Helical Turns: Part 3, Angle of Attack and Elevator Trim Models

W. Premerlani, and Peter Hollands, May 1, 2015

This document is part three of a three part series describing the theory and implementation of helical turn controls. It describes the theory and implementation of angle of attack and elevator trim models.

Up until now, the MatrixPilot computations assumed that the angle of attack is zero and that the errors due to a non zero angle of attack are small. That must have been approximately true, because MatrixPilot did a reasonably good job of controlling pitch and altitude. However, as improvements were made in other areas, we suspected that we would eventually want to account for angle of attack, so here we are.

The theory says that angle of attack should be approximately a linear function of wing loading divided by the square of the airspeed. We made measurements with real aircraft as well as with hardware in the loop simulation (HILSIM), and verified that to be the case. A typical plot of angle of attack as a function of relative wing loading is shown at the top of Figure 1, where relative wing loading is equal to the wing g loading times the square of the ratio of cruise air speed divided by actual airspeed. The data is taken from a HILSIM flight and processed by Peter Holland's flight analysis program.

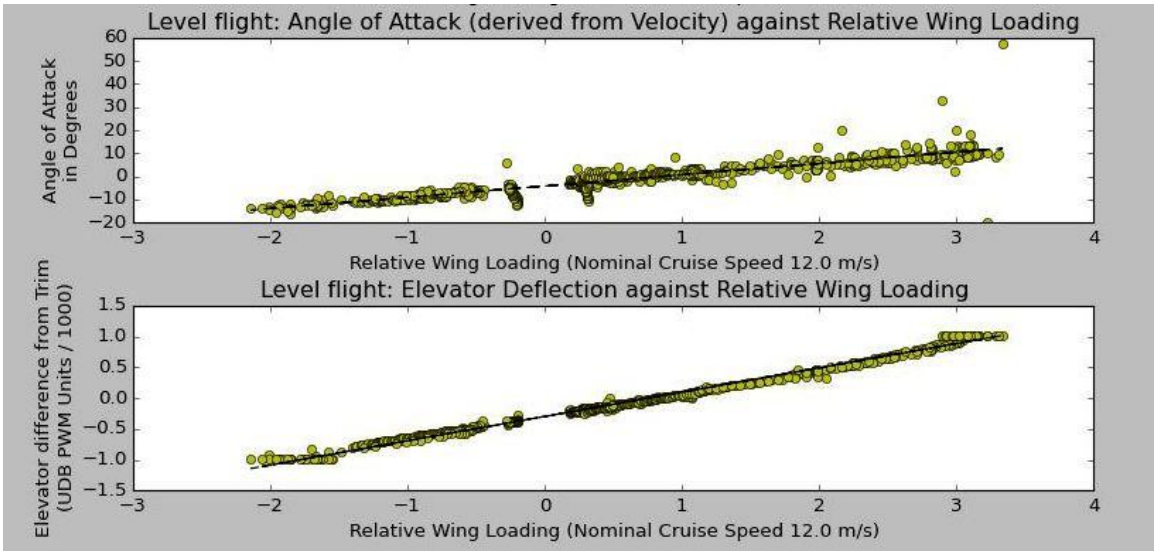

Figure 1 - Angle of attack and elevator trim as a function of relative wing loading

Note that in addition to a plot of angle of attack as a function of relative wing loading, Figure 1 also has a plot of elevator trim as a function of relative wing loading. The reason for showing both plots is that early in our work to implement an angle of attack model, we discovered that an elevator trim model is equally important. We started with an angle of attack model. We manually estimated slope and intercept from flight data, and used them to estimate the parameters in our implementation of a real time estimate of angle of attack, and included that in pitch and altitude computations. We got a bit of a surprise with inverted flight during our first few test flights. We thought that angle of attack

would improve pitch and altitude control during inverted flight, but what we found was that after flipping over with altitude controls engaged, the aircraft would lose quite a bit of altitude before the altitude controls would call for enough of a change in elevator deflection to balance things out. A close look at the flight data revealed what was going on: the elevator trim depends on relative wing loading.

That observation is consistent with the theory: the separation between center of gravity and center of pressure generates a torque that must be compensated for by the elevator. What surprised us was that quite a bit of elevator trim was typically required for inverted flight, in some cases we saw inverted trim values as large as 60% of full deflection. This also explained why, in those cases, maximum inverted turn rate was lower than maximum normal turn rate. Therefore we realized we needed an elevator trim model as well as an angle of attack model. The two models were defined within the same framework (relative wing loading) so there was not much extra work to do. There were two parts: parameter estimation and model implementation.

We realized that the models would not do anyone much good unless we could find a way to estimate their parameters from flight data. Fortunately, that was easy enough to do. We tried analyzing the data under various flight conditions such as normal and inverted flight, turns, straight flight, varying the airspeed and found that a simple technique provided the best plots such as the ones shown in Figure 1: engage altitude control in fly-by-wire mode with straight and level flight, and use elevator trim to vary the speed, going all the way down to a stall. Ideally perform both normal and inverted flight, but if you do not intend to fly inverted, then you do not need to gather data in inverted flight. You may need to use the elevator boost parameter in order to get significant variation in speed.

So, there is a relatively easy way to get the model parameters: perform a test flight in which there are several segments in which you vary the speed with the elevator trim while flying more or less straight. Use only those points in which the plane is approximately level with small vertical velocity, they provide the best fit. Also, for real flights, it is best to do the test flight when there is not much wind, because wind can create updrafts or downdrafts which will bias the estimation of angle of attack parameters. For elevator trim parameters, simply plot the elevator deflection versus the relative wing loading. For angle of attack parameters, we recommend comparing the angle of the air velocity vector with the pitch orientation reported by the IMU.

Regarding the IMU reference for level, the IMU accelerometer offsets should be determined with the accelerometer level with the earth. As a result, IMU estimates of pitch orientation will be the orientation of the board with respect to level. The board itself should be mounted approximately parallel to the air velocity vector, but it does not need to be exact, because any slight pitch offset will be accounted for with an offset in the angle of attack model.

Once the parameters of the two models are established, it remains to implement them. The obvious approach would be to compute the relative wing loading from estimated airspeed and accelerometer measurements. However, that does not work, there are issues

with that approach. Although any reasonable sort of air speed estimate can be used, the accelerometers should not be used. The reason for that is they will create a feedback instability because of transient effects not accounted for in the previous explanation. Consideration of what happens during takeoff will shed some light on the issue:

During takeoff, the takeoff roll will generate extra loading transient on the wing. If the angle of attack estimate responds to it, it will increase the target pitch in response to the load transient, which will increase the load, in a positive feedback loop. The result will be that the plane will go full pitch up and stall.

There is a simple solution to the issue: use the computed wing loading according to the equation developed in the theory of helical turn control:

$$\frac{Lift}{g \cdot mass} = \frac{1}{Z} \cdot \frac{1}{1+p^2} = \frac{1}{Z} \cdot \left(1 - Y^2\right) = \frac{1}{Z} \cdot \left(X^2 + Z^2\right) = \left(\frac{X}{Z}\right) \cdot X + Z \qquad \text{Equation 1}$$

Equation 1 is shown in that form because it suggests an approximation that is easy to compute and has been shown through testing to perform very well:

$$\frac{Lift}{g \cdot mass} \approx \left(\frac{\check{X}}{\check{Z}}\right) \cdot X + Z \approx \left(\frac{-\check{\omega} \cdot S}{g}\right) \cdot X + Z$$

$\check{\omega}$ = desired rotation rate in radians per second Equation 2

S = air speed in meters per second

Equation 2 will track the actual wing loading most of the time, with departures just when we need them, during transients. Hence it can be used to accurately compute angle of attack and pitch trim in a feed forward method, without risking positive feedback instability. Of course, the lift computed by equation 2 must then be multiplied by the square of the ratio of cruise airspeed divided by actual airspeed.

Some consideration must be given stall conditions and the range of airspeed over which equation 2 is valid. You can get some idea of the stall speed with a test flight with fly by wire and altitude controls engaged. Pull back on the elevator slowly, it will cause the plane to climb and slow down. At some point it just will not go any slower, and will show signs of stalling. You can then read the stall speed from the flight data. A good rule of thumb is to set the cruise speed to be equal to twice the stall speed, and perform the computations as long as the airspeed is greater than 1/2 of the cruise speed. Below that, turn off the angle of attack and pitch trim computations, and set the angle of attack and pitch trim to zero.

The pitch trim is used only to adjust the elevator trim. The angle of attack is used in several places, including:

- Centrifugal compensation. There are now two components of the airspeed vector in the body frame. Use the complete airspeed vector in the cross product of the body frame rotation vector with the airspeed vector to get the complete centrifugal acceleration vector.
- Wind estimation. Angle of attack must be accounted for.
- Pitch control. Multiply the angle of attack by Z to get the earth frame pitch adjustment.
- Air velocity yaw reference vector. There is an adjustment in the horizontal plane yaw reference vector that is a function of angle of attack and roll orientation.

In MatrixPilot, the pitch adjustment is implemented by:

```
// project angle of attack into the earth frame
accum.WW = ( __builtin_mulss( angleOfAttack , rmat[8] ) ) << 2 ;
pitchAdjustAngleOfAttack = accum._.W1 ;
```

The yaw reference adjustment is implemented by:

```
// compute horizontal projection of air velocity,
// taking into account the angle of attack.
longaccum.WW = ( __builtin_mulss( rmat[2] , angleOfAttack ) ) << 2 ;
dirOverGndHrmat[0] = rmat[1] + longaccum._.W1 ;
longaccum.WW = ( __builtin_mulss( rmat[5] , angleOfAttack ) ) << 2 ;
dirOverGndHrmat[1] = rmat[4] + longaccum._.W1 ;
dirOverGndHrmat[2] = 0;
```