Advanced Aircraft Control Systems With MATLAB / Simulink

Prof. Prabhjeet Singh

Department of Aerospace Engineering

Indian Institute of Technology Kanpur

Lecture 56

MATLAB and SIMULINK implementation of aircraft dynamics

Hello friends, welcome back. Last week, we modeled three-degree-of-freedom equations of motion. So, let us simulate that in MATLAB, as well as in Simulink, and finally in the 3DOF block. First, we have to consider aircraft data. So, let me write the aircraft data. Consider a UAV which has a mass of 13.5 kg. I_y is given as 1.135 kilogram-meter square, area is 0.55 meter square, chord length is 0.19 meters. I am considering sea-level density, that is 1.225 kilogram per meter cube. Maximum thrust I'm assuming is 2 kg, and delta T will be kept constant at 0.5, that is 50% throttle is always applied. Then, the aerodynamic parameters:

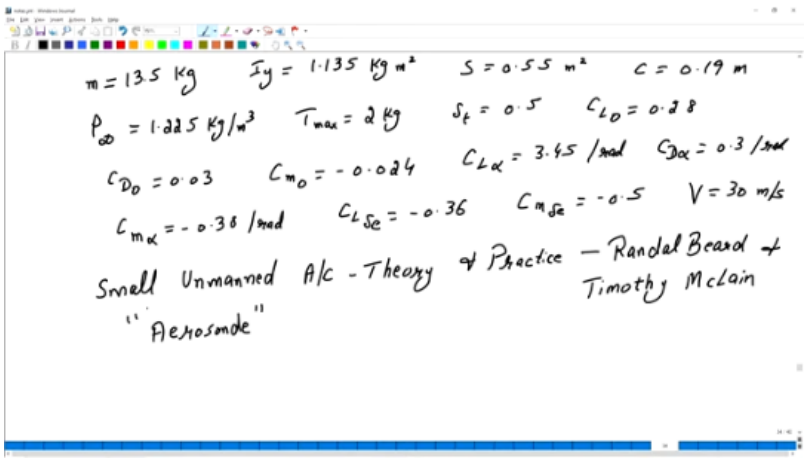
$$C_{L0} = 0.28, \quad C_{D0} = 0.03, \qquad C_{m0} = -0.024, \qquad C_{L\alpha} = 3.45\frac{rad}{s},$$

$$C_{D\alpha} = 0.3\, per\ rad, \qquad C_{m\alpha} = -0.38\, per\ rad, \quad C_{L\delta e} = -0.36$$

$$C_{m\delta e} = -0.5, \quad V = 30\ m/s$$

So, this aircraft data (UAV data) I have considered from a beautiful book named Small Unmanned Aircraft: Theory and Practice by Randal Beard and Timothy McLain. So, you can refer to this book for the details. And the UAV name is Aerosonde. So in the simulation, we will need an α_{trim} and $\delta_{e,trim}$. Now, what is trim? When do we say that an aircraft is in trim condition? The aircraft is in trim condition when the total forces and moments acting on the aircraft about the CG are in equilibrium. That is, it is at 0.

(Refer Slide Time: 03:35)



All the forces are acting. All the forces and moments about the CG are 0. So that means the pitching moment should be equal to 0. Our aim is next to find what is α trim and what is delta E trim given the aircraft conditions. So in trim, we know that we set the pitching moment equation to 0.

$$C_m = C_{m0} + C_{m\alpha}\alpha_{trim} + C_{m\delta e}\delta_{e,trim} = 0$$

So many students have doubts. Do we have any q dependencies over here? Obviously, no. For trim, we have rates equal to 0. All right, so let me find $\delta_{e,trim}$. From here, we can easily write it as

$$\delta_{e,trim} = \frac{-C_{m0} - C_{m\alpha}\alpha_{trim}}{C_{m\delta e}} \ldots Eq(20)$$

Now, the lift coefficient—we need the lift coefficient as well.

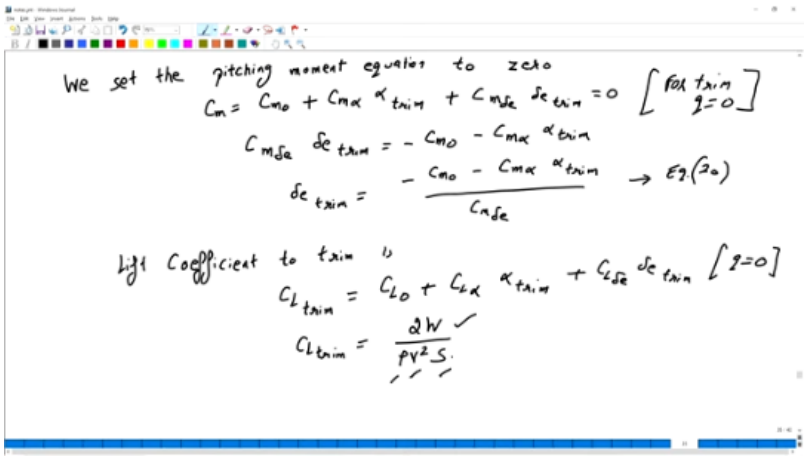
$$C_{L,trim} = C_{L0} + C_{L\alpha}\alpha_{trim} + C_{L\delta e}\delta_{e,trim}$$

$$C_{L,trim} = \frac{2W}{\rho V^2 S}$$

Again, w dependencies is 0. So, we know the weight of the aircraft, we know the density, we know at what altitude the aircraft is flying, we know the velocity, and we know the area. So, this $C_{L,trim}$ is actually known to us. So, we can write

$$\alpha_{trim} = \frac{C_{L,trim} - C_{L0} - C_{L\delta e}\delta_{e,trim}}{C_{L\alpha}} \ldots Eq(21)$$

(Refer Slide Time: 07:02)



We substitute equation 21 in equation 20 and solving we get

$$\delta_{e,trim} = -\frac{C_{m0}C_{L\alpha} - C_{m\alpha}C_{L,trim} + C_{m\alpha}C_{L0}}{C_{L\alpha}C_{m\delta e} - C_{m\alpha}C_{L\delta e}} \quad \dots Eq(22)$$

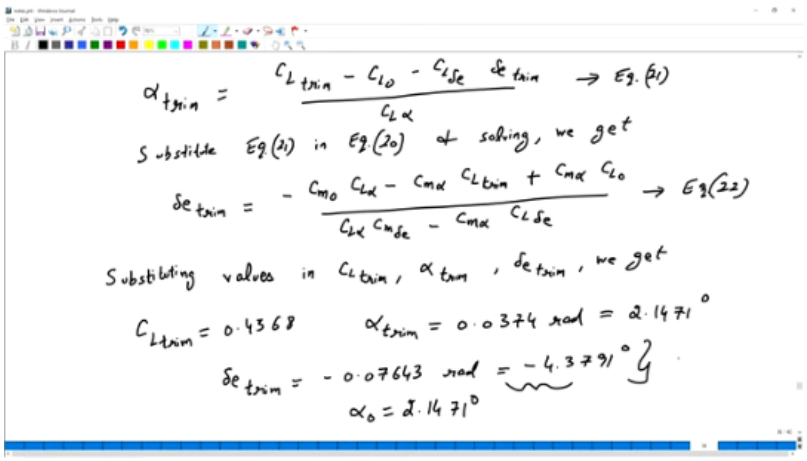
Since we already have the aircraft data available, substituting values in a $C_{L,trim}$, α_{trim}, $\delta_{e,trim}$, we get $C_{L,trim}$ value we get, please do it by yourself

$$C_{L,trim} = 0.4368, \quad \alpha_{trim} = 0.0374\, rad = 2.1471^{\circ}$$

$$\delta_{e,trim} = -0.07643\, rad = -4.3791^{\circ}\ , \quad \alpha_0 = 2.1471^{\circ}$$

so one thing you can note from here is it is having a negative $\delta_{e,trim}$ that is the elevator is deflected upwards so it needs elevator deflected upwards mean there is force is acting in the below direction in the tail that means it the nose will pitch up all right negative elevator deflection force is acting in the downward direction body z direction in at the tail that will make the nose pitch up all right next we also know that continuously α is changing How you can calculate α? So, let me draw the free body diagram here. So, we have body x-axis, all right.

(Refer Slide Time: 10:39)



This is where we have small u, the velocity in this direction, total velocity, all right. We have small w; this is the small angle, the attack angle. So, if I drop a perpendicular from here, this will be nothing but w. So, I can write

$$\tan\alpha = \frac{P}{B} = \frac{w}{u}$$

$$\alpha = \tan^{-1}\left(\frac{w}{u}\right)$$

And one more thing we know: when α is not equal to 0, then obviously w will also not be equal to 0. But when α is 0, that means velocity is acting directly in the forward direction, the body x-direction, then w will be 0. So, always remember that there is a one-to-one relation between α and w.That is why in many textbooks, we write in place of w; we can replace it as α as well, all right. Now, if you resolve this component of velocity, so in this direction, we have v cos α, and here we have v sin α. So, we need some initial conditions to run our model, right? So, we have initial conditions. I am denoting it as 0 in the subscript. So,

$$u_0 = V\cos\alpha$$

$$w_0 = V\sin\alpha$$

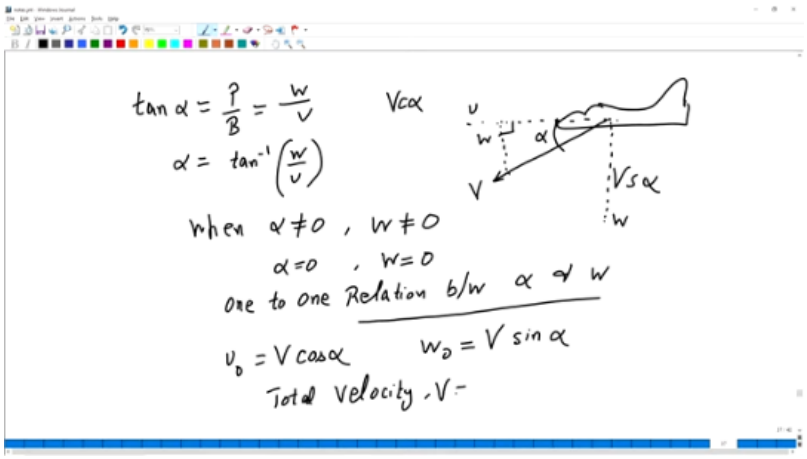
And total velocity, we know it is written as which is capital V

$$V = \sqrt{u^2 + v^2 + w^2}$$

So, for this case, we have assumed this is 0, so we finally have u square plus w square. So, these relations we will actually be using in our simulation model, all right. Now, also

to make things simpler, consider initial pitch angle that is θ naught equals to same as attack angle, 2.1471 degrees, since we are assuming here flight path angle, that is γ.

(Refer Slide Time: 13:48)



γ is zero. We know that the pitch angle is given as θ equals to α plus γ when beta or side slip or side velocity is zero. All right, so in this simulation, we are assuming γ to be zero. It is not climbing, so then we can assume

$$\theta_0 = \alpha_0$$

The other initial conditions we will assume are

$$q_0 = x_0 = z_0 = 0$$

We had written equations, right. So we need to assume a few initial conditions, so that I am assuming this. I am assuming them to be 0. Now, more or less, we are ready with all the parameters required to run the simulation. Let us switch to MATLAB. So, let me just rerun this now. Now, these are the initial conditions that I have just explained. u0 is v into cos of angle. So, this is the method to convert directly from degrees to radians. So, we got α trim as 2.147 degrees. So, this is the initial condition for u_0. u_0 is v 30 into cos d initial forward velocity. It is converting it to radians. Similarly, W naught is V sin α sin d 2.1471 degrees initial vertical velocity. Θ naught initial condition is again the same as the attack angle. So, I am converting this from degrees to radians. I'm letting MATLAB know that this is in degrees. So, you need to convert it to radians. This is the initial pitch angle. And q_0, x_0, and z_0, I've taken them as 0.

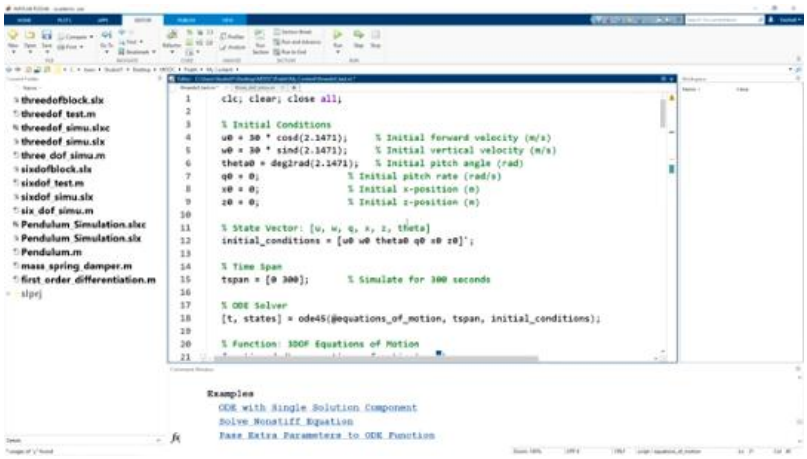
And I've considered these initial conditions in a state vector, in vector form, which is represented in this form. You can also add a comma here if you want. It is completely up

to you. So, these are the initial conditions: $u_0, w_0, \theta_0, q_0, x_0, z_0$. I have written them in a vector format, and the time span I am giving is 0 to 300 seconds, simulating it for 300 seconds, all right. Now, we need an ODE solver where these are the outputs: t comma states. t is the time, and the states are the 12 states that we have written before. I am using ode45, an ordinary differential equation solver, which basically uses the Runge-Kutta method, widely used for simulating any equations of motion. This is the function file: equations of motion. The time span is 0 to 300 seconds, and these are the initial conditions. You can have a look at this format syntax: help ode45. So, this is a non-stiff differential equation, a medium-order method.

This matlab function where t span is initial condition is t naught, final time is t f, integrates the system of the differential equations, y dash equals to f of t comma y, from initial condition t naught to final time t f with initial conditions y naught. All right, so the syntax for this is time comma states, that is y, ode 45, the function file that you have written, comma the time span, comma the initial conditions. There are other ODEs also available, ODE 23, ODE 78, ODE 89. I encourage you to use these as well and observe the differences. In this course, we'll have used ODE 45. This is the syntax, ODE 45, added equations of motion, T span, initial conditions. And when MATLAB runs this line command 18, it searches for equations of motion. So either it searches within the MATLAB code or it searches within the folder. So I have provided it within the code.

So here we have the function file equations of motion. Remember the name should be exact. Otherwise, it will not match and it will throw you an error. This is the output vector output dy dt. I mentioned it as dy dt. The function file is equations of motion, and the input is t comma y. You can write t here as well. No issues. But it will show that since we have not used t here, the input argument might be unused. Consider replacing it with a tilde. Hence, I have replaced the derivative with a tilde. And the input is, of course, y. All right. Hope it is clear till here. Then I have added the UAV parameters: mass 13.5 kg, moment of inertia about the y-axis in kilogram meters squared. These values I have already indicated earlier. I have just written them here.

(Refer Slide Time: 20:04)



So, thrust should be in newtons, 2 into 9.81, and I have considered δ_t equals 0.5, that is, 50 percent of throttle is always applied throughout the simulation. All right. Next, the aerodynamic coefficients: C_{L0} 0.28, $C_{L\alpha}$ 3.45, $C_{L\delta e}$, C_{D0}, $C_{D\alpha}$, C_{m0}, $C_{m\alpha}$, and so on and so forth. Now, after coming to this, for example, line number 45, we need to unpack the states. We need to let MATLAB know that that the 12 states what are those 12 states the first state is y of 1 is nothing but the forward velocity u so here we have the second state belongs to w third state belongs to θ similarly 4 to q fifth to x 6 to z here it is saying we have not used it so this is which is not required as well then we have written total velocity and attack angle so v equals to square root of u square plus w square α equals to a tan 2 w comma u i encourage you to find out why i have used here here a tan 2 i could have simply written a tan w comma u it is related to some coordinate systems so i encourage you to explore this term a tan 2 as well and finally delta e we calculated around 4 point minus 4.3791 degrees Immediately, I have converted into radians, into pi by 180. And these are the aerodynamic coefficients, $C_{L0} + C_{L\alpha}\alpha$, $C_{L\delta e}\delta_e$, C_D, $C_{D0} + C_{D\alpha}\alpha$, and then $C_m$is $C_{m0} + C_{m\alpha}\alpha + C_{m\delta e}\delta_e$. Aerodynamic forces I have mentioned before,

$$L = \frac{1}{2}\rho V^2 S C_L$$

$$D = \frac{1}{2}\rho V^2 S C_D$$

Then the pitching moment,

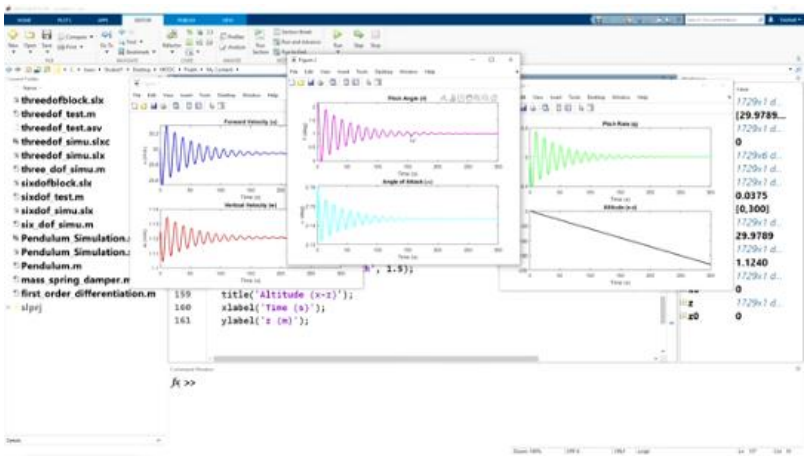
$$m_y = \frac{1}{2}\rho V^2 Sc\ C_m$$

So, these are the equations of motion u dot w dot θ dot q dot x dot and z dot and the final output vector is dy dt. I have written here output vector was dy dt. So, I am letting MATLAB know. So, these should be the outputs from this function file. Now finally we we need to run this simulation and for plotting we need to tell matlab again that u belongs to the first column vector second column is w and so and so forth since now we are out of this function file we need to redefine these variables Similarly, α as well. A equals a tan 2 w comma u. Now, finally, we are ready with the simulation. So, let me put a breakpoint here. Now, I am running the code. Let us see if everything is working fine. And yes, you can see here in the workspace, we have all the variables: α, u, w, θ, q, x, and z. And these are the states.

1, 2, 3, 4, 5, 6. So, we are letting MATLAB know that the first column belongs to u, the second column belongs to w, and so on and so forth. Hence, we have written it in this way. Then finally, I'm removing this breakpoint here. In Figure 1, I'm plotting t comma u. Since I've already run it, that's why it was showing me the values of u; you can see here.

In the second subplot is t comma w. In Figure 2, I plotted time with θ and time with α. Figure three has been plotted with q and z. So, I've written here minus z because in our equations, the z-axis is pointing downward, that is, toward the center of the Earth. So, we need to see whether the aircraft is increasing or losing altitude. That's why we have written here minus. Now, finally, let me run this code. Continue. Now, let us see what we have got. So, this is Figure 1. In Figure 1, we have u and w. So, with the α_{trim} and $\delta_{e,trim}$, the u velocity remains still over there. It is constant at around 30 meters per second. w is around 1.12 meters per second. Θ is approximately 1 degrees and α is 2.14 degrees. Q is approximately going to 0. So, we have beautiful results over here and altitude is reducing actually in 300 seconds up to about minus 200 meters. All right.

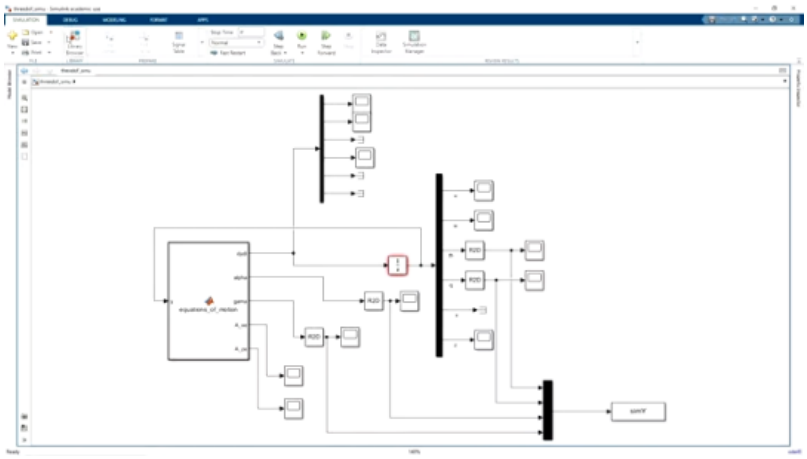
(Refer Slide Time: 25:31)



Now, let us simulate same in Simulink as well using the MATLAB Simulink block. all right so in the last lecture i have explained about this matlab function so in matlab function everything remains same so we have output vector and it is up to me what other outputs i want from here i want output α i want output γ Similarly, a x e accelerations in x e direction similarly a z e all right everything remains same. So, I am not going to write this code from scratch since I have already explained in the last week how to do this and see here we are actually integrating these equations. I have written initial conditions y naught and it is giving me a vector u, w, θ, q, x and z. All right. Then I am also getting output vector α, γ, A_{xe} and A_{ze} . All right. And I have also used the two workspace that we have used it before. So it will send the data to the workspace in directly in MATLAB. Further, I have also plotted these curves. That is the vector u dot w dot directly before integration.

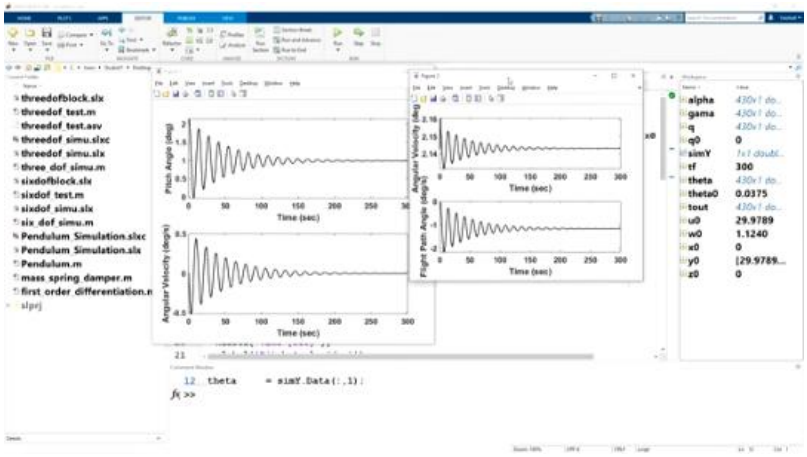
All right. And to run this, to run this Simulink model, I have written another MATLAB code. This is the one. So everything remains same over here. Except I'm calling simulink sim simulate 3dof simu dot slx. So this is the file name. 3dof simu. All right. Let me see if it is running well or if there is any error. I'm putting a breakpoint over here.

(Refer Slide Time: 27:54)



And click on run. So it seems it is working fine. So let us continue plotting. So I have extracted θ, q, α and γ from that two workspace model and I'm plotting it here. The format remains same. I'm clicking on continue. So we have pitch angle and angular velocity. And similarly, we have angular. All right. We have θ pitch angle, this is q angular velocity, this is α, this is attack angle in degrees.

(Refer Slide Time: 28:34)

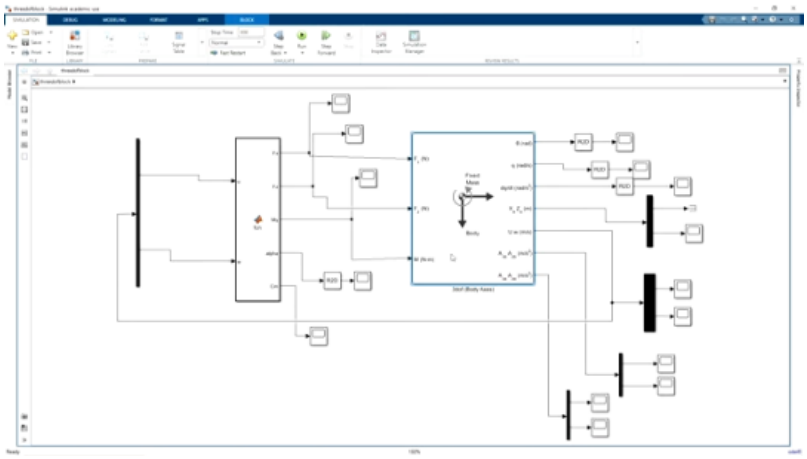


You can directly convert it to degrees. Currently, it is showing in α, or let me check it. Yeah. So it is already converted into degrees. So that's fine. This is attack angle, flight path angle. All right. So here we have θ pitch angle, q angular velocity, attack angle in degrees, γ flight path angle, which is in degrees, not degrees per second. Right. Let me rerun it. Now it's OK. So we get similar results as per the previous MATLAB code. Now we can also check it here in the code as well. If you see here, this is the u scope we're getting over here. It is a trim value of approximately 230 meters per second.

Let me close it. Similarly, w. θ is approximately 1 degree. q should go to 0 for the trim value. Yes, it is going to 0. So here we have the pitch angle θ. You can confirm it from here. Yes. All right. Now, next, let us check γ. We have entered γ here, right? So, θ is the total angle of attack plus γ. So, let us see what γ is over here. It is approximately minus 1 degree. And what is α over here? It is coming to around 2.145 degrees. So, the total angle would be around 1 degree. All right. Now, we are done with this block of the MATLAB function. Initially, we thought of using this block directly.

The 3-DOF block that I have shown you earlier. Let me open it again. So, this is the 3-DOF block. So, instead of writing a separate MATLAB code or using the MATLAB function block, you can use it directly. So, if you notice it carefully, we need three inputs: forces in the x direction, forces in the z direction, and then the moments, which is newtons per meter. The output is θ, q acceleration, dq by dt, x_e, z_e, u, w—that is, small u and small w.

(Refer Slide Time: 31:08)

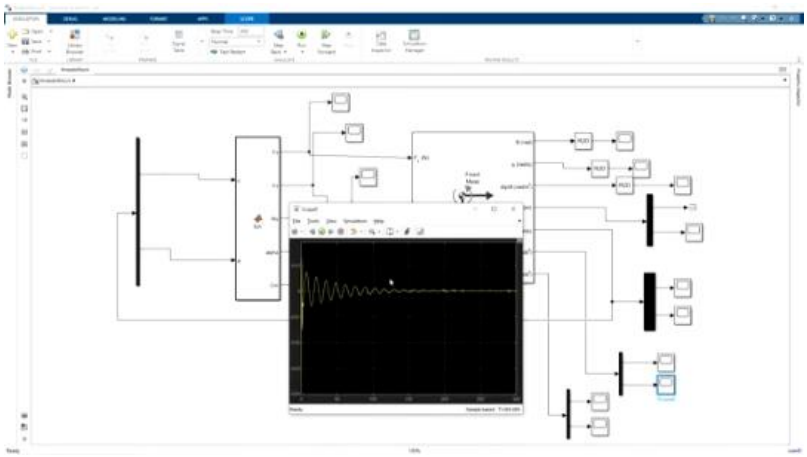


Acceleration in body axis axb, azb, similarly axc towards the with respect to inertial reference frame, all right. Now, I can directly run this code and see. Let me close these figures first, all right. Now, let me run this code directly. Now, before running, let me show you what I have here. I need to give this an input vector: forces in x and z direction. So, I have written output here as fx and fz, moment about y-axis. It is up to me, and similarly cm and inputs. It only needs now u and w. So, we need not write the complete code over here. So, we need only fx and fz so that we can give that information directly to the block. Now, the equation that we have written here, After these, so these are already embedded into the block.

So, need not write those equations here, which is unnecessary. So, we need fx and fz, and we also need moments about y-axis. Hence, I have written the output vector here as fx, fz, and my. The number of outputs that you write over here will directly appear over here. fx, fz, my, α, and cm. So, this signal is going to fx, fz is going to fz, and fmy is going to the moments m. All right, I hope it is clear. I've converted directly to radians to degrees and all these to the desired units as we want. All right, now we are ready to run this code. Yes, so it is showing over here ready. So, let me show here what is θ over here. So, θ is coming around 1 degrees. Let me go to the previous block. The θ was approximately 1 degrees. So, we got the same. Similarly, α. A is the trim values around 2.145 degrees. So, let me go to the block. So, here we have approximately 2.14 degrees here as well. So, you can compare this. So, this

In this block, the equations are already inbuilt. So you need not to write the equations and integrate separately. All right. So you can use this block directly if you have the aircraft model. Similarly, you can compare the other parameters as well. For example, A_{xb}. which is approximately 0 accelerations. You go to the previous block. So, here we have accelerations. It is going approximately equals to 0. This is nothing but u dot. All right. The first vector for this dy by dt is $\dot{u}$. This is before integration. After integration, obviously, will be u. Similarly, this is $\dot{w}$. all right. It is approximately going to 0. So, in this way you can actually compare all the plots whether are you getting a similar plots or not. So, these are different methodologies that you can use to simulate 3 degree of freedom equations of motion. I hope now you know how to simulate the 3 degree of freedom equations of motion.

(Refer Slide Time: 34:45)



In the next lecture, we will begin with the simulation of 6-degree-of-freedom equations of motion. Thank you.