

```

%% Attitude Determination using UKF
%% Mohamed Temam Nasri
%% Matlab Implementation
%% University of Manitoba
%% Advisor: Prof. Witold Kinsner
%% October 23, 2012
%% Version 4.1
clc
clear all
close all

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

%% Sampling Time
Ts=0.864;
%% Initializations

w_k_1=0;
p_quat=[];
p_c=[];
p_bias=[];
integration=0;
q_mean=0;
y_mean=0;
alpha=1e-3;           %default, tunable
ki=0;                  %default, tunable
beta=2;                %default, tunable
L=6;
lambda=alpha^2*(L+ki)-L; %scaling factor

%% setting up Matrixes for Graphs
R=[];B=[];A=[];S=[];P=[];
co_var=[];co_var1=[];co_var2=[];
t=[0:.01:20];
bias1=[];bias2=[];bias3=[];
theta_a=[];

wka=[];
wk=[];

```

```

k1=[];
k2=[];
bias=[0.1;0.1;0.1]*10^-6;

bias_e1=[];
bias_e2=[];
bias_e3=[];
si_a=[];
phi_a=[];
t_sec=[];
theta=[];
q1=[];q2=[];q3=[];q4=[];
qe1=[];qe2=[];qe3=[];qe4=[];
qa1=[];qa2=[];qa3=[];qa4=[];

%%
tlefile = 'dtusat1.TLE';
tle = readTLE(tlefile);
[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);

%% Defining Noise Terms

sigma_v1 =0.001; % (rad) Angle sensor noise (sigma n Crassidis).
sigma_n = sigma_v1; % crassidis
r = sigma_v1^2;
sigma_v2 = 5*1.e-6; % (rad/sec^(1/2)) sampled rate sensor noise( gyro 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

R_k=[r*eye(3,3) zeros(3);zeros(3) r*eye(3,3) ];

Q_k=(Ts/2)*[[(sigma_v^2-(1/6)*Ts^2)*eye(3)      zeros(3)];
            [zeros(3)                          sigma_u^2*eye(3)]]

```

```

%% Initial Kalman using any initial value to q_k_1
[qs, co_var]=kalman_triad (0,jd1, 0)

%q_k_1=[0; 0; 0; 1];
q_k_1=[-0.34684;-0.61613;-0.47316;0.52555];
%q_k_1=[qs(1) ;qs(2) ;qs(3); qs(4)]

%% Initial Bias

%b_k_1=[0.2;0.2; 0.2]*10^-6;
b_k_1=[0;0;0];
Dy=6; % Measurements
x_k = [q_k_1(1:3,1); b_k_1];
Dx=size(x_k,1);
L=Dx;
NSig=2*Dx+1;
w_k=[0.0001 ;0.0001; 0.0002];

%% Initial covariance matrix
P_k = [[1.e-1*eye(3) zeros(3)];
        [zeros(3) 1.e-3*eye(3)]]; % matrix 6*6

qmean=zeros(4,1);
q_update=q_k_1;
qe=zeros(4,13);
for i=0:1:3000
    sec=0000;
    delta_sim=i*(0.0144);
    x_k = [q_k_1(1:3,1); b_k_1];

    %% Calculating Weights
    W_m(1) = lambda/(L + lambda);
    W_c(1) = lambda/(L + lambda) + (1 - alpha^2 + beta);
    for j1=1:2*L
        W_m(j1+1,1) = 1/(2*(L + lambda));
        W_c(j1+1,1) = 1/(2*(L + lambda));
    end
    for j3=1:6
        sig_x1(:,j3) = quatprod([qe(1,1);qe(2,1);qe(3,1);sqrt(1-
qe(1,1)^2+qe(2,1)^2+qe(3,1)^2)],q_update);
        sig_x2(:,j3) = quatprod([-qe(1,1);-qe(2,1);-qe(3,1);sqrt(1-
qe(1,1)^2+qe(2,1)^2+qe(3,1)^2)],q_update);
    end
end

```

```

sig_q = [qmean sig_x1 sig_x2];

sig_x=chol((Dx+lambda)*(P_k+Q_k))';

sig_x=chol((Dx+lambda)*(P_k+Q_k))';
chi_sig_k1=x_k*ones(1,NSig)+[zeros(Dx,1) sig_x -sig_x];

sig_pts=[sig_q(1,:);sig_q(2,:);sig_q(3,:);chi_sig_k1(4,:);chi_sig_k1(5,:);chi_sig_k1(6,:)];

%% *****Inertial Reference Vectors*****
rsun=[sun_inertial_00001(i+1,2) sun_inertial_00001(i+1,3) sun_inertial_00001(i+1,4)];

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

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

%% ***** Body Frame Reference Vectors*****
mag_meas_b=[igrf_005_body_00001(i+1,6) ;igrf_005_body_00001(i+1,7);
igrf_005_body_00001(i+1,8)]; %unit len.

[Hterm_b resd_b dot_prod_m]=bvector_kalman_EKF(q_k_1,mag_meas_b,bvect);

sun_meas_b=[sun_005_body_00001(i+1,2); sun_005_body_00001(i+1,3);
sun_005_body_00001(i+1,4)];
sun_meas_b=sun_meas_b/sun_005_body_00001(i+1,5);

[Hterm_s resd_s dot_prod_s] = sun_vector_EKF( q_k_1,sun_meas_b,rsun);
resd=[resd_b;resd_s];
%% Predicted observation
y_predicted=zeros(6,13);
for j=1:13
[Hterm_b resd_b dot_prod_m]=bvector_kalman_EKF(sig_q(:,j),mag_meas_b,bvect);
[Hterm_s resd_s dot_prod_s] = sun_vector_EKF( sig_q(:,j),sun_meas_b,rsun);
y_predicted(:,j)=[resd_b;resd_s];
end
size(y_predicted)
Pyy=R_k; %(6x6) Measurement covariance
Pxx=Q_k;
Pxy=zeros(size(Q_k));
%% Calculate the mean of the predicted measurements

ymean=zeros(6,1);
xmean=zeros(6,1);

```

```

for j2=1:NSig
    ymean=ymean+W_m(j2,1)*y_predicted(:,j2);
size(sig_pts)
    xmean =xmean + W_m(j2,1)*sig_pts(:,j2);
end

qmean=xmean(1:3,1)/norm(xmean(1:3,1));
xmean=[qmean;xmean(4:6)];
for j=1:NSig

    xdif=sig_pts(:,j)- xmean;
    ydif=y_predicted(:,j)-ymean;

    Pxx=Pxx+ xdif*xdif'*W_c(j,1);

    Pyy=Py y+ydif*ydif'*W_c(j,1);

    Pxy= Pxy+xdif*ydif'*W_c(j,1);
%
end

%% Calculate Kalman Gain

K_k = Pxy/Py y; % Gain
K_k
K_k_q = [K_k(1,:); K_k(2,:); K_k(3,:)]; %3*6 matrix

K_k_b = [K_k(4,:); K_k(5,:); K_k(6,:)]; % 3*6 matrix

k1=[k1;K_k_q(1,1)];
k2=[k2;K_k_b(1,1)];

%% COMPUTE THE CORRECTION TERM
deltaq_up_red = K_k_q*resd ; % 3*1 matrix
deltab_up = K_k_b*resd ; % 3*1 matrix
%% UPDATE THE QUATERNION AND THE BIAS
mag_deltaq_up = sqrt(deltaq_up_red'*deltaq_up_red);
if(mag_deltaq_up > 4)
    %normalize
    deltaq_up_red = deltaq_up_red/mag_deltaq_up;
end

q_update = quatprod([deltaq_up_red;1],q_k_1);
mag_q_up = sqrt(q_update'*q_update);
if(mag_q_up > 1)
    %normalize

```

```

        q_update = q_update/mag_q_up;
    end

    b_k = b_k_1+deltab_up ;    %bk|k

%% Update of the covariance

    Pxx= Pxx - K_k*Pxy*K_k'; % update covaiance

%% UPDATE THE ESTIMATED TURN RATE USING THE NEW ESTIMATE FOR THE
BIAS
% Read body rates
    w_ka=[quaternion_005_00001(i+1,6);                    quaternion_005_00001(i+1,7);
quaternion_005_00001(i+1,8)];
    w_ka=w_ka*pi/180;
% Adding the noise to the Actual Gyro Readings
    noise=(sigma_u+sigma_v)*randn([3,1]);
    w_k_1=w_ka+noise+bias;

    wka=[wka; w_ka'];
    wk=[wk; w_k_1'];

% Remove Estimated Bias from the Gyro readings
    w_k_1=w_k_1-b_k;    % wk+1|k+1=wmk+1- bk+1|k+1

%% *****
%% *****
%%
%% PROPAGATION EQUATIONS
%%
%% *****
%% *****

Mag_w = norm(w_k_1);
psik = (sin(1/2*Mag_w*Ts)/Mag_w)*w_k_1;
zk = cos(1/2*Mag_w*Ts)*eye(3);

```

```

Omega = [ zk psik;
          -psik', cos(1/2*Mag_w*Ts) ];% Ref [3] 29
pred_q = Omega*q_update % Ref [3] 34

```

```

q_k_1=pred_q;

```

```

%store a copy of current estimates before going to next time step
w_k = w_k_1;

```

```

%% the mean of the propagated quaternion and observed data

```

```

%% PROPAGATE THE BIAS
b_k_1=b_k; % Next Cycle Bias Propagation b_k_1=bk+1|k

```

```

%% Error Calculation

```

```

bias_e1=[bias_e1;bias(1)-b_k(1)];
bias_e2=[bias_e2;bias(2)-b_k(2)];
bias_e3=[bias_e3;bias(3)-b_k(3)];

```

```

%q_actual=[quaternion_005_00001(i+1,2);quaternion_005_00001(i+1,3);quaternion_005_0000
1(i+1,4);quaternion_005_00001(i+1,5)];

```

```

q_actual=[quaternion_bi_005_00001(i+1,1);quaternion_bi_005_00001(i+1,2);quaternion_bi_00
5_00001(i+1,3);quaternion_bi_005_00001(i+1,4)];

```

```

qe=[ q_actual(4) q_actual(3) -q_actual(2) q_actual(1);
     -q_actual(3) q_actual(4) q_actual(1) q_actual(2);
     q_actual(2) -q_actual(1) q_actual(4) q_actual(3);
     -q_actual(1) -q_actual(2) -q_actual(3) q_actual(4)]*[-q_update(1);
                  -q_update(2);
                  -q_update(3);
                  q_update(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));

theta=[theta;a];

q1=[q1;q_update(1)];
q2=[q2;q_update(2)];
q3=[q3;q_update(3)];
q4=[q4;q_update(4)];

qa1=[qa1;q_actual(1)];
qa2=[qa2;q_actual(2)];
qa3=[qa3;q_actual(3)];
qa4=[qa4;q_actual(4)];

qe1=[qe1;qe(1)];
qe2=[qe2;qe(2)];
qe3=[qe3;qe(3)];
qe4=[qe4;qe(4)-1];

bias1=[bias1;b_k_1(1)];
bias2=[bias2;b_k_1(2)];
bias3=[bias3;b_k_1(3)];

t_sec=[t_sec; i*0.0144];
end
error=0;
error1=[];

for j=1:1:length(t_sec)
    error=error+theta(j);
end
error=error/length(t_sec)
for j=1:1:length(t_sec)
    error1=[error1;error];
end

```



```
% % ***** Output *****
```

```
figure(1)
title(' Estimated and Actual Values');
```

```
plot(t_sec,q3,'-')
hold on
plot(t_sec,qa3,'-')
axis([0 length(t_sec)*0.0144 -1 1]);
xlabel('Minutes')
ylabel('quaternion 3')
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 15 15])
print -dtiff -r300 q3.png
figure(2)
plot(t_sec,q4,'-')
hold on
plot(t_sec,qa4,'-')
axis([0 length(t_sec)*0.0144 -1 1]);
xlabel('Minutes')
ylabel('quaternion 4')
```

```
legend('Estimated','Actual')
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 10 10])
print -dtiff -r300 q4.png
```

```
figure(3)
```

```
title(' Estimated and Actual Values');
plot(t_sec,q1,'-')
hold on
plot(t_sec,qa1,'-')
axis([0 length(t_sec)*0.0144 -1 1]);
xlabel('Minutes')
ylabel('quaternion 1')
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 10 10])
print -dtiff -r300 q1.png
```

```
figure(4)
plot(t_sec,q2,'-')
hold on
plot(t_sec,qa2,'-')
axis([0 length(t_sec)*0.0144 -1 1]);
xlabel('Minutes')
```

```
ylabel('quaternion 2')
```

```
legend('Estimated','Actual')  
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 10 10])  
print -dtiff -r300 q2.png
```

```
figure(5)  
plot(t_sec,theta,'-')  
hold on  
plot(t_sec,error1,'--')  
xlabel('Minutes')  
ylabel('Error Angle (Rad)')  
hh= legend('Error','Average', 'Location','NorthWest')
```

```
set(gcf, 'paperunits', 'centimeters', 'paperposition', [0 0 8 6])  
print -dtiff -r300 error4.png
```