Consider the Simulink file attached as *geometric_control_template.slx*. Within this file, you can find a template to implement yourself the geometric control. You must fill in the inner and outer loops. Simulate the scheme and report the plots you believe are most interesting. You may add further scopes to the scheme to extract data you believe most interesting to show. [Hint: consider that, if it happens, a constant error in the angular part is acceptable.]

The changes made were carried out in the Outer-loop control and Inner-loop control blocks

Outer loop control

$$u_T = -(-K_p e_p - K_v \dot{e}_p - mge_3 + m\ddot{p}_{b,d})^T R_b e_3$$

$$z_{b,d} = -\frac{-K_p e_p - K_v \dot{e}_p - mge_3 + m\ddot{p}_{b,d}}{\left\|-K_p e_p - K_v \dot{e}_p - mge_3 + m\ddot{p}_{b,d}\right\|}$$

Inner loop control (after retrieved $R_{b,d}$)

$$\tau^b = -K_R e_R - K_\omega e_\omega + S(\omega_b^b) I_b \, \omega_b^b - I_b (S(\omega_b^b) R_b^T R_{b,d} \omega_{b,d}^{b,d} - R_b^T R_{b,d} \dot{\omega}_{b,d}^{b,d})$$

Outler-loop control

```
function [uT, zb_des, err_p, dot_err_p] = fcn(Kp, Kv, position, linear_vel, Rb, mass, ...
    position_des, vel_linear_des, acc_linear_des)

err_p=position-position_des;
dot_err_p=linear_vel-vel_linear_des;

uT=-(-Kp*err_p-Kv*dot_err_p-mass*9.81.*[0 0 1]'+mass.*acc_linear_des)'*(Rb*[0 0 1]');

zb_des=-(-Kp*err_p-Kv*dot_err_p-mass*9.81.*[0 0 1]'+mass.*acc_linear_des)/...
    norm(-Kp*err_p-Kv*dot_err_p-mass*9.81.*[0 0 1]'+mass.*acc_linear_des);
```

In addition in this block, we got $e_p = p_b - p_{b,d}$ and $\dot{e}_p = \dot{p}_b - \dot{p}_{b,d}$

Inner-loop control

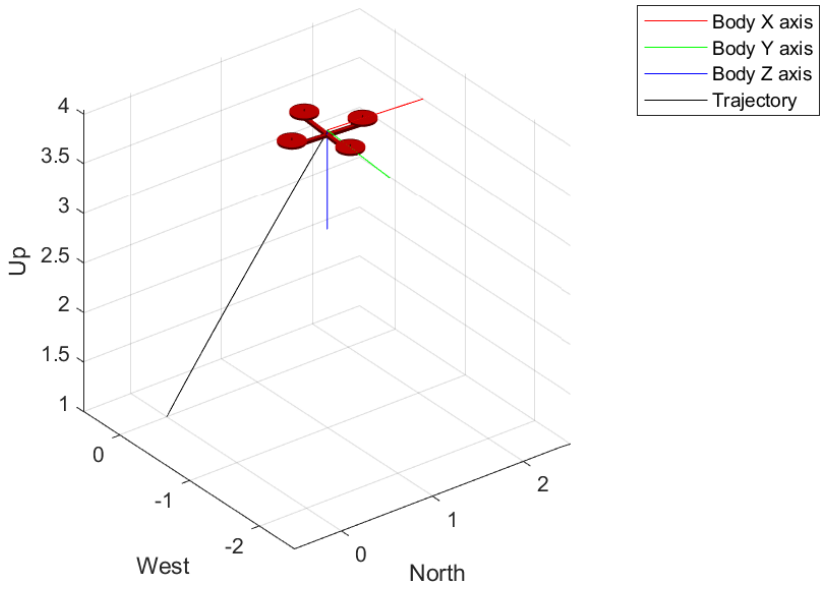
```
function [Rb_des,omega_bb_des,tau,err_R,err_W] = fcn(zb_des,Rb,dot_Rb_des,...
    omega_bb,dot_omega_bb_des,xb_des,Ib, Kw, Kr)

yb_des=skew(zb_des)*xb_des/norm(skew(zb_des)*xb_des);
xb_des_proj=skew(yb_des)*zb_des;

Rb_des=[xb_des_proj yb_des zb_des];
omega_bb_des = wedge(Rb_des'*dot_Rb_des);
err_R=0.5*wedge(Rb_des'*Rb-Rb'*Rb_des);
err_W=omega_bb-Rb'*Rb_des*omega_bb_des;
tau=-Kr*err_R-Kw*err_W+skew(omega_bb)*Ib*omega_bb-Ib*(skew(omega_bb)*...
    Rb'*Rb_des*omega_bb_des-Rb'*Rb_des*dot_omega_bb_des);

end
function S = skew(v)
if(numel(v)~= 1)
S= [0 -v(3) v(2);
    v(3) 0 -v(1);
    -v(2) v(1) 0];
else
S= zeros(3);
end
end
function v = wedge(S)
v1 = S(3,2);
v2 = S(1,3);
v3 = S(2,1);
v = [v1 v2 v3]';
end
```

In addition at τ^b; we have calculated also $\omega_{b,d}^{b,d}$, e_R e_ω and $R_{b,d}$

$$R_{b,d} = \left[S(y_{b,d}) z_{b,d} \quad \underbrace{\frac{S(z_{b,d}) x_{b,d}}{\| S(z_{b,d}) x_{b,d} \|}}_{y_{b,d}} \quad z_{b,d} \right]$$

The gains were chosen through a trial and error mechanism; $f$or the outer loop

$K_p = \begin{bmatrix} 8 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 8 \end{bmatrix}, K_v = \begin{bmatrix} 0.6 & 0 & 0 \\ 0 & 0.6 & 0 \\ 0 & 0 & 0.6 \end{bmatrix}$,for the inner loop $K_r = \begin{bmatrix} 7 & 0 & 0 \\ 0 & 7 & 0 \\ 0 & 0 & 7 \end{bmatrix}, K_w = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$



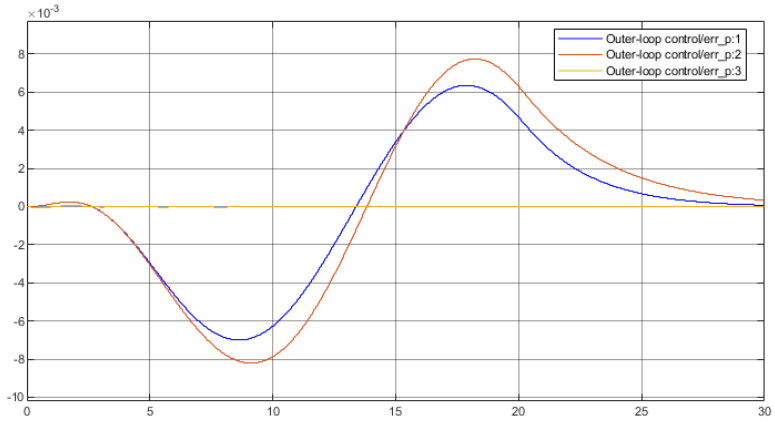


Figure 1: position error

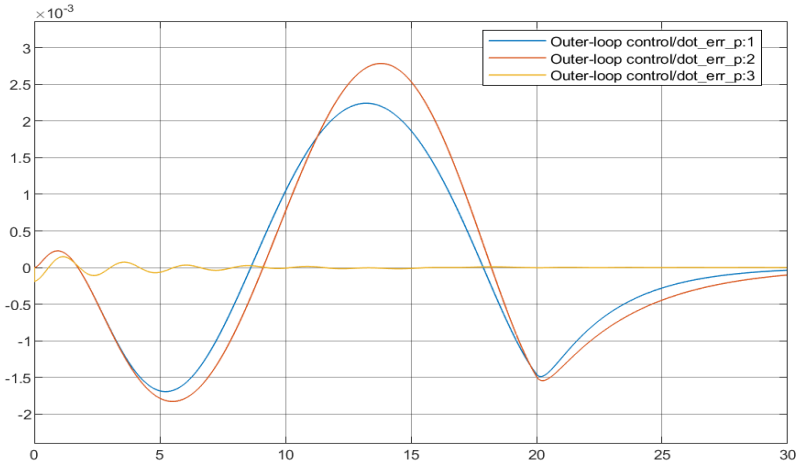


Figure 2:linear velocity error

The position error and the linear velocity error are of the order of 10^{-3}.

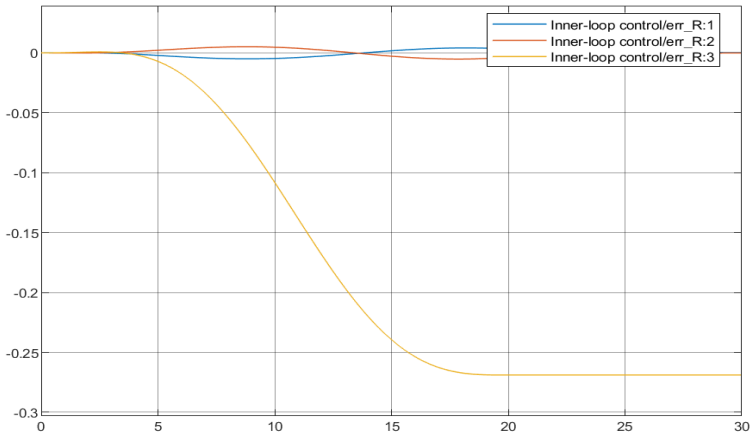


Figure 3:rotation error

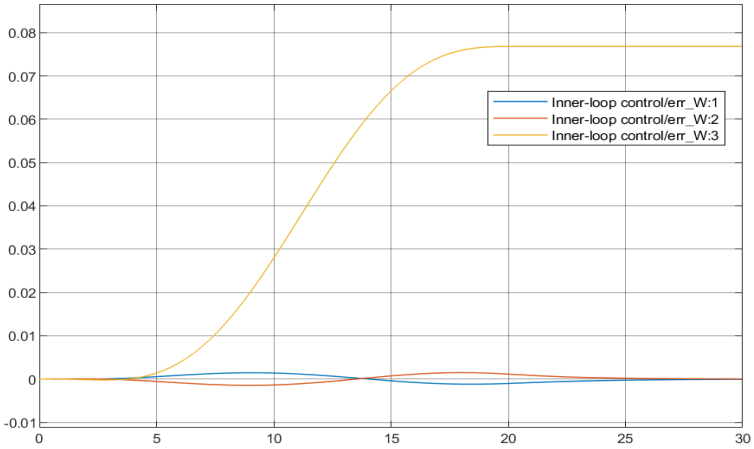


Figure 4:angular velocity error

The third component of the angular velocity error is of the order of 10^{-2}, while the third component of the angular error is bigger. In both cases the first 2 components of the errors tend to have a much lower steady-state value than the third components

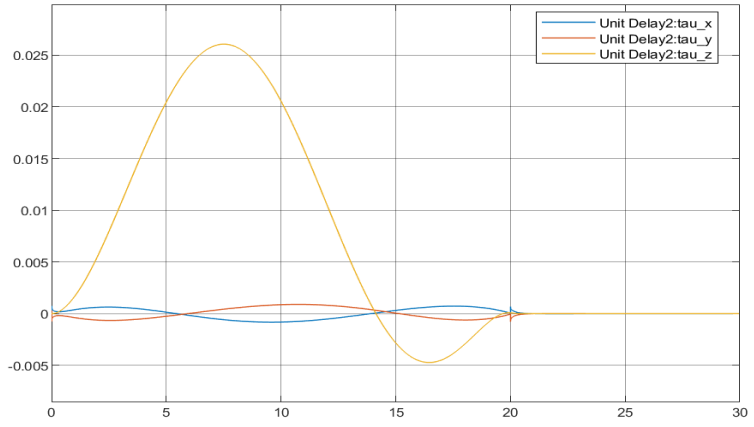


Figure 5:tau_b

The torque becomes constant after almost 20s

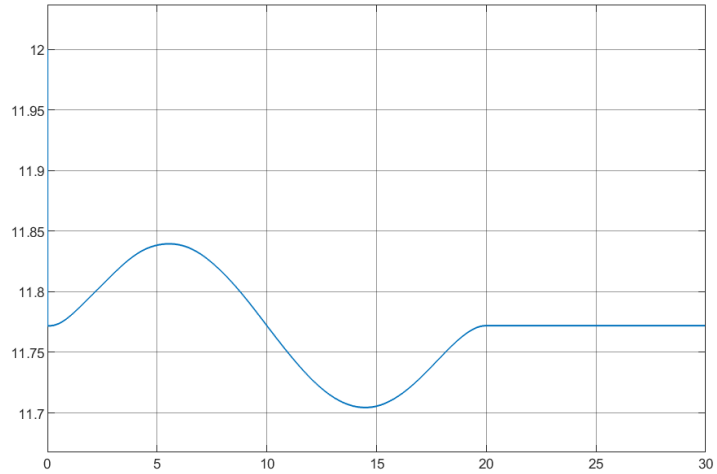


Figure 6:uT

The total thrust becomes constant after almost 20s, while before there is an oscillation.

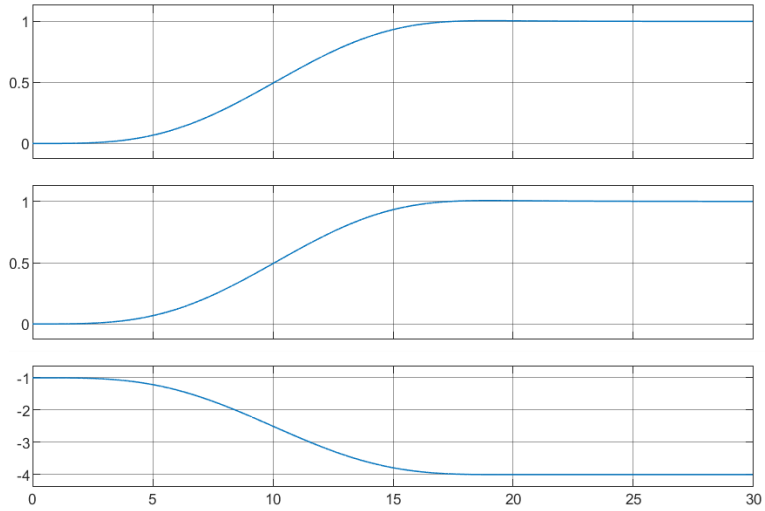


Figure 7:position

The quadrotor starts from (0,0,-1) and arrive to (1,1,-4) , the position tends to stabilize in around 20s

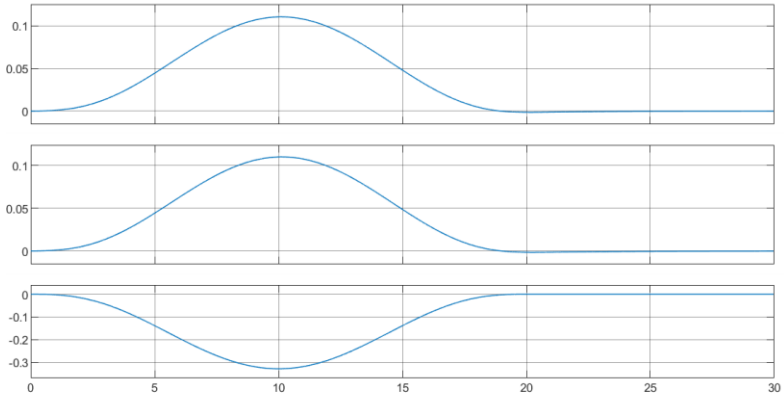


Figure 8:linear velocity

Speeds tend to reach 0 in about 20s.

The file containing the algorithm is called: " geometric_control_template.slx"