乘性扩展卡尔曼滤波器

matthewhampsey

18 Jul 2020

MEKF 是 Kalman 滤波器的一个重要改进，使其适用于方位估计。不幸的是，当我试图研究多旋翼姿态估计的主题时，我没有找到一个简单的 (或最近的！) 解释。这就是我试图提供的简单总结。

你可以在此处看到使用 MPU-9250 IMU 在 Beagle Bone Black 上运行的简短片段：

`https://youtu.be/61FvgZV2Mls`

1 设置场景

让我们想象一下，你正在尝试编写一种多旋翼控制系统以使飞机稳定在空中。或者你正在为你选择的 VR 系统编写软件，需要估计手持控制器的姿势。或者你正在编写一个软件，根据检测到的手机方位，在横向和纵向模式之间切换手机的显示。在所有这些情况下，设备将至少有一个机载 IMU(至少有一个陀螺仪和加速度计，还可以选择一个磁力计)，你必须使用来自 IMU 的测量值来精确估计 3D 空间中的设备方位。不幸的是，陀螺仪和加速计将是大多数消费级硬件的 MEMS 设备，这意味着它们将相当嘈杂。幸运的是，这些设备提供了冗余信息，我们可以使用这些信息来显著提高任何单个设备的航位推算测量。要做到这一点，我们需要一些 NASA 的工程师在 60 年代提出的一些想法。

你可能还猜到了 Kalman 滤波器会在这里有所帮助。让我们考虑一下这种方法。在本文的其余部分，我们将使用四元数的 Hamiltonian 约定，并使用 R_b^i 表示从机体坐标系变换为世界坐标系的旋转，因此 $R_b^i v^b = v^i$。R_i^b 将表示世界坐标系到机体坐标系的变换，因此 $R_i^b = R_b^{iT}$。我们需要方位，所以我们的状态向量 $\boldsymbol{x}$ 将至少部分地由方位参数化形成。对于这个思想实验，我们假设我们使用四元数参数化，并且我们的状态向量只由参数化 $\boldsymbol{x} = q$(为了简单起见) 形成。让我们假设我们已经完成了我们的过程和观察模型 (或一些等效的线性化)，并遵循了常用的卡尔曼滤波更新步骤来进行**后验**状态校正：$\boldsymbol{x_{k|k}} = \boldsymbol{x_{k|k-1}} + K_k \boldsymbol{y_k}$。$\boldsymbol{x_{k|k}}$ 是什么东西？好吧，$\boldsymbol{x}$ 是单位四元数，我们在它上面**加上**了非零值，因此 $\boldsymbol{x_k} + K_k \boldsymbol{y_k}$ **不可能**是单位数。如果它不是单位数，那就**不可能**是旋转。Kalman 打破了我们需要的旋转的一个不变性质。

发生了什么？好吧，我们已经成为经典错误之一的受害者：永远不要假设你的操作数是向量！旋转四元数**不是**向量。欧拉角也不是 (无论你想怎么排列它们。不，实际上，它们**不是**向量)，旋转矩阵也不是。他们是**群**。并且群不是有效的 Kalman 滤波器状态，因为它们仅在各自的群运算下封闭，而在向量加法下不封闭。

“我们不要把孩子和洗澡水一起扔出去。”你争辩说：“我们可以从中恢复过来。只需重新计算总和即可。”这**可以**工作 (例如，参见 Markeley 的 Multiplicative vs. Additive Filtering for Spacecraft Attitude Determination 论文)。但是它抛弃了我们的**后验**误差校正，并且误差协方差

$P_{k|k} = cov(\boldsymbol{x} - \hat{\boldsymbol{x}_{\boldsymbol{k}|\boldsymbol{k}}})$ 现在**意味**着什么？我们可以进行数学计算，但是它变得相当可怕，只是有一些难看的东西堵住了这些漏洞。让我们尝试另一种方法。

幸运的是，NASA 在 60 年代提出了 MEKF，从而解决了这一问题。诀窍是我们要假设我们的方向估计不再被状态向量中的一个条目捕获。相反，完整的方位表示 q(我们仍将使用四元数) 由两个参数捕获：累积的方位估计 $\hat{q}$ 和单独的小角度误差向量 $\boldsymbol{\alpha}$，用于参数化误差四元数，$\delta q(\boldsymbol{\alpha}) = \begin{pmatrix} 1 \\ \frac{\boldsymbol{\alpha}}{2} \end{pmatrix}$，因此我们的完整估计是 $q = \hat{q}\delta q(\boldsymbol{\alpha})$。请注意，此误差运算与朴素的误差校正相比有所不同：因为我们已经按照群运算公式化了误差，所以我们保持了此旋转表示的不变性。

每次滤波器更新，全状态估计 $\hat{q}$ 首先根据测得的角速度 $\hat{\boldsymbol{\omega}}$ 进行更新。然后，根据标准的 Kalman 滤波实现来更新小角度误差 (现在可以正常工作，因为 $\boldsymbol{\alpha}$ 保证**很小**，因此我们不会遇到任何奇异问题)。最后，我们用误差更新了全状态估计 $q_k = q_{k-1}\delta q_k$，并将四元数小角度误差重置为单位数：$\boldsymbol{\alpha_{k+1}} = \mathbf{0}$ 和 $\delta q_{k+1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$。

2 乘性扩展卡尔曼滤波器

因此，完整的步骤 (记住状态向量最初值是 $\mathbf{0}$) 是：

首先，使用测得的角速度更新方位估计值 (这是 MEKF 独有的[1])：

$$q_{k|k-1} = q_{k-1|k-1} + \dot{q}_{k-1|k-1}\Delta t = q_{k-1|k-1} + \frac{1}{2} q_{k-1|k-1} \begin{pmatrix} 0 \\ \boldsymbol{\omega} \end{pmatrix} \Delta t$$

然后，更新过程模型：

$$\Phi_k = (I + F\Delta t)$$

其中 $\dot{\boldsymbol{x}} = F\boldsymbol{x}$

然后，更新**先验**估计协方差和 kalman 增益 (以下与普通 KF 相同)：

$$P_{k|k-1} = \Phi_k P_{k-1|k-1} \Phi_k^T + Q_k$$

$$K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1}$$

接下来，更新**后验**状态估计和协方差：

$$\boldsymbol{x_{k|k}} = K_k(\boldsymbol{z_k} - R_i^b(q_{k|k-1})\boldsymbol{f}^g)$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1}$$

最后，更新**后验**方位估计 (同样，MEKF 独有的[2])：

$$q_{k|k} = q_{k|k-1}\delta q(x_{k|k})$$

[1]译注：这是旋转的运动学方程，基本上每个用角速度估计姿态的 KF 都有，非 MEKF 独有。

[2]译注：MEKF 要求显式重置，而其它 KF 是隐式要求。

3 陀螺仪和加速度计的具体例子

让我们通过一个具体的实现，我们有一个单独的观测器，一个加速度计测量。

我们假设陀螺仪和加速度计的标准高斯噪声漂移模型，因此测量的角速度是真实角速度 $\boldsymbol{\omega}$，偏差 $\boldsymbol{\beta_\omega}$ 和白噪音 $\boldsymbol{\eta_\omega}$ 之和：$\hat{\boldsymbol{\omega}} = \boldsymbol{\omega} + \boldsymbol{\beta_\omega} + \boldsymbol{\eta_\omega}$。偏差漂移根据 $\dot{\boldsymbol{\beta_\omega}} = \boldsymbol{\nu_\beta}$，其中 $\boldsymbol{\eta}$ 和 $\boldsymbol{\nu}$ 是白噪声随机过程。

类似地，加速计测量 $\hat{\boldsymbol{f}} = \boldsymbol{f} + \boldsymbol{\beta_f} + \boldsymbol{\eta_f}$，带有漂移偏差 β_f 和白噪声 η_f。真正的加速度计测量值将是来自机体坐标系惯性力 $\boldsymbol{f}^b$ 的加速度和重力加速度的总和：$\boldsymbol{f}^b = \boldsymbol{a}^b + R(q)_i^b \begin{pmatrix} 0 \\ 0 \\ -g \end{pmatrix}^i$

让我们形成我们的状态向量。我们需要在世界坐标系中的方位误差参数化 ($\boldsymbol{\alpha}$)、位置误差 ($\delta\boldsymbol{r}$) 和线速度误差 ($\delta\boldsymbol{v}$)，以及陀螺仪和加速度计偏差 ($\boldsymbol{\beta_\omega}$ 和 $\boldsymbol{\beta_f}$)：

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{\alpha} \\ \delta\boldsymbol{v} \\ \delta\boldsymbol{r} \\ \boldsymbol{\beta_\omega} \\ \boldsymbol{\beta_f} \end{pmatrix}$$

为了形成 Kalman 滤波器，我们需要形成我们的离散状态转移模型，该模型由误差状态向量的动力学以明显的形式确定：$\Phi = I + F\Delta t$，其中 $\dot{\boldsymbol{x}} = F\boldsymbol{x}$。

参见附录对于以下推导。

$$\dot{\boldsymbol{\alpha}} = -[\hat{\boldsymbol{\omega}}\times]\boldsymbol{\alpha} - \boldsymbol{\beta_\omega} - \boldsymbol{\eta_\omega}$$

$$\delta\dot{\boldsymbol{v}} = -R_b^i(\hat{q})[\hat{\boldsymbol{f}}^b\times]\boldsymbol{\alpha} - R_b^i(\hat{q})\boldsymbol{\beta_f} - R_b^i(\hat{q})\boldsymbol{\eta_f}$$

$$\delta\dot{\boldsymbol{r}} = \delta\boldsymbol{v}$$

$$\dot{\boldsymbol{\beta_\omega}} = \boldsymbol{\nu_\omega}$$

$$\dot{\boldsymbol{\beta_f}} = \boldsymbol{\nu_f}$$

所以，

$$F = \begin{pmatrix} -[\hat{\boldsymbol{\omega}}\times] & 0 & 0 & -I & 0 \\ -R_b^i(\hat{q})[\hat{\boldsymbol{f}}^b\times] & 0 & 0 & 0 & -R_b^i(\hat{q}) \\ 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

并且

$$Q_k = \int_0^{\Delta t} e^{F(\Delta t-\tau)} Q_c e^{F^T(\Delta t-\tau)} d\tau$$

其中

$$Q_c = \begin{pmatrix} diag(\sigma_\omega^2) & 0 & 0 & 0 & 0 \\ 0 & diag(\sigma_f^2) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & diag(\sigma_{\beta_\omega}^2) & 0 \\ 0 & 0 & 0 & 0 & diag(\sigma_{\beta_f}^2) \end{pmatrix}$$

我们可以在 $\hat{\omega} = \hat{f} = 0$ 和 $R_i^b(\hat{q}) = I$ 周围线性化 F，以获得：

$$Q_k = \begin{pmatrix} \wedge(\sigma_\omega^2)\Delta t + \wedge(\sigma_{\beta_\omega}^2)\frac{\Delta t^3}{3} & 0 & 0 & -\wedge(\sigma_{\beta_\omega}^2)\frac{\Delta t^2}{2} & 0 \\ 0 & \wedge(\sigma_f^2)\Delta t + \wedge(\sigma_{\beta_f}^2)\frac{\Delta t^3}{3} & \wedge(\sigma_{\beta_f}^2)\frac{\Delta t^4}{8} + \wedge(\sigma_f^2)\frac{\Delta t^2}{2} & 0 & -\wedge(\sigma_{\beta_f}^2)\frac{\Delta t^2}{2} \\ 0 & \wedge(\sigma_f^2)\frac{\Delta t^2}{2} + \wedge(\sigma_{\beta_f}^2)\frac{\Delta t^4}{8} & \wedge(\sigma_f^2)\frac{\Delta t^3}{3} + \wedge(\sigma_{\beta_f}^2)\frac{\Delta t^5}{20} & 0 & -\wedge(\sigma_{\beta_f}^2)\frac{\Delta t^3}{6} \\ -\wedge(\sigma_{\beta_\omega}^2)\frac{\Delta t^2}{2} & 0 & 0 & \wedge(\sigma_{\beta_\omega}^2)\Delta t & 0 \\ 0 & -\wedge(\sigma_{\beta_f}^2)\frac{\Delta t^2}{2} & -\wedge(\sigma_{\beta_f}^2)\frac{\Delta t^3}{6} & 0 & \wedge(\sigma_{\beta_f}^2)\Delta t \end{pmatrix}$$

其中 $\wedge(\boldsymbol{x}) = \begin{pmatrix} x_0 & 0 & 0 \\ 0 & x_1 & 0 \\ 0 & 0 & x_2 \end{pmatrix}$。

对于加速度计观测，

$$\delta \boldsymbol{f}^b = -\boldsymbol{\alpha} \times R_i^b(\hat{q}) \begin{pmatrix} 0 \\ 0 \\ -g \end{pmatrix} + \boldsymbol{\beta_f} + \boldsymbol{\eta_f}$$

所以

$$H = \begin{pmatrix} \left[R_i^b(\hat{q}) \begin{pmatrix} 0 \\ 0 \\ -g \end{pmatrix} \times \right] & 0 & 0 & 0 & I \end{pmatrix}$$

4 代码

下面是等效的 python 代码 (你可以在此处找到完整的代码)。你会注意到上面没有提到的一些细节，例如从测量值中减去估计的偏差：

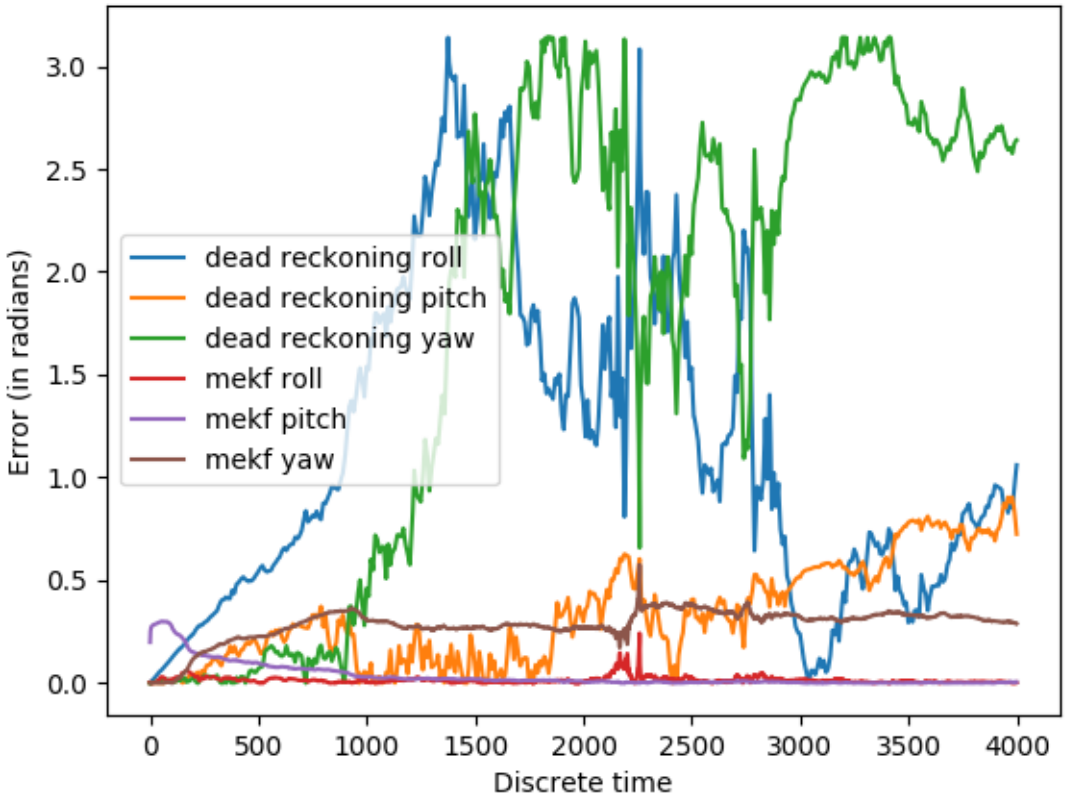
```
class Kalman:

    #state vector:
    # [0:3] orientation error
    # [3:6] velocity error
    # [6:9] position error
    # [9:12] gyro bias
    # [12:15] accelerometer bias
    def __init__(self, initial_est, estimate_covariance,
                       gyro_cov, gyro_bias_cov, accel_proc_cov,
                       accel_bias_cov, accel_obs_cov):
        self.estimate = initial_est
        self.estimate_covariance = estimate_covariance*np.identity(15, dtype=float)

        self.observation_covariance = accel_obs_cov*np.identity(3, dtype=float)
        self.gyro_bias = np.array([0.0, 0.0, 0.0])
        self.accelerometer_bias = np.array([0.0, 0.0, 0.0])

        self.G = np.zeros(shape=(15, 15), dtype=float)
        self.G[0:3, 9:12] = -np.identity(3)
        self.G[6:9, 3:6] =  np.identity(3)

        self.gyro_cov_mat = gyro_cov*np.identity(3, dtype=float)
        self.gyro_bias_cov_mat = gyro_bias_cov*np.identity(3, dtype=float)
```

```
        self.accel_cov_mat = accel_proc_cov*np.identity(3, dtype=float)
        self.accel_bias_cov_mat = accel_bias_cov*np.identity(3, dtype=float)

    def process_covariance(self, time_delta):
        Q = np.zeros(shape=(15, 15), dtype=float)
        Q[0:3, 0:3] = self.gyro_cov_mat*time_delta + self.gyro_bias_cov_mat*(time_delta**3)/3.0
        Q[0:3, 9:12] = -self.gyro_bias_cov_mat*(time_delta**2)/2.0
        Q[3:6, 3:6] = self.accel_cov_mat*time_delta + self.accel_bias_cov_mat*(time_delta**3)/3.0
        Q[3:6, 6:9] = self.accel_bias_cov_mat*(time_delta**4)/8.0 + self.accel_cov_mat*(time_delta
            **2)/2.0
        Q[3:6, 12:15] = -self.accel_bias_cov_mat*(time_delta**2)/2.0
        Q[6:9, 3:6] = self.accel_cov_mat*(time_delta**2)/2.0 + self.accel_bias_cov_mat*(time_delta
            **4)/8.0
        Q[6:9, 6:9] = self.accel_cov_mat*(time_delta**3)/3.0 + self.accel_bias_cov_mat*(time_delta
            **5)/20.0
        Q[6:9, 12:15] = -self.accel_bias_cov_mat*(time_delta**3)/6.0
        Q[9:12, 0:3] = -self.gyro_bias_cov_mat*(time_delta**2)/2.0
        Q[9:12, 9:12] = self.gyro_bias_cov_mat*time_delta
        Q[12:15, 3:6] = -self.accel_bias_cov_mat*(time_delta**2)/2.0
        Q[12:15, 6:9] = -self.accel_bias_cov_mat*(time_delta**3)/6.0
        Q[12:15, 12:15] = self.accel_bias_cov_mat*time_delta

        return Q

    def update(self, gyro_meas, acc_meas, time_delta):

        gyro_meas = gyro_meas - self.gyro_bias
        acc_meas = acc_meas - self.accelerometer_bias
        #Integrate angular velocity through forming quaternion derivative
        self.estimate = self.estimate + time_delta*0.5*self.estimate*Quaternion(scalar = 0, vector=
            gyro_meas)
        self.estimate = self.estimate.normalised

        #Form process model
        self.G[0:3, 0:3] = -skewSymmetric(gyro_meas)
        self.G[3:6, 0:3] = -quatToMatrix(self.estimate).dot(skewSymmetric(acc_meas))
        self.G[3:6, 12:15] = -quatToMatrix(self.estimate)
        F = np.identity(15, dtype=float) + self.G*time_delta

        #Update with a priori covariance
        self.estimate_covariance = np.dot(np.dot(F, self.estimate_covariance), F.transpose()) + self.
            process_covariance(time_delta)

        #Form Kalman gain
        H = np.zeros(shape=(3,15), dtype=float)
        H[0:3, 0:3] = skewSymmetric(self.estimate.inverse.rotate(np.array([0.0, 0.0, -1.0])))
        H[0:3, 12:15] = np.identity(3, dtype=float)
        PH_T = np.dot(self.estimate_covariance, H.transpose())
        inn_cov = H.dot(PH_T) + self.observation_covariance
        K = np.dot(PH_T, np.linalg.inv(inn_cov))

        #Update with a posteriori covariance
        self.estimate_covariance = (np.identity(15) - np.dot(K, H)).dot(self.estimate_covariance)

        aposteriori_state = np.dot(K, (acc_meas - self.estimate.inverse.rotate(np.array([0.0, 0.0,
            -1.0]))))
```

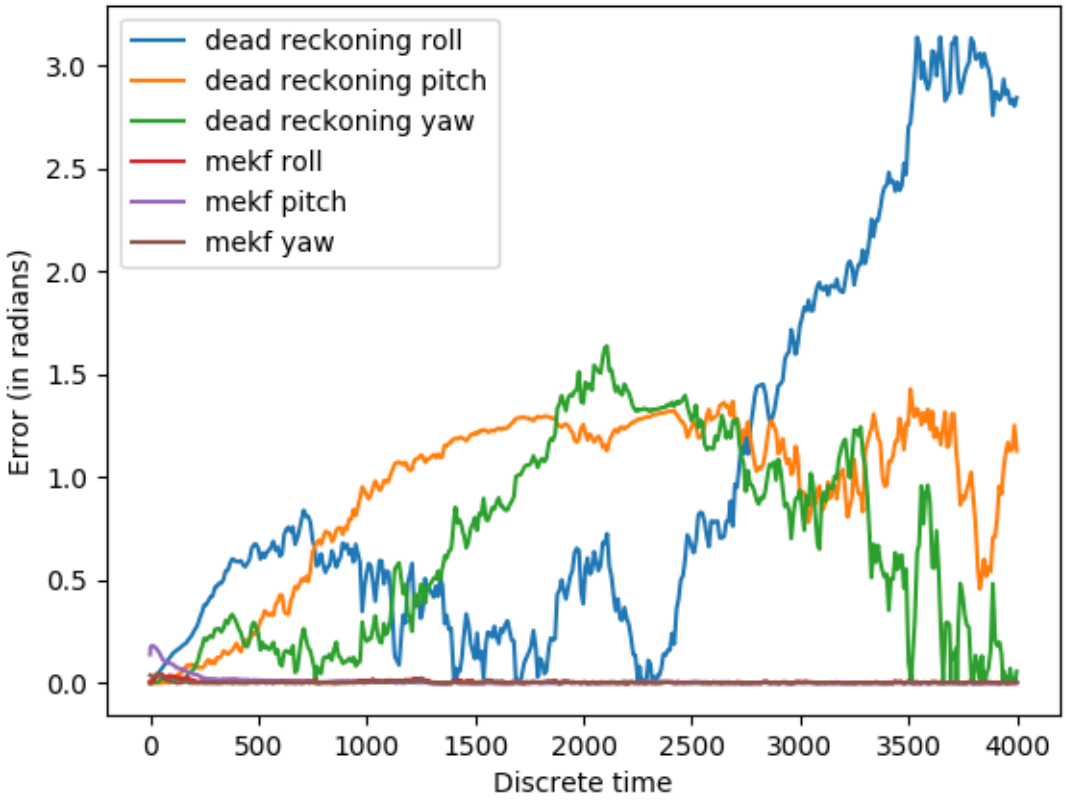
```
#Fold filtered error state back into full state estimates
self.estimate = self.estimate * Quaternion(scalar = 1, vector = 0.5*aposteriori_state[0:3])
self.estimate = self.estimate.normalised
self.gyro_bias += aposteriori_state[9:12]
self.accelerometer_bias += aposteriori_state[12:15]
```

5 结果

效果如何？让我们模拟一个物体，它从静止开始，没有旋转，然后根据某种随机角速度开始旋转。这样，对象将随机移动，但是轨迹平滑。我们在模拟的陀螺和加速度计测量中加入了一些偏差和高斯噪声。以下是与陀螺仪测量的原始航位推算积分相比的结果：



roll 和 pitch 可以精确预测，但不能预测 yaw。这是预期的：加速度计测量重力，重力与初始方位的 yaw 轴平行。我们可以通过增加一个额外的参考测量值来修正这个问题，比如从磁力计上。我们用和加速度计完全相同的方式来做这个，只是现在假设参考方位是 $\begin{pmatrix}1\\0\\0\end{pmatrix}$。方位并不重要，只要它与机体坐标系 z 轴线性无关即可。



这样更好。

6 附录

以下推导来自 James M.Maley 的优秀论文《Multiplicative Quaternion Extended Kalman Filtering for Nonspinning Guided Projectiles》，我对过程协方差矩阵的推导进行了一些更正。

证明 $\dot{\boldsymbol{\alpha}} = -[\hat{\boldsymbol{\omega}}\times]\boldsymbol{\alpha} - \boldsymbol{\beta}_{\boldsymbol{\omega}} - \boldsymbol{\eta}_{\boldsymbol{\omega}}$：
$\delta\dot{q} = \hat{q}^{-1}\dot{q} + \dot{\hat{q}}^{-1}q$
$\hat{q}^{-1}\hat{q} = 1$，所以 $\frac{d(\hat{q}^{-1}\hat{q})}{dt} = 0$。
另外，根据乘积规则：$\frac{d(\hat{q}^{-1}\hat{q})}{dt} = \dot{\hat{q}}^{-1}\hat{q} + \hat{q}^{-1}\dot{\hat{q}}$，
所以 $\dot{\hat{q}}^{-1} = -\hat{q}^{-1}\dot{\hat{q}}\hat{q}^{-1} = -\hat{q}^{-1}\frac{1}{2}\hat{q}\begin{pmatrix}0\\ \hat{\boldsymbol{\omega}}^b\end{pmatrix}\hat{q}^{-1} = -\frac{1}{2}\begin{pmatrix}0\\ \hat{\boldsymbol{\omega}}^b\end{pmatrix}\hat{q}^{-1}$。
使用此定义于 $\delta\dot{q} = \hat{q}^{-1}\dot{q} + \dot{\hat{q}}^{-1}q$ 中有：

$$\delta\dot{q} = \hat{q}^{-1}\frac{1}{2}q\begin{pmatrix}0\\ \boldsymbol{\omega}^b\end{pmatrix} - \frac{1}{2}\begin{pmatrix}0\\ \hat{\boldsymbol{\omega}}^b\end{pmatrix}\hat{q}^{-1}q$$

$$\delta\dot{q} = \frac{1}{2}\delta q\begin{pmatrix}0\\ \boldsymbol{\omega}^b\end{pmatrix} - \frac{1}{2}\begin{pmatrix}0\\ \hat{\boldsymbol{\omega}}^b\end{pmatrix}\delta q$$

$$\delta\dot{q} = \frac{1}{2}\delta q \begin{pmatrix} 0 \\ \hat{\boldsymbol{\omega}}^b \end{pmatrix} - \frac{1}{2}\begin{pmatrix} 0 \\ \hat{\boldsymbol{\omega}}^b \end{pmatrix}\delta q + \frac{1}{2}\delta q \begin{pmatrix} 0 \\ \delta\boldsymbol{\omega} \end{pmatrix}$$

使用四元数乘法的定义和 $\delta q_r \approx 1$：

$$\delta\dot{q} = \frac{1}{2}\begin{pmatrix} -\delta\boldsymbol{q_v}\cdot\hat{\boldsymbol{\omega}} \\ \hat{\boldsymbol{\omega}}^b + \delta\boldsymbol{q_v}\times\hat{\boldsymbol{\omega}} \end{pmatrix} + \frac{1}{2}\begin{pmatrix} \hat{\boldsymbol{\omega}}\cdot\delta\boldsymbol{q_v} \\ -\hat{\boldsymbol{\omega}}^b - \hat{\boldsymbol{\omega}}\times\delta\boldsymbol{q_v} \end{pmatrix} + \frac{1}{2}\begin{pmatrix} -\delta\boldsymbol{q_v}\cdot\delta\hat{\boldsymbol{\omega}} \\ \delta\hat{\boldsymbol{\omega}} + \delta\boldsymbol{q_v}\times\delta\hat{\boldsymbol{\omega}} \end{pmatrix}$$

简化此四元数的向量分量，并将二阶误差分量设置为 0，我们获得[3]：

$$\delta\dot{\boldsymbol{q}}_{\boldsymbol{v}} = -\hat{\boldsymbol{\omega}}\times\delta\boldsymbol{q_v} + \frac{1}{2}\delta\boldsymbol{\omega}$$

并使用我们定义的 $\boldsymbol{\alpha}$，$\boldsymbol{\alpha} = 2\delta\boldsymbol{q_v}$，则变为：

$$\dot{\boldsymbol{\alpha}} = -[\hat{\boldsymbol{\omega}}\times]\boldsymbol{\alpha} + \delta\boldsymbol{\omega}$$

这就是我们想要的结果。

证明 $\delta\boldsymbol{f}^b = -\boldsymbol{\alpha}\times R_i^b(\hat{q})\boldsymbol{m}^i + \boldsymbol{\beta_f} + \boldsymbol{\eta_f}$：
测得的参考将是参考向量，真实方位以及测量偏差和噪声的函数：

$$\boldsymbol{m} = R_i^b(q)\boldsymbol{m}^i + \boldsymbol{\beta_m} + \boldsymbol{\eta_m}$$

估计的观测值是参考向量和当前四元数估计值的函数：

$$\hat{\boldsymbol{m}} = R_i^b(\hat{q})\boldsymbol{m}^i$$

我们具有以下等价关系 (参见 Rodrigues 的旋转公式)：

$$R_i^b(q) = 2\boldsymbol{q_v}\boldsymbol{q_v}^T + I_{3\times3}(q_r^2 - \boldsymbol{q_v}^T\boldsymbol{q_v}) - 2q_r[\boldsymbol{q_v}\times]$$

所以 (忽略二阶误差项)：

$$R_i^b(\delta q) = I - [\boldsymbol{\alpha}\times]$$

并且

$$R_i^b(q) = R_i^b(\delta q)R_i^b(\hat{q}) = (I - [\boldsymbol{\alpha}\times])R_i^b(\hat{q})$$

所以

$$\delta\boldsymbol{m}^b = \boldsymbol{m}^b - \hat{\boldsymbol{m}}^b = (I - [\boldsymbol{\alpha}\times])R_i^b(\hat{q})\boldsymbol{m}^i + \boldsymbol{\beta_m} + \boldsymbol{\eta_m} - R_i^b(\hat{q})\boldsymbol{m}^i$$

所以

$$\delta\boldsymbol{m}^b = -[\boldsymbol{\alpha}\times]R_i^b(\hat{q})\boldsymbol{m}^i + \boldsymbol{\beta_m} + \boldsymbol{\eta_m}$$

这是必需的结果。

[3]译注：原文方程格式有点瑕疵。方程左边少了一个导数符号 $\dot{\square}$。

先验协方差传播的证明

从前面开始，我们有：

$$\dot{\boldsymbol{\alpha}} = -[\hat{\boldsymbol{\omega}}\times]\boldsymbol{\alpha} - \boldsymbol{\beta}_{\boldsymbol{\omega}} - \boldsymbol{\eta}_{\boldsymbol{\omega}}$$

$$\delta\dot{\boldsymbol{v}} = -R_b^i(\hat{q})[\hat{\boldsymbol{f}}^b\times]\boldsymbol{\alpha} - R_b^i(\hat{q})\boldsymbol{\beta}_{\boldsymbol{f}} - R_b^i(\hat{q})\boldsymbol{\eta}_{\boldsymbol{f}}$$

$$\delta\dot{\boldsymbol{r}} = \delta\boldsymbol{v}$$

$$\dot{\boldsymbol{\beta}}_{\boldsymbol{\omega}} = \boldsymbol{\nu}_{\boldsymbol{\omega}}$$

$$\dot{\boldsymbol{\beta}}_{\boldsymbol{f}} = \boldsymbol{\nu}_{\boldsymbol{f}}$$

使用之前的定义 $\boldsymbol{x}$ 和 F，我们可以将其重申为：

$$\dot{\boldsymbol{x}} = F\boldsymbol{x} + G\boldsymbol{w}$$

其中

$$G = \begin{pmatrix} diag(-\sigma_\omega) & 0 & 0 & 0 & 0 \\ 0 & -R_b^i(\hat{q})diag(\sigma_f) & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & diag(\sigma_{\beta_\omega}) & 0 \\ 0 & 0 & 0 & 0 & diag(\sigma_{\beta_f})) \end{pmatrix}$$

并且 $\boldsymbol{w}$ 为单位方差，零均值白噪声向量。

经检查，$G\boldsymbol{w}$ 的互相关 (方差) 为

$$E[(G\boldsymbol{w}(t))(G\boldsymbol{w}(\tau))^T] = Q_c\delta(t-\tau)$$

$\dot{\boldsymbol{x}} = F\boldsymbol{x} + G\boldsymbol{w}$ 的解是众所周知的结果：

$$\boldsymbol{x}(t) = e^{F(t-t_0)}\boldsymbol{x}(t_0) + \int_{t_0}^{t} e^{F(t-\tau)}G\boldsymbol{w}(\tau)d\tau$$

我们感兴趣的是 $t = t_k$ 和 $t_0 = t_{k-1}$，因此 $t_k - t_{k-1} = \Delta t$，并在 $\tau\text{'} = \tau - t_{k-1}$ 的积分中进行替换

$$\boldsymbol{x}(t_k) = e^{F\Delta t}\boldsymbol{x}(t_{k-1}) + \int_0^{\Delta t} e^{F(\Delta t-\tau')}G\boldsymbol{w}(\tau' + t_{k-1})d\tau'$$

$\boldsymbol{x}(t_k)$ 的协方差由以下公式给出：

$$\begin{aligned} P_{k|k-1} &= E[\boldsymbol{x}(t_k)\boldsymbol{x}(t_k)^T] \\ &= E[\Big(e^{F\Delta t}\boldsymbol{x}(t_{k-1}) + \textstyle\int_0^{\Delta t} e^{F(\Delta t-\tau')}G\boldsymbol{w}(\tau' + t_{k-1})d\tau'\Big)\Big(e^{F\Delta t}\boldsymbol{x}(t_{k-1}) + \textstyle\int_0^{\Delta t} e^{F(\Delta t-\tau'')}G\boldsymbol{w}(\tau'' + t_{k-1})d\tau''\Big)^T] \end{aligned}$$

在此基础上，我们删除了 $\boldsymbol{x}$ 和 $\boldsymbol{w}$ 的乘积，因为它们是不相关的，并且都有 0 的期望值：

$$\begin{aligned} E[\boldsymbol{x}(t_k)\boldsymbol{x}(t_k)^T] =& e^{F\Delta t}E[\boldsymbol{x}(t_{k-1})\boldsymbol{x}(t_{k-1})^T]e^{F^T\Delta t} \\ &+ \int_0^{\Delta t}\int_0^{\Delta t} e^{F(\Delta t-\tau')}E[G\boldsymbol{w}(\tau' + t_{k-1})\boldsymbol{w}(\tau'' + t_{k-1})^T G^T]e^{F^T(\Delta t-\tau'')}d\tau' d\tau'' \end{aligned}$$

$$E[\boldsymbol{x}(t_k)\boldsymbol{x}(t_k)^T] = e^{F\Delta t}P_{(k-1|k-1)}e^{F^T\Delta t} + \int_0^{\Delta t}\int_0^{\Delta t} e^{F(\Delta t-\tau')}Q_c\delta(\tau' - \tau'')e^{F^T(\Delta t-\tau'')}d\tau' d\tau''$$

$$E[\boldsymbol{x}(t_k)\boldsymbol{x}(t_k)^T] = \Phi P_{(k-1|k-1)}\Phi^T + \int_0^{\Delta t} e^{F(\Delta t-\tau'')}Q_c e^{F^T(\Delta t-\tau'')}d\tau''$$

因此，$Q_k = \int_0^{\Delta t} e^{F(\Delta t-\tau)}Q_c e^{F^T(\Delta t-\tau)}d\tau$，这是所需的结果。
我们用 $I + F(\Delta t-\tau) + \frac{1}{2}F^2(\Delta t-\tau)^2$ 近似 $e^{F(\Delta t-\tau)}$，并在 $\hat{\omega} = \hat{f} = 0$ 和 $R_i^b(\hat{q}) = I$ 周围线性化 F，以便

$$F \approx \begin{pmatrix} 0 & 0 & 0 & -I & 0 & 0 \\ 0 & 0 & 0 & 0 & -I & 0 \\ 0 & I & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

并且

$$I + F(\Delta t-\tau) + \frac{1}{2}F^2(\Delta t-\tau)^2 = \begin{pmatrix} I & 0 & 0 & -I(\Delta t-\tau) & 0 & 0 \\ 0 & I & 0 & 0 & -I(\Delta t-\tau) & 0 \\ 0 & I(\Delta t-\tau) & I & 0 & -\frac{I}{2}(\Delta t-\tau)^2 & 0 \\ 0 & 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & 0 & I \end{pmatrix}$$

在 $e^{F(\Delta t-\tau)}Q_c e^{F^T(\Delta t-\tau)}$ 中使用这个近似值可以得到：

$$Q_k = \int_0^{\Delta t} \begin{pmatrix} \wedge(\sigma_\omega^2) + \wedge(\sigma_{\beta_\omega}^2)(\Delta t-\tau)^2 & 0 & 0 & -\wedge(\sigma \\ 0 & \wedge(\sigma_f^2) + \wedge(\sigma_{\beta_f}^2)(\Delta t-\tau)^2 & \wedge(\sigma_{\beta_f}^2)\frac{(\Delta t-\tau)^3}{2} + \wedge(\sigma_f^2)(\Delta t-\tau) & \\ 0 & \wedge(\sigma_f^2)(\Delta t-\tau) + \wedge(\sigma_{\beta_f}^2)\frac{(\Delta t-\tau)^3}{2} & \wedge(\sigma_f^2)(\Delta t-\tau)^2 + \wedge(\sigma_{\beta_f}^2)\frac{(\Delta t-\tau)^4}{4} & \\ -\wedge(\sigma_{\beta_\omega}^2)(\Delta t-\tau) & 0 & 0 & \wedge \\ 0 & -\wedge(\sigma_{\beta_f}^2)(\Delta t-\tau) & -\wedge(\sigma_{\beta_f}^2)\frac{(\Delta t-\tau)^2}{2} & \end{pmatrix}$$

计算这个积分可以得到 Q_k 过程噪声近似的期望结果。

磁力计观测的扩展方程
我们扩展状态向量以包括磁力计偏差：

$$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{\alpha} \\ \delta\boldsymbol{v} \\ \delta\boldsymbol{r} \\ \boldsymbol{\beta_\omega} \\ \boldsymbol{\beta_f} \\ \boldsymbol{\beta_m} \end{pmatrix}$$

其中 $\dot{\boldsymbol{\beta_m}} = \boldsymbol{\eta_m}$。

然后 F 变成

$$F = \begin{pmatrix} -[\hat{\boldsymbol{\omega}}\times] & 0 & 0 & -I & 0 & 0 \\ -R_b^i(\hat{q})[\hat{\boldsymbol{f}}^b\times] & 0 & 0 & 0 & -R_b^i(\hat{q}) & 0 \\ 0 & I & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

然后 Q_c 和 Q_k 变成

$$Q_c = \begin{pmatrix} diag(\sigma_\omega^2) & 0 & 0 & 0 & 0 & 0 \\ 0 & diag(\sigma_f^2) & 0 &) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & diag(\sigma_{\beta_\omega}^2) & 0 & 0 \\ 0 & 0 & 0 & 0 & diag(\sigma_{\beta_f}^2) & 0 \\ 0 & 0 & 0 & 0 & 0 & diag(\sigma_{\beta_m}^2) \end{pmatrix}$$

$$Q_k = \begin{pmatrix} \wedge(\sigma_\omega^2)\Delta t + \wedge(\sigma_{\beta_\omega}^2)\frac{\Delta t^3}{3} & 0 & 0 & -\wedge(\sigma_{\beta_\omega}^2)\frac{\Delta t^2}{2} & 0 \\ 0 & \wedge(\sigma_f^2)\Delta t + \wedge(\sigma_{\beta_f}^2)\frac{\Delta t^3}{3} & \wedge(\sigma_{\beta_f}^2)\frac{\Delta t^4}{8} + \wedge(\sigma_f^2)\frac{\Delta t^2}{2} & 0 & -\wedge(\sigma_{\beta_f}^2)\frac{\Delta t^2}{2} \\ 0 & \wedge(\sigma_f^2)\frac{\Delta t^2}{2} + \wedge(\sigma_{\beta_f}^2)\frac{\Delta t^4}{8} & \wedge(\sigma_f^2)\frac{\Delta t^3}{3} + \wedge(\sigma_{\beta_f}^2)\frac{\Delta t^5}{20} & 0 & -\wedge(\sigma_{\beta_f}^2)\frac{\Delta t^3}{6} \\ -\wedge(\sigma_{\beta_\omega}^2)\frac{\Delta t^2}{2} & 0 & 0 & \wedge(\sigma_{\beta_\omega}^2)\Delta t & 0 \\ 0 & -\wedge(\sigma_{\beta_f}^2)\frac{\Delta t^2}{2} & -\wedge(\sigma_{\beta_f}^2)\frac{\Delta t^3}{6} & 0 & \wedge(\sigma_{\beta_f}^2)\Delta t \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \wedge(\sigma$$

同样，观察矩阵 H 变成

$$H = \begin{pmatrix} \begin{bmatrix} R_i^b(\hat{q}) \begin{pmatrix} 0 \\ 0 \\ -g \end{pmatrix} \times \end{bmatrix} & 0 & 0 & 0 & I & 0 \\ \begin{bmatrix} R_i^b(\hat{q}) \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \times \end{bmatrix} & 0 & 0 & 0 & 0 & I \end{pmatrix}$$

---

email github.com/matthewhampsey