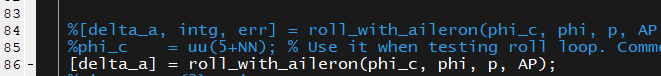
**Case1**

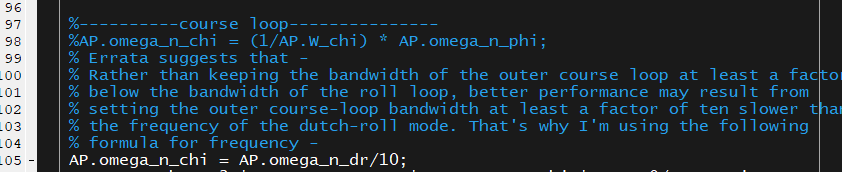
1\_roll\_loop

* Initially there was an error in the transfer function of the roll loop. I’ve fixed it and tuned the roll loop Simulink model first and then ran the autopilot code and found that the roll plot looks perfect.
* For testing the roll plot alone without concerning about course loop, have to uncomment phi\_c = uu(5+NN) so that manual input will be sent to the roll loop to bypass the course command which used to dictate the phi\_c –

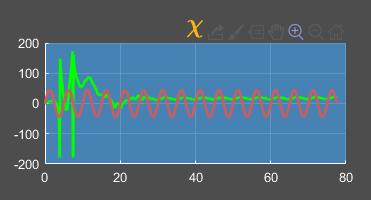


2\_course\_loop

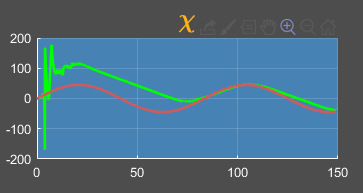
* Since the transfer function issue is fixed, course loop also works accurately.
* I now changed the factor to exactly what was suggested in the book (=1/10th of dutch roll frequency in the *compute\_autopilot\_gains.m* file) and tuned again and the course loop looks works well.



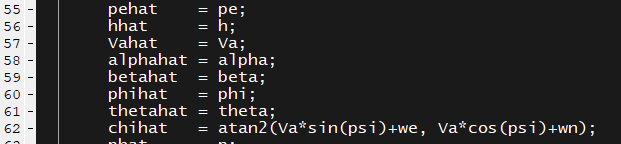
* Note that the in the signal generator through which we set the command signal has to be se properly –
  + If we set a frequency of input signal to as high as 0.2, course hold does not work well—



* + But I then tried setting it 0.012 which is lower frequency it works well—

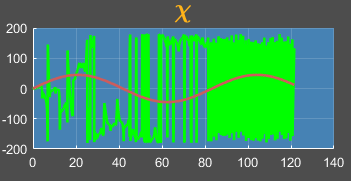


* In the successive loop closure technique, the very premise is that the inner loops are 5-10 times faster than the outer loops. Hence we need to set the frequency of course loop at least 5-10 times lower than the roll loop.
* Also check that in *true\_states.m* file, **chi** depends on wind as well—



To first check the course loop, I’ve made all the **wind components zero**.

* Later on I tried one case by setting **WIND.Va0 = 10 m/s** inside *wind\_parameters.m* the course loop was **unable to handle** it –



* But when I tried **WIND.Va0 = 5 m/s,** it works fine.

Case files were not saved for 1\_test folder. So tried to do again in 2\_test to generate similar results.

**Case2**

1\_roll\_loop

Commands:

Va\_c = 35

h\_c = 100

theta\_c = 0

chi\_c = 0

roll\_c = square, 45, 0.1

Throttle limit = 1.5

2\_course\_loop

Commands:

Va\_c = 35 +/- 7

h\_c = 100

theta\_c = 0

chi\_c = sine, 45, 0.012

roll\_c = 0

Throttle limit = 1.5

3\_pitch\_loop

Commands:

Va\_c = 35

h\_c = 100

theta\_c = sine, 20, 0.1

chi\_c = 0

roll\_c = 0

Throttle limit = 1.5

4\_airspeed\_loop

1. Commands:

Result file: ***4\_airspeed\_throttle\_1.5.jpg***

Va\_c = 100 +/- 25

h\_c = 100

theta\_c = 0

chi\_c = 0

roll\_c = 0

Throttle limit = 1.5

1. Commands:

Result file: ***4\_airspeed\_throttle\_4.5.jpg***

Va\_c = 100 +/- 25

h\_c = 100

theta\_c = 0

chi\_c = 0

roll\_c = 0

Throttle limit = 4.5

No matter how much I increase throttle, there seems to be a saturation limit on the airspeed. Never able to go beyond 100 m/s. Have to fix this bug. If I give too high a throttle to make airspeed cross 100, there is error and the code stops running. But it works fine for Va\_c = 70 or such values lesser than 100. Look at the case below.

1. Commands:

Result file: ***4\_airspeed\_throttle\_2.5+.jpg***

Va\_c = square **70 +/- 5**, at freq 0.012

h\_c = 100

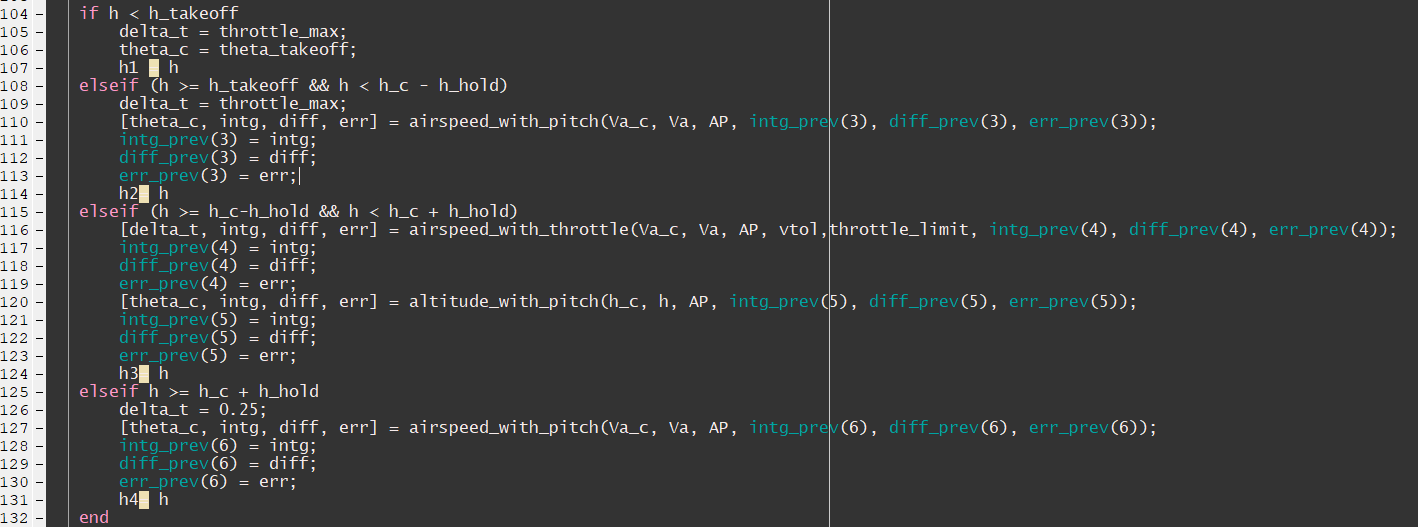
theta\_c = 0

chi\_c = 0

roll\_c = 0

Throttle limit = 2.5

Does achieve commanded velocity but not commanded height. Height saturates below 25 meters. It means that the throttle is not sufficient for this aircraft to reach the commanded height. So I increased **throttle\_max** to 2.5 and also set this as the throttle value during takeoff and climb as well.



5\_altitude\_loop

1. Commands:

File: ***5\_altitude\_loop\_1000m.jpg***

Va\_c = square **70 +/- 5**, at freq 0.012

h\_c = **1000**

theta\_c = 0

chi\_c = 0

roll\_c = 0

Throttle limit = 2.5

1. Commands:

File: ***5\_altitude\_loop\_10000m.jpg***

Va\_c = square **70 +/- 5**, at freq 0.012

h\_c = **10000**

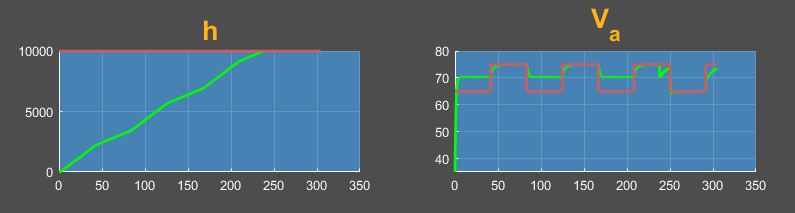
theta\_c = 0

chi\_c = 0

roll\_c = 0

Throttle limit = 2.5

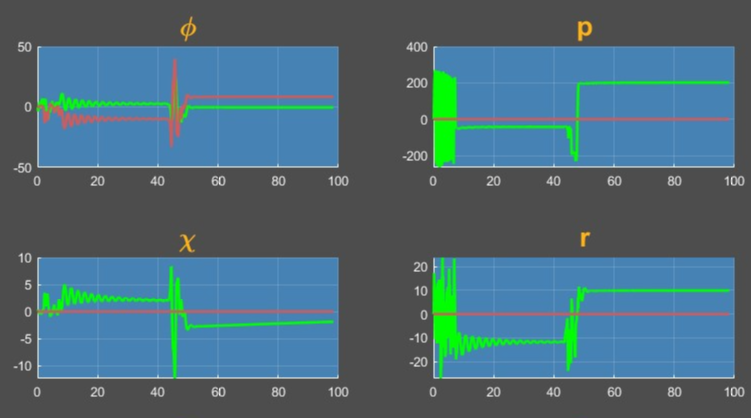
During the climb it runs for entire 10 km climb in full throttle. Hence to be able to climb, it is maintaining a higher velocity. So it is achieving 70 and above values. But it is not achieving commanded velocity below 70 because it cannot climb if it reduces velocity. But you can observe that once it reaches 10 km altitude, it reduced the throttle and it now achieves both lower and upper limits on velocity.



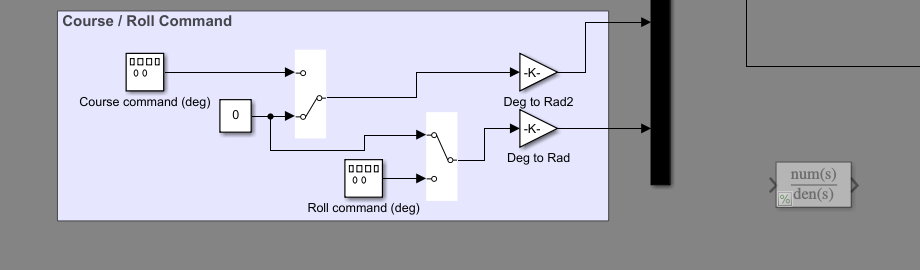
**Case3**

File: ***Issue - when P const why Phi const.jpg***

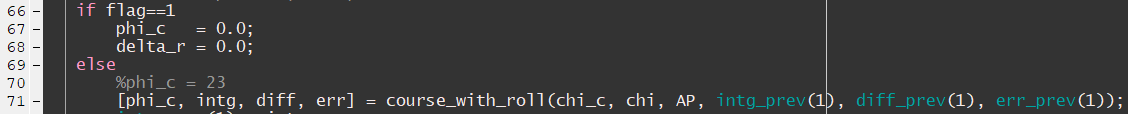
* Issue – Rajat pointed out that when ***p*** is constant ***phi*** should vary. But both P and Phi cannot be constant.
* Similarly when ***r*** is constant why is ***chi*** constant?



* Another puzzle is when I input zero ***phi\_c***, why in the above plot it is non-zero?



* + In the final autopilot we usually do not command phi\_c but we demand chi\_c. So unless when we are exclusively testing whether roll\_loop is working fine, phi\_c is overwritten based on chi\_c demand as follows—



However note that the chi\_c remains zero in the plot – hence there is no issue here.