Multiplicative EKF

Core Concept: The MEKF is a specialized recursive estimator that is designed to track 3D orientation without the mathematical singularities of Euler angles. Unlike a standard "Additive" EKF, the MEKF maintains a unit-length constraint on its state by splitting orientation into a 4D Nominal Quaternion and a 3D Error-State Vector.

1. The Big Picture - Euler Singularities vs Quaternions:

While the AEKF is function for level flight, it fails in aggressive maneuvers:

- **Gimbal Lock:** In AEKF, terms like $\tan\theta$ and $1/\cos\theta$ in the F_t Jacobian explode as pitch (θ) approaches $\pm 90°$. This singularity causes the filter to crash during fixed-wing loops or multirotor ACRO flight.
- **Quaternions:** We replace Euler angles with a 4D quaternion, given by $q = [q_w, q_x, q_y, q_z]^T$. The extra DoF is constrained by requiring $\|q\| = 1$. Quaternions are numerically stable at all orientations and thus do not suffer from Gimbal Lock.

2. The Variables - Nominal vs Error State:

Almost all variables from normal EKF carry over. To get around the fact that an AEKF cannot add noise to a quaternion since q + noise would no longer be a unit quaternion, we split the belief $\vec{x}_t$ into:

↳ $\bar{q}_t$ (nominal state): A 4D unit quaternion tracking the nominal state orientation estimate. It is updated via nonlinear kinematics but carries no noise/uncertainty.

↳ $\delta\theta_t$ (error state): A 3D vector representing small-angle rotations. This is what interacts with the covariance matrix P and the noise matrices Q, R.

The Relationship: The true orientation is the Hamiltonian product of the nominal state and the error.

$$q_{true} = \bar{q} \otimes \delta q(\delta\theta)$$

(Hamiltonian product of two quaternions represents their angular composition, and $\delta q(\delta\theta)$ is simply quaternion representation of the rotation vector $\delta\theta$).

3. The Recursive MEKF Algorithm:

- Step A: Prediction (The Prior)

i) State Prediction: $\dot{q}_t = \frac{1}{2} q_{t-1} \otimes [0, p, q, r]^T$; $\hat{q}_t = \frac{\dot{q}_t}{\|\dot{q}_t\|}$ ① ← renormalization.

- We integrate raw gyro rates ($u = [p, q, r]^T$) using quaternion kinematics as in equation ①.
- We also renormalise q to 1.0 to maintain a unit vector.

ii) Uncertainty Propagation: $P_t = F_t P_{t-1} F_t^T + Q$ ②

- Exact same as EKF.
- In this case the Jacobian $F_t = I - [w_x]\Delta t$ where $w = [p, q, r]^T$ ← taken wrt. $\delta\theta$ and $[w_x]$ is the skew-symmetric matrix for w (i.e. cross product in matrix form) defined as:

$$[w_x] = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}$$

- If you want to understand why, see proof #1 in appendix.

- Step B: Correction (The Posterior)

i) Kalman Gain: $K_t = P_t H_t^T (H_t P_t H_t^T + R)^{-1}$ ③

- Exact same as EKF.
- For our case of accel/mag $h(\hat{q}_t)$ will be the rotation matrix for the nominal quaternion, $R_{\hat{q}_t}$, multiplied by some interial vector (gravity for accel, b_{earth} for mag). So, can write $\vec{h}(\hat{q}_t) = R_{\hat{q}_t} v_{inertial}$.
- Jacobian $H_t = \frac{\partial h}{\partial(\delta\theta)} = [h(\hat{q}_t)_x]$, for reasoning see proof #2 in appendix.

iii) State Update: $\delta\theta = K_t(z_t - h(\hat{q}_t))$ ④

$\delta\dot{q} = [1, \delta\theta_x/2, \delta\theta_y/2, \delta\theta_z/2]$; $\delta q = \delta\dot{q}/\|\delta\dot{q}\|$ ⑤

$\dot{q}_t = \hat{q}_t \otimes \delta q$, $q_t = \dot{q}_t/\|\dot{q}_t\|$ ⑥

$\delta\theta = [0,0,0]^T$ ⑦

- This looks different but in spirit is the same as EKF.
- In ④ we use the Kalman Gain to produce a 3D error vector $\delta\theta$.
- In ⑤ we convert $\delta\theta$ to a small "delta quaternion" via small angle approximation. We also renormalise.
- In ⑥ we inject δq into q as a correction via Hamiltonian multiplication. Again, we renormalise. Note ④, ⑤, ⑥ are functionally equivalent to $x_t = \hat{x}_t + K_t(z_t - h(\hat{x}_t))$ from EKF!
- In ⑦ we reset error state back to 0 since it has been injected to q.

iii) Uncertainty Update: $P_t = (I - K_t H_t)\hat{P}_t$

- Identical to EKF, no further explanation needed.

4. Implementation Details - 3D Drone AHRS EKF:

- The problem statement is identical to our plain EKF example.
- The MEKF code in mekf.py should be followable if the previous content has been read, especially Section 3 in this note. I highly recommend to try and map all sections of the algorithm described to the code. Some utility functions also exist:
  - `_q_mult`: implement Hamiltonian multiplication for two quaternions.
  - `_get_rot_matrix`: returns 3×3 rotation matrix R_q.
  - `@property self.x`: converts internal 4D quaternion to 3D Euler angles (roll, pitch, yaw) for comparison to AEKF.

5. Looking Ahead:

- This is far from the best drone AHRS EKF! We can also estimate gyro and accel bias (each 3 states) to move to a robust 9-state AHRS EKF.
- Can also construct INS (inertial navigation system) MEKF by using gyro + accel as control input and mag, barometer and GPS for correction!

Appendix

Proof #1: $F_t = I - [\omega_x]\,\Delta t$

To derive F_t, we find the linearized dynamics of the 3D rotation error vector $\delta\theta$. The time derivative of a small-angle rotation error in the body frame is defined by the cross product of the angular velocity ω and the error vector:

$$\dot{\delta\theta} \approx -\omega \times \delta\theta$$

(Note: Negative sign appears as err vector rotates opposite to body's motion since error lives in the body frame and not world frame).

We now represent the cross-product via a skew-symmetric matrix:

$$\dot{\delta\theta} \approx -[\omega_x]\,\delta\theta, \text{ where } [\omega_x] = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}$$

We then take the first order discretization over a small Δt:

$$\delta\theta_t = \delta\theta_{t-1} + \dot{\delta\theta}\,\Delta t \Rightarrow \delta\theta_t = \delta\theta_{t-1} - ([\omega_x]\,\delta\theta_{t-1})\Delta t$$

$$\Rightarrow \delta\theta_t = (I - [\omega_x]\,\Delta t)\,\delta\theta_{t-1}$$

Note the highlighted term is just the local linear transition matrix for the error state $\delta\theta$, such that $\delta\theta_t = F_t\,\delta\theta_{t-1}$.

So, $F_t = I - [\omega_x]\,\Delta t$, QED.

Proof #2: $H_t = [h(\bar{q})_x]$

To derive H_t, we linearize the relationship between the small angle error $\delta\theta$ and the resulting sensor observation $h(q)$. A body-frame observation $h(q_{true})$ is the nominal prediction $h(\bar{q})$ rotated by the error state. We express this as:

$$h(q_{true}) \approx (I - [\delta\theta_x])\,h(\bar{q})$$

Now note that the innovation (y) is given by:

$$\begin{aligned} y = z - h(\bar{q}) &\approx h(q_{true}) - h(\bar{q}) \\ &\approx (I - [\delta\theta_\times])h(\bar{q}) - h(\bar{q}) \\ &\approx (\cancel{I} - [\delta\theta_\times] - \cancel{I})\, h(\bar{q}) \\ \Rightarrow \underline{y} &\approx \underline{-[\delta\theta_\times]\, h(\bar{q})} \end{aligned}$$

Now, we apply a property of cross products: $(a \times b) = (-b \times a)$:

$$\Rightarrow \underline{y \approx [h(\bar{q})_\times]\, \delta\theta}$$

Note $\delta\theta$ is equivalent to $e = x - \hat{x}$ in plain EKF, where we linearized error as $y \approx H_t e$. Thus this equation is the MEKF equivalent!
By the same logic it follows that $H_t = [h(\bar{q})_\times]$, QED.