which is 1 over the sampling 1 over the sampling frequency, so it is going to be $1.0/\text{sam}$ alright. So, why that $1/\text{sam}$ because once you have this ordering you must realize that you are not sampling over all the frequencies the maximum frequency you are going to sample or rather the smallest time interval that is going to be $1/16000$ once again 16000 that is this per cycle that is a smallest Δt that exists alright.

So, that the inverse of that is going to be the frequency. Actually, let me put it this way. So, you have 16000 samples per second so per cycle. So, 1 cycle consists of $1/16000$ cycles per sample ok. So, that is the smallest unit you have and that is what you need to give to the effective frequency to tell that this is the unit, step unit of the frequency ok. So, this will give you x_k so far so good.

Apart from this if you look at the formula, over here if you set y ; if you set $k = 0$ then it is simply the following sum. If you set $k = 0$ and you have y is 0 , $\sum_{n=0}^{N-1} e^{0} x(n)$ and it is simply the sum of all the peaks ok. So, let us see whether that is true.

(Refer Slide Time: 26:10)

```
[12]: from scipy.fft import fft, fftfreq
[16]: dk = fft(d);
      xk = fftfreq(N, 1.0/sam);
      q1 = np.sum(dk);
      q2 = dk[0];
      print(q1);
      print(q2);
(-8.940696716308594e-07+2.384185791015625e-07j)
(-23744835-0j)
[14]: F:\anaconda\lib\site-packages\numpy\core\_asarray.py:85: ComplexWarning: Casting complex values to real discards the imaginary part
      return array(a, dtype, copy=False, order=order)
[14]: [matplotlib.lines.Line2D at 0x265d23c9b20b]
```

So, we are going to plot the two quantities. So, q_1 is sum or `np.sum(dk)` and q_2 is going to be the Fourier transform, but the 0 th mode. So, let us now print q_1 and print q_2 . This is not going to be sum of dk it is going to be sum of d alright.

(Refer Slide Time: 27:07)

The image shows a Jupyter Notebook interface with three main panes. The left pane is a file explorer showing a directory structure with audio files like 'APPLAUSE.WAV', 'benchgrinder.mp3', 'freezer-hum-1.wav', and 'lec39.ipynb'. The center pane contains Python code for FFT analysis:

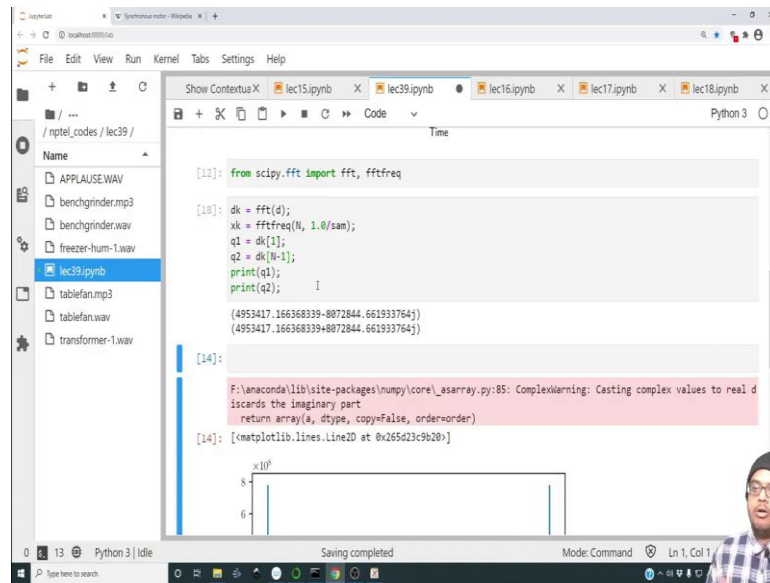
```
[12]: from scipy.fft import fft, fftfreq
[17]: d1 = fft(d);
      xk = fftfreq(N, 1.0/sam);
      q1 = np.sum(d);
      q2 = d1[0];
      print(q1);
      print(q2);
-23744835
(-23744835-0j)
[14]: [cmatplotlib.Lines.Line2D at 0x26...
```

The right pane shows handwritten mathematical notes and a diagram. The notes include the equation $x(n) = \frac{1}{N} \sum_{k=0}^{N-1} e^{j2\pi kn} y(k)$, the definition $x = \sin(5 \cdot 2\pi t)$, and the calculation $y(0) = \sum_{n=0}^{N-1} x(n)$. A diagram shows a discrete-time signal $x(n]$ with a period of 16000 samples per cycle and a sampling rate of $\frac{1}{16000}$ cycle/sample. A small video inset of a person is visible in the bottom right corner of the right pane.

So, look it does check out the sum of all the data points is equal to the 0th or the once you set the $k = 0$, the Fourier transform at the 0th frequency if you will ok that gives you the sum. So, that is true.

So, we have verified this. Now, we can proceed to show another thing that it is symmetric the distribution is symmetric, meaning the first term and the $N - 1$ th term will simply be complex conjugates because these are all the positive frequencies sorry these are all the positive frequencies, these are all the negative frequencies. So, let me show that as well.

(Refer Slide Time: 28:06)



```
[12]: from scipy.fft import fft, fftfreq

[13]: dk = fft(d);
      xk = fftfreq(N, 1.0/sam);
      q1 = dk[1];
      q2 = dk[N-1];
      print(q1);
      print(q2);

(4953417.166368339-8072844.661933764j)
(4953417.166368339+8072844.661933764j)

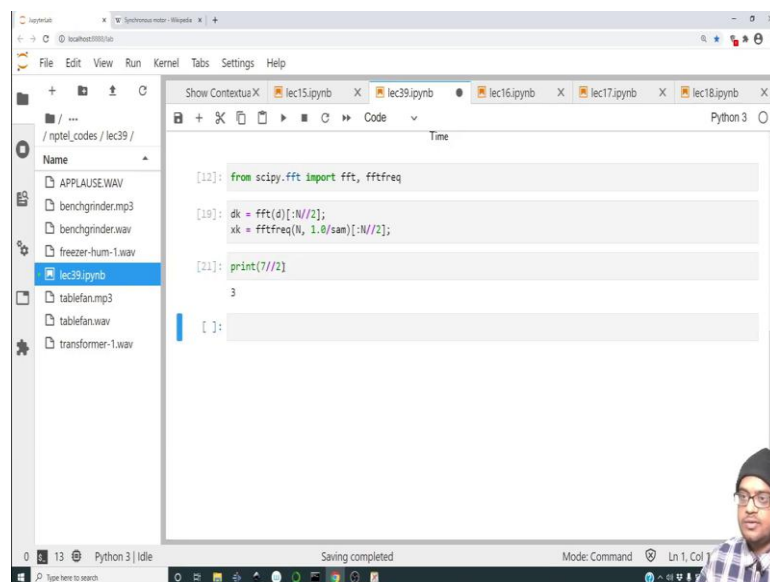
[14]: F:\anaconda\lib\site-packages\numpy\core\_asarray.py:85: ComplexWarning: Casting complex values to real discards the imaginary part
      return array(a, dtype, copy=False, order=order)

[14]: [matplotlib.figure.Figure at 0x265023c9620]
```

The plot shows a magnitude spectrum with two prominent peaks at the lowest and highest frequencies, representing complex conjugate components. The y-axis is labeled $\times 10^8$.

In order, to show that I am going to assign $q1$ to $dk[1]$ and this as $N-1$. So, as you can see they are simply complex conjugate. So, it is good enough if we can simply plot half the values we do not need to take all the values into consideration ok.

(Refer Slide Time: 28:27)



```
[12]: from scipy.fft import fft, fftfreq

[13]: dk = fft(d)[:N//2];
      xk = fftfreq(N, 1.0/sam)[:N//2];

[14]: print(7//2)

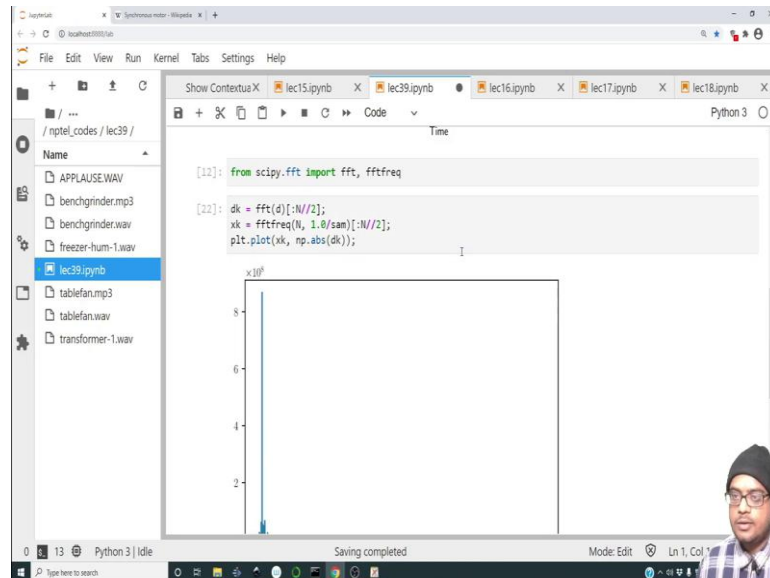
3

[ ]:
```

So, in order to take half the values we can simply take N ceiling 2 alright sorry. So, in case you are wondering what that operator means let me show you, ok the lowest is the floor function ok it is the floor function. So, essentially we are picking out only half the

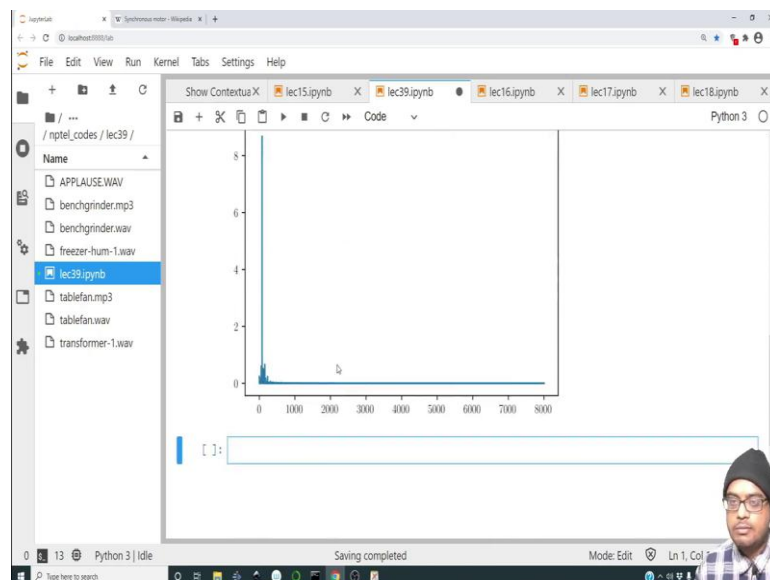
range. So, once we have picked out half the range what do we do? We can now make a plot of it.

(Refer Slide Time: 29:16)



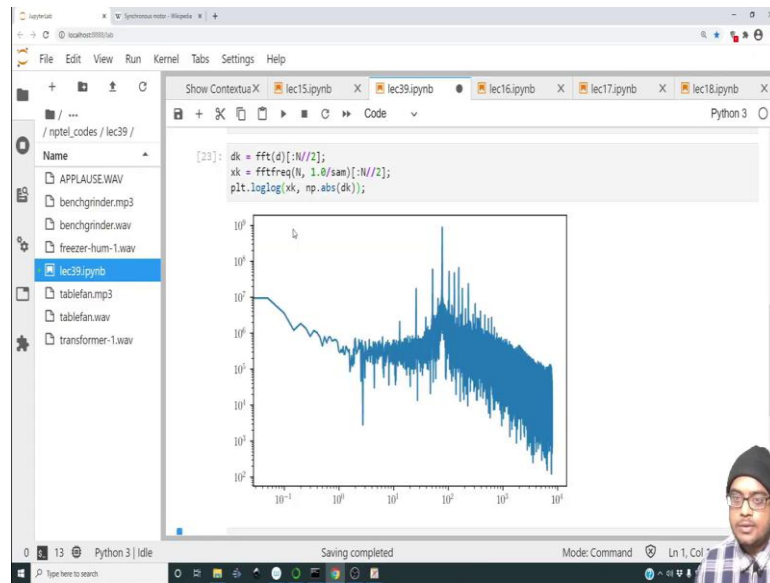
So, let me do that. So, we are going to plot xk and the magnitude of dk . So, $np.abs(dk)$.

(Refer Slide Time: 29:29)



So, let us see ok we have something, things are very crowded, but we do expect things to be crowded near the small numbers. So, that is why we make a log of it.

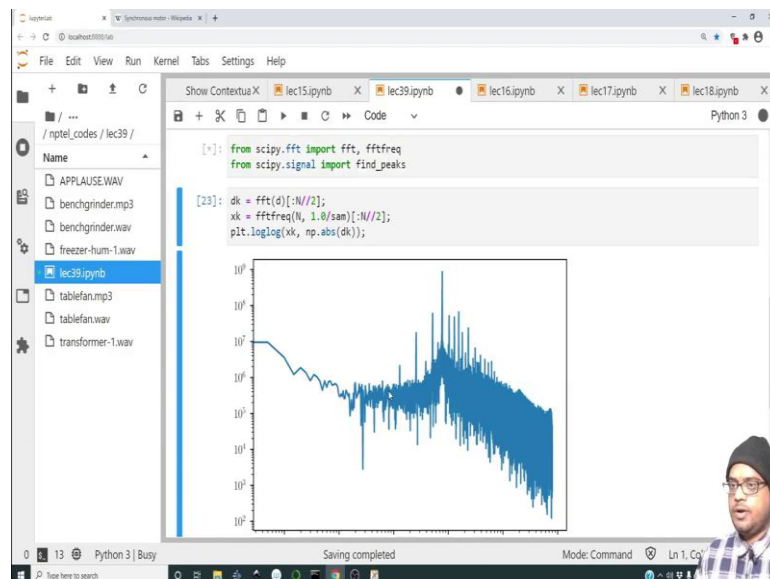
(Refer Slide Time: 29:45)



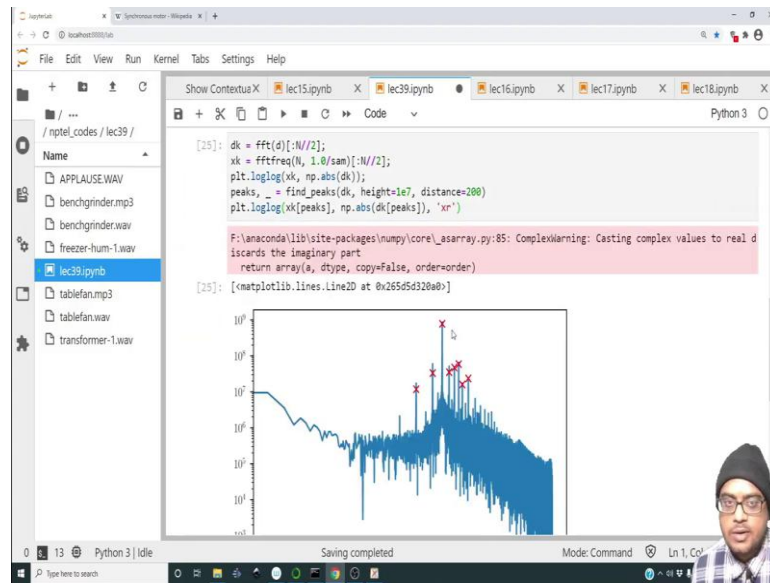
So, instead of plot we do a log log alright. So, this is the log log plot for the fan problem alright. So, now, from this plot you see that at a certain frequency there is a peak in the amplitude and this frequency is something near, well it has different modes this, this, this, this.

So, we would like to isolate those modes alright. So, let us try to isolate those modes, we have done this before as well in order to isolate the modes we are going to use the scipy function peaks find peaks.

(Refer Slide Time: 30:41)

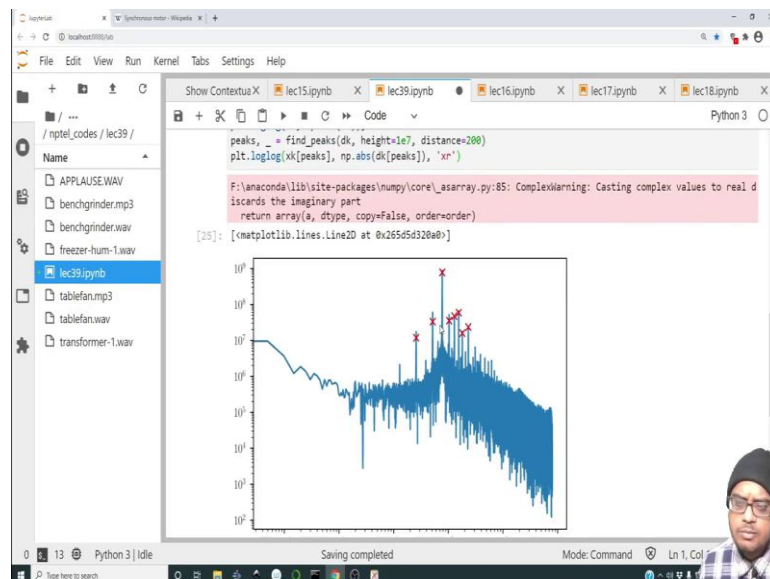


(Refer Slide Time: 30:55)

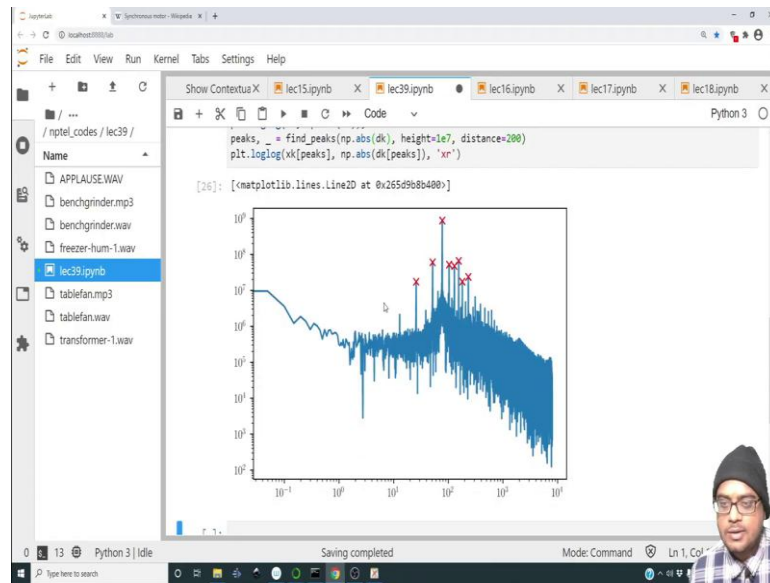


So, let us import that as well. So, from scipy dot signal dot import find peaks alright. Let me run that now we are going to say peaks comma nothing is equal to find peaks of dk height on the peak should be above 10^7 and the distance between the peaks has to be maybe 200 array points alright. So, once we have done this. So, let us do `plt.loglog(xk[peaks], np.abs(dk[peaks]), 'xr')`. So, let us see what we have alright.

(Refer Slide Time: 31:57)

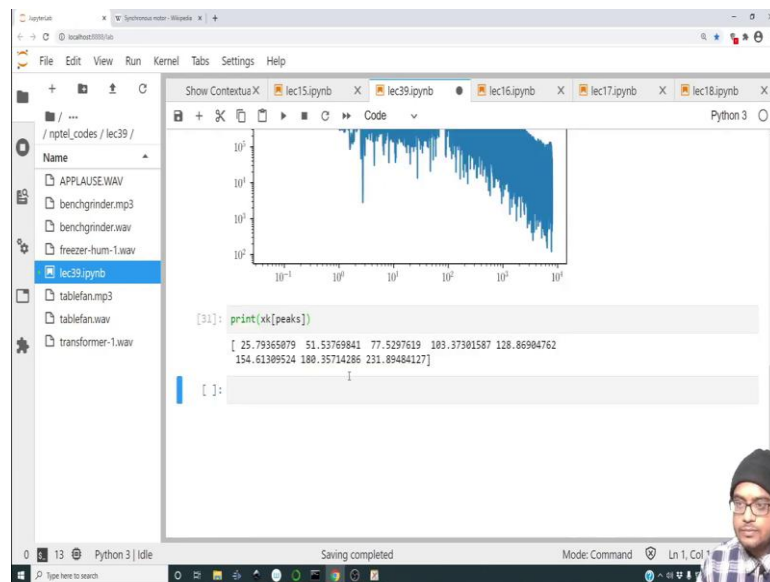


(Refer Slide Time: 32:07)



So, we do have the peaks, but ok I think yeah because find peaks we need to give it np.abs alright great. So, now, we do have the peaks let us see what the values of the peaks are ok. So, let me simply plot the frequencies at which the peaks occur.

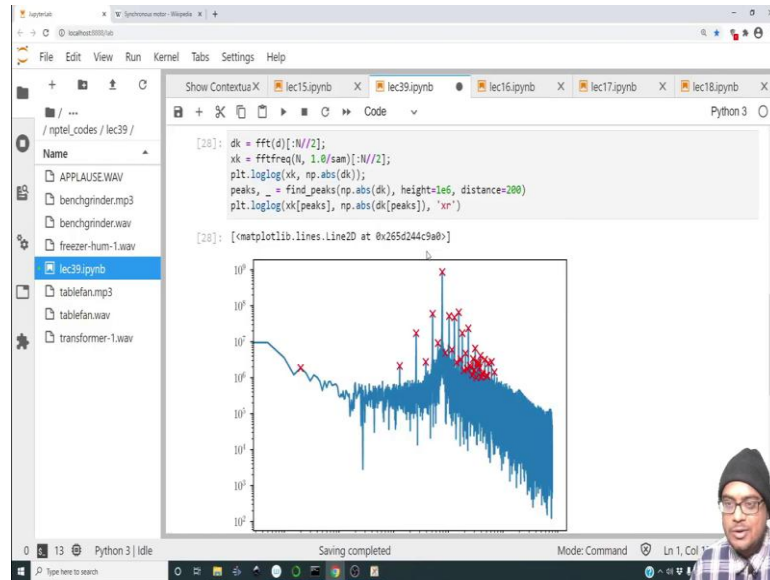
(Refer Slide Time: 32:26)



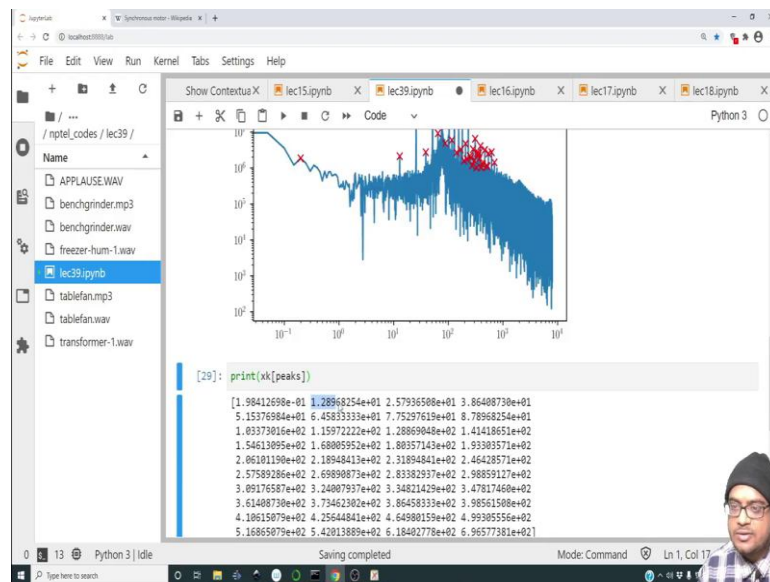
So, xk peaks print xk peaks alright. So, it is 25, 51, 77, 103 and so on. What is the frequency that we had found out and yeah for us for synchronous motor it does come out to be 25 hertz. And that is great I mean you could figure that out using a very simple program I mean it literally is a few lines of code and audio signal with which you could

figure out what that frequency of operation is and you do see that you get multiples of those frequency like 25, 51, 77, 100. So, it is like 25, 50, 75, 100, 125 and so on and I am going to bet that this is something like 12.5.

(Refer Slide Time: 33:35)



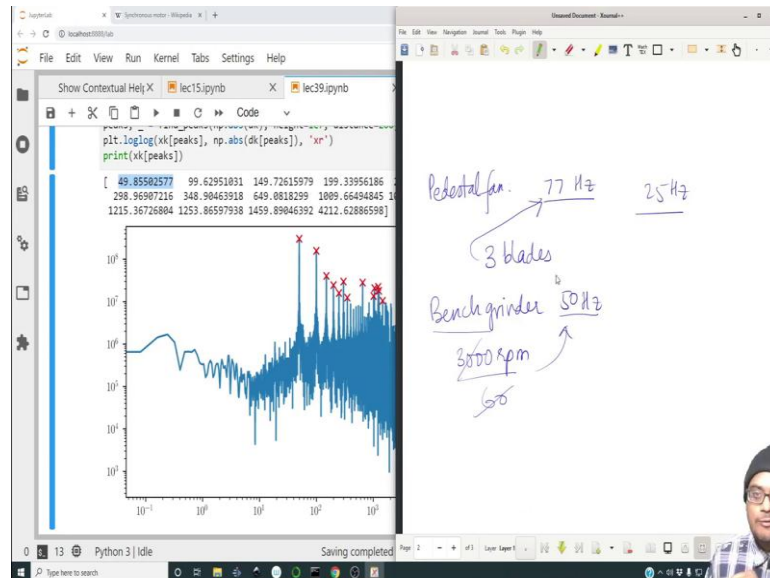
(Refer Slide Time: 33:54)



So, let me just reduce the threshold of the peak finding function let me make it $1e^6$ let me run it. And let me see what this contains ok it contains a bunch of values because it is grabbing this as well. I do not want to grab this well it should be the second peak and it is actually 12.89 it is good enough. So, we are getting all those harmonics it is let me revert

back to $1e^7$. So, the real peak occurs at something like 77. Now, ok we are going to go with the primary peak which is 77 alright. So, what do we have from the audio signal?

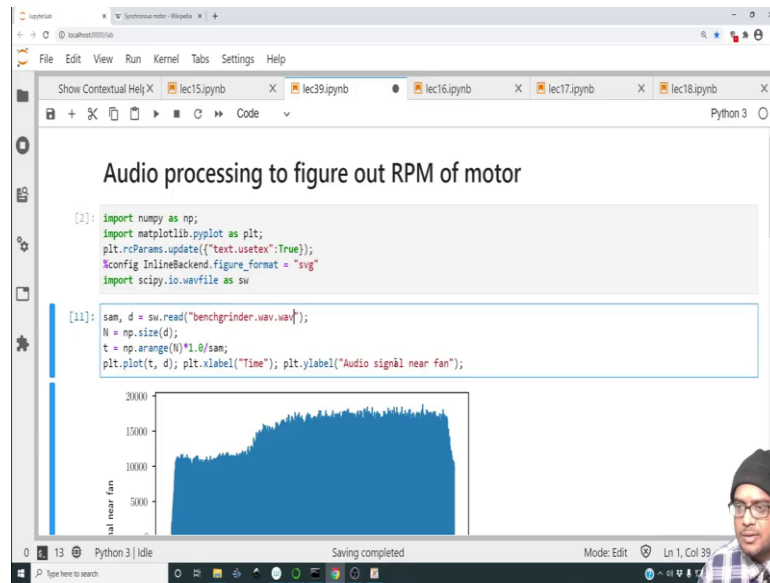
(Refer Slide Time: 34:30)



So, from the audio signal for a pedestal fan we have a peak at 77 hertz alright, that is the peak signal all the other signals are sort of at least a decade smaller in magnitude than that peak signal. But, why 77? I thought it was supposed to be 25 know and the reason is clear this 3 fans these 3 blades in the fan the 3 blades are cutting air thrice in 1 cycle.

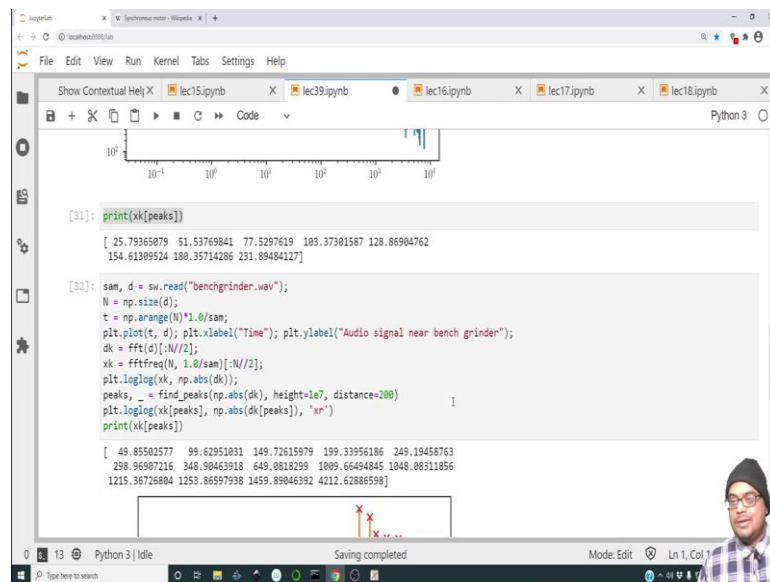
So, we must divide this by 3 to get the true frequency of the fan ok. So, in the cycle you are getting 3 pressure fluctuations because of the presence of 3 blades and hence it comes at something like 75 hertz which corresponds to the synchronous speed of the AC motor. That is quite fantastic I mean think about it you are able to get a very good estimate of the synchronous speed just by the audio signal no fancy electronics nothing just an audio recorder.

(Refer Slide Time: 35:46)



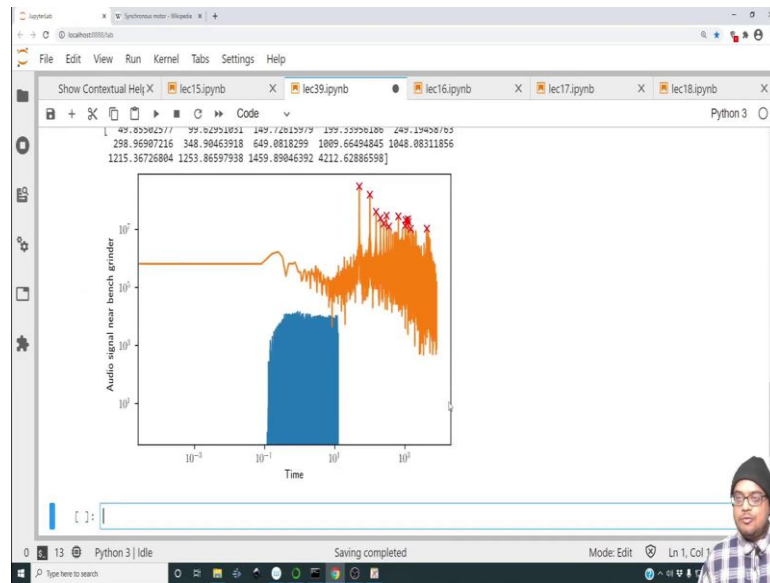
Well not, so first let us change the input file to the bench grinder and let us run the entire thing again for a bench grinder. In fact, let me make a new set of cells to be imported all this.

(Refer Slide Time: 36:12)



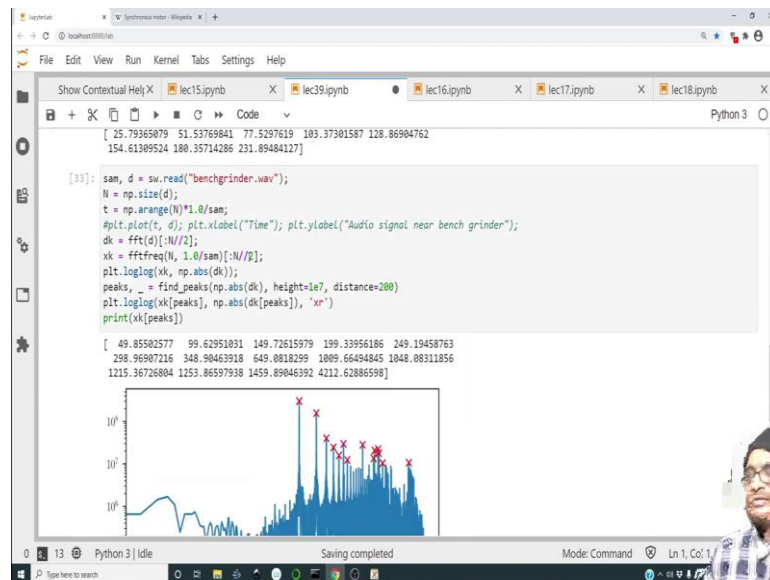
I am just going to change this to benchgrinder and I am going to write this as audio signal near bench grinder.

(Refer Slide Time: 36:39)



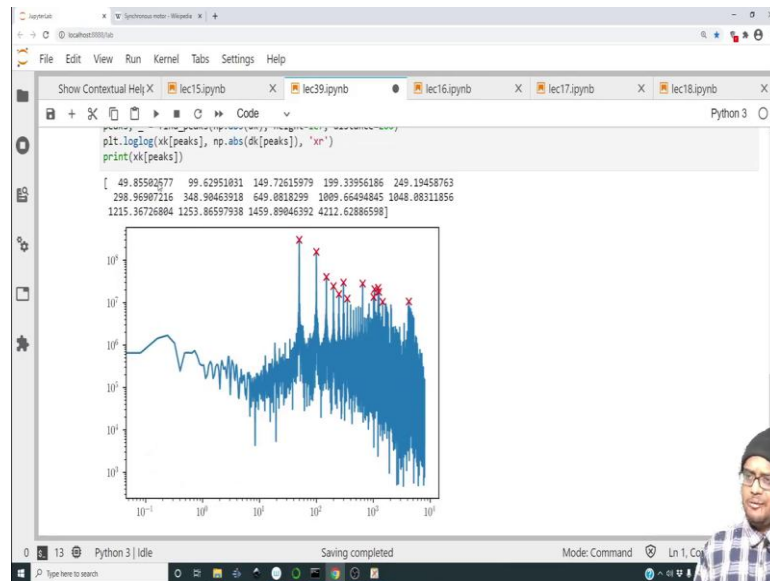
Let me run this cell let me see whether it gives me the same synchronous speed and the answer is no. Well, what is this blue plot let us see we are plotting everything in.

(Refer Slide Time: 36:52)



So, let us suppress this particular plot alright.

(Refer Slide Time: 36:57)



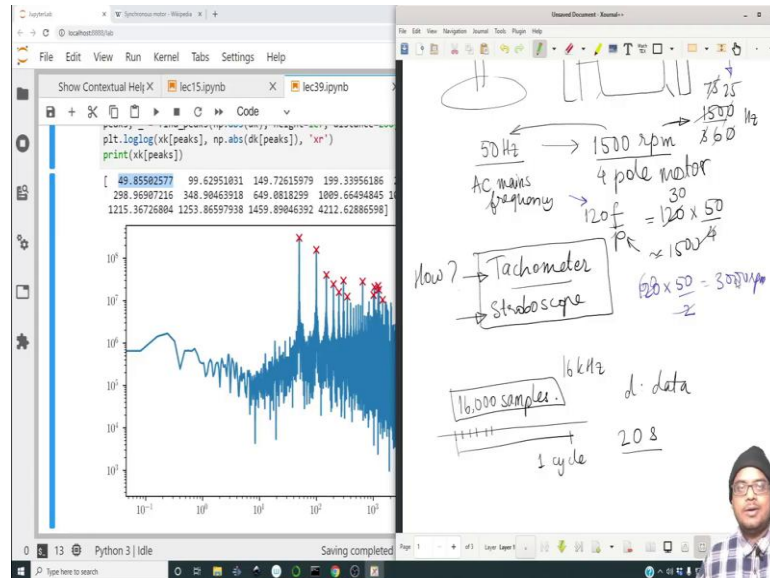
So, for a bench grinder peak frequency appears to be around this which is 50 hertz, well that changes everything. So, for the bench grinder it is around 50 hertz. And you have multiples of 50 hertz that those are the overtones, so to say 50, 100, 150, 200 and so on and that is fine, but 50 hertz.

(Refer Slide Time: 37:32)



And the fact of the matter is I checked the label of the bench grinder and it did say that the bench grinder runs at 3000 RPM and not 15000 RPM, what could be the reason it runs at 3000 RPM? The reason is over here instead of having 4 poles it has 2 poles.

(Refer Slide Time: 37:53)



So, the AC motor of a bench grinder it has 2 poles. So, it is $120 \times \frac{50}{2}$ and it does give you 3000 rpm alright. So, 3000 rpm corresponds to 50 hertz and (Refer Time: 38:10) you do have a good estimate of what the synchronous speed of the bench grinder is. Typically, bench grinders have a higher speed because you are doing a lot of grinding in order to have good material removal rate you need to have a high rpm in the grinder.

So, with the help of this simple experiment you are able to figure out all the different frequencies that are at play. So, with this I am going to conclude this experiment all the sound files will be up for upload and you can try this piece of sample code for many roto-dynamic machines in and around your place. So, with this I say goodbye I will see you again next time bye.