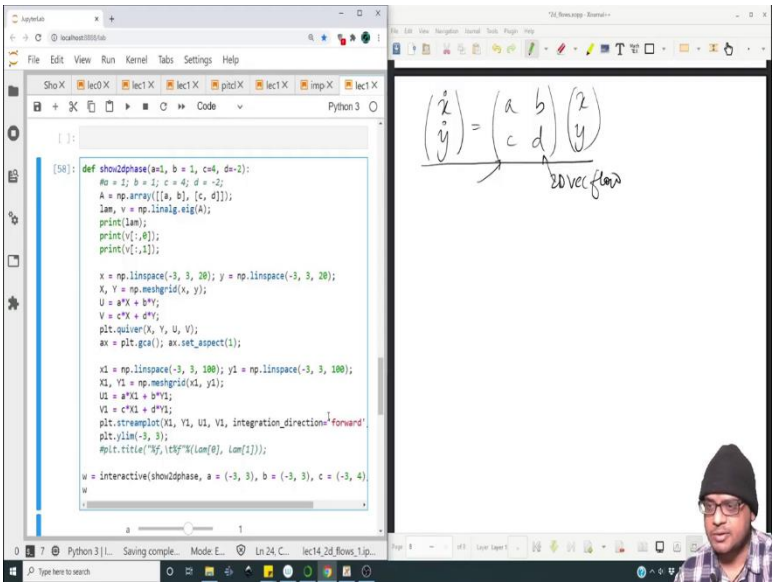
Tools in Scientific Computing
Prof. Aditya Bandopadhyay
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

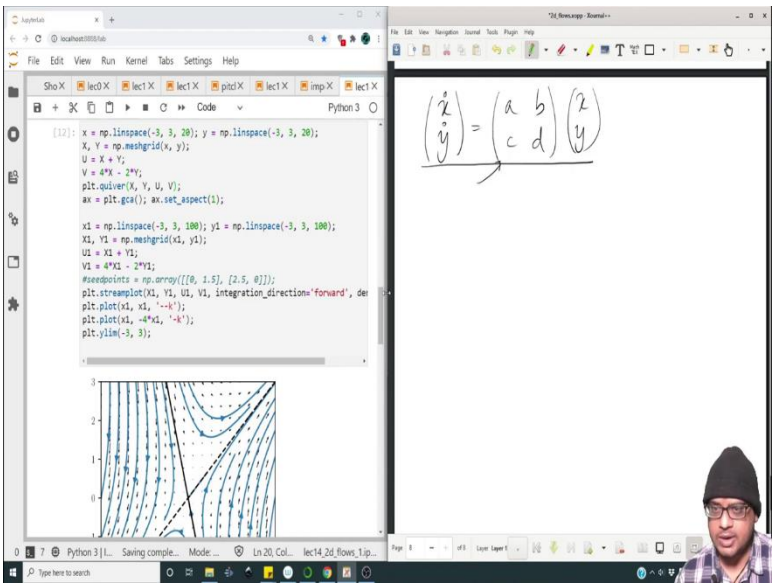
Lecture – 15
2D flows – Trajectories: spirals, star, and degeneracy

(Refer Slide Time: 00:27)



Hello everyone, in this lecture, we are going to continue off where we dropped last time. So, in the last class, we had solved this particular equation.

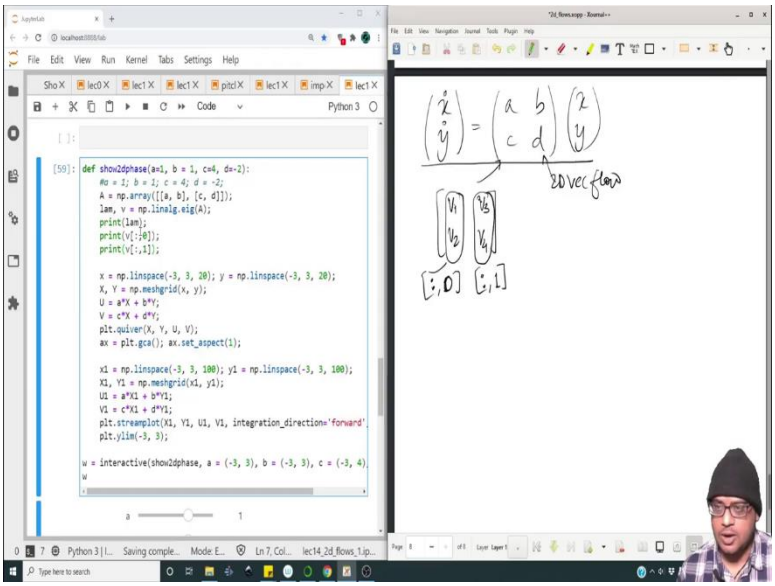
(Refer Slide Time: 00:36)



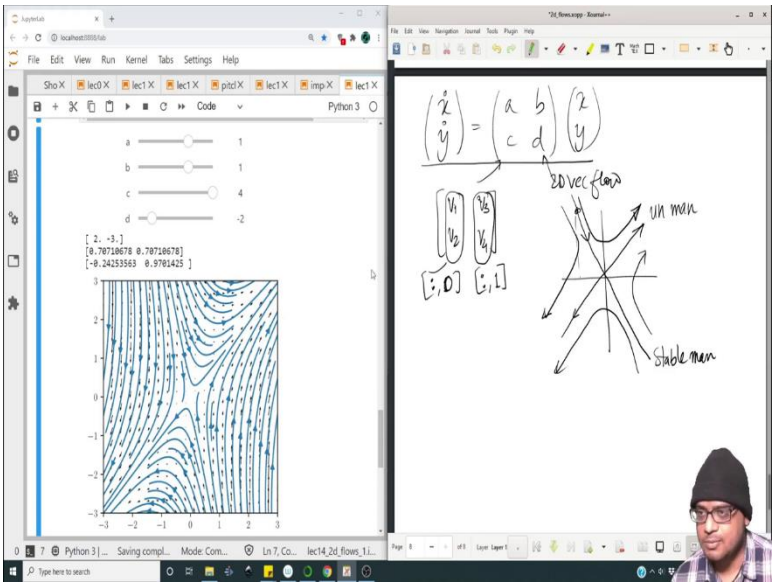
And in this particular class, we are going to start off with the generic form; by this generic form I mean, the matrix is not hard coded, but a b c d are four values, ok. So, the equations are $U = aX + bY$ and $V = cX + dY$. We are going to make a quiver plot of this particular 2 D vector flow. And after that we are going to plot the stream plot of it, ok.

And because we are going to pass a b c d as four parameters; I have gone ahead and made a function which wraps everything inside this function and I am going to make an interactive widget, which takes as an input a, b, c and d, ok. There is nothing very difficult; I have just modified the previous program and I have removed the eigenvectors, so that when things become complex, we have something to work with. So, let me run this cell, ok.

(Refer Slide Time: 01:33)

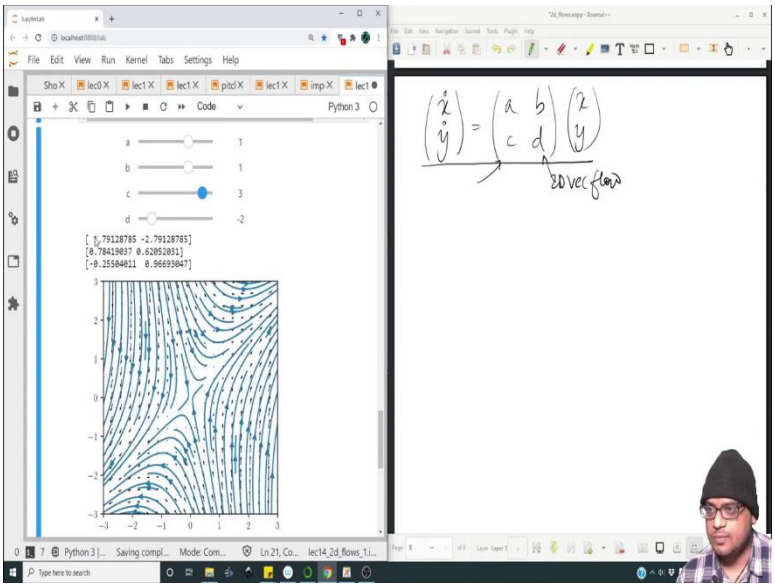


(Refer Slide Time: 01:44)



So, this is the plot that we have. And just by looking at this plot we can see that, there is one distinct eigen direction like this and another distinct eigen direction like this, ok. And let us now change the parameters and see what happen.

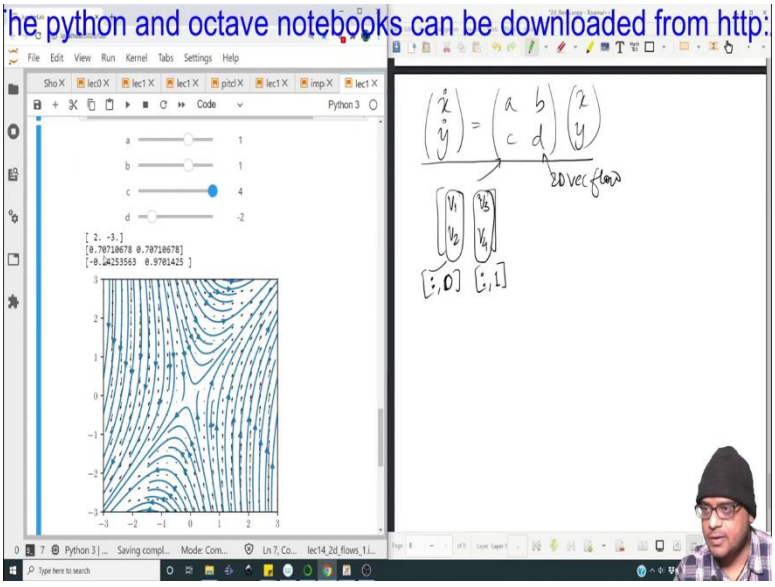
(Refer Slide Time: 02:04)



So, when c is changed. So, I have printed out the eigen values over here, ok. So, let me just go over here, ok. So, before proceeding, the eigen values of a matrix A can be found out by using np.linalg.eig; this is the function that returns the eigen values λ and the eigen vectors v.

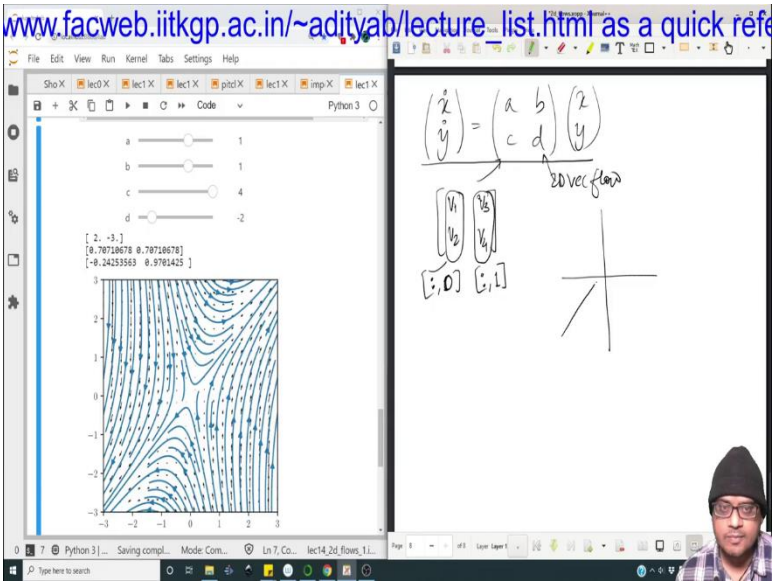
And this nature of the eigen vectors is such that, eigenvectors return a 2 D array and V_1, V_2. So, all the rows of one column correspond to one eigenvector. So, this is one eigenvector. And V_3, V_4; so this is another eigenvector. So, all the rows of the second column this, it represents the second eigenvector. So, this is all rows of the first column and this is all rows of the second column and the indexing starts from 0 that should be obvious.

(Refer Slide Time: 03:17)



So, this is how we can extract the eigenvectors as well. So, with the help of the eigen value and eigenvector, we can clearly see where the flows are going, ok. So, let me just go back to the original case, ok.

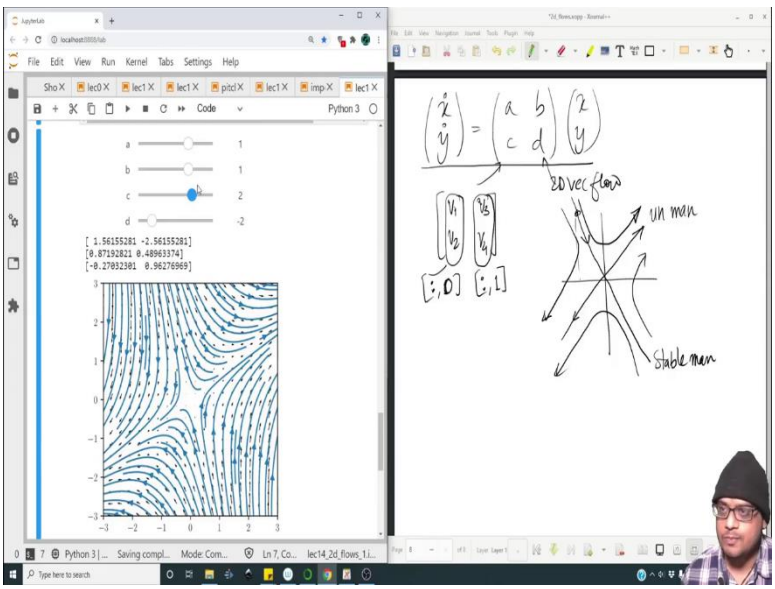
(Refer Slide Time: 03:35)



So, over here the eigen value 2 corresponds to the direction [0.7, 0.7]; that is essentially [1, 1]. And the eigenvector -3, rather the eigenvalue -3 corresponds to the eigenvector – [0.24, 0.97]. So, it is like, if we draw this direction [1, 1], it is this and this is corresponding to an eigen value of 2.

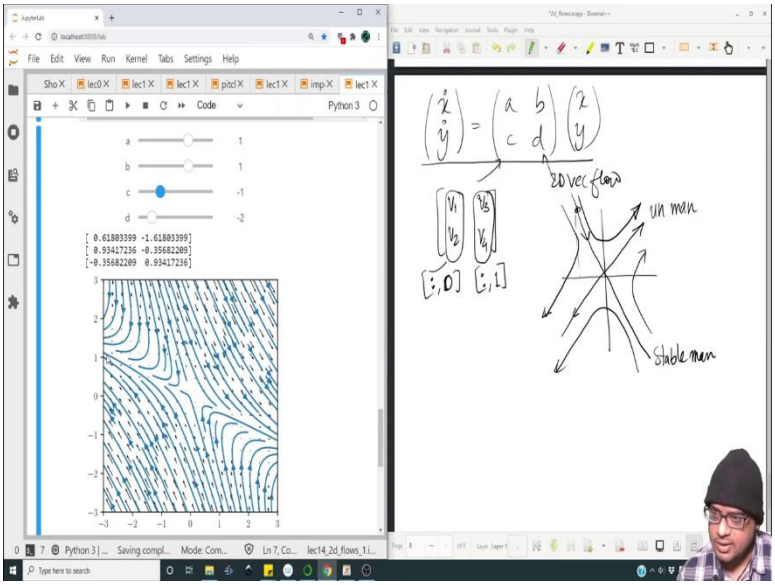
So, it is unstable, this is the unstable manifold. While this particular thing, so minus sorry y is 0.9 and x is -2; so something quite steep. So, this corresponds to the stable manifold; because it corresponds to a negative eigen value. So, as t tends to infinity, the trajectories are attracted towards the unstable manifold like this, ok. This is something which we had discussed in the previous class.

(Refer Slide Time: 04:41)



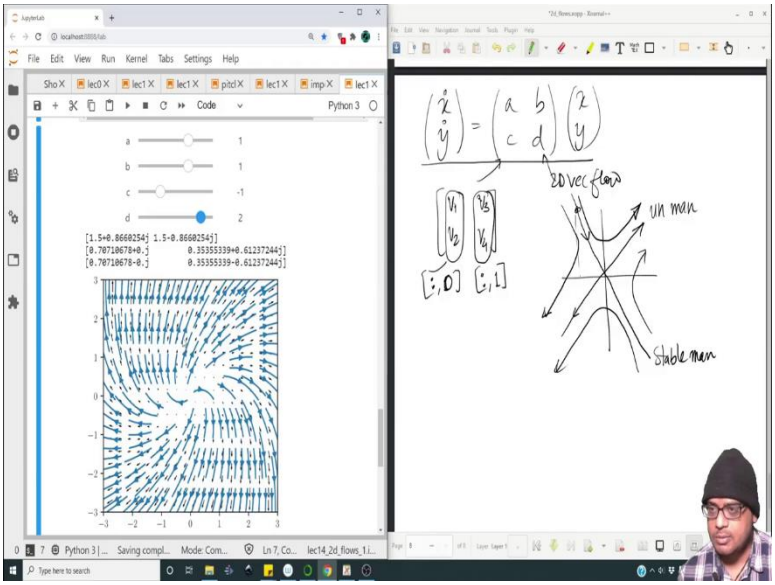
Now, when we change the values of a and a, b, c, d; the eigen values change, so do the eigenvectors and correspondingly the flow also changes shape, ok.

(Refer Slide Time: 04:48)



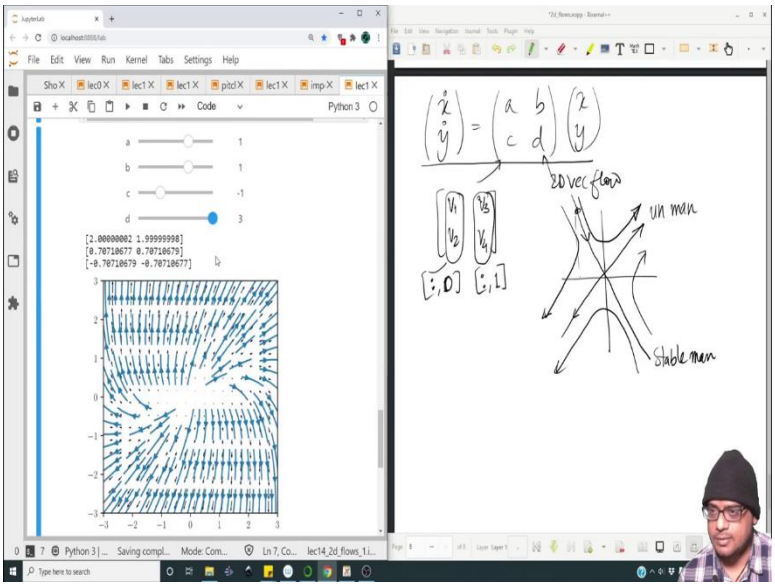
There is nothing quite difficult about it, ok. For example, in this you can clearly make out that, this is one eigen direction and this is the other eigen direction. And this eigen direction is stable, while this is unstable, ok. But what happens when things become complex?

(Refer Slide Time: 05:14)



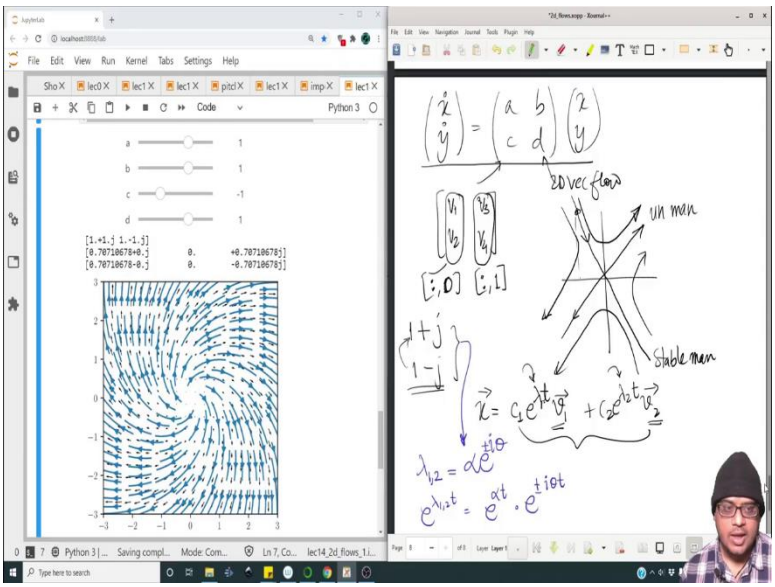
So, let me increase the value of d alright, something weird happens. So, we no longer have a stable or an unstable thing going on; things tend to look like a spiral, ok.

(Refer Slide Time: 05:30)



So, for d=3, we do have real eigenvalues 2 and 2; but they are degenerate eigenvalues ok, this is a special case.

(Refer Slide Time: 05:45)

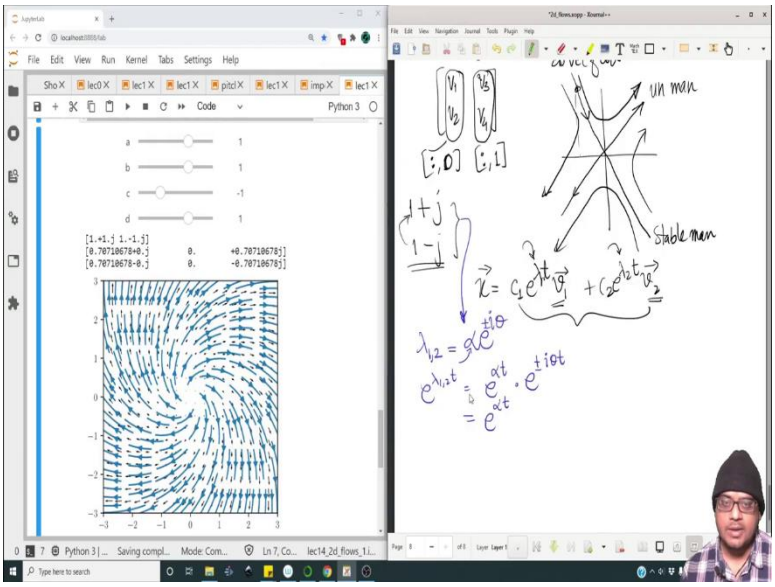


Let us look at this particular case. So, the eigen values are 1+j and 1-j. In fact, if you are doing it for such kinds of matrices; you are finding out the eigen values of such kinds of matrices, you are bound to end up with complex conjugates, ok. So, this is the complex conjugate of this.

So, we know that the solution x it grows as $c_1 e^{-\lambda t}$ t in the direction of v_1 and c_2 and this should be $\lambda_1 e^{-\lambda_2 t} v_2$. We know that is just a linear combination of this and once we diagonalize it. So, finding out the eigenvalue is tantamount to diagonalizing it. So, the solution grows along this eigenvector with a rate of λ_1 and the solution grows at a rate of λ_2 along the eigenvector λ_2.

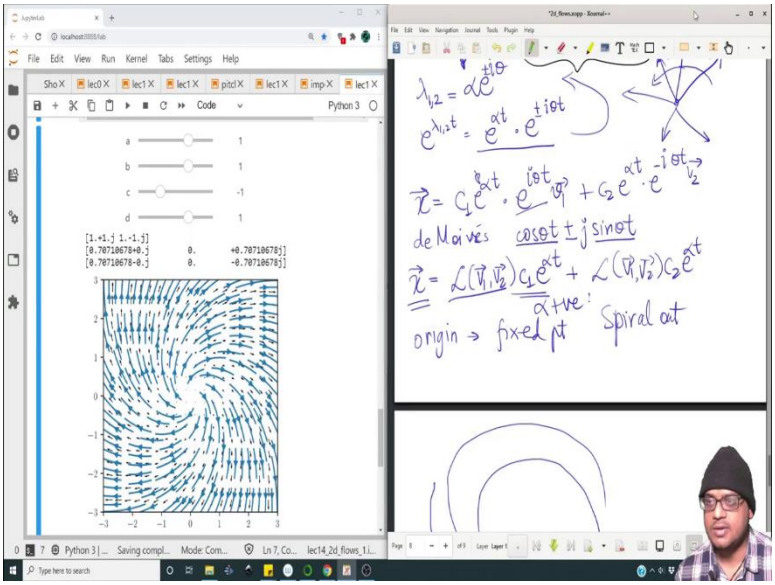
But what is going on when the λ they become complex? So, $\lambda_{1,2}$ can be written as $\alpha e^{i\theta}$ or rather $\alpha e^{\pm i\theta}$. So, this is like the polar representation of a complex number, where it is essentially representing this in a polar form. So, $e^{-\lambda_{1,2} t}$ is simply going to be $e^{\alpha t} e^{\pm i\theta t}$. And the meaning of this is some, so α is obviously real.

(Refer Slide Time: 07:22)



So, this is representing that in time, how does the magnitude of e to the power is reduced? Actually, it will be clearer, if we substitute this over here.

(Refer Slide Time: 07:38)

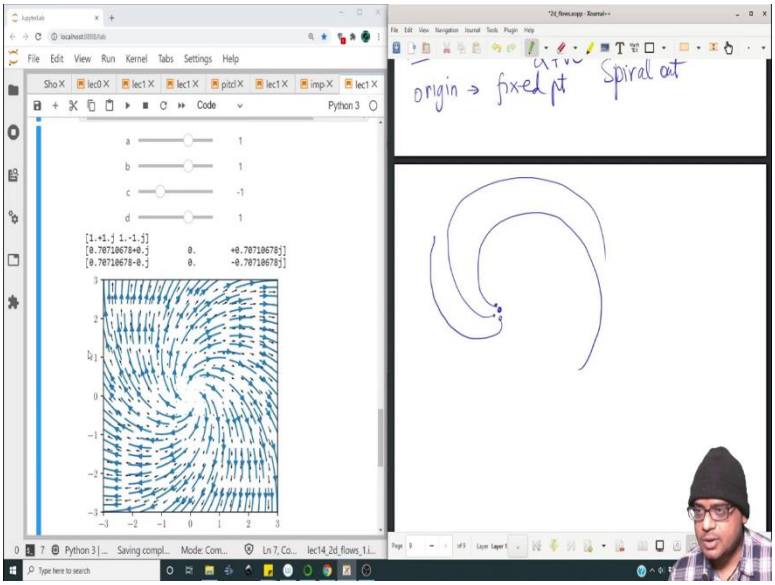


So, we have x is equal to $c_1 e^{\alpha t} e^{i\theta t} v_1$ and this will be $+c_2 e^{\alpha t} e^{-i\theta t} v_2$. And by using the de Moivres theorem, we can sort of write this as $\cos\theta t + j \sin\theta t$ and this thing as $-j \sin\theta t$. So, effectively x becomes some sort of a linear combination of V_1, V_2; some linear combination of V_1, V_2 times $c_1 e^{\alpha t}$ and some other linear combination of V_1, V_2 times $c_2 e^{\alpha t}$.

So, essentially, we have the amplitudes of this x reducing in time or increasing in time. So, if α is positive, the magnitude of these terms they grow and this linear combination obviously involves $\cos\theta t$ and $\sin\theta t$. As a result, we have a rotationality over time; meaning if we have a vector like this and if you multiply it by $\cos\omega t$, this vector will start changing its direction, ok.

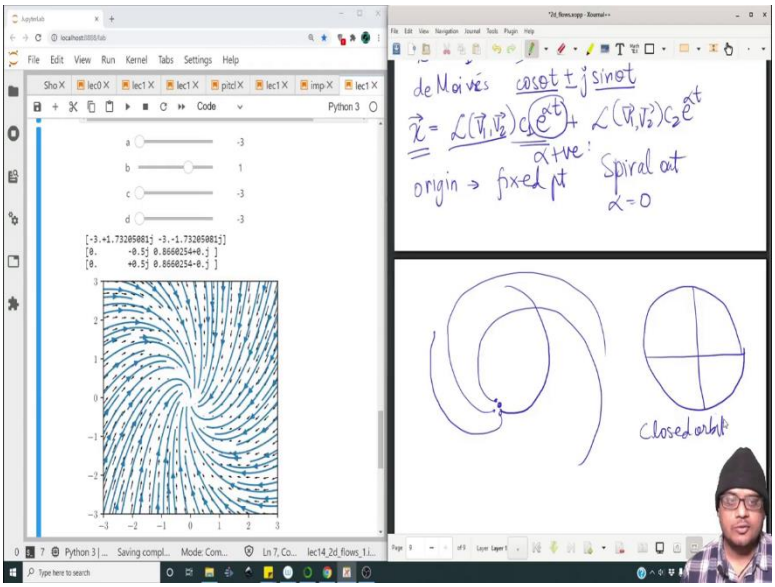
But in time multiplication by $\cos\theta t$ and $\sin\theta t$, it starts making this vector rotate. So, that is why, so look at this origin. So, origin was obviously, the fixed point. So, origin is always a fixed point for a linear system and because of this eigenvalue, α over here is positive. So, the magnitude of this grows and the thing begins to rotate. If we look at a trajectory, it is spiraling out ok; all the trajectories seem to be spiraling out.

(Refer Slide Time: 09:59)



So, it starts near the origin and it is it will start rotating and it is magnitude will grow, it is something like this ok. Now, if that eigenvector becomes negative, rather not the eigenvector; but the eigenvalue corresponding to this α ok, this real part if it becomes negative. Let us see for what combination we can have that, ok.

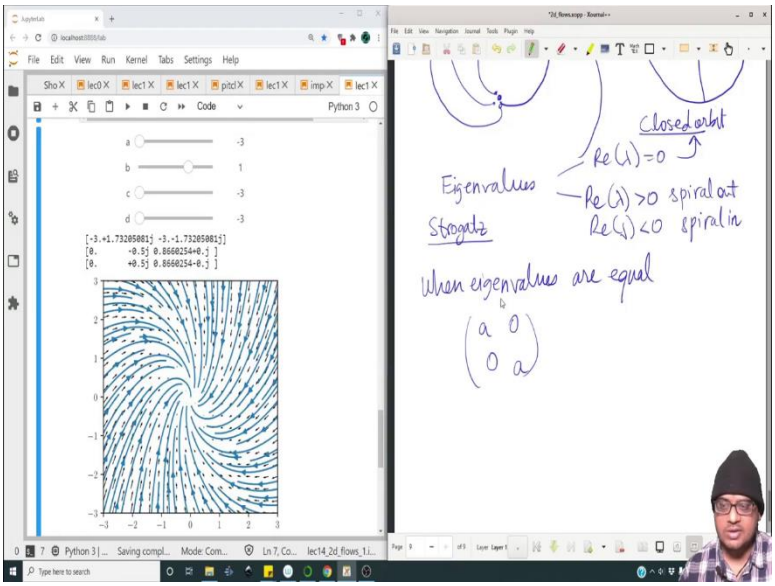
(Refer Slide Time: 10:30)



So, for this particular combination, we do have a negative value of. So, we have a negative real value. So, it means that, in time the amplitude decays. And so, it is spiraling inwards, but the amplitude is decaying. In fact, if the eigen values are purely complex, then this particular flow.

So, there is no change in amplitude, because of this αt; if α becomes 0, there is no change in amplitude. And as you can imagine, if you have two vectors which are rotating, it will simply go off as a circle or a closed orbit; it is not a circle, it will be a closed orbit.

(Refer Slide Time: 11:39)

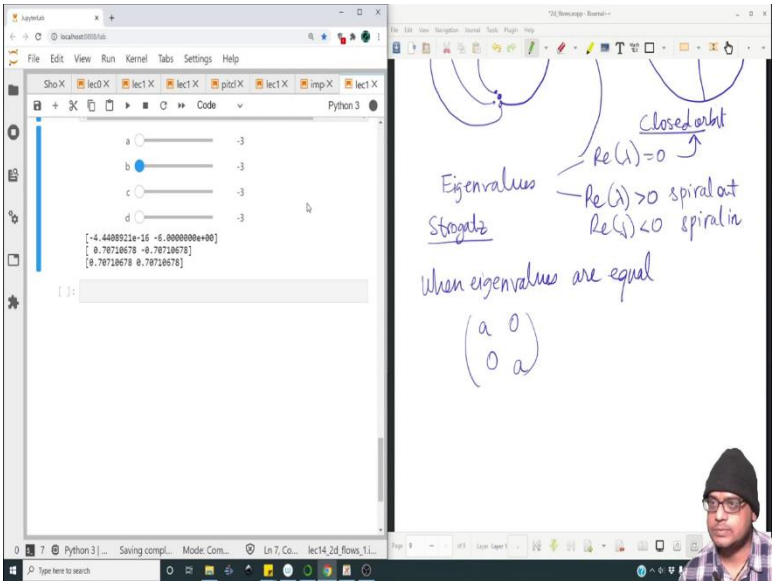


So, the conclusion is, if the eigenvalues a have zero real part; that is real part of λ is 0, then it boils down to a closed orbit, where the fixed point is still the origin. If the real part of λ is greater than 0, then it is a spiral out; if the real part is less than 0, it is a spiral in, ok.

I am not going to go into the elaborate theory of it; but you can look up the book by Strogatz, ok. So, the reference to the book will be there in the description. So, I am showing you how we can get it done using a computer.

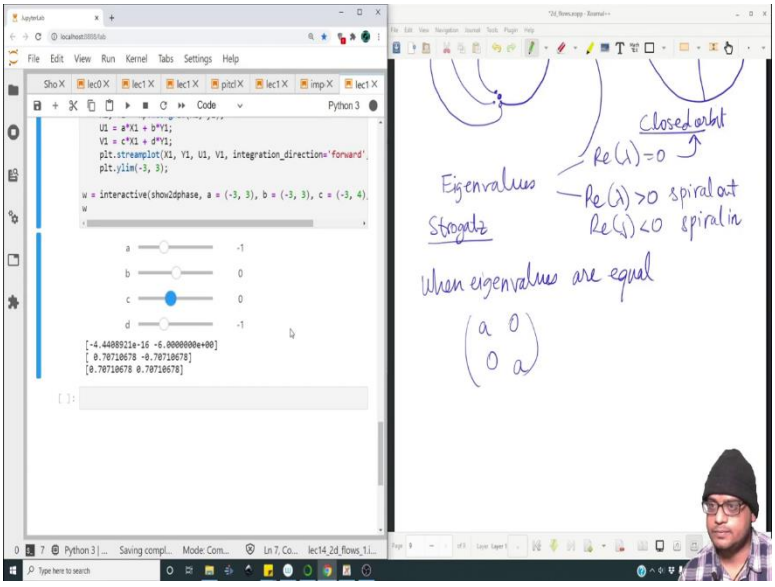
Now, the other case that is quite useful is when the two eigenvalues become equal, ok. When eigenvalues are equal; which mean that and the system will look something like, something like this, ok. This has equal eigen values; then what happens, let us see.

(Refer Slide Time: 13:01)



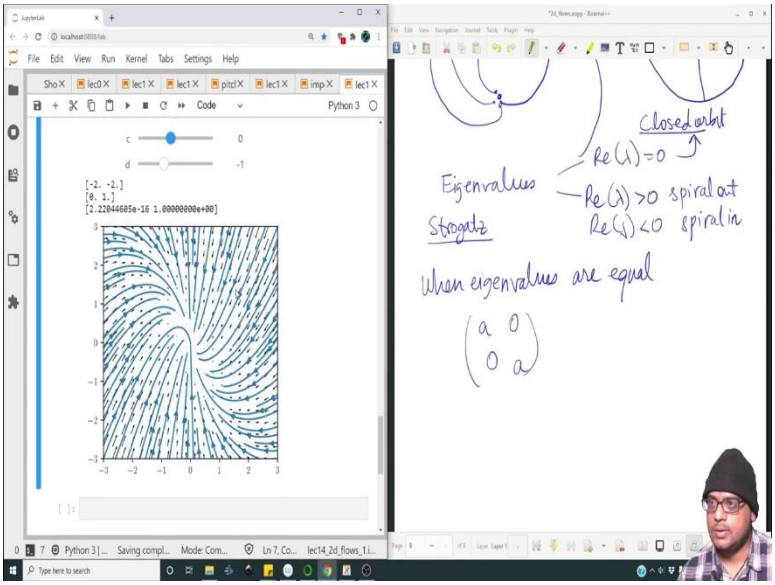
Let us change the slider and see. So, when a and d become equal and the other two things become 0, ok.

(Refer Slide Time: 13:10)

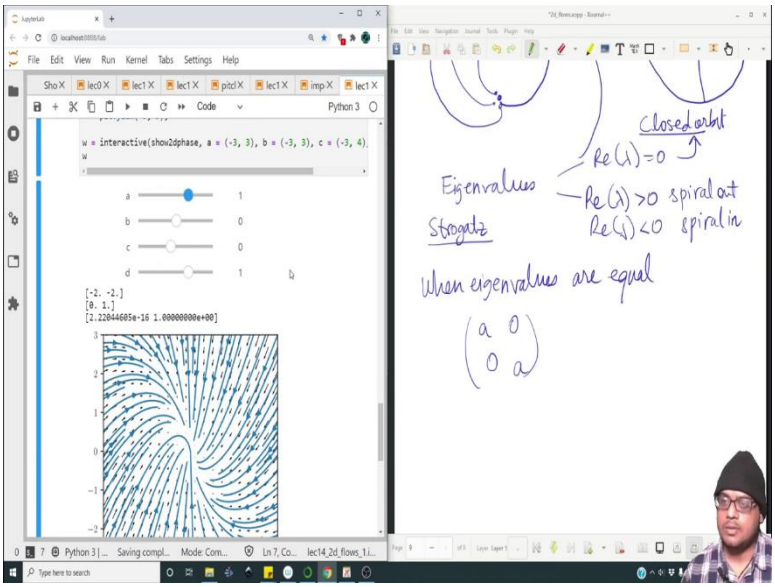


It is not showing any plot, oops it is up to be 0, ok.

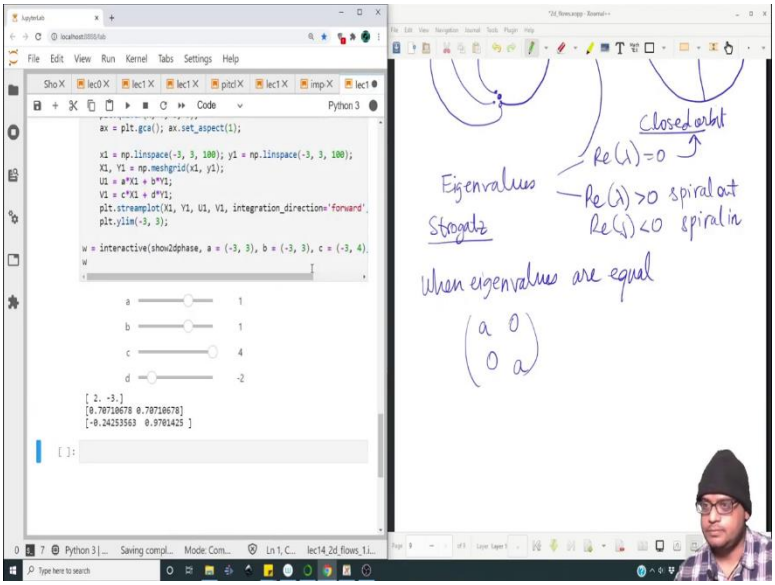
(Refer Slide Time: 13:21)



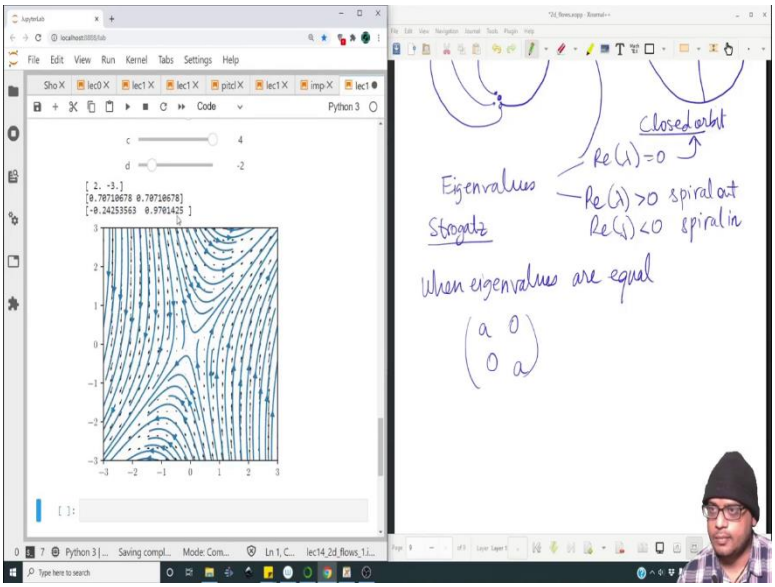
(Refer Slide Time: 13:32)



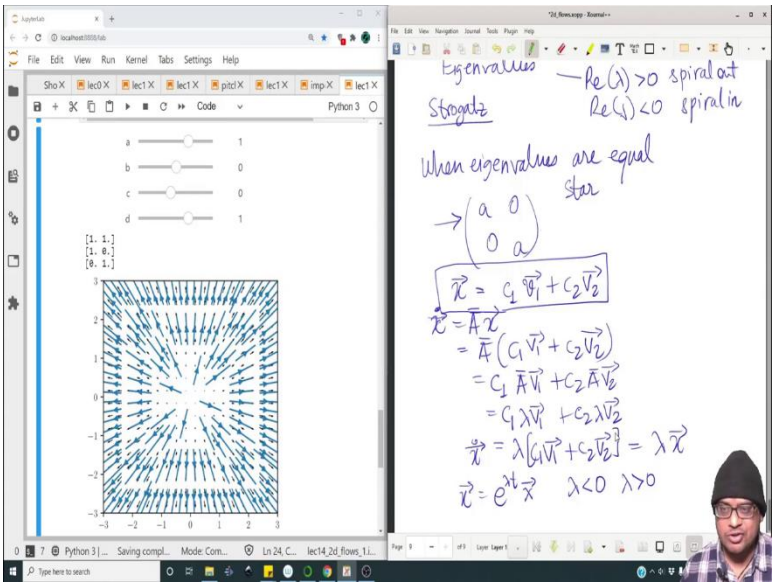
(Refer Slide Time: 13:43)



(Refer Slide Time: 13:44)



(Refer Slide Time: 13:47)



So, when the two eigenvalues are equal, in this particular case it is equal to 1; we see that the center is sort of emanating trajectories all over the space, ok. So, this particular case is called as a star. So, the origin is still a fixed point; but in this case, all the trajectories originating from the origin are repelled at equal rates away from it.

This is distinct to this particular case, where the manifolds where they are attracting or repelling ones. And everywhere else, the trajectory depending on how it is a linear combination of the two eigenvectors; it its trajectory was dictated, but in this particular case, all the trajectories are equivalent and the reason is quite simple.

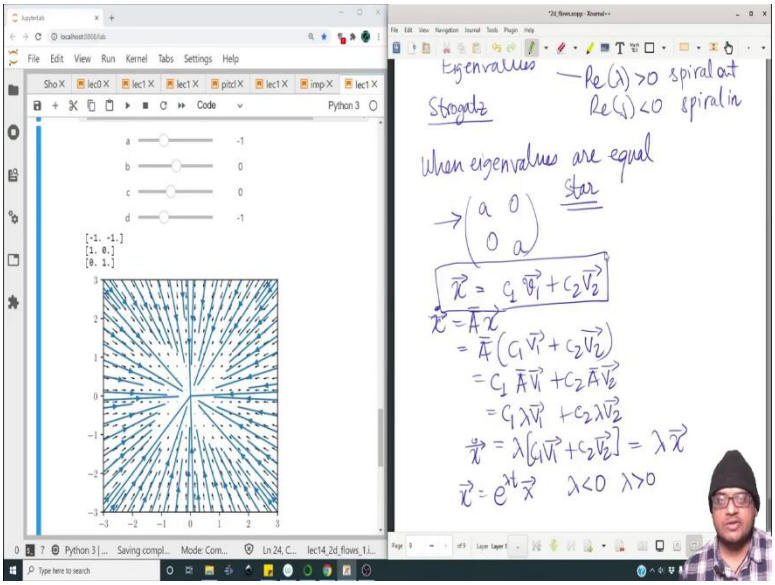
So, if we have two eigen values, then any vector can be written as a linear combination $c_1v_1 + c_2v_2$, so where v_1 and v_2 are two eigenvectors for this system. And when we substitute this form inside $\dot{x} = Ax$; what do we have? It is equal to $A(\ c_1v_1 + \ c_2v_2)$ and this becomes equal to $c_1Av_1 + c_2Av_2$, but $Av_1 = \lambda v_1$ and $Av_2 = \lambda v_2$.

And both are equal because we are considering the case, where the eigen values are equal, ok. So, in that case, this becomes $\lambda(c_1v_1 + \ c_2v_2)$; but $c_1v_1 + \ c_2v_2$ is simply this vector *x*. So, this becomes λx. So, this means, $\dot{x} = \lambda x$ and so the solution for *x* is $e^{\lambda t}x$.

And this means that, all the vectors are exponentially growing or decaying depending on the sign of λ; whether $\lambda < 0$, $\lambda > 0$, the vectors in that entire phase space will either grow

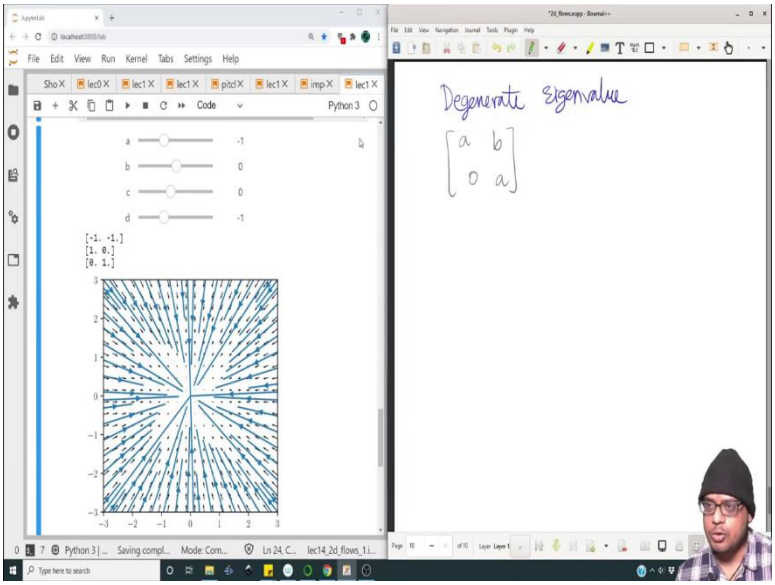
exponentially or decay exponentially or the trajectory will blow in exponentially fast or decay exponentially fast. In this case, the eigenvectors are both positive.

(Refer Slide Time: 16:28)



And let me go to this particular case, when they are both negative; we should have a attracting star, all the trajectories should converge to the origin as we see over here, ok. So, this is the condition for having a star. And all these cases will be quite useful when we analyze non-linear systems; like I have already said, understanding linear systems this is like the first step towards the analysis of a non-linear system.

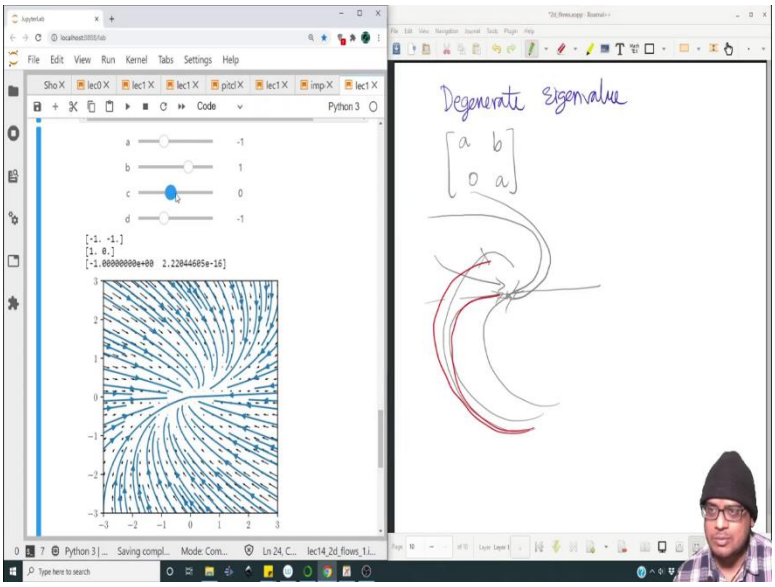
(Refer Slide Time: 17:05)



The last interesting part is that of a degenerate eigen value. So, a degenerate eigen value means that, you have only one eigen value or it is a part of a system which looks something like this, ok. So, we should try to find out the two eigen values, you will see that nothing really works out.

And this is the case, where there is one eigen value; but something quite bizarre happens. And if you actually try to find out the solution for this; you will have linear terms appearing in your system, rather than exponential terms, ok. So, we are not going to go into the theory; but I am going to show you how it looks. So, in order to mimic the system; c is bound to 0, b is something positive, ok.

(Refer Slide Time: 17:56)

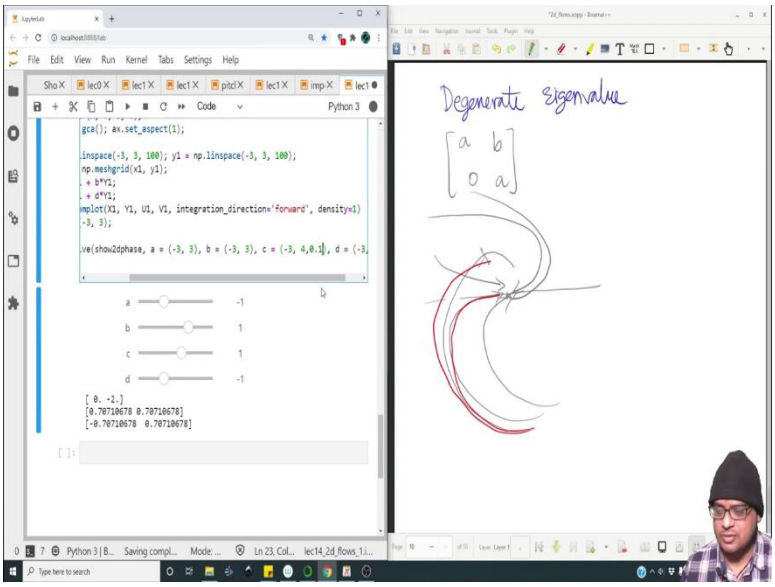


So, in this kind of system, we have one eigen direction, which is [1, 0], which is the x axis ok; the x axis is the eigen direction. The other direction has also the x-axis; because we cannot span the entire space with just one eigen value ok, the eigenvalue are both equal and equal to -1.

So, the x-axis is sort of attracting all trajectories towards itself. So, this is the x-axis and it is attracting trajectories towards the origin; but it is not actually going like this, no it is not, it is actually overshooting from this and converging to the origin, ok. The trajectories are doing this, on the other hand they are doing something like this; it is almost like it is trying to form a spiral, but it is not able to form the spiral.

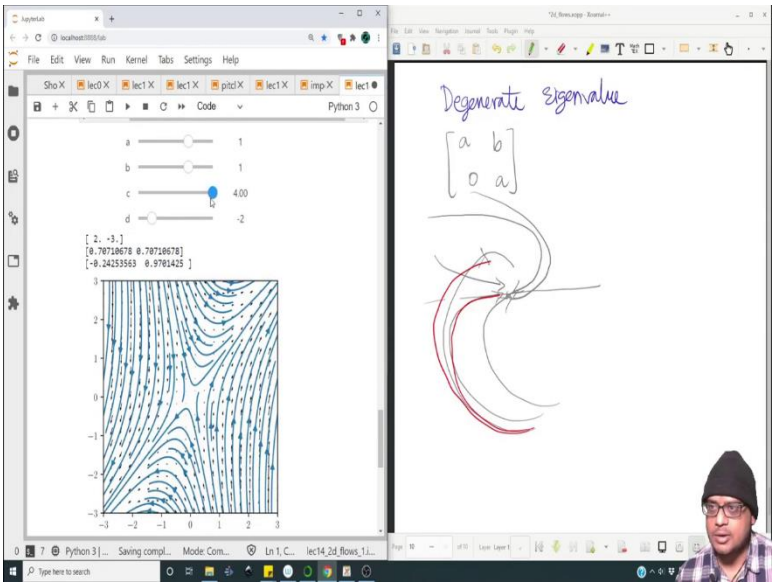
Even before doing a first round of the spiral, it is getting attracted towards it, ok. So, it indicates that the degenerate eigen value has to be something which transitions from the behavior of a spiral. So, let me just change this particular, I have value of the matrix to something, so that we can we can make sense of it.

(Refer Slide Time: 19:29)

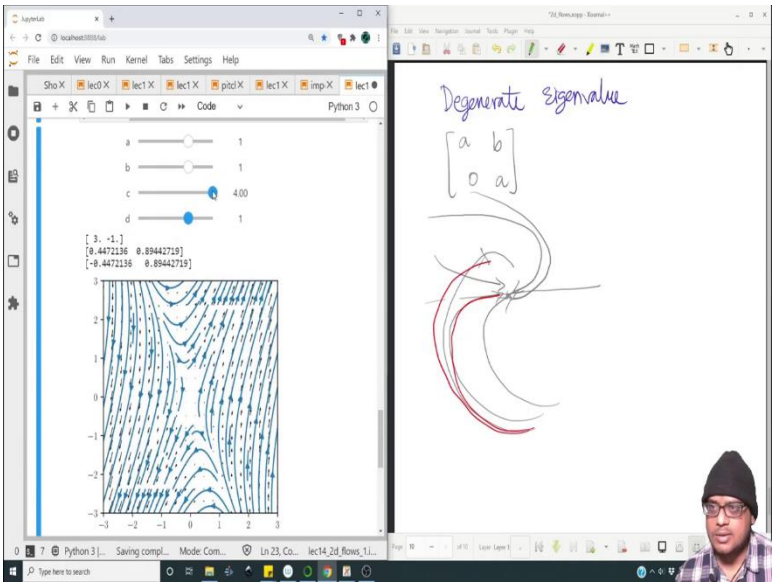


In fact, let me, let c vary by 0.1, so that the small changes can be reflected; let me run this again.

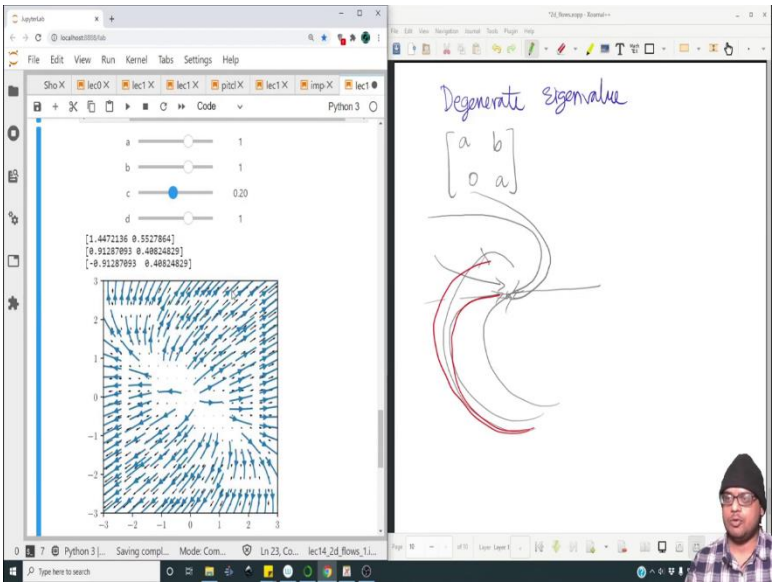
(Refer Slide Time: 19:48)



(Refer Slide Time: 19:53)

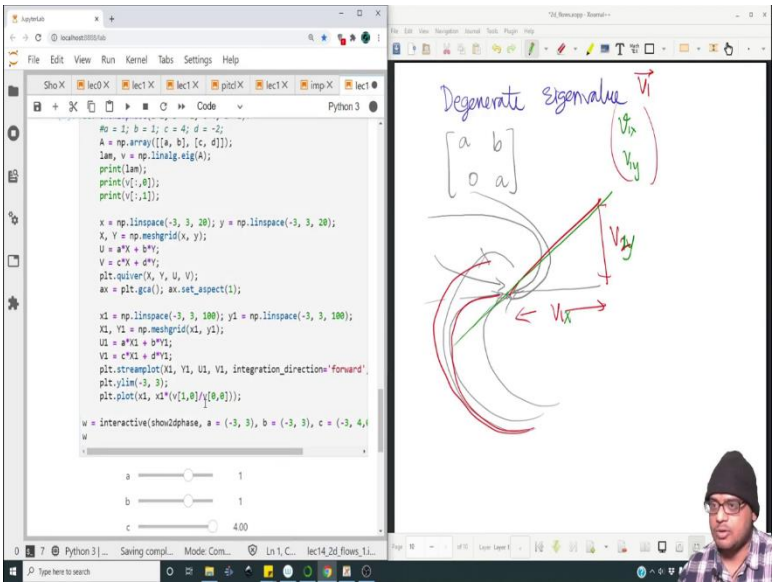


(Refer Slide Time: 20:00)



Let me go back to a and d being equal to 1 and let us c be something very close to 0. In fact, let us try to plot the eigen directions as well, that will be quite useful. So, the eigen directions we can plot very easily with the help of the eigen vectors giving us insight into the direction, ok.

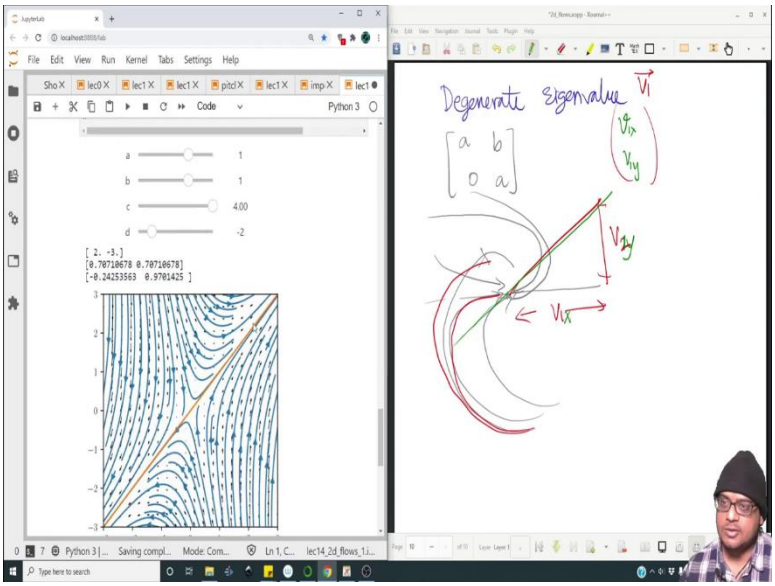
(Refer Slide Time: 20:17)



So, plt.plot x1, x1 times; 1 now the direction will be simply the eigenvector. So, this is the eigenvector, the direction of the eigenvector is simply V_2 by V_1 ok; because this will be like V_2, this will be like v_1. And V_2, V_1 are components of some eigenvector ok, I am just writing it like this; do not be confused that V_1 and V_2 are the two eigenvectors.

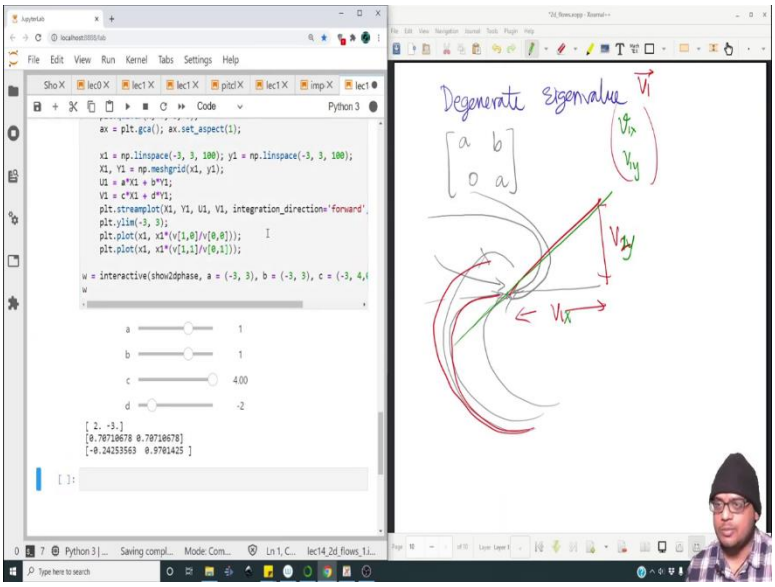
In fact, in order to avoid confusion, let me write them as V_{1x} and V_{1y} that make sense. So, this will be V_{1y} and this will be V_{1x}. So, the slope of that eigen vector will be V_{1y} by V_{1x}. So, we will plot on the x-axis it will be x, on the y axis it will be x times the slope, and the slope will be equal to $\frac{v_{1,0}}{v_{0,0}}$; let me plot this and see what it, ok.

(Refer Slide Time: 21:32)



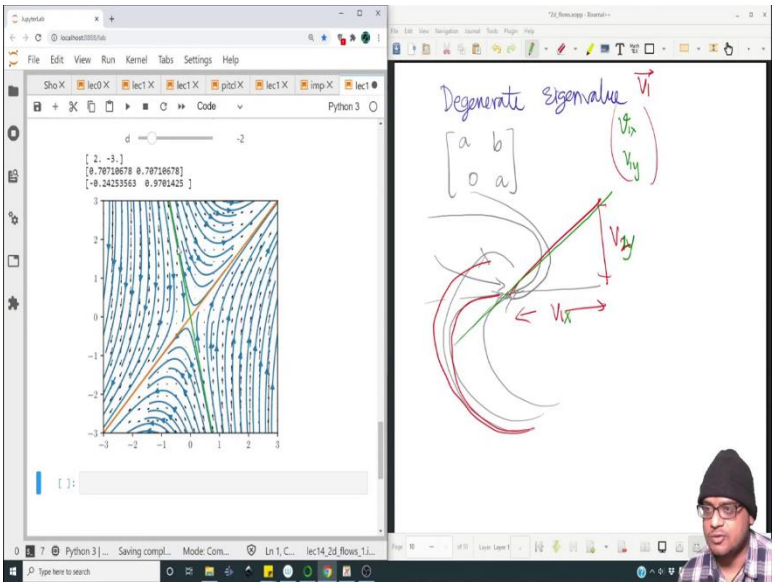
So, this is obviously, one eigenvalue and similarly we cannot eigenvalue eigenvector.

(Refer Slide Time: 21:39)



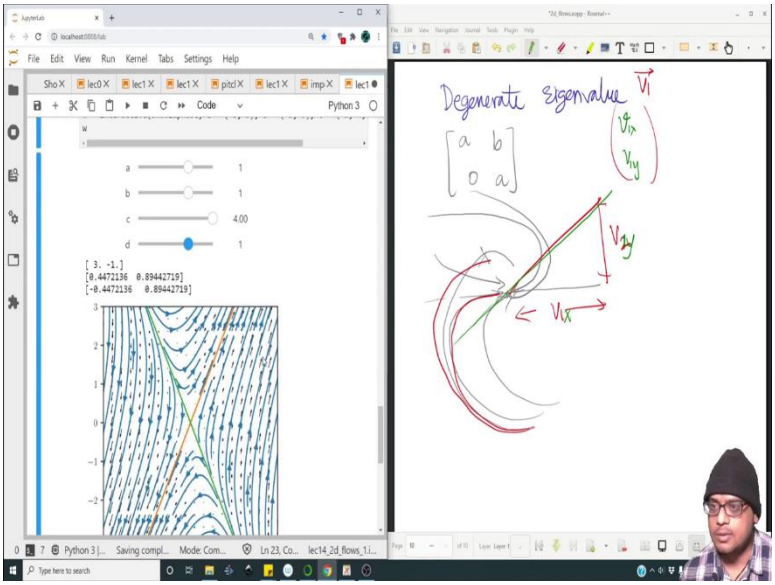
Similarly, we can plot the other eigenvector; this will be 1 and this will be 1, oops. So, these are the two eigen directions, let me plot this, ok.

(Refer Slide Time: 21:50)

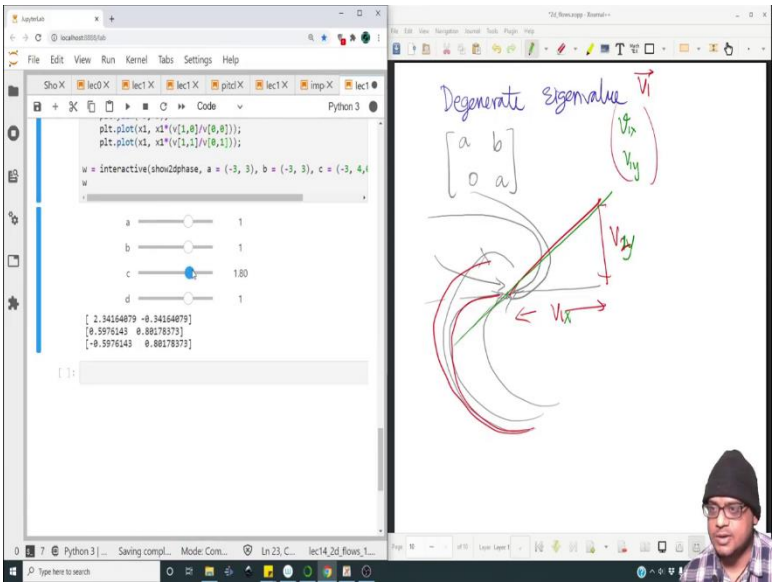


So, the green is one eigen direction, the orange is one eigen direction and the blues are all the trajectories. Let us go back to this degenerate case.

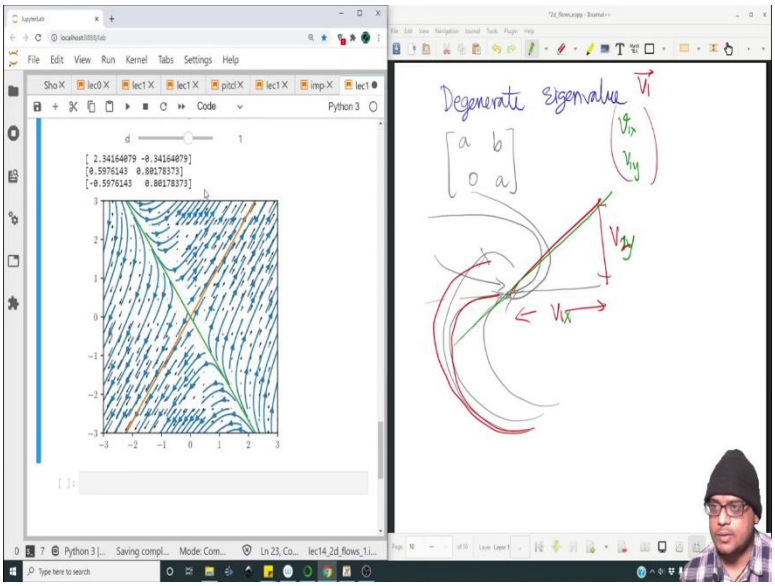
(Refer Slide Time: 22:03)



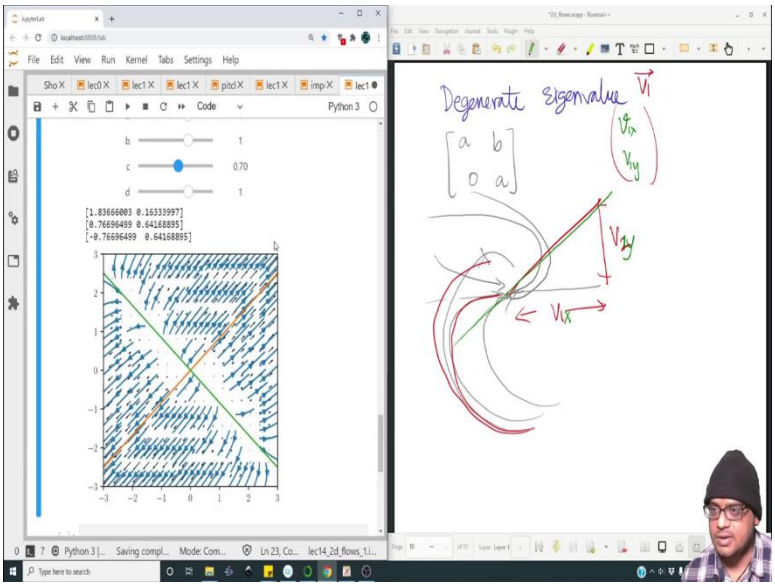
(Refer Slide Time: 22:07)



(Refer Slide Time: 22:08)

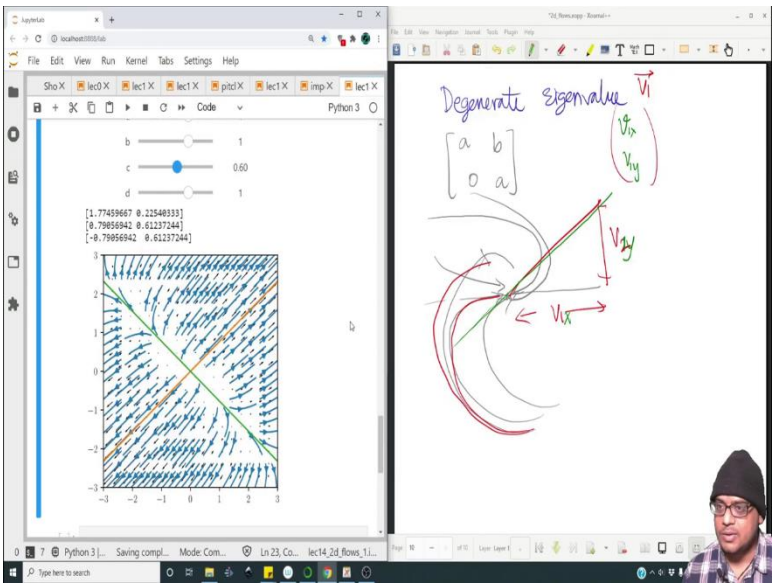


(Refer Slide Time: 22:16)

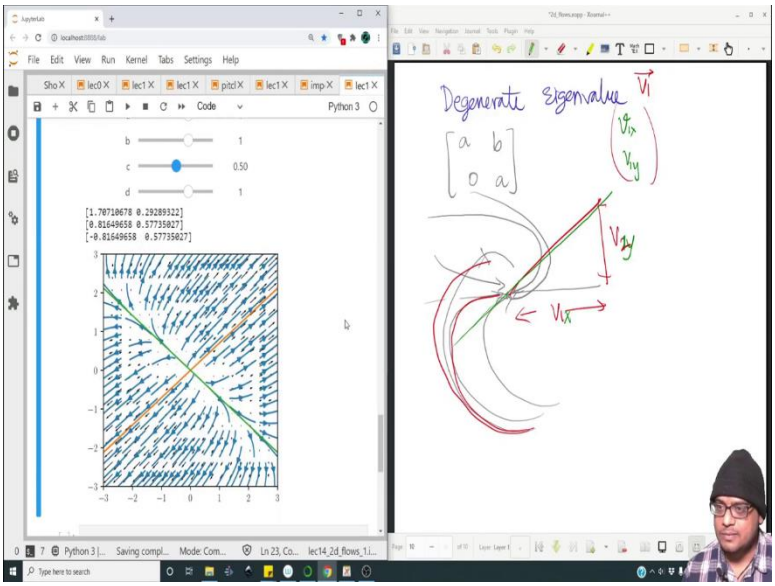


So, a and d are equal, alright. Now, c's value is decreasing. So, let us reduce c slowly. So, at 0.7 these are the two directions ok. So, the green line is obviously repelling, while the orange line is the unstable manifold. So, in this case the orange line is the unstable manifold and the green line is also the unstable manifold. So, so the origin is now repelling node, it is an unstable node ok; because both the eigen values are positive.

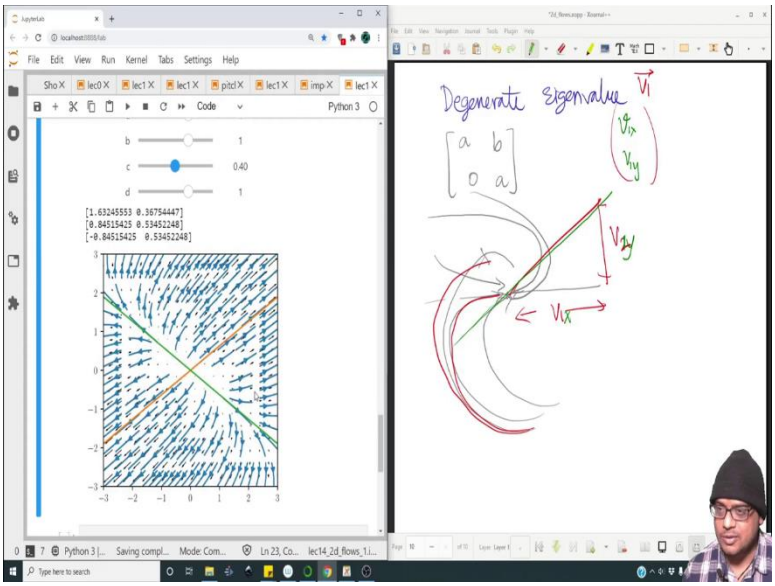
(Refer Slide Time: 22:51)



(Refer Slide Time: 22:54)

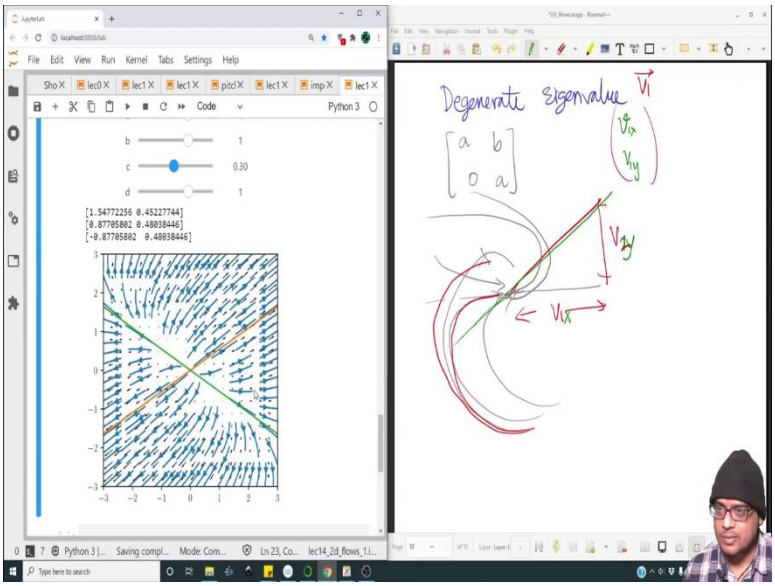


(Refer Slide Time: 22:59)

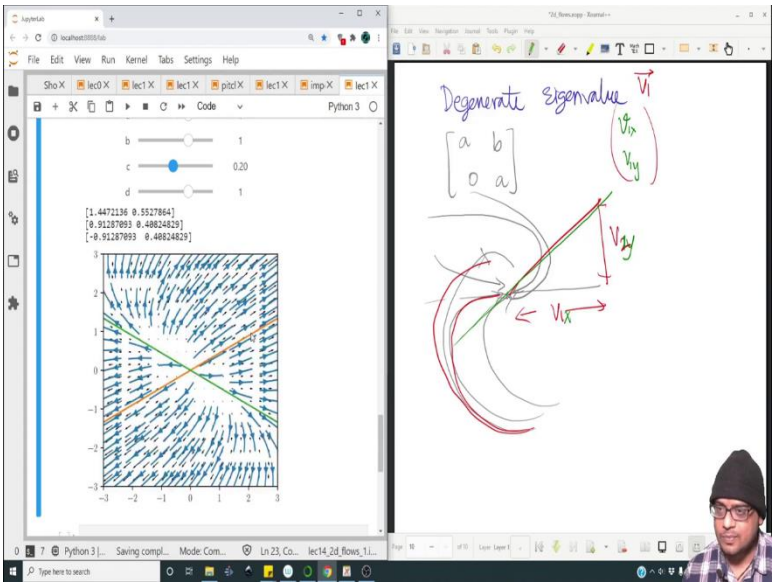


Now, when I further reduce c, let us see what happens; the two eigen directions, you can see that they are trying to come close together.

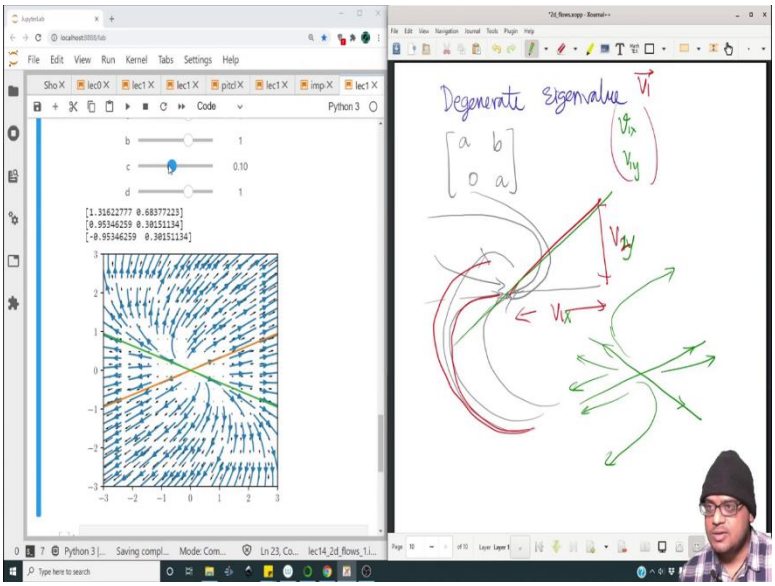
(Refer Slide Time: 23:01)



(Refer Slide Time: 23:03)

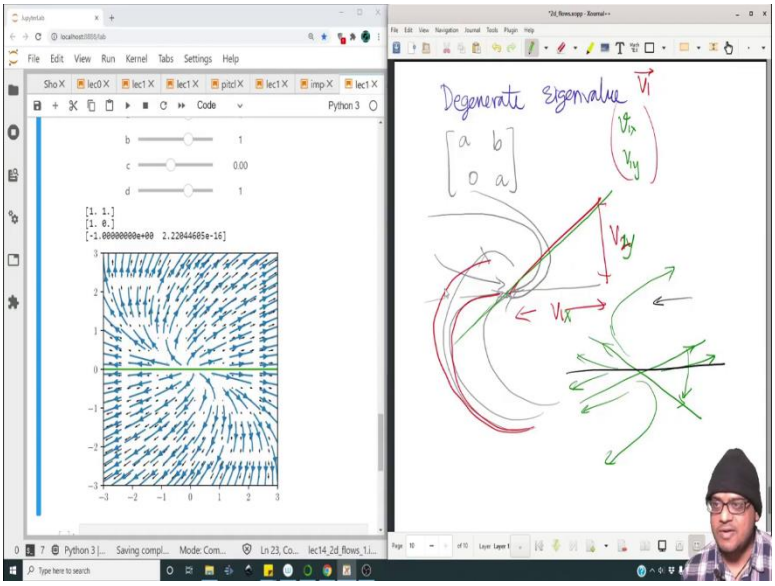


(Refer Slide Time: 23:04)



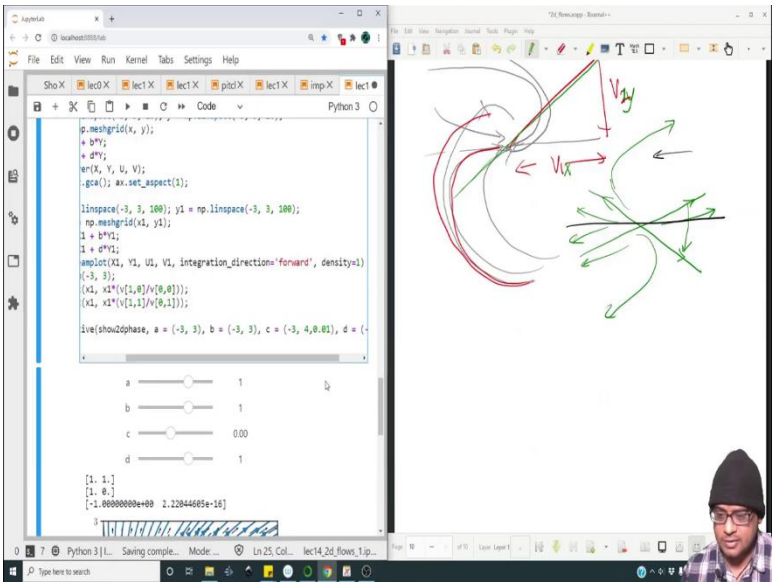
They are coming close; they are coming close and so both of them are repelling ok; both of them are repelling, this is repelling, this is also repelling. It is repelling like this, it is repelling towards this direction strongly; it is repelling towards this direction strongly, it is repelling like this, like this and like this.

(Refer Slide Time: 23:34)



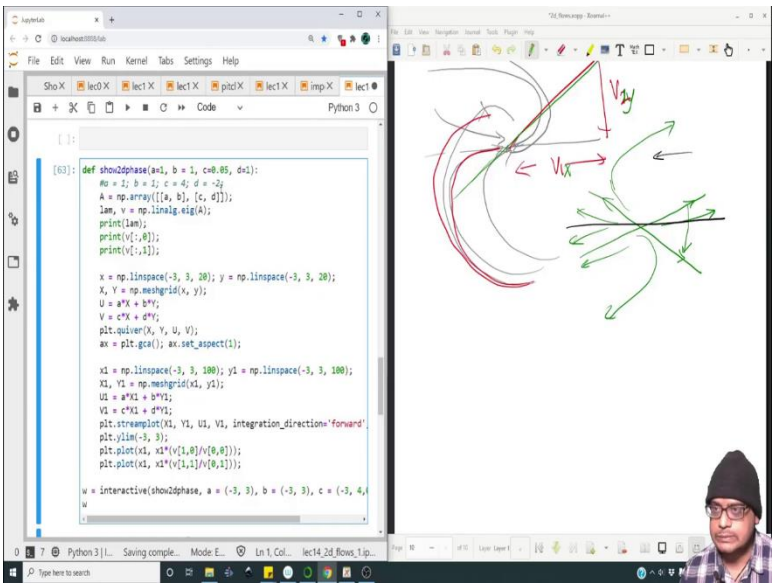
So, now when this becomes 0, these two eigen directions have merged, ok. So, essentially these two have merged into this axis and it is strongly repelling everywhere. So, whatever this pre merging direction was is like the behavior of this entire line.

(Refer Slide Time: 23:59)

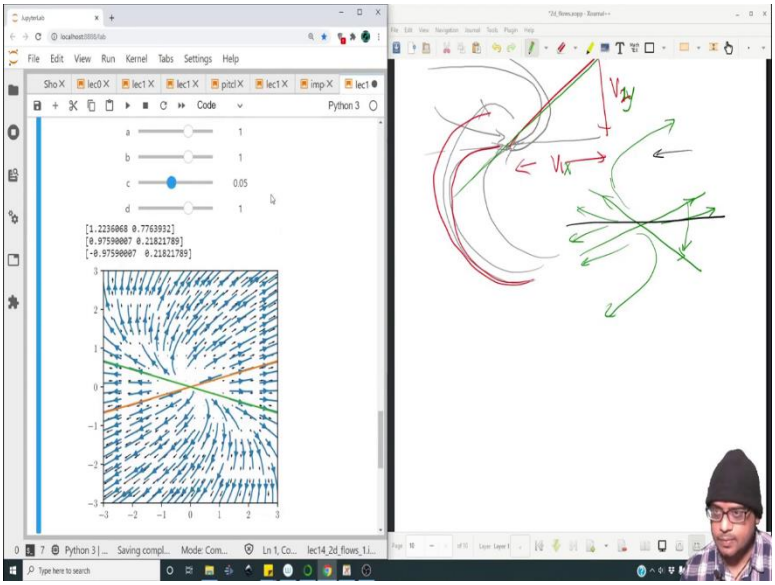


In fact, let me reduce the step size of this further and let me make the default values as something like this, ok.

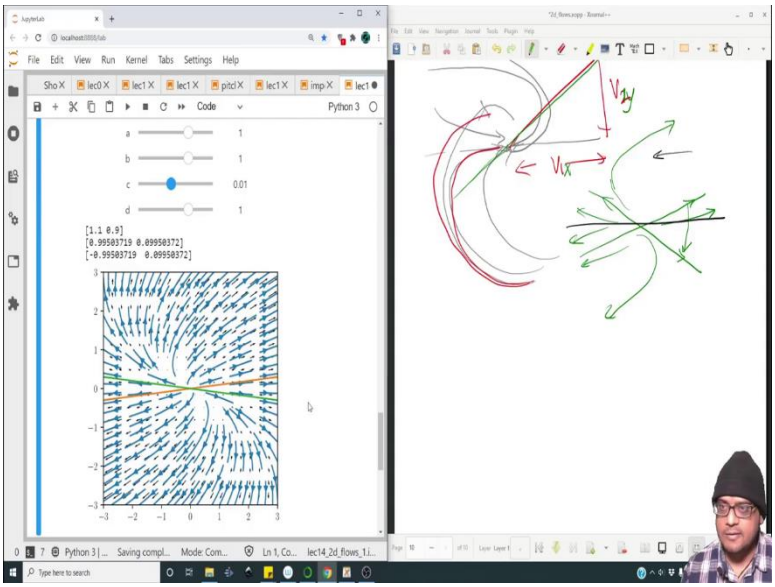
(Refer Slide Time: 24:05)



(Refer Slide Time: 24:11)

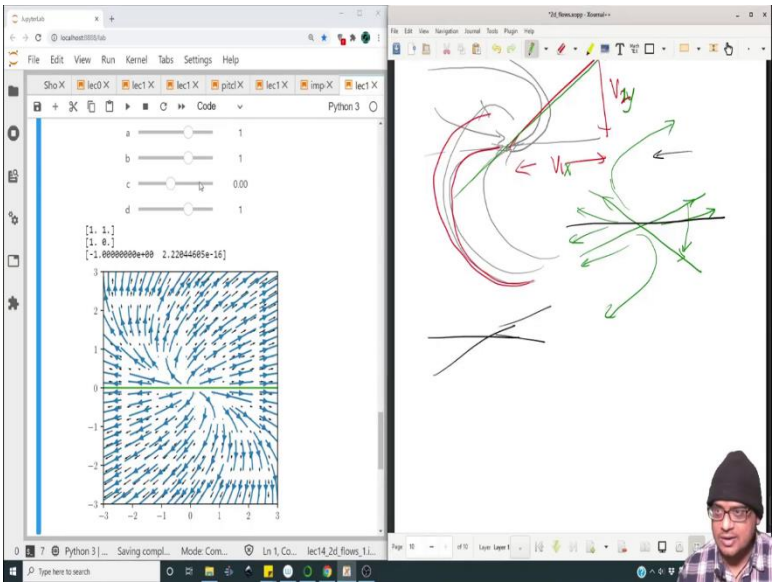


(Refer Slide Time: 24:22)



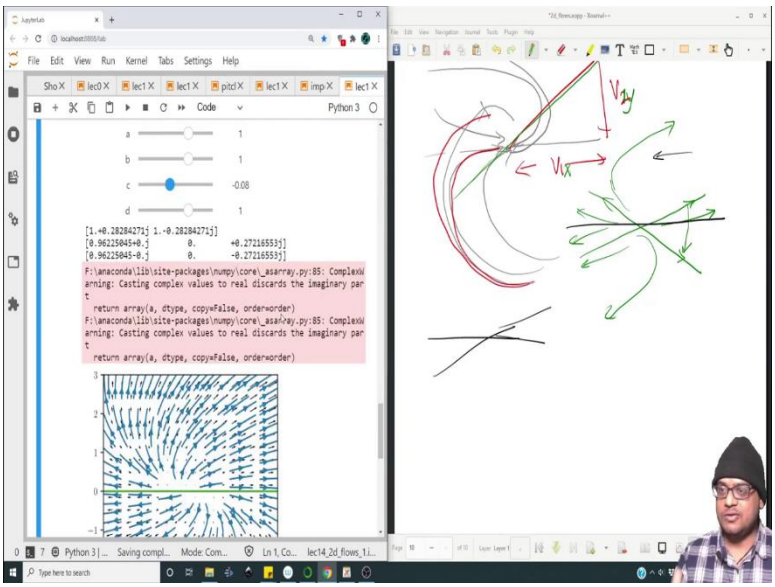
So, the eigenvalues, the eigenvectors have become really close and both these eigenvectors are repelling from the origin and once they merge, they become a degenerate eigenvector case, ok.

(Refer Slide Time: 24:36)



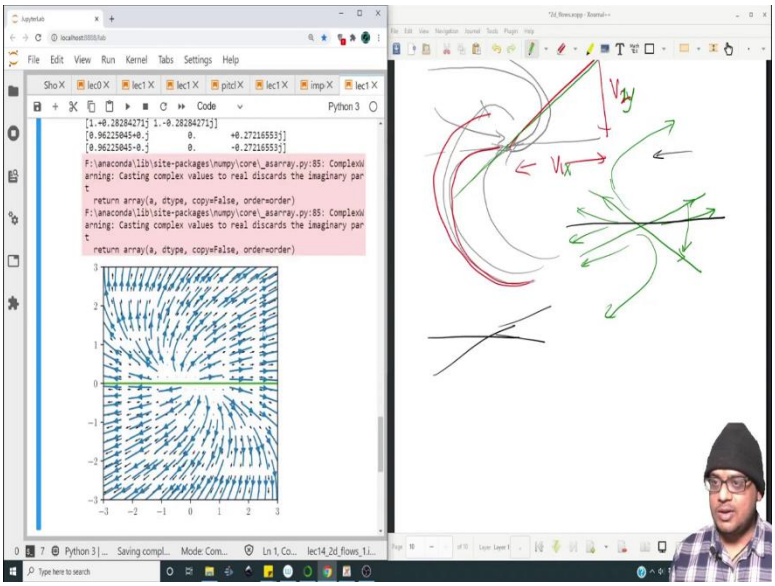
So, degenerate eigenvector case is somewhat of a transition between two repelling eigen directions.

(Refer Slide Time: 24:50)



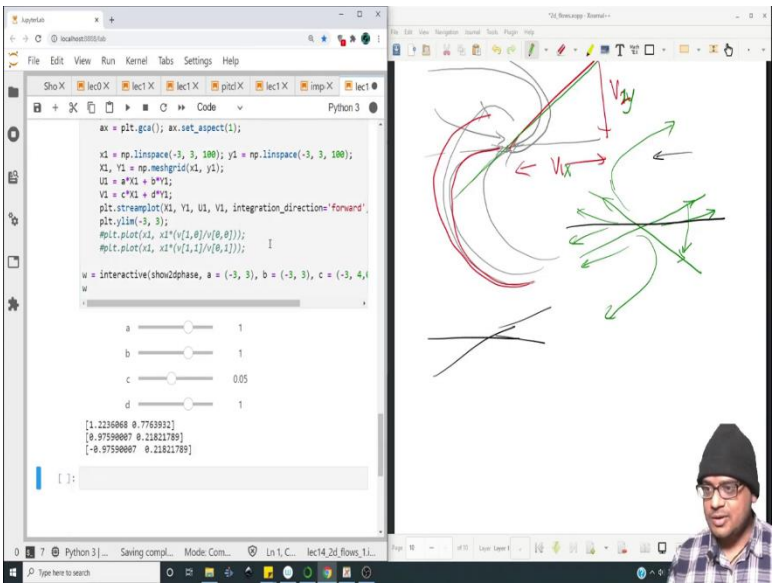
And let us see what happens, when we reduce c, ok.

(Refer Slide Time: 24:52)



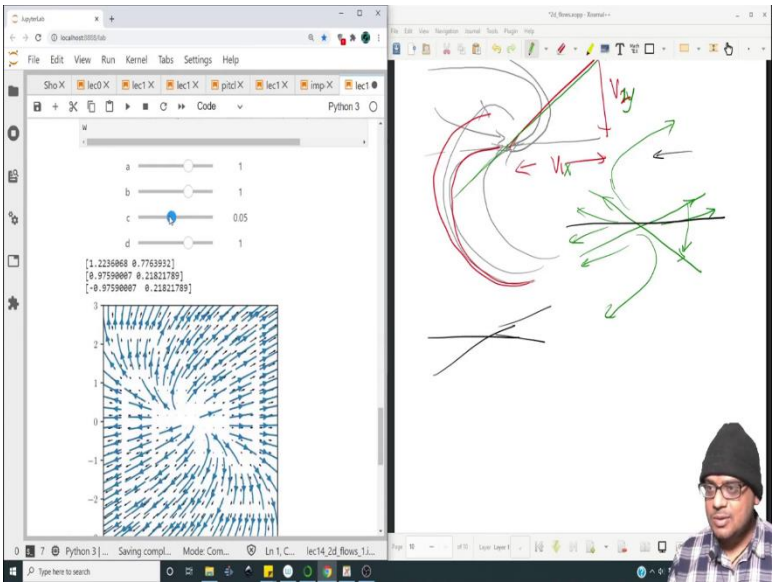
So, then it goes to a spiral case. So, forget about these errors, because we are trying to divide complex numbers and all.

(Refer Slide Time: 25:02)



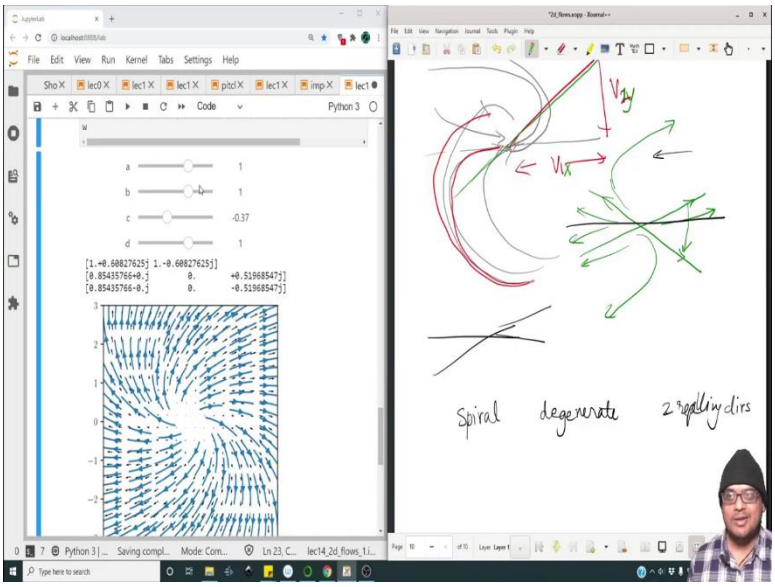
So, the errors are primarily because of this; let me get rid of this, let me run it again.

(Refer Slide Time: 25:07)



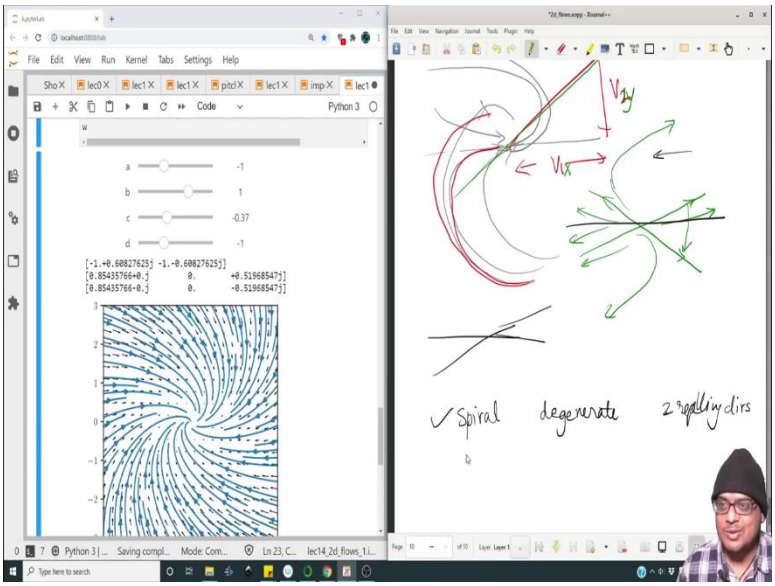
So, now, when c becomes negative, it transitions into a spiral.

(Refer Slide Time: 25:09)



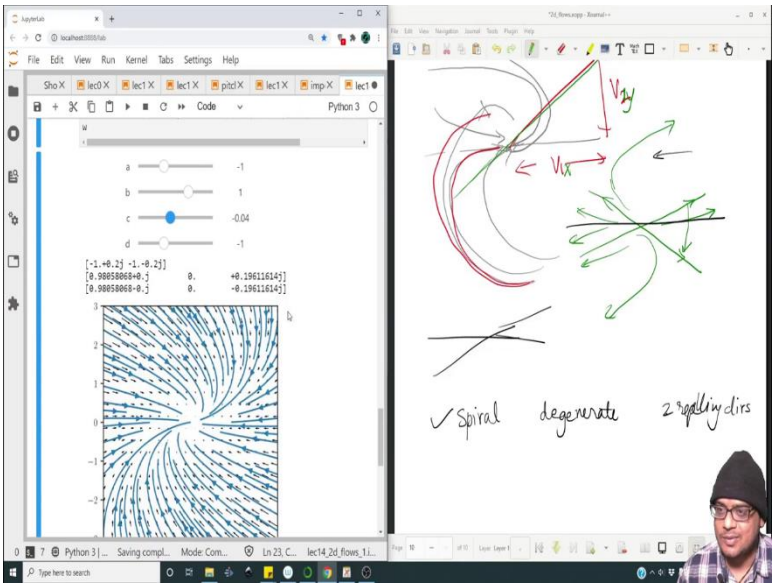
So, it is like spiral is one case and it is a growing spiral; we have the degenerate one case and 2 repelling directions as one case. And degenerate sits at the boundary between a spiral system and 2 repelling systems, ok.

(Refer Slide Time: 25:43)



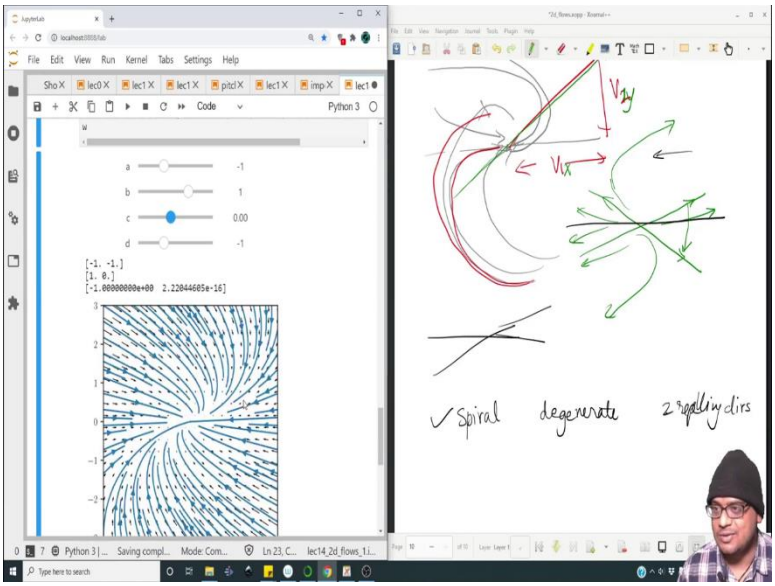
I hope it is clear by this illustrative example; let us go to a negative value and a negative value. In this case, it will be attracting; now, so it is incoming spiral. So, it is an incoming spiral.

(Refer Slide Time: 26: 04)



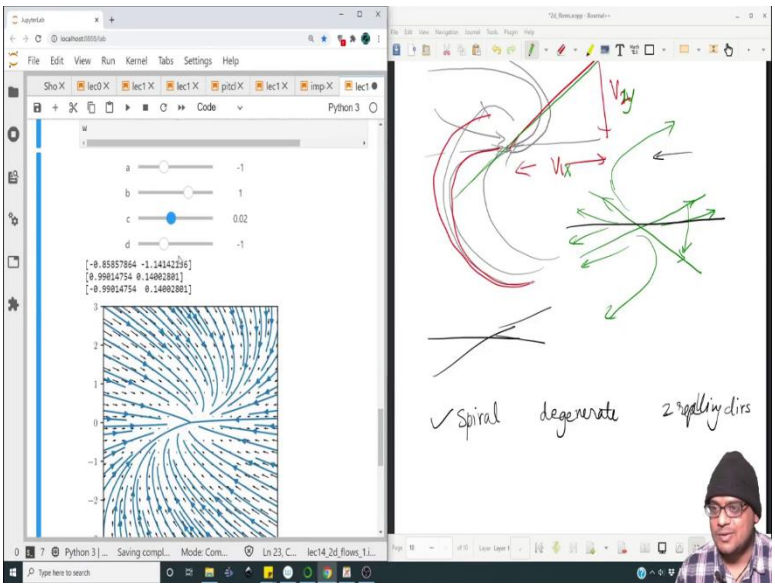
And let us now make c 0, so that it becomes degenerate again.

(Refer Slide Time: 26:07)



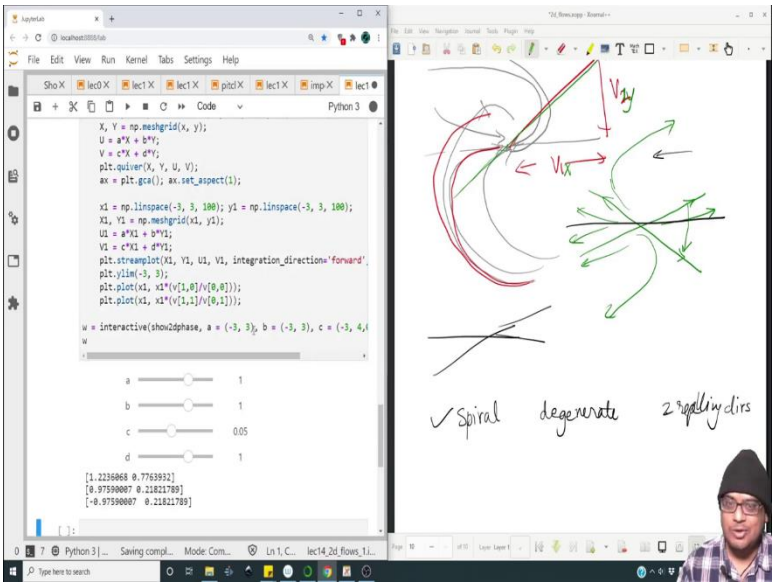
So, it is trying to become degenerate, boom it becomes degenerate; because the eigenvalues are now equal.

(Refer Slide Time: 26:13)



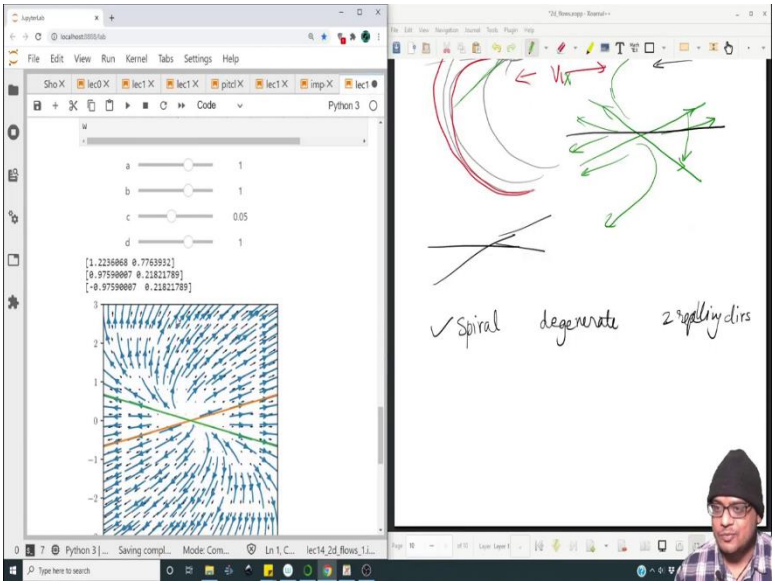
And once I go beyond this, we can switch on the plotting of the eigen direction.

(Refer Slide Time: 26:20)



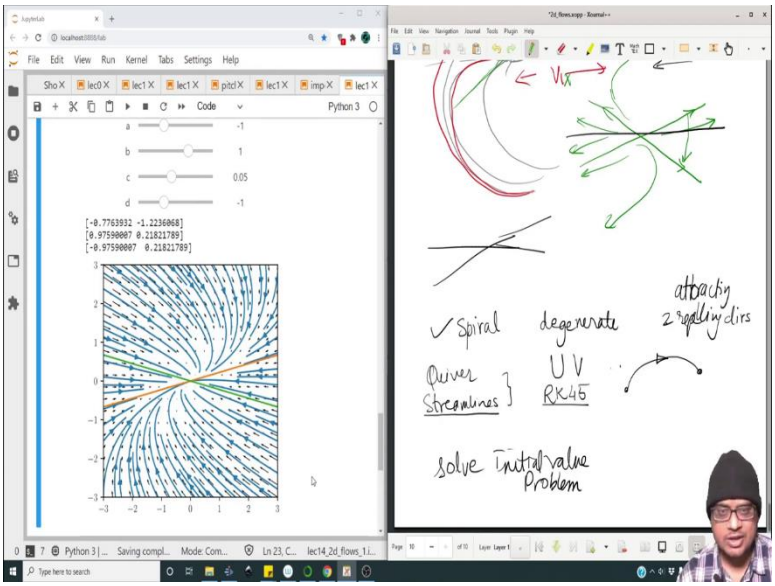
Actually, plotting the eigen direction over a spiral case makes no sense; because in time, it is rotating, it is not fixed.

(Refer Slide Time: 26:27)



So, for a -1 and b -1 and c positive.

(Refer Slide Time: 26:41)



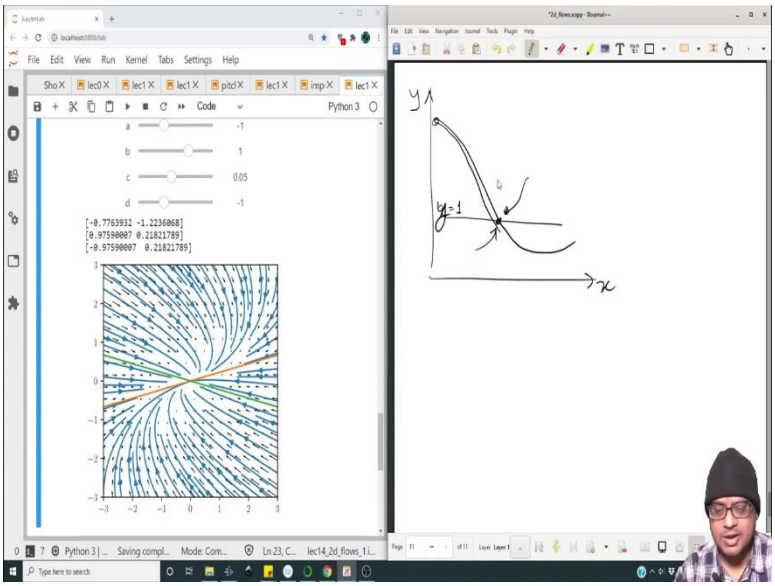
So, in this case, both are two attracting directions; it is not just repelling, ok. So, in this case this, these two eigen values are negative. So, both the eigen directions are attracting and the degenerate lies between these two cases; it lies between either a repelling in attracting spiral or it lies between two eigen directions which are attracting or repelling and at the boundary of those, we have a degenerate condition.

So, far I have discussed how you can do all these, you can make a quiver plot; you can find out the stream lines and they do give you an overview of what this particular two-dimensional flow looks like. But we can also perform a numerical integration and actually find out the trajectory instead of using streamline.

So, basically what streamline does, it uses the velocities that we have declared in all the snippets; using those velocities it performs an integration using the Runge Kutta method or whatever it is inbuilt. And based on this, it integrates this velocity field in time and finds out the trajectory.

But, in order to have a better control; we can actually solve the particular initial value problem and find out the trajectory ourselves as well, without having to appealed to this particular thing. And the benefit of doing the solution by means of one of these specialized algorithms is that, you can have various conditions inside the code itself.

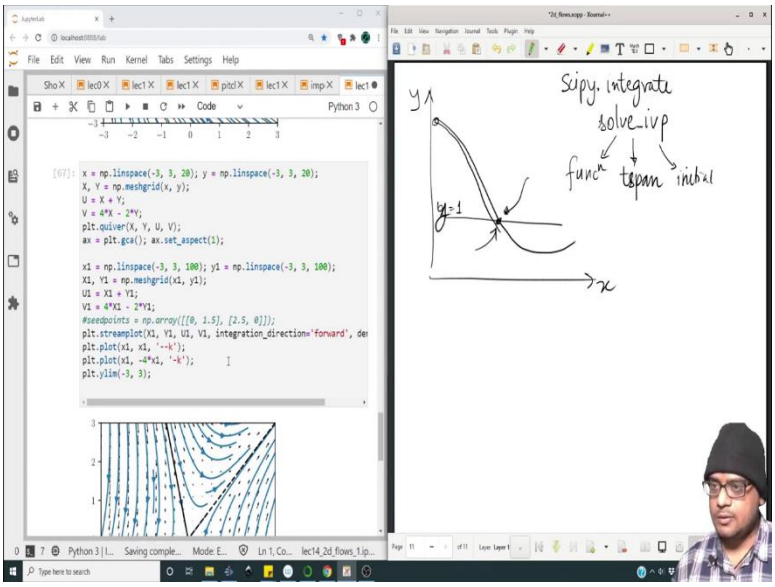
(Refer Slide Time: 28:45)



So, suppose you are solving an initial value problem and the trajectory looks something like this; we can ask the code to tell you at what particular time. So, this is the x and y axis.

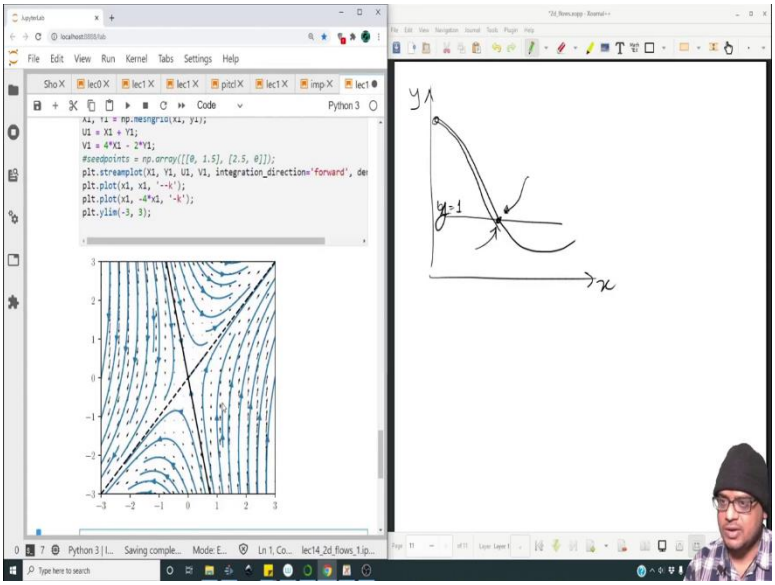
Now, suppose this is x=1, y=1; you can tell the code to tell you at what time this particular event has occurred, ok. So, in time, the trajectory follows this path and the code can actually tell you at what time event this has happened; but anyway, for now we will not require all these let me just grab a snippet really quick.

(Refer Slide Time: 29:23)



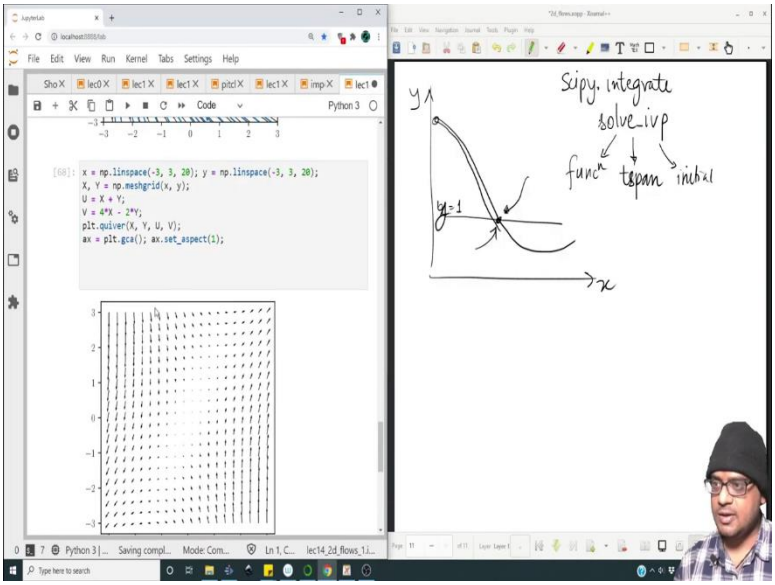
And let me show you how we can actually solve all of these using the IVP, the initial value problem solution, ok.

(Refer Slide Time: 29:36)



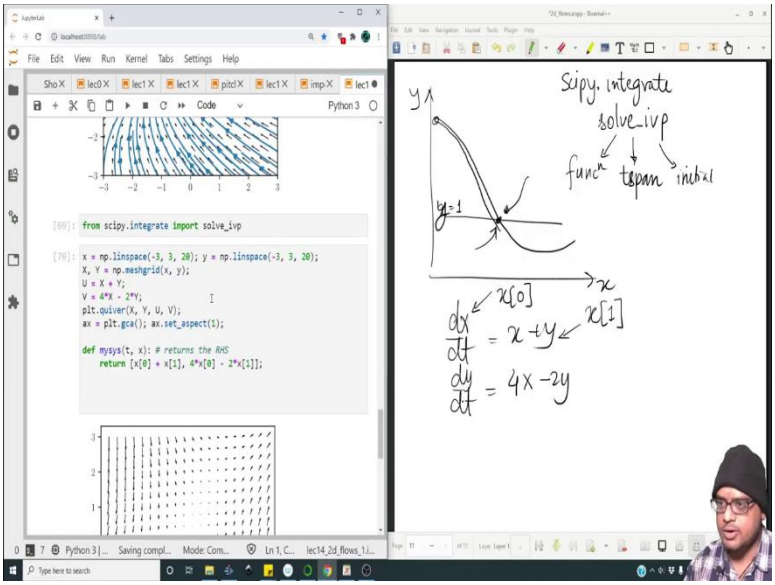
So, this is the particular flow. So, let me just run this code to make sure everything is fine ok, everything is fine, alright. So, how does it work? Inside the scipy dot integrate sub module, the function that resides is called as solve underscore ivp. And it expects as an input function handle; it expects as an input a time span, set of initial conditions and various other conditions and we will look at the contextual help when we encode this, ok.

(Refer Slide Time: 30:20)



So, we will not to be needing the stream plot in this particular case; we will only be needing, we will keep the quiver plot just in case, ok.

(Refer Slide Time: 30:27)

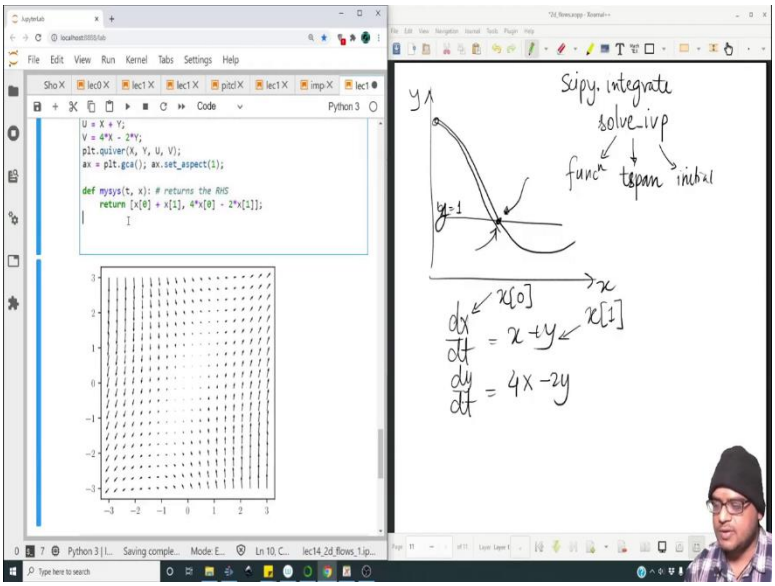


So, we will define the right-hand side. So, in this case the function that we are solving is $\frac{dx}{dt} = x + y$ and $\frac{dy}{dt} = 4x - 2y$. So, this is what we are solving. So, define mysys as the inputs will be time and x and it will return these two outputs. So, it will be $x_1 + x_0 + x_1$ and it will return $4x_0 - 2x_1$.

So, essentially the x variable appearing over here is equivalent to x_0 in that particular function and y appearing in this particular equation is equivalent to x_1 appearing in this particular function. So, this is what it has to return; it has to return an array of the right-hand side nothing else, ok. So, this is simply returns the RHS. In fact, it can be non-linear as well, it does not have to be linear; in this particular lecture, we are doing linear problems and so it is linear, ok.

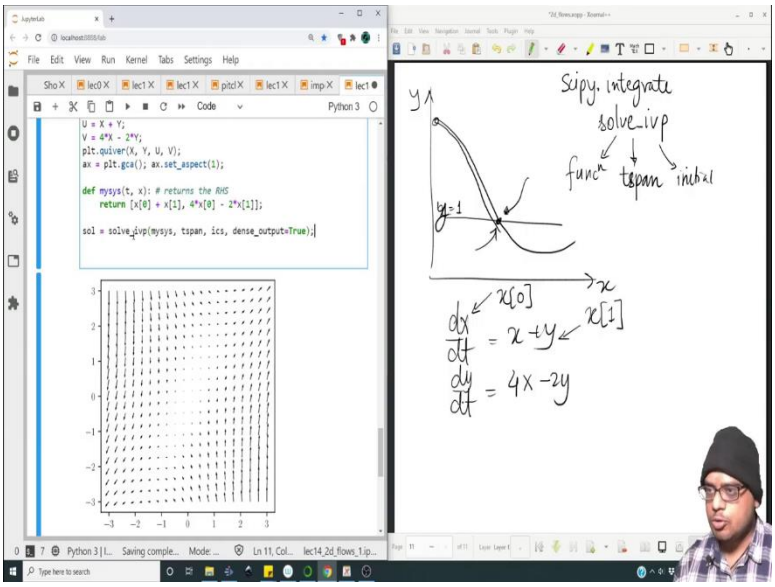
So, then once we have defined the function handle, we are going to go ahead and call the function. So, before that, in fact let me add a cell before this to import solve ivp. So, from scipy.integrate import solve_ivp.

(Refer Slide Time: 32:22)



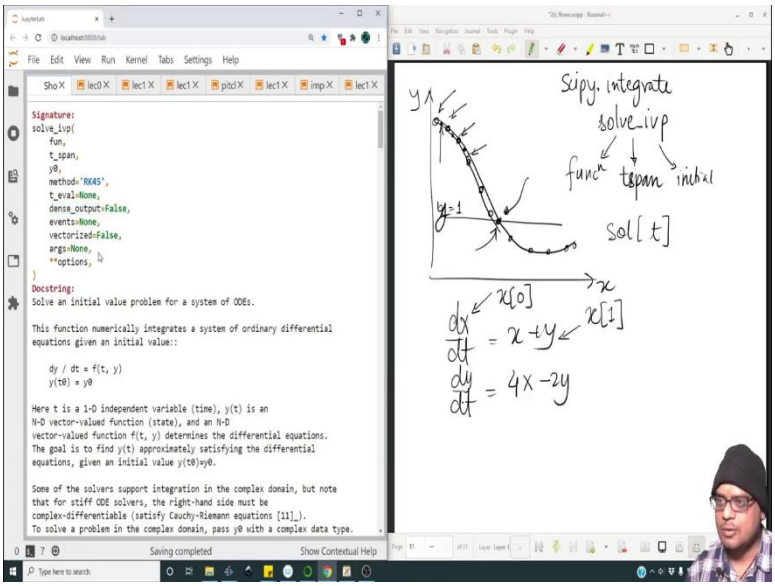
So, now, that is loaded; we can now go ahead and call this solve underscore ivp, ok.

(Refer Slide Time: 32:31)



So, sol is equal to solve underscore ivp mysys t span; we will define t span all that, ics. So, we do not have any additional arguments; so and dense output equal to true. So, let me go to the contextual help and show you what these things mean.

(Refer Slide Time: 33:01)



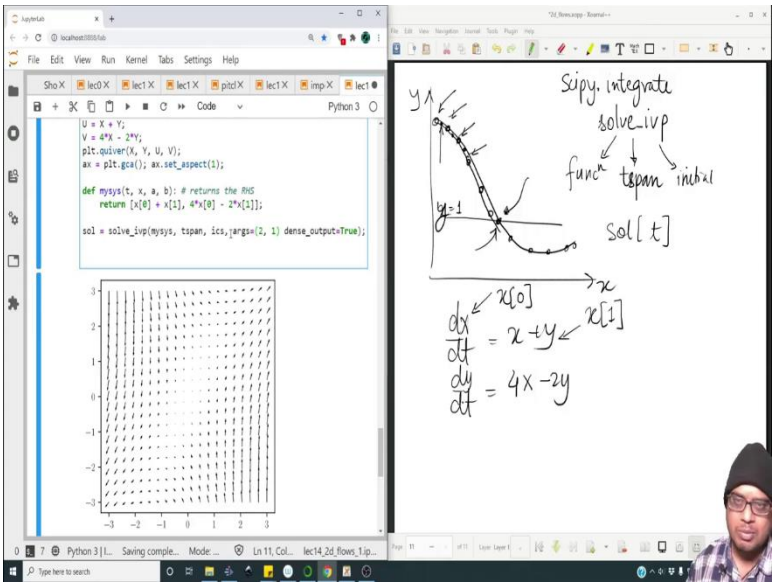
So, solve ivp takes an input the function handle, the time span, the initial condition; additionally, you can specify what method you would like the initial value problem solver to take, by default it is Runge Kutta 45 method. And if we have time later on in this course, we will discuss the bare bone implementation of Runge Kutta method.

t eval, that is if you want to evaluate the solution at some specified times; you can supply, we do not need to do that at this moment, dense output is by default False, ok. Dense output is a way of sort of outputting something which is an interpolating function. So, when you solve a differential equation numerically; you will obtain output at some discrete points ok, you will obtain outputs at this discrete point.

But if you say dense output equal to true; not only will you have the discrete outputs at the specified, not specified points at some discrete points. But if you ask the computer to give you these intermediate values at whatever times they are; it will also give you those values.

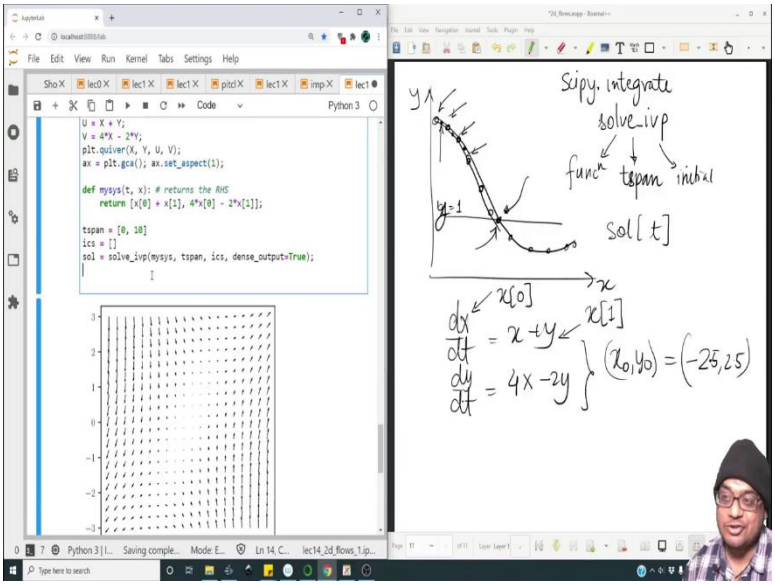
So, in a sense, if you say sol some other time which is other than those discrete time steps; it will give you the value of the solution and those time steps as well. So, it gives you an interpolating function. Once we implemented, it will be clear what it means; vectorized equal to false, args equal to none.

(Refer Slide Time: 34:43)



So, args actually if you have the function which takes as an input some additional parameters like a, b; then you can pass additional arguments as 1 2 1, suppose a would be passed as 2 and b would be passed as 1, this is how you would pass them. But in this particular case, we do not have any additional arguments that we want to pass. So, we do not put them.

(Refer Slide Time: 34:57)

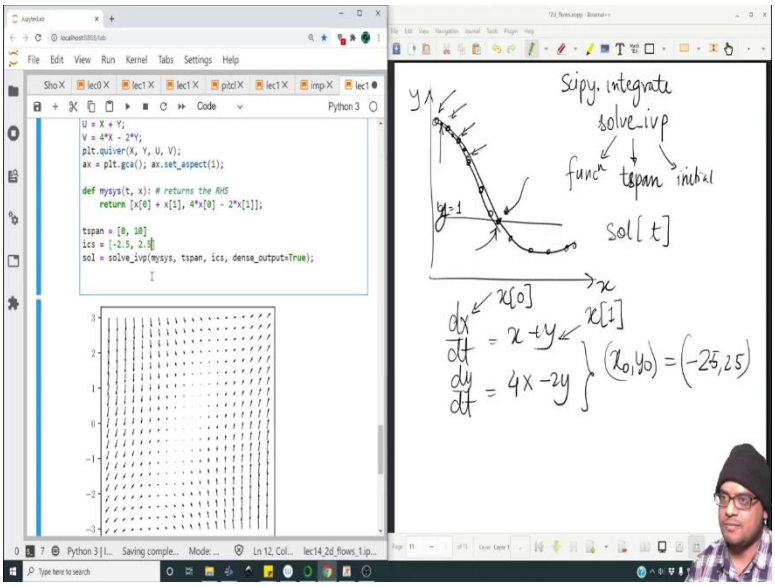


So, now we need to tell this snippet what tspan is going to be. So, tspan is an array which contains the initial time 0 and the final time say 10 second; not seconds, but 10, the initial

conditions. So, in the x y plane, the initial condition x_0, y_0 has to be specified. Let us take a condition somewhere over here.

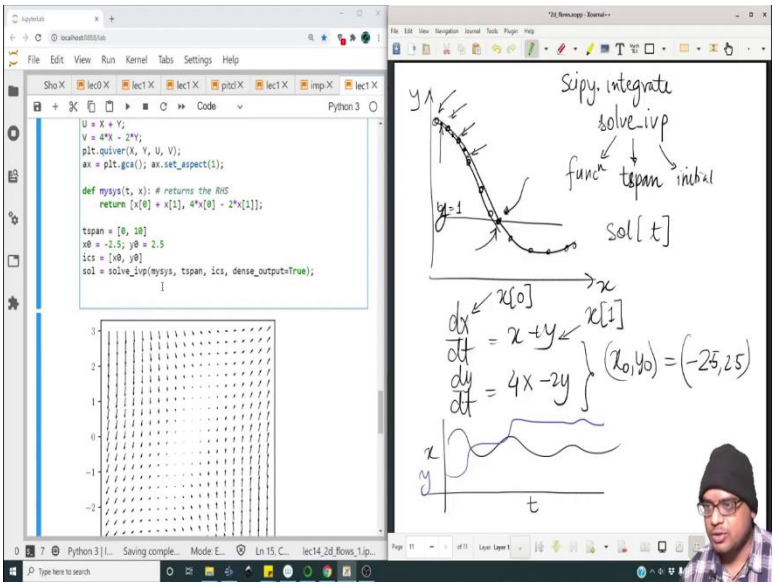
So, that point is going to be say (-2.5, 2.5); so (-2.5, 2.5). Actually, we can later wrap everything inside an interactive widget, which will show us that particular trajectory given that initial condition.

(Refer Slide Time: 35:53)



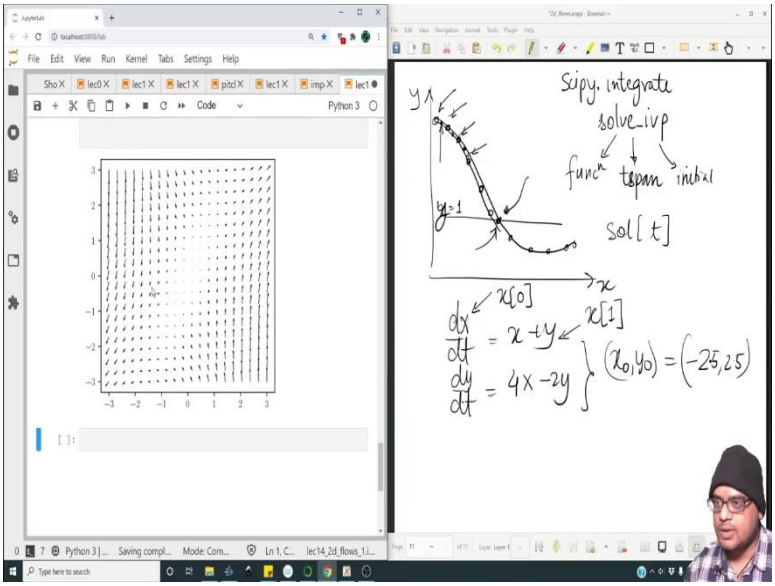
So, any ics will be simply (-2.5, 2.5) or actually we can soft code it as x_0 and y_0.

(Refer Slide Time: 36:00)



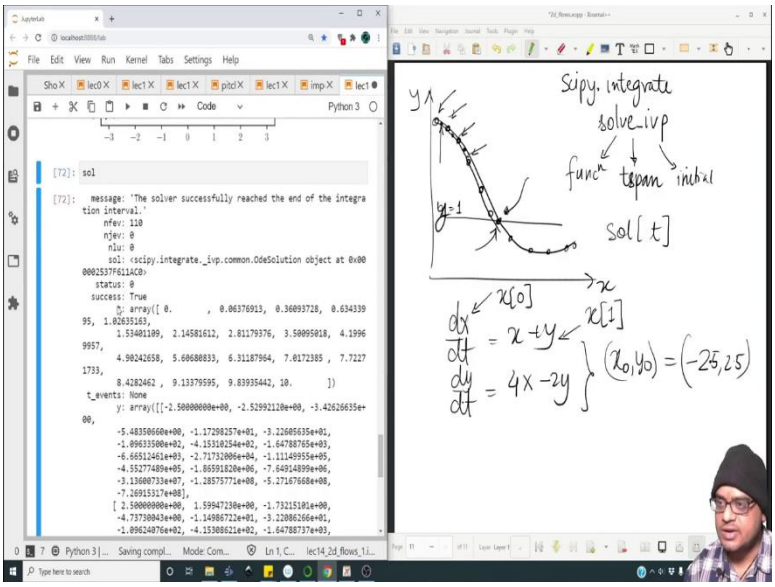
So, let me call it as (x_0,y_0). And let $x_0 = 2.5$ and let y_0 be equal; this is -2.5 and let $y_0 = 2.5$, ok.

(Refer Slide Time: 36:19)



So, let me run this and let me show you. So, after running it, there will be nothing which will be outputted; but let me show you what sol is.

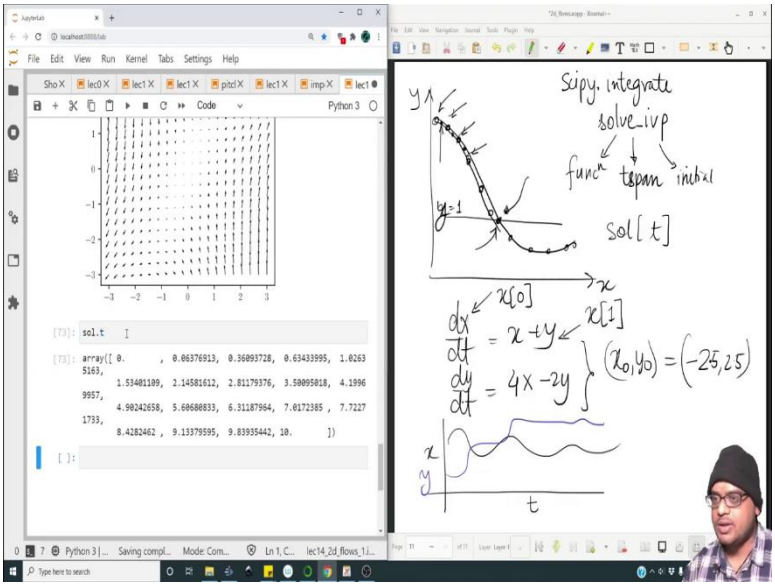
(Refer Slide Time: 36:26)



So, sol contains all this. So, because the dense output is said to be true; it says a message that the solver has reached the end of the integration steps, number of function evaluations is 110, it gives us an interpolation object called as sol, ok.

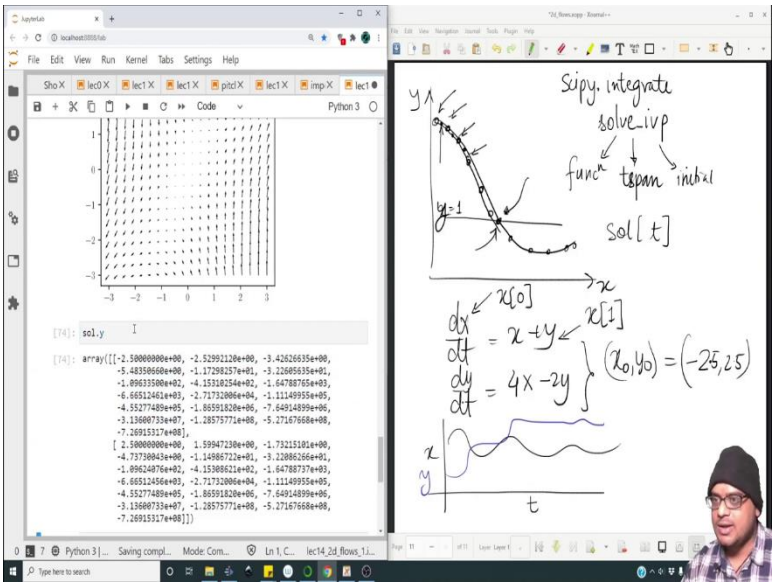
It gives us t and y arrays, ok. So, by default when we do this, it always outputs t and y. So, let me plot. So, essentially what will we have after solving this? We will have a time series of x as something and a time series of y as something. So, we have two time series x and y. So, let me show them; let me show them in this particular cell.

(Refer Slide Time: 37:19)



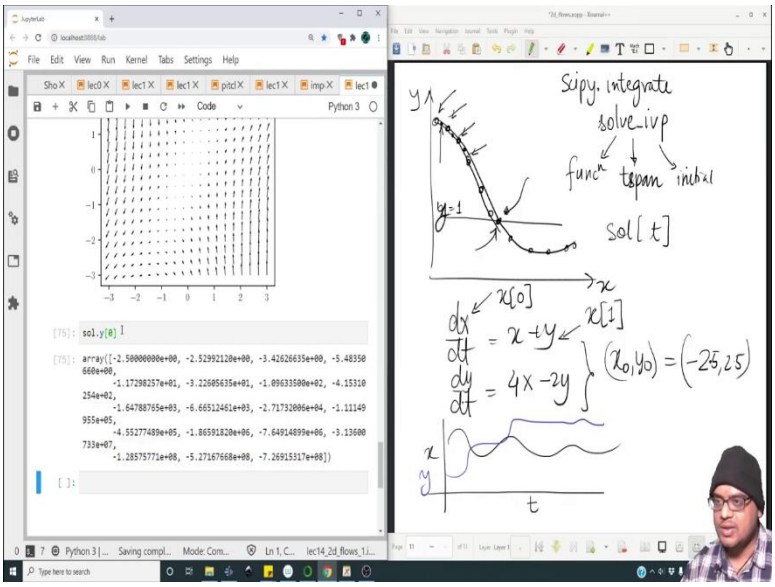
Sol dot t ok; so it contains all the different times. Then what do we have?

(Refer Slide Time: 37:25)

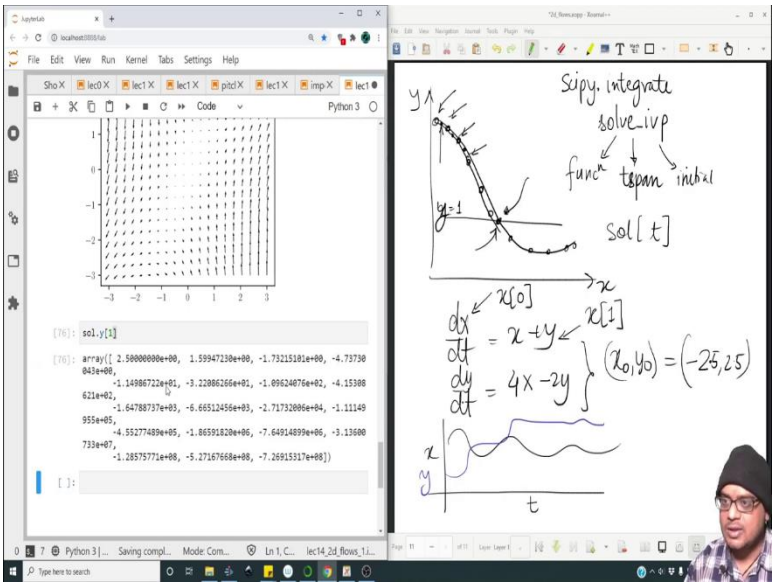


sol.y, it contains two arrays; one of them is x over here, one of them is y over here.

(Refer Slide Time: 37:34)

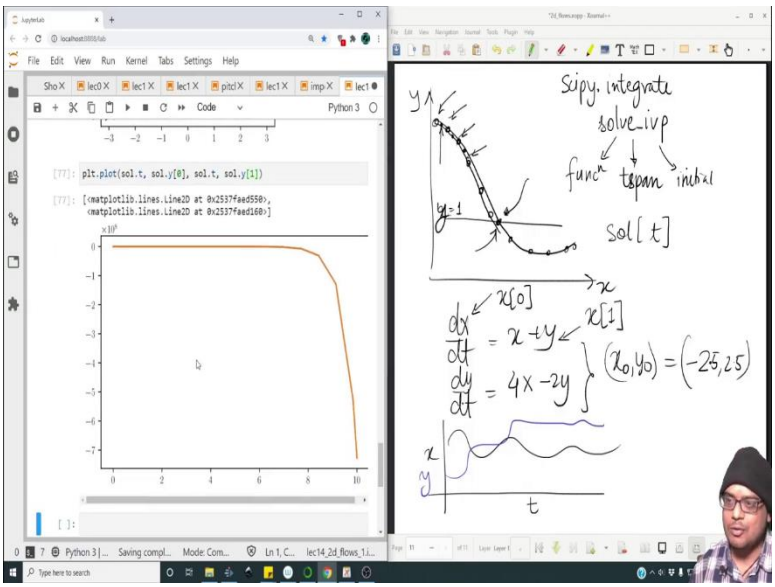


(Refer Slide Time: 37:40)



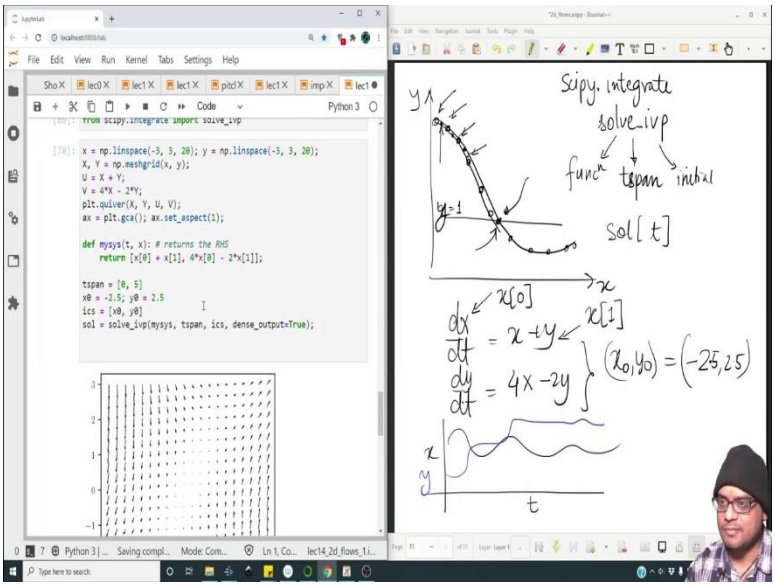
So, sol.0 is the x array corresponding to our solution and 1 will be the y array corresponding to our solution.

(Refer Slide Time: 37:47)

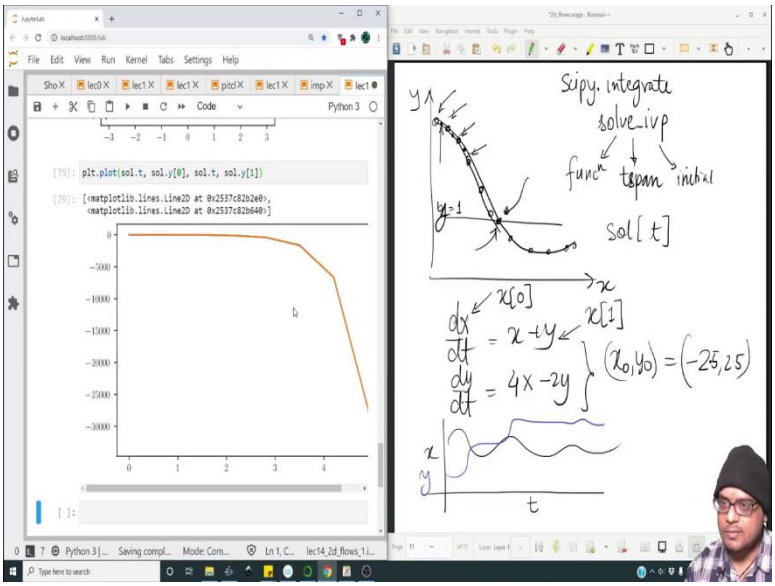


So, we have to do simply plt.plot(sol.p, sol.y0, sol.t, sol.y1) and this will give us the two time series that we are looking for, ok. Obviously, it appears to be unbounded for ok, that is it grows quite quickly. So, let us reduce the time step; because it has obviously flown off into -∞.

(Refer Slide Time: 38:22)

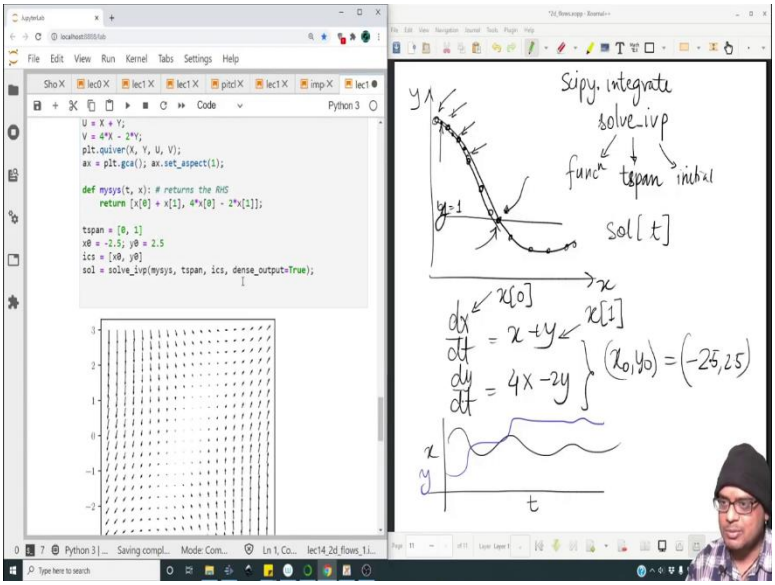


(Refer Slide Time: 38:25)

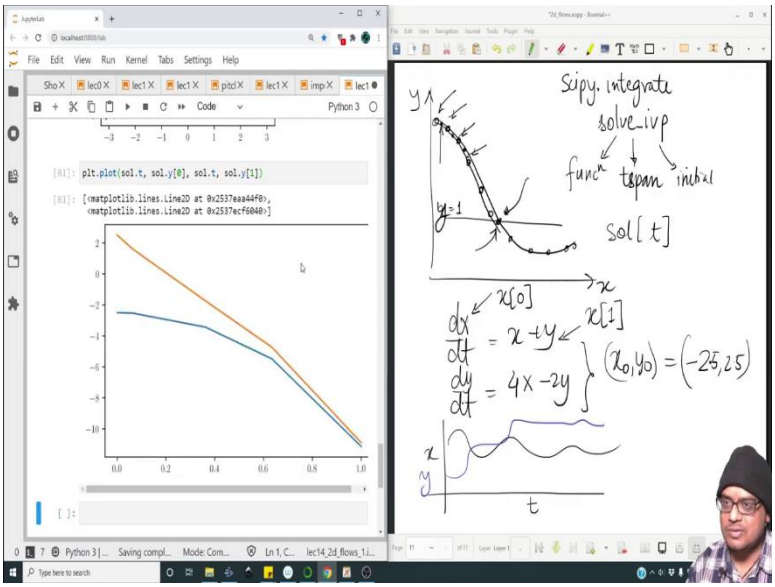


So, let me reduce it to 5. It will be the solution is $e^{-\lambda t}$, where if λ is quite high; then it will fly off to infinity quite quickly.

(Refer Slide Time: 38:44)

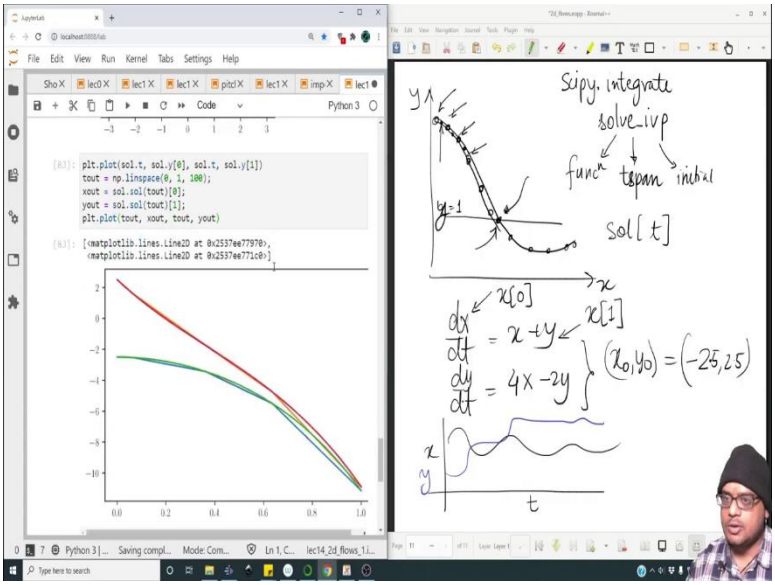


(Refer Slide Time: 38:47)



So, let us make it only 1 ok; let me make it only 1, let me execute this cell, ok. So, x starts from -2.5, y starts from 2.5 and this is how the solutions converge, ok.

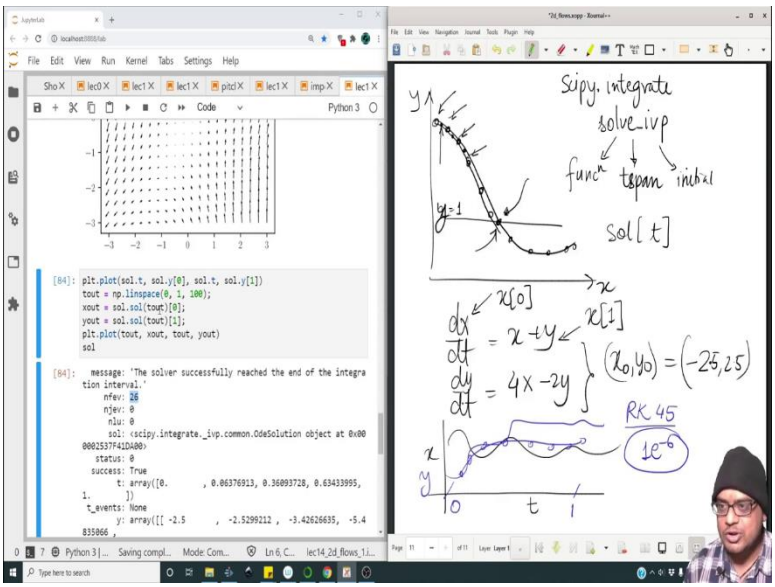
(Refer Slide Time: 39:03)



So, apart from this, we can alternately define the tout as np.linspace(0, 1) and say we take 100 points. Now, we want to interpolate the object. So, xout=sol.sol(tout)[0] and yout=sol.sol(tout)[1].

Let us plot all of this. So, plt.plot(tout, xout, tout, yout). So, the smoother curves are the interpolated objects, while the jagged curves are the outputs draw from the solver. So, the solver has only made how many evaluations I will show you that also.

(Refer Slide Time: 39:59)



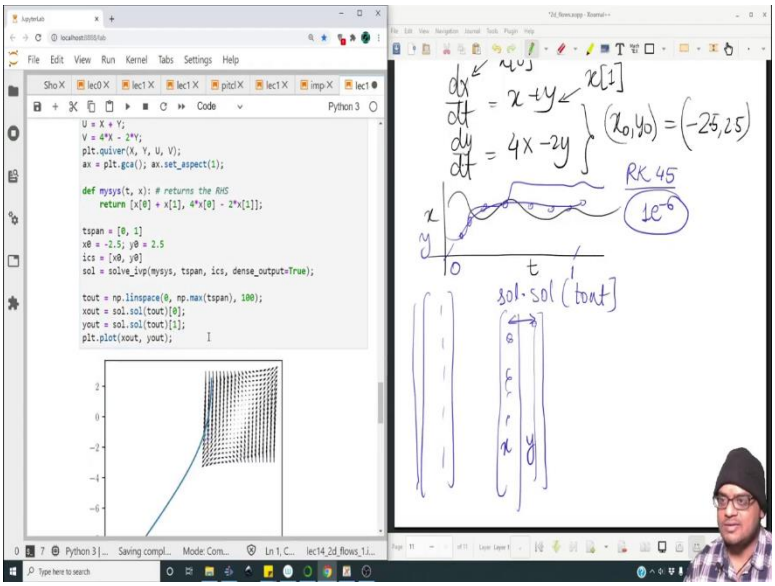
So, let me print; the object has only, the solver has only made 26 function evaluations. So, it means that, in going from t=0 to t=10; it only needed to make 26 function evaluations,

because the Runge Kutta solver, that is the default solver of the solve ivp function, it takes adaptive time steps in order to find the appropriate time step to minimize the error, ok.

It if it takes too large of a time step, it will induce more error into the solution; if it takes too little time step, you are not getting something which is magnificent, because there is some default tolerance of this. So, if you go below that tolerance, you do not really care; you want to maintain this error reference point. So, the RK 45 method tweaks the time step, so that it takes optimal time step. So, it has to make 26 iterations in order to integrate from t equal to 0 to 1 rather, this was 0 to 1 and hence that jagged line appears.

But if we use sol dot sol to interpolate, then we get smooth outputs. So, now, that I have shown you how to use the sol object; we can go back to this particular cell and we can plot the trajectory. So, how do you plot the trajectory? It is simple.

(Refer Slide Time: 41:37)



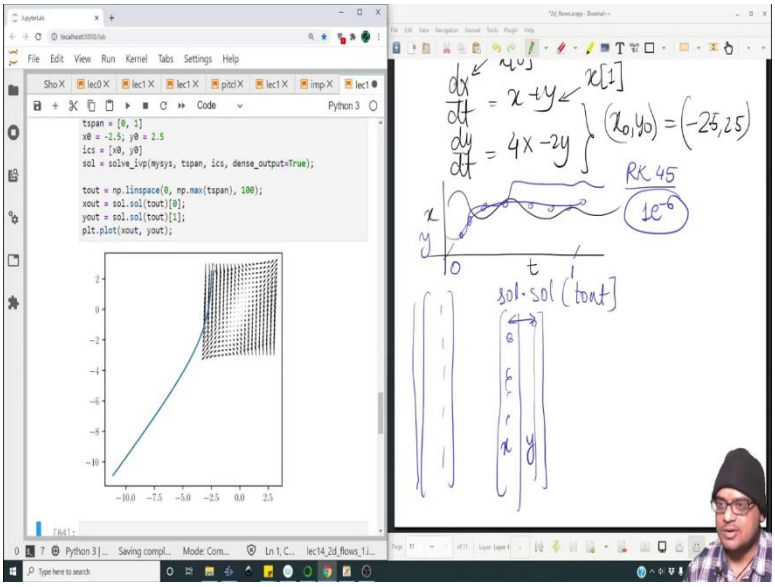
So, it says, it is simply let me define the output again. So, t out will be np.linspace(0, np.max(tspan)); I am soft coding it, because if I change tspan, I do not want to recode what tout will be.

Now, let me put 100 points; then x out will be sol.sol(tout). So, it is it will evaluate the solution using the interpolation object for all the t out specified by us, ok. These are all the t outs which are specified by us; x out we will make this as the first column of x out which is x, then let me copy this, y out will be the same thing, but the second column, ok.

So, essentially there is nothing to be confused about, t out is this array and when you call sol dot sol of t out, you get the same length, but two columns; the first column all the outputs are x values, second column all the outputs are y.

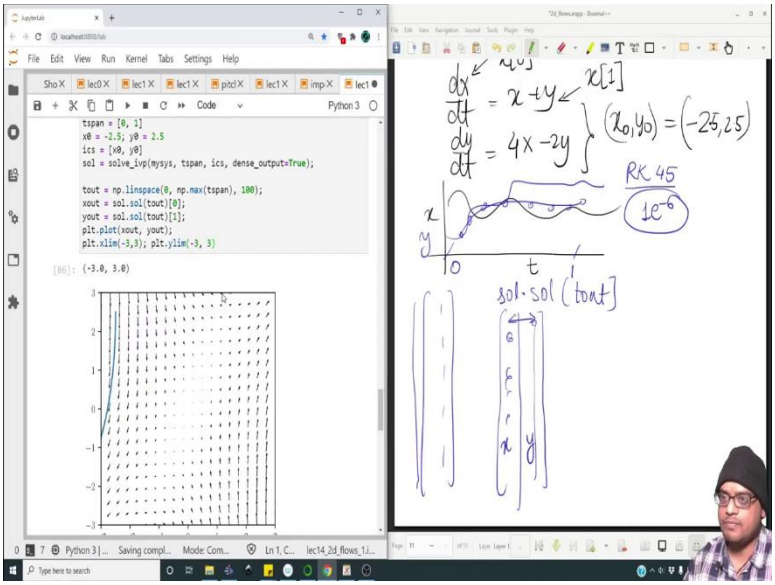
So, that is why I am assigning all the 0s to xout and all the 1s to yout ok. Once we are done with this, we will go ahead and simply plot the trajectory. So, in plotting the trajectory, time is not important; time is parameterized. So, we will simply do plt.plot on the x axis, we will have the x points; on the y axis, we will have the y points.

(Refer Slide Time: 43:37)



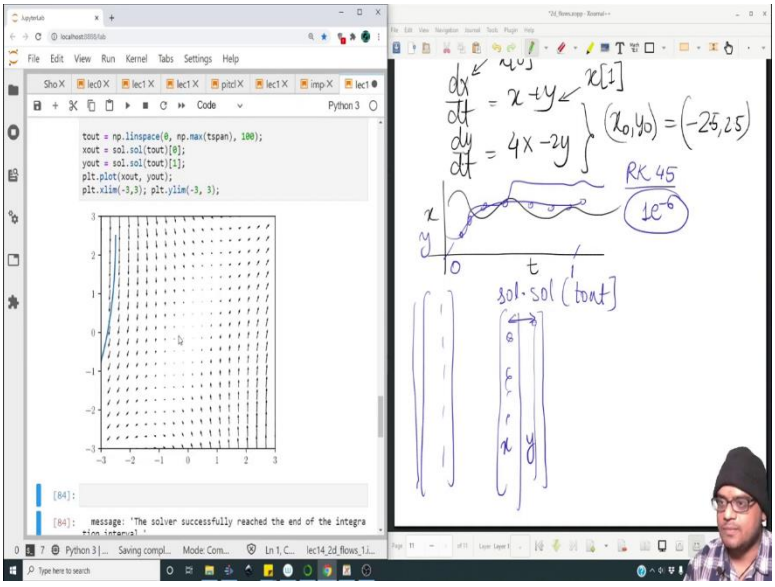
Let me run this ok, it is blowing away. So, let me bind the x limits and y limits on this problem.

(Refer Slide Time: 43:45)



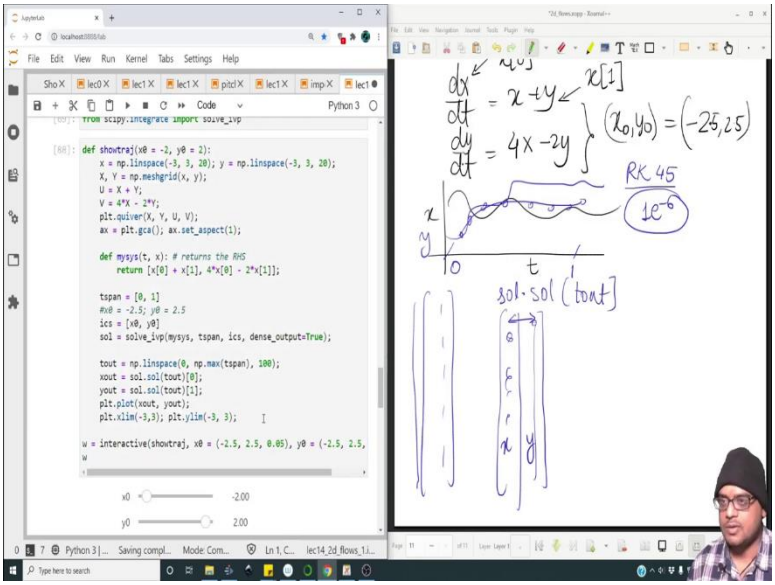
So, plt.xlim(-3, 3); plt.ylim(-3, 3). So, this is the trajectory, let me. So, I do not like to have this text; so, I am going to surprise it ok, excellent.

(Refer Slide Time: 44:02)



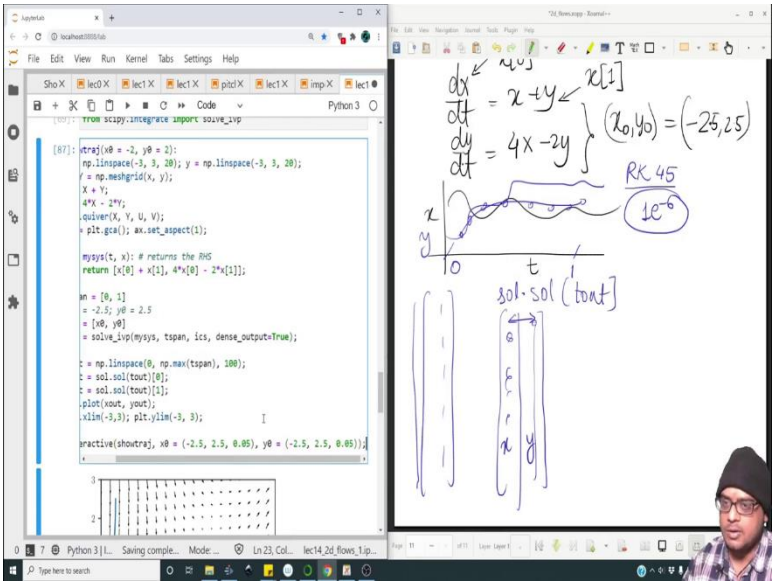
Now, let me wrap everything inside a function, so that we can pass different initial points and find out the trajectories corresponding to those particular initial points, ok. So, we are going to wrap all of this, let me give them an indent.

(Refer Slide Time: 44:26)

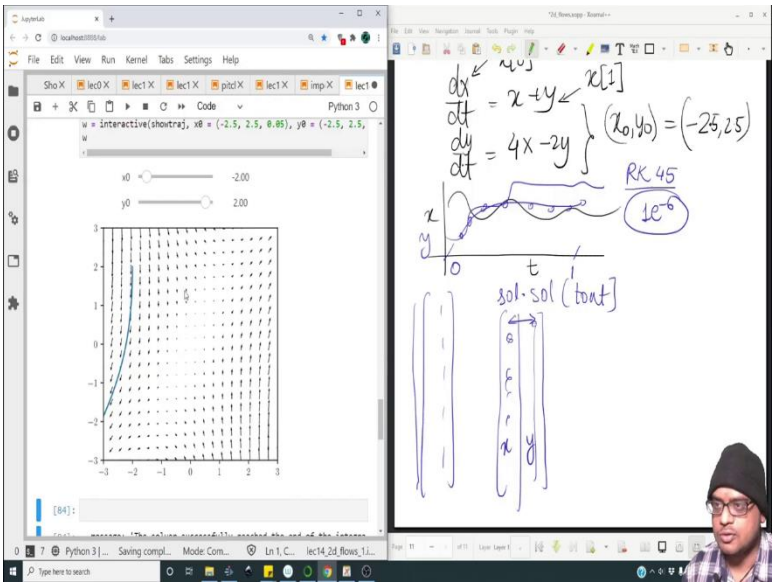


So, we are going to define showtraj(x0, y0); let me give some initial some default values of -2 and 2. So, these are just the default values, we are going to comment out this particular line; then we are going to write w equal to interactive showtraj, we are going to pass the function handle x0=-2.5, oh sorry this has to be -2.5 to 2.5 in steps of 0.05 and y0 will be -2.5 to 2.5 in steps of 0.05, then we are going to show the widget, ok.

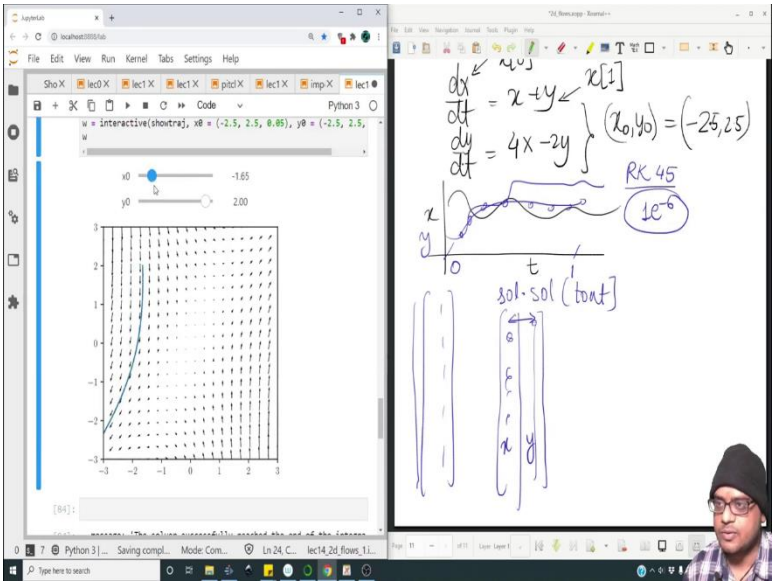
(Refer Slide Time: 45:12)



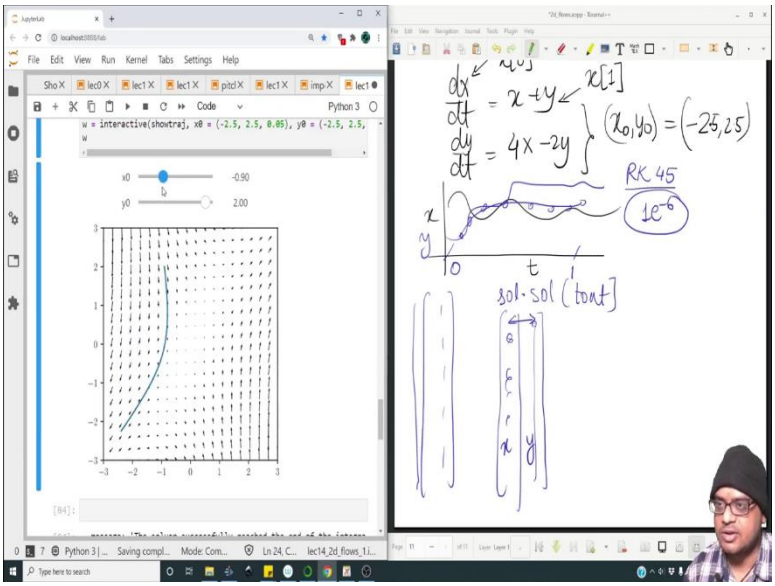
(Refer Slide Time: 45:17)



(Refer Slide Time: 45:20)

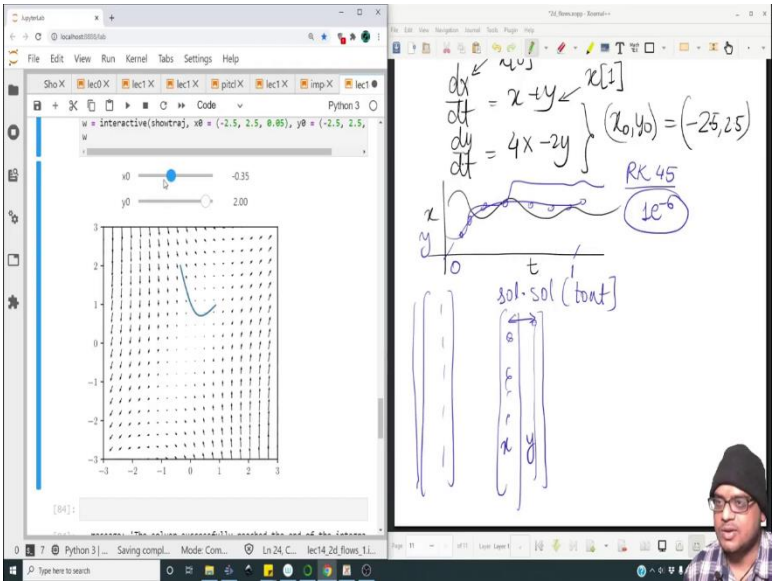


(Refer Slide Time: 45:22)

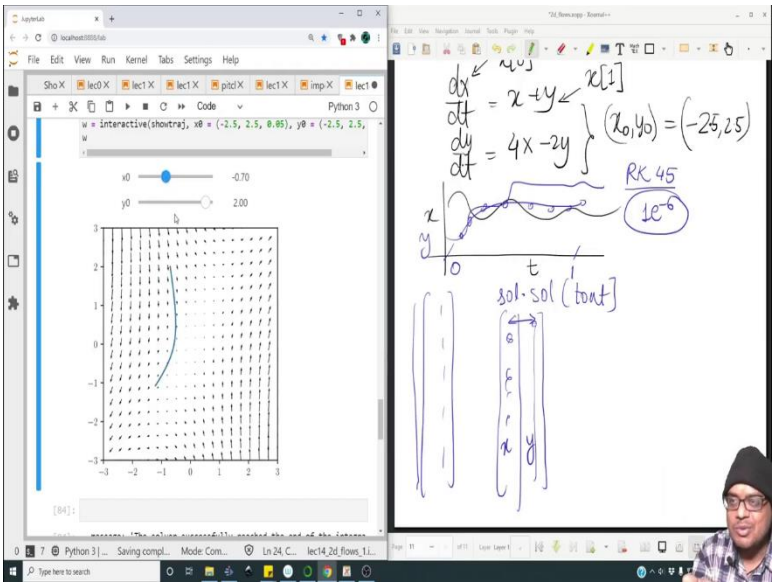


So, we have shown the widget, let me change, ok. So, now, this is how the trajectory appears, ok.

(Refer Slide Time: 45:24)

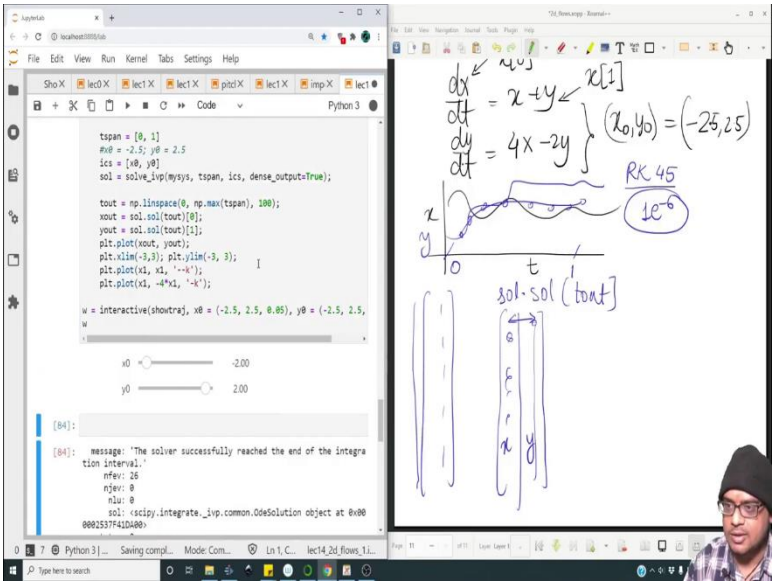


(Refer Slide Time: 45:27)



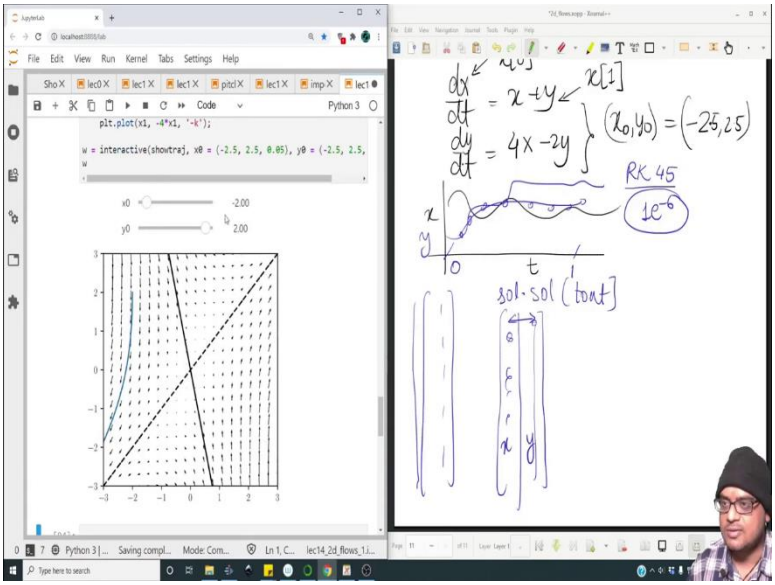
So, over here we have changing of the branch; let me also plot the eigenvalue, the eigenvectors, so that we can clearly see how the trajectory has changed.

(Refer Slide Time: 45:43)



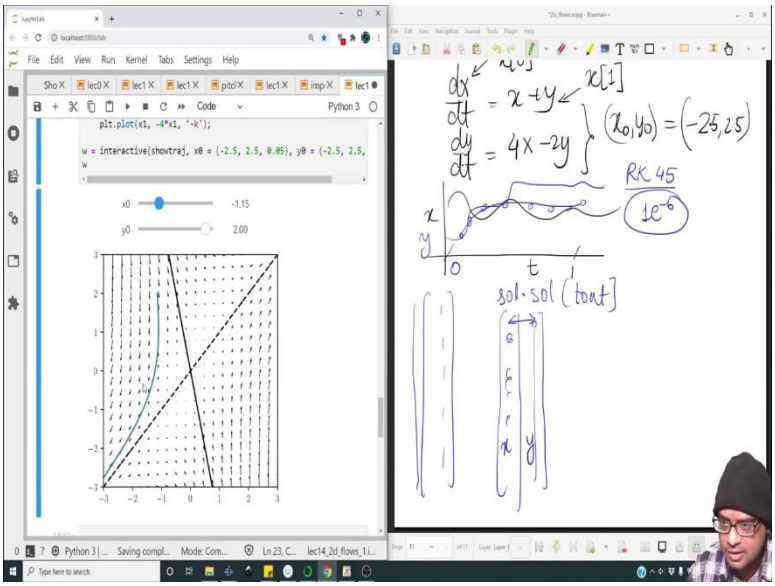
We are going to plot this as well; this has to be indented, ok.

(Refer Slide Time: 45:47)



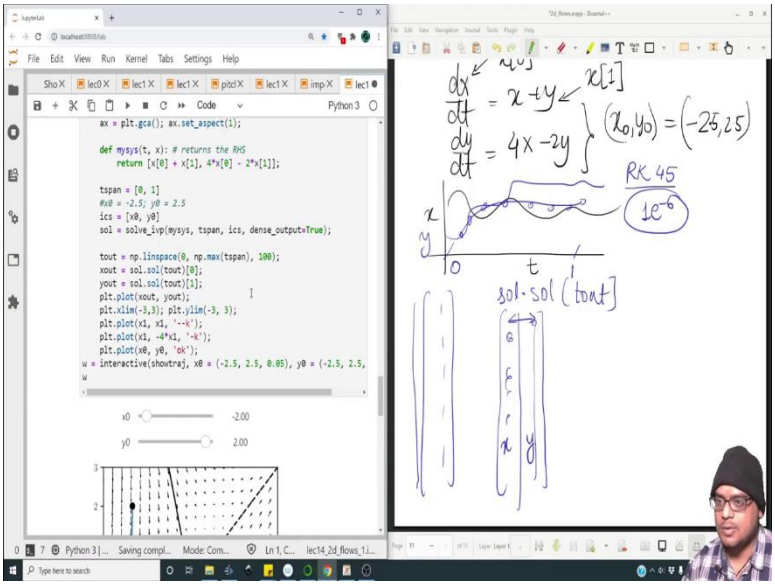
Let us run this.

(Refer Slide Time: 45:50)

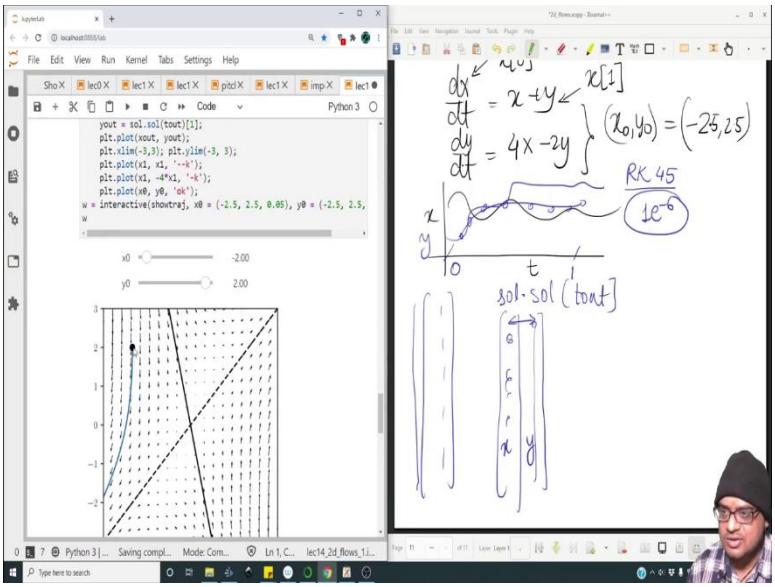


Let me change. So, you see how it is getting attracted; so, in order to show this better; let us plot the initial point by a small marker. So, plt.plot(x0, y0), ok.

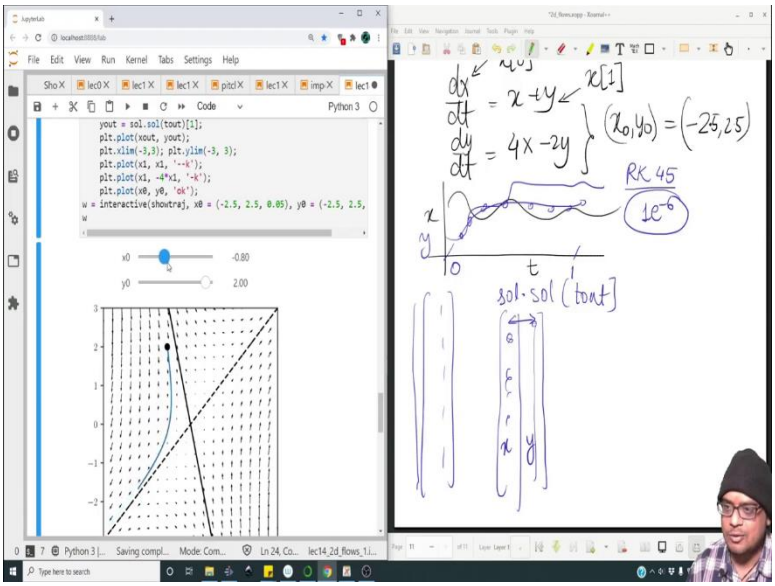
(Refer Slide Time: 46:02)



(Refer Slide Time: 46:10)

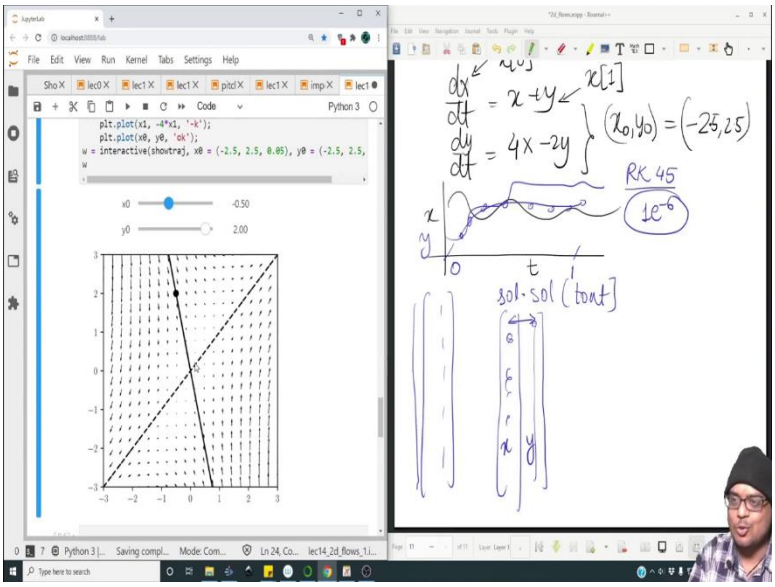


(Refer Slide Time: 46:13)



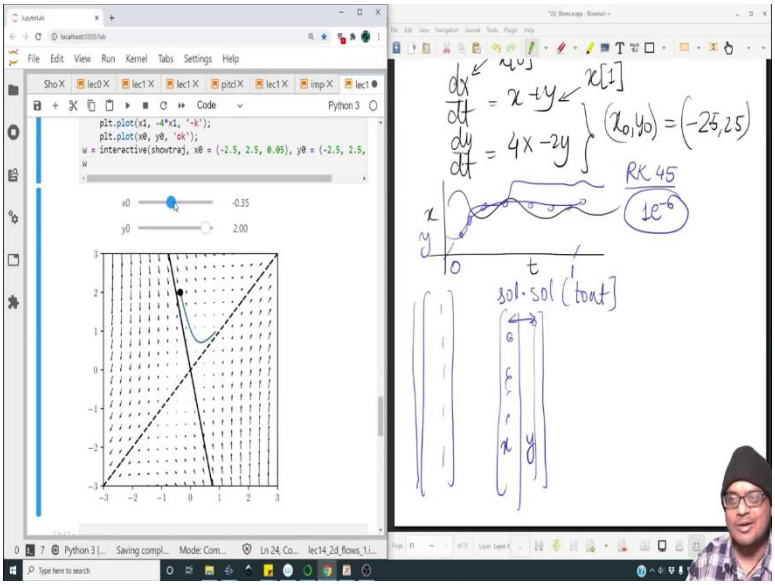
So, this is the initial point and this is how the trajectory evolves.

(Refer Slide Time: 46:18)



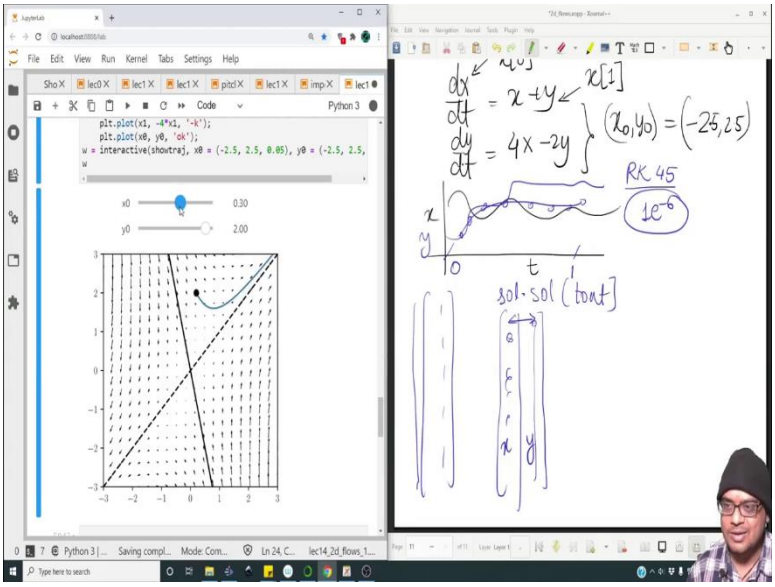
Once we cross that manifold. So, once this manifold, it will go towards the origin.

(Refer Slide Time: 46:25)

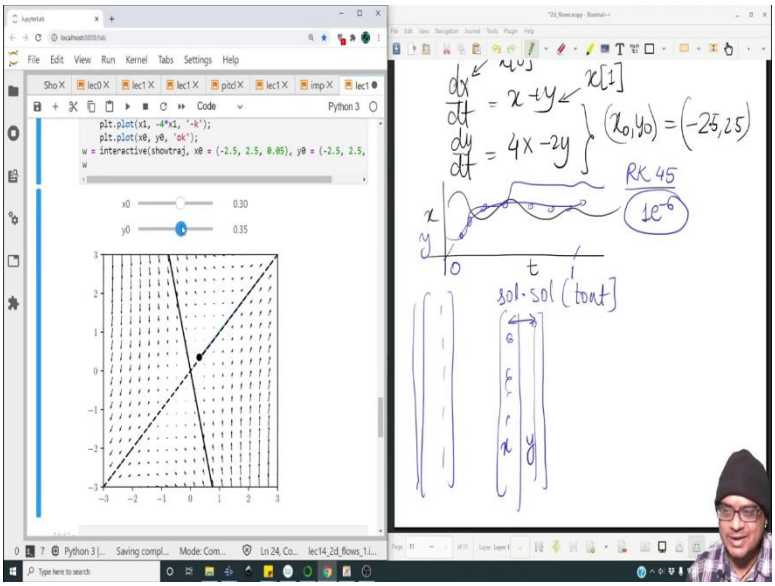


But once we cross it, it is now attracted towards the other unstable manifold; because as t tends to infinity, it is attracted towards the other direction, not the other direction. So, it is not going towards the unstable manifold along the positive direction, positive x and y direction, ok.

(Refer Slide Time: 46:41)

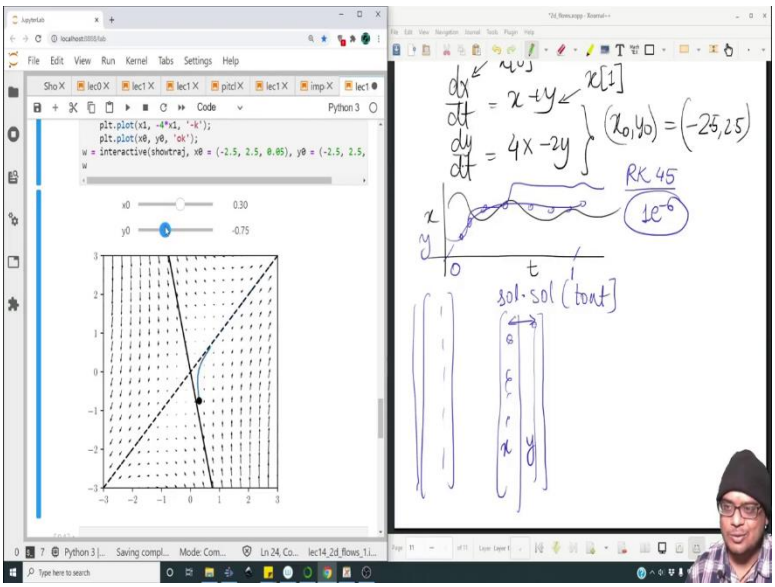


(Refer Slide Time: 46:44)



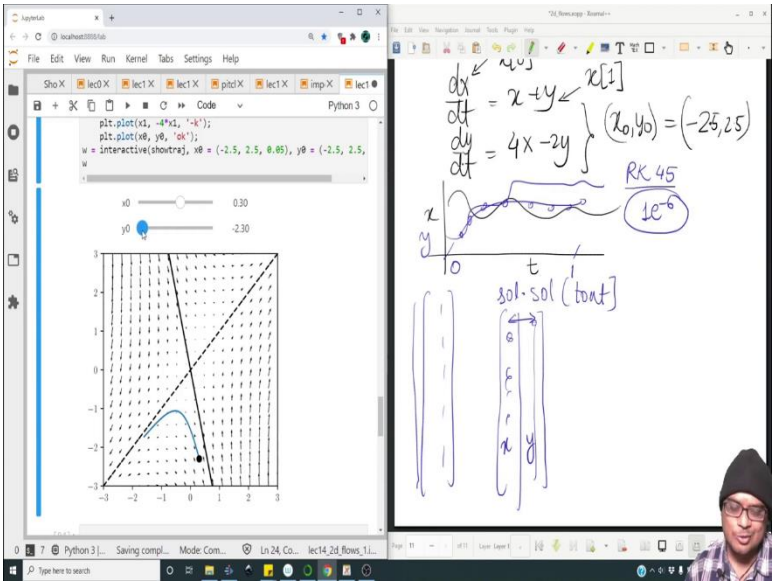
Let me now change y. So, it is. So, over here it is attracted towards plus infinity x and y going to plus infinity.

(Refer Slide Time: 46:57)



As we would do this; it is again attracted to that ok.

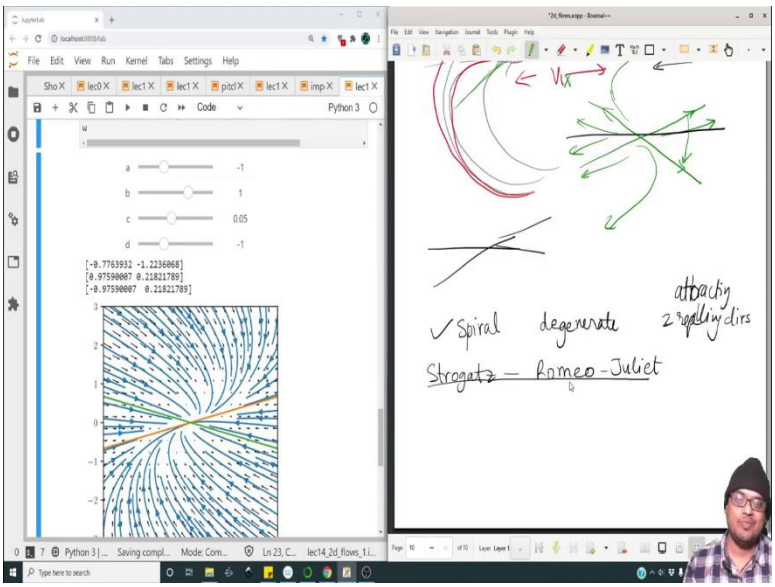
(Refer Slide Time: 47:02)



So, with the help of this, we can clearly figure out how trajectories in the phase space look like; this will be a very useful tool to analyze various problems of non-linear dynamics, how trajectories evolve and so on especially in 3 D.

So, it is worth spending some time and fooling around with various snippets; try to do various problems, try to do the assignments very carefully. So, with this I would like to end this particular lecture. I have given you a taste of how you can analyze various linear systems with the help of python and the assignment will be quite entertaining, in which you will actually solve a very famous problem proposed by Strogatz.

(Refer Slide Time: 47:48)



And it is called as the Romeo Juliet problem, where the level of attraction between Romeo and Juliet is modeled by means of this linear system, ok. Whether the case where if Romeo shows attraction towards Juliet; but Juliet does not show the same level of attraction towards Romeo, what will happen to that system.

Or if Romeo shows some attraction to Juliet and Juliet hates Romeo for it; then what happens to that system and so on. So, it is a very famous problem and it will help you really show insight into understanding what the physical meaning of such, of the evolution of such linear system means, ok.

So, with this I hope you learnt enough to analyze linear systems; because understanding this lecture and the previous lecture holds the key towards understanding whatever we are going to, we are going to we are going to study in the coming lecture that is non-linear systems, ok.

And this holds the key towards understanding non-linear systems; because obviously we are going to linearize a non-linear system in the vicinity of a point and try to gauge whether that point has these behaviors that we have discussed in this class. With this I will end this particular lecture and I will see you next time again, bye.