# 자이로리스 위성의 자세 및 각속도 결정 기법

## A Study on the "Gyroless Satellite" Attitude and Angular Rate Determination with Star Tracker and Reaction Wheel

**Dept. of Aerospace Engineering**

**Lab for Navigation, Control and Applications**

**Stephen. Park.**

Date: 2015. 10. 05. Tue.

# Contents


01 연구 동기 및 목적

02 최적 추정을 위한 확장칼만필터 설계

03 알고리즘 검증 시뮬레이션

04 결론 및 계획

# 1 연구의 목적(1/1)

- 위성의 기능 결함 중 상당부분이 Gyro의 고장이 원인이 되어 GAD(Gyroless Attitude Determintation) 기법이 개발됨.
- 기술의 발전으로 고정밀 센서들 사용과 위성 개발비 절감, 초소형 위성의 출현으로 Gyro를 탑재하지 않는 GAD 기법들이 연구됨.
- GAD 기법의 큰 두가지 분류
  - Star Tracker의 빠른 갱신주기 이용. 미분형태로 각속도 추정
  - 다른 센서 정보를 활용하여 Gyro를 대체

# 2 최적추정을 위한 확장칼만필터 설계(1/6)

CONTROLA

2.1 RW을 탑재한 위성의 회전 동역학

- 회전구동기를 탑재한 위성의 회전동역학

$$\dot{\boldsymbol{\omega}} = \boldsymbol{J}^{-1}\left\{-\boldsymbol{\omega}\times(\boldsymbol{J}\boldsymbol{\omega}+C_w^b\boldsymbol{J}_w\boldsymbol{\omega}_w)-C_w^b(\boldsymbol{\tau}_w+\boldsymbol{\eta}_c)+\boldsymbol{d}+\boldsymbol{\tau}_{dist}\right\}$$

- 회전동역학의 추정식

$$\dot{\hat{\boldsymbol{\omega}}} = \boldsymbol{J}^{-1}\left\{-\hat{\boldsymbol{\omega}}\times(\boldsymbol{J}\hat{\boldsymbol{\omega}}+C_w^b\boldsymbol{J}_w\hat{\boldsymbol{\omega}}_w)-C_w^b(\boldsymbol{\tau}_w)+\hat{\boldsymbol{d}}\right\}$$

**Table 1. Meaning of motion of dynamics of satellite**

| Symbol | Physical meaning |
|---|---|
| $\boldsymbol{\omega}$ | 위성의 3축 각속도 |
| $\boldsymbol{J}$ | 위성의 관성모멘트 행렬 |
| $C_w^b$ | RW에서 위성동체로의 변환행렬 |
| $\boldsymbol{J}_w$ | RW의 관성모멘트 행렬 |
| $\boldsymbol{\omega}_w$ | n개 RW의 각속도 |
| $\boldsymbol{\tau}_w$ | n개 RW의 토크명령 |
| $\boldsymbol{\eta}_c$ | n개 RW의 토크명령 오차 |
| $\boldsymbol{d}$ | 위성의 3축에 가해지는 랜덤한 External Disturbance |
| $\boldsymbol{\tau}_{dist}$ | 위성의 3축에 가해지는 랜덤한 External Torque |

# 2 최적추정을 위한 확장칼만필터 설계(2/6)

CONTROLA

2.2 Linearization of the State Error Equation

- 자세오차 방정식

– 오차 쿼터니언 ( $\delta \boldsymbol{q} = \boldsymbol{q} \otimes \hat{\boldsymbol{q}}^{-1}$ ) 으로부터 유도

$$\delta \dot{\boldsymbol{q}} \approx -\hat{\boldsymbol{\omega}} \times \delta \boldsymbol{q} + \frac{1}{2} \delta \boldsymbol{\omega}$$

충분히 작은 각도에서 쿼터니언은
다음과 같이 근사 될 수 있음.

$$\boldsymbol{q} \approx \boldsymbol{\alpha} / 2 \;\; and \;\; \boldsymbol{q}_4 \approx 1$$

$$\boxed{\delta \dot{\boldsymbol{\alpha}} \approx -\hat{\boldsymbol{\omega}} \times \delta \boldsymbol{\alpha} + \delta \boldsymbol{\omega}}$$

- 위성의 각속도오차 방정식

– 위성의 각속도 오차( $\delta \boldsymbol{\omega} = \boldsymbol{\omega} - \hat{\boldsymbol{\omega}}$ )로부터 유도

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

$$\dot{\boldsymbol{\omega}} = \boldsymbol{J}^{-1} \left\{ -\boldsymbol{\omega} \times (\boldsymbol{J}\boldsymbol{\omega} + C_w^b \boldsymbol{J}_w \boldsymbol{\omega}_w) - C_w^b (\boldsymbol{\tau}_w + \boldsymbol{\eta}_c) + \boldsymbol{d} + \boldsymbol{\tau}_{dist} \right\}$$

미분식의 Taylor전개 에서
1차 까지만 끊어서 근사시킴.

$$\boxed{\begin{aligned} \delta \dot{\boldsymbol{\omega}} = \boldsymbol{J}^{-1} \left\{ -\hat{\boldsymbol{\omega}} \times (\boldsymbol{J}\delta\boldsymbol{\omega} + C_w^b \boldsymbol{J}_w \delta\boldsymbol{\omega}_w) - \delta\boldsymbol{\omega} \times (\boldsymbol{J}\hat{\boldsymbol{\omega}} + C_w^b \boldsymbol{J}_w \hat{\boldsymbol{\omega}}_w) + \delta \boldsymbol{d} \right\} \\ + \boldsymbol{J}^{-1} \left\{ -C_w^b \boldsymbol{\eta}_c + \boldsymbol{\tau}_{dist} \right\} \end{aligned}}$$

# 2 최적추정을 위한 확장칼만필터 설계(3/6)

CONTROLA

2.2 Linearization of the State Error Equation

- 반작용휠의 각속도오차 방정식

– 휠의 각운동량 오차($\delta h_w = h_w - \hat{h}_w$) 으로부터 유도

$$\delta \dot{h}_w = \dot{h}_w - \dot{\hat{h}}_w = \eta_c \qquad [(\tau_w + \eta_c) - \tau_w]$$

본 연구에서는 반작용휠의 각속도를 측정 및 제어가능 하다고 가정함.

$$\delta \dot{\omega}_w = \dot{\omega}_w - \dot{\hat{\omega}}_w = \eta_c / J_w$$

- 외부교란오차 방정식

– 외부교란 오차( $\delta d = d - \hat{d}$ )로부터 유도

본 연구에서는 위성의 3축에 가해지는 랜덤한 External Disturbance도 같이 추정함.

$$\delta \dot{d} = \dot{d} - \dot{\hat{d}} = \eta_d \qquad (\eta_d - 0)$$

# 2 최적추정을 위한 확장칼만필터 설계(4/6)

CONTROLA

2.3 System Process Matrices

- 상태공간방정식 설정

– 선형화된 오차방정식들로 부터 유도

$$\boxed{\delta\dot{\boldsymbol{x}} = \boldsymbol{A}\delta\boldsymbol{x} + \boldsymbol{B}\delta\boldsymbol{u} + \boldsymbol{G}\boldsymbol{\eta}}$$

$$\delta\boldsymbol{x} = \left[\delta\boldsymbol{\alpha}^T \;\; \delta\boldsymbol{\omega}^T \;\; \delta\boldsymbol{\omega}_w^T \;\; \delta\boldsymbol{d}^T\right]^T$$

– 시스템 상태변수 오차

$$\boldsymbol{\eta} = \left[\boldsymbol{\eta}_c^T \;\; \boldsymbol{\tau}_{dist}^T \;\; \boldsymbol{\eta}_d^T\right]^T$$

– 시스템의 zero-mean Gaussian white-noise process

- 칼만필터링을 위한 System Matrices

$$\boldsymbol{A} = \begin{bmatrix} [-\hat{\boldsymbol{\omega}}]^\times & \boldsymbol{I}_{3\times3} & 0_{3\times n} & 0_{3\times3} \\ 0_{3\times3} & \boldsymbol{J}^{-1}\left\{[-\hat{\boldsymbol{\omega}}]^\times \boldsymbol{J} + [\boldsymbol{J}\hat{\boldsymbol{\omega}}]^\times + \left[C_w^b \boldsymbol{J}_w \hat{\boldsymbol{\omega}}_w\right]^\times\right\} & \boldsymbol{J}^{-1}\left\{[-\hat{\boldsymbol{\omega}}]^\times C_w^b \boldsymbol{J}_w\right\} & \boldsymbol{J}^{-1} \\ 0_{n\times3} & 0_{n\times3} & 0_{n\times n} & 0_{n\times3} \\ 0_{3\times3} & 0_{3\times3} & 0_{3\times n} & 0_{3\times3} \end{bmatrix}$$

- Skew-Symmetric Matrix

$$[\boldsymbol{X}]^\times = \begin{bmatrix} 0 & -X_z & X_y \\ X_z & 0 & -X_x \\ -X_y & X_x & 0 \end{bmatrix}$$

$$\boldsymbol{G} = \begin{bmatrix} 0_{3\times n} & 0_{3\times3} & 0_{3\times3} \\ -\boldsymbol{J}^{-1}C_w^b & \boldsymbol{J}^{-1} & 0_{3\times3} \\ (1/\boldsymbol{J}_w) & 0_{n\times3} & 0_{n\times3} \\ 0_{3\times n} & 0_{3\times3} & \boldsymbol{I}_{3\times3} \end{bmatrix} \qquad \boldsymbol{Q} = \begin{bmatrix} E\left\langle \boldsymbol{\eta}_c^T \boldsymbol{\eta}_c \right\rangle & 0_{n\times3} & 0_{n\times3} \\ 0_{3\times n} & E\left\langle \boldsymbol{\tau}_{dist}^T \boldsymbol{\tau}_{dist} \right\rangle & 0_{3\times3} \\ 0_{3\times n} & 0_{3\times3} & E\left\langle \boldsymbol{\eta}_d^T \boldsymbol{\eta}_d \right\rangle \end{bmatrix}$$

# 2 최적추정을 위한 확장칼만필터 설계(5/6)

CONTROLA

2.3 System Process Matrices

- 측정센서 모델링

– 두 개의 측정센서를 이용하여 칼만필터링 수행
  Star Traker(위성의 자세 측정)
  반작용휠 내부 휠 각속도 측정 센서

– Star Tracker와 휠 각속도 센서 모델링

$$\tilde{\boldsymbol{q}} = \delta \boldsymbol{q}_{st} \otimes \boldsymbol{q}, \qquad \tilde{\boldsymbol{\omega}}_w = \boldsymbol{\omega}_w + \boldsymbol{\eta}_w$$

$$\delta \boldsymbol{y} = \boldsymbol{H} \delta \boldsymbol{x} + \boldsymbol{v}$$ – 측정 모델식

$$\delta \boldsymbol{y} = \left[ \delta \boldsymbol{\alpha}^T \;\; \delta \boldsymbol{\omega}_w^T \right]^T$$ – 측정값과 추정값의 오차

$$\boldsymbol{v} = \left[ \boldsymbol{\eta}^T{}_{st} \;\; \boldsymbol{\eta}^T{}_w \right]^T$$ – 센서의 zero-mean
  Gaussian white-noise process

- 나머지 System Matrices

$$\boldsymbol{H} = \begin{bmatrix} \frac{1}{2}\boldsymbol{I}_{3\times3} & 0_{3\times3} & 0_{3\times n} & 0_{3\times3} \\ 0_{n\times3} & 0_{n\times3} & \boldsymbol{I}_{n\times n} & 0_{n\times3} \end{bmatrix}$$

$$\boldsymbol{R} = \begin{bmatrix} E\left\langle \boldsymbol{\eta}_{st}^T \boldsymbol{\eta}_{st} \right\rangle & 0_{3\times n} \\ 0_{n\times3} & E\left\langle \boldsymbol{\eta}_w^T \boldsymbol{\eta}_w \right\rangle \end{bmatrix}$$

# 2 최적추정을 위한 확장칼만필터 설계(6/6)

CONTROLA

2.4 Discretization Matrices

- Discrete type Algorithm

– System Process Matrices는 상태변수 오차의 미분식으로부터 얻어짐.
따라서 Continuous type 알고리즘에서만 사용될 수 있음.

– Discrete type 알고리즘에서는 이에 맞게 변형해서 사용해야함.

- 시스템 Transition Matrix

– Transition Matrix는 간단하면서 실제 값에 근사시키는 방법 사용.

$$\boxed{\Phi_k \approx \boldsymbol{I} + \boldsymbol{A}(t)\Delta t}$$

- 시스템 Noise Covariance Matrix

– 시스템의 Noise Covariance Matrix는 다음과 같이 근사 가능.

$$\boxed{\boldsymbol{Q}_k \approx \Delta t \boldsymbol{G}(t)\boldsymbol{Q}(t)\boldsymbol{G}(t)^T}$$

# 3 알고리즘 검증 시뮬레이션(1/6)

CONTROLA

- 알고리즘의 구성과 목표

– 제안된 알고리즘은 확장칼만필터를 기반으로 하여 위성에 탑재된 두 센서의 측정값을 가지고 필터링을 통해 위성의 자세와 각속도, External Disturbance를 동시에 최정 추정 한다.

– 시뮬레이션은 반작용휠이 위성의 3축에 정확히 배치되었다고 가정한 케이스에 대하여 수행함.

- 시뮬레이션 조건1

– 위성의 관성모멘트( $J$ )는 실제 궤도 환경과 유사하도록 나타내기 위해 10%의 오차를 추가한 값을 사용.

$$J = \begin{bmatrix} 16.3 & 0.0869 & 0.60167 \\ 0.0867 & 36.6 & 0.13571 \\ 0.60167 & 0.13571 & 38.6 \end{bmatrix}$$

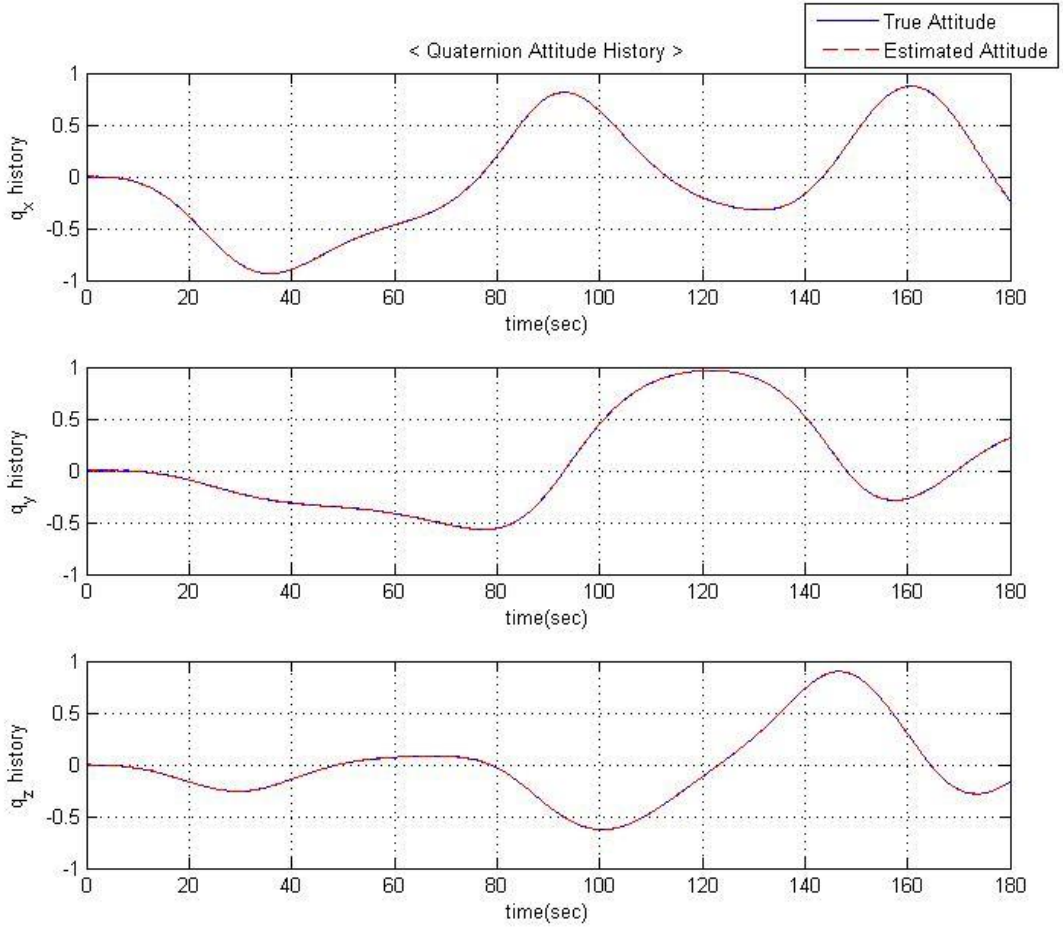
- 시뮬레이션 조건2

**Table 2. Numerical Simulation Conditions**

| list | | unit |
|---|---|---|
| RW 탑재 수량 | 3 | 개 |
| RW 각속도 측정 주기 | 20 | Hz |
| RW 각속도 측정 정밀도( $1\sigma$ ) | 2.86 | rad/s |
| RW 토크 명령 오차( $1\sigma$ ) | 0.0013 | Nm |
| RW의 관성모멘트( $J_w$ ) | 0.0014 | $Kg \cdot m^2$ |
| RW의 초기 각속도( $\omega_{w_0}$ ) | 360 | rad/s |
| Star Tracker 정밀도( $1\sigma$ ) | 0.01 | rad |
| External Disturbance( $d$ , $1\sigma$ ) | 0.0011 | Nm |
| External Torque( $\tau_{dist}$ , $1\sigma$ ) | 10E-5 | Nm |

# 3 알고리즘 검증 시뮬레이션(2/6)

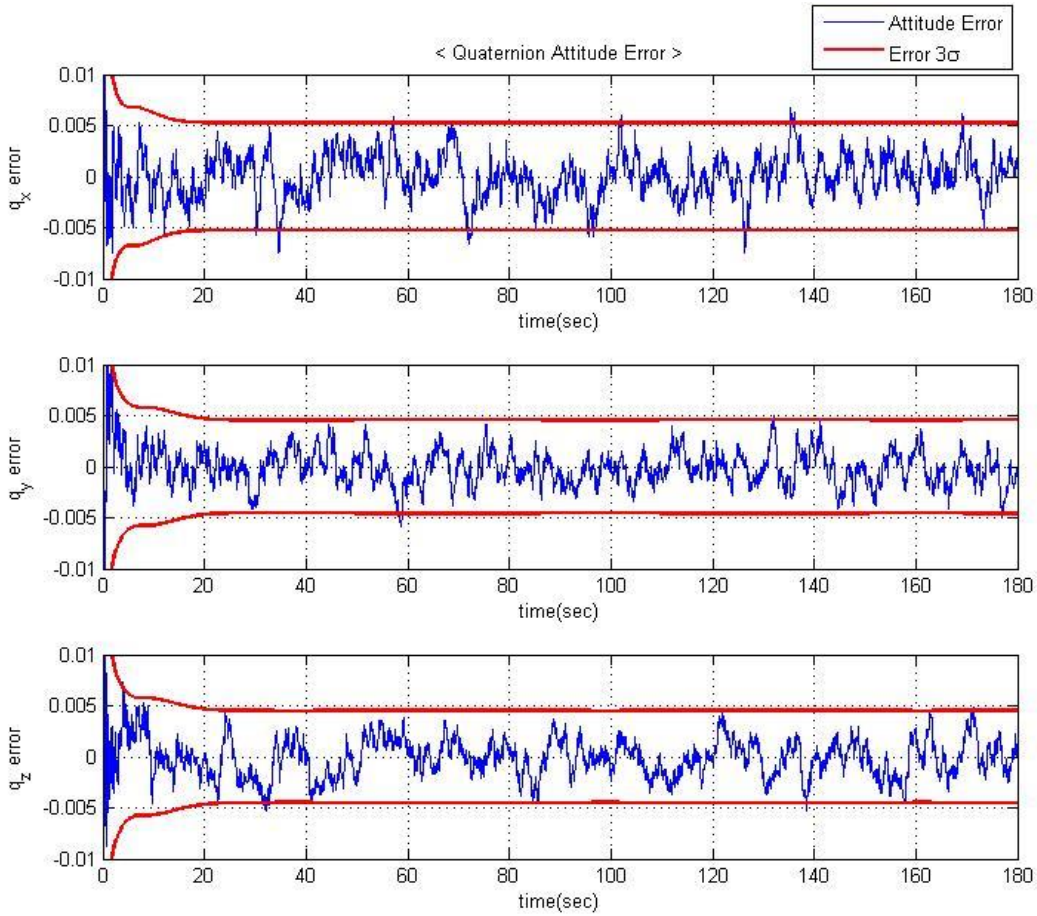
Attitude History(quaternion)




- 위성의 3축에 대한 180초 동안의 자세(quaternion)의 history를 나타냄.
- 파란색 선은 실제 자세값을 나타내며, 빨간색 선은 추정값을 나타낸다.

# 3 알고리즘 검증 시뮬레이션(3/6)

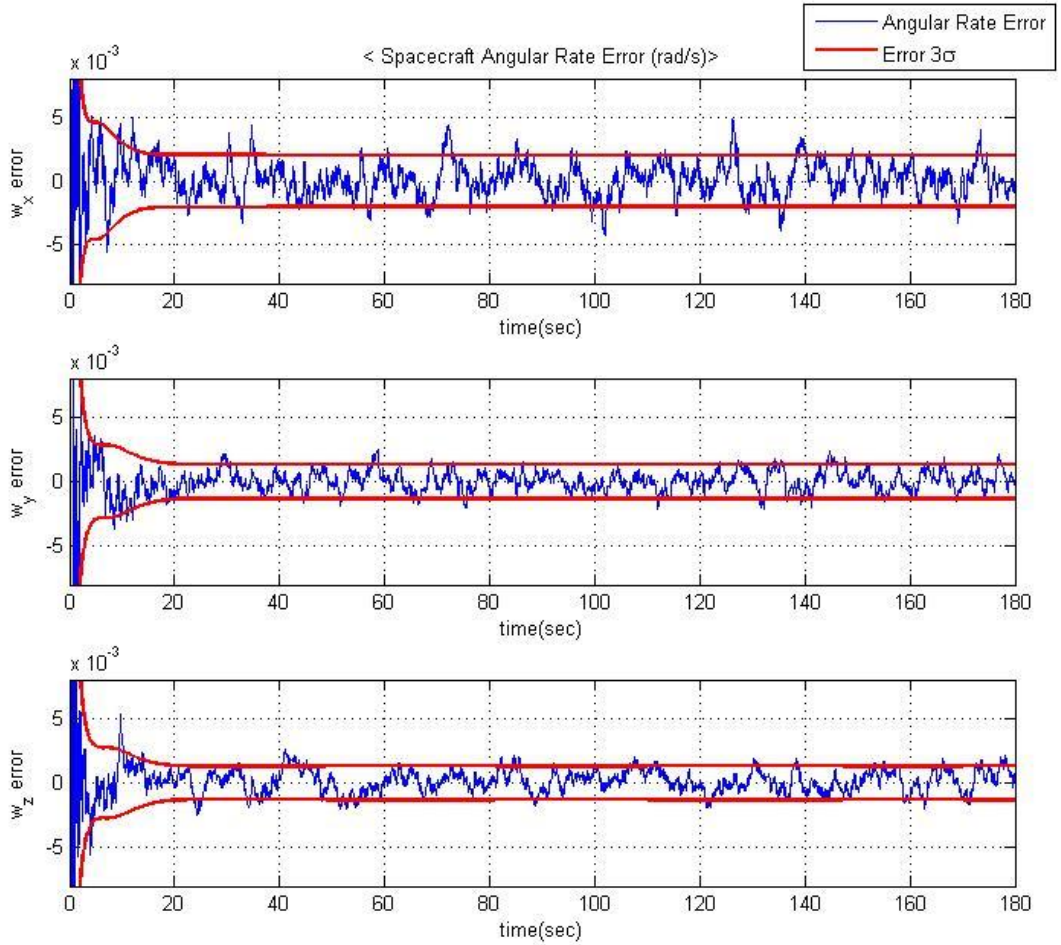
Attitude Error(quaternion)




- 위성의 3축에 같은 Star Tracker의 측정정밀도 0.01rad 를 갖는다고 가정하였음.
- 자세 결정정밀도는 x축(0.0052rad), y축(0.0045rad), z축(0.0044rad) 으로 달라짐.

# 3 알고리즘 검증 시뮬레이션(4/6)

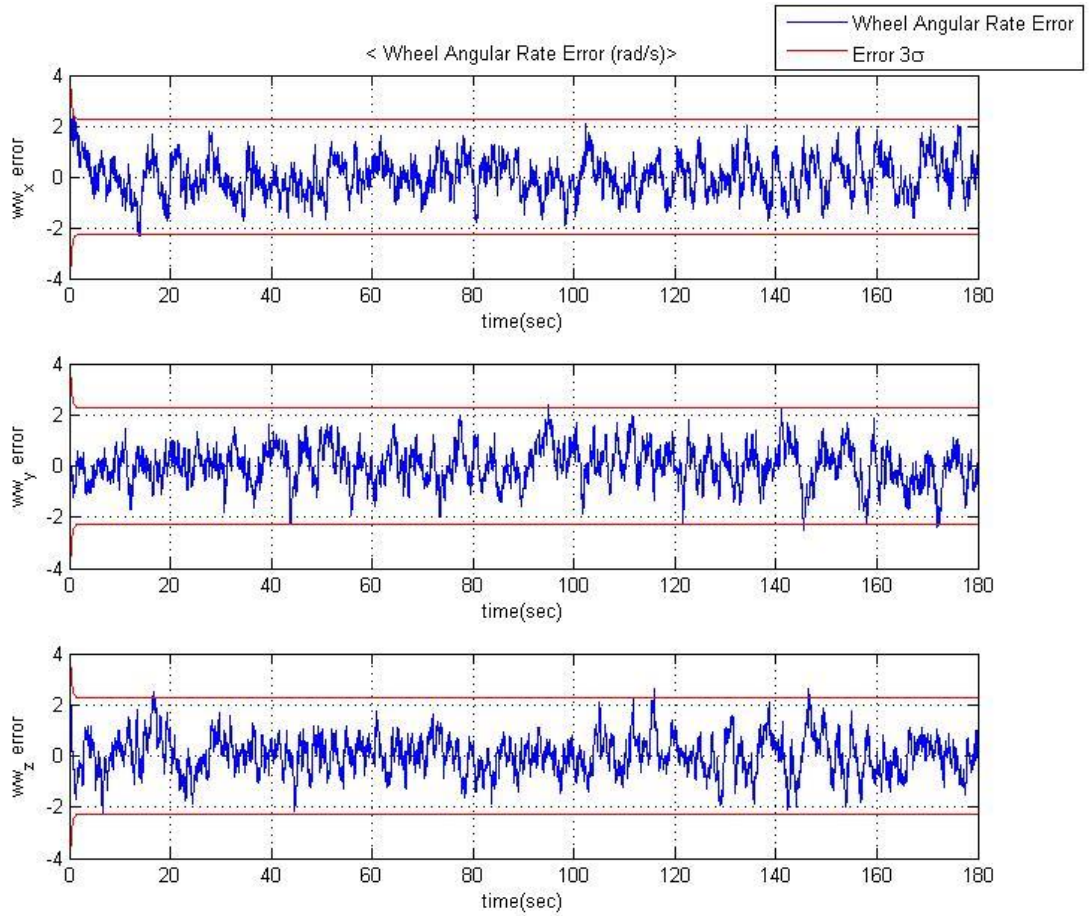
Angular Rate Error




- 각속도 결정정밀도는 x축에 비해 y,z축 정밀도가 눈에 띄게 차이를 보임.
- 각 축의 결정정밀도는 x축(0.002rad/s), y축과 z축은(0.0013rad/s) 값을 갖음.

# 3 알고리즘 검증 시뮬레이션(5/6)

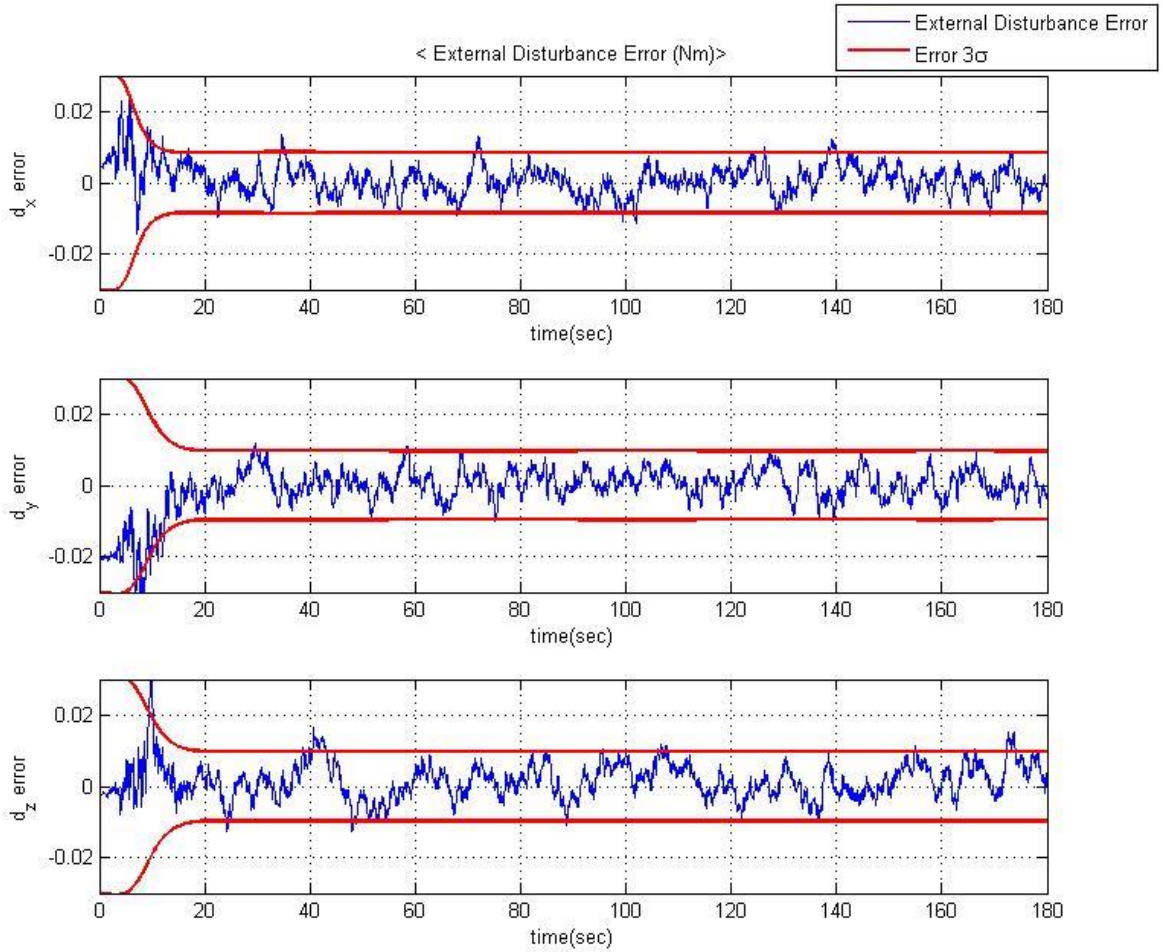
Wheel Angular Rate Error




- 반작용휠의 3축에 대한 측정정밀도는 2.86rad/s 를 갖는다고 가정하였음.
- 각 축의 결정정밀도는 x축(2.269rad/s), y축과 z축은(2.2691rad/s) 값을 갖음.

# 3 알고리즘 검증 시뮬레이션(6/6)

External Disturbance Error




- 외부교란 결정정밀도는 x,y,z축 정밀도가 큰 차이를 보이지 않음.
- 각 축의 결정정밀도는 x축(0.0084Nm), y축과 z축은(0.01Nm) 의 값을 갖는다.

# 4 결론 및 계획(1/1)

CONTROLA

- 본 연구에서는 위성의 각속도를 측정하는 센서 고장 상황 이나 각속도 센서를 아예 탑재하지 않는 경우에 대체할 수 있는 Gyroless Attitude Determination 기법을 이용하여 Gyroless 위성의 자세와 각속도를 추정할 수 있는 알고리즘을 제안함.
- 제안된 알고리즘은 확장칼만필터 기반으로 설계되었음. 각속도 센서 부재와 더불어 랜덤한 3축 External Disturbance와 External Torque의 외란환경에 있는 위성의 자세와 각속도를 Star Tracker와 반작용휠 내부 각속도 센서의 값을 가지고 최적으로 추정할 수 있음.
- 시뮬레이션을 통해 제안된 알고리즘의 성능과 타당성을 입증함.
- 반작용휠 뿐만 아니라 CMG와 같은 다른 회전 구동기를 탑재한 위성에도 제안된 알고리즘의 적용이 가능함.

- 필터의 성능에 영향을 줄 수 있는 Transition Matrix의 근사치가 아닌 정확한 솔루션값을 구해 사용하고, 반작용휠의 배치 케이스들에 대해서도 알고리즘의 성능 연구를 수행할 예정.
- 심화적으로 인공위성의 정확한 관성모멘트를 알 수 없는 경우를 포함한 전체적인 알고리즘 구성과 성능 및 타당성에 대한 연구 또한 수행할 예정이다.

MIKE SALWAY PHOTOGRAPHY
WWW.MIKESALWAY.COM.AU

# Thanks for your attention!

# Appendix 확장칼만필터 구성(1/1)

**Table 3. Extended Kalman Filter for proposed Algorithm**

| | |
|---|---|
| **Initialize** | $\hat{\boldsymbol{x}}(t_0) = \hat{\boldsymbol{x}}_0 \qquad \boldsymbol{P}(t_0) = \boldsymbol{P}_0$ |
| **Propagation** | $\dot{\hat{\boldsymbol{q}}} = \frac{1}{2}\Omega(\hat{\boldsymbol{\omega}}) \cdot \hat{\boldsymbol{q}}$<br>$\dot{\hat{\boldsymbol{\omega}}} = \boldsymbol{J}^{-1}\left\{-\hat{\boldsymbol{\omega}} \times (\boldsymbol{J}\hat{\boldsymbol{\omega}} + C_w^b \boldsymbol{J}_w \hat{\boldsymbol{\omega}}_w) - C_w^b(\boldsymbol{\tau}_w) + \hat{\boldsymbol{d}}\right\} \qquad \boldsymbol{P}_{k+1}^- = \Phi_k \boldsymbol{P}_k^+ \Phi_k^T + \boldsymbol{Q}_k$<br>$\dot{\hat{\boldsymbol{\omega}}}_w = \boldsymbol{\tau}_w / \boldsymbol{J}_w$<br>$\dot{\hat{\boldsymbol{d}}} = 0$ |
| **Gain** | $\boldsymbol{K}_{k+1} = \boldsymbol{P}_{k+1}^- \boldsymbol{H}^T \left(\boldsymbol{H}\boldsymbol{P}_{k+1}^- \boldsymbol{H}^T + \boldsymbol{R}\right)^{-1}$ |
| **Update** | $\delta \boldsymbol{y} = \left[\delta \boldsymbol{\alpha}^T \;\; \delta \boldsymbol{\omega}_w^T\right]^T$<br>$\delta \boldsymbol{x}_{k+1}^+ = K_{k+1} \delta \boldsymbol{y}$<br>$\hat{\boldsymbol{q}}_{k+1}^+ = \begin{bmatrix} \frac{1}{2}\delta\hat{\boldsymbol{\alpha}}_{k+1}^+ \\ 1 \end{bmatrix} \otimes \hat{\boldsymbol{q}}_{k+1}^- \qquad \hat{\boldsymbol{\omega}}_{k+1}^+ = \hat{\boldsymbol{\omega}}_{k+1}^- + \delta\boldsymbol{\omega}_{k+1}^+ \qquad \hat{\boldsymbol{\omega}}_{w_{k+1}}^+ = \hat{\boldsymbol{\omega}}_{w_{k+1}}^- + \delta\boldsymbol{\omega}_{w_{k+1}}^+ \qquad \hat{\boldsymbol{d}}_{k+1}^+ = \hat{\boldsymbol{d}}_{k+1}^- + \delta\boldsymbol{d}_{k+1}^+$<br>$\boldsymbol{P}_{k+1}^+ = \left[\boldsymbol{I} - \boldsymbol{K}_{k+1}\boldsymbol{H}\right]\boldsymbol{P}_{k+1}^-$ |

# Appendix 시뮬레이션 코드(1/2)

```
clear all; close all; clc;

% --- A Study on the Satellite Attitude Determination under Angular Rate Sensor Malfunction ---
% - simulation properties -
wheels = 3;              % 개
wheel_Hz = 20;           % Hz
wheel_h_obser = 0.004;   % Nms

star_Hz = 1;                    % Hz  assume that measurement observated at the same time.
star_obser = 0.01;              % rad
alpha_obser = star_obser / 2;   % rad

torque_oder_error = 0.0013;   % Nm
torque_dist = 10^(-5);        % Nm

external_distur = 0.0011;   % Nm

Jw = 0.0014;   % reaction wheel inertia
w_wheel = 360;      % rad/s
wheel_obser = wheel_h_obser / Jw;

% -- numerical conditions --
% - time step -
delt = 1/wheel_Hz;   % sec
tmax = 180;          % sec

% - covariances -
% reaction wheel torque oder error
Q_oder_error = torque_oder_error^2*eye(3);

% disturbance torque
Q_dist = torque_dist^2*eye(3);

% external disturbances
Q_exd = external_distur^2*eye(3);

% star tracker measure
Q_alpha = alpha_obser^2*eye(3);

% wheel angular momentum measure
Q_wheel_obser = wheel_obser^2*eye(3);
```

```
% wheel angular momentum measure
Q_wheel_obser = wheel_obser^2*eye(3);

% - system matrices -
% process noise covariance
Q = [ Q_oder_error zeros(3,6); zeros(3) Q_dist zeros(3,3);  zeros(3,6) Q_exd ];

% measurement noise covariance
R = [  Q_alpha  zeros(3);  zeros(3)  Q_wheel_obser  ];

% sensitivity
H = [ (1/2)*eye(3) zeros(3,9); zeros(3,6) eye(3) zeros(3) ];

% - simulation setting -
P = [ 10^1*eye(3)    zeros(3)    zeros(3)    zeros(3);
      zeros(3)    10^1*eye(3)    zeros(3)    zeros(3);
      zeros(3)       zeros(3) 10^5*eye(3)    zeros(3);
      zeros(3)       zeros(3)    zeros(3) 10^0*eye(3) ]*10^(-4);

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

% linearized matrices
G = [ zeros(3,9); -inv(J) inv(J) zeros(3); 1/Jw*eye(3) zeros(3,6); zeros(3,6) eye(3) ];
Qk = delt*(G*Q*G');

% intial states
qt = [ 0 0 0 1 ]';
qt = qt/norm(qt);
ws_t = [ 0  -0  0 ]';
ww_t = w_wheel*ones(3,1);
dt = 0.03*randn(3,1);   % External Disturbances  % 수정요함.

x = [ qt' ws_t' ww_t' dt' ]';

qh = [ 0 0 0 1 ]' + 0.051*randn(4,1);
qh = qh/norm(qh);
ws_h = [ 0  0  -0 ]';
ww_h = 355*ones(3,1);
dh = 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)

    % input torque order
    torque_oder = 0.1*sin(0.1*delt*index)*ones(3,1);   % wheel torque oder

    % spacecraft dynamics to generate true angular rate(wt), attitude(qt)
    x = srk4(@TrueDyn,x,torque_oder,Q,res,delt);

    qt = x(1:4);
    ws_t = x(5:7);
    ww_t = x(8:10);
    dt = x(11:13);

    % - measurement output -
    % star tracker output
    p_e = star_obser+randn(3,1);
    q_e = [ p_e'  sqrt(1-(p_e'*p_e)) ]';
    qstar = qcross(q_e) * qt;

    % reaction wheel angular momentum
    w_wheel = ww_t + wheel_h_obser/Jw*randn(3,1);


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

    % -- 1st step(Propagation) --
    % - state propagation -
    xh = [ qh' ws_h' ww_h' dh' ]';
    xh = rk4(@EstiDyn,xh,torque_oder,delt);

    qh = xh(1:4);
    ws_h = xh(5:7);
    ww_h = xh(8:10);

    % transition matrix
    A = [ -skew(ws_h)  eye(3)  zeros(3,6);
          zeros(3)  inv(J)*(-skew(ws_h)*J + skew(J*ws_h) + skew(Jw*ww_h))  inv(J)*(-skew(ws_h)*Jw)
          zeros(3,12);
          zeros(3,12) ];

    PHi = eye(12) + delt*A;

    % state error covariance propagation
    P = PHi*P*PHi' + Qk;
    % 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_q = qcross(qstar) * qinv(qh);
    del_p = del_q(1:3);

    del_ww = w_wheel - ww_h;

    del_y = [ 2*del_p' del_ww' ]';
    del_x = K * del_y;

    % state update
    del_alpha = del_x(1:3);
    qh = qcross([0.5*del_alpha; 1])*qh;
    qh = qh/norm(qh);

    ws_h = ws_h + del_x(4:6);
    ww_h = ww_h + del_x(7:9);
    dh = dh + del_x(10:12);

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

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

    % data save
    time(index) = index*delt;

    qe(index,:) = (qcross(qh)*qinv(qt))';
    wse(index,:) = ws_t' - ws_h';
    wwe(index,:) = ww_t' - ww_h';
    de(index,:) = dt' - dh';
    qtc(index,:) = qt';
    qhc(index,:) = qh';
    dtc(index,:) = dt';
    dhc(index,:) = dh';

    Pc(index,:) = 3*sqrt(diag(P)');


end
```