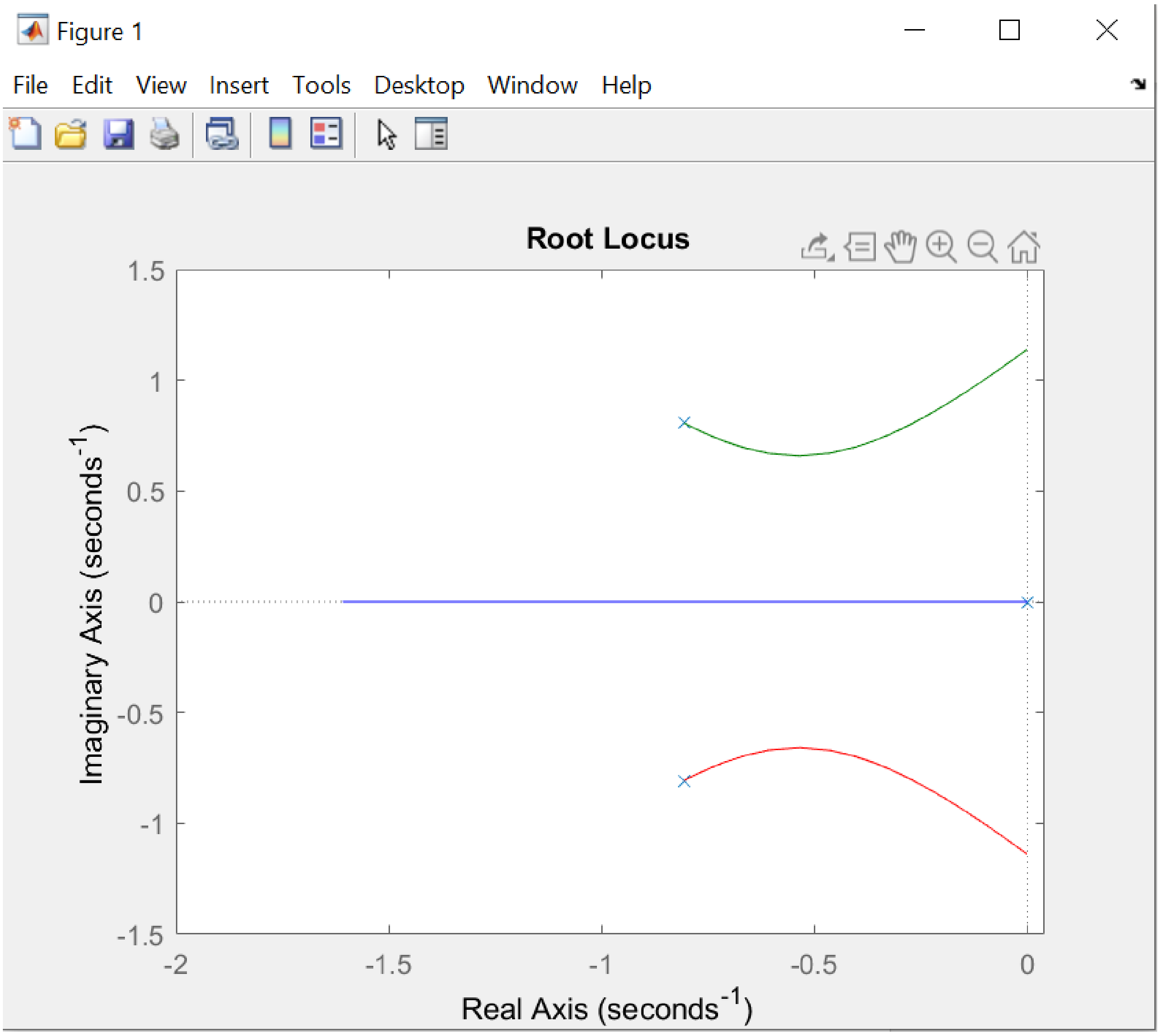
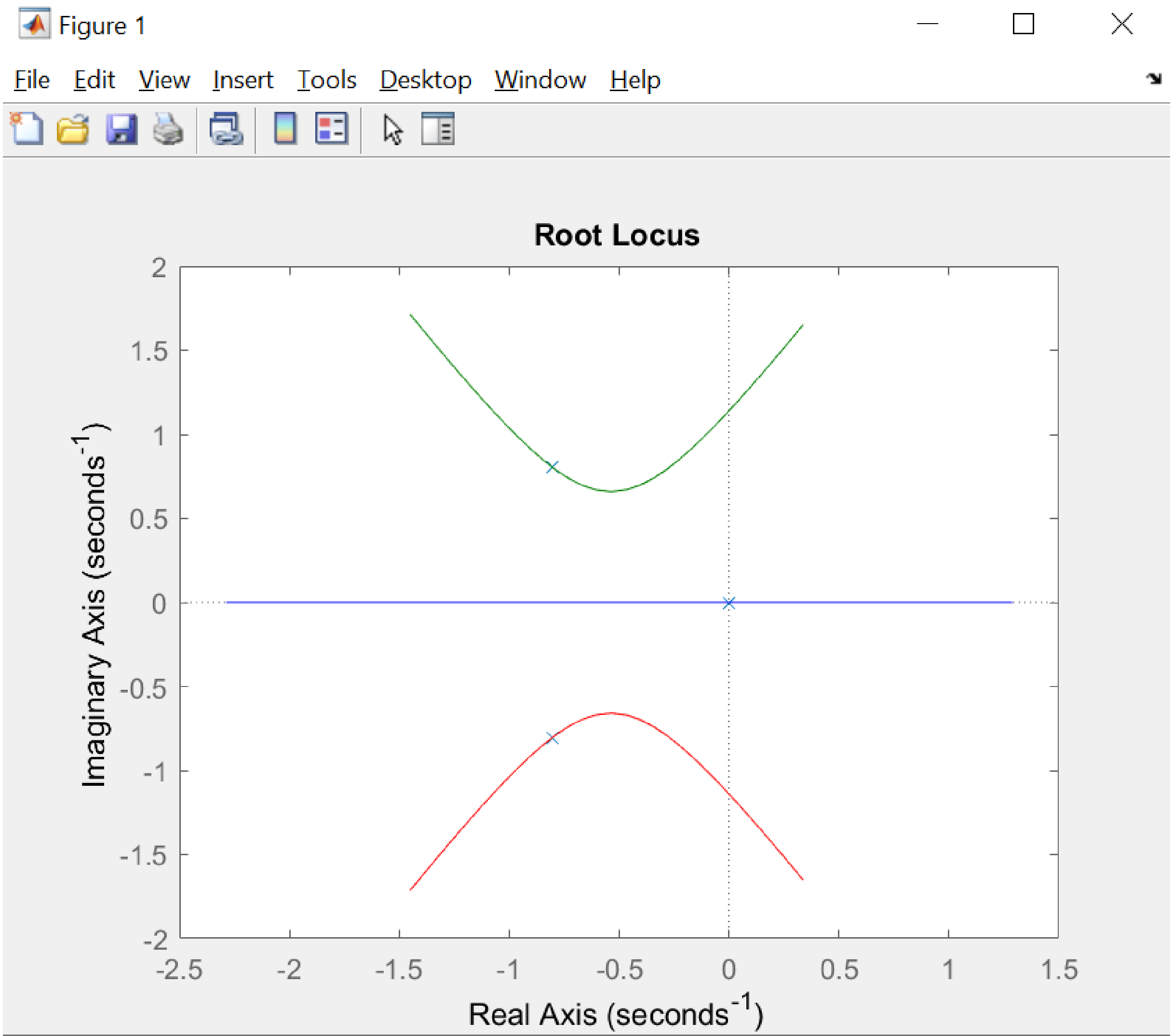
Task 2c)

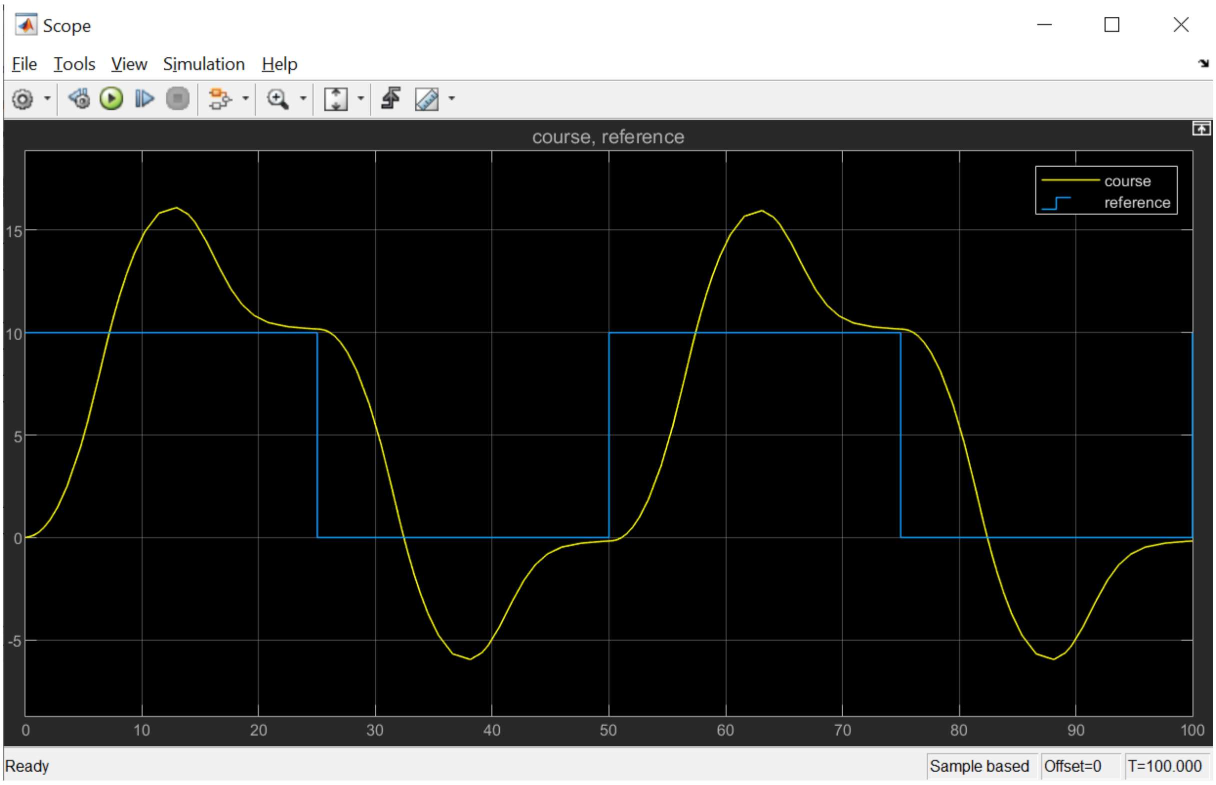


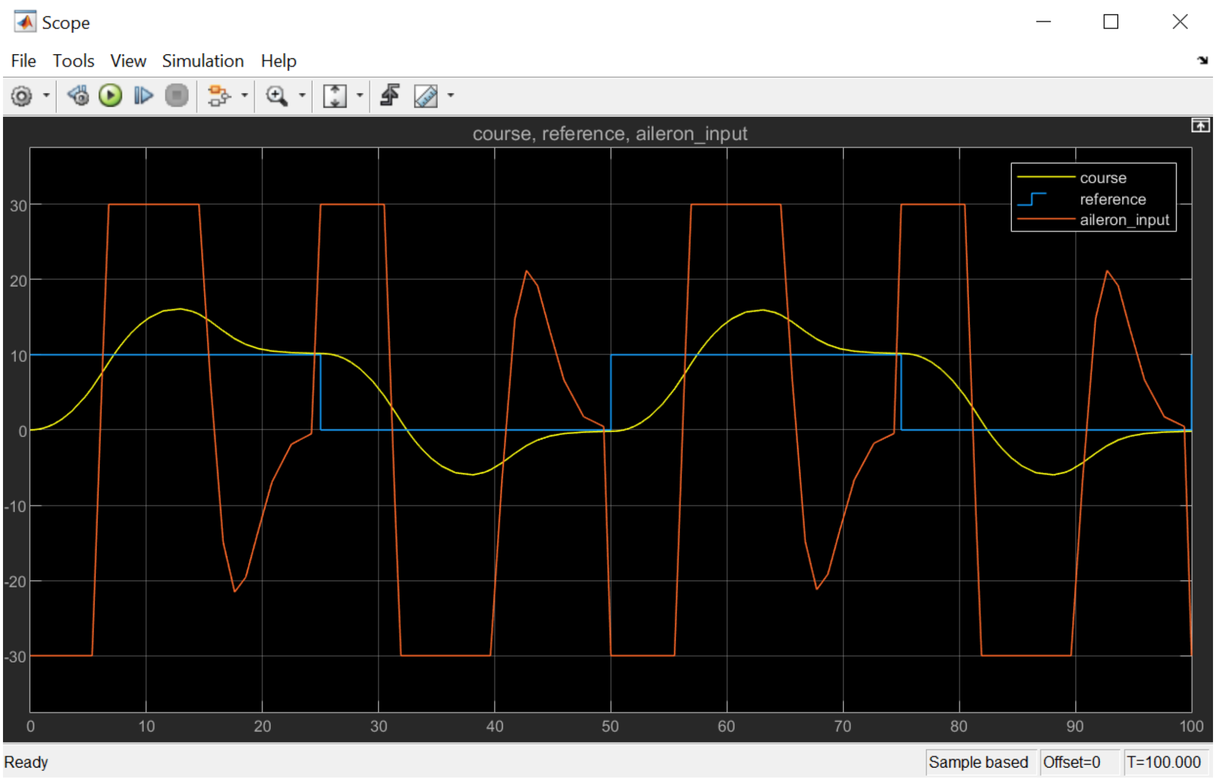
Root locust plot of range [-3.2:0.1:0], as we see, -3.2 is about the limit where the system is stable.

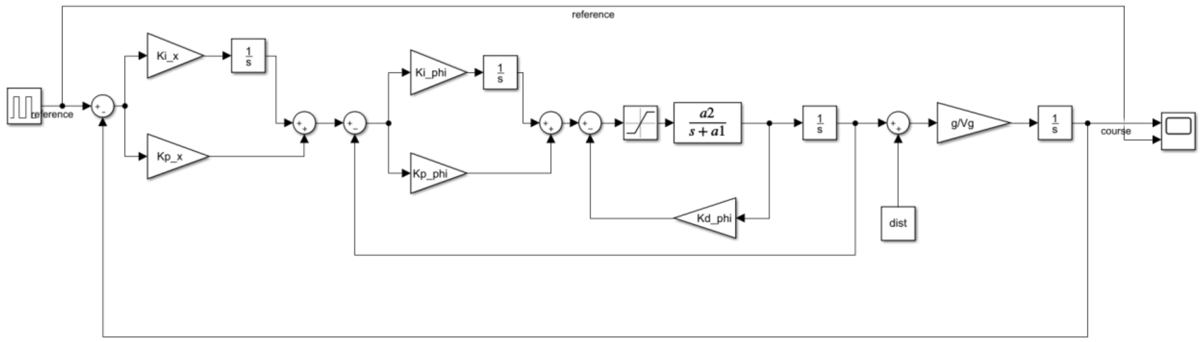


Root locus plot of range [-10:0.1:10]

2e)


Scope
File Tools View Simulation Help
course, reference
course
reference
15
10
5
0
-5
0 10 20 30 40 50 60 70 80 90 100
Ready
Sample based
Offset=0
T=100.000


Scope
File Tools View Simulation Help
course, reference, aileron_input
course
reference
aileron_input
30
20
10
0
-10
-20
-30
0 10 20 30 40 50 60 70 80 90 100
Ready
Sample based
Offset=0
T=100.000



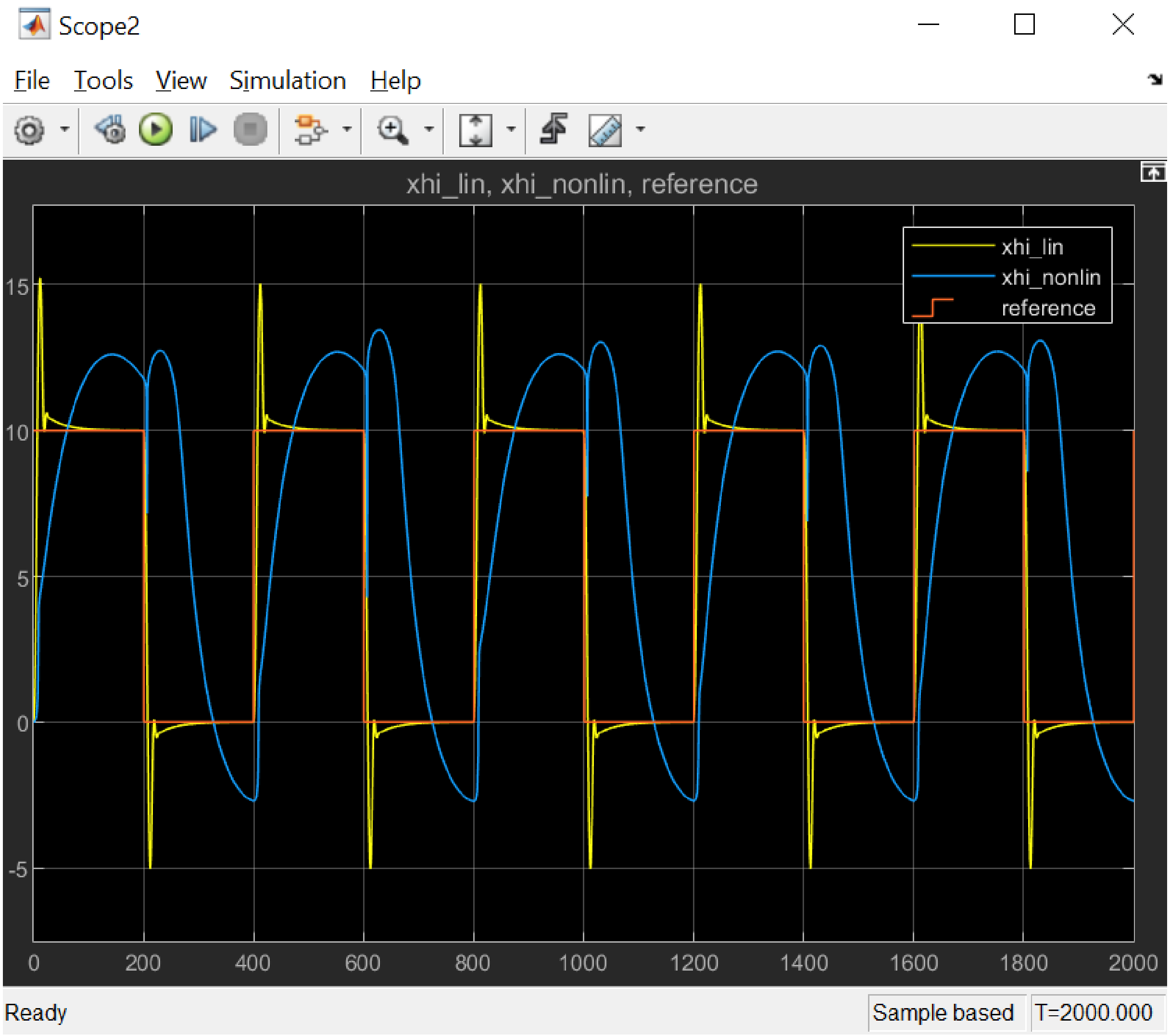

```
addpath(genpath("C:\Users\jacob\Documents\Student\Fartøy\MSS"));
Kp_phi = -2;
Kd_phi = 1.935;
Ki_phi = 0;

a2 = -0.65;
a1 = 2.87;
Vg = 161.1;
g = 9.81;

omega_n_phi = 1.14;
Wx = 7;
omega_n_x = omega_n_phi/Wx;
zeta_x = 1;

Kp_x = 2*zeta_x*omega_n_x*Vg/g;
Ki_x = omega_n_x^2 *Vg/g;

dist = 1.5;
e_phi_sat = 15;
sim("succ_loop_sim.slx")
```

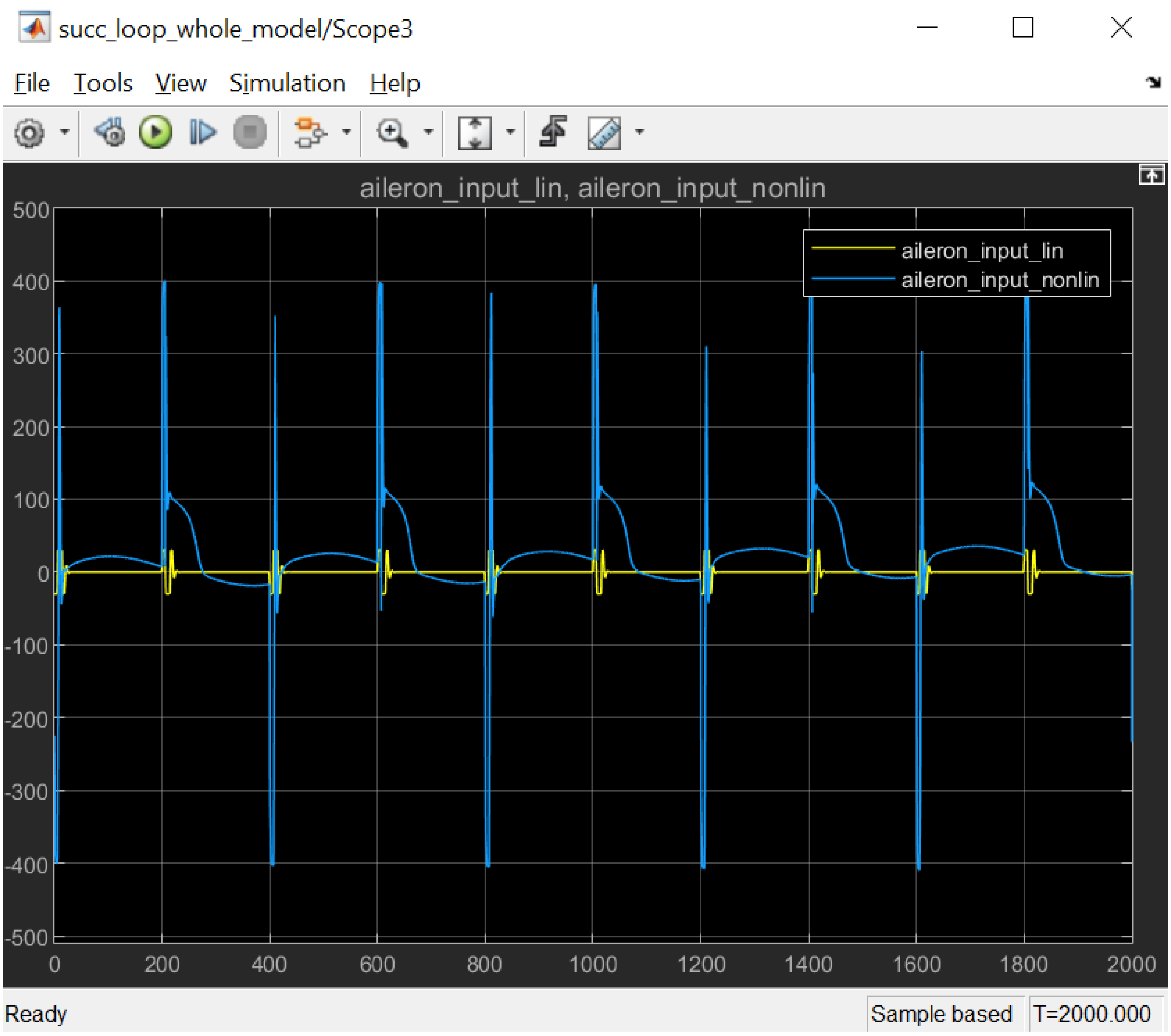
Kp_x = 5.35

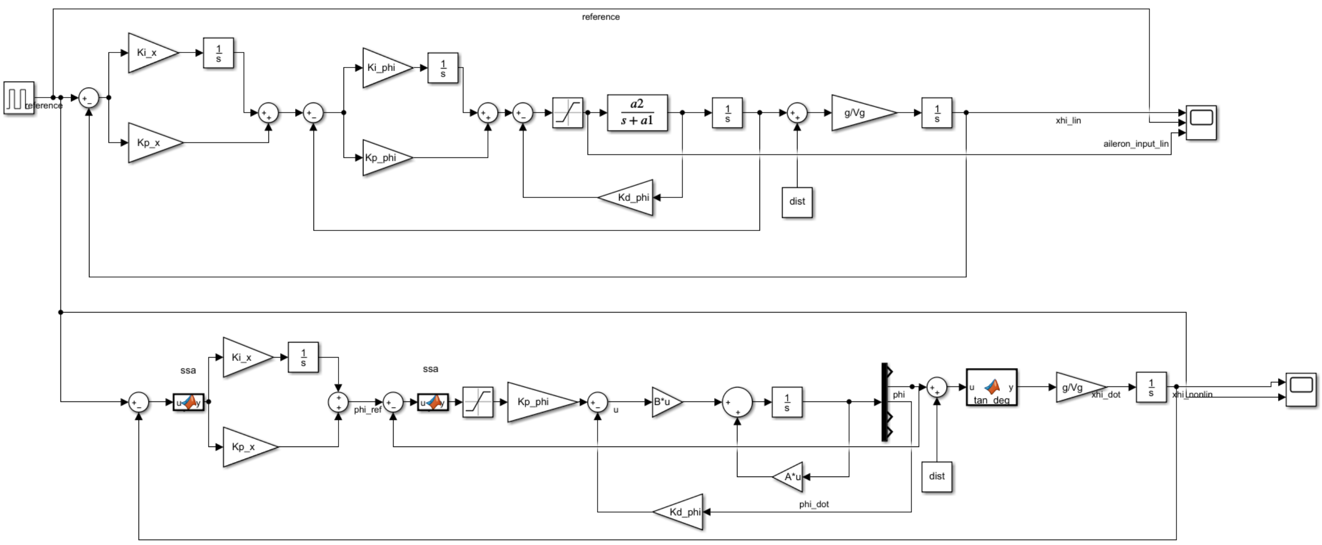
Ki_x = 0.44

As we see from the plots, the system reaches its reference with a lot of overshoot and after quite some time. But it is stable. Tuning it a bit I was not able to get much better results than the ones shown above. This was with tuning the Wx and the zeta_x only. The inner loop remained constant. Some tuning of the inner loop as well might give a better result.

2f)


Scope2
File Tools View Simulation Help
xhi_lin, xhi_nonlin, reference
xhi_lin
xhi_nonlin
reference
15
10
5
0
-5
0 200 400 600 800 1000 1200 1400 1600 1800 2000
Ready
Sample based
T=2000.000





We see that the real dynamics are much slower than the linearized ones used previously. The controller reaches its target, but much much slower. I also didn’t get the full version to work with saturation on the control-input. As we see from the aileron-plot the given u is extremely high at some points, and shows that the controller we made wouldnt work for the actual system. (Assuming solution is correct <3)

2g)

I didn’t get the simulation to converge with input-saturation so I coulnd’t any simulated results, but in general when you have a large error and an integral-term where the reference changes, you get integral windup. To avoid it you can have some adaptive cancellation of the i-term. For example disabling the integration-term while in the un-controllable area, or resetting the integral-value every time the reference is equal to the state.