16.07 Flight Simulation Laboratory

Assignment II Solution

Issued:

In this laboratory exercise you were asked to investigate the dynamics of an aircraft flying a loop in the vertical plane. Refer to the Assignment II handout for the entire problem text.

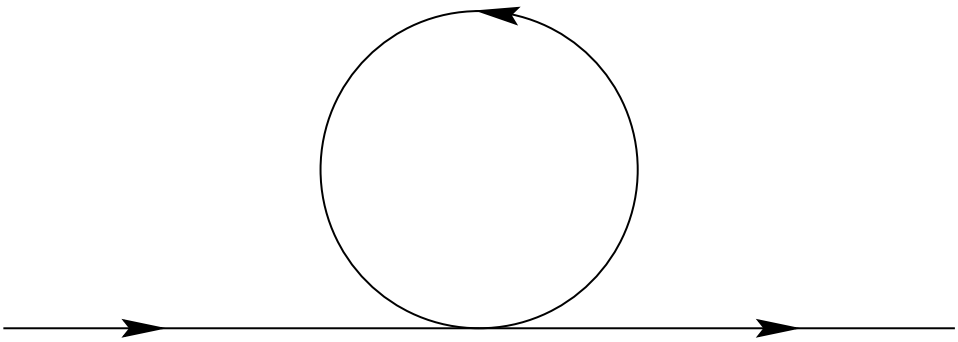

Figure 1: Flight path containing an ideal circular loop

In addition to the assumptions put forth in Assignment I, we are assuming that thrust is equal to drag at all times. It is simple to keep the drag term in your state derivatives routine and set the thrust equal to drag; it also simple to remove the thrust and drag terms from your code entirely. If you took the latter option, the acceleration equations would change as follows:

$$\dot{v_x} = \frac{1}{m}(-Lsin\beta) \tag{1}$$

$$\dot{v_y} = \frac{1}{m}(Lcos\beta) - g \tag{2}$$

1. *"Determine an expression for the lift and angle of attack required to fly the circular loop."*

   The key here is realizing that a circular loop has a constant radius of curvature. Don't get confused trying to think about the aircraft's pitch angle, stalls, or anything like that; in this simple model, if you set α to give you the proper balance between normal acceleration and aircraft velocity, the loop will occur automatically. So what is the normal acceleration? Remember that when a particle is traveling in a circular path:

   $$a_n = v^2/r \tag{3}$$

   Here's what you have to know to complete this problem:

- By Newton's 2nd law, accelerations come from a force imbalance
- By the assumptions of our model, the aircraft always points along the direction of travel
- Because the aircraft always points along the direction of travel, lift is always perpendicular to the direction of travel

Of course, the aircraft's weight always acts straight toward the ground. With these points in mind, we can write an equation for the normal acceleration needed to fly the loop.

$$a_n = \frac{L - W cos\theta}{m} \tag{4}$$

With some rearranging...

$$L = ma_n + W cos\theta = m\frac{v^2}{r} + W cos\theta \tag{5}$$

... we can write lift in terms of the aircraft state vector:

$$L = \frac{m(v_x^2 + v_y^2)}{r} + \frac{mgv_x}{\sqrt{v_x^2 + v_y^2}} \tag{6}$$

Knowing also...

$$L = \frac{1}{2}\rho v^2 S C_L = \frac{1}{2}\rho v^2 S K_1 \alpha \tag{7}$$

... we arrive at an expression for α:

$$\alpha = \frac{L}{\frac{1}{2}\rho v^2 S K_1} \tag{8}$$

The graders will not require you to write α entirely in terms of the aircraft state variables so long as you write lift in terms of the aircraft state variables.

2. *"Modify your code from Lab I so that the aircraft . . . goes into a perfectly circular loop . . ."*

   To complete exactly one loop, you will need some `if` statements and some trial-and-error. One way to start is to insert code into the state derivatives routine that changes α to the value needed for the loop at 20 seconds (note that this code should come after the steady-state α has been set and before the computation of C_L is made)...

```
if  (t  >  20)
   alpha  =  (m*vel^2/r  +  m*g*x_dot/vel)/(q*S*K1);
end
```

... and then making a plot of aircraft altitude vs. time in your main routine. With this plot, you can determine approximately when the aircraft comes back to its starting height. In the case of our model aircraft, the time is near 66.1 seconds. Armed with this information, we can write the complete section of code for the circular loop:

```
if  (t  >  20)  &&  (t  <  66.1)
   alpha  =  (m*vel^2/r  +  m*g*x_dot/vel)/(q*S*K1);
end
```

It's important to know that we never use the MATLAB comparison operator `==` in our code. Due to round-off errors, decimal numbers on a computer will virtually never equal one specific value. You must use `<` and `>` for reliable code.

3. *"By trial and error, find the constant value of the angle of attack α which will make the aircraft reach the same maximum altitude . . ."*

   Start a new state derivatives routine for this section of the assignment. The code will be very similar to the routine for the circular loop; in fact, it will be identical except for the section where α is set to the value needed to fly this new non-circular loop. Play around with the α until you get a loop that takes the aircraft to about 1600m above its starting height (circular loop diameter). Note that the time to fly the loop will be different, so you'll have to change the ending time in your `if` statement for the loop. After some experimentation, these values work (recall that α is in radians):

```
if  (t  >  20)  &&  (t  <  59.62)
   alpha  =  0.119;
end
```

4. *"For both loops, turn in a plot . . ."*

   See the solution code for help on how to make these plots using the `subplot` and `axis` commands that MATLAB offers. The `derivative` routine was written "in-house"; it takes a column vector and an associated time vector and returns a column vector of d/dt. This is convenient for us because the MATLAB function `ode45` gives its results as a matrix with the value of the four aircraft state variables x, y, v_x, and v_y extending downward in their own columns.

To decompose the aircraft acceleration into normal and tangential components, we must convert the acceleration from the inertial frame into the aircraft body frame. The coordinate transformation for this is:

$$\mathbf{a}_b = \begin{bmatrix} cos(\theta) & sin(\theta) \\ -sin(\theta) & cos(\theta) \end{bmatrix} \mathbf{a}_i \tag{9}$$

$\mathbf{a}_i$, and $\mathbf{a}_b$ are the aircraft acceleration in the inertial and body frames, respectively. θ is the aircraft flight path direction, found using an arctangent on the velocity in the inertial frame:

$$\theta = tan^{-1}(v_y/v_x) \tag{10}$$

Note that in your code, you want to use the four-quadrant arctangent `atan2(`v_y`,`v_x`)`. All of this looks like:

```
accelvi  =  [accelx,  accely];
thetas   =  atan2(X(:,2),  X(:,1));

accelvb  =  zeros(length(t),2);
for  conv  =  1:length(t)
    %  direction  cosine  matrix
    T  =  [   cos(thetas(conv))   sin(thetas(conv))
          -sin(thetas(conv))   cos(thetas(conv))  ];
    accelvb(conv,:)  =  transpose(T*transpose(accelvi(conv,:)));
end
```

The MATLAB `transpose` functions are needed because `accelvi` and `accelvb` are arrays of row vectors and the matrix multiplication needs a column vector. This is just one way to implement the code. `accelvi` will now store the tangential components of acceleration a_t in its first column and the normal components of acceleration a_n in its second column. The radius of curvature plot can now also be made. Recall:

$$\rho = v^2/a_t \tag{11}$$

5. *". . . which of the two flight paths is preferable?"*

   It can be seen from the plots requested for the two loops that the circular loop requires larger maximum acceleration and therefore puts more severe structural requirements on the aircraft, hence the non-circular loop is preferable (the same result is obtained without simulations in the examples in lecture D6).