Detumbling using B-Dot Law

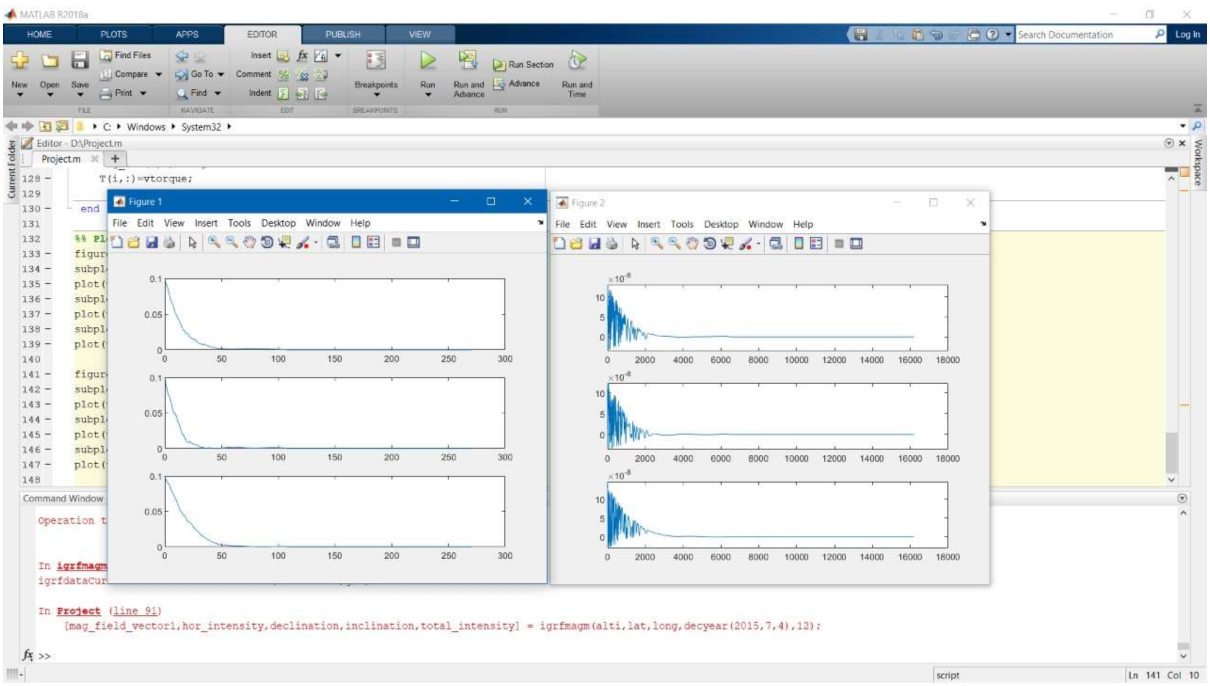
We have written a MATLAB code as shown below to generate example 7.5, comments in green starting with "%" explaining the code.

```
%bdot algorithm using lattitude longitude and altitude
for simulation
clear all; % syntax
clc;

%% Specify time
dt = 0.30; T = 90*60*3;% Step size and total time upto
which simulation will run

t = 0:dt:T;
%% Initializations as given in Example 7.5 and as
required
inertia=[6400,-76.4,-25.6;-76.4,4730,-40;-25.6,-40,8160]'
* 1e-7;
ini_pos = [1029.7743e3;6699.3469e3;3.7896e3]';% From the
Example 7.5
Rc = sqrt(dot(ini_pos, ini_pos)); %  Distance from earth
in metres

G = 6.67428e-11; % Earth gravitational constant
M = 5.972e24; % Earth mass
ang_vel = zeros(length(t),3);% Creating empty array for
storing values of angular velcity
T = zeros(length(t),3);% Empty array for Torque
cang = [0.01;0.01;0.01]';%initial angular velocity when
the satellite is launched
Re = 6371.2e3;

ini_pos1 = [1.0360e+06;6.7003e+06;8.1842e3]';
ini_vel = [6.2119e3;0.9524e3;4.3946e3]';
acc = ini_pos * (G * M / Rc^2) / sqrt(dot(-ini_pos, -
ini_pos));
lla = ecef2lla(ini_pos); % in meters
alti = lla(3);
linvel = sqrt(dot(ini_vel, ini_vel)); %linear velocity of
the satellite
normal = cross(ini_pos, ini_pos1);
normalDotK = normal(3) / sqrt(dot(normal, normal));
angleOfInclinationOfOrbit = acos(normalDotK);
```

```
angacc=[0;0;0]';
p=((111e3 * Rc)/Re);

J = 4726.01952;
timePeriod = 2*pi/sqrt(dot(cang, cang));

k = 4*pi*(1 +
sin(angleOfInclinationOfOrbit)*J)/timePeriod;% Gain using
the formula given
pos = ini_pos;
%% Main loop
 for i=1:length(t)

    lin_veli = linvel;
    posi = pos;
    cangi=cang;%cang is the current angular velocity
    lla = ecef2lla(posi);% function which returns
lattitude, longitutde and altitude for a given cordinate
    lat = lla(1);
    long = lla(2);
    alti = lla(3);
    alti = min(alti, 6.00000e5);

[mag_field_vector1,hor_intensity,declination,inclination,
total_intensity] =
igrfmagm(alti,lat,long,decyear(2015,7,4,4,56,36),12);%fun
ction returns Earth's Magnetic Field a given longitude,
latitude and altitude
    mag_field_vector2 = mag_field_vector1 * 1e-9;
    mag_field_vector2 = mag_field_vector2.';
    mag_field_vector =  mag_field_vector2;

    % B-dot
    detb =
sqrt(dot((mag_field_vector),(mag_field_vector)));
    m = ((-k)/detb*norm(mag_field_vector))*cross(cang,
mag_field_vector);

    vtorque = cross(m, mag_field_vector);
    angacc = vtorque * inv(inertia);

    cang = cangi - (angacc * dt); %calculates the new
angular velocity
    pos = posi + (lin_veli * dt); % change  in position
    linvel = lin_veli + (acc * dt);% change in linear
velocity
    acc = pos * (G * M / Rc^2) / sqrt(dot(-pos, -pos));
```

```
    disp('new angular velocity');
    disp(cang);
    ang_vel(i,:)=cang;
    T(i,:)=vtorque;

 end
%% Plots
figure(1)
subplot(3,1,1)
plot(t/60,ang_vel(:,1));
subplot(3,1,2)
plot(t/60,ang_vel(:,2));
subplot(3,1,3)
plot(t/60,ang_vel(:,3));
figure(2)
subplot(3,1,1)
plot(t,T(:,1));
subplot(3,1,2)
plot(t,T(:,2));
subplot(3,1,3)
plot(t,T(:,3));
```

Graphs



Here Figure 1 plots ω_1 , ω_2 and ω_3 while Figure 2 in the screenshot from MATLAB plots L_1 , L_2 and L_3

Equations

$$m = \frac{k}{\|B\|}\omega \times b$$

$$L = m \times B$$

$$\tau = I\alpha$$

$$\omega_f = \omega_i + \alpha t$$

$$a = \frac{GM}{R^2}$$

$$v_f = v_i + at$$