轮 趣 科 技

卡 尔 曼 滤 波 器 浅 析

推荐关注我们的公众号获取更新资料

版本说明：

版本	日期	内容说明
V1.0	2023/09/20	第一次发布

网址：www.wheeltec.net

目 录

1. 简介

1960 年，鲁道夫·埃米尔·卡尔曼(R·E·Kalman)在一篇论文中介绍了一种应用于离散线性滤波的迭代算法，这就是后来得到广泛应用的卡尔曼滤波[1]。而后卡尔曼在 NASA 埃姆斯研究中心访问时，发现他的方法对于解决阿波罗计划的轨道预测很有用，于是在阿波罗飞船的导航电脑中使用了这种滤波器。在测量值混有噪声而且系统的行为也可能受到随机干扰的情况下，卡尔曼滤波能够从含有噪声的测量值中得到系统状态的最优估计。

2. 一个耳熟能详的例子

下面借鉴论文《Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation》里的例子来对卡尔曼滤波器有一个初步的认识。

考虑一维轨道上的一辆小车（如图 2-1 所示），它的状态可以由位置和速度两个变量来刻画，即状态向量$X_t = [x_t \quad \dot{x}_t]^{\mathrm{T}}$。现有装置可以测量小车当前的位置$x_t$以及可以控制油门$u_t$的大小（$u_t$表示系统的输入，即加速度$\ddot{x}_t$），现在的问题是如何估计小车当前的位置？（这里考虑最简单的情况，即位置，而不是小车的状态）

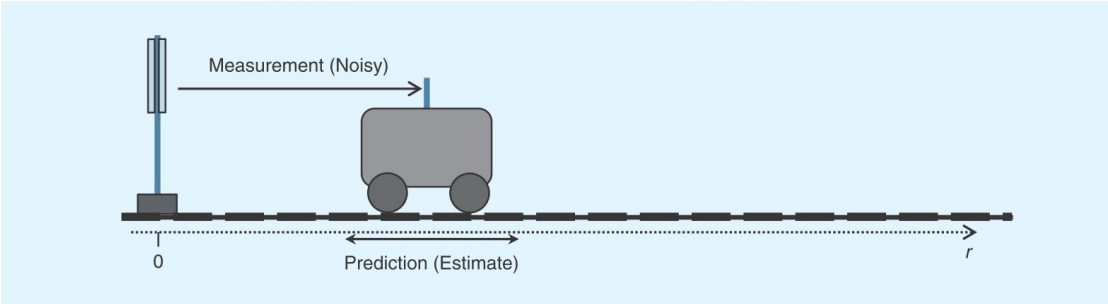

图 2-1 一维轨道上的小车[2]

首先，我们可以很容易的得到理想情况下该系统的数学模型，其中，系统（离散系统）的状态方程为

$$\begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ \dot{x}_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{(\Delta t)^2}{2} \\ \Delta t \end{bmatrix} u_{t-1} \tag{1}$$

输出方程为

$$y_{t-1} = [1 \quad 0] \begin{bmatrix} x_{t-1} \\ \dot{x}_{t-1} \end{bmatrix} \tag{2}$$

这样，我们只需知道初始状态X_0，就可以求得任意时刻小车的状态。然而，由于各种不确定因素（道路不平整，刮风下雨等天气因素，与滑轨的摩擦阻力等）导致这个数学模型不是完美的，存在过程噪声。你也许会说还可以用测量装置进行测量，很可惜，即便再精确的仪器都存在测量误差或者说测量噪声（在生活中，我们测量一个物理量，往往不会是一个值，而是一个区间）。既然无论数学模型还是观测都是不准确的，那么是否可以对两者进行一个数据融合得出一个相对准确的估计？答案是肯定的，接下来看看怎么操作：

(1) 假设过程噪声和测量噪声都服从高斯分布（中心极限定理告诉我们高斯分布是日常生活中最普遍存在的一种的概率分布），那么初始时刻(t=0)小车位置的分布由图 2-2 所示。

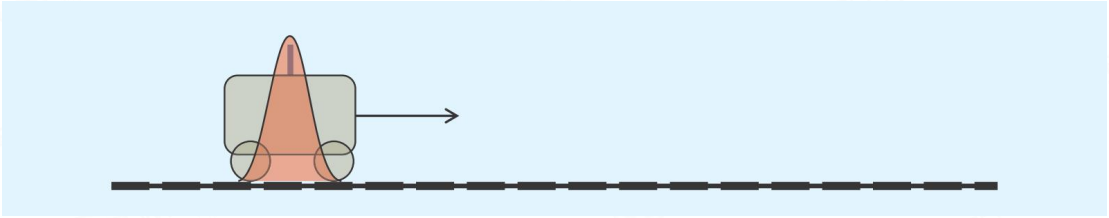

图 2-2 小车初始时刻的位置(t=0)

(2) 接着根据式(1)可以预测出下一时刻小车的位置，如图 2-3 所示。可以看到，这个时候的方差相比初始时刻变大了，这是因为各种不确定因素导致下一时刻的不确定性增大了。

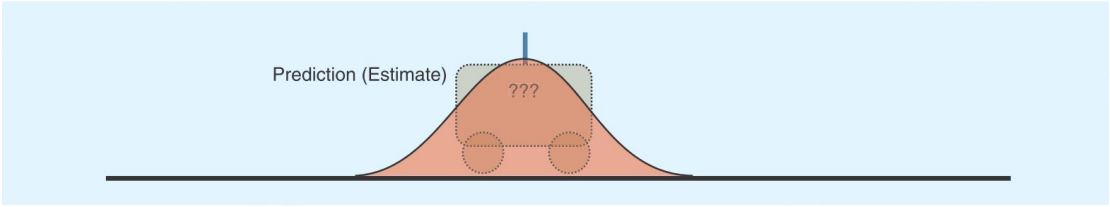

图 2-3 小车下一时刻位置的预测量(t=1)

(3) 为了减少纯估计带来的偏差，我们需要通过测量来“验证”这个估计，测量结果如图 2-4 所示。

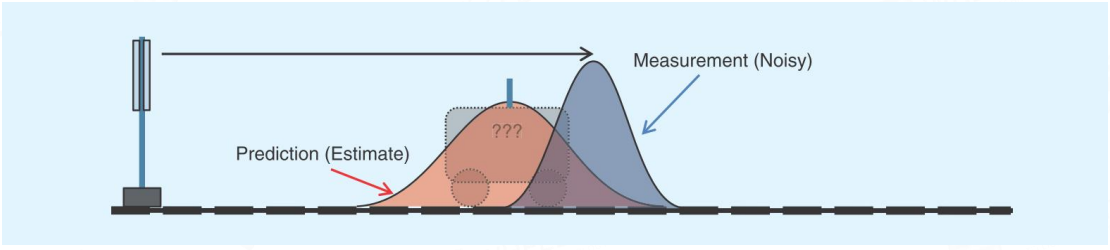

图 2-4 小车下一时刻位置的观测量(t=1)

(4) 然后通过将两个概率密度函数相乘得到一个新的概率密度函数，而这个新的分布不仅依然是高斯分布，而且还包含了系统模型和测量的信息（从图中可以看出方差减小了，说明不确定性也减小了）。

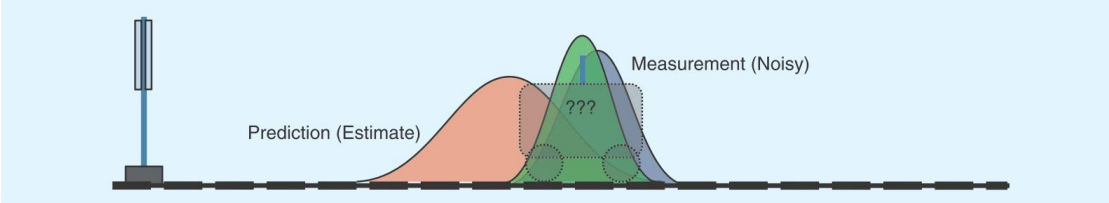

图 2-5 小车下一时刻位置的估计量(t=1)

(5) 最后，将 t=1 作为初始时刻，重复(2)~(4)的过程。

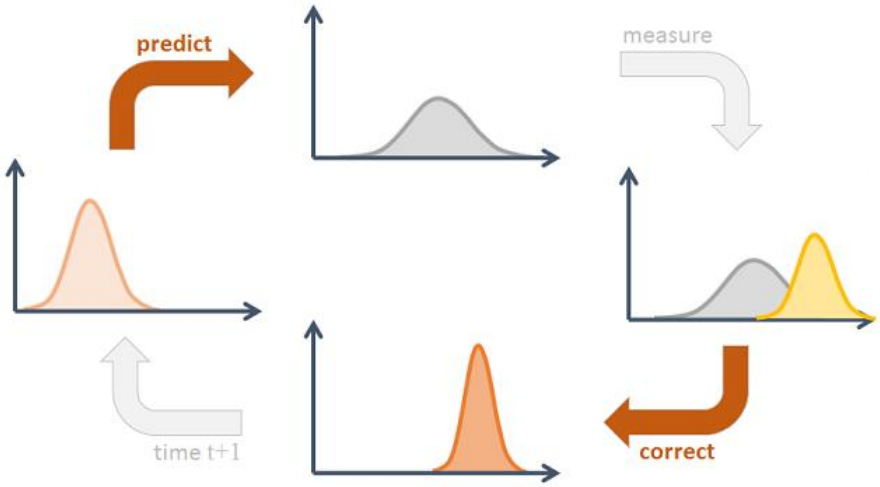

图 2-6 卡尔曼滤波过程图解

以上基本体现了卡尔曼滤波的全部过程，但是，还有一个疑惑，为什么可以将两个概率密度函数相乘的结果作为对位置的估计？

在回答这个问题之前，先对贝叶斯公式进行一个简要说明。贝叶斯公式（又称为贝叶斯定理）是概率统计中应用所观察到的现象对有关概率分布的主观判断（先验概率）进行修正的标准方法。离散随机变量的贝叶斯公式为

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \tag{3}$$

在贝叶斯公式中，每个名词都有约定俗成的名称：

(1) $P(B)$称为先验概率。

(2) $P(A)$称为边缘概率，也叫作标准化常量。

(3) $P(A|B)$称为似然概率。是在事件B发生的条件下事件A发生的概率。

(4) $P(B|A)$称为后验概率。是在事件A发生的条件下事件B发生的概率。

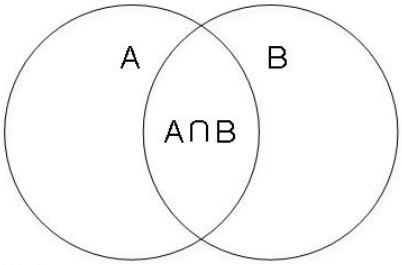

图 2-7 事件A与事件B的关系

这个公式表明在事件A（结果）已经发生的条件下由事件B（起因）引起的概率（也可以理解为通过事件A来“校正”事件B发生的概率。原本事件B发生的概率是$P(B)$，当事件A发生了以后就变成了$P(B|A)$，也就是说减少了不确定性）如何计算（从几何上直观的理解就是在事件A这个新的样本空间中，事件AB所占的比重）。

当离散随机变量拓展到连续随机变量时，有

$$f_{X|Y}(x|y)=\frac{f_{Y|X}(y|x)f_X(x)}{f_Y(y)} \tag{4}$$

类似的，它们也都有约定俗成的名称：

(1) $f_X(x)$称为先验分布。

(2) $f_Y(y)$称为边缘分布，也叫作标准化常量。

(3) $f_{Y|X}(y|x)$称为似然分布。

(4) $f_{X|Y}(x|y)$称为后验分布。

值得一提的是，在实际应用中往往估计的随机变量是X，也就是说分母$f_Y(y)$是一个与X无关的常数，其作用相当于使得$f_{X|Y}(x|y)$满足概率分布的基本要求（在整个实数域内积分等于 1）。所以，式 (4)也常写成

$$f_{X|Y}(x|y)=\eta f_{Y|X}(y|x)f_X(x) \tag{5}$$

这个式子表明，可以通过后验分布来“校正”先验分布。

现在来回答上述的问题。图 2-3 所示的分布实际上是一个条件分布，它是在已知系统上一个时刻的状态x_{t-1}和输入u_{t-1}的条件下产生的，即可以写成$f(x_t|x_{t-1},u_{t-1})$。图 2-4 所示的分布也是一个条件分布，它是在小车处于状态x_t的情况下通过测量得到的，即可以写成$f(y_t|x_t)$。现在我们假设测量y_t仅仅与当前状态x_t有关，那么分布$f(y_t|x_t)$可以改写成$f(y_t|x_t,x_{t-1},u_{t-1})$。毫无疑问，图

2-5 所示的分布也是一个条件分布，它是在已知系统上一个时刻的状态x_{t-1}，输入u_{t-1}以及这一时刻的测量y_t的条件下得到的，即可以写成$f(x_t|y_t,x_{t-1},u_{t-1})$。若令$m=(x_t|x_{t-1},u_{t-1}),n=y_t$，那么由式 (5)可知$f(m|n)=\eta f(n|m)f(m)$，即（这里只是提供一个简单的看法，更严谨的推导可以参考网络上相关的文章）

$$f(x_t|y_t,x_{t-1},u_{t-1})=\eta f(y_t|x_t,x_{t-1},u_{t-1})f(x_t|x_{t-1},u_{t-1}) \tag{6}$$

所以，通过数学模型预测的小车下一时刻位置的分布实际上是一个先验分布，而通过测量观测到的小车位置的分布是一个似然分布，将两者相乘这一操作实际是利用贝叶斯公式得到了一个后验分布，通过后验分布来“校正”先验分布。（之所以$\eta=1$是因为两个高斯分布相乘还是一个高斯分布，不需要系数来调节）

在回答了上述的问题之后，你可能会衍生出一个新的疑问：为什么采用贝叶斯公式？这个估计是最优的吗？我们知道所谓的“最优”是有条件的，在不同的条件下得到的最优解也不一样，而在我们的问题中，最常用的一种评价标准就是最小均方差(MMSE)，它指的是使得$\mathrm{E}[\hat{x}(y)-x]^2$最小的估计量$\hat{x}$是最优估计。你可以这样理解，已知有一系列的输入$y$和对应的输出$x$，你希望有这么一个估计器$\phi(y)$（也可以写成$\hat{x}(y)$）使得当有新的输入的时候能够估计输出。怎么样的估计器是最优的呢？在最小均方差意义下，使得$\mathrm{E}[\phi(y)-x]^2$最小的估计器是最优的。

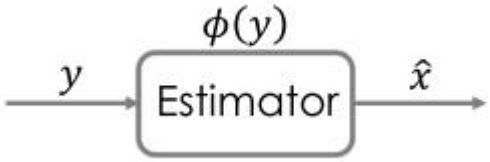

图 2-8 估计器

接下来，不加证明的给出一个定理：假设$(X,Y)\sim f_{XY}(X,Y)$，在已知$Y=y$的条件下，x的最小均方差估计$\hat{x}=\mathrm{E}(X|Y=y)$。对于高斯分布而言，知道了分布就相当于就知道了期望，所以也就表明了式(6)计算得到的后验分布在最小均方差意义下是最优的。

3. 离散卡尔曼滤波公式的推导

在这一章节，基本上就是公式和定理的堆砌，如果对公式推导不感兴趣的读者可以跳过这一部分。

在进行公式的推导之前，我先罗列一下可能会用到的定理：

定理一：

假设$(X,Y)\sim f_{XY}(X,Y)$，在已知$Y=y$的条件下，x的最小均方差估计$\hat{x}=\mathrm{E}(X|Y=y)$。

定理二：

两个互相独立的高斯分布的联合分布也服从高斯分布，称为联合高斯分布。

定理三：

若$X\sim\mathcal{N}(\mu,\Sigma)$，则$AX+b\sim\mathcal{N}(A\mu+b,A\Sigma A^{\mathrm{T}})$。也就是说服从高斯分布的随机向量的线性组合也服从高斯分布。

定理四：

若$\begin{bmatrix}X\\Y\end{bmatrix}\sim\mathcal{N}\left(\begin{bmatrix}\mu_X\\\mu_Y\end{bmatrix},\begin{bmatrix}\Sigma_X & \Sigma_{XY}\\\Sigma_{YX} & \Sigma_Y\end{bmatrix}\right)$，则$(X|Y=y)\sim\mathcal{N}\left(\mu_{X|Y=y},\Sigma_{X|Y=y}\right)$。其中，$\mu_{X|Y=y}=\mu_X+\Sigma_{XY}\Sigma_Y^{-1}(y-\mu_Y),\Sigma_{X|Y=y}=\Sigma_X-\Sigma_{XY}\Sigma_Y^{-1}\Sigma_{YX}$。

下面对离散卡尔曼滤波公式进行推导：

设离散线性随机系统的状态空间表达式为（假设系数矩阵和控制率已知）

$$\begin{cases}x_k=Ax_{k-1}+Bu_{k-1}+w_{k-1}\\y_{k-1}=Cx_{k-1}+v_{k-1}\end{cases}\tag{7}$$

其中，$x_{k-1}\in\mathbb{R}^n$，$u_{k-1}\in\mathbb{R}^p$，$w_{k-1}\in\mathbb{R}^n$，$y_{k-1}\in\mathbb{R}^m$，$v_{k-1}\in\mathbb{R}^m$。我们假设初始状态$x_0\sim\mathcal{N}(\mu_0,\sigma_0)$，过程噪声$w_{k-1}\sim\mathcal{N}(0,Q_{k-1})$，测量噪声$v_{k-1}\sim\mathcal{N}(0,R_{k-1})$，并且$w_k$，$v_k$与任意时刻系统的状态$x_k$以及系统的输出$y_k$无关（注意：$x_k$，$y_k$，$w_k$，$v_k$都是向量）。记$Y_k=[y_k\quad y_{k-1}\quad\cdots\quad y_0]^{\mathrm{T}}$。

由假设易知(x_0,w_0,v_0)服从联合高斯分布（定理二），即$\begin{bmatrix}x_0\\w_0\\v_0\end{bmatrix}\sim\mathcal{N}\left(\begin{bmatrix}\mu_0\\\mathbf{0}\\\mathbf{0}\end{bmatrix},\begin{bmatrix}\sigma_0 & \mathbf{0} & \mathbf{0}\\\mathbf{0} & Q_0 & \mathbf{0}\\\mathbf{0} & \mathbf{0} & R_0\end{bmatrix}\right)$。因为$\begin{bmatrix}x_1\\Y_0\end{bmatrix}=\begin{bmatrix}x_1\\y_0\end{bmatrix}=\begin{bmatrix}A & I & \mathbf{0}\\C & \mathbf{0} & \mathbf{0}\end{bmatrix}\begin{bmatrix}x_0\\w_0\\v_0\end{bmatrix}+\begin{bmatrix}B\\\mathbf{0}\end{bmatrix}u_0$（$u_0$是已知的，相当于常数），所以，$(x_1,Y_0)$服从联合高斯分布（定理三）。又因为$\begin{bmatrix}x_{k+1}\\Y_k\end{bmatrix}=\begin{bmatrix}x_{k+1}\\y_k\\Y_{k-1}\end{bmatrix}=\begin{bmatrix}A & \mathbf{0} & I & \mathbf{0}\\C & \mathbf{0} & \mathbf{0} & I\\\mathbf{0} & I & \mathbf{0} & \mathbf{0}\end{bmatrix}\begin{bmatrix}x_k\\Y_{k-1}\\w_k\\v_k\end{bmatrix}+\begin{bmatrix}B\\\mathbf{0}\\\mathbf{0}\end{bmatrix}u_k$。所以，只要$(x_k,Y_{k-1},w_k,v_k)$服从联合高斯分布，则$(x_{k+1},Y_k)$服从高斯分布。由于$w_k$，$v_k$与任意时刻系统的状态$x_k$以及系统的输出$y_k$无关，因此，要求$(x_k,Y_{k-1},w_k,v_k)$服从联合高斯分布等价于要求$(x_k,Y_{k-1})$服从联合高斯分布（这里相当于产生了一个递推关系，即如果(x_k,Y_{k-1})服从联合高斯分布，那么(x_{k+1},Y_k)也服从高斯分布），而(x_1,Y_0)服从

联合高斯分布，故有$\forall k,(x_k,Y_{k-1})$服从高斯分布。最后，由$\begin{bmatrix}x_k\\Y_k\end{bmatrix}=\begin{bmatrix}x_k\\y_k\\Y_{k-1}\end{bmatrix}=$

$\begin{bmatrix}I&\mathbf{0}&\mathbf{0}&\mathbf{0}\\C&\mathbf{0}&\mathbf{0}&I\\\mathbf{0}&I&\mathbf{0}&\mathbf{0}\end{bmatrix}\begin{bmatrix}x_k\\Y_{k-1}\\w_k\\v_k\end{bmatrix}$可知$\forall k,(x_k,Y_k)$服从联合高斯分布（定理三）。现在，我们产生了一个新的定理，如下所示：

定理五：

基于上述假设可以得知(x_k,Y_k)服从联合高斯分布。

现在，我们要做的事情是在已知$Y_k=\begin{bmatrix}y_0\\y_1\\\vdots\\y_k\end{bmatrix}$的条件下，求解$x_k$的最优估计$\widehat{x_k}^+$（在最小均方差意义下）。

(1) 先验估计公式：

首先，我们在仅有Y_{k-1}的条件下，由定理一可知x_k的最优估计$\widehat{x_k}^-=\mathrm{E}(x_k|Y_{k-1})$（注：$\widehat{x_k}^-$表示先验估计，$\widehat{x_k}^+$表示后验估计）。下面来计算这个条件期望：

$$\mathrm{E}(x_k|Y_{k-1})$$
$$=\mathrm{E}(Ax_{k-1}+Bu_{k-1}+w_{k-1}|Y_{k-1})$$
$$=\mathrm{E}(Ax_{k-1}|Y_{k-1})+\mathrm{E}(Bu_{k-1}|Y_{k-1})+\mathrm{E}(w_{k-1}|Y_{k-1})$$
$$=A\mathrm{E}(x_{k-1}|Y_{k-1})+Bu_{k-1}$$
$$=A\widehat{x_{k-1}}^+ +Bu_{k-1}$$

即

$$\widehat{x_k}^-=A\widehat{x_{k-1}}^+ +Bu_{k-1}$$

(2) 先验误差协方差公式：

由于x_k服从高斯分布，而描述高斯分布需要两个变量，期望向量和协方差矩阵，因此，我们进一步的求解先验误差协方差$\widehat{p_k}^-$。

$$\widehat{p_k}^-$$
$$=\mathrm{Cov}(x_k,x_k|Y_{k-1})$$
$$=\mathrm{E}\left((x_k-\widehat{x_k}^-)(x_k-\widehat{x_k}^-)^{\mathrm{T}}\middle|Y_{k-1}\right)$$
$$=\mathrm{E}\left((Ax_{k-1}+Bu_{k-1}+w_{k-1}-\widehat{x_k}^-)(Ax_{k-1}+Bu_{k-1}+w_{k-1}-\widehat{x_k}^-)^{\mathrm{T}}\middle|Y_{k-1}\right)$$

$$= \mathrm{E}\left((A(x_{k-1} - \widehat{x_{k-1}}) + w_{k-1})(A(x_{k-1} - \widehat{x_{k-1}}) + w_{k-1})^{\mathrm{T}} \middle| Y_{k-1}\right)$$

$$= A\mathrm{E}\left((x_{k-1} - \widehat{x_{k-1}})(x_{k-1} - \widehat{x_{k-1}})^{\mathrm{T}} \middle| Y_{k-1}\right)A^{\mathrm{T}} + \mathrm{E}(A(x_{k-1} - \widehat{x_{k-1}})w_{k-1}{}^{\mathrm{T}} | Y_{k-1})$$

$$+ \mathrm{E}(w_{k-1}(x_{k-1} - \widehat{x_{k-1}})^{\mathrm{T}}A^{\mathrm{T}} | Y_{k-1}) + \mathrm{E}(w_{k-1}w_{k-1}{}^{\mathrm{T}} | Y_{k-1})$$

$$= A\mathrm{E}\left((x_{k-1} - \widehat{x_{k-1}})(x_{k-1} - \widehat{x_{k-1}})^{\mathrm{T}} \middle| Y_{k-1}\right)A^{\mathrm{T}} + \mathrm{E}(w_{k-1}w_{k-1}{}^{\mathrm{T}})$$

$$= A\widehat{p_{k-1}}^{+}A^{\mathrm{T}} + Q_{k-1}$$

即

$$\widehat{p_k}^{-} = A\widehat{p_{k-1}}^{+}A^{\mathrm{T}} + Q_{k-1}$$

(3) 后验估计公式：

接下来，我们计算后验估计，即在获得新的测量信息y_k的时候，求解x_k的最优估计$\widehat{x_k}^{+}$。此时，x_k的最优估计为$\widehat{x_k}^{+} = \mathrm{E}(x_k|Y_k)$。下面来计算这个条件期望。由定理五可知$\begin{bmatrix} x_k \\ Y_k \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \\ Y_{k-1} \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} x_k \\ y_k \end{bmatrix} \\ Y_{k-1} \end{bmatrix}$服从联合高斯分布，而

$$\left(\begin{bmatrix} x_k \\ y_k \end{bmatrix} \middle| Y_{k-1}\right) \sim \mathcal{N}\left(\begin{bmatrix} E(x_k|Y_{k-1}) \\ E(y_k|Y_{k-1}) \end{bmatrix}, \begin{bmatrix} Cov(x_k, x_k|Y_{k-1}) & Cov(x_k, y_k|Y_{k-1}) \\ Cov(y_k, x_k|Y_{k-1}) & Cov(y_k, y_k|Y_{k-1}) \end{bmatrix}\right)。$$

又因为$\left(\begin{bmatrix} x_k \\ y_k \end{bmatrix} \middle| Y_{k-1}\right)$可以写成$\begin{pmatrix} x_k|Y_{k-1} \\ y_k|Y_{k-1} \end{pmatrix}$，则由定理四的条件期望计算公式可知

$$\mathrm{E}\left((x_k|y_k)|Y_{k-1}\right)$$

$$= \mathrm{E}\left(x_k \middle| \begin{bmatrix} Y_{k-1} \\ y_k \end{bmatrix}\right)$$

$$= \mathrm{E}(x_k|Y_k)$$

$$= \mathrm{E}(x_k|Y_{k-1}) + \mathrm{Cov}(x_k, y_k|Y_{k-1})\mathrm{Cov}^{-1}(y_k, y_k|Y_{k-1})\left(y_k - \mathrm{E}(y_k|Y_{k-1})\right)$$

其中

1) $\mathrm{E}(x_k|Y_{k-1}) = \widehat{x_k}^{-}$

2) $\mathrm{Cov}(x_k, y_k|Y_{k-1})$

$$= \mathrm{Cov}(x_k, Cx_k + v_k|Y_{k-1})$$

$$= \mathrm{Cov}(x_k, x_k|Y_{k-1})C^{\mathrm{T}} + \mathrm{Cov}(x_k, v_k|Y_{k-1})$$

$$= \widehat{p_k}^{-}C^{\mathrm{T}}$$

3) $\mathrm{Cov}^{-1}(y_k, y_k|Y_{k-1})$

$$= \mathrm{Cov}^{-1}(Cx_k + v_k, Cx_k + v_k|Y_{k-1})$$

$$= \left(C\mathrm{Cov}(x_k, x_k|Y_{k-1})C^{\mathrm{T}} + \mathrm{Cov}(v_k, v_k|Y_{k-1})\right)^{-1}$$

$$= \left(C\widehat{p_k}^{-}C^{\mathrm{T}} + \mathrm{Cov}(v_k, v_k|Y_{k-1})\right)^{-1}$$

$$= \left(C\widehat{p_k}^{-}C^{\mathrm{T}} + \mathrm{E}(v_k v_k{}^{\mathrm{T}}|Y_{k-1}) - \left(\mathrm{E}(v_k|Y_{k-1})\right)^2\right)^{-1}$$

$$= (C\widehat{p_k}^{-}C^{\mathrm{T}} + R_k)^{-1}$$

4) $\mathrm{E}(y_k|Y_{k-1}) = \mathrm{E}(Cx_k + v_k|Y_{k-1}) = C\mathrm{E}(x_k|Y_{k-1}) + \mathrm{E}(v_k|Y_{k-1}) = C\widehat{x_k}^{-}$

故

$$\widehat{x_k}^{+} = \widehat{x_k}^{-} + K_k(y_k - C\widehat{x_k}^{-})$$

(4) 卡尔曼增益公式：

其中

$$K_k = \widehat{p_k}^{-}C^{\mathrm{T}}(C\widehat{p_k}^{-}C^{\mathrm{T}} + R_k)^{-1}$$

(5) 更新误差协方差公式：

类似的，由定理四的条件协方差计算公式可知

$$\widehat{p_k}^{+} = \mathrm{Cov}(x_k, x_k|Y_{k-1}) - \mathrm{Cov}(x_k, y_k|Y_{k-1})\mathrm{Cov}^{-1}(y_k, y_k|Y_{k-1})\mathrm{Cov}(y_k, x_k|Y_{k-1})$$

其中

1) $\mathrm{Cov}(x_k, x_k|Y_{k-1}) = \widehat{p_k}^{-}$

2) $\mathrm{Cov}(x_k, y_k|Y_{k-1})\mathrm{Cov}^{-1}(y_k, y_k|Y_{k-1}) = K_k$

3) $\mathrm{Cov}(y_k, x_k|Y_{k-1})$

$$= \mathrm{Cov}(Cx_k + v_k, x_k|Y_{k-1})$$

$$= C\mathrm{Cov}(x_k, x_k|Y_{k-1}) + \mathrm{Cov}(v_k, x_k|Y_{k-1})$$

$$= C\widehat{p_k}^{-}$$

故

$$\widehat{p_k}^{+} = (I - K_k C)\widehat{p_k}^{-}$$

总结：

1) 预测部分：

先验估计公式：$\widehat{x_k}^{-} = A\widehat{x_{k-1}}^{+} + Bu_{k-1}$

先验误差协方差公式：$\widehat{p_k}^{-} = A\widehat{p_{k-1}}^{+}A^{\mathrm{T}} + Q_{k-1}$

2) 校正部分：

卡尔曼增益公式：$K_k = \widehat{p_k}^{-}C^{\mathrm{T}}(C\widehat{p_k}^{-}C^{\mathrm{T}} + R_k)^{-1}$

后验估计公式：$\widehat{x_k}^+ = \widehat{x_k}^- + K_k(y_k - C\widehat{x_k}^-)$

更新误差协方差公式：$\widehat{p_k}^+ = (I - K_k C)\widehat{p_k}^-$

以上，就是离散卡尔曼滤波算法的五个公式。

4. 实例

在这一章节中，将通过三个实例来了解卡尔曼滤波算法的应用。

4.1 实例一

对于第一个例子，我们先考虑一个简单的场景：长度的测量。假设你使用的是分度值为 1 毫米(mm)的厘米(cm)刻度尺，而你需要测量的物体的实际长度为 0.01234 米(m)，那么，如何利用卡尔曼滤波算法来帮助你估计这个物体的真实长度值？

设物体的长度值为x_k，则很容易得到描述这个系统的状态空间表达式，即

$$\begin{cases} x_k = x_{k-1} + w_{k-1} \\ y_{k-1} = x_{k-1} + v_{k-1} \end{cases} \tag{8}$$

其中，假设初始状态$x_0 \sim \mathcal{N}(0,1)$，过程噪声$w_{k-1} \sim \mathcal{N}(0, 10^{-10})$（方差设置得越小，说明对模型的信任程度越高），测量噪声$v_{k-1} \sim \mathcal{N}(0, 10^{-6})$（误差是±1mm，即$\pm 10^{-3}$m，也就是说标准差最大就是$10^{-3}$）。

接着，套用卡尔曼滤波算法的五个公式，即

1） 预测部分：

先验估计公式：$\widehat{x_k}^- = \widehat{x_{k-1}}^+$

先验误差协方差公式：$\widehat{p_k}^- = \widehat{p_{k-1}}^+ + Q_{k-1}$

2） 校正部分：

卡尔曼增益公式：$K_k = \widehat{p_k}^- (\widehat{p_k}^- + R_k)^{-1}$

后验估计公式：$\widehat{x_k}^+ = \widehat{x_k}^- + K_k(y_k - \widehat{x_k}^-)$

更新误差协方差公式：$\widehat{p_k}^+ = (I - K_k)\widehat{p_k}^-$

其中，$Q_k = 10^{-10}, R_k = 10^{-6}, \widehat{x_0}^+ = 0, \widehat{p_0}^+ = 1$。

最后，借助 Python 编程语言对上述递推过程进行程序化，如下所示：

```
import numpy as np
import matplotlib.pyplot as plt

n = 100  # 迭代次数
x = 0.01234  # 真实值
Q = 1e-10
R = 1e-3 ** 2
x_hat = np.zeros(n)
p = np.zeros(n)
x_hat_minus = np.zeros(n)
p_minus = np.zeros(n)
K = np.zeros(n)
y = np.random.normal(x, 1e-3, size=n)

#  初始化
x_hat[0] = 0.0
p[0] = 1.0

#  卡尔曼滤波
for k in range(1, n):
    #  预测部分
    x_hat_minus[k] = x_hat[k - 1]
    p_minus[k] = p[k - 1] + Q
    #  校正部分
    K[k] = p_minus[k] / (p_minus[k] + R)
    x_hat[k] = x_hat_minus[k] + K[k] * (y[k] - x_hat_minus[k])
    p[k] = (1 - K[k]) * p_minus[k]

#  绘制图像
plt.rcParams['figure.figsize'] = (7, 4)
plt.figure()
plt.plot(y, 'r+', label='noisy measurements')
plt.plot(x_hat, 'g-', label='a posteri estimate')
plt.axhline(x, color='b', label='truth value')
plt.legend()
plt.xlabel('Iteration')
plt.ylabel('Voltage')
plt.show()
```

运行后的结果如下图所示：

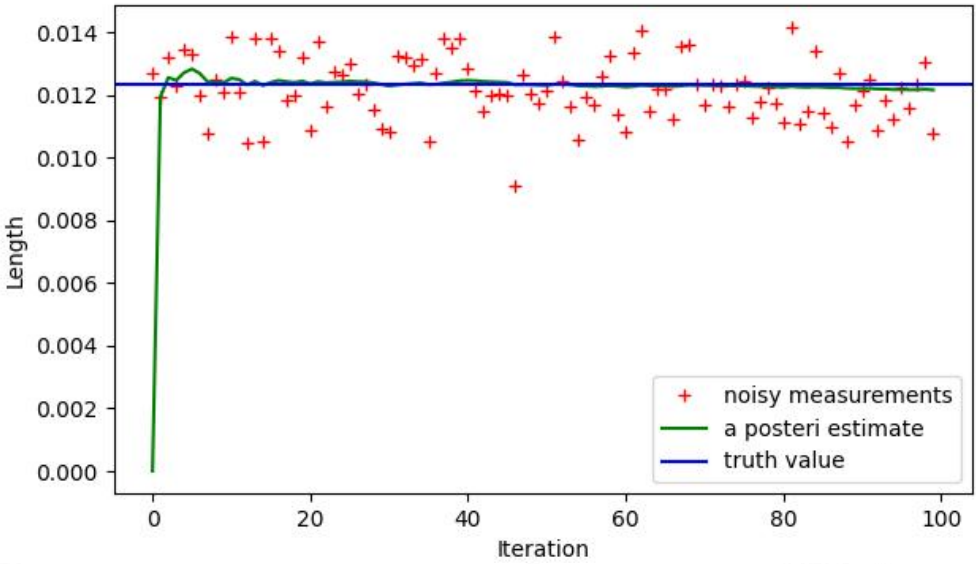

图 4-1 Python 程序的运行结果

可以看到，长度的估计值从一开始误差很大的初始值(0m)经过 100 次迭代以后逐渐逼近真实值(0.01234m)。

4.2 实例二

众所周知，单摆系统（倒立摆系统）是控制理论中的典型研究对象。所以在第二个例子中，我们考虑这样的一个场景：假设你仅有一个角位移传感器，对于小角度自由摆动的单摆，如何利用卡尔曼滤波算法来帮助你估计单摆摆动过程中的角度和角速度？

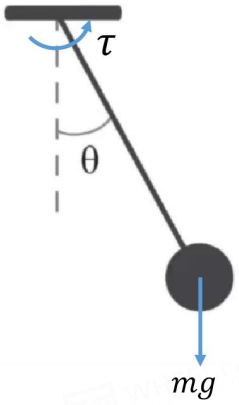

图 4-2 单摆系统

由刚体定轴转动定律可知：

$$\tau - mgl\sin\theta = I\frac{\mathrm{d}^2\theta}{\mathrm{d}t} \tag{9}$$

由于考虑的单摆是小角度自由摆动的，所以，令力矩$\tau=0$并用θ近似$\sin\theta$（线性化），则式(9)变成

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t}+\frac{mgl}{I}\theta=0 \tag{10}$$

则系统的状态空间表达式为

$$\begin{cases}\begin{bmatrix}\dot{\theta}\\ \ddot{\theta}\end{bmatrix}=\begin{bmatrix}0 & 1\\ -\dfrac{mgl}{I} & 0\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\end{bmatrix}\\ y=\begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\theta\\ \dot{\theta}\end{bmatrix}\end{cases} \tag{11}$$

设单摆的状态变量为$x_k=[\theta_k \quad \dot{\theta}_k]^{\mathrm{T}}$，系统离散化后的状态空间表达式（带噪声项），即

$$\begin{cases}x_k=A_{\mathrm{d}}x_{k-1}+w_{k-1}\\ y_{k-1}=C_{\mathrm{d}}x_{k-1}+v_{k-1}\end{cases} \tag{12}$$

假设初始状态$x_0\sim\mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\right)$，过程噪声$w_{k-1}\sim\mathcal{N}\left(\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}10^{-4} & 0\\0 & 10^{-4}\end{bmatrix}\right)$，测量噪声$v_{k-1}\sim\mathcal{N}(0,10^{-2})$。

接着，套用卡尔曼滤波算法的五个公式，即

1） 预测部分：

先验估计公式：$\widehat{x_k}^{-}=A_{\mathrm{d}}\widehat{x_{k-1}}^{+}$

先验误差协方差公式：$\widehat{p_k}^{-}=A_{\mathrm{d}}\widehat{p_{k-1}}^{+}A_{\mathrm{d}}^{\mathrm{T}}+Q_{k-1}$

2） 校正部分：

卡尔曼增益公式：$K_k=\widehat{p_k}^{-}C_{\mathrm{d}}^{\mathrm{T}}\left(C_{\mathrm{d}}\widehat{p_k}^{-}C_{\mathrm{d}}^{\mathrm{T}}+R_k\right)^{-1}$

后验估计公式：$\widehat{x_k}^{+}=\widehat{x_k}^{-}+K_k(y_k-C_{\mathrm{d}}\widehat{x_k}^{-})$

更新误差协方差公式：$\widehat{p_k}^{+}=(I-K_kC_{\mathrm{d}})\widehat{p_k}^{-}$

其中，$Q_k=\begin{bmatrix}10^{-4} & 0\\0 & 10^{-4}\end{bmatrix}$，$R_k=10^{-2}$，$\widehat{x_0}^{+}=\begin{bmatrix}0\\0\end{bmatrix}$，$\widehat{p_0}^{+}=\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$。

最后，借助 MATLAB 平台对上述递推过程进行程序化（程序详情见<实例2>文件夹），结果如下图所示：

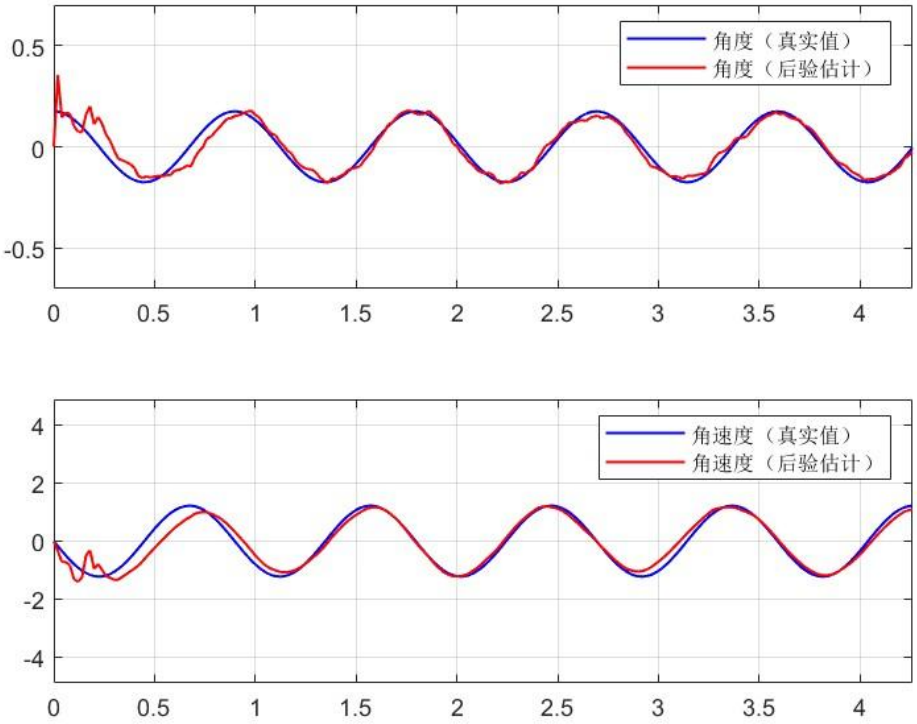

图 4-3 MATLAB 程序的运行结果

可以看到，随时间(单位：s)变化的角度(单位：rad)和角速度(单位：rad/s)的估计值经过迭代以后逐渐逼近真实值（其中，角速度无法直接测量）。

4.3 实例三

最后一个例子，考虑对平衡小车中常用的传感器芯片 MPU-6050（如图 4-4 所示）的三轴角速度和三轴加速度进行数据融合来得到车身倾角信息（这里以滚动角 ROLL 为例）。

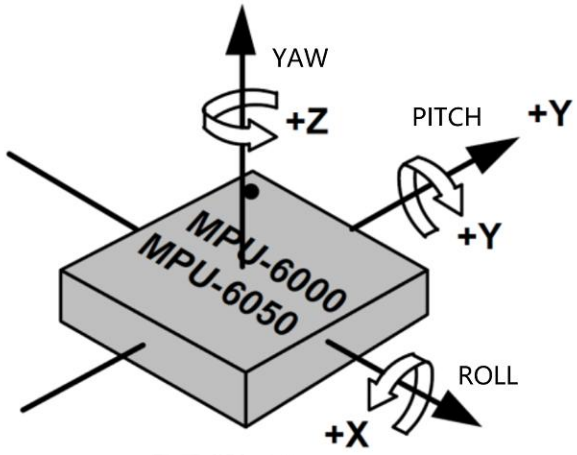

图 4-4 MPU-6050

我们选取车身倾角θ_k和陀螺仪常值偏差b_k作为状态变量，角速度$\dot{\theta}_k$为输入变量，则系统的状态空间表达式（带噪声项）可以表示成

$$\begin{cases}\begin{bmatrix}\theta_k\\ b_k\end{bmatrix}=\begin{bmatrix}1 & -\Delta t\\ 0 & 1\end{bmatrix}\begin{bmatrix}\theta_{k-1}\\ b_{k-1}\end{bmatrix}+\begin{bmatrix}\Delta t\\ 0\end{bmatrix}\dot{\theta}_{k-1}+w_{k-1}\\ y_{k-1}=\begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\theta_{k-1}\\ b_{k-1}\end{bmatrix}+v_{k-1}\end{cases} \tag{13}$$

其中，Δt是采样间隔，y_{k-1}是带有测量噪声的角度观测值（实际上是加速度的观测值，利用加速度估算角度，详情可以参考网络上相关的文章），其计算公式为

$$y_{k-1}=\operatorname{atan2}\left(-\frac{a_Y}{a_Z}\right) \tag{14}$$

假设初始状态$\begin{bmatrix}\theta_0\\ b_0\end{bmatrix}\sim\mathcal{N}\left(\begin{bmatrix}0\\ 0\end{bmatrix},\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\right)$，过程噪声$w_{k-1}\sim\mathcal{N}\left(\begin{bmatrix}0\\ 0\end{bmatrix},\begin{bmatrix}10^{-10} & 0\\ 0 & 10^{-10}\end{bmatrix}\right)$，测量噪声$v_{k-1}\sim\mathcal{N}(0,10^{-4})$。

和上面两个实例类似，套用卡尔曼滤波算法的五个公式，最后，借助MATLAB 平台对递推过程进行程序化（程序详情见<实例 3>文件夹），结果如下图所示：

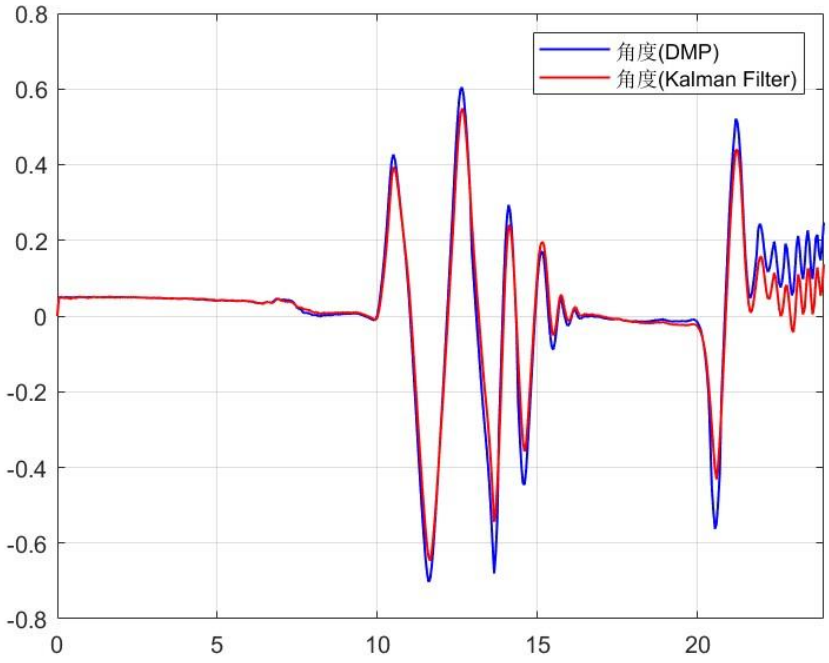

图 4-5 MATLAB 程序的运行结果

这个程序所用到的数据是事先利用上位机采集下来的，其目的是在进行 C 语言代码编写之前做一个数值上的模拟，以验证算法的有效性。DMP 是 InvenSense

公司提供的数字运动处理器，可以直接输出四元数，避免了繁琐的滤波和数据融合。为此，以 DMP 的姿态解算结果作为参考，可以很好的验证卡尔曼滤波算法的有效性。从图上可以看出，角度的估计值很好的跟随了 DMP 的姿态解算结果。

在这之后，编写相应的 C 语言代码（函数），如下所示：

```
// 头文件
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
// 函数声明
float KF(float Acce_Y, float Acce_Z, float Gyro_X);
void mul(int A_row, int A_col, int B_row, int B_col, float A[][A_col], float
B[][B_col], float C[][B_col]);
/**************************************************************************
Function: 卡尔曼滤波
Input   : 角速度，加速度
Output  : 无
**************************************************************************/
float KF(float acce_Y, float acce_Z, float gyro_X) // 输入量：Y 轴加速度，Z 轴加速度，
X 轴角速度。
{
    static float x_hat[2][1] = {0}; // 后验估计
    static float x_hat_minus[2][1] = {0}; // 先验估计
    static float p_hat[2][2] = {{1, 0}, {0, 1}}; // 后验误差协方差矩阵
    static float p_hat_minus[2][2] = {0}; // 先验误差协方差矩阵
    static float K[2][1] = {0}; // 卡尔曼增益
    const float Ts = 0.005; // 采样间隔(5ms)
    const float I[2][2] = {{1, 0}, {0, 1}};
    float u[1][1] = {{gyro_X}};
    float A[2][2] = {{1, -Ts}, {0, 1}}; // A 矩阵
    float B[2][1] = {{Ts}, {0}}; // B 矩阵
    float C[1][2] = {{1, 0}}; // C 矩阵
    float Q[2][2] = {{1e-10, 0}, {0, 1e-10}}; // 过程噪声
    float R[1][1] = {{1e-4}}; // 测量噪声
    float A_T[2][2] = {{1, 0}, {-Ts, 1}}; // A 矩阵的转置
    float C_T[2][1] = {{1}, {0}}; // C 矩阵的转置
    float temp_1[2][1] = {0}; // 用以存储中间计算结果
    float temp_2[2][1] = {0}; // 用以存储中间计算结果
    float temp_3[2][2] = {0}; // 用以存储中间计算结果
    float temp_4[2][2] = {0}; // 用以存储中间计算结果
    float temp_5[1][2] = {0}; // 用以存储中间计算结果
    float temp_6[1][1] = {0}; // 用以存储中间计算结果
    float y = atan2(-acce_Y, acce_Z); // 利用加速度计算角度
```

```
    // 预测部分
    // 先验估计公式
    mul(2, 2, 2, 1, A, x_hat, temp_1);
    mul(2, 1, 1, 1, B, u, temp_2);
    x_hat_minus[0][0] = temp_1[0][0] + temp_2[0][0];
    x_hat_minus[1][0] = temp_1[1][0] + temp_2[1][0];
    // 先验误差协方差公式
    mul(2, 2, 2, 2, A, p_hat, temp_3);
    mul(2, 2, 2, 2, temp_3, A_T, temp_4);
    p_hat_minus[0][0] = temp_4[0][0] + Q[0][0];
    p_hat_minus[0][1] = temp_4[0][1] + Q[0][1];
    p_hat_minus[1][0] = temp_4[1][0] + Q[1][0];
    p_hat_minus[1][1] = temp_4[1][1] + Q[1][1];
    // 校正部分
    // 卡尔曼增益公式
    mul(1, 2, 2, 2, C, p_hat_minus, temp_5);
    mul(1, 2, 2, 1, temp_5, C_T, temp_6);
    temp_6[0][0] = 1.0f / (temp_6[0][0] + R[0][0]);
    mul(2, 2, 2, 1, p_hat_minus, C_T, temp_1);
    mul(2, 1, 1, 1, temp_1, temp_6, K);
    // 后验估计公式
    mul(1, 2, 2, 1, C, x_hat_minus, temp_6);
    temp_6[0][0] = y - temp_6[0][0];
    mul(2, 1, 1, 1, K, temp_6, temp_1);
    x_hat[0][0] = x_hat_minus[0][0] + temp_1[0][0];
    x_hat[1][0] = x_hat_minus[1][0] + temp_1[1][0];
    // 更新误差协方差公式
    mul(2, 1, 1, 2, K, C, temp_3);
    temp_3[0][0] = I[0][0] - temp_3[0][0];
    temp_3[0][1] = I[0][1] - temp_3[0][1];
    temp_3[1][0] = I[1][0] - temp_3[1][0];
    temp_3[1][1] = I[1][1] - temp_3[1][1];
    mul(2, 2, 2, 2, temp_3, p_hat_minus, p_hat);
    // 返回值
    return x_hat[0][0];
}
/**************************************************************************
Function: 矩阵乘法
Input   : 需要相乘的两个矩阵以及它们的尺寸
Output  : 相乘后的矩阵
**************************************************************************/
void mul(int A_row, int A_col, int B_row, int B_col, float A[][A_col], float
B[][B_col], float C[][B_col])
{
```

```
    if (A_col == B_row)
    {
        for (int i = 0; i < A_row; i++)
        {
            for (int j = 0; j < B_col; j++)
            {
                C[i][j] = 0; // 初始化
                for (int k = 0; k < A_col; k++)
                {
                    C[i][j] += A[i][k]*B[k][j];
                }
            }
        }
    }
    else
    {
        printf("错误：矩阵的尺寸不对！");
    }
}
```

5. 参考文献

[1] Kalman, R. E. . "A New Approach To Linear Filtering and Prediction Problems." Journal of Basic Engineering 82D(1960):35-45.

[2] Faragher, R . "Understanding the Basis of the Kalman Filter Via a Simple and Intuitive Derivation [Lecture Notes]." IEEE Signal Processing Magazine 29.5(2012):128-132.