SMALL SATELLITE ATTITUDE CONTROL BASED ON A KALMAN FILTER[*]

ROBERT CLEMENTS, PEDRO TAVARES, PEDRO LIMA

Instituto de Sistemas e Robótica, Instituto Superior Técnico
Torre Norte, Av. Rovisco Pais, 1-1049-001 Lisboa
Fax: +351 218418291, E-mail: pal@isr.ist.utl.pt

Abstract: This paper describes the development of an Extended Kalman Filter attitude estimator and its implementation in a small satellite control loop. Sensor models have first been added to an existing satellite simulation. The attitude and angular velocities are estimated from the available sensors and methods of tuning the filter based on Genetic Algorithms are discussed. Simulation results are presented for the estimator algorithm used with different control algorithms.

Key Words: Extended Kalman Filter, Genetic Algorithms, Sun Sensor, Magnetometer, Small Satellites.

1. INTRODUCTION

Small satellites have gained increased popularity since the early eighties, due to their relative low cost and fast turn-around time (from contract to launch). Nevertheless, this comes at the cost of less powerful sensors and actuators, as well as reduced computational power, due to size and weight limitations. Among other sub-systems, the Attitude Determination and Control System (ADCS) is affected by this trade-off, leading to more challenging attitude control and determination problems. The attitude is described by the rotation of the satellite body frame w.r.t. a local orbital co-ordinate frame.

The purpose of this work was to develop an attitude estimator for small satellites and to test it within the control loop in a simulator. This was done in three parts: i) development of realistic simulations of the satellite's sensors; ii) development of an attitude estimator algorithm; iii) integration of the estimator and the attitude controllers previously developed in the closed loop. The complete system is shown in Fig. 1. A detailed description of the satellite simulator and controllers is given in references [1] [4], [5] and [6].

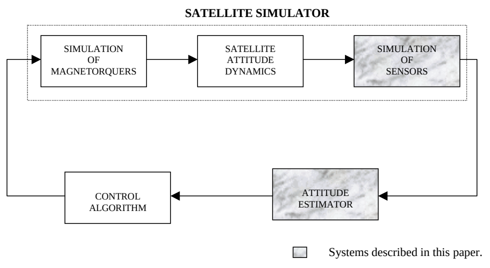
SATELLITE SIMULATOR
SIMULATION OF MAGNETORQUERS
SATELLITE ATTITUDE DYNAMICS
SIMULATION OF SENSORS
CONTROL ALGORITHM
ATTITUDE ESTIMATOR
Systems described in this paper.

Fig. 1 - Satellite closed loop control simulator.

The small satellite PoSat-1 was used as a case study. PoSat-1 is a 50Kg micro-satellite launched in 1993 as a technology demonstration. Throughout this work it is assumed that the system is being developed for this satellite. Nevertheless, the ideas and methodologies can be adapted to other satellite configurations. In Section 2 the realistic sensor simulation is described. Section 3 goes through the particularities of using an Extended Kalman Filter

[*] This work is supported by PRAXIS XXI program project PRAXIS/3/3.1/CTAE/1942/95 and by a grant from the Imperial College of Science and Technology, London, UK.

for attitude estimation. The filter tuning based on Genetic Algorithms is covered in Section 4. The results of closed loop attitude control with and without the estimator for three different control algorithms are discussed in Section 5. The paper ends with conclusions and prospects for future work, in Section 6.

2. SENSOR SIMULATION

Two PoSat-1 sensors are available for attitude estimation: the magnetometer and the sun sensor. The magnetometer measures the geomagnetic field vector and the attitude is determined by comparing this vector with the expected geomagnetic field at the current orbital location, based on the widely used IGRF model [8]. Therefore, the sensor reading can be simulated by rotating the current geomagnetic field vector into the satellite co-ordinate system (SCS) [5].

The Sun sensor measures the relative Sun and satellite locations from the angle of incidence of the sunlight on the sensor. From the knowledge of the Sun and spacecraft orbital locations, the current and expected measurements can be compared to determine the attitude. PoSat-1 has two Sun sensors with each sensor having two channels. Both sensor's field of view lie in the x-y plane, one looking in the positive y-direction, the other in the negative x-direction (in SCS).

A number of common Sun sensor designs are given in [8]. PoSat-1 Sun sensors layout is that shown in Fig. 2. Each sensor consists of two light sensitive cells, therefore producing the two-channel output. Each cell is angled to the horizontal as shown in the figure. To implement this design it is necessary to determine the geometry of the sensor (i.e. angles α and β). Two key characteristics in the telemetry data from PoSat-1 allow these angles to be determined. These are that both channels of the sensor start reading at the same time and that one channel starts reading at a non-zero, positive value. The satellite technical documentation also states that the field of view should be $\pm 60^0$. To satisfy these requirements $\alpha = \beta = 30^o$. This design is just shown in two dimensions. In three dimensions the surrounding structure would provide a similar $\pm 60^o$ view limit.

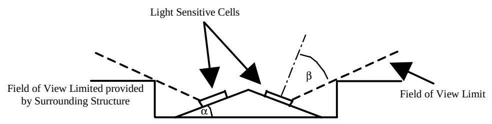

Fig. 2 - Sun Sensor Design

The simulator must also consider the possibility of the Earth being between the satellite and the Sun. This case is simply detected by calculating if the angle between the vector to the centre of the Earth and the vector to the Sun is less than the angle of the radius of the Earth as seen from the satellite. The angle of the radius of the Earth is pre-calculated within the simulator from the orbit altitude, so can be considered known at all time.

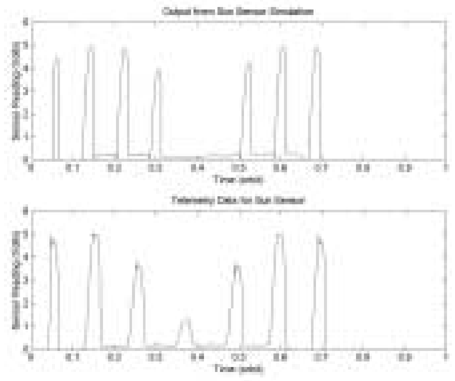

Fig. 3 - Comparison of Telemetry and simulation

It is clear from the telemetry that the value of output from the sensors when they can't see the Sun varies. Initial thought might suggest that the off value of the sensor should always be zero. However, it was realised that this variation was due to the sensor picking up sunlight reflected from the Earth. So, if the satellite is over the dark side of the Earth, the off value will be zero, if it is over the lit side it will have a slightly increased value. The magnitude of this increase will depend on factors like weather conditions, current altitude and whether the satellite is over land or sea. The 'off' value over the lit side of the Earth has thus been modelled using a random noise signal which gives values similar to those seen in the telemetry.

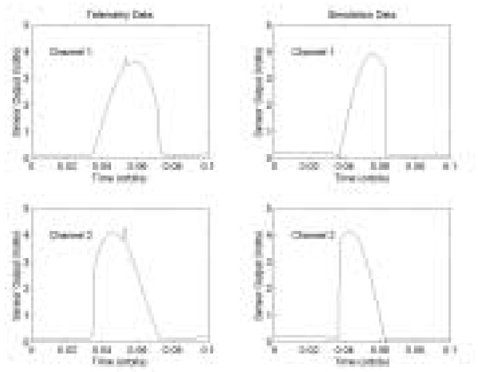

Fig. 4 - Comparison of Telemetry and Simulation

Comparison of the output from the simulator with telemetry data from PoSat-1 can be seen in Fig. 3 and 4. Figure 3 shows a comparison for one orbit (showing just one channel). It can be seen that real and simulated data are very similar. The differences in number and magnitude of each peak are due to differences in attitude and spin rate of the satellite in the simulation and in reality. The attitude of PoSat-1 is not known exactly so it is not possible to get the simulator to have exactly the same results. The modelling of the off-value can also be seen in this figure. Figure 4 shows the comparison of one revolution of the satellite (showing two channels). It can be seen that the cut-off times of the signals as the Sun passes out of the field of view of each channel are very similar. It is also clear to see that both channels start giving a reading at the same point in time. The spike that appears just before each peak is due to interference on the telemetry communication link and is thus not simulated.

3. KALMAN FILTER ESTIMATOR

In this section the attitude estimator based on an Extended Kalman Filter estimator algorithm introduced in [2] is described. This base algorithm, whose equations are presented in Appendix A, has been modified to include the sun sensor simulation that was developed in the previous section. A quaternion normalisation has also been added to the method.

A proper quaternion of rotation possesses the quality $q^Tq = 1$. It was shown in [2] that if this condition is enforced after each state calculation the estimator error will be reduced. This has therefore been included in the estimator. The calculation necessary to normalise the quaternion is explained fully in [2]. To use the Sun sensor within the estimator the Sun's position from the satellite given the sensor reading must be found. Each channel of the Sun sensor gives an angle to the Sun. Using one single channel the vector must be somewhere on a cone. From two channels, two cones can be generated, thus, giving two possible sun vectors. If the position of the Sun is known a few seconds before we can determine which of the two intersections is most likely to be correct.

All Sun sensors are in the x-y plane. The SCS co-ordinate system is rotated in the x-y plane so that the new x-axis is aligned with the sensor plane normal axis. This new co-ordinate system is referred to as the Sun Sensor Co-ordinate System (SSCS). In this co-ordinate system it can be easily seen that the vector of all possible Sun vectors is given by,

$$\begin{bmatrix}\cos\alpha & \sin\alpha\cos\phi & \sin\alpha\sin\phi\end{bmatrix}^T \qquad (1)$$

where α is the Sun sensor reading and ϕ is the parameter describing the cone. This can then be rotated back into SCS using the rotation matrix:

$$\begin{bmatrix}x\\y\\z\end{bmatrix} = \begin{bmatrix}\cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}{}^{ss}x\\{}^{ss}y\\{}^{ss}z\end{bmatrix}. \qquad (2)$$

Considering first the Sun sensor which points in the positive y-direction, θ for channel 1 will be 60^o and 120^o for channel two. Therefore, using (1) and (2) with the two values for θ gives the two cone vectors.

$$\begin{bmatrix}\frac{1}{2}\cos\alpha - \frac{\sqrt{3}}{2}\sin\alpha\cos\phi & \frac{\sqrt{3}}{2}\cos\alpha + \frac{1}{2}\sin\alpha\cos\phi & \sin\alpha\sin\phi\end{bmatrix}^T$$

and

$$\begin{bmatrix}-\frac{1}{2}\cos\alpha - \frac{\sqrt{3}}{2}\sin\alpha\cos\phi & \frac{\sqrt{3}}{2}\cos\alpha - \frac{1}{2}\sin\alpha\cos\phi & \sin\alpha\sin\phi\end{bmatrix}^T . \qquad (3)$$

Equating these two vectors and solving for ε and ϕ,

$$\cos\phi = \frac{2\cos\beta - \cos\alpha}{\sqrt{3}\sin\alpha} \qquad (4)$$

$$\cos\varepsilon = \frac{\cos\beta - 2\cos\alpha}{\sqrt{3}\sin\beta} \qquad (5)$$

Therefore, given both channels of the sensor reading, (i.e. α and β), ϕ or ε can be found from (4) and (5). Notice that there will be two values of ϕ or ε because of the inverse cosine. These values can then be substituted into (3) to give the two possible Sun vectors. The correct vector can then be selected on the basis of which one is closer to the previous estimate. Because the sample time is very small compared to the speed at which the satellite moves this should always be reliable.

The filter algorithm must now be modified to deal with two sensor readings. The equations in the estimator will be different depending on whether the Sun sensor measurements are available or not. Sun sensor measurements were not available, the equations do not change. Otherwise, equation (12) in Appendix A will change to

$$d\hat{z}_{k+1/k+1} = d\hat{z}_{k+1/k} + K_{k+1}\left[\begin{bmatrix}(b_{meas,k+1} - D\times b_{orb,k+1})\\(s_{meas,k+1} - D\times s_{orb,k+1})\end{bmatrix} - H_{k+1/k}d\hat{z}_{k+1/k}\right], \quad (6)$$

where $s_{meas,k+1}$ is the Sun Vector measurement (in SCS) at time k+1, $s_{orb,k+1}$ is the Sun Vector predicted by the model in the orbital co-ordinate system (OCS) [5] at time k+1. Note, if two Sun sensor readings are available the average of the two readings is used.

To use (6), K needs to be extended to a (6x7) matrix (previously it was (3x7)). This is achieved by redefining H, which is defined in the appendix as equation (11), the output matrix. So, if the Sun sensor measurement is available, (11) becomes

$$H_{k+1/k} = \begin{bmatrix}0_{6x3} & h_1 & h_2 & h_3 & h_4\end{bmatrix}, \qquad (7)$$

where $h_i = \begin{bmatrix}\dfrac{dD(\hat{q}_k)}{d\hat{q}_{i,k}}b_{orb,k+1}\\ \dfrac{dD(\hat{q}_k)}{d\hat{q}_{i,k}}s_{orb,k+1}\end{bmatrix}$.

The matrix R, the measurement error covariance, also needs to be redefined as a (6x6) matrix if the Sun sensor measurement is available.

4. ESTIMATOR TUNING

There are three matrices used within the estimator that must be determined before it can be used. These are $P_{o\ (7x7)}$ – the covariance of the error in the initial state, $Q_{(7x7)}$ – the covariance of the system model error, and $R_{(3x3\ \text{or}\ 6x6)}$[1]– the covariance of the measurement error. We have chosen to tune these covariance matrices instead of making them match sensor noise characteristics, so as to improve closed loop control performance.

Initially it would seem that there are 77 parameters which need to be determined (noting that covariance matrices are always symmetric). To simplify the problem, various parameters within each matrix can be grouped by assuming that the error covariance in the terms will be the same. For example, it is reasonable to assume that the covariance of the errors in the Ω^x and Ω^y terms will be the same. The Ω^z term may to different because it involves the spin of the satellite.

This grouping can reduce the total number of parameters to 32. The P matrix can be tuned separately from Q and R because P only effects the initial transient errors. Q and R can be tuned ignoring transient effects, therefore reducing the problem further.

Because of the multi-parameter, non-linear, multi-minimum nature of this optimisation problem a logical approach was to use a genetic algorithm (GA) [3]. A standard GA has been used, with chromosomes containing all the elements of the estimator covariance matrices. Each population was evaluated by simulating each chromosome over a number of orbits and evaluating a cost function containing the average estimator error. New populations were generated using mutations and crossover with parents being selected using a roulette wheel selection process. This is a very effective method of tuning the estimator although, due to simulation time, it can be very slow.

Table 1 - Estimator Accuracy Results

	Mean	Standard Deviation	Worst
Pointing Error (deg)	0.58	0.56	1.72
Spin Rate Error, %	0.39	0.37	1.23

The best average estimator results, shown in Table 1, were obtained in closed loop using a predictive controller, described in the following section, over a number of orbits. These results were obtained over twenty simulations each ten orbits long with different starting conditions.

[1] The R matrix size changes depending on whether the Sun sensor is included or not.

5. CLOSED LOOP CONTROL

There are three controllers of interest that are available within the simulator: the Predictive controller [4], the Energy controller [9] and the Alpha-Beta controller [7]. The predictive controller attempts to predict, for all possible actuations, the change in angular velocity of the satellite if that actuation was applied. It then determines which of these will cause the greatest reduction in the satellite's kinetic energy. As a result, a time-varying control law is applied. The energy controller uses the control law

$$^{c}m(t) = h\,^{c}\Omega_{co}(t) \times {}^{c}B(t)\,, \qquad (8)$$

where h is a positive constant. The alpha-beta controller is the controller currently used on PoSat-1. It uses the control law

$$^{c}m_{z} = k\left(\frac{d\beta}{dt} - \frac{d\alpha}{dt}\right), \qquad (9)$$

where α is the angle between the z-axis of the OCS and the expected geomagnetic field and β is the angle between the z-axis of the OCS and the measured geomagnetic field. The alpha-beta controller does not require an estimator. Both the Predictive controller and the Energy controller can control the spin of the satellite as well as the attitude, while the alpha-beta controller only controls attitude. These controllers are described and their performance without an estimator compared in detail in [5].

To compare the effectiveness of the controllers and estimator in the loop, twenty simulations, each ten orbits in duration with varying starting conditions, were made. The results presented below are average results from these tests, where spin control as well as attitude stabilisation were envisaged. Since the alpha-beta controller can not control spin, a different test with no spin control required was used to make comparisons with this controller.
To check the effect of the estimator on the energy and predictive controllers, tests were first conducted without the estimator. Table 2 shows results for the predictive and energy controllers.

Table 2 – Simulation Results without Estimator

	Settling Time to 5° (orbits)	Pointing Accuracy (deg)	Spin Rate Accuracy (rad/s)	Energy (Joules)
Predictive Controller	2.40	1.86	4.6×10^{-5}	2.70×10^{5}
Energy Controller	2.75	1.85	8.8×10^{-4}	2.56×10^{5}

Comparing the performance of the two controllers without the estimator, both have very similar results. Considering the standard deviations of these averages (not shown here) the slight differences are insignificant. This is true with the exception of the spin rate accuracy where the predictive controller is considerably better.

Table 3 – Simulation Results with Estimator

	Settling Time to 5^o (orbits)	Pointing Accuracy (deg)	Spin Rate Accuracy (rad/s)	Energy (Joules)
Predictive Controller (without Sun Sensor)	8.76	3.22	$1.3x10^{-4}$	$3.84x10^{5}$
Predictive Controller (with Sun Sensor)	9.18	3.14	$1.2x10^{-4}$	$4.09x10^{5}$
Energy Controller (with Sun Sensor)	2.74	2.04	$6.5x10^{-4}$	$2.70x10^{5}$

Table 3 shows the results for the two controllers with the estimator included. This table also shows the results for the estimate with and without the sun sensor, for the predictive controller.

Looking at the effect of the estimator on the predictive controller, the pointing accuracy has deteriorated from ≈1.8^o to ≈3.2^o. Similarly, the spin rate accuracy has been halved. The energy consumed has also increased slightly. Considering the effect of the estimator on the energy controller, the reduction in accuracy is much smaller. Pointing accuracy is reduced from ≈1.8^o to ≈2.0^o. Spin rate and energy consumed are not significantly affected. Thus, comparing the controller + estimator algorithms, the energy controller is better from pointing accuracy, rapid settling and energy standpoints. However, the predictive controller still has a better spin rate accuracy.

Table 4 – Comparisons to Alpha-Beta Controller

	Settling Time to 1^o (orbits)	Pointing Accuracy (deg)	Energy Consumed (Joules)
Alpha-Beta Controller	*No Settling*	1.26	$3.20x10^{3}$
Predictive Controller	4.00	0.14	$1.66x10^{5}$
Energy Controller	0.80	0.01	$2.51x10^{5}$
Tuned Predictive Controller	1.80	0.02	$7.67x10^{3}$

For a controller + estimator algorithm to be used with PoSat-1, it must perform better than its current alpha-beta controller. In Table 4, results of the alpha-beta, predictive and energy controllers are compared with no estimator in the loop and *no spin* control. The alpha-beta controller does not require an estimator and can not control spin, hence the three control algorithms were compared under the same circumstances.

The energy controller clearly has the best pointing accuracy and settling time. However, the energy consumed is considerably larger than that used by the alpha-beta controller. The predictive controller outperforms the alpha-beta controller regarding pointing accuracy and settling time, but energy consumption is high too. This shows that, to use either the predictive or the energy controller their energy consumption must be reduced. Both the predictive controller and the energy controller contain a number of parameters that can be adjusted to change their performance. To improve the overall controller + estimator performance, the estimator and the controller parameters can be tuned together in closed loop. Only limited exploration of this sort of tuning has been possible during this work, but the last row of Table 4 shows results that were obtained after the predictive controller was tuned to reduce energy consumption. Comparing these to the alpha-beta controller results, the controller now uses similar energy but with a pointing accuracy about 50 times better. It is also noticeable that these accuracy results are better than the pre-tuning controller results. Furthermore, it should be noted that the energy controller was tested with unrestricted actuators, while the predictive controller uses restricted actuators. Nonetheless, their performance under such distinct conditions is similar. Usually, the energy controller produces bad results with restricted actuators.

6. CONCLUSIONS AND FUTURE WORK

A Kalman Filter attitude estimator using magnetometers and sun sensors has been developed and implemented showing, in realistic simulations, a pointing accuracy of ≈0.6^o and a spin rate error of ≈0.4%. A genetic algorithm has been used to tune the Kalman filter estimator. With the estimator in the loop the system worked best with the energy controller where there was an accuracy loss of only 0.2 degrees. With the predictive controller the drop in performance was nearly 1.4 degrees. The obtained results suggest that these controllers outperform the benchmark alpha-beta controller regarding accuracy.

Further work on closed loop tuning could reap significant rewards. It has been shown that tuning the controllers can make major improvements in areas like energy consumption and pointing accuracy. Another important area not considered so far is the processing time required by the estimator and controller. On a small satellite it is important to minimise the computation effort that is required to control the attitude. Currently the predictive and energy controllers with the estimator use about three times more CPU time than the alpha-beta controller. One potential method to reduce the computation effort associated to the estimation would be to linearise the equations of motion about a number of key attitudes and then use a gain scheduling controller to determine the motion between these linearised points. At the moment the equations are linearised at every time step, a time consuming process. In a real satellite, the attitude perturbation from the stabilised position will be small, and so few attitude way-points would be required to achieve accurate results.

REFERENCES

[1] Bak, T., Blanke, B., Wisniewsky, R., Astolfi, A., Spindler, K., Tavares, P., Tabuada, P., Lima, P. (2000), "Satellite Attitude Control Problem", *in COSY Book*, Springer-Verlag.

[2] Bar-Itzhack, I.Y., Oshman, Y. (1985), "Attitude Determination from Vector Observations: Quaternion Estimation", *IEEE Transaction on Aerospace and Electronic Systems*, Vol.21, No.1, pp 128-135.

[3] Goldberg, D. (1989), *Genetic Algorithms in Search: Optimisation and Machine Learning*, Addison-Wesley.

[4] Tabuada, P., Alves, P., Tavares, P., Lima, P. (1999) "A Predictive Algorithm for Attitude Stabilisation and Spin Control of Small Satellites", *Proc. of the European Control Conference* (ECC'99), Karlsruhe, Germany.

[5] Tabuada, P., Alves, P., Tavares, P., Lima, P. (1998), *Attitude Control Strategies for Small Satellites*, Institute for Systems and Robotics internal report RT-404-98.

[6] Tavares, P., Sousa, B., Lima, P. (1998), "A Simulator of Satellite Attitude Dynamics", *Proc. of the 3rd Portuguese Conference on Automatic Control*, Vol. II, pp. 459-464, Coimbra, Portugal.

[7] Ong, W. T. (1992), *Attitude Determination and Control of Low Earth Orbit Satellites*, MSc Thesis, Dept. of Electric and Electrical Engineering, University of Surrey, U.K.

[8] Wertz, J.R. (1990), *Spacecraft Attitude Determination and Control*, Kluwer Academic Publishers.

[9] Wisniewski, R. (1996), *Satellite Attitude Control using only Electromagnetic Actuation*, Ph.D. Thesis, Dept. of Control Engineering, Aalborg University, Danmark

APPENDIX A
Extended Kalman Filter Algorithm [2]

1) Calculation of Kalman Gain

$$K_{k+1} = P_{k+1/k} H^T_{k+1/k} \left[H_{k+1/k} P_{k+1/k} H^T_{k+1/k} + R \right]^{-1} \quad (10)$$

where K_{k+1} is the Kalman Gain Matrix (7x3), $P_{k+1/k}$ is the perturbation covariance matrix (7x7) at time k+1 given the measurements at time k, R is the measurement error covariance matrix (3x3) and $H_{k+1/k}$ is a (3x7) matrix defined as follows:

$$H_{k+1/k} = \left[0_{3x3} \quad h_1 \quad h_2 \quad h_3 \quad h_4 \right] \quad (11)$$

where $h_i = \frac{dD(\hat{q}_k)}{d\hat{q}_{i,k}} b_{orb,k+1}$, $\hat{q}_{i,k}$ is the ith estimated component of quaternion q at time k and $b_{orb,k+1}$ is the model magnetic field vector (in OCS) at time k+1.

2) Update the State

$$d\hat{z}_{k+1/k+1} = K_{k+1}(b_{meas,k+1} - D \times b_{orb,k+1}) \quad (12)$$

$$\hat{z}_{k+1/k+1} = \hat{z}_{k+1/k} + d\hat{z}_{k+1/k+1} \quad (13)$$

where $\hat{z}_{k+1/k}$ is the estimated state vector at time k+1 given the measurements at k,

$$z = \left[\boldsymbol{\Omega}^x_{si} \quad \boldsymbol{\Omega}^y_{si} \quad \boldsymbol{\Omega}^z_{si} \quad q_1 \quad q_2 \quad q_3 \quad q_4 \right]^T$$

$b_{meas,k+1}$ is the measurement magnetic field vector (in SCS) at time k+1, Ω^x_{si} is the x component of the angular velocity in SCS w.r.t. ICS and D is the rotation matrix from OCS to SCS as defined below:

$$D = \begin{bmatrix} \hat{q}^2_{1,k} - \hat{q}^2_{2,k} - \hat{q}^2_{3,k} + \hat{q}^2_{4,k} & 2(\hat{q}_{1,k}\hat{q}_{2,k} + \hat{q}_{3,k}\hat{q}_{4,k}) & 2(\hat{q}_{1,k}\hat{q}_{3,k} - \hat{q}_{2,k}\hat{q}_{4,k}) \\ 2(\hat{q}_{1,k}\hat{q}_{2,k} - \hat{q}_{3,k}\hat{q}_{4,k}) & -\hat{q}^2_{1,k} + \hat{q}^2_{2,k} - \hat{q}^2_{3,k} + \hat{q}^2_{4,k} & 2(\hat{q}_{2,k}\hat{q}_{3,k} + \hat{q}_{1,k}\hat{q}_{4,k}) \\ 2(\hat{q}_{1,k}\hat{q}_{3,k} + \hat{q}_{2,k}\hat{q}_{4,k}) & 2(\hat{q}_{2,k}\hat{q}_{3,k} - \hat{q}_{1,k}\hat{q}_{4,k}) & -\hat{q}^2_{1,k} - \hat{q}^2_{2,k} + \hat{q}^2_{3,k} + \hat{q}^2_{4,k} \end{bmatrix} \quad (14)$$

3) Update the Covariance Matrix
The perturbation covariance matrix is updated using (16) where $H_{k+1/k+1}$ is calculated from (12) except that the quaternions at time k+1 are used instead of at time k.

$$P_{k+1/k+1} = \left[I_{(7x7)} - K_{k+1} H_{k+1/k+1} \right] P_{k+1/k} \left[I_{(7x7)} - K_{k+1} H_{k+1/k+1} \right]^T + K_{k+1} R K^T_{k+1} \quad (15)$$

4) Propagate State Vector

$$\hat{z}_{k+2/k+1} = \int_{k+1}^{k+2} f_{k+1}(\hat{z}_{k+1/k+1}, k+1)\,dt + \hat{z}_{k+1/k+1} \quad (16)$$

where $f_{k+1}(\hat{z}_{k+1/k+1}, k+1) = \begin{bmatrix} \dot{\boldsymbol{\Omega}}_{si} \\ \dot{q} \end{bmatrix}$,

$\dot{q} = \underline{\boldsymbol{\Omega}} q$, and $\dot{\boldsymbol{\Omega}}_{si} = I^{-1}\left[n_{gg} + n_{ctrl} - \boldsymbol{\Omega}_{si} \times (I\boldsymbol{\Omega}_{si}) \right]$

(n_{ctrl} is the control moments vector).

$$n_{gg} = 3\omega_o^2 (I_x - I_z) D_{33} \left[-D_{23} \quad D_{13} \quad 0 \right]^T$$

$$\underline{\boldsymbol{\Omega}} = \frac{1}{2} \begin{bmatrix} 0 & \boldsymbol{\Omega}^z_{so} & -\boldsymbol{\Omega}^y_{so} & \boldsymbol{\Omega}^x_{so} \\ -\boldsymbol{\Omega}^z_{so} & 0 & \boldsymbol{\Omega}^x_{so} & \boldsymbol{\Omega}^y_{so} \\ \boldsymbol{\Omega}^y_{so} & -\boldsymbol{\Omega}^x_{so} & 0 & \boldsymbol{\Omega}^z_{so} \\ -\boldsymbol{\Omega}^x_{so} & -\boldsymbol{\Omega}^y_{so} & -\boldsymbol{\Omega}^z_{so} & 0 \end{bmatrix}$$

D_{ij} is an element of D and Ω^x_{so} is the x component of angular velocity in SCS w.r.t. OCS.

5) Propagate the Perturbation Covariance Matrix

$$P_{k+2/k+1} = \boldsymbol{\Phi}_{k+2/k+1} P_{k+1/k+1} \boldsymbol{\Phi}^T_{k+2/k+1} + Q, \quad (17)$$

where Q is the system error covariance matrix,

$$\boldsymbol{\Phi}_{k+2/k+1} = I_{(7x7)} + F(\hat{z}_{k+1/k+1}, k+1)\Delta t,$$

$$F(\hat{z}_{k+1/k+1}, k+1)\delta z = \begin{bmatrix} \delta\dot{\boldsymbol{\Omega}}_{si} \\ \delta\dot{q} \end{bmatrix},$$

$$\delta\dot{\boldsymbol{\Omega}}_{si} = I^{-1}\left[\delta n_{gg} - \delta\boldsymbol{\Omega}_{si} \times I\boldsymbol{\Omega}_{si} - \boldsymbol{\Omega}_{si} \times I\delta\boldsymbol{\Omega}_{si} \right],$$

δn_{gg} is given by

$$\delta n_{gg} = 6\omega_o^2 (I_x - I_z). \quad (18)$$

$$\cdot \begin{bmatrix} -\hat{D}_{33}\hat{q}_4 + \hat{D}_{23}\hat{q}_1 & -\hat{D}_{33}\hat{q}_3 + \hat{D}_{23}\hat{q}_2 & -\hat{D}_{33}\hat{q}_2 - \hat{D}_{23}\hat{q}_3 & -\hat{D}_{33}\hat{q}_1 - \hat{D}_{23}\hat{q}_4 \\ \hat{D}_{33}\hat{q}_3 - \hat{D}_{13}\hat{q}_1 & -\hat{D}_{33}\hat{q}_4 - \hat{D}_{13}\hat{q}_2 & \hat{D}_{33}\hat{q}_1 + \hat{D}_{13}\hat{q}_3 & -\hat{D}_{33}\hat{q}_2 + \hat{D}_{13}\hat{q}_4 \\ & & & \end{bmatrix} \delta q$$

$$\delta\dot{q} = \underline{\boldsymbol{\Omega}}_{si}\delta q + \underline{\beta}\delta\boldsymbol{\Omega}_{si}, \quad \underline{\beta} = \frac{1}{2} \begin{bmatrix} \hat{q}_4 & -\hat{q}_3 & \hat{q}_2 \\ \hat{q}_3 & \hat{q}_4 & -\hat{q}_1 \\ -\hat{q}_2 & \hat{q}_1 & \hat{q}_4 \\ -\hat{q}_1 & -\hat{q}_2 & -\hat{q}_3 \end{bmatrix}.$$