Spacecraft Attitude Estimation with Low-Cost Sensors

Dept. of Aerospace Engineering

Lab for Navigation, Control and Applications

Stephen Park

Date: 2015. 07. 30. Thur.

Contents

01 연구의 목적

02 자세결정 알고리즘의 모델링

03 시뮬레이션 결과

04 결론 및 계획

1 연구의 목적 (1/1)

CONTROLA

- 확장칼만필터를 적용한 자세추정 알고리즘의 연구 및 시뮬레이션.
- 연구 논문에서 수행하는 선형화 모델링, 확장칼만필터의 구성 및 적용 방법등을 배우고, 자세추정을 시뮬레이션을 수행하여 본다.

2 자세결정 알고리즘의 모델링(1/6)

CONTROLA

quaternion kinematics and properties

- quatenion 의 정의

$$\boldsymbol{q} \equiv \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}^T \equiv \begin{bmatrix} \boldsymbol{p}^T & q_4 \end{bmatrix}^T$$

$\boldsymbol{q}^T\boldsymbol{q} = 1$ – quaternion constraint.
each components are not independent!

- quatenion kinematics

$$\dot{\boldsymbol{q}} = \frac{1}{2}\Xi(\boldsymbol{q})\boldsymbol{\omega} = \frac{1}{2}\Omega(\boldsymbol{\omega})\boldsymbol{q}$$

$$\frac{1}{2}\Omega(\boldsymbol{\omega})\boldsymbol{q} = \frac{1}{2}\left[\boldsymbol{q}_w \otimes\right]\boldsymbol{q}$$

$$\boldsymbol{q}_w = \begin{bmatrix} \boldsymbol{\omega}^T & 0 \end{bmatrix}^T$$

- skew–symmetric matrix calculation

$$[\boldsymbol{p}\times] \equiv \begin{bmatrix} 0 & -p_3 & p_2 \\ p_3 & 0 & -p_1 \\ -p_2 & p_1 & 0 \end{bmatrix}$$

$$[\boldsymbol{q}\otimes]\boldsymbol{q}' = \begin{bmatrix} q_4\boldsymbol{I}_{3\times3} - [\boldsymbol{p}\times] & \boldsymbol{p} \\ -\boldsymbol{p}^T & q_4 \end{bmatrix}\boldsymbol{q}'$$

2 자세결정 알고리즘의 모델링(2/6)

CONTROLA

spacecraft motion kinematics and sensor models

- kinematics of motion of spacecraft

(Euler's equation)

$$\dot{\boldsymbol{\omega}} = \boldsymbol{J}^{-1}(-\boldsymbol{\omega}^{\times}\boldsymbol{J}\boldsymbol{\omega} + \boldsymbol{u} + \boldsymbol{\eta}_{\omega})$$

$$\boldsymbol{J} \equiv \begin{bmatrix} J_{xx} & 0 & 0 \\ 0 & J_{yy} & 0 \\ 0 & 0 & J_{zz} \end{bmatrix}$$ – spacecraft moment of inertia

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_x & \omega_y & \omega_z \end{bmatrix}^T$$ – input value

$$\boldsymbol{u} = \begin{bmatrix} u_x & u_y & u_z \end{bmatrix}^T$$

$$\boldsymbol{\eta}_{\omega} = \begin{bmatrix} \eta_{\omega_x} & \eta_{\omega_y} & \eta_{\omega_z} \end{bmatrix}^T$$ – independent white Gaussian noise process

$$E\langle \boldsymbol{\eta}_{\omega}(t) \rangle = 0$$

$$E\langle \boldsymbol{\eta}_{\omega}(t_1)\boldsymbol{\eta}_{\omega}^T(t_2) \rangle = \boldsymbol{I}_{3\times3}\sigma_{\omega}^2\delta(t_1 - t_2)$$

- rate–integrating gyro model

widely used model is

1st order Markov process

$$\boldsymbol{\omega} = \tilde{\boldsymbol{\omega}} - \boldsymbol{\beta} - \boldsymbol{\eta}_a$$

$$\dot{\boldsymbol{\beta}} = \boldsymbol{\eta}_b$$

$\tilde{\boldsymbol{\omega}}$ – measured observation

$\boldsymbol{\beta}$ – gyro bias

$\boldsymbol{\eta}_a, \boldsymbol{\eta}_b$ – have same properties as $\boldsymbol{\eta}_{\omega}$

- star tracker model

$$\boldsymbol{q}_s = \delta\boldsymbol{q}_e \otimes \boldsymbol{q}$$

$$\delta\boldsymbol{p}_e = \boldsymbol{\eta}_s$$

$\boldsymbol{\eta}_s$ – have same properties as $\boldsymbol{\eta}_{\omega}$

2 자세결정 알고리즘의 모델링(3/6)

CONTROLA

Linearized model to use Extended Kalman Filter

- definition of state small variations

$$\delta\boldsymbol{\omega} = \boldsymbol{\omega} - \hat{\boldsymbol{\omega}}$$

$$\delta\boldsymbol{q} = \boldsymbol{q} \otimes \hat{\boldsymbol{q}}^{-1}$$

$$\delta\boldsymbol{b} = \boldsymbol{b} - \hat{\boldsymbol{b}}$$

- spacecraft moment of inertia at linearized error spacecraft dynamics

$$\boldsymbol{J}_{\hat{\omega}} \equiv \begin{bmatrix} 0 & J_x\hat{\omega}_z & J_x\hat{\omega}_y \\ J_y\hat{\omega}_z & 0 & J_y\hat{\omega}_x \\ J_z\hat{\omega}_y & J_z\hat{\omega}_x & 0 \end{bmatrix}$$

$$J_x = \frac{J_{yy} - J_{zz}}{J_{xx}}, \quad J_y = \frac{J_{zz} - J_{xx}}{J_{yy}}, \quad J_z = \frac{J_{xx} - J_{yy}}{J_{zz}}$$

2 자세결정 알고리즘의 모델링(4/6)

CONTROLA

Linearized model to use Extended Kalman Filter

- continuous-time error state-space equation

$$\delta \dot{\mathbf{x}} \equiv \mathbf{A}\delta \mathbf{x} + \mathbf{B}\delta \mathbf{u} + \mathbf{G}\boldsymbol{\eta}$$

$$\delta \mathbf{x} = \left[\delta \boldsymbol{\omega}^T, \delta \mathbf{p}^T, \delta \mathbf{b}^T\right]^T \in R^9$$

$$\boldsymbol{\eta} = \left[\boldsymbol{\eta}_\omega{}^T, \boldsymbol{\eta}_p{}^T, \boldsymbol{\eta}_b{}^T\right]^T \in R^9$$

$$\mathbf{A} = \begin{bmatrix} \mathbf{J}_{\hat{\omega}} & 0_{3\times3} & 0_{3\times3} \\ \frac{1}{2}\mathbf{I}_{3\times3} & -\hat{\boldsymbol{\omega}}^\times & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times3} \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} \mathbf{J}^{-1} \\ 0_{3\times3} \\ 0_{3\times3} \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} \mathbf{J}^{-1} & 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & \frac{1}{2}\mathbf{I}_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & 0_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix}$$

$Q = diag\left[\sigma_\omega^2 \mathbf{I}_{3\times3} \;\; \sigma_p^2 \mathbf{I}_{3\times3} \;\; \sigma_b^2 \mathbf{I}_{3\times3}\right]$ – covariance matrix

- measurement model

$$\tilde{\boldsymbol{\omega}}_g = \boldsymbol{\omega} + \boldsymbol{\beta} + \boldsymbol{\eta}_a$$

$$\mathbf{q}_s = \delta \mathbf{q}_e \otimes \mathbf{q}$$

$$\delta \mathbf{y}_k = \mathbf{H}_k \delta \mathbf{x}_k + \mathbf{v}_k$$

$$\delta \mathbf{y}_k = \left[\delta \boldsymbol{\omega}_{gk}^T, \delta \mathbf{p}_k^T\right]^T$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{I}_{3\times3} & 0_{3\times3} & \mathbf{I}_{3\times3} \\ 0_{3\times3} & \mathbf{I}_{3\times3} & 0_{3\times3} \end{bmatrix}$$

$$\mathbf{v}_k = \begin{bmatrix} \boldsymbol{\eta}_{a_k} \\ \boldsymbol{\eta}_{b_k} \end{bmatrix}$$

$\hat{\tilde{\boldsymbol{\omega}}}_g = \hat{\boldsymbol{\omega}} + \hat{\boldsymbol{\beta}}$ – not measurement model but it is necessary to innovation process

2 자세결정 알고리즘의 모델링(5/6)

CONTROLA

Extended Kalman Filtering

- State and Covariance Propagation

$$\hat{\boldsymbol{\omega}}_{k+1}^{-} = \tilde{\boldsymbol{\omega}}_g - \hat{\boldsymbol{b}}_{k+1}^{-}$$

$$\hat{\boldsymbol{q}}_{k+1}^{-} = \bar{\Omega}(\hat{\boldsymbol{\omega}}_k^{+})\hat{\boldsymbol{q}}_k^{+}$$

$$\Omega(\hat{\boldsymbol{\omega}}_k^{+}) \equiv \begin{bmatrix} cos\left(\frac{1}{2}\left\|\hat{\boldsymbol{\omega}}_k^{+}\right\|\Delta t\right) I_{3\times3} - \left[\hat{\psi}_k^{+}\times\right] & \hat{\psi}_k^{+} \\ -\psi_k^{+T} & cos\left(\frac{1}{2}\left\|\hat{\boldsymbol{\omega}}_k^{+}\right\|\Delta t\right) \end{bmatrix}$$

$$\hat{\psi}_k^{+} \equiv \frac{sin\left(\frac{1}{2}\left\|\hat{\boldsymbol{\omega}}_k^{+}\right\|\Delta t\right)\hat{\boldsymbol{\omega}}_k^{+}}{\left\|\hat{\boldsymbol{\omega}}_k^{+}\right\|}$$

$$\hat{\boldsymbol{\beta}}_{k+1}^{-} = \hat{\boldsymbol{\beta}}_k^{+} \quad because \ \dot{\hat{\boldsymbol{\beta}}} = 0$$

$$\dot{\boldsymbol{P}}(t) = \boldsymbol{A}(t)\boldsymbol{P}(t) + \boldsymbol{P}(t)\boldsymbol{A}^T(t) + \boldsymbol{G}(t)\boldsymbol{Q}(t)\boldsymbol{G}^T(t)$$

- Calculate Kalman Gain

$$\boldsymbol{K}_k = \boldsymbol{P}_k^{-}\boldsymbol{H}_k^T \times \left[\boldsymbol{H}_k\boldsymbol{P}_k^{-}\boldsymbol{H}_k^T + \boldsymbol{R}\right]^{-1}$$

2 자세결정 알고리즘의 모델링(6/6)

CONTROLA

Extended Kalman Filtering

- State Innovation

$$\delta\boldsymbol{\omega}_g = \tilde{\boldsymbol{\omega}}_g - \hat{\tilde{\boldsymbol{\omega}}}_g$$

$$\delta\boldsymbol{q}_{k+1} = \boldsymbol{q}_s \otimes \hat{\boldsymbol{q}}_{k+1}^{-}$$

$$\delta\boldsymbol{y}_{k+1} \equiv \left[\delta\boldsymbol{\omega}_g^T \;\; \delta\boldsymbol{p}_{k+1}^T\right]^T$$

$$\delta\hat{\boldsymbol{x}}_{k+1} = \boldsymbol{K}_{k+1}\delta\boldsymbol{y}_{k+1}$$

$$\delta\hat{\boldsymbol{x}}_{k+1} = \left[\delta\hat{\boldsymbol{\omega}}_{k+1}^T \;\; \delta\hat{\boldsymbol{p}}_{k+1}^T \;\; \delta\hat{\boldsymbol{b}}_{k+1}^T\right]^T$$

$$\delta\hat{\boldsymbol{q}}_{k+1} = \left[\delta\boldsymbol{p}_{k+1}^T \;\; \sqrt{(1-\delta\boldsymbol{p}_{k+1}^T\delta\boldsymbol{p}_{k+1})}\right]^T$$

- State and Covariance Update

$$\hat{\boldsymbol{\omega}}_{k+1}^{+} = \hat{\boldsymbol{\omega}}_{k+1}^{-} + \delta\hat{\boldsymbol{\omega}}_{k+1}$$

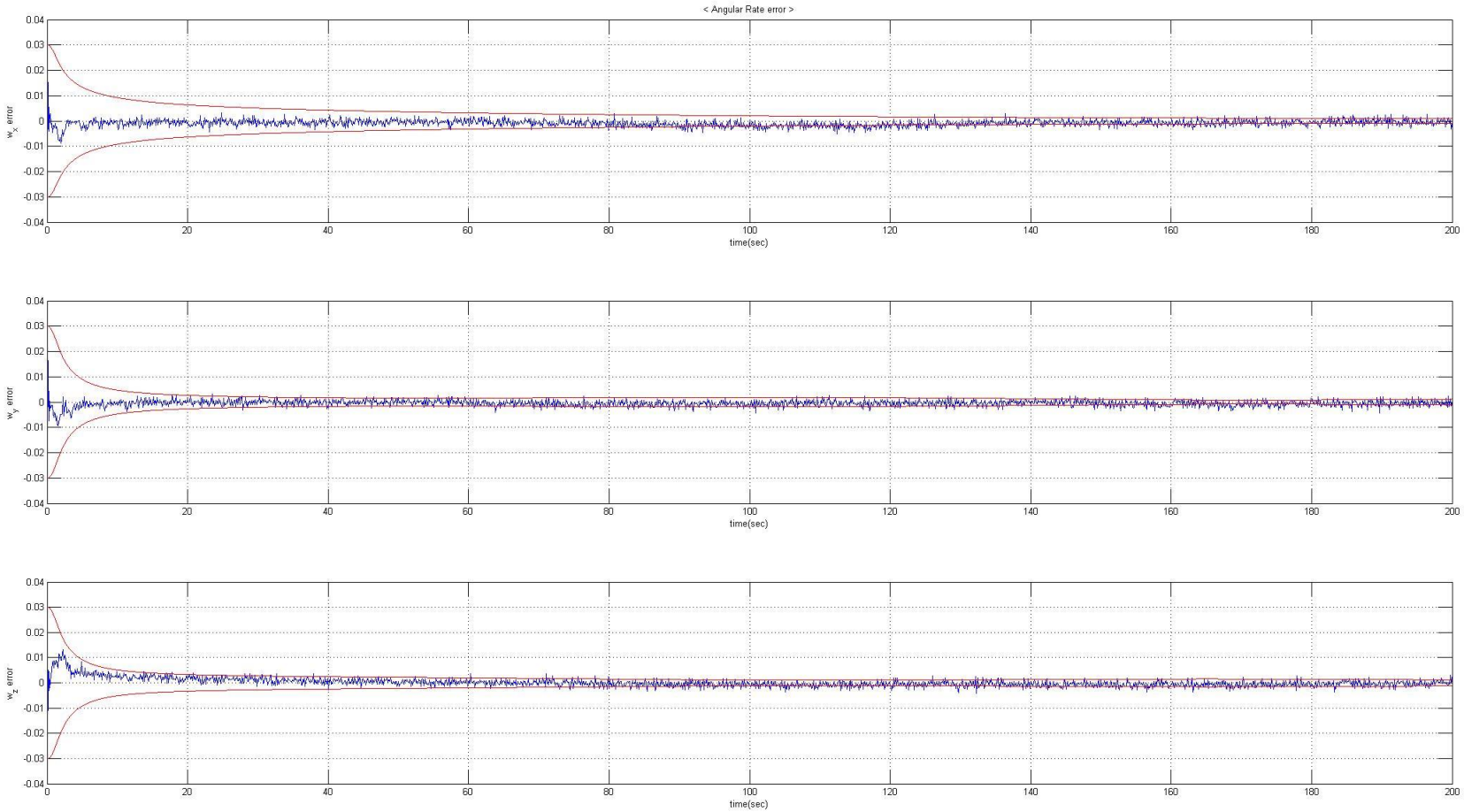
$$\hat{\boldsymbol{q}}_{k+1}^{+} = \delta\hat{\boldsymbol{q}}_{k+1} \otimes \hat{\boldsymbol{q}}_{k+1}^{-}$$

$$\hat{\boldsymbol{b}}_{k+1}^{+} = \hat{\boldsymbol{b}}_{k+1}^{-} + \delta\hat{\boldsymbol{b}}_{k+1}$$

$$\boldsymbol{P}_k^{+} = \left[\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k\right]\boldsymbol{P}_k^{-}$$

3 시뮬레이션 결과(1/3)

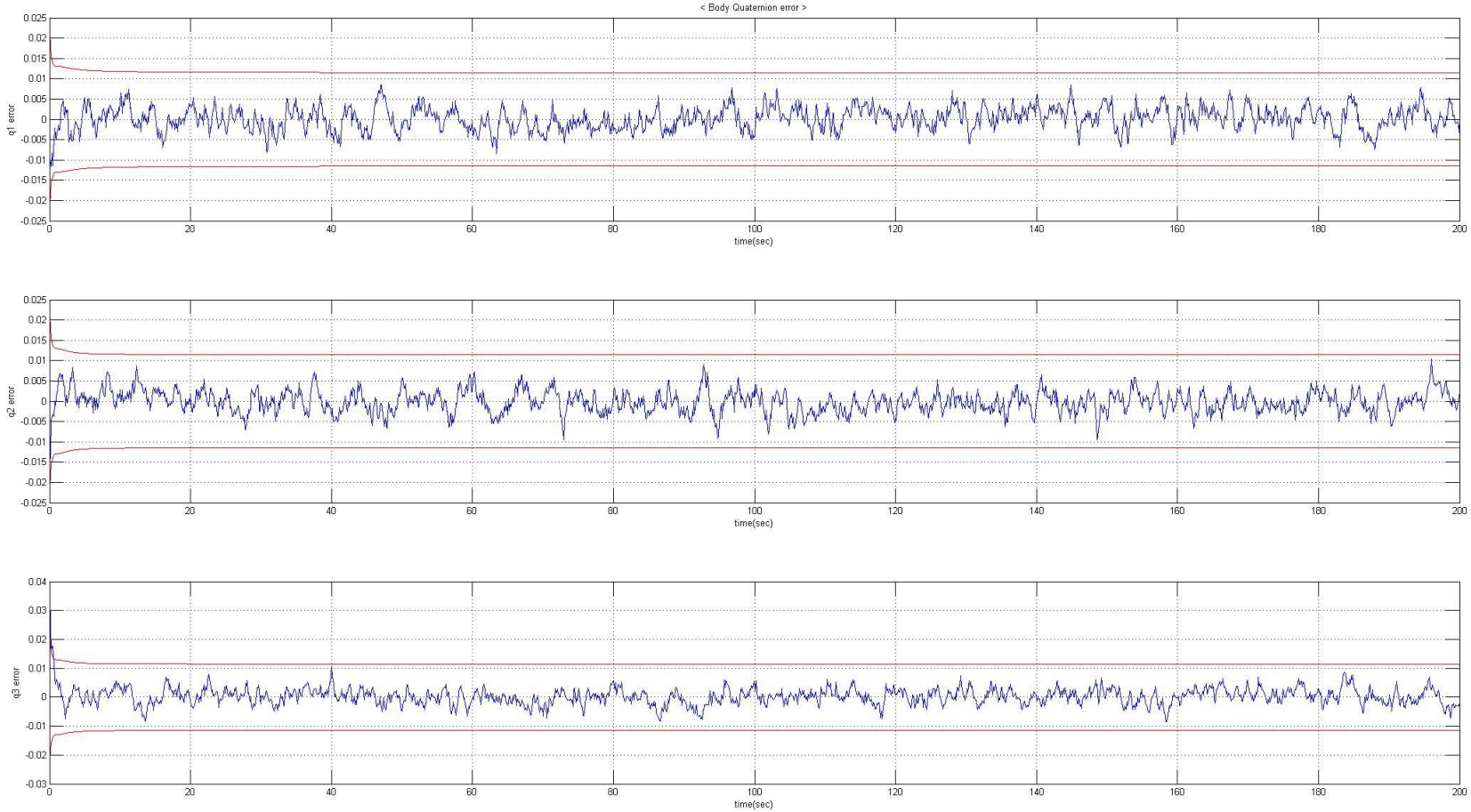
Angular rate error at each axies



- Angular rate error가 3σ 범위 밖으로 약간씩 튀는 부분들이 존재함.
- 전체적으로 볼 경우 수렴범위 내에 있다고 판단됨.

3 시뮬레이션 결과(2/3)

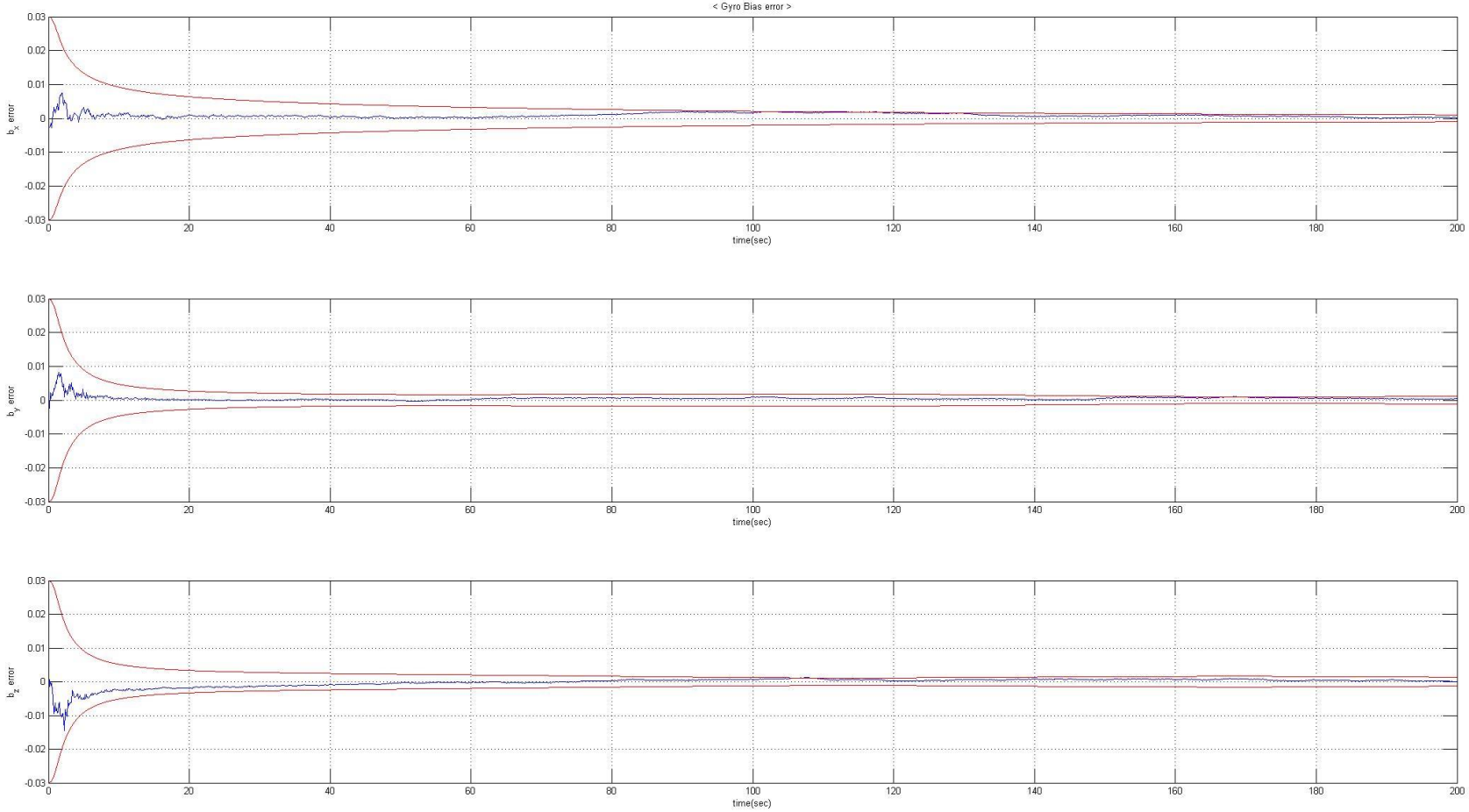
Body Quaternion error at each axies



- Body Quaternion error가 3σ 범위 내로 잘 수렴됨을 볼 수 있음.
- 위성의 자세에 대한 추정이 잘 되고 있다고 판단됨.

3 시뮬레이션 결과(1/3)

Angular rate error at each axies



- Gyro Bias error가 3σ 범위에 걸쳐있는 부분들이 존재함.
- Bias가 매우 잘 추정된다고 볼 수 없지만, 전체적으로 볼 경우 수렴범위 내에 있다고 판단됨.

4 결론 및 계획(1/1)

CONTROLA

- 주어진 연구 논문에 적용된 discrete Φ,Q,R 을 사용하여 시뮬레이션을 했을 경우 때 P값이 작게 수렴이 되고 추정값도 발산하는 문제 발생.
  수식 guarantee를 할 수 없기 때문에 P값은 continuous-time으로 propagation 하였음.
- 시뮬레이션 결과 각속도의 추정과 바이어스의 추정은 다소 수렴이 불안정한 모습이 보였지만 위성의 자세 추정은 잘 되었음.
- 다음 계획으로는 수식을 guarantee 하고 완전한 discrete한 방법으로도 시뮬레이션을 진행.
  또한 교수님께서 말씀하신 gyro가 없을 경우 reaction wheel의 각속도로 자세를 추정하는 시뮬레이션을 진행할 예정.

Thanks for your attention!

Appendix 시뮬레이션 코드(2/2)

```
% --- Spacecraft Attitude Estimation with Low-Cost Sensors Paper Simulation


% - simulation numerical conditions -
% time step
delt = 0.1;   % sec
tmax = 200;   % sec

% moment of inertia of inertia "J"
J = [  16.3     0.0869    0.60167 ;
       0.0867    36.6     0.13571 ;
       0.60167  0.13571    38.6    ];


% - standard deviations and covariances -
% % disturbance noise
% sigma_w = 0.00005*eye(3);   % Nm
% Q_w = sigma_w.^2;
%
% % gyro measurement noise
% sigma_a = 0.025*eye(3)*(pi/180);   % deg/root(s) -> rad/root(s)
% Q_a = sigma_a.^2;
%
% % gyro bias noise
% sigma_b = 0.0011*eye(3)*(pi/180);   % deg/s^(3/2) -> rad/s^(3/2)
% Q_b = sigma_b.^2;
%
% % star tracker
% sigma_s = [ 0.045   0     0 ;
%               0   0.045   0 ;
%               0     0   0.86 ]*(pi/180);   % deg -> rad
% Q_s = sigma_b.^2;

% alternative sigma
sigma_w = 1.0e-5;
sigma_a = 0.0011;
sigma_b = 0.0001;
sigma_s = 0.01;

Q_w = sigma_w^2*eye(3);
Q_a = sigma_a^2*eye(3);
Q_b = sigma_b^2*eye(3);
Q_s = sigma_s^2*eye(3);
```

```
% - system matrices -
% process noise covariance
% Q = [       (inv(J))^2*sigma_w^2*delt  *eye(3)  (1/4)*(inv(J))^2*sigma_w^2*delt^2*eye(
%       (1/4)*(inv(J))^2*sigma_w^2*delt^2*eye(3)  (1/8)*(inv(J))^2*sigma_w^2*delt^3*eye(
%                                       zeros(3)                                    zeros(

% alternative continuous Q
Q = [ Q_w zeros(3) zeros(3); zeros(3) Q_s zeros(3); zeros(3) zeros(3) Q_b ];

% measurement noise covariance
% R = [ ((1/3)*sigma_b^2*delt^3 + sigma_a^2)*eye(3)          zeros(3) ;
%                                          zeros(3)  sigma_s^2*eye(3)  ];

% alternative continuous R
R = [ Q_a*eye(3) zeros(3); zeros(3) Q_s*eye(3) ];

% sensitivity
H = [   eye(3)  zeros(3)    eye(3) ;
      zeros(3)    eye(3)  zeros(3)  ];


% - simulation settings -
% initial state
w_t = [ 0.05 -0.01  0.02 ]';
q_t = [ 0 0 0 1 ]';
%b_t = [ 1.66 -1.66 1.099 ]'*(pi/180);
b_t = 0.01*(rand(3,1)-0.5);

x = [ w_t'  q_t' ]';

w_h = [ 0.02  -0.03  0.05 ]';
q_h = [0 0 0 1]'+0.051*randn(4,1);
q_h = q_h/norm(q_h);
b_h = zeros(3,1);

P = [ 10^5*eye(3)     zeros(3)     zeros(3) ;
         zeros(3)  10^3*eye(3)     zeros(3) ;
         zeros(3)     zeros(3)  10^3*eye(3)  ]*0.001^2;

% input value
u = zeros(3,1);

% reserved parameter; it is not used in this simulation study
res = 0;
```

Appendix 시뮬레이션 코드(2/2)

```
% -- simulation start here --
for index = 1 : round(tmax/delt)

    % spacecraft dynamics to generate true angular rate(wt), attitude(qt) and etc.
    x = srk4(@SpaceCraftDyn, x, u, Q_w, res, delt);

    w_t = x(1:3);
    q_t = x(4:7);

    % gyro bias dynamics
    b_t = srk4(@GyroDyn, b_t, u, Q_b, res, delt);
    n_a = sigma_a*randn(3,1);

    % - measurement output -
    % gyro output
    wg = w_t + b_t + n_a;
    wg_h = w_h + b_h;

    % star tracker measurement - body attitude
    p_e = sigma_s*randn(3,1);
    q_e = [ p_e'  sqrt(1-(p_e'*p_e)) ]';
    q_star = qcross(q_e) * q_t;

%-----------------------------------------------------------------------%
%          Start Extended Kalman Filtering for attitude estimation      %
%-----------------------------------------------------------------------%

    % -- 1st step(Propagation) --
    % - state propagation -
    % angular rate
    w_h = wg - b_h;

    % quaternion of body attitude
    psi_hat = sin(0.5*norm(w_h)*delt)*w_h / norm(w_h);
    q_h = Capita_Omega(w_h,delt,psi_hat)*q_h;

    % gyro bias
    % b_h = b_h;  because b_h_dot = 0, best estimation of white-noise process is 0;


    % Transition matrix
    Jyy = (J(2,2)+J(3,3))/2;
    Kapa = (Jyy-J(1,1))/Jyy;

%     theta_h = w_h*delt;
%     theta_h_n = norm(theta_h);
%     theta_h_x = theta_h(1);
%     theta_h_s = [ theta_h(2) theta_h(3) ]';
%     phi = cos(theta_h_n)+eye(3) + ((1-cos(theta_h_n))/theta_h_n^2)*theta_h*theta_
%
%     Lamda = [  cos(Kapa+theta_h_x)  sin(Kapa+theta_h_x) ;
%               -sin(Kapa+theta_h_x)  cos(Kapa+theta_h_x)  ];
%
%     psi = [                  1                        zeros(1,2) ;
%            (Lamda+theta_h_s + theta_h_s)/theta_h_x     Lamda    ];
%
%     PHi = [       psi          zeros(3)  zeros(3) ;
%            (1/2)+eye(3)+delt     phi    zeros(3) ;
%                zeros(3)       zeros(3)    eye(3)  ];

    % alternative PHi
    J_w = Kapa*[ 0 0 0; w_h(3) 0 w_h(1); -w_h(2) -w_h(1) 0 ];
    A = [ J_w zeros(3) zeros(3); 0.5+eye(3) -skwsym(w_h) zeros(3); zeros(3,9)];
%     PHi = eye(9) + A+delt;

    % error covariance P
%     upsilon = [  inv(J)       zeros(3)  zeros(3) ;
%                zeros(3)  (1/2)+eye(3)  zeros(3) ;
%                zeros(3)      zeros(3)    eye(3)  ];
%
%     P = PHi+P+PHi' + upsilon+Q+upsilon';

%     alternative continuous-time error covariance P
%     G = [ inv(J) zeros(3,6); zeros(3) 0.5+eye(3) zeros(3); zeros(3,6) eye(3) ];
%     P = rk4cov(@pdyn,P,A,G,Q,delt);


    % -- 2nd step(Kalman Gain) --
    K = P*H'*(H+P+H' + R)^(-1);

    % -- 3rd step(Measurement Update) --
    % obtain innovation
    del_wg = wg - wg_h;

    del_q = qcross(q_star) * qinv(q_h);
    del_p = del_q(1:3);

    del_y = [ del_wg'  del_p' ]';
    del_x = K * del_y;

    del_p_h = del_x(4:6);
    del_q_h = [ del_p_h'  sqrt(1-(del_p_h'*del_p_h)) ]';

    % state update
    w_h = w_h + del_x(1:3);
    q_h = qcross(del_q_h) * q_h;
    q_h = q_h/norm(q_h);
    b_h = b_h + del_x(7:9);

    % covariance update
    P = (eye(length(P))-K*H)*P;

%-----------------------------------------------------------------------%
%                      Extended Kalman Filtering End                    %
%-----------------------------------------------------------------------%
```