
Table of Contents

.....	1
Constants	1
Extract Parameters	1
Add Spacecraft MOI, Face Objects, and Center of Mass to Params	2
Atmospheric Input Data for Model	2
Assemble Initial Conditions for Spacecraft	2
Simulate	2
Post Process Data	3
Creat Plots	3

```
function results = MainSim(params)

% Usage: results = SpacecraftSim(params)
%
% Description: Function takes in a struct of simulation parameters and
% returns a struct containing all of the results of the simulation
%
% Inputs:
%   params - struct of parameters for spacecraft, planet, and
%           general
%           simulation parameters
%
% Outputs:
%   results - struct of simulation results
%
```

Constants

```
deg2rad = pi/180;
rad2deg = 1/deg2rad; %#ok<NASGU>
```

Extract Parameters

```
Sim

nOrbits = params.nOrbits;
absTol = params.absTol;
relTol = params.relTol;
atm_model = params.atm_model; %#ok<NASGU>

% Earth
mu = params.Earth.mu_e;
R = params.Earth.Rmean;
atmDen_md1 = params.Earth.atmDen_md1;

% Orbit
a = params.sc.sma;
```

```

e = params.sc.ecc;
inc = params.sc.inc*deg2rad;

% Spacecraft Initial Conditions
AttType = params.sc.Attitude_Type;
omega0 = params.sc.omega0;

```

Add Spacecraft MOI, Face Objects, and Center of Mass to Params

```

[IB_b, rcz] = scMOI(params.sc);
params.sc.IB_b = IB_b;
params.sc.rcz = rcz;
params.sc = loadFaces(params.sc);

```

Atmospheric Input Data for Model

```

if strcmp(atmDen_mdl,'Jacchia70')
    LoadJacchia70;
    params.Earth.jacchiaInput = indata;
end

```

Assemble Initial Conditions for Spacecraft

```

OMEGA = 0; % assume perigee at equator with RAAN = 0;
omega = 0;
theta = 0;

[r,v,~] = orbEl2rv(a, e, theta, OMEGA, omega, inc, mu); % transform
                                                         % orbital
                                                         % to pos. and
                                                         % vel.
elements

if strcmp(AttType,'quaternion')
    x0 = [r; v; params.sc.qba0; omega0; params.sc.qbahat0];
elseif strcmp(AttType,'DCM')
    x0 = [r; v; params.sc.Cba0(:); omega0; params.sc.Cbahat0(:)];
else
    error('Incorrect attitude type!\n');
end

```

Simulate

```

options = odeset('AbsTol',absTol,'RelTol',relTol);

T = 2*pi*sqrt(a^3/mu);
tspan = [0 nOrbits*T];

[tout,xout] = ode45(@(t,x) CoupledDyn(t,x,params),tspan,x0,options);

```

Post Process Data

```
xout = xout';
Post_Process_v2;
results.tout = tout;
results.xout = xout;
results.E = E;
results.eulerAngs = eulerAngs;
results.constraint = constraint;
results.x0 = x0;
```

Creat Plots

```
q = zeros(4,length(tout));
q_IN = zeros(4,length(tout));
h = zeros(1,length(tout));
if strcmp(AttType, 'DCM')
    for lv1 = 1:length(tout)
        q(:,lv1) = DCM2Quat(reshape(xout(7:15,lv1),[3 3]));
        q_IN(:,lv1) = DCM2Quat(reshape(xout(19:27,lv1),[3 3]));

    end
    wba = xout(16:18,:);    %#ok<NASGU>
else
    q = xout(7:10,:);    %#ok<NASGU>
    q_IN = xout(14:17,:);    %#ok<NASGU>
    wba = xout(11:13,:);    %#ok<NASGU>
end

% Altitude
for lv1 = 1:length(tout)
    h(lv1) = norm(xout(1:3,lv1))-R;
end
Plotter_v2;
```

Published with MATLAB® R2019b