# 1. 基础内容

## 1.1. 样本均值和数学期望

期望是指:$E(X)=\sum_i P_i x_i$ 或者$E(X)=\int xf(x)dx$ $f(x)$ 是$X$的 PDF. 均值是相对于统计（或观测而言），对测得的样本求平均:$u=\frac{1}{n}\sum_i x_i$。 辛钦大数定律指出，当测量次数趋于无穷大时，均值依概率收敛于期望（前提是期望存在）

## 1.2. 样本方差和随机变量的方差

随机变量的方差定义为:

1. 离散随机变量

$$Var(X)=E[(X-E(X))^2]=E(X)^2-(E[X])^2 \tag{1}$$

2. 连续随机变量:

$$\begin{gathered} Var(X)=\int_{\mathbb{R}} x^2 f(x)dx-(E[X])^2; \\ E[X]=\int_{\mathbb{R}} xf(x)dx \end{gathered} \tag{2}$$

有的地方不加区分的以$\mu$ 代替$E[X]$, 以至于无法区分两者，十分讨厌

统计中的方差是指以观测样本值 减去样本均值后的平方均值，即:

$$\sigma^2=\frac{1}{N}\sum_i(x_i-\mu)^2 \tag{3}$$

贝塞尔修正（无偏）:

$$\sigma^2=\frac{1}{N-1}\sum_i(x_i-\mu)^2 \tag{4}$$

方差描述了随机变量或者样本的离散程度

## 1.3. 正态分布

正态分布也称为高斯分布，表达形式如下:

$$f(x;\mu,\sigma^2)=\sqrt{\frac{1}{2\pi\sigma^2}}e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{5}$$

表示随机变量$x$ 符合一个以$\mu$ 为期望, $\sigma^2$ 为方差的正态分布。其 pdf 是一个钟型曲线

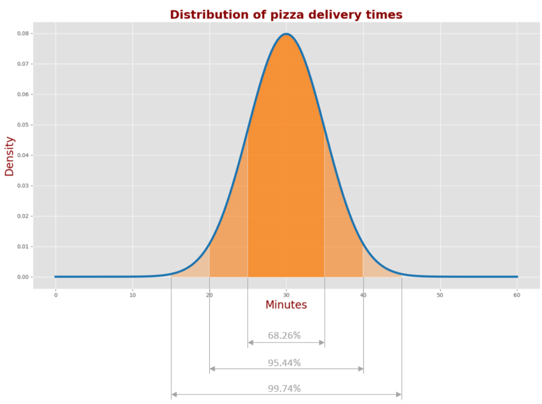


Figure 1: 正态分布示意

图中可见:

$$\begin{gathered} P(x\in(\mu-\sigma,\mu+\sigma))=68.26\% \\ P(x\in(\mu-2\sigma,\mu+2\sigma))=95.44\% \\ P(x\in(\mu-3\sigma,\mu+3\sigma))=99.74\% \end{gathered} \tag{6}$$

一般认为测量误差符合正态分布，卡尔曼滤波也假设测量误差符合正态分布

## 1.4. 估计， 准度和精度

估计是指对系统的真实（状态）情况进行估计，真实状态一般是不可见的。比如 imu 的姿态，通过传感器的测量和计算的一次过程就是对姿态的估计

准度 是指测量值（估计值）与真实值的接近情况

精度 是指测量值相对于一个真实值的分散情况

如下图所示:

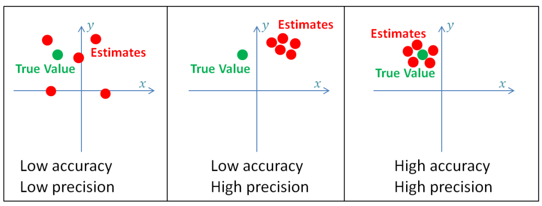


Figure 2: 准度和精度

1. 低准度、低精度：数据分散，数据的均值离真值较远
2. 低准度，高精度：数据集中，数据均值离真值较远
3. 高准度、高精度：数据集中，数据均值离真值较近

情况 2.（低准高精）通常是由系统的固定偏差引起的，无论多少次测量，其均值均与真值有个固定的偏差，这个系统称为有偏系统，比如 imu 的零偏

测量的分散性由测量误差引起，他们构成了测量系统的精度

## 2. $\alpha$ - $\beta$ 滤波器

### 2.1. 概念引入

以物体称重为例，对一个物体进行 10 次称重，在第 n 次测量的重量的估计值是前 n 次测量的平均值:

$$\hat{x}_{n,n} = \frac{1}{n}\sum_i (z_i) \tag{7}$$

其中:

| 符号 | 示意 |
|---|---|
| $x$ | 真值 |
| $z_n$ | 第 n 次的测量值 |
| $\hat{x}_{n,n}$ | 使用了$z_n$得到的$n$时刻的估计值 |
| $\hat{x}_{(n+1),n}$ | n 时刻对未来 n+1 时刻的预测值，称为外插 |

实际实现，如果按照式 7 直接实现，随着$n$ 增加，对内存和计算资源的消耗也不断增加，如果考虑服用上一次计算的值，或者 CS 中所说的"动态规划"、递推。我们对式 7 作适当变形:

$$\begin{aligned}\hat{x}_{n,n} &= \frac{1}{n}\sum z_i = \frac{1}{n}\left(\sum_{i=1}^{n-1}(z_i) + z_n\right) = \\ &= \frac{1}{n}\frac{n-1}{n-1}\sum^{n-1} z_i + \frac{1}{n}z_n = \frac{n-1}{n}\hat{x}_{n-1,n-1} + \frac{1}{n}z_n \\ &= \hat{x}_{n-1,n-1} + \frac{1}{n}(z_n - \hat{x}_{n-1,n-1})\end{aligned} \tag{8}$$

对于测量重量而言， 重量是一个静态的状态，因此外插就是直接使用当前时刻估计值作为下一时刻估计值（如果是 动态的系统，则需要经过计算得到外插值）。因此式 8 可以写作:

$$\hat{x}_{n,n} = \hat{x}(n, n-1) + \frac{1}{n}(z_n - \hat{x}_{n,n-1}) \tag{9}$$

这个方程就是一个简单的状态更新方程，对应于以下的形式:

$$\text{当前状态的估计值} = \text{当前状态的预测值} + \text{系数} \times (\text{测量值} - \text{当前状态的预测值}) \tag{10}$$

式 10 中的系数 称为卡尔曼增益，表示为$K_n$. 不同的系统，$K_n$的表示形式是不同的。 状态更新方程，是由预测值得到估计值的方程

式 10 中的测量值 – 当前状态的预测值 叫做残差，也叫做更新量.

对于式 8 而言，第一次测量需要一个初始的估计值， 比如对于一个重量为 1kg 的砝码而言，其第一次测量值是 0.995kg, 我们初始的估计值为 0，那么第一次的估计值就是:

$$\hat{x}_{1,1} = 0 + 1 \times (0.995 - 0) = 0.995 \tag{11}$$

当然，如果我们事先知道这个砝码的重量是 1kg， 那么设初始估计值为 1kg：

$$\hat{x}_{1,1} = 1 + 1 \times (0.995 - 1) = 0.995 \tag{12}$$

可见初始值可以比较粗略.

以下这段代码可以模拟一个测量和计算过程:

```python
#coding:utf-8
import matplotlib.pyplot as plt
import random
# 真实值
w = 1000
# 生成10个测量值
a = [w + random.randint(-5,5) for _ in range(20)]
# 初始估计
initial_prediction = 100

def EstimateWeight(measure, prediction, round):
  round += 1
  estimation = prediction + 1/round * (measure - prediction)
  return estimation

pred = initial_prediction
estimations = []
for i in range(len(a)):
  # 获得当前估计值
  estimation = EstimateWeight(a[i], pred, i)
  estimations.append(estimation)
  # 延迟和外插
  pred = estimation

print(f"Result:{pred}")

plt.plot(a, label=u"测量值")
plt.plot([w]*20, label=u"真实值")
plt.plot(estimations, label=u"估计值")
plt.legend()
plt.show()
```

最后得到的结果是:

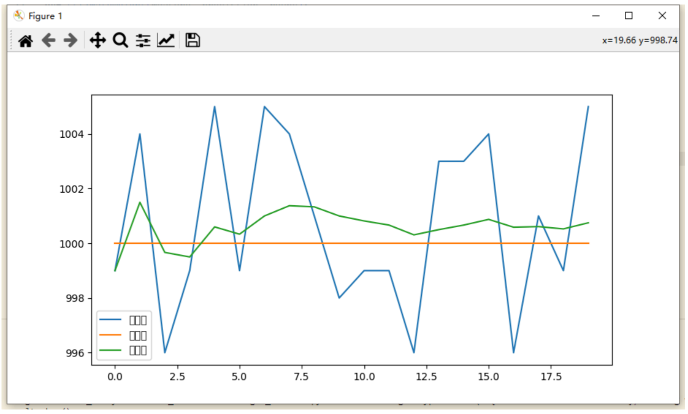

Figure 3: 质量测量模拟

图中绿色折线图是我们输出的估计值，可见随着$n$增加，估计值越来越贴近于橙色的真实值

接下来的例子是，我们用雷达测量车子的距离，$n$时刻的车子的距离是$x_n$，那么车子的速度可以用距离关于时间的导数得到:

$$x' = v \tag{13}$$

假设车子是匀速运动的，两次测量的间隔是$\Delta t$, 那么$n+1$时刻的值（理论）可以表示为:

$$\begin{aligned} x_{n+1} &= x_n + \Delta t x'; \\ x'_{n+1} &= x'_n = v \end{aligned} \tag{14}$$

式 14 称为状态外插方程或者状态转移方程、预测方程。 式 14 描述的实际上是一个一阶状态模型，概念来自于自动控制理论。 状态转移方程或预测方程 是指由前一刻的系统状态，得到当前系统状态的预测值的方程

如果式 14 使用估计值的话，那么就是：

$$\begin{aligned} \hat{x}_{n+1,n} &= \hat{x}_{n,n} + \Delta t \hat{x'}_{n,n} \\ \hat{x'}_{n+1,n} &= \hat{x'}_{n,n} \end{aligned} \tag{15}$$

式 15 得到的是由时刻$n$得到的外插值（预测）。 而时刻$n$的估计值，可以是直接使用该预测值，或者使用后文中的$\alpha$ - $\beta$ 滤波器得出。

现在假设$\Delta t$ 是 5，$x_{n-1} = 30000$, 速度为$v = 40$. 根据式 14, 我们预测$x_{n,n-1} = \hat{x}_{n-1,n-1} + \Delta t \hat{x'}_{n-1,n-1} = 30000 + 5 \cdot 40 = 30200$

同时

$$v_n = \hat{x}_{n,n-1} = \hat{x}_{n-1,n-1} = 40 \tag{16}$$

但是我们测得$z_n = 30110$, $e_n = 90$. 可能存在两个原因:
1. 雷达的测量精度较差（较为离散， 真值仍然在 30200 附近，但此次测量偏离真值较远）
2. 式 16 不成立，$v_n$ 发生了变化

我们可以写下速度的更新方程:

$$\hat{v}_{n,n} = \hat{x'}_{n,n} = \hat{x'}(n, n-1) + \beta \frac{z_n - \hat{x}_{n,n-1}}{\Delta t} \tag{17}$$

式 17 中的修正部分:$\beta\frac{z_n-\hat{x}_{n,n-1}}{\Delta t}$ 是很奇特的，$z_n - \hat{x}_{n,n-1}$ 虽然是一个距离，但是是残差，而不是与上一次估计位置的距离。 实际上$z_n$可以视作:估计值+偏差+速度偏差 × 时间，即$z_n = \hat{x}_n + \Delta v * \Delta t$. 因此$\frac{z_n-\hat{x}_{n,n-1}}{\Delta t}$ 似乎可以看做是速度的“残差”。

位置的更新，可以看做预测值和残差的加权平均:

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + \alpha(z_n - \hat{x}_{n,n-1}) \tag{18}$$

但看式 18，这个其实是一个一阶低通滤波器，系统的截止频率和$\alpha$ 有关。当$\alpha = 1$ 时， 式 18 的输出完全等价于测量输入， 当$\alpha = 0$ 时， 测量不起作用。 从概率的角度来看，前者相当于测量是完全准确的，后者相当于先验知识是完全可靠的，测量是完全不可信的。

式 17, 式 18 共同组成$\alpha$ - $\beta$ 滤波器。和测量重量不同的是，此处的这两个系数是固定值。

但是为什么$\alpha$ $\beta$ 滤波器可以跟踪系统状态就不清楚了，此处少类似于牛顿法收敛性的较为严格的数学证明

接下来是两个实例，用$\alpha$ - $\beta$ 滤波器分别追踪匀速直线运动的物体和匀加速直线运动的物体。

## 2.2. 实例 1. 测量匀速运动的物体

使用$\alpha$ - $\beta$ 滤波的算法步骤是:
1. 初始化，给定初始估计值， 选定$\alpha$, $\beta$, 并预测第一个周期的值
2. 拿到测量值，并更新当前状态（估计）
3. 做出下一周期预测

以下是代码实现:

```
import random
import matplotlib.pyplot as plt

x0 = 30000
v0 = 40
a = 0.2
b = 0.1
dt = 5
# 10 次测量值
measure = [ x0 + dt * i * (v0 + random.randint(-50, 50)/10) for i in range(1, 11)]
real = [x0 + dt * i * v0 for i in range(1, 11)]
def do_ab():
  # 初始 + 预测
  x = x0 + v0 * dt
  v = v0
  x_estimations = []
  v_estimations = []
  x_predictions = []
  bt = b/dt
  for m in measure:
    # 更新
    x = x + a * (m - x)
    x_estimations.append(x)
    v = v + bt * (m - x)
    v_estimations.append(v)
    # 预测
    x = x + v * dt
    x_predictions.append(x)
```

```
    # 预测v恒定
  return x_estimations, x_predictions, v_estimations


xe, xp, ve = do_ab()

plt.plot(xe, label="X estimation")
plt.plot(measure, label="X measure")
plt.plot(real, label="Real")
plt.legend()
plt.show()
```

下图是代码的结果:

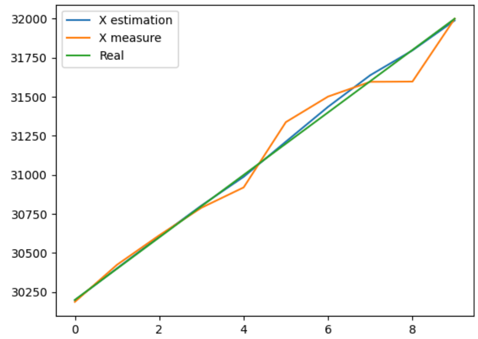


Figure 4: $\alpha$ - $\beta$ 滤波器追踪结果，$\alpha$=0.2 $\beta$=0.1

我们修改$\alpha = 0.6$ $\beta = 0.3$, 再观察结果：

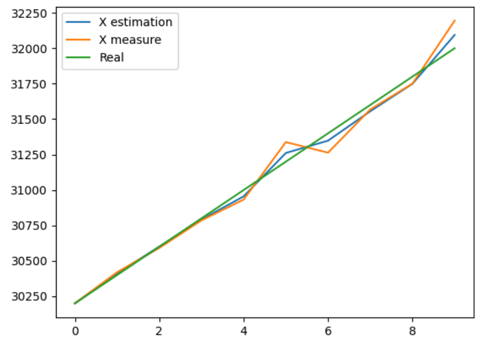


Figure 5: $\alpha$ - $\beta$ 滤波器追踪结果，$\alpha$=0.6 $\beta$=0.3

对比两张图可以发现： 较大的$\alpha$ $\beta$ 相当于更相信测量值，即认为测量系统较为准确，对输入的响应追随更快。但如果测量系统误差较大，那么可能会偏离真值较远。

较小的$\alpha$ $\beta$ 更相信预测值，对输入的响应较慢，但可以容忍更大的测量噪声。同时这意味着如果要追踪到系统真实状态可能需要更长的时间，对于某些场景而言，可能是无法接受的。

### 2.3. 实例 2. 追踪存在加速的物体

下面的代码，实现的是从 0-25s（5 个测量周期）保持匀速运动，并在后面保持匀加速运动:

```
import random
import matplotlib.pyplot as plt
x0 = 10000
v0 = 40
dt = 5
accelerate = 2

g=0.2
h=0.1

def get_distance(rnd):
  if rnd <= 5:
    return (x0 + rnd * dt *(v0))
  else:
    return (x0 + rnd * dt *v0 + 0.5 * accelerate * (((rnd - 5)* dt)**2))

real = [get_distance(i) for i in range(1,11)]
measure = [ r + random.randint(-50,50) for r in real]

def do_gh():
  # 初始预测
  x = x0 + v0 * dt
  v = v0
  xe = []
  ve = []
  nh = h/dt
  for m in measure:
    x = x + g * (m - x)
    v = v + nh * (m - x)
    # 估计
    xe.append(x)
    ve.append(v)
    # 预测
    x = x + v * dt
  return xe, ve

xe, ve = do_gh()

plt.plot(xe, label="Estimates")
plt.plot(real, label="Real")
plt.plot(measure, label="Measure")
plt.legend()
plt.show()
```

其追踪的结果:

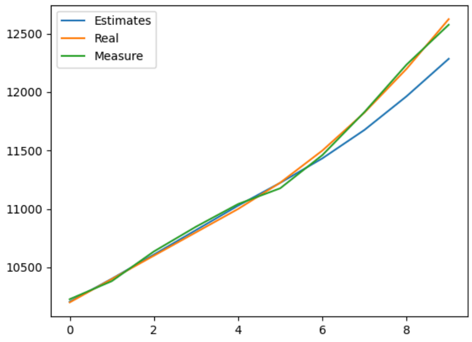


Figure 6: $\alpha$ - $\beta$ 滤波器追踪结果-匀加速-距离

我们可以看到追踪的结果，出现了滞后误差。速度的追踪图是:

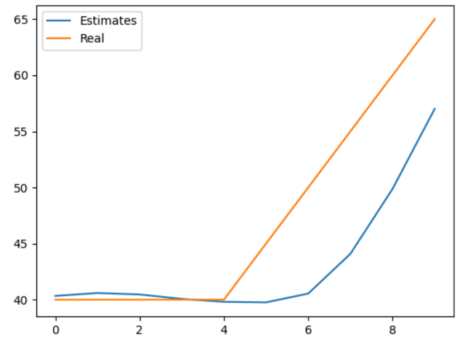


Figure 7: $\alpha$ - $\beta$ 滤波器追踪结果-匀加速-速度

可见从第 5 次估计开始，距离和速度都出现了滞后，如果后面被追踪的系统能到一个稳态，给足够的时间的话，滤波器是可以追上来的。不过实际场景中，较大的滞后意味着目标的丢失。

## 3. 单变量（一维）卡尔曼滤波器

卡尔曼滤波器和$\alpha$ - $\beta$ 滤波器一样 使用测量、更新、预测 这三个算法步骤。不同的是，卡尔曼滤波把测量值、估计值和预测值均看做符合正态分布的随机变量（无偏），这些随机变量可以用方差描述。 注意到式 5 中描述正态分布需要：均值、标准差（方差） 两个参数，因为卡尔曼假设系统无偏，因此，此处我们只需要一个方差即可描述描述这个正态分布。

在重量测量的例子中，物体真实的重量是一个不可知的、不变的常量（隐藏状态），每次测量相当于在在这个常量上叠加了一个随机的测量噪声， 这个噪声满足高斯分布。我们用$\sigma^2$ 描述，$\sigma$ 称为测量不确定性。

由于卡尔曼滤波器把 测量、估计、预测 都看做随机变量，因此式 14 不光要对状态外插，还要对随机变量的方差进行外插。

在质量测量的例子中，方差的外插公式是:

$$\hat{p}_{n+1,n} = \hat{p}_{n,n} \tag{19}$$

即质量测量这个模型中，我们预测质量测量的下一个时刻的方差等于当前时刻的方差，这是很容易理解的，因为测量手段没有变化。

在速度、位置的测量的例子中:

$$\begin{aligned} \hat{p}^v_{n+1,n} &= \hat{p}^v_{n,n} \\ \hat{p}^x_{n+1,n} &= \hat{p}^x_{n,n} + \Delta t^2 \hat{p}^v_{n,n} \end{aligned} \tag{20}$$

<式 20> 的预测方程可以理解为描述下一个时刻状态的随机变量，可以由当前状态的随机变量，加上$\Delta t \hat{p}^v_{n,n}$ 随机变量。 而$\Delta t \hat{p}^v_{n,n}$ 的方差是$\Delta t^2 \hat{p}^v_{n,n}$ . 理解这个概念十分重要。

<式 20> 也称作协方差外插方程, 协方差是相对于多维随机变量而言的。这是卡尔曼滤波器的第三个方程. 协方差外插方程适用于给出下一个时刻的协方差的预测

卡尔曼滤波器的状态更新方程写作:

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + w(z_n - \hat{x}_{n,n-1}) = wz_n + (1-w)\hat{x}_{n,n-1} \tag{21}$$

方差就可以写作:

$$p_{n,n} = w^2 r_n + (1-w)^2 p_{n,n-1} \tag{22}$$

<式 22> 中:

| 符号 | 含义 |
|---|---|
| $p_{n,n}$ | 时刻 n 的$x$的方差的估计 |
| $p_{n,n-1}$ | 时刻 n-1 的方差对时刻 n 的方差的外插，或者预测 |
| $r_n$ | n 时刻的$z_n$的方差 |

如果<式 22> 中的方差越来越小，或者取得最小值时，意味着我们以尽可能高的精度追踪到了系统的状态。接下来是经典的求导:

$$\begin{gathered} \frac{\partial(p_{n,n})}{\partial(w)} = 2wr_n - 2(1-w)p_{n,n-1} = 0 \Rightarrow \\ wr_n = (1-w)p_{n,n-1} \Rightarrow w(r_n + p_{n,n-1}) = p_{n,n-1} \Rightarrow \\ w = \frac{p_{n,n-1}}{r_n + p_{n,n-1}} \end{gathered} \tag{23}$$

把式 23 得到的结果 带入式 21 中:

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + \frac{p_{n,n-1}}{r_n + p_{n,n-1}}(z_n - \hat{x}_{n,n-1}) \tag{24}$$

对比式 24 和 式 10 中的系数， 这里的$K_n = \frac{p_{n,n-1}}{r_n+p_{n,n-1}}$ 就是卡尔曼增益。 文本化描述就是:

$$K_n = \frac{\text{预测值的方差}}{\text{测量值的方差} + \text{预测值的方差}} = \frac{p_{n,n-1}}{r_n + p_{n,n-1}} \tag{25}$$

这个是卡尔曼滤波的第四个方程，称为卡尔曼增益方程

最后，还需要更新当前状态的方差的估计。 这在式 22 中已经给出:

$$\begin{aligned} \hat{p}_{n,n} &= w^2 r_n + (1-w)^2 \hat{p}_{n,n-1} = K_n^2 r_n + (1-K_n)^2 \hat{p}_{n,n-1}; \\ 1 - K_n &= 1 - \frac{p_{n,n-1}}{r_n + p_{n,n-1}} = \frac{r_n}{r_n + p_{n,n-1}} \Rightarrow \\ p_{n,n} &= \left[\frac{p_{n,n-1}}{r_n + p_{n,n-1}}\right]^2 r_n + \left[\frac{r_n}{r_n + p_{n,n-1}}\right]^2 p_{n,n-1} \end{aligned} \tag{26}$$

为了简化，不妨以$q = p_{n,n-1}, r = r_n$, 则:

$$\begin{aligned} p_{n,n} &= \left(\frac{q}{r+q}\right)^2 r + \left(\frac{r}{r+q}\right)^2 q = \frac{q^2 r}{(r+q)^2} + \frac{q r^2}{(r+q)^2} = \frac{rq(q+r)}{(r+q)^2} \\ &= \frac{rq}{r+q}\left[\frac{q}{r+q} + \frac{r}{r+q}\right] \\ \text{又}\frac{r}{r+q} &= 1 - K_n, K_n = \frac{q}{r+q}, \Rightarrow \\ p_{n,n} &= q(1-K_n)(K_n + (1-K_n)) = (1-K_n)q = (1-K_n)p_{n,n-1} \end{aligned} \tag{27}$$

即

$$\hat{p}_{n,n} = (1 - K_n)\hat{p}_{n,n-1} \tag{28}$$

<式 28> 称作协方差更新方程， 这是卡尔曼滤波器的第五个方程。

就一维卡尔曼滤波器而言，其五个卡尔曼滤波方程式:

| 方程形式 | 方程描述 | 其他名称 |
| --- | --- | --- |

算法步骤：
1. 初始化

给定方差和状态的初始值

2. 测量:

给出测量值和测量的方差

3. 更新:

更新当前状态的估计值以及方差估计值

4. 预测:

预测下一个状态值以及其方差

### 3.1. 添加噪声

某些动态模型，如电阻的测量。电阻本身可能受环境温度影响产生轻微变动。与质量测量不同，这类模型本身的不确定性，称为过程噪声，有的也叫模型噪声。过程噪声的方差用$q$ 表示，卡尔曼滤波假设过程噪声也是符合高斯分布的。 那么，如果过程噪声不变的话，协方差外插（预测）方程为:

$$p_{n+1,n} = p_{n,n} + q_n \tag{29}$$

包含过程噪声的卡尔曼五个方程:
1. 卡尔曼增益 （权重方程）

$$K_n = \frac{p_{n,n-1}}{r_n + p_{n,n-1}} \tag{30}$$

2. 状态更新方程 (滤波方程)

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + K_{n(z_n - \hat{x}_{n,n-1})} \tag{31}$$

3. 协方差更新方程（滤波方程）

$$p_n = (1 - K_n)p_{n,n-1} \tag{32}$$

4. 状态外插（预测）方程

恒定动态（0 阶）

$$\hat{x}_{n+1,n} = \hat{x}_{n,n} \tag{33}$$

速度恒定（1 阶）

$$\begin{aligned} \hat{x'}_{n+1,n} &= \hat{x'}_{n,n} \\ \hat{x}_{n+1,n} &= \hat{x}_{n,n} + \Delta t \hat{x}'(n,n) \end{aligned} \tag{34}$$

5. 协方差外插（预测）方程

恒定动态（0 阶）

$$p_{n+1,n} = p(n,n) + q_n \tag{35}$$

速度恒定（1 阶）

$$\begin{aligned} p^v_{n+1,n} &= p^v_{n,n} + q_n \\ p^x_{n+1,n} &= p^x_{n,n} + \Delta t^2 p^v_{n,n} \end{aligned} \tag{36}$$

这里可见，过程噪声是添加在恒定项中的，为什么不添加在$p^x$ 上呢？

由 1-5 可见，状态的更新和卡尔曼增益计算是与模型无关的，但是状态外插、协方差外插是与具体的模型有关的

### 3.2. 例 使用一维卡尔曼滤波器测量物体质量

使用一维卡尔曼的步骤是：
1. 初始化和初始状态预测
2. 计算卡尔曼增益，获取测量值，更新当前值以及当前方差

3. 预测下一个状态，以及预测方差

对于一个 1kg 的物体，我们假设称的测量的方差是$r = 25$, 也就是$\sigma = 5$（这个称实际上精度已经不太好了）。初始阶段，我们手摇一下，给出一个估计值，这个估计值得方差可能特别大，此处假设为 250 ($\sigma = 50$)。 我们做 10 次测量，测量过程如下:

```
def DoKalman():
  # 初始估计
  x0 = 970
  # 初始方差
  p0 = 250
  # 生成10个测量值
  # 测量次数
  rnd = 50
  # 测量的方差
  r = 5
  randoms = np.random.normal(0, 5, rnd)
  measure = [1000 + randoms[i] for i in range(rnd)]
  # 预测第一个状态
  # 方差外插
  p = p0
  # 预测第一个周期的质量
  x = x0
  K = 0
  xe = []
  pe = []
  for m in measure:
    # 计算卡尔曼增益
    K = p /(r + p)
    # 更新估计
    x = x + K * (m - x)
    xe.append(x)
    # 方差更新
    p = (1-K) * p
    pe.append(p)
    # 预测, 质量是0阶恒定动态，因此当前估计值就是下个周期的预测
    x = x
    p = p
  plt.figure(figsize=(10,5))
  plt.subplot2grid((1,2), (0,0))
  plt.plot(xe, label="Estimates")
  plt.plot([1000]*rnd, label="Real")
  plt.legend()
  plt.subplot2grid((1,2),(0,1))
  plt.plot(pe, label="Estimates")
  plt.legend()
  plt.tight_layout()
  plt.show()
```

我们得到的结果如下：

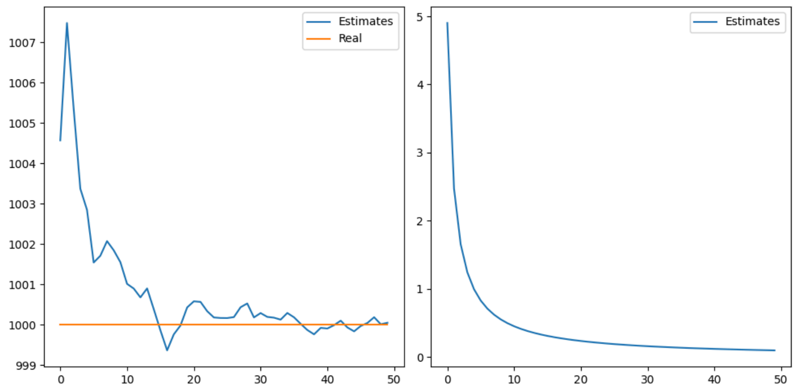


Figure 8: 卡尔曼滤波试验，重量估计（左），方差估计（右）

可见这个收敛速度是比较快的，10 次以内 方差就快速下降，后续不断减小。我们的估计值在 30 次左右，已经很贴近真实值了。 我们在对比 50 次的均值估计:

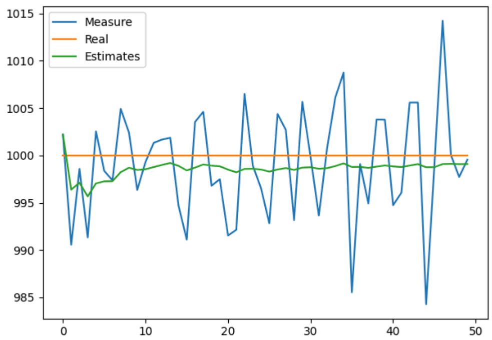


Figure 9: 均值估计

可见估计值收敛速度没有卡尔曼滤波的版本块，50 次后的测量结果的误差仍然大于 1。

## 3.3. 添加过程噪声的模型

### 3.3.1. 被测指标本身受到扰动

如电阻随温度的微小变化，测量水缸温度，但水缸温度本身会受到外界的影响产生微小的波动。这种模型本身的噪声，可以视为过程噪声。

接下来用代码实现带过程噪声的卡尔曼滤波:

```
def DoTemperatureTrace():
  # 测量50次
  rnd = 50
  # 给定初值
  t0 = 30
```

```python
# 人工估计的方差
p0 = 100
# 假设温度扰动的方差
q = 0.01
randoms = np.random.normal(0, 0.04, rnd)
# 真值
real = [33 + randoms[i] for i in range(rnd)]
# 测量误差, 假设测量精度可以到0.1
r = 0.25
me = np.random.normal(0, r, rnd)
# 加上测量误差的测量值
measure = [real[i] + me[i] for i in range(rnd)]
# s0: 由初始值，预测第一个状态
t = t0
p = p0 + q
te  =[]
pe = []
for m in measure:
  # 计算卡尔曼增益
  K = (p) / (r + p)
  # 状态估计
  t = t + K*(m - t)
  te.append(t)
  # 方差估计
  p = (1-K) * p
  pe.append(p)
  # 预测
  t = t
  p = p + q
plt.figure(figsize=(10,5))
plt.subplot2grid((1,2), (0,0))
plt.plot(real, label="Real")
plt.plot(measure, label="Measure")
plt.plot(te , label="Estimates")
plt.legend()
plt.subplot2grid((1,2),(0,1))
plt.plot(pe, label="Estimates")
plt.legend()
plt.tight_layout()
plt.show()
```

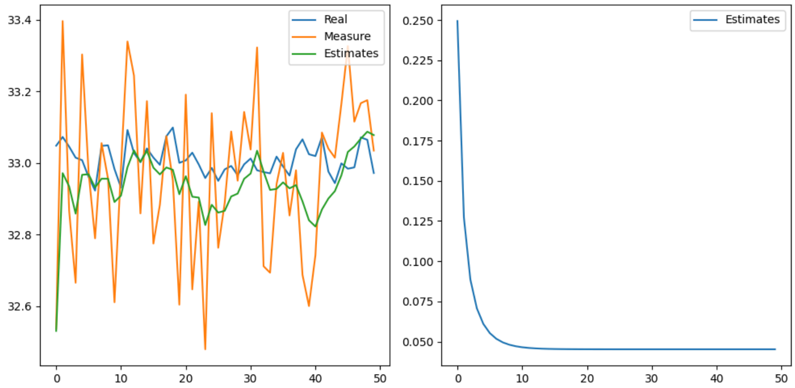


Figure 10: 添加了过程噪声的卡尔曼滤波

图中可见，真值在 33 度左右上下波动，卡尔曼估计同样也在 33 附近波动，并且变化趋势能比较好的跟随系统真值变化。右图的方差收敛也是比较快的。

### 3.3.2. 模型失准

我们仍然以上面的恒定动态模型（温度基本不变）来追踪一个持续加热的水缸的温度。相当于改变输入如下：

```python
real = [33 + 1.5*i + randoms[i] for i in range(rnd)]
```

我们得到的结果:

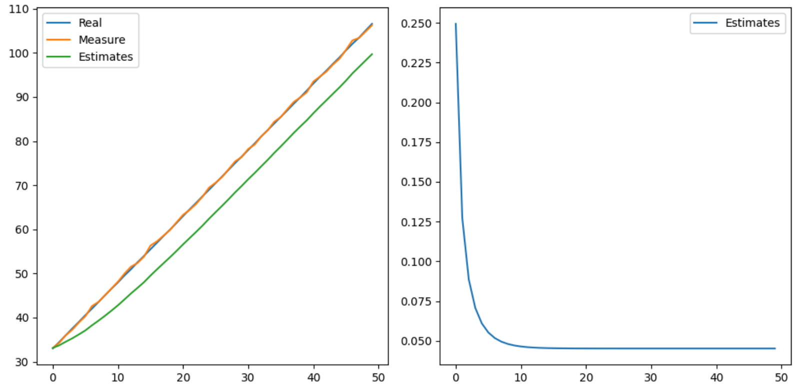


Figure 11: 模型失准

可见卡尔曼的估计值产生了滞后误差。这种滞后误差，一般是由于：

1. 对过程噪声的强度判定偏低
2. 模型失准

出现图中的估计方差很低，但是估计值明显错误的，属于一个错误设计的卡尔曼滤波器。

有些时候(什么时候?)，我们可以把模型失准，当做过程噪声，比如本例中，同样的输入，我们把过程噪声增强，得到结果如下:

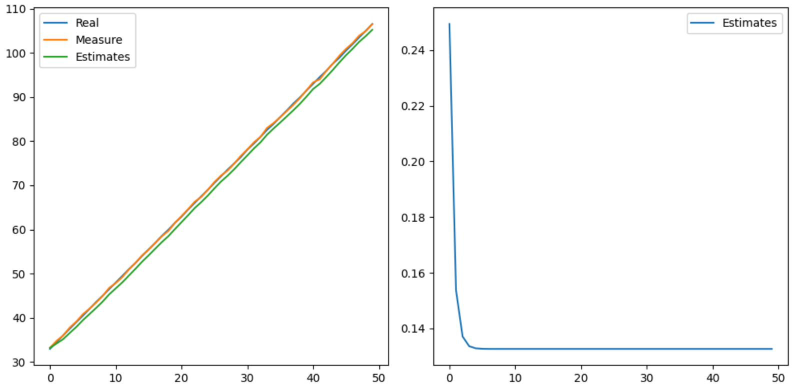


Figure 12: 模型失准, 增强过程噪声$q \to 0.15$

可见将模型失准当做过程噪声（扰动）后，增大过程噪声强度，仍然可以较好的追踪系统状态. 我们可以简单分析下卡尔曼增益和方差预测值的关系:

$$K_n = \frac{p}{r+p} \Rightarrow \frac{\partial(K_n)}{\partial(p)} = \frac{[1(r+p)-p]}{(r+p)^2} = \frac{r}{(r+p)^2} > 0 \tag{37}$$

式 37 用到了是导数商法则:$(\frac{u}{v})' = \frac{u'v-uv'}{(v)^2}$. 注意任何测量过程的方程都不可能为 0，因此式 37 始终大于 0，即$K_n$ 关于$p$ 是单调增函数。而增加$q_n$ 相当于增加每个步骤的$p$, 即调高更新的权重。

我们似乎可以做出这样的总结，过程噪声是模型的一部分，卡尔曼滤波器追踪的是叠加了过程噪声的模型值，测量值，是这个增加了过程噪声的模型值上再增加一个随机变量（测量噪声），而卡尔曼滤波器最主要要做的就是尽可能减少后者的影响。我们要注意，包含过程噪声的方差的最小估计值都是过程噪声方差。 这个结论是错误的

某些情况下，模型失准是不能用过程噪声补偿的。