**Quadcopter sumulation**

* **Data**

1. X=[3xN double] **The initial frame axes**
2. omega= [3x1 double]; **Initial Angular velocity**
3. T= [3x1 double]; **Thrust**
4. Input= [4xN double]
5. t = [1xN double]; **Simulation times, in seconds**
6. Dt= 0.0050
7. x = zeros(3, N);
8. xdot= zeros(3, N);**Initial Linear Velocity**
9. theta= zeros(3, N); **initial Body axes.**
10. thetadot = zeros(3, N); **Derivatives of roll, pitch, yaw.**
11. inputs=3; **Given Current Inputs**
12. **R**=[3x1 double];**Rotation transformation matrix**
13. **Fd**=[3x1 double];**Global drag forces**
14. a =[3x1 double];**Acceleration**
15. omegadot =[3x1 double];**Angular acceleration**
16. tau =[3x1 double];**Torques**
17. 𝜙= Roll angle. 𝜃=Pitch angle. 𝜓= Yaw angle.

* **Physical constants**.

1. g = 9.81; **Acceleration due to gravity**
2. m = 0.5; **Mass of quadcopter**
3. L = 0.25; **Length between the quadcopter center and any propellers**
4. k = 3e-6; **Thrust Coefficient**
5. b = 1e-7; **Drag coefficient**
6. I = [5e-3, 5e-3, 10e-3]; **Inertia matrix,** **Inertial tensor of the quadcopter**
7. kd = 0.25; **Global drag coefficient**
8. deviation = 100; **Deviation in the angular velocity. Deviation is in degrees/sec**.

* **Functions**

1. **omega = thetadot2omega(thetadot, theta);**

***(Compute angular velocity. Convert derivatives of roll, pitch, yaw to omega.)***

* phi = theta(1);
* theta\_ = theta(2);
* psi = theta(3);

1. **T = thrust(inputs, k)**

***(Compute thrust.)***

1. **a = acceleration(inputs, theta, xdot, m, g, k, kd)**

***(Compute acceleration.)***

* phi = theta(3);
* theta = theta (2);
* psi = theta (1);

1. **tau = torques(inputs, L, b, k)**

***(Compute torques.)***

1. **omegadot = angular acceleration** **(inputs, omega, I, L, b, k)**

***(Compute angular acceleration.)***

1. **thetadot = omega2thetadot(omega, theta)**

***(Compute roll, pitch, yaw derivatives.*** ***Convert omega to roll, pitch, yaw derivatives)***

* phi = theta(1);
* theta\_ = theta(2);
* psi = theta(3);