

Quadrotor Simulation

For ISL Lab

Author : Michael Chojnacki

Spring 2013-2014

Intelligent Systems Lab (ISL), CS Faculty
Technion, Israel Institute of Technology
32000, Haifa
Israel

Table of Contents

Abstract.....	3
1. Quadrotor basics Concepts	3
2. Simulation Overview	5
3. System Dynamics	7
4. Control	11
5. Sensor Readings and State Estimation	14
6. Actuators.....	18
7. Constants and Variables	21
8. Simulation Results	22
9. References	25
APPENDIX	26

Abstract

This report presents a Quad-rotor simulation built for the Intelligent Systems Lab at the Computer Science Faculty at the Technion IIT.

Different projects at the lab use quadrotors as platforms for research, mainly in the computer vision aided navigation field. A tailor made quadrotor simulation responds to the need to fully understand the system dynamics, the control laws and sensors modeling, in order to assess the algorithms developed in the frame of the research prior to flight testing

1. Quadrotor basics Concepts

The quadrotor considered in this simulation is made of 4 electrical motors and rotors joined in a cross configuration by 2 perpendicular links.

The 4 motors are considered identical, their thrust vectors are considered parallel and perpendicular to the plane formed by the two joining links. The two joining links are considered rigid and weightless.

Under those assumptions, the system can be modeled as follows :

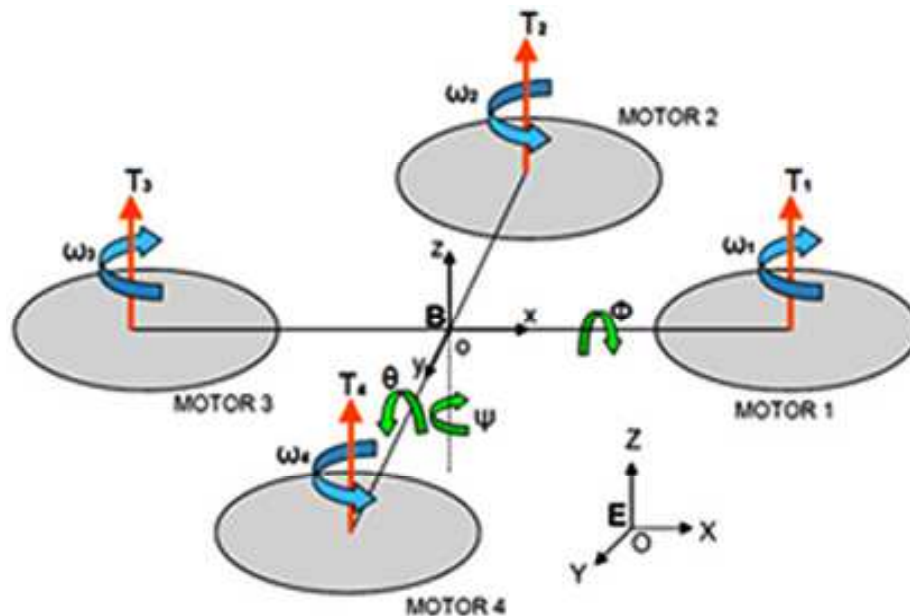


Figure 1 : Quadrotor free body diagram

Such a quadrotor can be controlled by manipulating thrust forces from each of the rotors in order to generate the required movement:

- Vertical Movement is achieved by simultaneously increasing or decreasing the motor speeds (Fig. 2c)
- Yaw movement (around z body axe) is achieved by manipulating the torque balance (Fig. 2a & 2b)

- Pitch & Roll movements are controlled by applying differential thrust forces on opposite rotors (Fig. 2d)

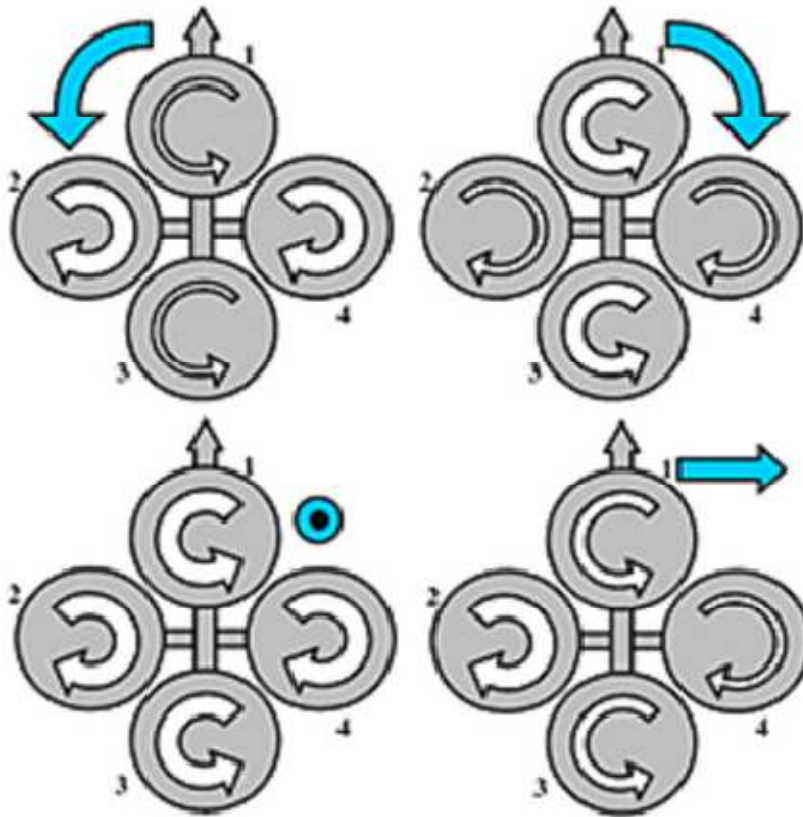


Figure 2 : Quadrotor dynamics

2. Simulation Overview

The system simulation has been implemented in Simulink® and comprises five main blocks, each performing a different task in the process :

- The **Sensor Readings & State Estimation** block receives the calculated state and outputs an estimation of the same state, after the addition of a simulated noise to the sensors readings.
- The **Control** block receives the state estimation and calculated the required rotor speeds in order to achieve a desired state (ρ, θ, ψ, z)
- The **Actuators** block receives controllers' values and translates them to rotor speeds. (In further eventual development, this could be the place to model the electrical motor)
- The **System Dynamics** block receives the commanded rotor speeds and applies them to the system. A new state is then generated and output.

Quadrotor Dynamics Simulator

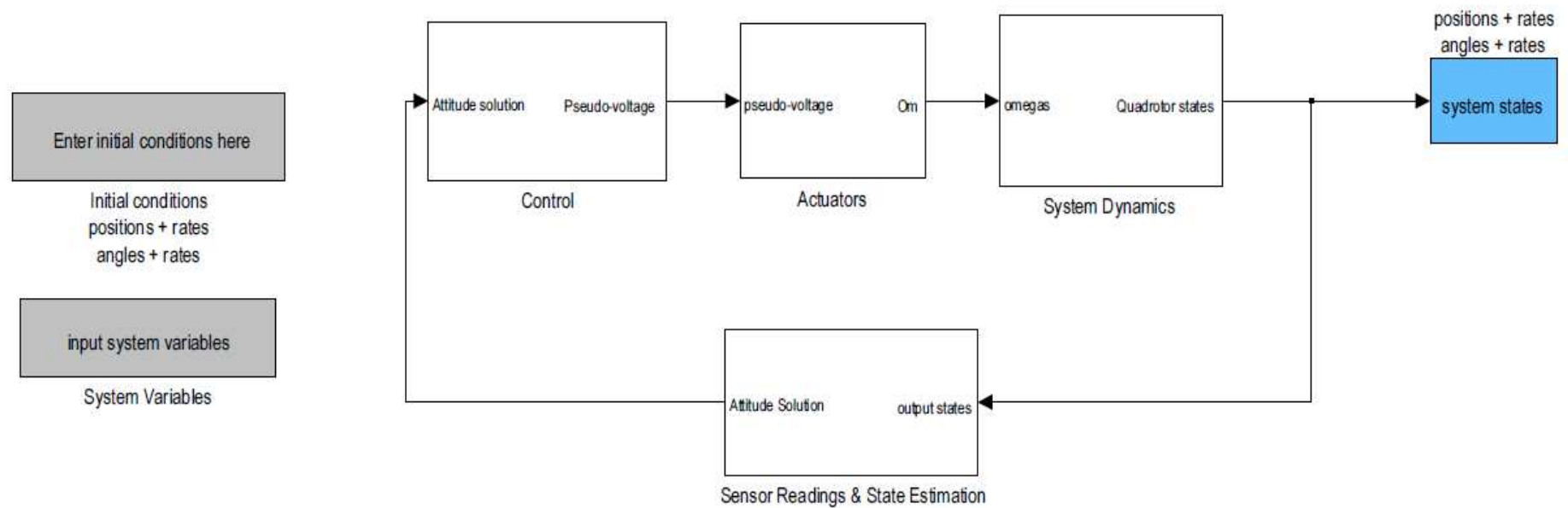


Figure 3 : Quadrotor simulation overview

3. System Dynamics

There are several techniques which can be used to derive the dynamics of a 6 DOF rigid body. In this work we have chosen the Newton Euler formulation, which can be described as follows¹ :

$$\begin{bmatrix} mI_{3 \times 3} & 0_{3 \times 3} \\ 0_{3 \times 3} & I \end{bmatrix} \begin{bmatrix} \dot{V} \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} \omega \times (mV) \\ \omega \times (I\omega) \end{bmatrix} = \begin{bmatrix} F \\ \tau \end{bmatrix} \quad (1)$$

Where :

- $I_{3 \times 3}$ is a 3x3 identity matrix
- m is the quadrotor's mass in $[kg]$
- ω is the quadrotor's angular speed vector in the earth-fixed frame, in $[rad / s]$
- I is the quadrotor's inertia matrix in the earth-fixed frame in $[N m s^2]$
- V is the quadrotor's linear speed in $[m / s]$

Moments :

Rolling moments :

- Body gyro effect $\dot{\theta}\psi(I_{yy} - I_{zz})$
- Propeller gyro effect $J_r \dot{\theta}\omega$
- Rotors action $dK_1(\omega_4^2 - \omega_2^2) \quad *$

Pitching moments:

- Body gyro effect $\dot{\phi}\psi(I_{zz} - I_{xx})$
- Propeller gyro effect $J_r \dot{\phi}\omega$
- Rotors action $dK_1(\omega_3^2 - \omega_1^2) \quad *$

Yawing Moments:

- Body gyro effect $\dot{\phi}\dot{\theta}(I_{xx} - I_{yy})$
- Counter-Torque unbalance $K_2(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \quad *$

Forces :

Along the earth-fixed frame (EFF), the forces are expressed simply as :

$$F^I = R_b^e \cdot F^b - \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} \quad (2)$$

where F^b are the forces exercised by the rotors action in the body-fixed-frame (BFF)

$$\text{and } F^b = \begin{bmatrix} 0 \\ 0 \\ \sum_{i=1}^4 T \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ K_1 (\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \end{bmatrix} \quad (3)$$

R_b^e is the Body-to-Earth rotation matrix with the x-y-z sequence (${}^eR_{XYZ}(\rho, \theta, \psi)$) and is defined as :

$${}^eR_{XYZ}(\rho, \theta, \psi) = \begin{bmatrix} c\psi c\theta & c\psi s\theta s\phi - s\psi c\phi & s\psi s\phi + c\psi s\theta c\phi \\ s\psi c\theta & s\psi s\theta s\phi + c\psi c\phi & -c\psi s\phi + s\psi s\theta c\phi \\ -s\theta & c\theta s\phi & c\psi c\phi \end{bmatrix} \quad (4)$$

The forces applied on the quadrotor in the EFF are therefore :

$$F^I = \begin{bmatrix} (s\psi s\phi + c\psi s\theta c\phi) K_1 (\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ (-c\psi s\phi + s\psi s\theta c\phi) K_1 (\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ (c\psi c\phi) K_1 (\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) - mg \end{bmatrix} \quad (5)$$

Where:

- J_r is the rotor's moment of inertia in $[N m s^2]$
- d is the distance from the quadrotor's center of mass to one of the rotors, in $[m]$.
- Ω_r is the body's angular speed in $[rad / s]$
- K_1 is the thrust factor (see Paragraph 6)
- K_2 is the drag factor (see Paragraph 6)
- I_{xx}, I_{yy}, I_{zz} are the body moments of inertia around x, y, z respectively. Their derivation is explained in details in Ref. 1.

Equations of Motion :

Deriving from (1) and from all the forces and moments described above, we get the equations of motions in the earth-fixed frame²:

$$\left\{ \begin{array}{l} I_{xx}\ddot{\phi} = \dot{\theta}\dot{\psi}(I_{yy} - I_{zz}) + J_r\dot{\theta}\omega + dK_1(\omega_4^2 - \omega_2^2) \\ I_{yy}\ddot{\theta} = \dot{\phi}\dot{\psi}(I_{zz} - I_{xx}) - J_r\dot{\theta}\omega + dK_1(\omega_3^2 - \omega_1^2) \\ I_{zz}\ddot{\psi} = \dot{\phi}\dot{\psi}(I_{xx} - I_{yy}) + K_2(-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \\ m\ddot{x} = (s\psi s\phi + c\psi s\theta c\phi)K_1(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ m\ddot{y} = (-c\psi s\phi + s\psi s\theta c\phi)K_1(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ m\ddot{z} = c\psi c\phi K_1(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) - mg \end{array} \right. \quad (6)$$

With these equations, we'll be able determine the quadrotor's state by double-integrating its linear and angular accelerations.

The Simulink© flow is presented in figure 4

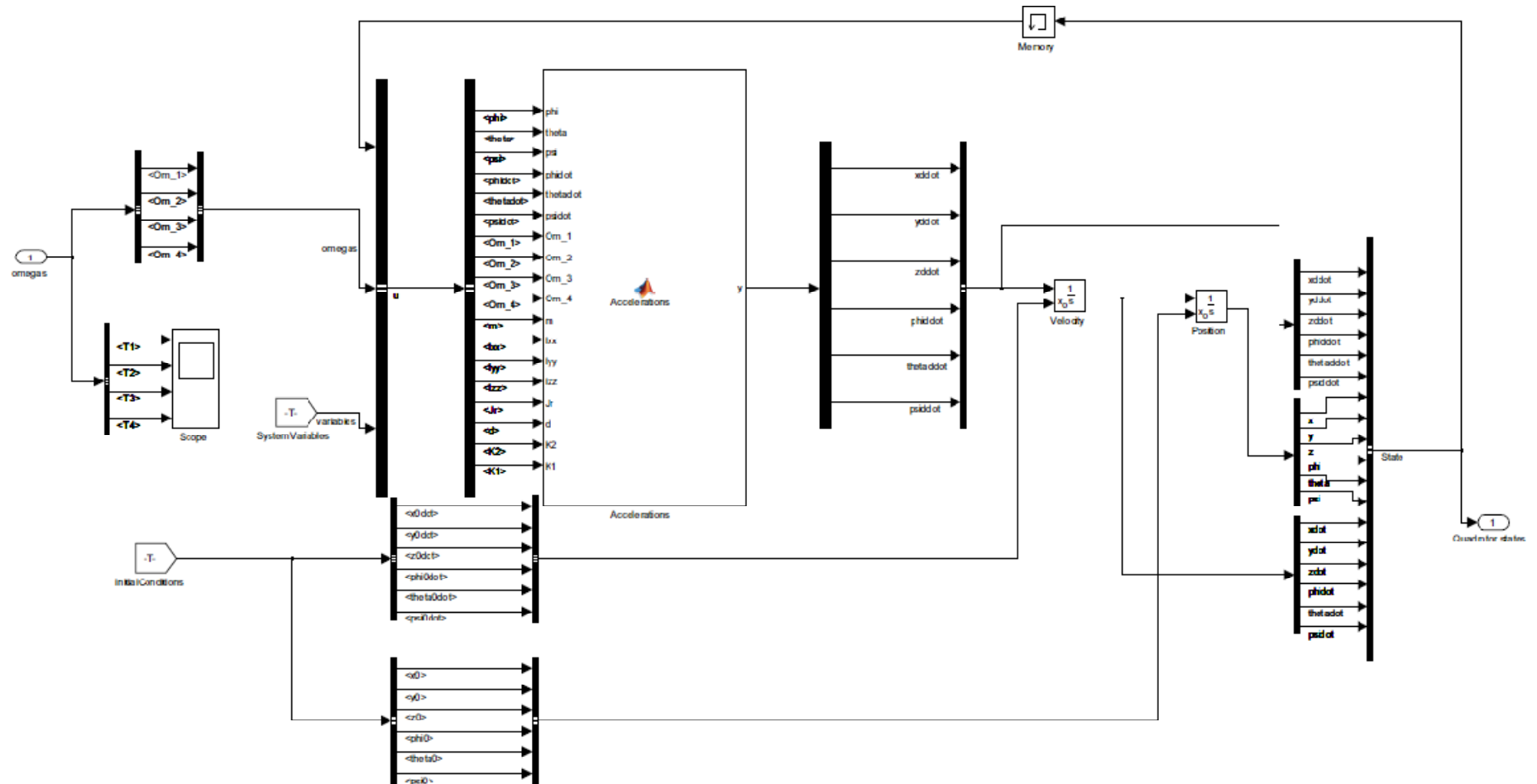


Figure 4 : System Dynamics Block

4. Control

The control algorithm that was derived tries to “invert” the equations of motion (6) presented above.

In order to use the equations of motion, some simplification and assumptions need to be made :

1. In this work, the quadrotor's motion can be considered close to hovering condition since small angular changes occur. Therefore, for control purposes, we'll cancel the gyro effects – body and gyro – on all axes.
2. For this work's purposes, only attitude (Euler angles) and height need to be controlled. The control technique used will therefore handle those four parameters only.

Basing ourselves on (6), we will chose four controllers, which will handle each of the parameters we wish to control : Height, Roll, Pitch and Yaw; respectively :

$$\begin{cases} U_1 = K_1 (\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ U_2 = dK_1 (\omega_4^2 - \omega_2^2) \\ U_3 = dK_1 (\omega_3^2 - \omega_1^2) \\ U_4 = K_2 (-\omega_1^2 + \omega_2^2 - \omega_3^2 + \omega_4^2) \end{cases} \quad (7)$$

Implementing the simplifications mentioned above into (6), we get¹ :

$$\begin{cases} \ddot{z} = g - c\psi c\phi \frac{U_1}{m} \\ \ddot{\phi} = \frac{U_2}{I_{xx}} \\ \ddot{\theta} = \frac{U_3}{I_{yy}} \\ \ddot{\psi} = \frac{U_4}{I_{zz}} \end{cases} \quad (8)$$

The actual control is performed using traditional PID technique, which attempts to minimize the error of the controlled parameter.

The PID structure is composed of a proportional, a derivative and an integral element, minimizing respectively the present error (P), the accumulation on past errors (I), and the prediction of future error (D).

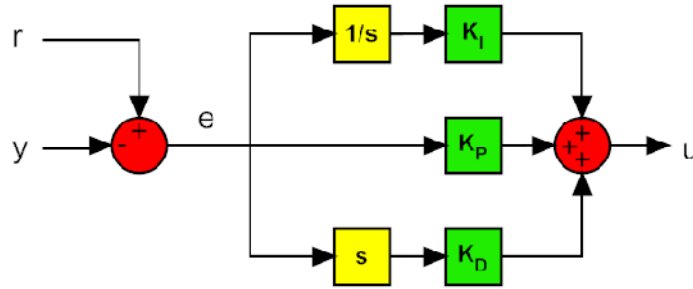


Figure 5 : General PID Block Diagram¹

For our purposes, the integrative element was ignored, and therefore, we get from (8) :

$$\begin{cases} U_1 = \frac{mg + (K_{P1} \cdot \Delta z + K_{D1} \cdot \Delta \dot{z})}{c\phi c\theta} \\ U_2 = I_{xx}(K_{P2} \cdot \Delta \phi + K_{D2} \cdot \Delta \dot{\phi}) \\ U_3 = I_{yy}(K_{P3} \cdot \Delta \theta + K_{D3} \cdot \Delta \dot{\theta}) \\ U_4 = I_{zz}(K_{P4} \cdot \Delta \psi + K_{D4} \cdot \Delta \dot{\psi}) \end{cases} \quad (9)$$

Where Δ represents the difference between the commanded and the estimated value of a parameter. The weights K_P, K_D - were tuned manually.

Once calculated, those controllers are converted back to rotors angular speeds with (7):

$$\begin{cases} \omega_1 = \sqrt{\frac{1}{4K_1}U_1 - \frac{1}{2dK_1}U_3 - \frac{1}{4K_2}U_4} \\ \omega_2 = \sqrt{\frac{1}{4K_1}U_1 - \frac{1}{2dK_1}U_2 + \frac{1}{4K_2}U_4} \\ \omega_3 = \sqrt{\frac{1}{4K_1}U_1 + \frac{1}{2dK_1}U_3 - \frac{1}{4K_2}U_4} \\ \omega_4 = \sqrt{\frac{1}{4K_1}U_1 + \frac{1}{2dK_1}U_2 + \frac{1}{4K_2}U_4} \end{cases} \quad (10)$$

Which are applied back into (6) to derive the next quadrotor's state.

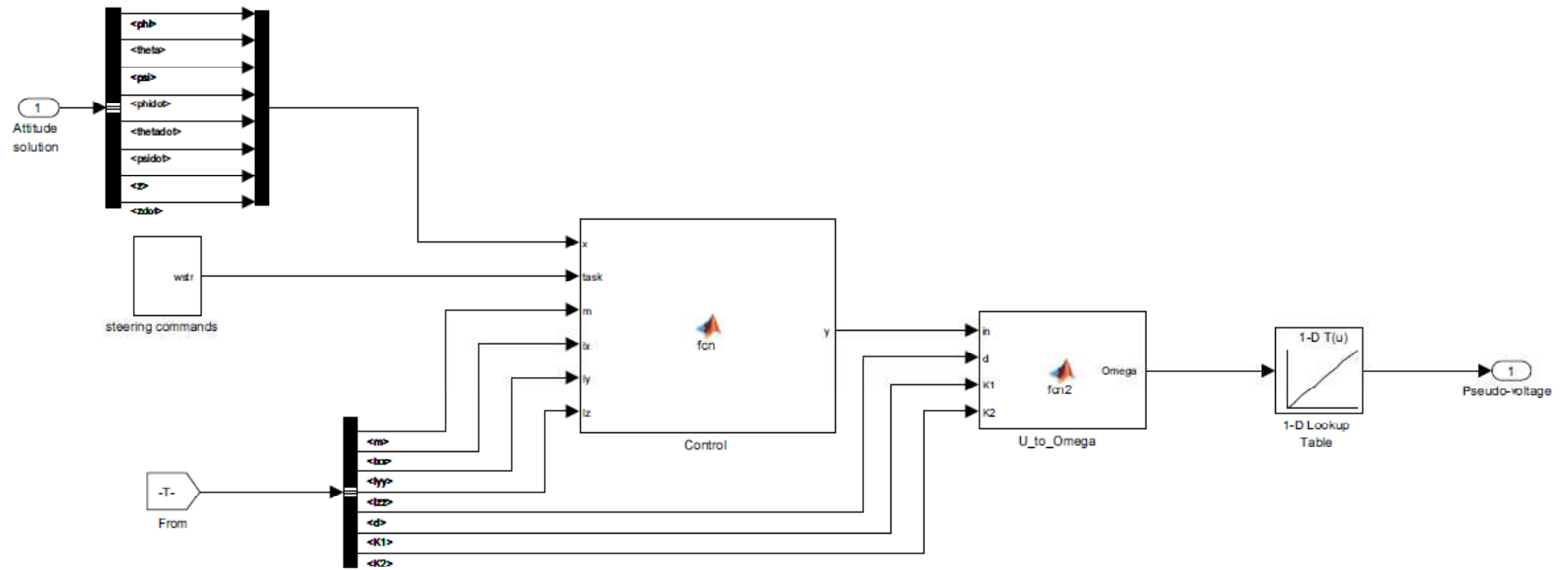


Figure 6 :Control Block

5. Sensor Reading and State Estimation

As introduced in Par. 2, the “Sensors Reading and State Estimation” block receives the calculated state and outputs an estimated state, which will be controlled in the “Control” block.

The simulated quadrotor uses a MEMS including a 3-axes accelerometer, a 3-axes magnetometer, a 3-axes gyroscope, a barometer and a temperature sensor. The sensors noises were simulated by adding to the calculated state a white Gaussian noise with a mean value of 0. Their variances (σ^2) were defined by recording and analyzing the outputs of a similar device (See Figure 7) in laboratory conditions, at rest.

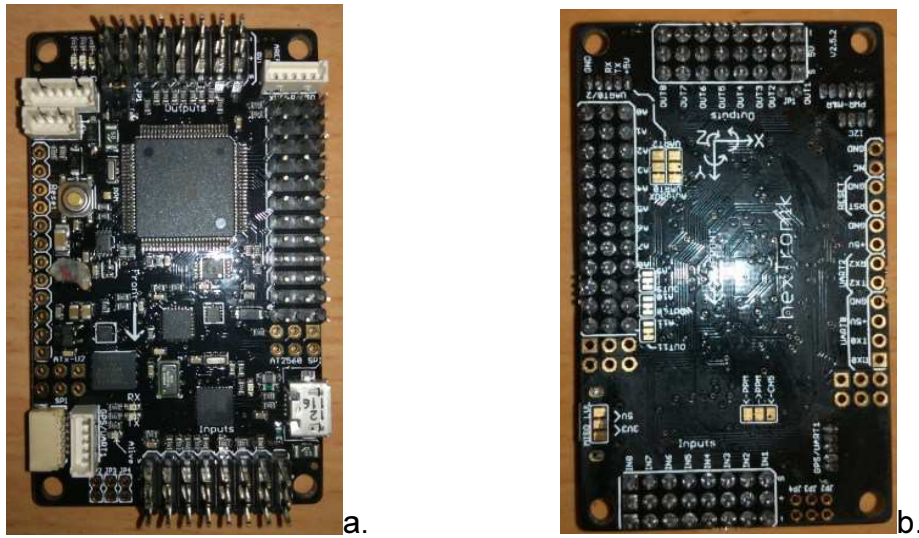


Figure 7. a./b.- HexTronik© 3-axes Accelerometer/Magnetometer/Gyroscope & Barometer used for noises variances determination

Pitch/Roll (θ, ϕ):

For θ, ϕ estimation, we define a “pseudo-acceleration” vector $\begin{bmatrix} a_x, a_y, a_z \end{bmatrix}^b$, based on the received quadrotor’s attitude only, ignoring linear acceleration applied on the body :

$$\text{Knowing that } \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}^b = R_e^b \cdot \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}^e \quad (11)$$

$$\text{where } \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}^e = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} [g] \quad (12)$$

$$\text{and } R_e^b = [R_e^b]^{-1} = [R_e^b]^T = \begin{bmatrix} c\psi c\theta & s\psi c\theta & -s\theta \\ c\psi s\theta s\phi - s\psi c\phi & s\psi s\theta s\phi + c\psi c\phi & c\theta s\phi \\ s\psi s\theta s\phi + c\psi s\theta c\phi & -c\psi s\phi + s\psi s\theta c\phi & c\psi c\phi \end{bmatrix} \quad (13)$$

we get

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}^b = \begin{bmatrix} s\theta \\ -c\theta s\phi \\ -c\psi c\theta \end{bmatrix} \quad (14).$$

Having determined the accelerometer noise's variance, we then add noise to (14) and reconvert it to θ and ϕ estimations:

$$\begin{cases} \theta = \text{atan2}\left(a_x, \sqrt{a_z^2 + a_y^2}\right) \\ \rho = -\text{atan2}\left(a_y, \sqrt{a_x^2 + a_z^2}\right) \end{cases} \quad (15)$$

Attitude (z):

Altitude is considered to be calculated from the barometer and Temperature sensors' output only:

$$P = P_0 e^{-\frac{gz}{RT}} \quad (16)$$

Where:

- P is the sensed static pressure in $[psi]$
- P_0 is the pressure at field altitude in $[psi]$
- R is the gaz constant and $R = 287 \left[\frac{J}{Kg \cdot K} \right]$
- T is the sensed Temperature in $[K]$

Having determined the barometer and temperature sensors' variances we then add noise to T and P and reconvert them to z using (16):

$$z = -\frac{RT}{g} \ln\left(\frac{P}{P_0}\right) \quad (17)$$

Angular velocities $\dot{\phi}, \dot{\theta}, \dot{\psi}$:

Noise were added to the angular velocities $(\dot{\phi}, \dot{\theta}, \dot{\psi})$ using the gyroscope's noise variances.

Yaw (ψ) and Vertical velocity (\dot{z}):

For the sake of simplicity, and to avoid the use of unnecessary elaborate state estimation techniques, the Yaw (ψ) and Vertical velocity (\dot{z}) noise's variances were determined by analysis of the Yaw and Vertical velocity output of the MEMS itself, which comprises an estimation algorithm of its own.

With the understanding that those are probably filtered parameters, we arbitrarily multiplied their variances by two.

A more accurate state estimation algorithm is kept for further development, if needed.

The measured variances are :

$$\begin{bmatrix} \sigma_{a_x}^2 & \sigma_{a_y}^2 & \sigma_{a_z}^2 & \sigma_{\psi}^2 & \sigma_p^2 & \sigma_T^2 & \sigma_{\dot{z}}^2 & \sigma_{\dot{\rho}}^2 & \sigma_{\dot{\theta}}^2 & \sigma_{\ddot{\psi}}^2 \end{bmatrix} = \begin{bmatrix} 1.05 & 0.93 & 1.56 & 0.47 \cdot 10^{-5} & 1.3 \cdot 10^{-4} & 38.2 & 0.12 & 0.29 \cdot 10^{-6} & 0.27 \cdot 10^{-6} & 0.22 \cdot 10^{-6} \end{bmatrix}$$

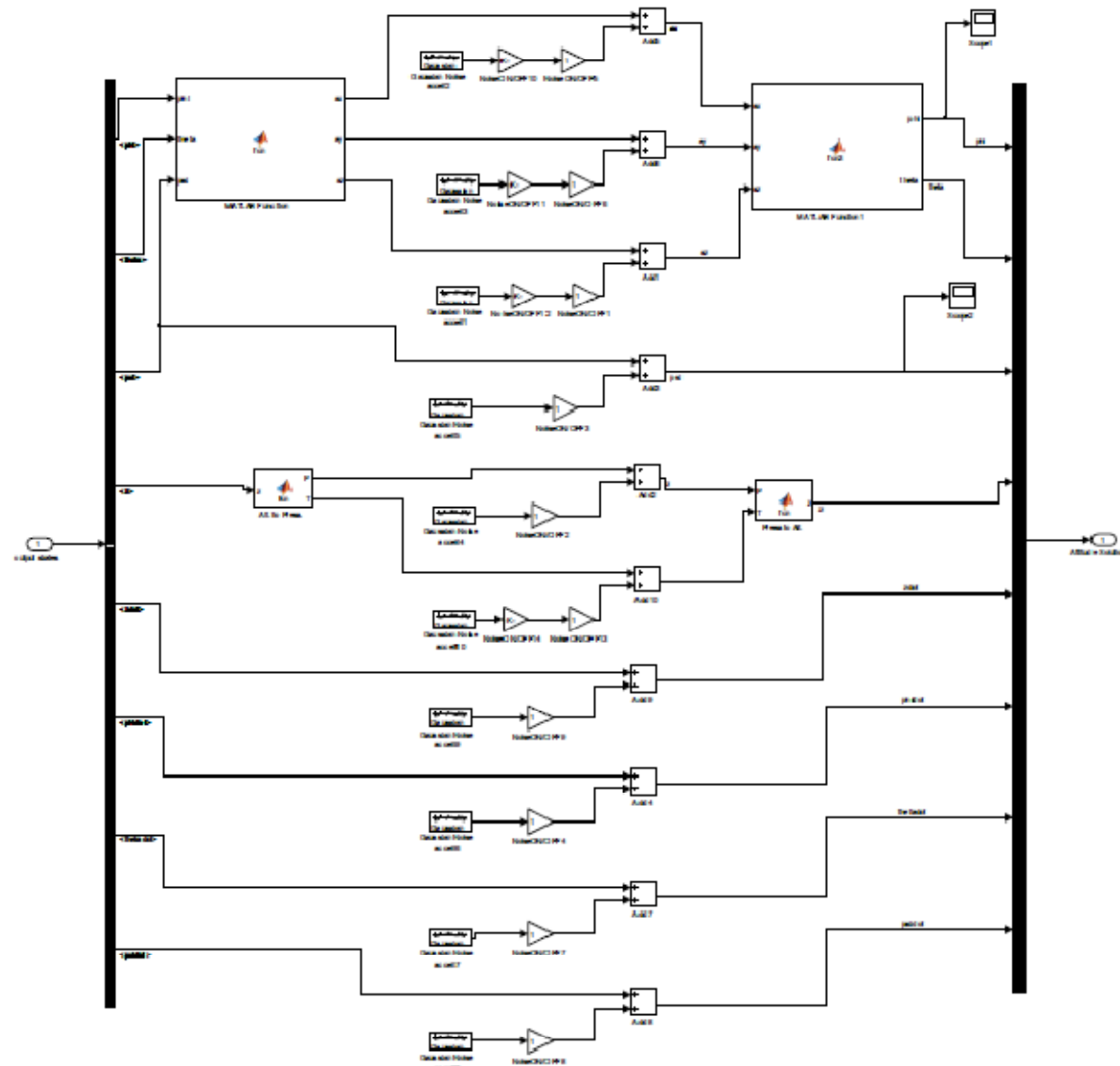


Figure 8 : Sensors Readings Block

6. Actuators

Four electric brushless rotors are used to power the quadrotor. In order to control it, we need to define:

1. The link between the thrust (T) provided by one rotor and its angular speed (ω_i)
2. The link between the rotors' angular speeds and the moments (Q) applied on the quadrotor.

Forces

The electrical power provided by one rotor is given by

$$P = IV \quad (18)$$

where V is the voltage across the motor in $[V]$ and I is the input current in $[A]$.

The Torque produced by one rotor is given by

$$\tau = K_t (I - I_0) \quad (19)$$

where K_t is the Torque-to-Current constant and I_0 is the current when there is no load on the rotor, in $[A]$.

The voltage drop across the motor is given by

$$V = IR_m + K_v \omega \quad (20)$$

where R_m is the motor resistance and K_v is a proportionality constant, indicating back EMF generated per RPM.

Defining $\tau = K_\tau T$, where K_τ is the Torque-to-Thrust constant; and setting (19) and (20) back into (18) while assuming⁷ $R_m = 0$ and $K_t I_0 \ll \tau$ we get

$$P \approx \frac{K_v K_\tau}{K_t} T \omega \quad (21)$$

Equivalently, the power needed to hover is equal to the Thrust times the air velocity (v_h) when passing through the rotor, at hover.

$$P = T v_h \quad (22)$$

Assuming the free stream velocity $v_\infty = 0$, momentum theory⁶ gives us

$$v_h = \sqrt{\frac{T}{2\rho A}} \quad (23)$$

where ρ is the air density in $\left[\frac{kg}{m^3}\right]$ and A is the area swept out by the rotor, in $[m^2]$.

From (22) and (23), we get

$$P = \frac{T^{\frac{3}{2}}}{\sqrt{2\rho A}} \quad (24)$$

Finally, from (24) and (21), we get

$$T = \underbrace{\frac{2\rho A K_v^2 K_\tau^2}{K_t^2}}_{K_1} \omega^2 \quad (25)$$

K_1 is called the thrust factor.

Moments

The yawing moment induced by the spinning of one rotor is partially due to the torque required by the rotors to counteract the drag forces applied on the propellers while spinning.

Assuming the force is applied at the tip of the blade, we can define the yawing moment applied by one rotor as

$$Q_y = R \cdot 2F_D \quad (26)$$

where F_D is the drag force applied on the rotor by one propeller and is defined

$$F_D = \frac{1}{2} \rho C_D A v^2 = \frac{1}{2} \rho C_D A (R\omega)^2 \quad (27)$$

where C_D is the propeller's drag coefficient.

From (26) and (27), we get

$$Q_y = \underbrace{C_D \rho A R^3}_{K_2} \omega^2 \quad (28)$$

K_2 is called the drag factor.

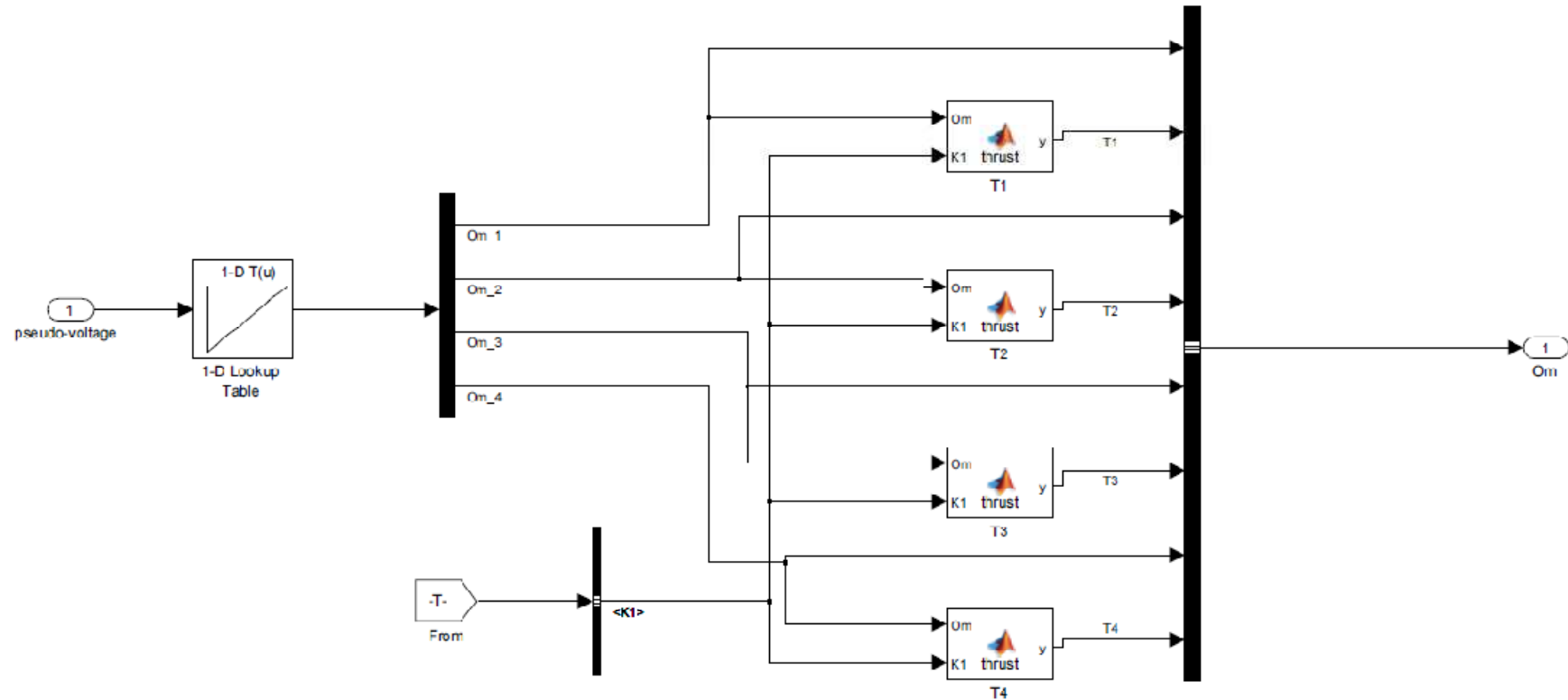


Figure 9 : Actuators Block

7. Constants and Variables

Symbol	Unit	Value	Description
K_1	$N\ s^2$	54.2e-6	Thrust factor
K_2	$N\ m\ s^2$	101.3e-7	Drag factor
g	$m\ s^{-2}$	9.81	Acceleration due to gravity
l	m	0.3	Distance from rotor center to quadrotor center
m	kg	1.7	Mass of the quadrotor
C_D	-	0.5	Propeller's drag coefficient
I_{xx}	$N\ m\ s^2$	8.1e-3	Body moment of inertia around x axis
I_{yy}	$N\ m\ s^2$	8.1e-3	Body moment of inertia around y axis
I_{zz}	$N\ m\ s^2$	14.2e-3	Body moment of inertia around z axis
ρ	$kg\ m^{-3}$	1.2	Air density
J_r	$N\ m\ s^2$	106e-6	Rotor moment of inertia
R	m	0.14	Rotor radius
P_0	psi	1013	Static field pressure

8. Simulation Results

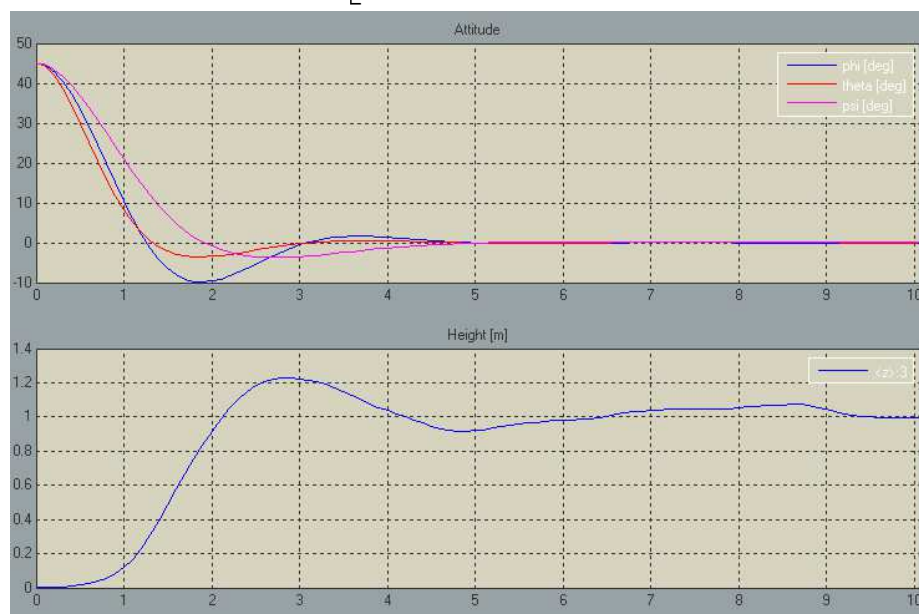
The simulations were performed inserting several different initial conditions with several different noise variances. The desired state was defined as “hover” at 1m height -

$$\begin{bmatrix} \rho & \theta & \psi & z \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}$$

Case 1 : (measured variances have been multiplied by 4)

$$\begin{bmatrix} \rho_0 & \theta_0 & \psi_0 & x_0 & y_0 & z_0 & \dot{\rho}_0 & \dot{\theta}_0 & \dot{\psi}_0 & \dot{x}_0 & \dot{y}_0 & \dot{z}_0 \end{bmatrix} = \begin{bmatrix} 45 & 45 & 45 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} \sigma_{a_x}^2 & \sigma_{a_y}^2 & \sigma_{a_z}^2 & \sigma_{\psi}^2 & \sigma_P^2 & \sigma_T^2 & \sigma_z^2 & \sigma_{\rho}^2 & \sigma_{\dot{\theta}}^2 & \sigma_{\dot{\psi}}^2 \end{bmatrix} = \begin{bmatrix} 4 & 4 & 6 & 2 \cdot 10^{-5} & 6 \cdot 10^{-4} & 160 & 0.5 & 1.2 \cdot 10^{-6} & 1.2 \cdot 10^{-6} & 10^{-6} \end{bmatrix}$$



Case 1:Convergence



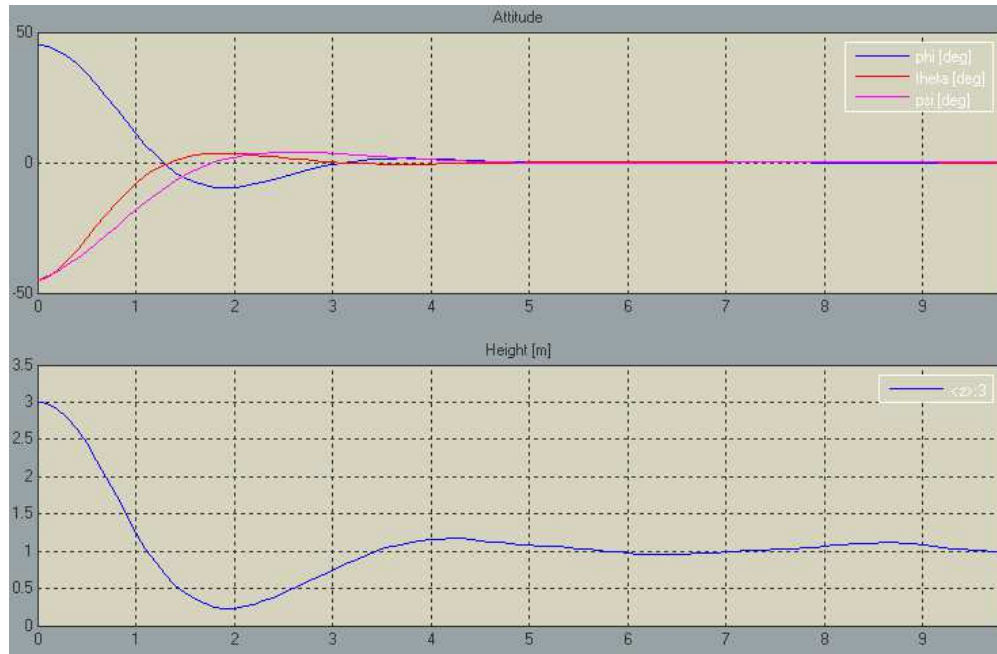
Case 1 : Steady State

Case 2 : (measured variances have been multiplied by 8)

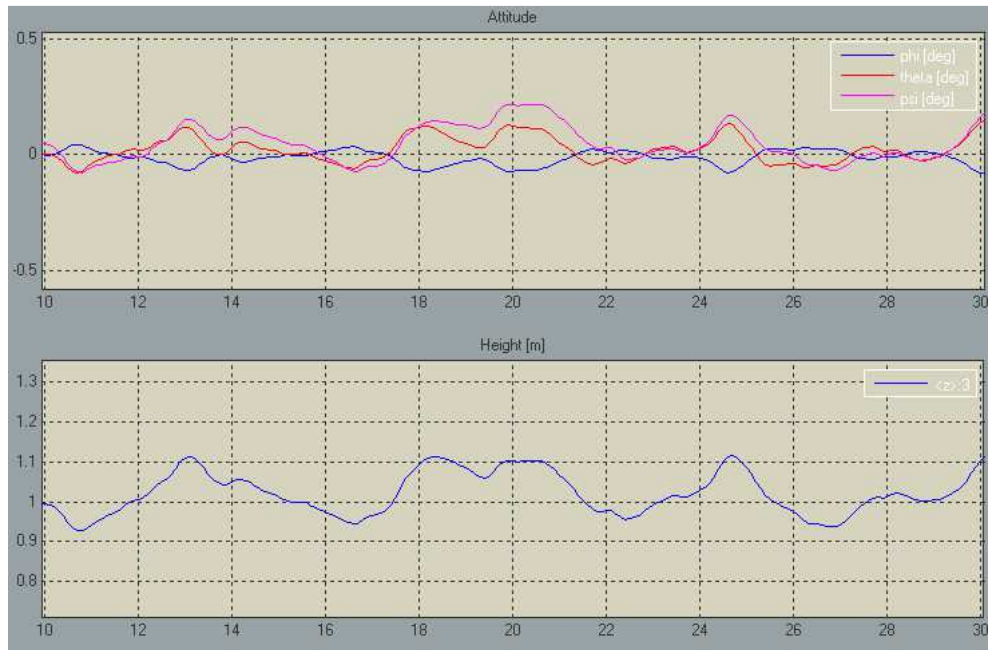
$$\begin{bmatrix} \rho_0 & \theta_0 & \psi_0 & x_0 & y_0 & z_0 & \dot{\rho}_0 & \dot{\theta}_0 & \dot{\psi}_0 & \dot{x}_0 & \dot{y}_0 & \dot{z}_0 \end{bmatrix} = \begin{bmatrix} 45 & -45 & -45 & 0 & 0 & 3 & 2 & 3 & 10 & 0 & 0 & 0.1 \end{bmatrix}$$

$$\begin{bmatrix} \sigma_{a_x}^2 & \sigma_{a_y}^2 & \sigma_{a_z}^2 & \sigma_{\psi}^2 & \sigma_p^2 & \sigma_T^2 & \sigma_z^2 & \sigma_{\dot{\rho}}^2 & \sigma_{\dot{\theta}}^2 & \sigma_{\dot{\psi}}^2 \end{bmatrix}$$

$$= \begin{bmatrix} 8 & 8 & 12 & 4 \cdot 10^{-5} & 12 \cdot 10^{-4} & 320 & 1 & 2.4 \cdot 10^{-6} & 2.4 \cdot 10^{-6} & 2 \cdot 10^{-6} \end{bmatrix}$$



Case 2 : Convergence

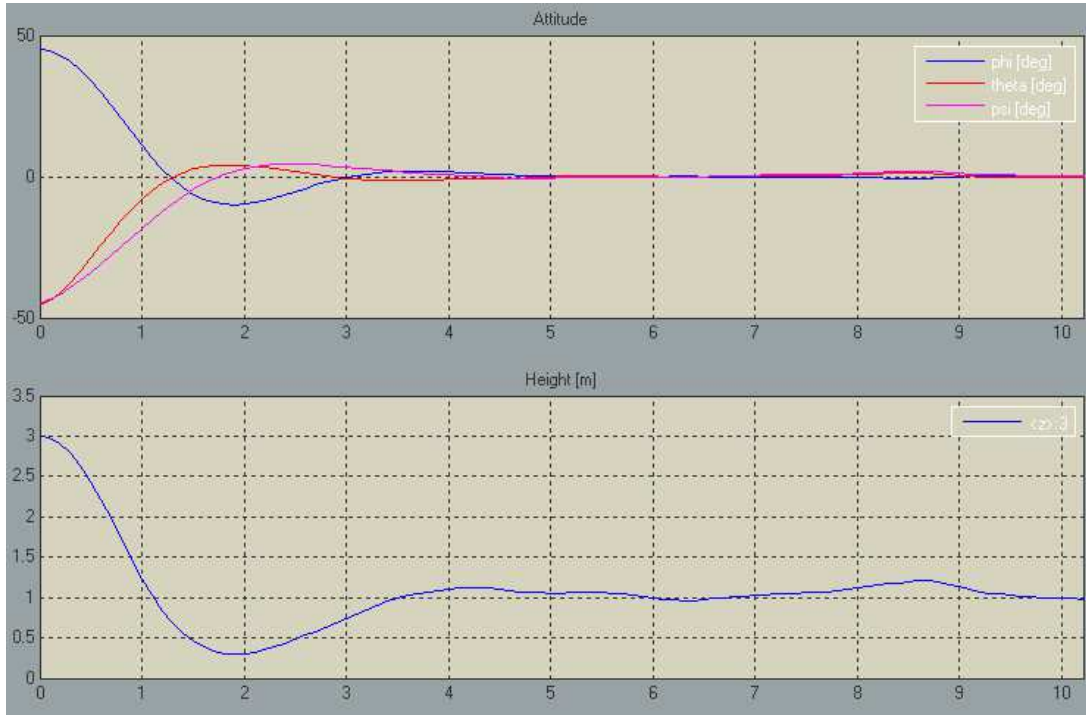


Case 2 : Steady State

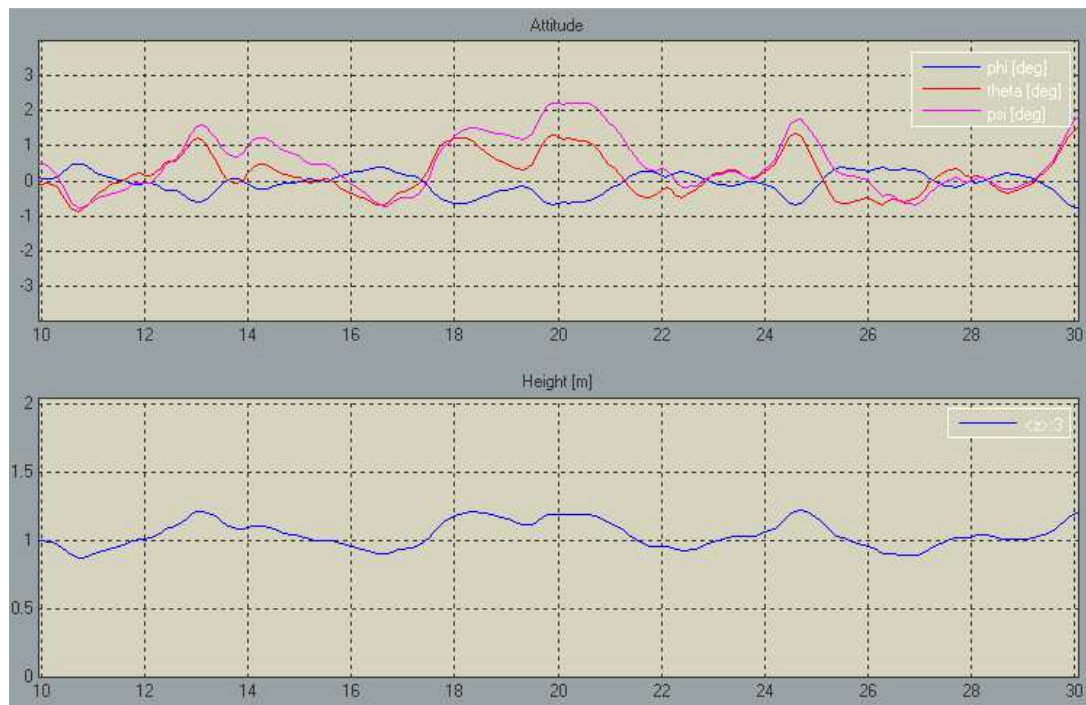
Case 3 : (Noises related to attitude have been multiplied by 30, noises related to altitude by 5)

$$\begin{bmatrix} \rho_0 & \theta_0 & \psi_0 & x_0 & y_0 & z_0 & \dot{\rho}_0 & \dot{\theta}_0 & \dot{\psi}_0 & \dot{x}_0 & \dot{y}_0 & \dot{z}_0 \end{bmatrix} = \begin{bmatrix} 45 & -45 & -45 & 0 & 0 & 3 & 2 & 3 & 10 & 0 & 0 & 0.1 \end{bmatrix}$$

$$\begin{bmatrix} \sigma_{a_x}^2 & \sigma_{a_y}^2 & \sigma_{a_z}^2 & \sigma_{\psi}^2 & \sigma_p^2 & \sigma_T^2 & \sigma_z^2 & \sigma_{\rho}^2 & \sigma_{\theta}^2 & \sigma_{\psi}^2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1.5 & 0.5 \cdot 10^{-5} & 13 \cdot 10^{-5} & 38 & 0.12 & 0.29 \cdot 10^{-6} & 0.28 \cdot 10^{-6} & 0.22 \cdot 10^{-6} \end{bmatrix}$$



Case 3 : Convergence



Case 3 : Steady State

9. References

1. "Modelling, Identification and Control of a Quadrotor Helicopter", T. Bresciani, Lund University, 2008
2. "Design and Control of Quadrotors with Application to Autonomous Flying", S. Bouabdallah, Ecole Polytechnique the Lausanne, 2007
3. "An Application of the Extended Kalman Filter to the Attitude Control of a Quadrotor", L. Ascorti, Politecnico Di Milano, 2012-2013
4. "Data Fusion of Accelerometric, Gyroscopic, Magnetometric and Barometric Information for Attitude and Altitude Estimation of an Unmanned Quadrocopter", R. Kesten, Technische Universitat Hamburg-Harburg, 2010.
5. "Principles of Helicopter Aerodynamics" Leishman, J. G., Cambridge University Press, New York, NY, 2000.
6. "Quadcopter Dynamics and Simulation", A. Gibiansky, blog
<http://andrew.gibiansky.com/blog/physics/quadcopter-dynamics/>

APPENDIX – Matlab Codes

```

function y =
Accelerations(phi,theta,psi,phidot,thetadot,psidot,Om_1,Om_2,Om_3,Om_4,m,Ixx,
Iyy,Izz,Jr,d,K2,K1)

omega1=Om_1;
omega2=Om_2;
omega3=Om_3;
omega4=Om_4;

U1=K1*(omega1^2+omega2^2+omega3^2+omega4^2);
U2=d*K1*(omega4^2-omega2^2);
U3=d*K1*(omega3^2-omega1^2);
U4=K2*(-omega1^2+omega2^2-omega3^2+omega4^2);
omega=(-omega1+omega2-omega3+omega4);

phiddot=(thetadot*psidot*(Iyy-Izz)-Jr*thetadot*omega+U2)/Ixx;
thetaddot=(phidot*psidot*(Izz-Ixx)+Jr*phidot*omega+U3)/Iyy;
psiddot=(thetadot*phidot*(Ixx-Izz)+U4)/Izz;
xddot=(sin(psi)*sin(phi)+cos(psi)*sin(theta)*cos(phi))*U1/m;
yddot=(-cos(psi)*sin(phi)+sin(psi)*sin(theta)*cos(phi))*U1/m;
zddot=(-m*9.81+cos(theta)*cos(phi)*U1)/m;

y = [xddot, yddot, zddot, phiddot, thetaddot, psiddot];

end

function y = Control(x,task,m,Ix,Iy,Iz)
% ?????????????????????????????????????????????????????????????
% This script applies the control laws to the input
% data inferred from the sensors to obtain the values
% of the four control variables Ui.
% ?????????????????????????????????????????????????????????????
%#codegen
% State inferred from the sensors
phi = x(1); % angular positions
theta = x(2);
psi = x(3);
dphi = x(4); % angular velocities
dtheta = x(5);
dpsi = x(6);
z = x(7);
dz = x(8);

% Task data
heightd = task(1);
rolld = task(2);
pitchd = task(3);
yawd = task(4);
% Control parameters
k1 = 4;
k2 = 2;
k3 = 4;
k4 = 2;
k5 = 4;

```

```

k6 = 2 ;
k7 = 2 ;
k8 = 2 ;

% Control laws
control_1 = -k1*(z-heightd)-k2*dz ;
U1 = (m*9.81+control_1)/(cos(phi)*cos(theta)) ;
U2 = Ix*(-k3*(phi-rolld) - k4* dphi) ;
U3 = Iy*(-k5*(theta-pitchd) - k6*dtheta) ;
U4 = Iz*(-k7 *(psi - yawd) - k8*dpsi) ;

y = zeros (4,1) ;
y(1) = U1 ;
y(2) = U2 ;
y(3) = U3 ;
y(4) = U4 ;

end

function Omega = U_to_Omega(in,d,K1,K2)
%#codegen
%
% This script converts the values o f the control
% variables Ui into the corresponding angular speeds .

% Control variables
U1=in(1) ;
U2=in(2) ;
U3=in(3) ;
U4=in(4) ;

W1=sqrt(abs(U1/(4*K1)-U3/(2*K1*d)-U4/(4*K2))) ;
W2=sqrt(abs(U1/(4*K1)-U2/(2*K1*d)+U4/(4*K2))) ;
W3=sqrt(abs(U1/(4*K1)+U3/(2*K1*d)-U4/(4*K2))) ;
W4=sqrt(abs(U1/(4*K1)+U2/(2*K1*d)+U4/(4*K2))) ;

Omega=[W1,W2,W3,W4];

function [ax,ay,az] = ang_to_acc(phi, theta, psi)

ax=sin(theta);
ay=-cos(theta)*sin(phi);
az=-cos(psi)*cos(theta);

end

function [phi, theta] = acc_to_ang(ax,ay,az)

theta = atan2(ax,sqrt(az^2+ay^2));
phi = -atan2 (ay,sqrt(ax^2+az^2));

end

```

```
function [P,T] = alt_to_press(z)
%#codegen

R=286.9; %gas cst
T=303.15; %kelvin temp
Po=1013; %std pressure
g=9.81;

P = Po*exp(-g*z/(R*T));
T=T-273.15;

end
```

```
function [P,T] = alt_to_press(z)
%#codegen

R=286.9; %gas cst
T=303.15; %kelvin temp
Po=1013; %std pressure
g=9.81;

P = Po*exp(-g*z/(R*T));
T=T-273.15;

end
```

```
function y = thrust(Om, K1)

y = K1*Om^2;

end
```