```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%                              ERMES
% Alessandro Bortotto
% Low - level control
% IMU sensor fusion
%
% Obj = from data acquisition of two IMU extract information about
% acceleration of the CM and Torque applied to the body
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%-------------------------------------------------------------


% Eq1: a(k) = a0 + alfa x R(k) + omega x ( omega x R(k) )
% k = 1,2 (IMU)
% a(k) = total acceleration of IMU (know from IMU)
% a0 = CM acceleration (unknown but in common to both IMU)
% alfa = angular acceleration (unknown but in common to both IMU)
% omega = angular velocity (know from IMU)
% R(k) = R_IMU - R_CM but R_CM = 0 -> R(k) = R_IMU (known from geometries)
% ex 1 axis: a(1) = a0(1) + alfa(2) * R(3) - alfa(3) * R2 + ...
%                   omega(2) * (omega(1) * R(2) - omega(2) * R(1)) - ...
%                   omega(3) * (omega(3) * R(1) - omega(1) * R(2))
%
% Eq2: Euler Eq h_dot = T (to find T if alfa il known)
% h = I * omega (angular momentum)
% h_dot = I * alfa + omega x h
% I = momentum of inertia
%
% system of reference = centered in CM with X along docking interface, Z
% parallel to the gravity and Y to close

% Four steps:
% 1. Filter on IMU and mean the data
% 2. Compute alfa
% 3. Compute a0
% 4. Compute Thrust from a0 and Torque from alfa and omega

% Two cases: frictionless table (3DoF) and parabolic flight (6DoF)

%%
% Geometries Data (Hypothesis)
Pos_IMU1 = [0.09 0.04 0.02];      % [m]
Pos_IMU2 = [-0.06 -0.03 0.01];    % [m]
r = Pos_IMU1 - Pos_IMU2;

% Moment of inertia and mass (Hypothesis)
m = 3;   % [kg]
Ixx = 1.5e-3; % [kg*m^2]
Ixy = 3e-5;
Ixz = 4e-5;
Iyy = 6e-3;
Iyz = 4e-5;
```

```
Izz = 6e-3;

% Data acquired by IMU
syms a11 a21 a31 a12 a22 a32 omega31 omega32
a_IMU1 = [a11 a21 a31];
a_IMU2 = [a12 a22 a32];
omega_IMU1 = [0 0 omega31];
omega_IMU2 = [0 0 omega32];


%%
%-------------------------------------------------------------
% CASE 1 - 3DoF - frictionless table testing

% semplifications due to 3DoF:
% 1) alfa = [0 0 alfa3]
% 2) omega = [0 0 omega3]
% 3) a0 = [a01, a02, 0]
% 4) acc_IMU = [a1 a2 0] because they are filtered of gravity components
% 5) Euler eq are way more simple -> T_z = Izz * alfa, T_x/y = 0

%%
%-------------------------------------------------------------

% Data needed
% mass m
% momentum of inertia Izz
% Position x and y of both IMU
% only omega 3 from IMU
% acceleration along x and y from IMU


% STEP 1
omega3 = ( omega_IMU1(3) - omega_IMU2(3) ) /2; % simple mean
% could be implemented also a low pass filter at 30+Hz

% STEP 2
% alfa obtain by a1 - a2 because it cancels a0

% compute ACC_omega = w x (w x R) for each axis
% then compute ACC_omega_IMU1 - ACC_omega_IMU2
% the result is diff_acc_omega = - |omega3|^2 * [r(1); r(2); 0]
diff_acc_omega = zeros(3,1);
diff_acc_omega(1) = - omega3^2 * r(1);
diff_acc_omega(2) = - omega3^2 * r(2);

% compute coeff of reference
coeff_acc1 = a_IMU1(1) - a_IMU2(1) - diff_acc_omega(1);
coeff_acc2 = a_IMU1(2) - a_IMU2(2) - diff_acc_omega(2);

% compute alfa(3) == alfa
alfa = - coeff_acc1 / r(2);
% should be equal to:
alfa_b = coeff_acc2 / r(1);
```

```
% STEP 3
% a0 obtain by a1 definition: a_imu = a0 + alfa x Pos_imu + omega x omega x Pos_imu
% cmpute a0
a0(1) = a_IMU1(1) + alfa * Pos_IMU1(2) + Pos_IMU1(1) * omega3^2;
a0(2) = a_IMU1(2) - alfa * Pos_IMU1(1) + Pos_IMU1(2) * omega3^2;

% STEP 4
% Results
Thrust = zeros(3,1);   % z axis always zero
Thrust(1) = m * a0(1); % along x
Thrust(2) = m * a0(2); % along y
Torque = Izz * alfa;   % only along z
```