A quaternion based extended Kalman filter for GNSS-INS

Internal name code: sensor fusion.

October 2, 2017

1 Introduction

In this work we employed Brain One, a smart telemetry sensor, oriented to motorbike market, to retrieve the raw data for our experiments. Brain one core components are the navigation system and the estimation of roll angle: a good estimation of roll angle is very important, for example, for controlling the traction of high-level motorbikes (e.g. MotoGP). The navigation system has been formalized and implemented form scratch using quaternion representation and Extended Kalman Filter. The code was written in Matlab and C. In this report we present both the direct and indirect navigation models, while only the direct model has been implemented.

2 Navigation system

2.1 Problem statement

Brain One is a smart telemetry sensor, oriented to motorbike market, whose core components are

- a 9-axes inertial measurement unit (henceforth called IMU)
- a high frequency GPS-GLONASS sensor allowing for 20Hz sample rate (henceforth called GPS sensor for the sake of simplicity)

It aims at providing accurate estimates of

- motorbike position
- motorbike roll angle
- speed
- lateral acceleration
- motion acceleration/deceleration
- accurate lap time

- RPM

Position, velocity, roll angle, acceleration and lap time can be estimated through a GNSS-aided inertial navigation algorithm. The main advantages of GNSS-INS algorithm are the following:

- Position and velocity are estimated even if no GPS data are available.
- Position estimates are updated according to IMU frequency. As a consequence, we can attain higher precision in lap time estimates.
- Attitude is one of the available outputs. It allows us to compute linear acceleration and correctly project it along motion and lateral direction.
- It is possible to compensate bias drift that affects IMU measurements.
- Kalman filtering paradigm can be used for roll angle estimation through an adequate dynamic model.

2.2 Main results

We propose an efficient quaternion-based Extended Kalman Filter algorithm for GNSS-aided inertial navigation. It leverages quaternion properties to achieve numerical robustness and exploits a straightforward direct Kalman Filter implementation that makes it intuitive and computationally efficient.

We prototyped the algorithm in Matlab, then implemented it in C and test it on Brain One GNSS-INS platform.

2.3 EKF for aided inertial navigation

2.3.1 Inertial navigation state-space model

Next we introduce a state-space model for a GNSS-INS. We will consider the geographic frame as the navigation reference frame. Since the origin of the geographic frame moves with the vehicle and is the projection of vehicle frame origin onto the reference ellipsoid, the position of the vehicle in the geographic frame is $\mathbf{x}_g = \begin{bmatrix} 0 & 0 & -h \end{bmatrix}^T$, where h stands for the altitude of the body, while latitude ϕ and longitude λ define the position of the geographic frame origin (vehicle frame projection) on the reference ellipsoid. By adding velocity $\mathbf{v}(\mathrm{t})$ and attitude $\mathbf{q}(\mathrm{t})$ we get a complete description of the inertial navigation system. Thus we define

$$\mathbf{p} := \begin{bmatrix} \phi \\ \lambda \\ h \end{bmatrix}.$$

We assume NED (i.e. North East Down) convention for the navigation frame. Then, velocity is given by

$$\mathbf{v} = \begin{bmatrix} v_n \\ v_e \\ v_d \end{bmatrix},$$

where $\mathbf{v_n}$ stands for velocity in North direction, $\mathbf{v_e}$ for velocity in East direction and $\mathbf{v_d}$ for velocity in Down direction. They are all measured in m/s. For the ex of numerical robustness we chose to represent attitude by means of quaternions.

Let $q := [q_0\, q_1\, q_2\, q_3]$ be the quaternion associated to the rotation from navigation to body frame (i.e. corresponding to the rotation matrix R_n^b). Then, if $\mathbf{r}^n$ is a vector expressed in the navigation frame, its coordinated w.r.t. the body frame are given by

$$\mathbf{r}^b = \mathbf{q} \otimes \mathbf{r}^n \otimes \mathbf{q}^*, \tag{1}$$

where $\otimes$ denotes quaternion product and

$$\mathbf{q}^* := [q_0\ -q_1\ -q_2\ -q_3] \tag{2}$$

is the conjugate quaternion of $\mathbf{q}$. In order to compensate gyroscopes bias drift we will consider the augmented state-space

$$\mathbf{x}(t) := \begin{bmatrix}\mathbf{p} & \mathbf{v} & \mathbf{q} & \mathbf{x_g}\end{bmatrix}^T, \tag{3}$$

where x_g describe the bias drift that affects gyros. Thus, we can write the following state-space model:

$$\begin{bmatrix}\dot{\phi}\\ \dot{\lambda}\\ \dot{h}\end{bmatrix} = \begin{bmatrix}\frac{1}{R_M+h} & 0 & 0\\ 0 & \frac{1}{(R_N+h)cos(\phi)} & 0\\ 0 & 0 & -1\end{bmatrix}\begin{bmatrix}v_n\\ v_e\\ v_d\end{bmatrix} \tag{4}$$

$$\begin{bmatrix}\dot{v_n}\\ \dot{v_e}\\ \dot{v_d}\end{bmatrix} = \begin{bmatrix}f_n\\ f_e\\ f_d\end{bmatrix} + g^n - \left[\begin{bmatrix}(\dot{\lambda}+2\omega_{ie})cos(\phi)\\ -\dot{\phi}\\ -(\dot{\lambda}+2\omega_{ie})sin(\phi)\end{bmatrix}\times\right]\begin{bmatrix}v_n\\ v_e\\ v_d\end{bmatrix} \tag{5}$$

$$\dot{q} = \frac{1}{2}\mathbf{Q}(q)(\Omega_{ib}^b - \Omega_{in}^b) \tag{6}$$

2.3.2 Extended Kalman filter design

The corresponding Kalman process model is given by

$$\begin{bmatrix}\dot{\phi}\\ \dot{\lambda}\\ \dot{h}\end{bmatrix} = \begin{bmatrix}\frac{1}{R_M+h} & 0 & 0\\ 0 & \frac{1}{(R_N+h)cos(\phi)} & 0\\ 0 & 0 & -1\end{bmatrix}\begin{bmatrix}v_n\\ v_e\\ v_d\end{bmatrix} + \nu_p, \tag{7}$$

$$\begin{bmatrix}\dot{v_n}\\ \dot{v_e}\\ \dot{v_d}\end{bmatrix} = \begin{bmatrix}f_n\\ f_e\\ f_d\end{bmatrix} + g^n - \left[\begin{bmatrix}(\dot{\lambda}+2\omega_{ie})cos(\phi)\\ -\dot{\phi}\\ -(\dot{\lambda}+2\omega_{ie})sin(\phi)\end{bmatrix}\times\right]\begin{bmatrix}v_n\\ v_e\\ v_d\end{bmatrix} + \nu_v, \tag{8}$$

$$\dot{q} = 0.5\mathbf{Q}(q)(\omega_{in}^b - \omega_{ib}^b) + \nu_q \tag{9}$$

The filter is set up by specifying the initial estimates $\hat{p}(0)$, $\hat{v}(0)$, $\hat{q}(0)$ and process noise second order stochastic description.

2.4 Kalman prediction

For the sake of simplicity, we assume that IMU data are synchronous (i.e. gyroscope, accelerometer and magnetometer provide data simultaneously). State is predicted according to the following model:

$$\hat{x}(t+1) = \hat{x}(t) + f(\hat{x}(t))\, dt.$$

Then prediction variance is updated according to

$$P(t+1|t) = (I + F_t dt)P(t|t)(I + F_t dt)' + Q_t$$

where Q_t is given by the discretization of process noise covariance matrix, whereas F_t is the Jacobian of $f(x)$ evaluated in $\hat{x}(t)$.

Process noise covariance Q is assumed to be diagonal. The design of Q is part of parameters tuning stage.

2.5 Measurement model

We consider a GNSS-aided inertial navigation system, so we can assume position and velocity measurements are available as geodetic coordinates and NED velocity vector, respectively. Let $\mathbf{z}$ be the available position and velocity data, then

$$\mathbf{z} := \begin{bmatrix} \phi_{GPS} \\ \lambda_{GPS} \\ h_{GPS} \end{bmatrix}$$

and the measurement model is

$$\mathbf{z} = \begin{bmatrix} I_{6\times 6} & 0_{6\times 7} \end{bmatrix} x + w_{p,v}$$

Based on accelerometer measurements, under the assumption that the sensor is at rest, we introduce $\mathbf{g}$ as an artificial measurement, whose observation model is given by

$$\mathbf{y} = h_{grav}(x) + w = \mathbf{q}^* \otimes \mathbf{a} \otimes \mathbf{q}$$

where $\mathbf{a}$ is the accelerometer measurement.

Based on magnetometer measurements, we introduce the artificial measurement $\mathbf{b}^n$, defined as an estimate of the local earth magnetic flux. We compute $\mathbf{b}^n$ *a la* Madgwick First we rotate the magnetometer measurement $\mathbf{m}$ according to current attitude estimate and obtain $\mathbf{m}^n$:

$$\mathbf{m}^n = \mathbf{q}^* \otimes \mathbf{m} \otimes \mathbf{q};$$

then, based on the assumption that Earth magnetic field has the form

$$\mathbf{b} = B \begin{bmatrix} cos\delta \\ 0 \\ sin\delta \end{bmatrix},$$

where δ is the magnetic dip ,we force the resulting vector to have the second component equal to zero and obtain

$$\mathbf{b}^n := \begin{bmatrix} \sqrt{{m_x}^{n\,2} + {m_y}^{n\,2}} & 0 & {m_z}^n \end{bmatrix}.$$

The corresponding measurement model is given by

$$y = h_{mag}(x) + w = \mathbf{q}^* \otimes \mathbf{m} \otimes \mathbf{q}.$$

2.5.1 Kalman filter update

We assume that

- Update is performed as soon as new IMU data are available
- We disregard accelerometer measurements in KF update if the sensor is not at rest

State is updated through equation

$$\hat{x}(t+1 \mid t+1) = \hat{x}(t+1 \mid t) + K(\tilde{z}(t+1) - \hat{y}(t+1))$$

where $\tilde{z}$ is the available measurement and $\hat{y}(t+1)$ is the output prediction

$$\hat{y}(t+1) := h(\hat{x}(t+1 \mid t))$$

and the Kalman gain K is given by

$$K := (I - PH')(HPH' + R)^{-1}.$$

where H is the Jacobian of observation function $h(\cdot)$.

2.5.2 Output metrics

Next we describe how the KPI's are derived based on Kalman filter state estimation.

Speed Speed is defined as the absolute value of velocity along motion direction. By assumins sideslip is negligible (notice this could be an oversimplification in other contexts), speed is defined as

$$s := \sqrt{\hat{v}_n^2 + \hat{v}_e^2}. \tag{10}$$

Motion and lateral acceleration Once an attitude estimate $\hat{q}$ is available, we can estimate linear acceleration in navigation frame by subtracting gravity acceleration form measurements. Let $\mathbf{v}$ be the velocity estimate (i.e. $\mathbf{v} := \hat{\mathbf{v}}_n + \hat{\mathbf{v}}_e$) and $\hat{\mathbf{a}}_{nav}$ be the estimate of linear acceleration in navigation frame. Then we have

$$\mathbf{v} \cdot \hat{\mathbf{a}}_{nav} := \|\mathbf{v}\| \|\hat{\mathbf{a}}_{nav}\| \, cos\alpha, \tag{11}$$

where α is the angle comprised between $\mathbf{v}$ and $\hat{\mathbf{a}}_{nav}$. Then, the absolute value of motion acceleration (i.e. acceleration along the direction of motion which is assumed to be the direction of $\mathbf{v}$ by neglecting sideslip) is given by

$$\|\hat{\mathbf{a}}_{mot}\| = \|\hat{\mathbf{a}}_{nav} \, cos\alpha\| = \frac{s}{\|\mathbf{v}\|}. \tag{12}$$

Finally, the absolute value of lateral acceleration is given by

$$\|\hat{\mathbf{a}}_{lat}\| = \|\hat{\mathbf{a}}_{nav} sin\alpha\| = \sqrt{\|\hat{\mathbf{a}}_{nav}\|^2 - \|\hat{\mathbf{a}}_{mot}\|^2}. \tag{13}$$

2.5.3 Advantages of the quaternion based approach

Attitude can be represented by means of

- Euler angles
- Rotation matrix
- Quaternions

Euler angles are often chosen because of their intuitive interpretation. Indeed, let ϕ, θ, and ψ be roll, pitch and yaw angle, respectively. Let $\{x_b, y_b, z_b\}$ be the current body frame. By rotating the body of $-\phi$ around its x-axis we obtain the new frame $\{x'_b \equiv x_b, y'_b, z'_b\}$. Then a rotation of $-\theta$ around y'_b axis yields the new frame $\{x_b", y_b" = y'_b, z_b"\}$. Finally, by rotating the body of $-\psi$ around axis $z_b"$ we get $\{x'''_b = x_n, y'''_b = y_n, z'''_b = z_b" = z_n\}$ where $\{x_n, y_n, z_n\}$ is the navigation frame. Unfortunately, Euler angles suffer from numerical instability and the Gimbal lock.

Alternatively, we could consider rotation matrix R_n^b. However, this imply dealing with a bigger state space representation and non-linear constraints (e.g. the orthogonality constraints on R_n^b.

Finally, we can represent 3D rotations by means of unit quaternions, also known as Euler–Rodrigues parameter. Quaternions are equivalent to the special unitary group description. Expressing rotations in 3D as unit quaternions has some advantages:

- Quaternions do not suffer from gimbal lock as Euler angles do.
- Concatenating rotations is computationally faster and numerically more stable.
- Extracting the angle and axis of rotation is simpler.

We propose an implementation of Extended Kalman Filtering for aided inertial navigation that is based on quaternions completely. Usually, even if quaternions appear in state space, Kalman Filtering update requires rotation matrix computation. This is not needed in our approach, avoiding potential numerical issues as the ones we experienced in an early stage when dealing with indirect Kalman Filter implementation that involved rotation matrix based update. As a result, we obtain a numerically robust attitude dynamics description.

2.5.4 Advantages of direct implementation

At the beginning we set up an indirect implementation of GPS-aided inertial navigation. This approach consists in two steps:

1. Mechanization
2. Kalman filter update

Firstly, mechanization provides a deterministic model for estimating position, velocity and attitude. Then, Kalman Filter provides a correction based on available measurements.

We next describe this approach for the sake of completeness. However, it turned out to be very difficult to optimize, because every time Kalman Filter

algorithm runs the state estimation computed through mechanization equations is updated and Kalman filter state is reset. Thus, it is very difficult to take into account uncertainty propagation and leverage on past estimates for smoothing.

Direct implementation is much more intuitive because is it a straightforward application of Kalman filtering theory. Kalman prediction occurs as soon as new IMU data are available. Then, based in measurements availability, Kalman update is performed to correct predictions. As a consequence, uncertainty propagation much easier to deal with and the algorithm is not as computationally burdensome as the indirect approach.

2.6 Implementation

2.6.1 Assumptions

The algorithm was implemented under the following assumptions:

Initialization Sensor is at rest at the beginning. This assumption is required for correct gyro bias, position and attitude initialization

Velocity constraints The velocity along vertical direction is assumed to be zero, because we model 2D motion of a vehicle over ground. This assumption is implemented by defining the artificial measurement $v_d = 0$.

Synchronous data IMU data, i.e. data from accelerometer, gyroscope and magnetometer are available simultaneously.

GPS data timing The time delay between GPS data and previous IMU data is assumed to be negligible.

2.7 Future work

2.7.1 Optimization

2.7.2 Filter performance

In order to improve the performance of the proposed GNSS-INS a better characterization of the sensor equipment is undeniably necessary. In particular we propose the following strategies:

1. Allan variance test for gyroscopes accelerometers and magnetometers.
2. Finer calibration.
3. Turn-on calibration routines for accelerometers and magnetometers.

2.7.3 Computational efficiency

The C implementation of the algorithm described in **??** can be improved in several ways. First of all all quaternion products should be written in close form, thus avoiding function calls. Secondly one should consider the option of defining a third observation function h_{mag} to account for the cases in which only magnetometer measurements are trusted. The Kalman gain computation will then involve the inversion of a 3×3 matrix, further eliminating the useless computational power consumption when accelerometer measurements are deemed unreliable by setting their variance to a huge number.

More performance, depending on the architecture the algorithm is running on, could be recovered by compiling the CBLAS and LAPACKE libraries included the Automatically Tuned Linear Algebra System (ATLAS) distribution. This should enable multithreaded vector and matrix computations.

2.7.4 Position and velocity constraints for improved dead reckoning

Position and velocity divergence rapidly occurs as soon as GPS data are not available, as a result of bias drift in IMU sensors. We can impose constraints on velocity in order to ease this problem.

For instance, we may assume lateral and vertical velocities are close to zero in body frame-of-reference . Notice that imposing that lateral velocity is exactly zero is misleading, because motorbikes can sideslip. This occurs even more often with a kart or a motocross. We can deal with this by introducing a high variance for the zero artificial observation of lateral velocity in body frame, so the constraint is relaxed.

2.7.5 Model for Position and Velocity Contraints

Assume the body frame is oriented so that axis x denotes forward direction. Consider the constraint $v_y = 0$, $v_z = 0$. It can be modeled as an artificial observation, and the observation of the corresponding residual is given by

$$\mathbf{z} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - R_b^n(2:3,:)\hat{\mathbf{v}}.$$

Thus we can consider the following model in terms of the error state-space:

$$\delta y = \begin{bmatrix} 0_{2\times 3} & R_b^n(2:3,:) & 0_{2\times 7} \end{bmatrix} \delta x + w$$

where the variance of the artificial measurement noise can be chosen in order to define how hard the constraint on velocity is. For instance, since a motorbike can sideslip, the variance associate to wlat should not be too small. For vehicles with more relevant sidesliding dynamics, it should be fixed to a value that is high enough to adequately relax the constraint on lateral velocity.

Another point to discuss is when to impose the constraints in KF. Basically, we could take them into account every time we update the filter.

2.8 Space-state augmentation

In order to improve performance, we could model bias drift for accelerometers and magnetometer, too. A simple choice can be a random walk process with low variance. Notice that small variations in scale factors and misalignment parameters could be modelled through augmented state-space, too.

2.8.1 Indirect approach implementation

Consider the NED convention for geographic reference frame. Then we can derive the following mechanization equations:

$$\begin{bmatrix}\dot{\phi}\\ \dot{\lambda}\\ \dot{h}\end{bmatrix} = \begin{bmatrix}\frac{1}{R_M+h} & 0 & 0\\ 0 & \frac{1}{(R_N+h)cos(\phi)} & 0\\ 0 & 0 & -1\end{bmatrix}\begin{bmatrix}v_n\\ v_e\\ v_d\end{bmatrix}, \tag{14}$$

$$\begin{bmatrix}\dot{v_n}\\ \dot{v_e}\\ \dot{v_d}\end{bmatrix} = \begin{bmatrix}f_n\\ f_e\\ f_d\end{bmatrix} + g^n - \left[\begin{bmatrix}(\dot{\lambda}+2\omega_{ie})cos(\phi)\\ -\dot{\phi}\\ -(\dot{\lambda}+2\omega_{ie})sin(\phi)\end{bmatrix}\times\right]\begin{bmatrix}v_n\\ v_e\\ v_d\end{bmatrix}, \tag{15}$$

$$\dot{R_n}^n = R_n{}^n(\Omega_{ib}^b - \Omega_{in}^b) \tag{16}$$

Based on initial estimates $\hat{p}(0)$, $\hat{v}(0)$ and $\hat{q}(0)$ it is possible to solve the previous equations on-line.

Then, we introduce GNSS/INS error state space dynamics. Thus, Kalman filtering is used for updating the error state space, that is given by

$$[\delta p,\, \delta v,\, \delta q,\, \delta x_g]^T$$

where

$$\delta p = p - \hat{p}, \quad \delta v = v - \hat{v}, \quad \delta x_g = x_g - \hat{x_g} \tag{17}$$

and δq is such that

$$q = \delta q \otimes \hat{q}$$

As for position error model, let

$$f_p(p,v) := \begin{bmatrix}\dot{\phi}\\ \dot{\lambda}\\ \dot{h}\end{bmatrix} = \begin{bmatrix}\frac{1}{R_M+h} & 0 & 0\\ 0 & \frac{1}{(R_N+h)cos(\phi)} & 0\\ 0 & 0 & -1\end{bmatrix}\begin{bmatrix}v_n\\ v_e\\ v_d\end{bmatrix}$$

Then

$$\dot{p} = f_p(p,v), \quad \dot{\hat{p}} = f_p(\hat{p},\hat{v}).$$

Define

$$\delta p := p - \hat{p}.$$

Thus $\dot{\delta p} := \dot{p} - \dot{\hat{p}}$ and by linearizing $f_p(p,v)$ around $(\hat{p},\hat{v})$ we can write

$$\dot{\delta p} = F_{pp}\delta p + F_{pv}\delta v,$$

where

$$F_{pp} := \begin{bmatrix}0 & 0 & -\frac{v_n}{(R_M+h)^2}\\ \frac{v_e sin(\phi)}{(R_N+h)cos^2\phi} & 0 & -\frac{v_e}{(R_N+h)^2 cos\phi}\\ 0 & 0 & 0\end{bmatrix},$$

$$F_{pv} := \begin{bmatrix}\frac{1}{R_M+h} & 0 & 0\\ 0 & \frac{1}{(R_N+h)cos(\phi)} & 0\\ 0 & 0 & -1\end{bmatrix}$$

Attitude error model is obtained by computing $\dot{\delta q}$ as a function of $\hat{p}, \hat{v}, \hat{q}, \delta p, \delta v, \delta q$ and δx_g. Recall that

$$\dot{\delta q} = \frac{1}{2}\left(\omega_{bn}^b \otimes \delta q - \delta q \otimes \hat{\omega_{bn}^b}\right). \tag{18}$$

As for ω_{bn}^b, we have

$$\omega_{bn}^b = q \otimes \omega_{in}^n \otimes q^* - \omega_{ib}^b, \tag{19}$$

where

$$\omega_{in}^n = \begin{bmatrix} \left(\frac{v_e}{(R_N+h)cos(\phi)} + \omega_{ie}\right) cos(\phi) \\ -\frac{v_n}{R_M+h} \\ -\left(\frac{v_e}{(R_N+h)cos(\phi)} + \omega_{ie}\right) sin(\phi) \end{bmatrix} =: f_\omega(\phi, \lambda, h, v_n, v_e). \tag{20}$$

As for body angular velocity w.r.t. inertial frame, given measured angular velocity ω_g we have

$$\omega_g = \omega_{ib}^b + x_g + \nu_g \tag{21}$$

where x_g is bias drift and ν_g describes measurement noise. It is easy to compute

$$\omega_{ib}^b = \hat{\omega_{ib}^b} - \delta x_g - \nu_g \tag{22}$$

where

$$\hat{\omega_{ib}^b} = \omega_g - \hat{x_g}. \tag{23}$$

Given previous results, we can write

$$\begin{aligned} \dot{\delta q} &= \frac{1}{2}\left\{\left[\delta q \otimes \hat{q} \otimes f_\omega(\phi, h, v_n, v_e) \otimes \hat{q}^* \otimes \delta q^* - \omega_{ib}^b\right] \otimes \delta q \right. \\ &\quad \left. -\delta q \otimes \left[\hat{q} \otimes f_\omega(\hat{\phi}, \hat{\lambda}, \hat{h}, \hat{v_n}, \hat{v_e}) \otimes \hat{q}^* - \hat{\omega_{ib}^b}\right]\right\} \\ &= \frac{1}{2}\left\{\alpha(\delta q, \hat{q}, \phi, h, v_n, v_e, \hat{\phi}, \hat{h}, \hat{v_n}, \hat{v_e}) + \delta q \otimes \hat{\omega_{ib}^b} - \hat{\omega_{ib}^b} \otimes \delta q + [\delta x_g + \nu_g] \otimes \delta q\right\} \\ &= \frac{1}{2}\left\{\alpha(\delta q, \hat{q}, \phi, h, v_n, v_e, \hat{\phi}, \hat{h}, \hat{v_n}, \hat{v_e}) + \begin{bmatrix} 0 & 0 \\ 0 & [-2\hat{\omega_{ib}^b}\times] \end{bmatrix} \delta q + \begin{bmatrix} 0 \\ \delta x_g + \nu_g \end{bmatrix}\right\} \end{aligned} \tag{24}$$

Let us now focus on

$$\begin{aligned} \alpha(\delta q, \hat{q}, \phi, h, v_n, v_e, \hat{\phi}, \hat{h}, \hat{v_n}, \hat{v_e}) := \delta q \otimes \hat{q} \otimes f_\omega(\phi, h, v_n, v_e) \otimes \hat{q}^* \otimes \delta q^* \otimes \delta q \\ - \delta q \otimes \hat{q} \otimes f_\omega(\hat{\phi}, \hat{h}, \hat{v_n}, \hat{v_e}) \otimes \hat{q}^* \end{aligned} \tag{25}$$

Since $q \otimes q^* = q_I$, we can simplify the previous function as follows:

$$\alpha(\delta q, \hat{q}, \phi, h, v_n, v_e, \hat{\phi}, \hat{h}, \hat{v_n}, \hat{v_e}) := \delta q \otimes \hat{q} \otimes \left[f_\omega(\phi, h, v_n, v_e) - f_\omega(\hat{\phi}, \hat{h}, \hat{v_n}, \hat{v_e})\right] \otimes \hat{q}^* \tag{26}$$

We need an expression for $\dot{\delta q}$ in terms of $\delta p, \delta v$ and δq. Consider the first order approximation

$$\begin{aligned} f_\omega(p, v) \approx f_\omega(\hat{p}, \hat{v}) + \frac{\partial f_\omega}{\partial \phi}(\hat{p}, \hat{v})\,\delta\phi + \frac{\partial f_\omega}{\partial \lambda}(\hat{p}, \hat{v})\,\delta\lambda + \frac{\partial f_\omega}{\partial h}(\hat{p}, \hat{v})\,\delta h + \\ \frac{\partial f_\omega}{\partial v_n}(\hat{p}, \hat{v})\,\delta v_n + \frac{\partial f_\omega}{\partial v_e}(\hat{p}, \hat{v})\,\delta v_e + \frac{\partial f_\omega}{\partial v_d}(\hat{p}, \hat{v})\,\delta v_d \end{aligned} \tag{27}$$

Thus, we can write

$$\begin{aligned}\alpha &\approx \delta q \otimes \hat{q} \otimes \begin{bmatrix} 0 \\ -\omega_{ie} sin\hat{\phi}\,\delta\phi - \frac{\hat{v_e}}{(R_N+\hat{h})^2}\,\delta h + \frac{1}{R_n+\hat{h}}\delta v_e \\ \frac{\hat{v_n}}{(R_M+\hat{h})^2}\,\delta h - \frac{1}{R_M+\hat{h}}\,\delta v_n \\ -\left[\frac{\hat{v_e}}{cos^2\hat{\phi}(R_N+\hat{h})} + \omega_{ie}cos\hat{\phi}\right]\delta\phi + \frac{\hat{v_e}}{(R_N+\hat{h})^2} tg\hat{\phi}\,\delta h - tg\hat{\phi}\frac{1}{R_N+\hat{h}}\,\delta v_e \end{bmatrix} \otimes \hat{q}^* \\ &\approx \delta q \otimes \begin{bmatrix} 0 \\ \beta \end{bmatrix} \\ &\approx \begin{bmatrix} 0 & -\beta^T \\ \beta & [-\beta\times] \end{bmatrix} \begin{bmatrix} 1 \\ \delta q_1 \\ \delta q_2 \\ \delta q_3 \end{bmatrix}. \end{aligned} \tag{28}$$

Since we are interested in first order approximation, we disregard higher order terms and consider

$$\alpha(p, v, \delta p, \delta v, \hat{q}, \delta q) \approx \begin{bmatrix} 0 \\ \beta(p, v, \delta p, \delta v, \hat{q}) \end{bmatrix}.$$

The expression for $\beta(p, v, \delta p, \delta v, \hat{q})$ is given below. For the sake of simplicity, it is also computed as

$$\beta(p, v, \delta p, \delta v, \hat{q}) = M(p, v, \hat{q}) \begin{bmatrix} \delta p \\ \delta v \end{bmatrix}.$$

Thus, we end up with

$$\dot{\delta q} = \frac{1}{2}\left\{ \begin{bmatrix} 0 \\ M(p, v, \hat{q}) \end{bmatrix} \begin{bmatrix} \delta p \\ \delta v \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & [-2\hat{\omega_{ib}^b}\times] \end{bmatrix} \delta q + \begin{bmatrix} 0 \\ \delta x_g + \nu_g \end{bmatrix} \right\}$$

Finally, we compute velocity error model. Recall that

$$\dot{v} = R_b^n f^b + g^n - \left[\begin{bmatrix} (\dot{\lambda} + 2\omega_{ie})cos\phi \\ -\dot{\phi} \\ -(\dot{\lambda} + 2\omega_{ie})sin\phi \end{bmatrix} \times \right] \begin{bmatrix} v_n \\ v_e \\ v_d \end{bmatrix},$$

while

$$\dot{\hat{v}} = \hat{R_b}^n \hat{f}^b + \hat{g}^n - \left[\begin{bmatrix} (\dot{\hat{\lambda}} + 2\omega_{ie})cos\hat{\phi} \\ -\dot{\hat{\phi}} \\ -(\dot{\hat{\lambda}} + 2\omega_{ie})sin\hat{\phi} \end{bmatrix} \times \right] \begin{bmatrix} \hat{v_n} \\ \hat{v_e} \\ \hat{v_d} \end{bmatrix},$$

Define $\delta v := v - \hat{v}$. Then

$$\dot{\delta v} = \dot{v} - \dot{\hat{v}} = \alpha + \delta g^n + f_v(p, v) - f_v(\hat{p}, \hat{v})$$

where,

$$\begin{aligned} \alpha :&= q^* \otimes f^b \otimes q - \hat{q}^* \otimes \hat{f}^b \otimes \hat{q} \\ &= \hat{q}^* \otimes \delta q^* \otimes f^b \otimes \delta q \otimes \hat{q} - \hat{q}^* \otimes \hat{f}^b \otimes \hat{q} \\ &= \hat{q}^* \otimes \left(\delta q^* \otimes f^b \otimes \delta q - \hat{f}^b\right) \otimes \hat{q} \\ &= \hat{q}^* \otimes \left[\delta q^* \otimes \left(\hat{f}^b - \delta f\right) \otimes \delta q - \hat{f}^b\right] \otimes \hat{q} \\ &= R_n^q(\hat{q})^T \left[\delta q^* \otimes \left(\hat{f}^b - \delta f\right) \otimes \delta q - \hat{f}^b\right] \end{aligned} \tag{29}$$

and

$$f_v(p,v) = -\begin{bmatrix} 0 & \left(\frac{v_e}{(R_N+h)cos\phi} + 2\omega_{ie}\right) sin\phi & -\frac{v_n}{R_M+h} \\ -\left(\frac{v_e}{(R_N+h)cos\phi} + 2\omega_{ie}\right) sin\phi & 0 & -\left(\frac{v_e}{(R_N+h)cos\phi} + 2\omega_{ie}\right) cos\phi \\ \frac{v_n}{R_M+h} & \left(\frac{v_e}{(R_N+h)cos\phi} + 2\omega_{ie}\right) cos\phi & 0 \end{bmatrix} \begin{bmatrix} v_n \\ v_e \\ v_d \end{bmatrix} \tag{30}$$

In the above equations we also define

$$\delta f := \Delta f - \Delta \hat{f}, \quad \delta g^n := g^n - \hat{g}^n$$

We can compute $\alpha = [\alpha_1\, \alpha_2\, \alpha_3]^T$ and, that is a non-linear function of δq. Then we can

- Use EKF in order to deal with non-linearity (the prediction is updated by means of the non-linear funtion but the variance is propagated according to its Jacobian)
- Consider its linearization.

Now we need an expression for $f_v(p,v) - f_v(\hat{p},\hat{v})$ in terms of δp and δv. We consider the approximation

$$f_v(p,v) - f_v(\hat{p},\hat{v}) \approx \frac{\partial f_v}{\partial p}(\hat{p},\hat{v})\delta p + \frac{\partial f_v}{\partial v}(\hat{p},\hat{v})\delta v.$$

Recall that

$$f_v(\phi,\lambda,h,v_n,v_e,v_d) = \begin{bmatrix} \frac{v_d v_n}{(R_M+h)} + v_e\left(-2\omega_{ie} - \frac{ve}{(R_N+h)cos(\phi)}\right) sin(\phi) \\ v_d\left(2\omega_{ie} + \frac{v_e}{(R_N+h)cos(\phi)}\right) cos(\phi) + v_n\left(2\omega_{ie} + \frac{v_e}{(R_N+h)cos(\phi)}\right) sin(\phi) \\ v_e\left(-2\omega_{ie} - \frac{v_e}{(R_N+h)cos(\phi)}\right) cos(\phi) - \frac{v_n^2}{R_M+h} \end{bmatrix}$$

Its Jacobian F can be computed straightforwardly.

As for the term δg, recall that

$$\delta g^n = -\frac{g}{R}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{bmatrix} \delta p$$

where $g := GM/R^2$. Finally, we can write

$$\dot{\delta v} = \alpha(\delta q, \hat{q}, \delta f, \hat{f}) + F_v(\hat{p},\hat{v}) \begin{bmatrix} \delta p \\ \delta v \end{bmatrix} - \frac{g}{R}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{bmatrix} \delta p$$

Brief note on the implemented roll angle system

The navigation systems in the previous section, works well in tracking the position and velocity of motorbike with sufficiently adequate velocity and GPS sensors (in our formulation we don't account of velocity sensors but they can be easily used in a motorbike). However, the estimate attitude $\boldsymbol{q}(t)$ can be very

distorted due to nuisance factors present in low-cost IMU. More in detail, a fundamental quantity for motorbike applications (at all the levels) is the roll angle. Thus, the second objective of this work is to estimate the roll angle without biases.
The convention used is:

- x-axis follows the motion direction of the motorbike,
- z-axis follows the vertical direction of the motorbike,
- y -axis is the direction such that the coordinate system is right-handed.

The notation used in indicated in Table 1.

Table 1: Notation.

Symbol	**Description**
ϕ	Roll angle
ω_ϕ	Roll rate
ω_x	Angular velocity arounf the x-axis of the motorbike
ω_z	Angular velocity around the z-axis of the motorbike
u	longitudinal speed
I_ω	Overall spin inertia
m	mass
h_1	CoG position
r_t	Radius of the tyre cross section
g	gravity magnitude

In an ideal scenario it may be computed by the general formula

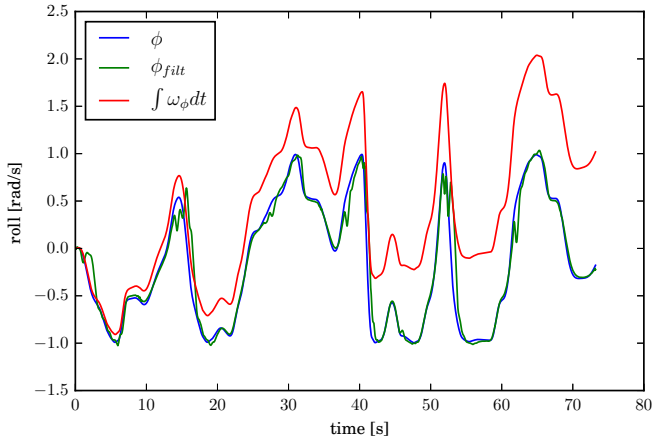
$$\dot{\phi} = \omega_\phi$$

where ω_ϕ is the roll rate. Unfortunately, $\omega_\phi \neq \omega_x$ is depending of the pitch angle μ, which is hard to estimate through MEMS sensors. Moreover, low-cost gyrometers are highly affected by bias and drift, which cause the divergence of ϕ estimate; another nuisance factor consists of gyroscopic effect caused by wheel's rotation. All these problem can be mitigated by exploiting the roll *dynamics*:

$$\begin{cases} \dot{\phi} = \omega_\phi \\ \dot{\omega}_\phi = \dfrac{1}{I_x + mh_1\left(h_1 + r_t cos\phi\right)} \cdot \\ \quad \left[I_w \omega_z \left(\dfrac{\omega_z sin\phi}{cos\phi} - \dfrac{u}{r_r + r_t\left(cos\phi - 1\right)} \right) + m \left(h_1 \left(r_t \omega_\phi^2 sin\phi - \omega_z u + g sin\phi \right) - \dfrac{u r_t \omega_z}{cos\phi} \right) \right. \\ \qquad \left. + \dfrac{m\omega_z^2 h_1 sin\phi \left(h_1 cos\phi + r_t\right)}{cos^2\phi} \right] \\ z = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} \phi \\ \omega_\phi \end{bmatrix} \end{cases} \tag{31}$$

Correlating the roll angle to both the yaw rate and forward speed with the laws of motorcycle dynamics and in particular of the roll motion permits to compensate drift factors. This dynamical model has been implemented with a

Figure 1: Roll angle estimation. The blue line represents the true roll angle. Red line shows what we would obtain a diverging estimate of ϕ due to offset and drift affecting ω_ϕ. The green line is the estimated roll angle throw an EKF modeled using (31).



EKF whose state vector is $\mathbf{x} = \begin{bmatrix}\phi & \omega_\phi\end{bmatrix}^T$ and the observed variable is z, that is ω_ϕ. Obviously the variable measured from the gyrometer is ω_x.

The model (31) has a Kalman representation:

$$\begin{cases} \dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), u(t); \boldsymbol{\theta}) + \boldsymbol{v}(t) \\ y(t) = Hx + w(t) \end{cases} \tag{32}$$

where $\boldsymbol{\theta}$ is the parameter vector, u is the input (the speed), $y(t)$ is the roll angle, $w(t)$ and $\boldsymbol{v}(t)$ are uncorrelated, the observation matrix is $H = \begin{bmatrix}0 & 1\end{bmatrix}$, $Q = E\boldsymbol{v}(t)\boldsymbol{v}^T(t)$ is the covariance of the model noise (which takes into account of unmodeled feature, nuisance factor, etc.) and $R = Ew(t)w^T(t)$ is the covariance of the measurement model.

After discretization (with sampling time ΔT) we obtain the discrete model:

$$\begin{cases} \boldsymbol{x}_k = \boldsymbol{x}_{k-1} + f\left(\boldsymbol{x}_{k-1}, u_k\right)\Delta T + \tilde{\boldsymbol{v}}_k \\ y_k = H\boldsymbol{x}_k + \tilde{w}_k \end{cases} \tag{33}$$

over which we apply the EKF.