

16.07 Flight Simulation Laboratory

Lab I

Issued: Sep 07, 2004

Due: Sep 24, 2004

In this laboratory you will develop a two-degrees-of-freedom (2DOF) model to simulate the motion of an aircraft. You will code up your model in MATLAB, test different flight profiles, and visualize the flight trajectories.

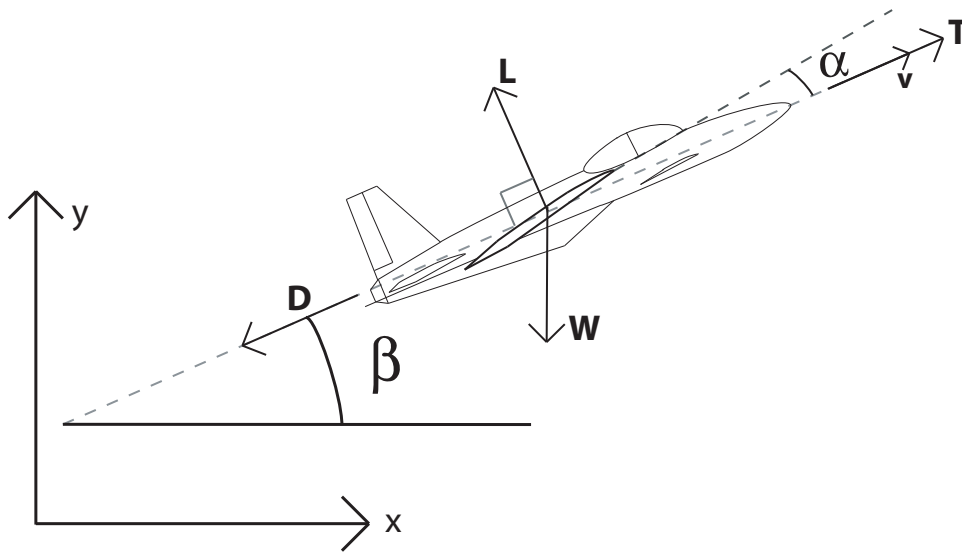


Figure 1: Aircraft and coordinate systems

In the model shown in Figure 1 the two degrees of freedom are the horizontal and the vertical positions of the center of mass of the aircraft, x and y . When we write this model in state space form as explained in lecture D3, we require four state variables: horizontal position x , vertical position y , horizontal speed v_x , and vertical speed v_y . These can be grouped into a single vector of variables \mathbf{X} :

$$\mathbf{X} = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} \quad (1)$$

We will consider a simple model with only two inputs: the angle of attack α and the thrust T . The thrust T is assumed to be always aligned with the flight direction (in the notation of lecture D3, we are assuming that $\alpha_T = -\alpha$). This is not usually the case in the real situation, but it could be achieved using thrust-vectoring by an appropriate control system. Note that the flight direction β is not an input nor a degree of freedom because you can determine it from the state vector. The equations of motion can be written as:

$$\dot{\mathbf{X}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{v}_x \\ \dot{v}_y \end{bmatrix} = \mathbf{F}(\mathbf{X}, \alpha(t), T(t)), \quad \mathbf{X}(t=0) = \mathbf{X}_0, \quad (2)$$

where t is the time and \mathbf{X}_0 is the initial state of the system.

1. Using Newton's second law, find the explicit form of the vector \mathbf{F} . You can assume that $C_L = K_1\alpha$ and $C_D = C_{D0} + C_L^2/(\pi\mathcal{R}e)$, where α is the angle of attack. The constant K_1 , the wing aspect ratio \mathcal{R} , the air density ρ , the wing planform area S , the wingspan b , the efficiency factor e , and the aircraft mass m are also given, and assumed to be constant for simplicity. In addition, we will assume that there is no friction with the ground when the aircraft is taking off or landing.
2. Code the model equation (2) into a MATLAB function. As you will have to integrate this equation, consult the help about the ODE integrator `ode45` in MATLAB on how to write your function¹. You should also consult the spring-mass example which is available together with this document on the 16.07 website. Use the same `odeset` options that are used in the spring-mass example. Note that the example handles time-varying frequency; in your simulation, the angle of attack α and the thrust T will depend explicitly on the time t in Part 4, so make sure your code can handle time-dependent α and T .

Note: If you find the need to use inverse trigonometric functions, use the MATLAB function `atan2` in place of the usual arctangent `atan`. `atan2` is a "four-quadrant" inverse tangent; $-\pi \leq \text{atan2}(y,x) \leq \pi$. The use of `atan2` will be needed in Lab 2.

¹Use MATLAB Help or <http://www.mathworks.com/access/helpdesk/help/techdoc/ref/ode45.html>

3. Test your code with the set of *constant* inputs² in Table 1, based loosely upon the Russian Federation Sukhoi Su-35³.

Table 1: Constant test inputs for Part 3

$x(t = 0) = 0$ m	$y(t = 0) = 0$ m	$v_x(t = 0) = 0.001$ m/s	$v_y(t = 0) = 0$ m/s
$b = 15.16$ m	$S = 65.66$ m ²	$e = 0.8$	$\rho = 1.225$ kg/m ³
$g = 9.81$ m/s ²	$C_{D0} = .06$	$m = 27250$ kg	$K_1 = 5$
	$T = 170000$ N	$\alpha = .04$ rad	$t_{end} = 300$ s

Does the airplane fall below the ground surface? Modify your code to simulate the effect of the ground in such a way that it will prevent that from happening. Compare the trajectory you obtain with our solution in Figure 2.

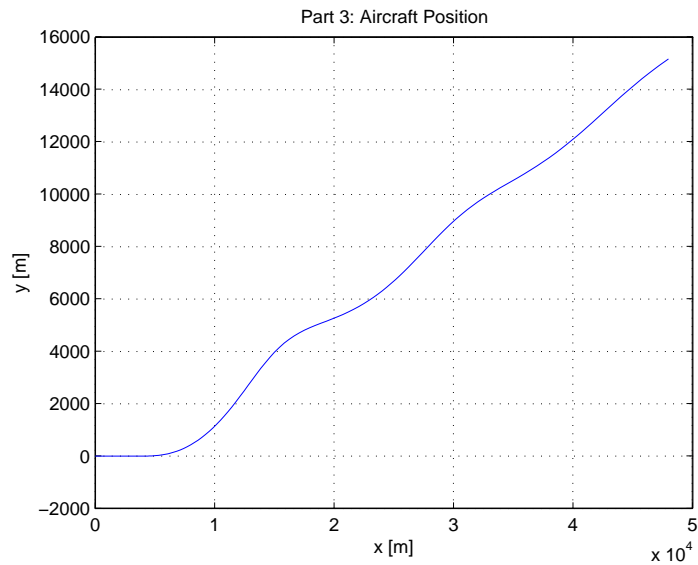


Figure 2: Solution for part 3

² $v_x(0)$ is given a very small positive initial value in order to avoid dealing with singularities of the type $\frac{0}{0}$

³Source: *Jane's All The World's Aircraft 2004-2005*

4. Find a set of *time-varying* inputs α and T such that the airplane takes off from the runway, climbs to a certain altitude, and continues on a steady-state level flight. Table 2 contains the constraints which must be satisfied.

Table 2: Constraints for Part 4

Maximum runway length	$L = 2000$ m
Final altitude	$H = 10000 \pm 250$ m
Maximum climb time	$t_c = 8$ min
Maximum simulation time	$t_{end} = 16$ min
Maximum thrust	$T_{max} = 196043$ N
Maximum angle of attack	$\alpha = 8^\circ$

Explain how you found α and T . Hand in four plots : 1) angle of attack α vs. time, 2) Thrust T vs. time, 3) aircraft y vs. x , and 4) aircraft v_y vs. v_x (this is called a hodograph). There are multiple solutions to this problem; all of them will receive full credit as long as all conditions given in Table 2 are satisfied.

Hint: Do *not* attempt to compute the length of the takeoff roll or the time to climb on paper; there are no analytical solutions! Use your intuition and try different values until you are satisfied with your results. Conversely, you can do a force balance by hand to determine the steady-state flight conditions. The climb section needs special care due to the presence of the lightly damped phugoid mode. You may find that you need to hold one or more inputs or variables fixed to accomplish Part 4; once again, use your intuition.

5. For this part, modify your code so that the thrust always cancels drag exactly (i.e. first compute the drag and set the thrust equal to it). Calculate the angle of attack necessary to maintain steady level flight at a speed of 170 m/s, and set α to that value. Start a simulation at an altitude of 10000 m and a horizontal velocity of 185 m/s without changing α ; run the simulation for 500 seconds. Plot aircraft y vs. x position and describe what you observe. This is undamped phugoid motion.

You must turn in electronic versions of your MATLAB code for Part 4 **only** on the **server** (do not turn in code for Part 3 or Part 5); no

paper copies of the code will be accepted and the **server** will not accept electronic turn-ins after the due date and time. Your code must produce all figures requested in Part 4 in a single run without any manual changes in it, and all plots should be titled, with all axes labelled, and with a grid (consult the spring-mass example). Calculations, derivations and all plots must be turned in on paper by the due date.