

```
%% Attitude Determination using Extended QUEST
%% Mohamed Temam Nasri
%% Matlab Implementation
%% University of Manitoba
%% Advisor: Prof. Witold Kinsner
%% October 23, 2012
%% Version 4.1
```

```
clear all
close all
clc
```

```
load sun_inertial_00001.csv
load igrf_inertial_00001.csv
load sun_005_body_00001.csv
load igrf_005_body_00001.csv
load quaternion_005_00001.csv
load quaternion_bi_005_00001.csv
```

```
tfile = 'dtusat1.TLE';
tle = readTLE(tfile);
[epoch_yr, epoch_m, epoch_d, epoch_h, epoch_min, epoch_s] = days2ymdhms(tle.epoch_year,
tle.epoch_day);
Ttdb = cal_Ttdb(epoch_yr, epoch_m, epoch_d, epoch_h, epoch_min, epoch_s);

jd1 = juliandate(epoch_yr, epoch_m, epoch_d, epoch_h, epoch_min, epoch_s);
```

```
R=[];B=[];A=[];S=[];P=[];
co_var=[];co_var1=[];co_var2=[];
q1=[];
q2=[];
q3=[];
q4=[];
t_sec=[];
theta=[];
angle_sun=[];
angle_mag=[];
```

```
sigma=[0.005,0.01];
sigma1 = 0.05; % Sun Sensor Noise(Unit Vector)
sigma2 = 0.05; % Magnetometer Noise (Unit Vector)
```

```
sigma_v2 = 5*1.e-6; % (rad/sec^(1/2)) sampled rate sensor noise
sigma_v =sigma_v2; % sigma v crassidis
sigma_u1 = 2*1.e-8; % (rad/sec^(3/2) bias drift
```

```
sigma_u = sigma_u1; % sigma u crassidis
```

```
alpha_not=1;
```

```
sun_true=[];
```

```
sun_meas=[];
```

```
mag_true=[];
```

```
mag_meas=[];
```

```
Ts=0.864;
```

```
q_4=[];
```

```
qs=[];
```

```
qs_EQA=[];
```

```
theta_EQA=[];
```

```
alphas=[];
```

```
% *****magnetic field*****
```

```
% IGRF model initializations 21001.5
```

```
[G,H] = IGRF2005; % IGRF coefficients for 2005
```

```
nmax = 13; % max degree of geopotential
```

```
mmax = 13; % max order of geopotential
```

```
Kschmidt = schmidt(nmax,mmax);
```

```
for i=0:1:5000
```

```
delta_sim=i*(0.0144);
```

```
jd=jd1+(i*0.00001);
```

```
R1=[sun_inertial_00001(i+1,2) sun_inertial_00001(i+1,3) sun_inertial_00001(i+1,4)];
```

```
R1=R1/(sqrt(sun_inertial_00001(i+1,2)^2+sun_inertial_00001(i+1,3)^2+sun_inertial_00001(i+1,4)^2));
```

```
R2=[igrf_inertial_00001(i+1,1) igrf_inertial_00001(i+1,2) igrf_inertial_00001(i+1,3)];
```

```
B1=[sun_005_body_00001(i+1,2) sun_005_body_00001(i+1,3)  
sun_005_body_00001(i+1,4)];
```

```
B1=B1/sun_005_body_00001(i+1,5);
```

```
B2=[igrf_005_body_00001(i+1,6) igrf_005_body_00001(i+1,7)  
igrf_005_body_00001(i+1,8)];
```

```
%% ***** Adding the Noise Values to body Frame ****
```

```

NOISE1 = sigma1*(eye(3,3)-B1'*B1)*randn([3,1]);
NOISE2 = sigma2*(eye(3,3)-B2'*B2)*randn([3,1]);

```

```

B1N=B1+NOISE1';
B2N=B2+NOISE2';

```

```

B=[B1N',B2N'];
R=[R1',R2'];

```

```

%% Define the K matrix and the P matrix

```

```

K = zeros(4,4);

```

```

BB = zeros(3,3);

```

```

F = zeros(3,3);

```

```

for j = 1:1:2

```

```

    bi = B(:,j);

```

```

    ri = R(:,j);

```

```

    Omega = [ 0    bi(3) -bi(2) bi(1)
              -bi(3) 0    bi(1) bi(2)
              bi(2) -bi(1) 0    bi(3)
              -bi(1) -bi(2) -bi(3) 0];

```

```

    Gamma = [ 0 -ri(3) ri(2) ri(1)
              ri(3) 0 -ri(1) ri(2)
              -ri(2) ri(1) 0 ri(3)
              -ri(1) -ri(2) -ri(3) 0];

```

```

    b_cross = [ 0 -bi(3) bi(2)
                bi(3) 0, -bi(1)
                -bi(2) bi(1) 0];

```

```

    BB = BB+sigma(j)^2*bi*ri';

```

```

    K = K + sigma(j)^2*Omega'*Gamma;

```

```

    F = F - b_cross*b_cross/sigma(j)^2;

```

```

end

```

```

%% Solve the Eigenvalue problem

```

```

%[EVECT, eig_value,cond_number] = condeig(K);

```

```

[EVECT, eig_value] = eig(K);

```

```

%% Select the maximum eigenvalue

```

```

EV = [eig_value(1,1);eig_value(2,2);eig_value(3,3);eig_value(4,4)];

```

```

E = [EV,EVECT'];

```

```

sortrows(E,1);

```

```

lambda_max = E(4,1);

```

```

q_quest = E(4,2:5)'

```

```

if i==0

```

```

    q_EQA=q_quest;

```

```

end

%% Starting the Extended Q Method
if(i~=0)
    sun_true=[sun_true; B1];
    sun_meas=[sun_meas; B1N];
    mag_true=[mag_true; B2];
    mag_meas=[mag_meas; B2N];

    qs=[qs;q_quest(4)];

    w_k=[quaternion_005_00001(i+1,6); quaternion_005_00001(i+1,7);
quaternion_005_00001(i+1,8)];
    w_k=w_k*pi/180;
    w_k=w_k;
    %% Adding the noise to the Actual Gyro Readings
    noise=(sigma_u+sigma_v)*randn([3,1]);
    w_k=w_k+noise;

    % zero order quaternion propagation
    OMEGA_w_k = omegaop(w_k);
    wmag = sqrt(w_k'*w_k);
    q_gyro =
((cos(wmag*Ts/2)*eye(4))+((1/wmag)*sin(wmag*Ts/2)*OMEGA_w_k))*q_EQA;
    q_gyro_mag=sqrt(q_gyro(1)^2+q_gyro(2)^2+q_gyro(3)^2+q_gyro(4)^2);
    q_gyro=q_gyro/q_gyro_mag;

    % q_k=q_triad;
    cross_prod=cross(B1,B2);
    mag_cross=cross_prod(1)^2+cross_prod(2)^2+cross_prod(3)^2;
    alpha=(1-mag_cross)*alpha_not;
    alphas=[alphas;alpha];
    q_EQA= (alpha*q_gyro)+((1-alpha)*q_quest);
    q_EQA_mag=sqrt(q_EQA(1)^2+q_EQA(2)^2+q_EQA(3)^2+q_EQA(4)^2);
    q_EQA=q_EQA/q_EQA_mag;
    if(q_EQA(4)<0)
        q_EQA=-q_EQA;

    end
    qs_EQA=[qs_EQA;q_EQA(4)];

    %qt=[quaternion_bi_005_00001(i+1,1);quaternion_bi_005_00001(i+1,2);quaternion_bi_005_00
001(i+1,3);quaternion_bi_005_00001(i+1,4)]

```

%% Error for the Q Method Only

```
qt=[quaternion_005_00001(i+1,2) quaternion_005_00001(i+1,3)
quaternion_005_00001(i+1,4) quaternion_005_00001(i+1,5)];
```

```
qe=[ qt(4) qt(3) -qt(2) qt(1);
    -qt(3) qt(4) qt(1) qt(2);
    qt(2) -qt(1) qt(4) qt(3);
    -qt(1) -qt(2) -qt(3) qt(4)]*[-q_quest(1);
    -q_quest(2);
    -q_quest(3);
    q_quest(4)];

qe_mag=sqrt(qe(1)^2+qe(2)^2+qe(3)^2+qe(4)^2);
qe=qe/qe_mag;
```

```
if(qe(4)<0)
    qe=-qe;
```

```
end
[e1,e2,e3,a]=Quaternion_to_Axis(qe(1),qe(2),qe(3),qe(4));
```

```
q_4=[q_4; qe(4)];
a=a*180/pi;
```

```
theta=[theta;a];
```

%% Error for EQA only

```
[e1,e2,e3,a]=Quaternion_to_Axis(q_EQA(1),q_EQA(2),q_EQA(3),q_EQA(4));
```

```
qe_EQA=[ qt(4) qt(3) -qt(2) qt(1);
    -qt(3) qt(4) qt(1) qt(2);
    qt(2) -qt(1) qt(4) qt(3);
    -qt(1) -qt(2) -qt(3) qt(4)]*[-q_EQA(1);
    -q_EQA(2);
    -q_EQA(3);
    q_EQA(4)];
```

```
%qe_ETA(4)=sqrt(1-qe_ETA(1)^2-qe_ETA(2)^2-qe_ETA(3)^2);
qe_EQA_mag=sqrt(qe_EQA(1)^2+qe_EQA(2)^2+qe_EQA(3)^2+qe_EQA(4)^2);
```

```

qe_EQA=qe_EQA/qe_EQA_mag;

if(qe_EQA(4)<0)
    qe_EQA=-qe_EQA;

end
[e1,e2,e3,a_EQA]=Quaternion_to_Axis(qe_EQA(1),qe_EQA(2),qe_EQA(3),qe_EQA(4));
a_EQA=a_EQA*180/pi;
theta_EQA=[theta_EQA;a_EQA];

ang=acos(dot(B2,B1));
% ang=ang*acos(dot(R2,S2));
angle_sun=[angle_sun; ang*180/pi];

ang=acos(dot(R2,R1));
angle_mag=[angle_mag; ang*180/pi];

    t_sec=[t_sec; i*0.0144];
    %% Covariance estimate
    P = inv(F);
    cov(1) = P(1,1);
    cov(2) = P(2,2);
    cov(3) = P(3,3);

end
end

er_quest=0;
er_EQA=0;

for j=1:1:length(t_sec)
    er_EQA=er_EQA+theta_EQA(j);
end
er_EQA=er_EQA/length(t_sec)
for j=1:1:length(t_sec)
    er_quest=er_quest+theta(j);
end
er_quest=er_quest/length(t_sec)

```

```

%% Printing results
figure(1)
%%title('Error Angle QUEST')
plot(t_sec,theta)
hold on
plot(t_sec,er_quest,'--')
xlabel('Minutes')
ylabel('Error Angle [Deg]')
hh= legend('Error','Average', 'Location','NorthEast')
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 10 10])
print -dtiff -r300 ErrAng_Q_deg.png

```

```

figure(2)
% title(' Measured and Reference Sun Data ');
plot(t_sec,sun_meas(:,1),'');
hold on
plot(t_sec,sun_true(:,1),'-');
xlabel('Minutes')
ylabel('Unit X')
hh= legend('Measured','True', 'Location','NorthEast');
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 10 10])
print -dtiff -r300 MeasRef_S_Q_unitX.png

```

```

figure(3)
plot(t_sec,sun_meas(:,2),'');
hold on
plot(t_sec,sun_true(:,2),'-');
xlabel('Minutes')
ylabel('Unit Y')
hh= legend('Measured','True', 'Location','NorthWest');
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 10 10])
print -dtiff -r300 MeasRef_S_Q_unitY.png

```

```

figure(4)
plot(t_sec,sun_meas(:,3),'');
hold on
plot(t_sec,sun_true(:,3),'-');
xlabel('Minutes')
ylabel('Unit Z')
hh= legend('Measured','True', 'Location','NorthEast');
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 10 10])
print -dtiff -r300 MeasRef_S_Q_unitZ.png

```

```

figure(5)
plot(t_sec,mag_meas(:,1),'');

```

```

hold on
plot(t_sec,mag_true(:,1),'-');
xlabel('Minutes')
ylabel('Unit X')
hh= legend('Measured','True', 'Location','NorthEast');
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 10 10])
print -dtiff -r300 MeasRef_B_Q_unitX.png

```

```

figure(6)
plot(t_sec,mag_meas(:,2),'-');
hold on
plot(t_sec,mag_true(:,2),'-');
xlabel('Minutes')
ylabel('Unit Y')
hh= legend('Measured','True', 'Location','NorthWest');
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 10 10])
print -dtiff -r300 MeasRef_B_Q_unitY.png

```

```

figure(7)
plot(t_sec,mag_meas(:,3),'-');
hold on
plot(t_sec,mag_true(:,3),'-');
xlabel('Minutes')
ylabel('Unit Z')
hh= legend('Measured','True', 'Location','SouthWest');
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 10 10])
print -dtiff -r300 MeasRef_B_Q_unitZ.png

```

```

figure(8)
plot(t_sec,angle_sun)
hold on
plot(t_sec,angle_mag,'r--')
xlabel('Minutes')
ylabel('Dot Product')
hh= legend('Sun','Mag', 'Location','NorthWest');
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 10 10])
print -dtiff -r300 BSangle_Q.png

```