Equations of Motion

6-DOF equations of motion are composed of or 3 translational equations of motion, and 3 angular equations of motion. We represent these equations as:

$$ {}^{B}F = m.(\ {}^{B}\dot{v}\ +\ {}^{B}\omega \times {}^{B}v\ ) $$

$$ {}^{B}M = {}^{B}I.{}^{B}\dot{\omega}\ +\ {}^{B}\omega \times ({}^{B}I\,.\,{}^{B}\omega) $$

where ${}^{B}X$ means that the components of X are expressed in the vehicles body fixed frame. To convert the body rates ${}^{B}\omega$ into Euler rates we'll use the following rotation sequence:

Rotation	START frame	End frame	Angular rate vector Associated with Rotation
R1Z(φ)	G-frame	a-frame	$\begin{pmatrix} 0 \\ 0 \\ \varphi\_DOT \end{pmatrix}$ `in G`
R2Y(θ)	a-frame	c-frame	$\begin{pmatrix} 0 \\ \theta\_DOT \\ 0 \end{pmatrix}$ `in a`
R3X(ψ)	c-frame	B-frame (the body frame)	$\begin{pmatrix} \psi\_DOT \\ 0 \\ 0 \end{pmatrix}$ `in c`

NOTE: a slightly more descriptive and verbose nomenclature for our 6-DOF equations of motion would be the following:

$$ {}^{B}F = m.(\ {}^{B}_{-}\dot{v}_C\ +\ {}^{B}_{G}\omega_B \times {}^{B}_{G}v_C\ ) $$

$$ {}^{B}M = {}^{B}I\,.{}^{B}_{-}\dot{\omega}_B\ +\ {}^{B}_{G}\omega_B \times ({}^{B}I\,.\,{}^{B}_{G}\omega_B) $$

where:

- ${}^{B}_{G}v_C$: A vector representing the vehicles ***velocity*** of the centre of mass C. The vector is expressed in components of the B-frame. The G subscript indicates that the "measurement" of the velocity is as seen by the G-frame.

- ${}^{B}_{-}\dot{v}_C = {}_{B}\left(\frac{d\ {}^{B}_{G}v_C}{dt}\right)$: the derivative of ${}^{B}_{G}v_C$ as seen by the B-frame, and expressed in components of the B-frame. So if we integrate ${}^{B}_{-}\dot{v}_C$, then we'll get ${}^{B}_{G}v_C$.

- ${}^{B}_{G}\omega_B$: the angular ***velocity*** of the B-frame as observed by the G-frame, and expressed in components of the B-frame.

- ${}^{B}_{-}\dot{\omega}_B = {}_B\left(\frac{d\ {}^{B}_{G}\omega_B}{dt}\right)$: the derivative of ${}^{B}_{G}\omega_B$ as seen by the B-frame, and expressed in components of the B-frame. So if we integrate ${}^{B}_{-}\dot{\omega}_B$, then we'll get ${}^{B}_{G}\omega_B$.

- ${}^{B}I$: the Inertia of the body computed about the B-frame which is attached to the body's center of mass.
- *B-frame* : the body fixed frame attached to the body's center of mass.
- G-*frame* : the inertial reference frame.
- A . B : matrix A multiplied by matrix B.
- $a \times b$: vector CROSS product.

Define vehicle Mass, Inertia and Initial Conditions

```
I = [ ...
            0.005831943165131, 0,                 0;
            0,                 0.005831943165131, 0;
            0,                 0,                 0.011188595733333;
          ]; % (kg.m^2)

m  = 0.9272;    % (kg)
g = 9.81;
% Vehicle INITIAL conditions
Vb_init     = [0;0;0]; % (m/sec)    Initial velocity in BODY axes
wb_init     = [0;0;0]; % (rad/sec)  Initial body rates
eul_init    = [0;0;0]; % (rad)      Initial EULER angles [yaw,pitch,roll]
Xe_init     = [0;0;0]; % (m)        Initial position in INERTIAl axes
```

```
% state vector INITIAL conditions
q_init = [ Vb_init;
           wb_init;
           eul_init;
           Xe_init   ];
```

define the ODE system to solve

we need to write a MATLAB function that defines the "state derivatives" of the system of interest. ie: I need to write a MATLAB function that represents a general system:

$$\dot{q} = f(t, q)$$

For our system we're going to define the following 12 element state vector $\mathbf{q}$ and the corresponsing 12 element vector of state derivatives $\dot{\mathbf{q}}$:

- $\mathbf{q} = \left({}^{B}v_x \ , \ {}^{B}v_y \ , \ {}^{B}v_z \ , \ {}^{B}\omega_x \ , \ {}^{B}\omega_y \ , \ {}^{B}\omega_z \ , \ \phi \ , \ \theta \ , \ \psi \ , \ {}^{G}X \ , \ {}^{G}Y \ , \ {}^{G}Z \right)$

- $\dot{\mathbf{q}} = \left({}^{B}\dot{v}_x \ , \ {}^{B}\dot{v}_y \ , \ {}^{B}\dot{v}_z \ , \ {}^{B}\dot{\omega}_x \ , \ {}^{B}\dot{\omega}_y \ , \ {}^{B}\dot{\omega}_z \ , \ \dot{\phi} \ , \ \dot{\theta} \ , \ \dot{\psi} \ , \ {}^{G}\dot{X} \ , \ {}^{G}\dot{Y} \ , \ {}^{G}\dot{Z} \right)$

```
% Define Excitation Forces and Moments (in BODY frame)
% simulate free fall assuming Z-axis is pointing down (NED)
Fb = zeros(3,1);
Mb = zeros(3,1);
Fb(3)=9.8*m; % due to gravity
Fb_and_Mb = [ Fb;
              Mb ];
FM_at_t   = @(t)(Fb_and_Mb);
```

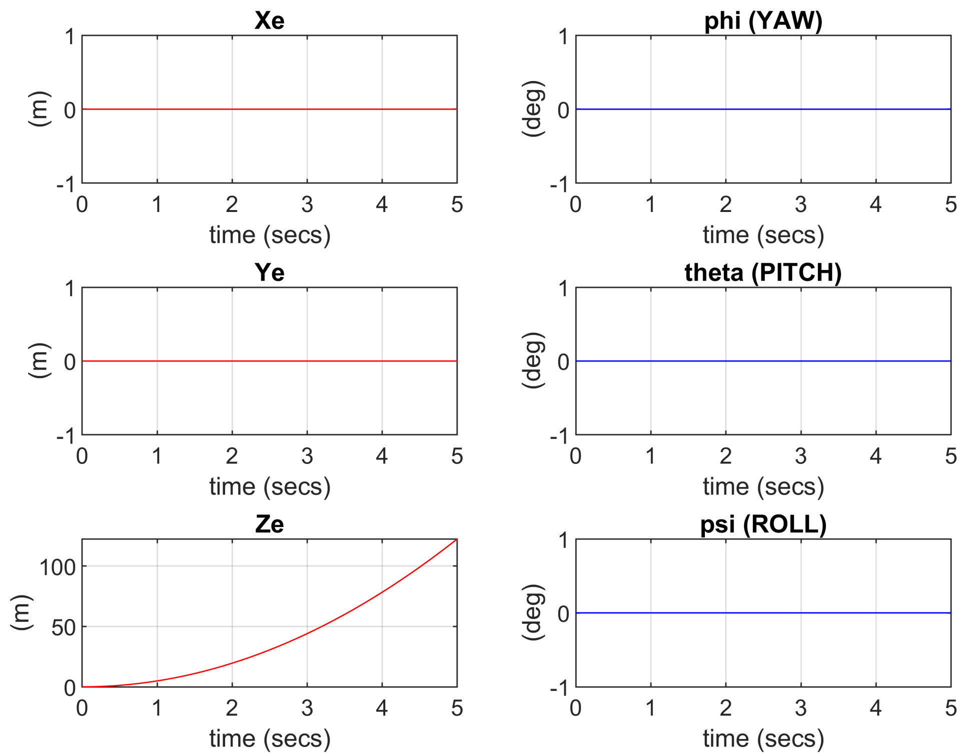
```
% define the system that we want to solve
dqdt_at_t = @(t, q) quad_6dof_eoms( t, q, m, I, FM_at_t );

% Define some ODE solver settings
t_span     = [0 5];   % (sec), [tstart, tend]
my_options = odeset('RelTol', 1e-7, 'AbsTol', 1e-7);

% OK: let's use  ODE solver
[T, Q]  = ode45(dqdt_at_t, t_span, q_init, my_options);
```

Plot the solution:

```
% plot our solution
figure;  plot_pose(T, Q)
```



Simulink simulation

The equations can be implemented in Simulink the same way.

```
open_system("quad_eom.slx");
sim('quad_eom.slx',5);
```

References

Bradley Horton : 01-Mar-2016, bradley.horton@mathworks.com.au