Table of Contents


```
function xdot = dynamics_CFwB(t,x,params)

% Usage: [tout,xout] = ode45(@(t,x) Coupledyn(t,x,params),tspan,x0,...
%
% Written by Garrett Ailts
%
% Description: Function takes in the current time, state, and a struct
 of
% simulation parameters for a continuouse rigid body's (CRB) angular
% dynamics using the quaternion or DCM representation and returns the
% derivative of the state vector
%
% Inputs:
%    t        -   time since t0 (s)
%    x        -   20 x 1 (quaternion) or 30 x 1 (DCM) state vector
%                 representing CRB's attitude and angular rates
%    params   -   struct containing CRB and simulation parameters
%
% Outputs:
%    xdot     -   20 x 1 or 30 x 1 vector containing the rates of change
 for the state
%                 paramters
%
```

Extract Parameters from Struct

sim

```
gg_model = params.gg_model;
mag_model = params.mag_model;
atm_model = params.atm_model;
SRP_model = params.SRP_model;
J2on = params.J2on;

% Earth
mu = params.Earth.mu_e;
```

```
R = params.Earth.Rmean;
J2 = params.Earth.J2const;
mag_epoch = params.Earth.mag_epoch;

% spacecraft
AttType = params.sc.Attitude_Type;
I = params.sc.IB_b;
start_epoch = params.sc.start_epoch;
mb = params.sc.mom_b;
est_method = params.sc.est_method;
ctrl_method = params.sc.ctrl_method;
kp = params.sc.kprop;
kd = params.sc.kderiv;
ke = params.sc.kerror;
kb = params.sc.kbias;
```

Useful Values

```
day2sec = 86400;
I3 = [0 0 1]';
r = norm(x(1:3));
if strcmp(AttType,'DCM')
    Cba = reshape(x(7:15),[3 3]);
    wba = x(16:18);
else
    Cba = Quat2DCM(x(7:10));
    wba = x(11:13);
end
wbaX = crossMatrix(wba);
```

Check for Earth Impact and J2 Inclusion

```
if r<=R
    warning('Earth impact!')
end
% Check For J2 Inclusion
if ~J2on
    J2 = 0;
end
```

Forces and Moments

gravity gradient torque

```
if gg_model
    tau_gg = (3*mu/r^5)*crossMatrix(Cba*x(1:3))*I*Cba*x(1:3);
else
    tau_gg = 0;
end

% magnetic moment
if mag_model
```

```
    telapsed = t+day2sec*(start_epoch-mag_epoch);
    b_a = EarthMagField(x(1:3),telapsed);
    tau_mag = crossMatrix(mb)*Cba*b_a;
else
    tau_mag = 0;
end

% atmospheric pressure force and torque
if atm_model
    [f_atm, tau_atm] = atmosphereMdl(t,x,Cba,params);
else
    f_atm = 0;
    tau_atm = 0;
end

% solar radiation pressure force
if SRP_model
    f_srp = 0; % placeholder for solar radiation pressure model
               % implementation
else
    f_srp = 0;
end

force = f_atm+f_srp; % sum of forces besides gravity of primary body
mom = tau_gg+tau_mag+tau_atm; % sum of moments imparted on spacecraft
```

Sensors

rate gyroscope

```
wbahat = RateGyroNoisy(wba,t);

% Earth horizon sensor
rpc_b = EarthSensorNoisy(-x(1:3),Cba,t);

if mag_model
    b_b = MagnetometerNoisy(b_a,Cba,t);
end
```

Navigation

```
if strcmp(est_method,'CF')
    CbaTRIAD = TRIAD(-x(1:3),b_a,rpc_b,b_b);
    bgyro = x(end-2:end); % estimate of gyro bias
    if strcmp(AttType,'DCM')
        % pull estimate from state vector and ensure its normalized
        Cea = reshape(x(19:27),[3 3]);
        CbeTRIAD = CbaTRIAD*Cea';
        e = uncross(CbeTRIAD-CbeTRIAD');
        CeaDot = -crossMatrix(wbahat-bgyro-ke*e)*Cea;
        bgyroDot = -kb*e;
    else
        qhat = x(14:17);
```

```
        qTRIAD = DCM2Quaternion(CbaTRIAD);
        qerr = 2*GammaQuat(qhat)'*qTRIAD;
        qhatDot = GammaQuat(qhat)*[(wbahat-bgyro
+ke*qerr(4)*qerr(1:3)); 0];
        bgyroDot = -kb*qerr(4)*qerr(1:3);
    end
elseif strcmp(est_method,'none')
    if strcmp(AttType,'DCM')
        % pull estimate from state vector and ensure its normalized
        CeaDot = zeros(3);
    else
        qhatDot = zeros(4,1);
    end
end
```

Control

```
if strcmp(ctrl_method,'PD ideal')
    % method a)
    if strcmp(AttType,'DCM')
        q = DCM2Quat(Cba);
    else
        q = x(7:10);
    end
    tau_ctrl = -kp*q(1:3)-kd*wba;
elseif strcmp(ctrl_method,'PD true')
    if strcmp(AttType,'DCM')
        qhat = DCM2Quat(Cea);
    end
    % method b)
    tau_ctrl = -kp*qhat(1:3)-kd*wbahat;
elseif strcmp(ctrl_method,'none')
    tau_ctrl = 0;
else
    error('Not a valid control method for this function!\n');
end

mom = mom+tau_ctrl;
```

Orbital Dynamics

Calculate xdot1 with gravity and other forces

```
xdot1 = [x(4:6); -mu*x(1:3)/r^3 + (3*mu*J2*R^2/2/r^5)*(((5/r^2)* ...
                 (x(1:3)'*I3)-1)*x(1:3)-2*(x(1:3)'*I3)*I3) + force];
```

Attitude Dynamics

```
if strcmp(AttType,'DCM')
    xdot2 = [-wbaX*x(7:9); -wbaX*x(10:12); -wbaX*x(13:15); ...
             I\(mom-wbaX*I*wba); CeaDot(:); bgyroDot];
    % DCM calc
```

```
else
    xdot2 = [GammaQuat(x(7:10))*[wba;0]; I\(mom-wbaX*I*wba); ...
             qhatDot; bgyroDot];
    % quaternion calculation
end
```

Package Dynamics Together

```
xdot = [xdot1; xdot2];

end
```

Published with MATLAB® R2019b