# 卡尔曼滤波器

version 1.0.0 author dungloi

## 理论

卡尔曼滤波基于线性、高斯系统的假设，能够利用前一时刻的状态（和可能的测量值）来得到当前时刻下的状态的最优估计。算法由预测、更新（校正）两步骤组成，既能够基于状态空间模型进行状态预测，本质上又是一个数据融合算法。

### 概述

以状态空间的形式描述一个线性高斯系统，下标表示时刻：

$$ \begin{align} x_k&=Ax_{k-1}+Bu_{k}+w_{k-1} \tag{1} \ z_k&= Hx_k + v_k \tag{2} \end{align} $$

其中 $A, B$ 分别表示状态转移矩阵和控制量输入矩阵，$H$ 表示状态观测矩阵； $x_k$ 为系统状态量矩阵，$z_k$ 为实测得到的状态量矩阵； 引入高斯分布，$w_{k-1}$ 表示过程噪声（理论模型与现实模型的误差），$v_k$ 表示观测噪声（量测与现实数据的误差），分别满足 $P(w) \sim N(0,Q),P(v)\sim N(0, R)$，其中 $Q$ 表示过程噪声协方差， $R$ 表示观测噪声协方差。

为区分状态的多种表示，定义以下符号：

$$ \begin{align} x_k:& 状态观测值\ \hat x_{k\ measure}:& 由观测初步得到的状态估计值\ \hat x_k:& 最优状态估计值，或称后验状态估计值\ \hat x_k^-:& 状态预测值，或称先验状态估计值 \end{align} $$

---

现实中我们并不知道噪声的实际值，但我们可以通过迭代优化，逼近最优的状态估计。由 $(1)$ 式，将状态预测方程表示为：

$$ \hat x_k^- = A\hat x_{k-1}+Bu_k \tag{a} $$

由 $(2)$ 式，由当下的状态量观测初步得到一个状态估计：

$$ \hat x_{k\ measure}=H^{-1}z_k $$

由于噪声的存在，$\hat x_k^-$ 和 $\hat x_{k\ measure}$ 都不能准确表示当下的状态，我们采用一个朴素的思想，即对这两者进行加权平均。状态最优估计方程：

$$ \hat x_k = \hat x_{k}^-+G(\hat x_{k\ measure}-\hat x_{k}^-) $$

其中系数 $G = 0$ 时，完全相信状态预测；系数 $G = 1$ 时，完全相信状态观测。记 $G = KH$，上式化为：

$$ \hat x_k = \hat x_{k}^-+K(z_k-H\hat x_{k}^-) \tag {b} $$

此处 $K\in [0, H^{-1}]$ 即为卡尔曼增益，求取状态最优估计的问题转化为求取最优 $K$ 的问题。

### 卡尔曼增益

我们希望最优估计与实际值误差最小，即后验估计误差 $e_k = x_k-\hat x_k$ 绝对值最小。引入后验估计误差的协方差矩阵 $P_k = E[e_ke_k^T]$，对 $n$ 个状态变量相互独立，有：

# $$ P= \begin{bmatrix} {\rm cov}(e_1,e_1)& ... &{\rm cov}(e_1,e_n)\ ... & ... & ...\ {\rm cov}(e_n,e_1)& ... &{\rm cov}(e_n,e_n)\ \end{bmatrix}

\begin{bmatrix} \sigma_1e_1^2& 0&... &0\ 0 & \sigma_2e_2^2 & ... &0\ ...&...&...&...&\ 0& 0 &... &\sigma_ne_n^2\ \end{bmatrix} $$

卡尔曼增益最优问题可转化为求 $K_k$ 使得 ${\rm tr} (P_k)$ 最小，思路为对 ${\rm tr}(P_k)$ 求导并令结果为 $0$ ，进而用其他量表示 $K_k$。以下进行求解。

由 $(2)(b)$ 式，有：

$$ \begin{align} \hat x_k &= \hat x_{k}^-+K(Hx_k + v_k-H\hat x_{k}^-)\ \Rightarrow -\hat x_k &= -\hat x_k^{-}-KH(x_k - \hat x_k^{-})-Kv_k\ \Rightarrow x_k - \hat x_k &= x_k-\hat x_k^{-}-KH(x_k - \hat x_k^{-})-Kv_k\ \Rightarrow e_k &= (I-KH)e_k^--Kv_k \end{align} $$

经计算得到：

$$ P_k=E[e_ke_k^T]=P_k^--K_kHP_k^--P_k^-H^TK_k^T+K_k(HP_k^-H^T+R)K_k^T \tag 3 $$

${\rm tr} (P_k)$ 对 $K_k$ 求导，经计算得到：

$$ {\partial {\rm tr} (P_k) \over \partial K_k}=-2P_k^{-T}H^T+2KHP_k^-H^T+2KR \triangleq 0 $$

则卡尔曼增益为：

$$ K_k={{P_k^-H^T}\over {HP_k^-H^T+R}} \tag{c} $$

观察上式可知，观测噪声协方差 $R$ 越小，增益越大，越相信状态观测值；$R$ 越大，越相信状态预测值。

## 卡尔曼滤波

先验估计误差 $e_k^-=x_k-\hat x_k^-$ ，其协方差矩阵 $P^−_k=E[{e^−_k}{e^{−T}_k}]$。由 $(1)(a)$ 式,，有：

$$ \begin{align} e_k^-&=Ax_{k-1}+Bu_{k}+w_{k-1} -A\hat x_{k-1}-Bu_k \ &=Ae_{k-1}+w_{k-1} \end{align} $$

由 $(c)$ 式，为得到 $K_k$，计算先验估计协方差 $P_k^-$ ，并化简得到：

$$ \begin{align} P_k^-&=E[(Ae_{k-1}+w_{k-1})(Ae_{k-1}+w_{k-1})^T]\ &=AE[e_{k-1}e_{k-1}^T]A^T+E[w_{k-1}w_{k-1}^T]\ &=AP_{k-1}A^T+Q \tag d \end{align} $$

需要上一时刻的后验协方差 $P_{k-1}$. 进一步联立 $(3)(c)$ 式，计算 $P_{k}$：

$$ \begin{align} P_k=&P_k^--K_kHP_k^--P_k^-H^TK_k^T+{{P_k^-H^T}\over {HP_k^-H^T+R}}(HP_k^-H^T+R)K_k^T \ &=(I-K_kH)P_k^- \tag e \end{align} $$

此时 $(a)\sim(e)$ 五式形成了迭代预测、更新的关系。此五式即为完整的卡尔曼滤波算法：

$$ \begin{split} 预测： & \hat x_k^- = A\hat x_{k-1}+Bu_k \ & P_k^-=AP_{k-1}A^T+Q\ 更新： &K_k={{P_k^-H^T}\over {HP_k^-H^T+R}}\ &\hat x_k = \hat x_{k}^-+K_k(z_k-H\hat x_{k}^-)\ &P_k=(I-K_kH)P_k^- \end{split} $$

其中，预测部分将后验状态估计和后验协方差从 $k-1$ 时刻推向 $k$ 时刻，形成预测；更新部分首先计算卡尔曼增益，根据该增益，参考状态预测值和观测值，得到状态后验估计，然后更新后验协方差。该过程不断循环；初始状态下，需要给出 $\hat x_0$ 和 $P_0$。

### 观测数据融合

由于卡尔曼滤波的本质是数据融合，在没有状态空间模型的情况下，若有两个观测器对同一个状态进行观测，产生两组数据，则它们也可以通过卡尔曼滤波的思想进行融合。此时，$A = I, B = 0, H = I$，算法简化为：

$$ \begin{split} 预测：&\hat x_{k}^-={\rm Data1}\ &P_k^-=P_{k-1}+Q\ 更新：&K_k=P_{k|k-1}(P_{k|k-1}+R)^{-1}\ &\hat x_{k}=\hat x_{k|k-1}+K_k ({\rm Data2}-\hat x_k^-)\ &P_k=(I-K_k)P_{k}^- \end{split} $$

### 参数调整

$A,B$ 及 $H$ 基于状态空间模型设置。

实际实现时，观测噪声协方差 $R$ 一般可以观测到；但过程噪声协方差 $Q$ 较难确定。根据上文，调参的一个基本原则是 $R$ 越小，越相信状态观测值；$Q$ 越小，越相信状态预测值。一般给 $Q$ 一个较小的值有助于更方便地调整 $R$。

调整后验估计误差协方差初值 $P_0$ 时，一般给一个较小的初值有利于更快收敛。运行时，若参数逐步收敛并保持为常量，则滤波器系数可以参考此结果进行调整，以期在后续在线运行时获得更好的收敛效率。

此外，可以在运行时根据模型的实际情况改变 $Q,R$ 的值。

## 快速开始

组件源码仓库地址：https://github.com/ZJU-HelloWorld/HW-Components

要在项目中使用该组件，需添加仓库内的以下文件：

```
algorithms/filter.c
algorithms/filter.h
tools.h
system.h
```

### 使用前准备

本组件涉及 CMSIS-DSP 矩阵运算等操作。

使用本组件前需要做以下准备：

- 添加源文件，包含头文件路径；注意 DSP 版本须在 1.10.0 及以上
- 添加预处理宏以开启浮点运算单元（FPU）
- 在使用 STM32CubeMX 生成项目时，请在 Code Generator 界面 Enable Full Assert，来帮助断言设备驱动中的错误；在 `main.c` 中修改 `assert_failed` 函数以指示断言结果，如添加 `while(1);`
- 在 `system.h` 中 `system options: user config` 处进行系统设置

### 示例

在项目中引用头文件：

```
#include "filter.h"
```

实例化一个卡尔曼滤波器：

=== "Kalman Filter" `c Kf_t kf;` === "Extend Kalman Filter (TODO)" `c Ekf_t ekf;`

指定状态维度、滤波器类型和各矩阵初值，初始化卡尔曼滤波器。

注意以下几点：

- 若有状态空间模型，需要 $\hat x_0,P_0,Q,R,A,B,H$ 七组矩阵数据；若为 2 传感器数据融合，需要 $\hat x_0,P_0,Q,R$ 四组矩阵数据。
- 注意矩阵维度，注意协方差矩阵均为对角阵
- 矩阵数据的排列顺序：从左至右，从上到下。

=== "KF_MODEL_BASED" `c float x[2] = {0, 1}; float p[4] = {1, 0, 0, 1}; float q[4] = {1.00, 0, 0, 1.00}; float r[4] = {0.1, 0, 0, 0.1}; float a[4] = {1, 1, 0, 1}; float b[4] = {0, 0, 0, 0}; float h[4] = {1, 0, 0, 1}; KfInit(&kf, 2, KF_MODEL_BASED, x, p, q, r, a, b, h);`

=== "KF_SENSOR_FUSION" `c float x[2] = {0, 1}; float p[4] = {1, 0, 0, 1}; float q[4] = {1.00, 0, 0, 1.00}; float r[4] = {0.1, 0, 0, 0.1}; KfInit(&kf, 2, KF_SENSOR_FUSION, x, p, q, r);`

对于 KF_MODEL_BASED 模式，调用控制量设定方法以及预测方法，获得状态量预测值；对于 KF_SENSOR_FUSION 模式，传入第一个传感器观测状态量的数组指针，得到初步状态估计值。如：

=== "KF_MODEL_BASED" `c kf.setU(&kf, u); kf.predict(&kf); kf.getX(&kf, x_prediction);`

=== "KF_SENSOR_FUSION" `c kf.predict(&kf, data1); kf.getX(&kf, x_prediction);`

传入（另一个）传感器观测状态量数组指针，调用更新方法，可获得状态量后验估计，如：

=== "KF_MODEL_BASED" `c kf.update(&kf, z); kf.getX(&kf, x_estimation);`

=== "KF_SENSOR_FUSION" `c kf.update(&kf, data2); kf.getX(&kf, x_estimation);`

## 组件说明

### `Kf` 类

线性卡尔曼滤波器。

属性

| 名称 | 类型 | 示例值 | 描述 |
|---|---|---|---|

| 名称 | 类型 | 示例值 | 描述 |
|---|---|---|---|
| `type` | `KfType_t` | `KF_MODEL_BASED` `KF_SENSOR_FUSION` | 滤波器类型 |
| `x_dim` | `uint8_t` | `2` | 状态维度 |
| `x_x1d, P_xxd, Q_xxd, R_zzd, A_xxd, B_xxd, u_x1d, H_zxd` | `arm_matrix_instance_f32` | / | 各种参数矩阵 |

方法

| 名称 | 参数说明 | 描述 |
|---|---|---|
| `KfInit` | 按声明要求传入状态维度、滤波器类型和各矩阵初值等参数 | 按指定的类型和给定的数据，初始化一个卡尔曼滤波器 |
| `predict` | （仅 KF_SENSOR_FUSION 类型）传入另一组 `float*` 观测量数组地址 | 运行卡尔曼滤波预测部分 |
| `update` | 传入 `float*` 观测量数组地址 | 根据传入的观测量，运行卡尔曼滤波更新部分 |
| `getX` | 传入 `float*` 数组地址，用于存储状态量 | 返回当前状态量 |
| `setU` | （仅 KF_MODEL_BASED 类型）传入 `float*` u 控制量数组地址 | 设置控制量 |
| `setQ` | 传入 `float*` Q 矩阵数组地址 | 设置过程噪声协方差 Q 矩阵 |
| `setR` | 传入 `float*` R 矩阵数组地址 | 设置观测噪声协方差 R 矩阵 |

# 附录

## 版本说明

| 版本号 | 发布日期 | 说明 | 贡献者 |
|---|---|---|---|
| version 1.0.0 | 2023.4.11 | 完成卡尔曼滤波说明文档 | 薛东来 |

## 参考资料

[1] 卡尔曼滤波器介绍

[2] DR_CAN 视频 文档